



HAL
open science

Normalization Without Syntax

Willem Heijltjes, Dominic Hughes, Lutz Strassburger

► **To cite this version:**

Willem Heijltjes, Dominic Hughes, Lutz Strassburger. Normalization Without Syntax. FSCD 2022, Aug 2022, Haifa, Israel. hal-03654060

HAL Id: hal-03654060

<https://inria.hal.science/hal-03654060v1>

Submitted on 28 Apr 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

1 Normalization Without Syntax

2 **Willem B. Heijltjes**

3 Department of Computer Science, University of Bath, UK

4 **Dominic J. D. Hughes**

5 Logic Group, U.C. Berkeley, USA

6 **Lutz Straßburger**

7 Equipe Partout, Inria Saclay, France

8 — Abstract —

9 We present normalization for intuitionistic combinatorial proofs (ICPs) and relate it to the simply-
10 typed lambda-calculus. We prove confluence and strong normalization. Combinatorial proofs, or
11 “proofs without syntax”, form a graphical semantics of proof in various logics that is canonical
12 yet complexity-aware: they are a polynomial-sized representation of sequent proofs that factors
13 out exactly the non-duplicating permutations. Our approach to normalization aligns with these
14 characteristics: it is canonical (free of permutations) and generic (readily applied to other logics).
15 Our reduction mechanism is a canonical representation of reduction in sequent calculus with closed
16 cuts (no abstraction is allowed below a cut), and relates to closed reduction in lambda-calculus and
17 supercombinators. While we will use ICPs concretely, the notion of reduction is completely abstract,
18 and can be specialized to give a reduction mechanism for any representation of typed normal forms.

19 **2012 ACM Subject Classification** Theory of computation → Proof theory; Theory of computation
20 → Lambda calculus

21 **Keywords and phrases** combinatorial proofs, intuitionistic logic, lambda-calculus, Curry–Howard,
22 proof nets

23 **Digital Object Identifier** 10.4230/LIPIcs...

24 **Funding** This work was supported by EPSRC Grant EP/R029121/1 *Typed Lambda-Calculi with*
25 *Sharing and Unsharing*.

26 **1** Introduction

27 The sequent calculus was introduced by Gentzen [7] as a meta-calculus, to describe the
28 construction of proofs in natural deduction, the object-calculus. The sequent calculus has good
29 proof-theoretic properties, such as isolating the cut-rule as the distinction between normal
30 and non-normal proofs and avoiding the ad-hoc construction of open and closed assumptions.
31 However, it features many permutations, that relate different ways of constructing the same
32 natural deduction proof. This is a problem for proof normalization in particular, since
33 permutations come to dominate the cut-elimination process.

34 When Girard introduced Linear Logic [8], it was naturally expressed in sequent calculus,
35 which defined clear and natural meta-level operations for proof construction. But there was
36 no object-level calculus to which these applied, and which might capture its computational
37 content. Constructing one became the project of *proof nets* [8, 10, 16, 12], with the aim of
38 *canonicity*: proof nets aim to represent sequent proofs canonically, modulo permutations.

39 Combinatorial proofs, first developed for classical propositional logic by Hughes [14],
40 continue the tradition of proof nets with a refined aim, called *local canonicity* [15]. The issue
41 is that permutations may *duplicate* subproofs; to factor them out then generally causes an
42 exponential blowup of the representation. Figure 1 illustrates such a permutation. The idea of
43 *local canonicity* is to give a complexity-sensitive, polynomial representation of sequent proofs,
44 modulo the non-duplicating permutations. This is achieved in combinatorial proofs by a clean
45 separation of the logical content (the logical rules of a sequent proof) and the structural
46 content (the structural rules, contraction and weakening), each captured in a distinct part of a
47 combinatorial proof. Sequent calculi are generally unable to stratify proofs in this way, but it



© Author: Please provide a copyright holder;

licensed under Creative Commons License CC-BY

Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

$$\frac{\frac{\Gamma \vdash A \quad \frac{B, B, \Delta \vdash C}{B, \Delta \vdash C} c}{\Gamma, A \Rightarrow B, \Delta \vdash C} \Rightarrow L}{\Gamma \vdash A \quad \frac{B, B, \Delta \vdash C}{B, \Delta \vdash C} c} \approx \frac{\frac{\Gamma \vdash A \quad \frac{B, B, \Delta \vdash C}{B, \Gamma, A \Rightarrow B, \Delta \vdash C} \Rightarrow L}{\Gamma, A \Rightarrow B, \Gamma, A \Rightarrow B, \Delta \vdash C} \Rightarrow L}{\frac{\Gamma, A \Rightarrow B, \Gamma, A \Rightarrow B, \Delta \vdash C}{\Gamma, A \Rightarrow B, \Delta \vdash C} c} c$$

■ **Figure 1 A duplicating permutation.** Intuitionistic sequent calculus, as we will use it, has exactly one duplicating permutation, illustrated here. Permuting the contraction rule c and the implication-left rule $\Rightarrow L$ duplicates the subproof on the left. Iterating the permutation gives exponential growth. It is instructive to consider the translation to natural deduction, which unfolds along this permutation and does indeed grow exponentially.

48 is a natural form of decomposition in deep inference [26]. Beyond classical propositional logic,
 49 combinatorial proofs have been given for intuitionistic propositional logic [13], first-order
 50 classical logic [17, 18], relevance logics [2], and modal logics [3].

51 We are interested in the question: what is a natural and general notion of composition
 52 for combinatorial proofs? In this paper we consider the intuitionistic case, Intuitionistic
 53 Combinatorial Proofs (ICPs) [13], where the question is particularly pertinent due to the
 54 Curry–Howard correspondence with typed lambda-calculi.

55 Our aim has been twofold: 1) to implement sequent-calculus reduction canonically (i.e.
 56 without permutations), and 2) to ensure our notion of reduction is sufficiently abstract that it
 57 will (plausibly) generalize to combinatorial proofs more widely.

58 Our solution is a notion of composition in conjunction-implication intuitionistic logic that
 59 is canonical for sequent calculus normalization, in the sense that permutations on cuts are
 60 factored out. Reduction operates on trees of normal forms, where edges represent cuts, giving
 61 a simple and natural structure that may easily generalize to other logics. A reduction step on
 62 a given edge is determined by how the attached nodes may sequentialize, not by their internal
 63 structure. Consequently, the reduction mechanism is *abstract* in the sense that it is agnostic
 64 about the actual contents of nodes, which can be any representation of normal forms.

65 1.1 Composition

66 Composition of proofs in intuitionistic sequent calculus is by the following cut-rule, followed
 67 by cut-elimination. We would like to transport this operation to combinatorial proofs.

$$68 \frac{\Gamma \vdash A \quad A, \Delta \vdash B}{\Gamma, \Delta \vdash B} \text{cut}$$

69 We identify two prominent approaches for similar composition operations in the literature
 70 (our classification is not intended to be comprehensive, only helpful in setting out similarities):

71 **Internal rewriting.** An object-calculus may support non-normal forms and rewriting internally.

72 In the λ -calculus, composition creates a redex, which is then beta-reduced. Likewise, many
 73 notions of proof net admit an explicit notion of cut, as a node or as a *cut-link* connecting
 74 dual formulae, that is eliminated by rewriting [10, 15], giving rise to the interaction nets
 75 paradigm [23].

76 **Direct composition.** For an object calculus that admits only normal forms, composition
 77 may be computed by a single-shot operation. Examples are the Geometry of Interaction
 78 paradigm, which computes a normal form via the execution formula [9]; game semantics,
 79 which composes strategies by *interaction + hiding* [1, 21]; the evaluation of cut-nets in
 80 ludics [11]; and various notions of proof net where composition is a form of relational
 81 composition over links [16, 12, 19]. Observe that object-level proofs become an invariant
 82 for sequent-calculus cut-elimination.

83 Based on prior art, one may readily imagine what either approach would involve for ICPs.
 84 For internal rewriting, an ICP may be constructed over a sequent that includes internal
 85 cut-formulas as special antecedents $A \Rightarrow A$ (marked below by underlining), introduced by a cut
 86 as analogous to a $\Rightarrow L$ rule, and eliminated by rewriting. One may transport sequent-calculus
 87 cut-elimination to this setting by identifying sub-proofs of ICPs, via *kingdoms* [4].

$$88 \quad \Gamma, \underline{A_1 \Rightarrow A_1}, \dots, \underline{A_n \Rightarrow A_n} \vdash B$$

89 For direct composition, ICPs may be interpreted as games with *sharing* [13], for which the
 90 *interaction + hiding* approach can be explored. Both these approaches are interesting and
 91 deserve to be investigated, and we may do so in future. However, they will inevitably require
 92 some intricate combinatorics, and are not likely to generalize across combinatorial proofs.

93 Here, we describe a normalization method for ICPs that is simple, natural, and achieves
 94 both our main objectives: 1) it is effectively a permutation-free implementation of sequent
 95 calculus cut-elimination, and 2) it is sufficiently abstract that it is likely to generalize well.
 96 Technically, ICPs will form the nodes of a *combinatorial tree*, connected by edges that represent
 97 cuts. Combinatorial trees are then reduced by cut-elimination, following the reduction in
 98 sequent calculus. Interestingly, this approach fits neither of the above categories well, and
 99 instead suggests to identify a third category:

100 **External rewriting.** An object calculus without internal composition may be extended by
 101 a secondary structure, which is then evaluated by rewriting. The prime example is
 102 *supercombinators* [20, 25], where normalization takes place on a tree of normal-form λ -terms
 103 (restricted to having no abstractions inside applications).

104 There are interesting parallels between our combinatorial trees and supercombinators, which
 105 we explore in Section 7. In addition, we will connect ICP normalization to *closed reduction* in
 106 λ -calculus [6], via a novel explicit-substitution calculus.

107 2 Intuitionistic Combinatorial Proofs

108 We give a concise inductive definition of ICPs; see [13] for a full treatment including an
 109 informal introduction and a geometric definition. For the purposes of this paper, it would also
 110 be sufficient to view ICPs as sequent proofs modulo permutations.

111 We work in conjunction-implication intuitionistic logic. **Formulas** A, B, C are given by
 112 the grammar below, where P, Q are propositional atoms. A **context** Γ, Δ is a multiset of
 113 formulas and a **sequent** $\Gamma \vdash A$ is a context with a formula.

$$114 \quad A, B, C ::= P \mid A \wedge B \mid A \Rightarrow B$$

115 An ICP for a formula A will be a graph homomorphism $f: \mathcal{G} \rightarrow \llbracket A \rrbracket$ consisting of:

116 ■ an **arena** $\llbracket A \rrbracket$, a graph representing the formula A modulo the non-duplicating isomorphisms
 117 of symmetry, associativity, and currying;

$$118 \quad A \wedge B \sim B \wedge A \quad A \wedge (B \wedge C) \sim (A \wedge B) \wedge C \quad (A \wedge B) \Rightarrow C \sim A \Rightarrow (B \Rightarrow C)$$

119 ■ a **linked arena** \mathcal{G} , a proof net in IMLL (intuitionistic multiplicative linear logic) over an
 120 arena rather than a formula, to represent the *logical* rules of the sequent calculus;

121 ■ a **skew fibration** f , a graph homomorphism from \mathcal{G} to $\llbracket A \rrbracket$ representing the *structural*
 122 rules of contraction and weakening.

123 We define each component inductively. An arena will be a DAG (directed acyclic graph)
 124 $\mathcal{G} = (V_{\mathcal{G}}, \rightarrow_{\mathcal{G}})$ with vertices $V_{\mathcal{G}}$ and edges $\rightarrow_{\mathcal{G}} \subseteq V_{\mathcal{G}} \times V_{\mathcal{G}}$. We indicate the **root vertices** of
 125 \mathcal{G} (those without outgoing edges) by $R_{\mathcal{G}}$. Consider the following two operations: a **sum** of

XX:4 Normalization Without Syntax

two graphs $\mathcal{G} + \mathcal{H}$ is their disjoint union, and a **subjunction** $\mathcal{G} \triangleright \mathcal{H}$ is a disjoint union that in addition connects all the roots of \mathcal{G} to the roots of \mathcal{H} .

$$\begin{array}{l} \text{sum:} \quad \mathcal{G} + \mathcal{H} = (V_{\mathcal{G}} \uplus V_{\mathcal{H}}, \rightarrow_{\mathcal{G}} \uplus \rightarrow_{\mathcal{H}}) \\ \text{subjunction:} \quad \mathcal{G} \triangleright \mathcal{H} = (V_{\mathcal{G}} \uplus V_{\mathcal{H}}, \rightarrow_{\mathcal{G}} \uplus \rightarrow_{\mathcal{H}} \uplus (R_{\mathcal{G}} \times R_{\mathcal{H}})) \end{array}$$

► **Definition 1.** An **arena** is a graph \mathcal{G} constructed from single vertices by $(+)$ and (\triangleright) , with an L -labelling $\ell_{\mathcal{G}} : V_{\mathcal{G}} \rightarrow L$ assigning each vertex a label from a set L . The arena $\llbracket A \rrbracket$ of a formula A is given inductively by: $\llbracket P \rrbracket$ is a single vertex labelled P , and

$$\llbracket A \wedge B \rrbracket = \llbracket A \rrbracket + \llbracket B \rrbracket \quad \llbracket A \Rightarrow B \rrbracket = \llbracket A \rrbracket \triangleright \llbracket B \rrbracket .$$

Note that arenas are linear in the size of formulas, and while they factor out symmetry, associativity, and currying, they do not factor out distributivity.

$$\llbracket A \Rightarrow (B \wedge C) \rrbracket \neq \llbracket (A \Rightarrow B) \wedge (A \Rightarrow C) \rrbracket$$

An ICP will be an *arena morphism*: a map $f : \mathcal{G} \rightarrow \llbracket A \rrbracket$ given by an underlying function on vertices $f : V_{\mathcal{G}} \rightarrow V_{\llbracket A \rrbracket}$ that preserves edges, roots, and the equivalence given by labelling, i.e. if $\ell_{\mathcal{G}}(v) = \ell_{\mathcal{G}}(w)$ then $\ell_{\llbracket A \rrbracket}(f(v)) = \ell_{\llbracket A \rrbracket}(f(w))$. We will construct arena morphisms inductively, which guarantees these conditions. For $g : \mathcal{G} \rightarrow \llbracket A \rrbracket$ and $h : \mathcal{H} \rightarrow \llbracket B \rrbracket$ we have the operations

$$\begin{array}{l} \text{implication:} \quad g \triangleright h : \mathcal{G} \triangleright \mathcal{H} \rightarrow \llbracket A \rrbracket \triangleright \llbracket B \rrbracket \\ \text{sum:} \quad g + h : \mathcal{G} + \mathcal{H} \rightarrow \llbracket A \rrbracket + \llbracket B \rrbracket \\ \text{contraction:} \quad [g, h] : \mathcal{G} + \mathcal{H} \rightarrow \llbracket A \rrbracket \quad (\text{where } \llbracket A \rrbracket = \llbracket B \rrbracket) \end{array}$$

where each case is given by the union of the underlying functions on vertex sets: for implication and sum, $g \cup h : (V_{\mathcal{G}} \uplus V_{\mathcal{H}}) \rightarrow (V_{\llbracket A \rrbracket} \uplus V_{\llbracket B \rrbracket})$, and for contraction $g \cup h : (V_{\mathcal{G}} \uplus V_{\mathcal{H}}) \rightarrow V_{\llbracket A \rrbracket}$. In addition, we use the following constructions, where \emptyset is the empty graph.

$$\begin{array}{l} \text{axiom:} \quad 1_{P,Q} : \llbracket P \rrbracket \rightarrow \llbracket Q \rrbracket \\ \text{weakening:} \quad \varnothing_{\llbracket A \rrbracket} : \emptyset \rightarrow \llbracket A \rrbracket \end{array}$$

The axiom is the trivial map from one singleton arena (with vertex labelled P) to another (with vertex labelled Q). Weakening is the empty morphism. Note that because arenas are non-empty, in isolation it is not an arena morphism, but we will use it only in the context of an implication, sum, or contraction, so that this is not an issue.

We write $f :: A$ for $f : \mathcal{G} \rightarrow \llbracket A \rrbracket$. To construct ICPs from sequent proofs we use **sequents** of arena morphisms (and weakenings), that represent a single arena morphism as follows.

$$k_1 :: A_1, \dots, k_n :: A_n \vdash f :: B \iff (k_1 + \dots + k_n) \triangleright f :: (A_1 \wedge \dots \wedge A_n) \Rightarrow B$$

We refer to f and the k_i as **ports**, where k_i is an **antecedent** and f the **consequent**, and we write $\varphi :: \Gamma$ for the **context** $k_1 :: A_1, \dots, k_n :: A_n$.

► **Definition 2.** An **intuitionistic combinatorial proof (ICP)** of a formula A is an arena morphism $f :: A$ constructed by the sequent calculus of Figure 2.

Figure 3 gives examples of ICPs, with corresponding types and λ -terms (the translation will be made formal in Section 8). Figure 4 gives non-examples of ICPs.

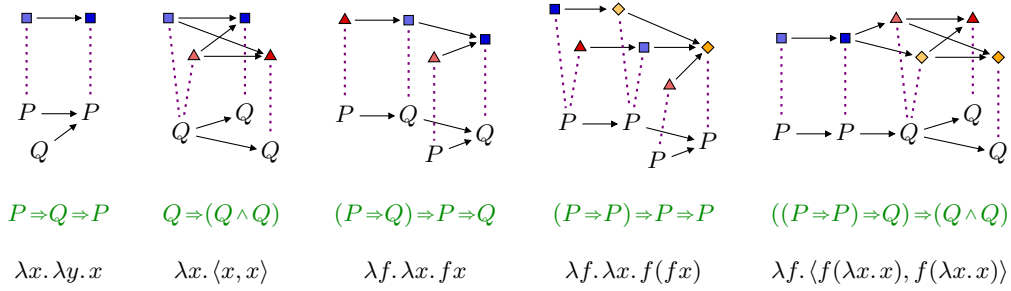
For clarity, an axiom 1 generates the ICP below.

$$1 :: P \vdash 1 :: P = \begin{array}{ccc} \blacksquare & \longrightarrow & \blacksquare \\ \vdots & & \vdots \\ P & \longrightarrow & P \end{array}$$

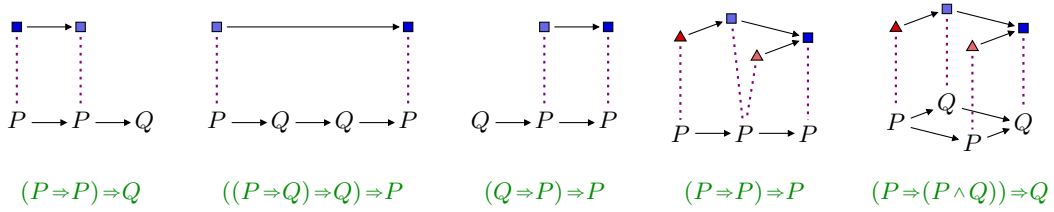
We call the subgraph $\blacksquare \rightarrow \blacksquare$ a **link**, where the side condition (\star) in Figure 2 requires that every link receives a different label $\blacksquare, \blacktriangle, \blacklozenge$, etc. Vertices are **equivalent** if they have the same label, and ICPs as arena morphisms preserve equivalence by construction.

$$\begin{array}{c}
 \frac{}{1 :: P \vdash 1 :: P} 1^* \quad \frac{\varphi :: \Gamma \vdash f :: B}{\varphi :: \Gamma, \emptyset :: A \vdash f :: B} w \quad \frac{\varphi :: \Gamma, k :: A, l :: A \vdash f :: B}{\varphi :: \Gamma, [k, l] :: A \vdash f :: B} c^\dagger \\
 \\
 \frac{\varphi :: \Gamma, k :: A, l :: B \vdash f :: C}{\varphi :: \Gamma, k+l :: A \wedge B \vdash f :: C} \wedge L \quad \frac{\varphi :: \Gamma \vdash f :: A \quad \psi :: \Delta \vdash g :: B}{\varphi :: \Gamma, \psi :: \Delta \vdash f+g :: A \wedge B} \wedge R \\
 \\
 \frac{\varphi :: \Gamma, k :: A \vdash f :: B}{\varphi :: \Gamma \vdash k \triangleright f :: A \Rightarrow B} \Rightarrow R \quad \frac{\varphi :: \Gamma \vdash f :: A \quad k :: B, \psi :: \Delta \vdash g :: C}{\varphi :: \Gamma, f \triangleright k :: A \Rightarrow B, \psi :: \Delta \vdash g :: C} \Rightarrow L^\ddagger
 \end{array}$$

■ **Figure 2** Inductive construction of ICPs. (*) Each instance of 1 is given a distinct label in the source arena. (†) For c we require $k, l \neq \emptyset$. (‡) For $\Rightarrow L$ we require $k \neq \emptyset$.



■ **Figure 3** Examples of ICPs with corresponding λ -terms. The source arena is at the top, with its labelling given by coloured shapes. The target arena is at the bottom, labelled with propositional atoms, and the arena morphism is given by dotted (purple) lines.



■ **Figure 4** Non-examples of ICPs. They cannot be constructed with the sequent calculus in Figure 2.

163 To *decompose* an ICP, the unary rules $\wedge L$, $\Rightarrow R$, c , w apply whenever the given port is
 164 of the right kind, respectively $k+l$, $k \triangleright f$, $[k, l]$, and \emptyset . The binary rules $\wedge R$, $\Rightarrow L$ apply only
 165 when the ICP can be split into two without breaking up any links in the source graph. We
 166 write $\varphi \parallel \psi$ when the sources of φ and ψ do not share any labels; then the rules $\wedge R$, $\Rightarrow L$ as
 167 given in Figure 2 apply in reverse exactly when respectively $\varphi, f \parallel \psi, g$ and $\varphi, f \parallel k, \psi, g$. We
 168 call a port **open** if the ICP can be decomposed along it, and **closed** otherwise.

169 We refer to [13] for a *geometric* definition of ICPs, where the equivalence with the *inductive*
 170 definition given here is a theorem. We recall the following from [13].

171 ▶ **Theorem 3** (Local canonicity). *Two sequent proofs construct the same ICP if and only if*
 172 *they are equivalent modulo non-duplicating formula-isomorphisms and rule permutations.*

$$\begin{array}{c}
 \frac{\Gamma \vdash A \quad A, \Delta \vdash B}{\Gamma, \Delta \vdash B} \text{cut} \\
 \text{a)}
 \end{array}
 \qquad
 \begin{array}{c}
 \varphi :: \Gamma \vdash f :: A = \frac{\varphi :: \Gamma}{f :: A} \\
 k :: A, \psi :: \Delta \vdash g :: B = \frac{k :: A \quad \psi :: \Delta}{g :: B} \\
 \text{b)}
 \end{array}
 \qquad
 \begin{array}{c}
 \frac{\varphi :: \Gamma}{f :: A} \\
 \text{---} \\
 \frac{k :: A \quad \psi :: \Delta}{g :: B} \\
 \text{c)}
 \end{array}$$

■ **Figure 5** Composition of combinatorial proofs into combinatorial trees. **a)** The sequent calculus cut-rule. **b)** Presenting ICP sequents as nodes of a tree, with antecedent ports above and consequent port below a central line. **c)** Connecting both nodes by an edge, represented by a dashed line, to form a tree.

173 3 Composition of combinatorial proofs

174 Combinatorial proofs represent normal forms: the sequent calculus for constructing them, in
 175 Figure 2, does not have a cut-rule (Figure 5a). What is expected is a notion of composition,
 176 of an ICP for $\Gamma \vdash A$ and one for $A, \Delta \vdash B$ into one for $\Gamma, \Delta \vdash B$.

177 We give a direct interpretation of composition by taking ICPs as the nodes of a tree,
 178 connected by cuts as edges; see Figure 5, where solid lines represent the nodes in the tree and
 179 the dashed lines the edges. We formalize this construction as a notion of **combinatorial**
 180 **tree**, which we will then proceed to reduce. The nature of reduction will make it desirable to
 181 have constants available.

182 ▶ **Definition 4** (Combinatorial tree). *A **combinatorial tree** $t :: C$ with **conclusion formula** C*
 183 *is an inductive tree consisting of either:*

- 184 ■ a **premiss** $\star :: C$, representing (the arena of) C , or
- 185 ■ a **constant** $c :: C$ where $C = P_1 \Rightarrow \dots \Rightarrow P_n \Rightarrow P$ ($n \geq 0$), or
- 186 ■ a **node** $k_1 :: A_1, \dots, k_n :: A_n \vdash f :: C$ with a sequence of **subtrees** $t_1 :: A_1 \dots t_n :: A_n$, written:

$$\frac{\frac{t_1 :: A_1}{k_1 :: A_1} \cdots \frac{t_n :: A_n}{k_n :: A_n}}{f :: C}$$

188 For a concrete example, Figure 7 gives a reduction featuring various combinatorial trees. We
 189 abbreviate $t :: C$ to t , and write $\tau :: \Gamma$ for a **forest** $t_1 :: A_1 \dots t_n :: A_n$ (where $\Gamma = A_1, \dots, A_n$).
 190 Edges connecting τ to antecedents $\varphi = k_1, \dots, k_n$ are drawn like a single dashed edge,
 191 rendering the above tree as (a) below. We indicate a forest of premisses by $\star :: \Gamma$, as in (b),
 192 and denote the premisses of a tree t by $\star t$. A tree **for** the sequent $\Gamma \vdash A$ is one $t :: A$ with
 193 $\star t = \Gamma$. We visually identify the premisses of a tree by a double dashed edge, as in (c) below
 194 for s with $\star s = A, \Delta$. Then (d) is the result of replacing $\star :: A$ in s by a tree t for $\Gamma \vdash A$,
 195 imitating the cut rule of Figure 5.

$$\begin{array}{c}
 \frac{\tau :: \Gamma}{\varphi :: \Gamma} \\
 \text{---} \\
 f :: C \\
 \text{(a)}
 \end{array}
 \qquad
 \begin{array}{c}
 \star :: \Gamma \\
 \text{---} \\
 \varphi :: \Gamma \\
 \text{---} \\
 f :: C \\
 \text{(b)}
 \end{array}
 \qquad
 \begin{array}{c}
 \star :: A \quad \star :: \Delta \\
 \text{-----} \\
 s :: B \\
 \text{(c)}
 \end{array}
 \qquad
 \begin{array}{c}
 \star :: \Gamma \\
 \text{-----} \\
 t :: A \quad \star :: \Delta \\
 \text{-----} \\
 s :: B \\
 \text{(d)}
 \end{array}$$

197 The reduction rules will essentially be those of sequent calculus, but now in a setting that
 198 is free of permutations. Observe that while combinatorial trees involve a good amount of
 199 notation, the notion of a tree of normal forms is in fact highly conceptual. For reduction, the
 200 particular use of ICPs is secondary, and any representation of normal forms would do, since
 201 the reduction rules are determined entirely by the *sequentialization* or *decomposition* of nodes.

202 ▶ **Definition 5** (Reduction). *Reduction of combinatorial trees is by the rules in Figure 6.*

$$\begin{array}{c}
\frac{\frac{t}{1::P}}{1::P} \xrightarrow{[1]} t \qquad \frac{\frac{\frac{\tau \quad \sigma}{\varphi \quad \psi}}{f+g::A \wedge B} \quad \frac{\rho}{k+l::A \wedge B} \quad \theta}{h::C} \xrightarrow{[\wedge]} \frac{\frac{\tau \quad \sigma}{\varphi \quad \psi}}{f::A \quad g::B} \quad \frac{\rho}{k::A \quad l::B} \quad \theta}{h::C} \\
\frac{\frac{\tau \quad s}{\varphi \quad [k,l]::A}}{f::B} \xrightarrow{[c]} \frac{\tau \quad s \quad s}{\varphi \quad k::A \quad l::A} \quad \frac{f::B}{f::B} \quad \frac{\sigma}{\psi} \xrightarrow{[\Rightarrow]} \frac{\tau}{\varphi} \quad \frac{\sigma}{\psi} \\
\frac{\tau \quad s}{\varphi \quad \emptyset::A} \xrightarrow{[w]} \frac{\tau}{\varphi} \quad \frac{f::B}{f::B} \quad \frac{\tau \quad k \triangleright g::A \Rightarrow B}{\varphi \quad f \triangleright l::A \Rightarrow B} \quad \frac{\rho}{\theta} \xrightarrow{[\Rightarrow]} \frac{\tau}{\varphi} \quad \frac{\sigma}{\psi} \\
\frac{f::B}{f::B} \quad \frac{\tau}{\varphi} \quad \frac{f::B}{f::B} \quad \frac{\tau}{\varphi} \quad \frac{f::B}{f::B} \quad \frac{g::B}{l::B} \quad \frac{\rho}{\theta}}{h::C}
\end{array}$$

■ **Figure 6** Reduction rules.

203 We will assume that constants represent primitives of base type, such as integers and
 204 booleans, and functions over base types, such as addition. We extend the reduction rule $[\Rightarrow]$
 205 to the latter case as below; an example instance would be where c is the integer 7 and c' is a
 206 squaring function, with the resulting constant c'' the integer 49.

$$\frac{\frac{c}{1::P} \quad \frac{c'}{1 \triangleright k::P \Rightarrow A} \quad \frac{\tau}{\varphi}}{f::B} \xrightarrow{[\Rightarrow]} \frac{c''}{k::A} \quad \frac{\tau}{\varphi} \quad \frac{f::B}{f::B}$$

(1, 1 || k, φ, f)

208 3.1 Reduction examples

209 We illustrate reduction with an example analogous to the following lambda calculus reduction,
 210 applying the Church numeral two $\lambda f.\lambda x.f(fx) : (N \Rightarrow N) \Rightarrow N \Rightarrow N$ to the squaring function
 211 constant $S : N \Rightarrow N$ and the integer constant $3 : N$.

$$\text{212 } (\lambda f.\lambda x.f(fx))S3 \rightarrow (\lambda x.S(Sx))3 \rightarrow S(S3) \rightarrow S9 \rightarrow 81$$

213 The combinatorial proof **TWO** corresponding to the Church numeral is the penultimate one
 214 displayed in Figure 3. Below, from left to right, we have: numeral two in compact form;
 215 two in sequent form; two as a node in a combinatorial tree; and the combinatorial tree representing
 216 $(\lambda f.\lambda x.f(fx))S3$.

$$\text{217 } \begin{array}{c} \text{Diagram 1: } (N \Rightarrow N) \Rightarrow N \Rightarrow N \\ \text{Diagram 2: } N \Rightarrow N, N \vdash N \\ \text{Diagram 3: } \frac{N \Rightarrow N \quad N}{N} \text{ TWO} \\ \text{Diagram 4: } \frac{S \quad 3}{N \Rightarrow N \quad N} \text{ TWO} \end{array}$$

218 The reduction sequence is as follows:

$$\text{219 } \frac{S \quad 3}{N \Rightarrow N \quad N} \text{ TWO} \xrightarrow{[c]} \frac{S \quad S \quad 3}{N \Rightarrow N \quad N \Rightarrow N \quad N} \xrightarrow{[\Rightarrow]} \frac{S \quad 9}{N \Rightarrow N \quad N} \xrightarrow{[\Rightarrow]} \frac{81}{N \Rightarrow N \quad N} \xrightarrow{[1]} 81$$

220 For a richer example we consider the ICP version of the Church successor $\lambda n.\lambda f.\lambda x.f(nfx)$
 221 applied to Church zero $\lambda f.\lambda x.x$, the squaring function $S : N \Rightarrow N$ and 4, to yield 16.

$$\text{222 } (\lambda n.\lambda f.\lambda x.f(nfx)) (\lambda f.\lambda x.x) S 4 \rightarrow 16$$

223 The ICP reduction is shown in Figure 7.

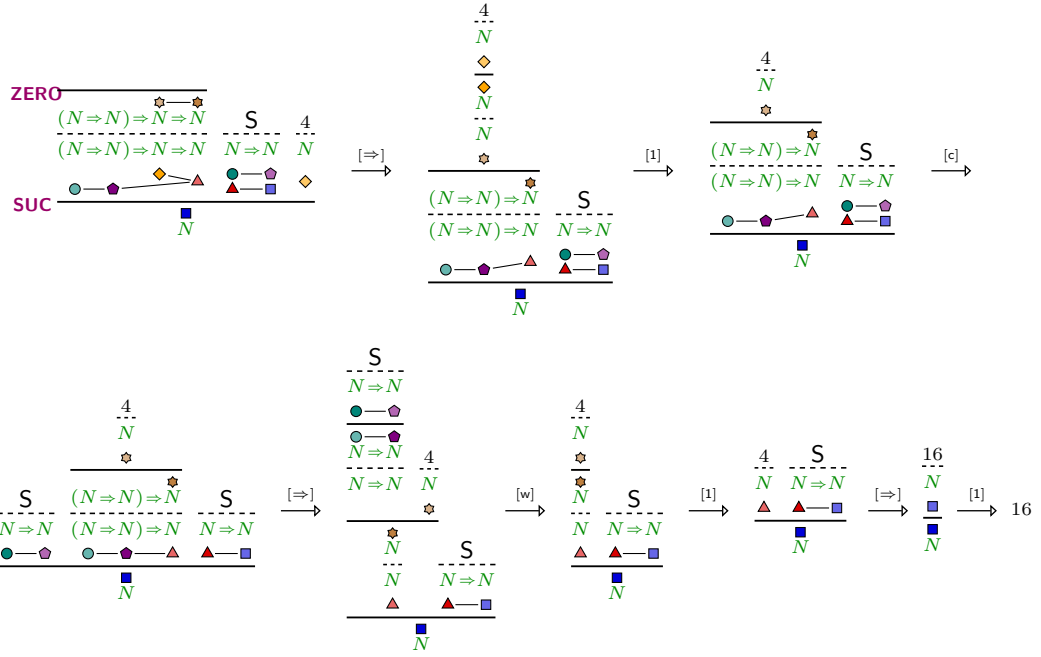


Figure 7 Example of ICP normalization corresponding to the lambda calculus normalization of the Church successor function applied to Church zero, the squaring function constant S, and the constant 4: $(\lambda n.\lambda f.\lambda x.f(nfx)) (\lambda f.\lambda x.x) S 4 \rightarrow^* 16$.

224 **4 Strong Reduction**

225 The reduction rules $[\wedge], [=>]$ apply only when the two ports involved are both open (this is
 226 what the side-conditions on the reduction rules entail). We briefly show that this does not
 227 lead to a deadlock. In a combinatorial tree, a port is **extremal** if it is connected to a premiss
 228 or the consequent of the root node, otherwise **internal**.

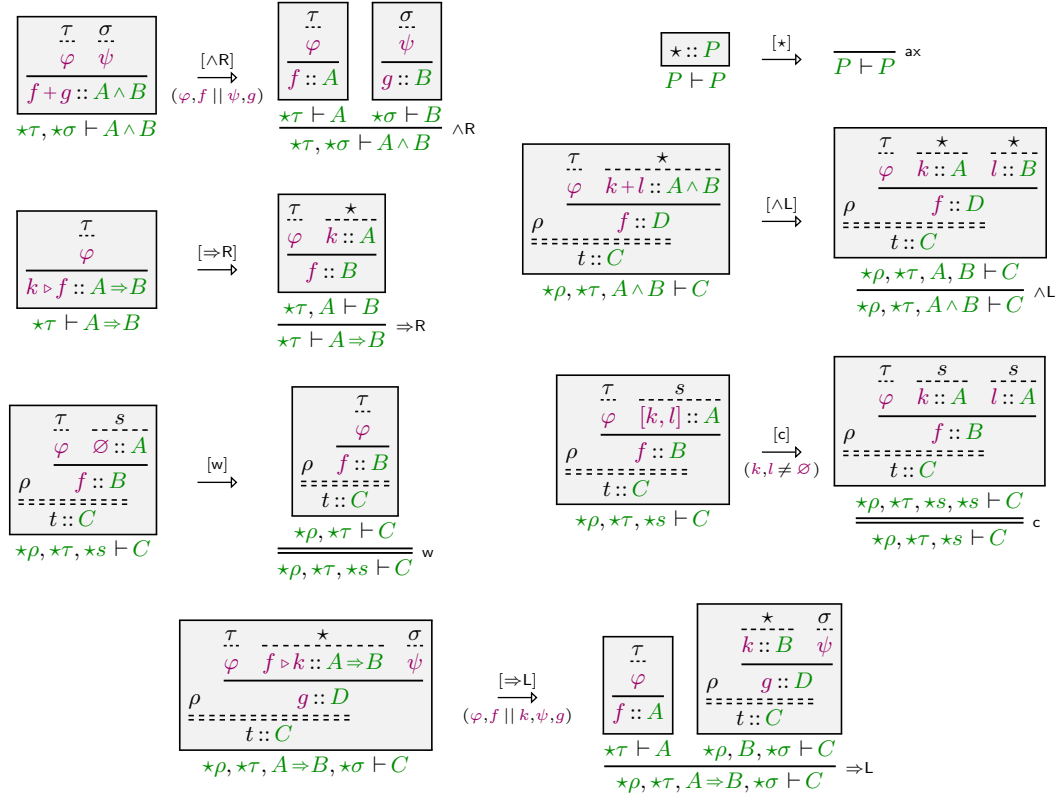
229 ▶ **Lemma 6 (Progression)**. *For a combinatorial tree t with at least one edge, if no extremal*
 230 *port is open, then a reduction step applies.*

231 The progression lemma illustrates a limitation of the normalization process: reduction
 232 may become deadlocked if an extremal port remains open. This is closely related to *weak*
 233 reduction in the λ -calculus, which does not reduce under an abstraction, though note it is not
 234 the same: internal reduction in a combinatorial tree is allowed, and may still be possible,
 235 when the root node is an abstraction. As with weak reduction, this is no limitation in practice:
 236 we expect a real program to be of base type, and without free variables (the premisses of
 237 a combinatorial tree). In that case the progression lemma guarantees we will not reach a
 238 deadlock. This explains also the reason to include constants: without them it is impossible to
 239 create a combinatorial tree of base type with no premisses, as it would logically be unsound.

240 To reduce any combinatorial tree, we combine reduction with sequentialization. This enables
 241 us to reduce open extremal ports, by interpreting them as sequent rules. We add a special
 242 axiom (icp), given below, to the cut-free sequent calculus. It incorporates a combinatorial tree
 243 t for $\Gamma \vdash A$ as a sub-proof of $\Gamma \vdash A$. A proof in this calculus is a **hybrid proof**.

244
$$\frac{t :: A}{\star t \vdash A} \text{ (icp)}$$

245 The reduction rules $[ax], [\wedge], [=>]$ apply directly to hybrid proofs, since they preserve the
 246 premisses and conclusion of a combinatorial tree. The rules $[c]$ and $[w]$ duplicate or delete



■ **Figure 8** Hybrid sequentialization and reduction rules

247 premisses; to accommodate this in hybrid proofs, contraction or weakening rules are added.
 248 The resulting rules are the last two in Figure 8, which gives the rules needed for strong
 249 reduction.

250 ▶ **Definition 7** (Hybrid reduction). *Hybrid proof reduction* is the rewrite relation on hybrid
 251 proofs generated by the rules [ax], [∧], [⇒] in Figure 6 plus the rules in Figure 8.

252 Progression (Lemma 6) gives the following.

253 ▶ **Lemma 8** (Hybrid progression). *If a hybrid proof contains an (icp) axiom, a hybrid reduction*
 254 *step applies.*

255 A normal form of a hybrid proof is then a regular, cut-free sequent proof. This may directly be
 256 used to construct an ICP, to obtain fully general ICP normalization. The effect of embedding
 257 a combinatorial tree in a hybrid proof is akin to *normalization-by-evaluation* [5]: it provides an
 258 environment that supplies sufficient arguments to any function (it is an *applicative context*),
 259 and other similar services, to ensure continued reduction.

260 For the remainder of this paper, we will use the rules in Figure 8 on combinatorial trees
 261 in isolation, without reference to the surrounding hybrid proof.

262 5 Confluence and strong normalization

263 Combinatorial-tree reduction is confluent and strongly normalizing. In this section we will
 264 consider only *local confluence*, which demonstrates the intricacies arising from the local

XX:10 Normalization Without Syntax

265 canoncity property of ICPs. Strong normalization is stated here, and proved in Appendix C;
 266 confluence then follows from local confluence and strong normalization.

267 The reduction rules for ICPs interact in several intricate ways. Not only can a single node
 268 have multiple redexes along different edges, even a single edge may reduce in more than one
 269 way. This is due to the multiple ways an arena morphism can be composed inductively, which
 270 factor out the formula equivalences of associativity, symmetry, and currying, as well as the
 271 interaction of conjunction with contraction. Concretely, we have the following equations:

$$\begin{array}{ll}
 f+g = g+f & \emptyset + \emptyset = \emptyset \\
 f+(g+h) = (f+g)+h & [k, \emptyset] = k \\
 (k+l) \triangleright f = k \triangleright (l \triangleright f) & [k_1, k_2] + [l_1, l_2] = [k_1+l_1, k_2+l_2]
 \end{array}$$

273 We recognize two kinds of critical pairs:

274 **Single-edge** when multiple reduction steps apply to a single cut-edge, due to the above
 275 equations;

276 **Single-node** when multiple reduction steps on distinct edges split the same node.

277 The single-node critical pairs are similar to those of the λ -calculus and proof nets, and these
 278 converge accordingly; we give the diagrams in Appendix B. The single-edge critical pairs are
 279 new and delicate. We resolve them in Figure 9. For convenience we introduce the notation
 280 $s + t$, below. In the first four diagrams we depict only the ports and subtrees involved, and in
 281 the last diagram we use a different colouring scheme to identify ports across diagrams.

$$282 \quad t + s = \frac{\frac{\tau}{\varphi} \quad \frac{\sigma}{\psi}}{f+g} \quad \text{where} \quad t = \frac{\tau}{f} \quad s = \frac{\sigma}{g}$$

283 We use \twoheadrightarrow for the reflexive-transitive closure of \rightarrow , and dashed arrows are implied by the
 284 diagram. From top to bottom the diagrams are due to the equations:

$$\begin{array}{ll}
 \emptyset + \emptyset = \emptyset & [k_1, k_2] + l = [k_1+l, k_2+\emptyset] \\
 k+(l+m) = (k+l)+m & (k+l) \triangleright f = k \triangleright (l \triangleright f) \\
 [k_1, k_2] + [l_1, l_2] = [k_1+l_1, k_2+l_2] &
 \end{array}$$

286 **► Proposition 9.** *Reduction \twoheadrightarrow is locally confluent.*

287 **Proof.** By Figure 9 and Figures 11–13 in Appendix B. ◀

288 **► Theorem 10** (Strong normalization). *Combinatorial-tree reduction is strongly normalizing.*

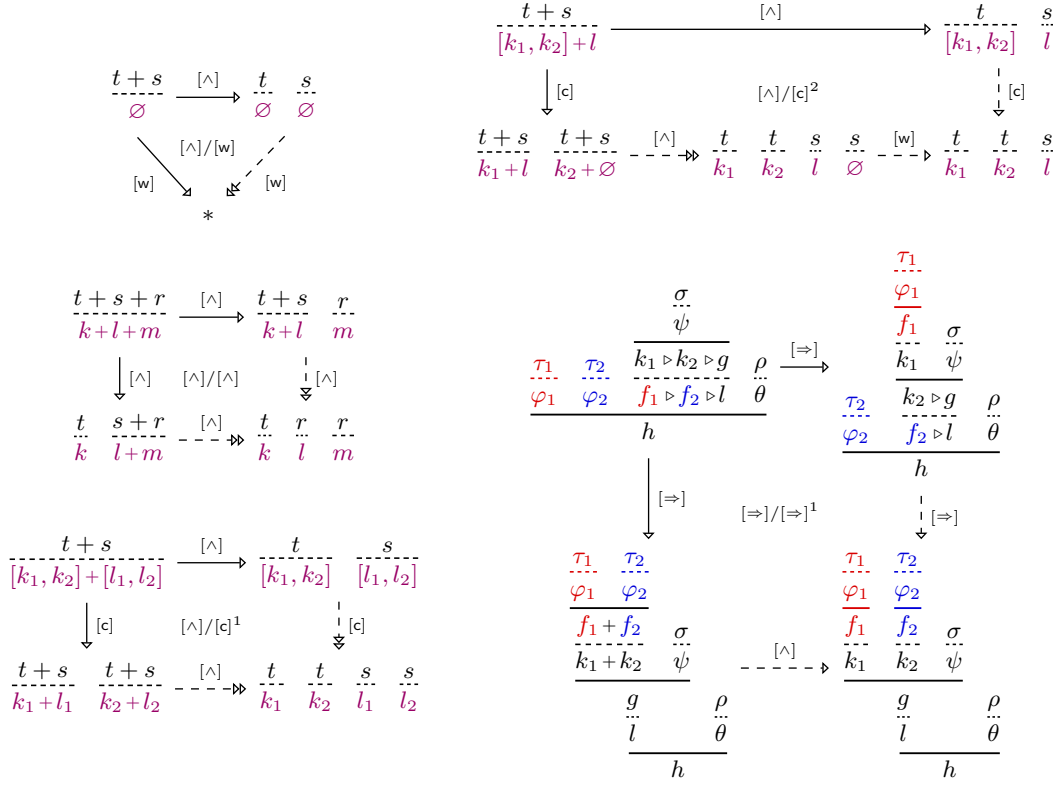
289 **Proof.** See Appendix C. ◀

290 6 Combinatory lambda-calculus

291 To further illustrate the reduction process, we connect ICPs to the λ -calculus, via an
 292 explicit-substitution λ -calculus that we call the **combinatory λ -calculus**. The calculus is a
 293 Curry–Howard interpretation of sequent calculus, of the kind studied by Graham-Lengrand
 294 [24]. We include constants c to match those of combinatorial trees.

295 **► Definition 11.** *The **combinatory λ -calculus** has **normal terms** N, M , **patterns** p, q ,
 296 and **terms** S, T given by the following grammars.*

$$\begin{array}{l}
 297 \quad N, M ::= x \mid \langle N, M \rangle \mid \lambda p. N \mid N[p \leftarrow xM] \\
 p, q ::= x \mid \langle p, q \rangle \quad S, T ::= c \mid N[p_1 \leftarrow T_1, \dots, p_n \leftarrow T_n]
 \end{array}$$



■ **Figure 9** Single-edge confluence diagrams

298 The **binding variables** $\text{bv}(p)$ of p and the **free variables** $\text{fv}(N)$ of N are as follows; in
 299 $N[p \leftarrow xM]$ we require that $\text{fv}(N) \cap \text{bv}(p) \neq \emptyset$, and in $\langle p, q \rangle$ that $\text{bv}(p) \cap \text{bv}(q) = \emptyset$.

$$\begin{aligned}
 \text{bv}(x) &= x & \text{bv}(\langle p, q \rangle) &= \text{bv}(p) \cup \text{bv}(q) \\
 \text{fv}(x) &= x & \text{fv}(\langle N, M \rangle) &= \text{fv}(N) \cup \text{fv}(M) \\
 \text{fv}(\lambda p. N) &= \text{fv}(N) - \text{bv}(p) & \text{fv}(N[p \leftarrow xM]) &= (\text{fv}(N) - \text{bv}(p)) \cup \{x\} \cup \text{fv}(M)
 \end{aligned}$$

301 In $\lambda p. N$, $N[p \leftarrow xM]$, and $N[p_1 \leftarrow T_1, \dots, p_n \leftarrow T_n]$ the variables in the patterns p and p_i bind
 302 in N . The construction $N[p \leftarrow xM]$ is a **shared application**, with a variable x as function and
 303 the term M as argument, where the pattern p may bind variables with multiple occurrences
 304 in N . The condition that $\text{bv}(p)$ and $\text{fv}(N)$ must intersect means at least one variable becomes
 305 bound; this corresponds to the condition (\dagger) on the rule $\Rightarrow\text{L}$ for ICPs in Figure 2 (that the
 306 consequent of a left-implication must not be introduced by weakening). The construction
 307 $[p_1 \leftarrow T_1, \dots, p_n \leftarrow T_n]$ is an **environment**, and corresponds to attaching the subtrees to a
 308 node in a combinatorial tree. We abbreviate it by $[e]$, or $[p_1 \leftarrow T_1, e]$, etc.

309 ▶ **Definition 12.** Figure 10 gives the (non-deterministic) **translation** from ICPs to simply-
 310 typed, normal terms of the combinatory λ -calculus. We extend it to combinatorial trees as
 311 follows: \Rightarrow is the identity on constants, and if

$$312 \quad k_1, \dots, k_n, \varphi \vdash f \quad \Rightarrow \quad p_1 : A_1, \dots, p_n : A_n, \Delta \vdash N : B$$

313 and if $t_i \Rightarrow \Gamma_i \vdash T_i : A_i$ (with $t_i \neq \star$) for all $i \leq n$, then

XX:12 Normalization Without Syntax

$$\begin{array}{c}
\frac{}{1 \vdash 1 \Rightarrow x:P \vdash x:P} \langle\langle 1 \rangle\rangle \\
\frac{\varphi \vdash f \Rightarrow \Gamma \vdash N:C}{\varphi, \emptyset \vdash f \Rightarrow \Gamma, p:A \vdash N:C} \langle\langle w \rangle\rangle \\
\frac{\varphi, k, l \vdash f \Rightarrow \Gamma, p:A, p:A \vdash N:C}{\varphi, [k, l] \vdash f \Rightarrow \Gamma, p:A \vdash N:C} \langle\langle c \rangle\rangle \\
\frac{\varphi, k \vdash f \Rightarrow \Gamma, p:A \vdash N:B}{\varphi \vdash k \triangleright f \Rightarrow \Gamma \vdash \lambda p. N : A \Rightarrow B} \langle\langle \Rightarrow R \rangle\rangle \\
\frac{\varphi, k, l \vdash f \Rightarrow \Gamma, p:A, q:B \vdash N:C}{\varphi, k+l \vdash f \Rightarrow \Gamma, \langle p, q \rangle : A \wedge B \vdash N:C} \langle\langle \wedge L \rangle\rangle \\
\frac{\varphi \vdash f \Rightarrow \Gamma \vdash N:A \quad \psi \vdash g \Rightarrow \Delta \vdash M:B}{\varphi, \psi \vdash f+g \Rightarrow \Gamma, \Delta \vdash \langle N, M \rangle : A \wedge B} \langle\langle \wedge R \rangle\rangle \\
\frac{\varphi \vdash f \Rightarrow \Gamma \vdash M:A \quad k, \psi \vdash g \Rightarrow p:B, \Delta \vdash N:C}{\varphi, f \triangleright k, \psi \vdash g \Rightarrow \Gamma, x:A \Rightarrow B, \Delta \vdash N[p \leftarrow xM] : C} \langle\langle \Rightarrow L \rangle\rangle
\end{array}$$

■ **Figure 10** From ICPs to simply-typed normal terms

$$\frac{\frac{t_1}{k_1} \cdots \frac{t_n}{k_n} \star}{f} \quad \Rightarrow \quad \Gamma_1, \dots, \Gamma_n, \Delta \vdash N[p_1 \leftarrow T_1, \dots, p_n \leftarrow T_n] : B .$$

315 The shared applications $[p \leftarrow xM]$ of the combinatory λ -calculus are subject to permutations,
316 creating an equivalence \sim on terms. We define it below, where we abbreviate $[p \leftarrow xM]$ by $[a]$,
317 with $\text{bv}(a) = \text{bv}(p)$ and $\text{fv}(a) = \{x\} \cup \text{fv}(M)$.

$$\begin{array}{l}
\langle N[a], M \rangle \sim \langle N, M \rangle[a] \quad \text{bv}(a) \cap \text{fv}(M) = \emptyset \\
\langle N, M[a] \rangle \sim \langle N, M \rangle[a] \quad \text{bv}(a) \cap \text{fv}(N) = \emptyset \\
\lambda p. \langle N[a] \rangle \sim \langle \lambda p. N \rangle[a] \quad \text{bv}(p) \cap \text{fv}(a) = \emptyset \\
N[p \leftarrow xM[a]] \sim N[p \leftarrow xM][a] \quad \text{bv}(a) \cap \text{fv}(N) = \emptyset \\
N[a][b] \sim N[b][a] \quad \text{bv}(b) \cap \text{fv}(a) = \emptyset, \text{bv}(a) \cap \text{fv}(b) = \emptyset
\end{array}$$

319 The above equivalence factors out sequent calculus permutations. We will further assume
320 combinatory λ -terms equivalent modulo the formula-isomorphisms (symmetry, associativity,
321 and currying). These are factored out simply by considering patterns modulo these rules, but
322 there is a catch: patterns and pairs are connected through cuts, or explicit substitutions,
323 and laws must be applied to both simultaneously. We show an example with currying to
324 demonstrate that a full definition is intricate, and leave it implicit.

$$325 \quad N[z \leftarrow x \langle P, Q \rangle][x \leftarrow \lambda \langle p, q \rangle. M] \sim N[z \leftarrow yQ][y \leftarrow xP][x \leftarrow \lambda p. \lambda q. M]$$

326 With the above equivalence on terms, a direct corollary of local canonicity, Theorem 3, is the
327 following.

328 ► **Proposition 13.** *Combinatorial trees canonically represent typed combinatory λ -terms:*

$$329 \quad S \sim T \iff \exists t. t \Rightarrow S \wedge t \Rightarrow T$$

330 We reduce combinatory λ -terms modulo the equivalence \sim . We write $\{T/x\}$ for the substitution
331 of x by T , and if the patterns p, q are isomorphic as trees and $\text{bv}(p) \cap \text{bv}(q) = \emptyset$ then $\{q/p\}$
332 is the substitution induced by

$$333 \quad \{\langle q_1, q_2 \rangle / \langle p_1, p_2 \rangle\} = \{q_1/p_1\} \{q_2/p_2\} .$$

334 ► **Definition 14.** *Reduction of combinatory λ -terms modulo \sim is by the following rules, where:*
335 *$[e_P]$ and $[e_Q]$ bind only in P respectively Q ; in $\langle \Rightarrow \rangle$ we require $x \notin \text{fv}(P) \cup \text{fv}(Q)$; in $\langle c \rangle$ we*
336 *require $\text{bv}(q) \cap \text{fv}(N) \neq \emptyset$; and in $\langle w \rangle$ that $\text{bv}(p) \cap \text{fv}(N) = \emptyset$.*

$$\begin{array}{l}
N[x \leftarrow y[e], e'] \xrightarrow{\langle 1 \rangle} N\{y/x\}[e, e'] \\
N[\langle p, q \rangle \leftarrow \langle P, Q \rangle[e_P, e_Q], e] \xrightarrow{\langle \wedge \rangle} N[p \leftarrow P[e_P], q \leftarrow Q[e_Q], e] \\
P[p \leftarrow xQ][e_Q, x \leftarrow \lambda q. N[e], e_P] \xrightarrow{\langle \Rightarrow \rangle} P[p \leftarrow N[q \leftarrow Q[e_Q], e], e_P] \\
N\{p/q\}[p \leftarrow T, e] \xrightarrow{\langle c \rangle} N[q \leftarrow T, p \leftarrow T, e] \\
N[p \leftarrow T, e] \xrightarrow{\langle w \rangle} N[e]
\end{array}$$

338 Comparing the reduction rules with the corresponding ones for ICPs in Figure 6, together
339 with Proposition 13, gives:

340 ▶ **Proposition 15.** *Reduction on ICPs and combinatory λ -terms (modulo equivalence) com-*
341 *mutates with interpretation*

$$342 \quad \begin{array}{ccc} t & \xrightarrow{[x]} & s \\ \Downarrow & & \Downarrow \\ T & \xrightarrow{\langle x \rangle} & S \end{array}$$

343 The comparison with λ -calculus allows us to make a further observation. ICP normalization
344 is a form of *closed reduction* [6] (there called *weak reduction*), where a redex $(\lambda x.N)M$ may
345 not be reduced if M contains free variables that are bound by the surrounding context. This
346 has the enormous benefit to implementation that alpha-conversion becomes unnecessary. Our
347 construction of combinatorial trees is even stronger: it is impossible to construct such a redex,
348 or to produce one by reduction. This can be observed from the combinatory λ -calculus, which
349 does not support abstraction at the level of terms T , only at the level of normal terms.

350 Abstraction on terms can be introduced as a defined operation, called **lambda-lifting**
351 [22]. The analogous operation on ICP combinatorial trees would be a transformation

$$352 \quad \frac{\star :: A \quad \star :: \Gamma}{t :: B} \mapsto \frac{\star :: \Gamma}{t' :: A \Rightarrow B}.$$

353 We can perform it by abstracting over $\star :: A$ locally, in the node where it resides, and transform
354 every node on the path from there to the root as follows,

$$355 \quad \frac{k :: C \quad \varphi}{f :: D} \mapsto \frac{i \triangleright k :: A \Rightarrow C \quad \varphi}{i \triangleright f :: A \Rightarrow D}$$

356 where the port $k :: C$ is that on the path to $\star :: A$, and the arena morphism $i : \llbracket A \rrbracket \rightarrow \llbracket A \rrbracket$ is
357 the identity on $\llbracket A \rrbracket$. In effect, one is threading the abstraction over A *through* the cuts in the
358 tree, rather than adding it as a connection *outside* of them.

359 By way of example, below is the reduction corresponding to the ICP normalization
360 sequence in Figure 7.

$$361 \quad \begin{array}{l} v[v \leftarrow gw][w \leftarrow yz][y \leftarrow ng][n \leftarrow \lambda f. \lambda x. x, g \leftarrow S, z \leftarrow 4] \\ \sim v[v \leftarrow gw][w \leftarrow yg][y \leftarrow nz][n \leftarrow \lambda x. \lambda f. x, z \leftarrow 4, g \leftarrow S] \\ \xrightarrow{\langle \Rightarrow \rangle} v[v \leftarrow gw][w \leftarrow yg][y \leftarrow \lambda f. x[x \leftarrow z[z \leftarrow 4]], g \leftarrow S] \\ \xrightarrow{\langle 1 \rangle} v[v \leftarrow gw][w \leftarrow yg][y \leftarrow \lambda f. x[x \leftarrow 4]], g \leftarrow S] \\ \xrightarrow{\langle c \rangle} v[v \leftarrow gw][w \leftarrow yh][y \leftarrow \lambda f. x[x \leftarrow 4]], g \leftarrow S, h \leftarrow S] \\ \xrightarrow{\langle \Rightarrow \rangle} v[v \leftarrow gw][w \leftarrow x[f \leftarrow h[h \leftarrow S], x \leftarrow 4], g \leftarrow S] \dots \end{array} \left| \begin{array}{l} \dots \\ \xrightarrow{\langle w \rangle} v[v \leftarrow gw][w \leftarrow x[x \leftarrow 4], g \leftarrow S] \\ \xrightarrow{\langle 1 \rangle} v[v \leftarrow gw][w \leftarrow 4, g \leftarrow S] \\ \xrightarrow{\langle \Rightarrow \rangle} v[v \leftarrow 16] \\ \xrightarrow{\langle 1 \rangle} 16 \end{array} \right.$$

362 7 Supercombinators

363 Supercombinators [20] are the basis of an efficient implementation of functional program-
364 ming [25], used for the Haskell programming language. The main reason for their efficiency is
365 that expressions are compiled into trees (or graphs) over a fixed set of operators, each given
366 as an instruction set that implements the appropriate reduction sequence.

367 ▶ **Definition 16.** *Supercombinators C, D and supercombinator expressions E_X, F_X ,*
368 *where X is a set of variables, are given by the following grammars.*

$$369 \quad C, D ::= \lambda x_1 \dots \lambda x_n. E_{\{x_1, \dots, x_n\}} \quad E_X, F_X ::= x \in X \mid C \mid F_X E_X$$

XX:14 Normalization Without Syntax

370 The set X restricts which variables may occur free in a supercombinator expression,
 371 so that each supercombinator is a closed term; we may omit it as superscript for brevity.
 372 The grammar for supercombinators C may be extended to include constants. Reduction
 373 is *weak head reduction* on an expression E_{\emptyset} , as given by the rule below. It applies only at
 374 top-level, not in context, and if there are fewer than n arguments to a supercombinator with
 375 n abstractions, reduction halts.

$$376 \quad (\lambda x_1 \dots \lambda x_n. E) F_1 \dots F_n F_{n+1} \dots F_{n+m} \mapsto E\{F_1/x_1\} \dots \{F_n/x_n\} F_{n+1} \dots F_{n+m}$$

377 During reduction, substitutions are applied only to the top-level E_{\emptyset} expression, and not to
 378 supercombinators, which remain fixed. This allows them to be compiled into instruction sets
 379 to carry out the appropriate reduction by the rule \mapsto above.

380 Structurally, supercombinators are trees or graphs where each node is a supercombinator
 381 C in which each occurring supercombinator D is considered as a *pointer* to the node for D .
 382 This is highly similar to combinatorial trees, which feature the same tree structure except
 383 with ICPs for nodes. The main dissimilarities between supercombinators and combinatorial
 384 trees are then as follows.

- 385 ■ Supercombinator reduction is by an abstract machine, where combinatorial-tree reduction
- 386 is a variant of cut-elimination.
- 387 ■ Supercombinators are trees over β -normal λ -terms where abstractions may not occur under
- 388 an application, where nodes in combinatorial trees are η -expanded β -normal sequent proofs
- 389 modulo permutations.

390 These differences are conceptually shallow, but risk burying a formal comparison in
 391 technicalities. We will therefore interpret supercombinators in the combinatory λ -calculus
 392 instead (which, mainly, does not require η -expansion), and simulate reduction only up to
 393 explicit substitutions.

394 ► **Definition 17.** *The relations \blacktriangleright and \triangleright , defined inductively below, interpret supercombinators*
 395 *respectively supercombinator expressions into the combinatory λ -calculus.*

$$396 \quad \frac{E \triangleright N[e]}{\lambda x_1 \dots \lambda x_n. E \blacktriangleright (\lambda x_1 \dots \lambda x_n. N)[e]} \quad \frac{C \blacktriangleright T}{C \triangleright x[x \leftarrow T]} \quad \frac{E \triangleright x[a_1] \dots [a_k][e] \quad F \triangleright N[f]}{EF \triangleright y[y \leftarrow xN][a_1] \dots [a_k][e, f]}$$

397 Note how this indeed translates a supercombinator to a term $(\lambda x_1 \dots \lambda x_n. N)[e]$ consisting
 398 of a normal form $\lambda x_1 \dots \lambda x_n. N$ with a subtree for each occurring supercombinator in the
 399 explicit substitutions $[e]$. To simulate reduction, a reduct is translated as follows.

$$400 \quad \frac{\frac{E \triangleright M[e]}{\lambda x_1 \dots \lambda x_n. E \blacktriangleright (\lambda x_1 \dots \lambda x_n. M)[e]}}{\lambda x_1 \dots \lambda x_n. E \triangleright y[y \leftarrow (\lambda x_1 \dots \lambda x_n. M)[e]]} \quad F_1 \triangleright N_1[f_1] \quad \dots \quad F_n \triangleright N_n[f_n]}{(\lambda x_1 \dots \lambda x_n. E) F_1 \dots F_n \triangleright z_n[z_n \leftarrow z_{n-1}N_n] \dots [z_1 \leftarrow yN_1][y \leftarrow (\lambda x_1 \dots \lambda x_n. M)[e], f_1, \dots, f_n]}$$

401 Reduction for this term proceeds as follows.

$$402 \quad \begin{aligned} & z_n[z_n \leftarrow z_{n-1}N_n] \dots [z_2 \leftarrow z_1N_2][z_1 \leftarrow yN_1][y \leftarrow (\lambda x_1. \lambda x_2 \dots \lambda x_n. M)[e], f_1, f_2, \dots, f_n] \\ \langle \Rightarrow \rangle & \rightarrow z_n[z_n \leftarrow z_{n-1}N_n] \dots [z_2 \leftarrow z_1N_2][z_1 \leftarrow (\lambda x_2 \dots \lambda x_n. M)[x \leftarrow N_1[f_1], e], f_2, \dots, f_n] \\ \langle \Rightarrow \rangle & \rightarrow z_n[z_n \leftarrow M[x_1 \leftarrow N_1[f_1], \dots, x_n \leftarrow N_n[f_n], e]] \end{aligned}$$

403 The result corresponds to the supercombinator reduct $E\{F_1/x_1\} \dots \{F_n/x_n\}$, except that
 404 the explicit substitutions $[x_i \leftarrow N_i[f_i]]$ are not evaluated as substitutions. They cannot be:
 405 combinatory λ -term reduction does not differentiate between the interpretation of the top-
 406 level supercombinator expression E_{\emptyset} on which reduction takes place, and which does admit
 407 substitutions, and internal subcombinator expressions which do not. We will therefore contend
 408 ourselves with the moral equivalence of both reductions.

8 Lambda-calculus

To complete the exposition, we map the combinatory λ -calculus onto the regular λ -calculus with surjective pairing. We have the following terms and rewrite rules, where $i \in \{1, 2\}$.

$$M, N ::= x \mid \lambda x. N \mid NM \mid \pi_i N \mid \langle N, M \rangle \quad (\lambda x. N)M \rightarrow_\beta N\{M/x\} \quad \pi_i \langle N_1, N_2 \rangle \rightarrow_\pi N_i$$

The translation from combinatory λ -terms into λ -terms $\llbracket \cdot \rrbracket$ is as follows, where we substitute for a pattern via $\{N/\langle p, q \rangle\} = \{\pi_1 N/p, \pi_2 N/q\}$.

$$\begin{aligned} \llbracket x \rrbracket &= x \\ \llbracket \langle M, N \rangle \rrbracket &= \langle \llbracket M \rrbracket, \llbracket N \rrbracket \rangle \\ \llbracket \lambda p. N \rrbracket &= \lambda x. \llbracket N \rrbracket \{x/p\} \\ \llbracket N[p \leftarrow x M] \rrbracket &= \llbracket M \rrbracket \{x \llbracket N \rrbracket / p\} \\ \llbracket N[p_1 \leftarrow T_1, \dots, p_n \leftarrow T_n] \rrbracket &= \llbracket N \rrbracket \{ \llbracket T_1 \rrbracket / p_1 \} \dots \{ \llbracket T_n \rrbracket / p_n \} \end{aligned}$$

The combined translation then takes ICP combinatorial trees to λ -terms. As with the combinatory λ -calculus, we assume λ -terms equivalent (\sim) modulo formula-isomorphisms (symmetry, associativity, currying). Sequent permutations are already naturally factored out, but at the cost of exponential growth. We will demonstrate this here.

In the combinatory λ -calculus, the reason that an application must occur in an explicit substitution is precisely that the consequent of a left-implication may have been contracted, the situation highlighted in the introduction:

$$\frac{\Gamma \vdash A \quad \frac{B, B, \Delta \vdash C}{B, \Delta \vdash C}^c}{\Gamma, A \Rightarrow B, \Delta \vdash C}^{\Rightarrow L} \quad \approx \quad \frac{\Gamma \vdash A \quad \frac{B, B, \Delta \vdash C}{B, \Gamma, A \Rightarrow B, \Delta \vdash C}^{\Rightarrow L}}{\Gamma, A \Rightarrow B, \Gamma, A \Rightarrow B, \Delta \vdash C}^{\Rightarrow L}}{\Gamma, A \Rightarrow B, \Delta \vdash C}^c$$

The corresponding equivalence on combinatory terms is:

$$N\{p/q\}[p \leftarrow x M] \quad \approx \quad N[q \leftarrow x M][p \leftarrow x M]$$

(where $\text{bv}(q) \cap \text{fv}(N) \neq \emptyset$), while both translate to the same λ -term $\llbracket N \rrbracket \{x \llbracket M \rrbracket / p\}$. Repeated duplication incurred in this way gives rise to exponential growth.

Let **strong equivalence** $S \approx T$ on combinatory λ -terms be the equivalence generated by the above and \sim . We have the following proposition.

► **Proposition 18.** *For combinatory terms S, T , we have*

$$S \approx T \iff \llbracket S \rrbracket = \llbracket T \rrbracket .$$

9 Conclusion

We have given a direct and natural account of normalization for intuitionistic combinatorial proofs. We believe our approach of *external rewriting*, here manifested in the notion of *combinatorial tree*, applies much more broadly: in the abstract, what we have are simply trees of normal forms, with the natural reduction rules given by the meta-level sequent calculus. As a generalization of super-combinators, a correspondence we aim to make more precise in future work, we hope that our approach leads to improvements in compiler design. Perhaps the ability to express all normal forms, and the more fine-grained reduction steps, will allow more efficient program transformations, while retaining the benefits of super-combinators.

Acknowledgments

Dominic Hughes would like to thank Wes Holliday, his host at U.C. Berkeley.

443 — References

- 444 1 Samson Abramsky, Radha Jagadeesan, and Pasquale Malacaria. Full abstraction for PCF.
445 *Information and Computation*, 163:409–470, 1996.
- 446 2 Matteo Acclavio and Lutz Straßburger. On combinatorial proofs for logics of relevance and
447 entailment. In Rosalie Iemhoff and Michael Moortgat, editors, *26th Workshop on Logic,
448 Language, Information and Computation (WoLLIC 2019)*. Springer, 2019.
- 449 3 Matteo Acclavio and Lutz Straßburger. On combinatorial proofs for modal logic. In Serenella
450 Cerrito and Andrei Popescu, editors, *Automated Reasoning with Analytic Tableaux and
451 Related Methods - 28th International Conference, TABLEUX 2019, London, UK, September
452 3-5, 2019, Proceedings*, volume 11714 of *Lecture Notes in Computer Science*, pages 223–
453 240. Springer, 2019. URL: https://doi.org/10.1007/978-3-030-29026-9_13, doi:10.1007/
454 978-3-030-29026-9_13.
- 455 4 Gianluigi Bellin and Jacques van de Wiele. Subnets of proof-nets in MLL⁻. In *Advances in
456 Linear Logic*, pages 249–270, 1995.
- 457 5 Ulrich Berger and Helmut Schwichtenberg. An inverse of the evaluation functional for typed
458 λ -calculus. In *6th Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages
459 203–212, 1991.
- 460 6 Naim Cagman and J. Roger Hindley. Combinatory weak reduction in lambda calculus.
461 *Theoretical Computer Science*, 198(1–2):239–249, 1998.
- 462 7 Gerhard Gentzen. Untersuchungen über das logische Schließen I, II. *Mathematische Zeitschrift*,
463 39:176–210, 405–431, 1934–1935. English translation in: *The Collected Papers of Gerhard
464 Gentzen*, M.E. Szabo (ed.), North-Holland 1969.
- 465 8 Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50(1):1–102, 1987.
- 466 9 Jean-Yves Girard. Geometry of interaction 2: Deadlock-free algorithms. In *International
467 Conference on Computer Logic*, pages 76–93, 1988.
- 468 10 Jean-Yves Girard. Proof-nets: the parallel syntax for proof-theory. *Logic and Algebra*, pages
469 97–124, 1996.
- 470 11 Jean-Yves Girard. Locus solum: From the rules of logic to the logic of rules. *Mathematical
471 Structures in Computer Science*, 11(3):301–506, 2001. doi:10.1017/S096012950100336X.
- 472 12 Willem Heijltjes. Proof nets for additive linear logic with units. In *IEEE 26th Annual
473 Symposium on Logic in Computer Science (LICS)*, pages 207–216, 2011.
- 474 13 Willem Heijltjes, Dominic Hughes, and Lutz Straßburger. Intuitionistic proofs without syntax.
475 In *34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, 2019.
- 476 14 Dominic Hughes. Proofs without syntax. *Annals of Mathematics*, 164(3):1065–1076, 2006.
- 477 15 Dominic Hughes and Willem Heijltjes. Conflict nets: efficient locally canonical mall proof nets.
478 In *31st Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, 2016.
- 479 16 Dominic Hughes and Rob van Glabbeek. Proof nets for unit-free multiplicative-additive linear
480 logic. *Transactions on Computational Logic*, 6(4):784–842, 2005.
- 481 17 Dominic J. D. Hughes. First-order proofs without syntax, 2019. arXiv preprint 1906.11236.
482 arXiv:1906.11236.
- 483 18 Dominic J. D. Hughes, Lutz Straßburger, and Jui-Hsuan Wu. Combinatorial proofs and
484 decomposition theorems for first-order logic. In *36th Annual ACM/IEEE Symposium on Logic
485 in Computer Science, LICS 2021, Rome, Italy, June 29 - July 2, 2021*, pages 1–13. IEEE,
486 2021. URL: <https://doi.org/10.1109/LICS52264.2021.9470579>, doi:10.1109/LICS52264.
487 2021.9470579.
- 488 19 Dominic J.D. Hughes. Simple free star-autonomous categories and full coherence. *Journal of
489 Pure and Applied Algebra*, 216(11):2386–2410, 2012.
- 490 20 R.J.M. Hughes. Super-combinators: a new implementation method for applicative languages.
491 In *ACM Symposium on Lisp and Functional Programming*, pages 1–10, 1982.
- 492 21 J. M. E. Hyland and C.-H. Luke Ong. On full abstraction for PCF: I, II, and III. *Information
493 and Computation*, 163(2):285–408, 2000.
- 494 22 Thomas Johnsson. Lambda lifting: Transforming programs to recursive equations. In *Conference
495 on Functional Programming Languages and Computer Architecture*, volume 201 of *LNCS*, pages
496 190–203, 1985.
- 497 23 Yves Lafont. Interaction nets. In *Proceedings of the 17th ACM SIGPLAN-SIGACT Symposium
498 on Principles of Programming Languages (POPL)*, pages 95–108, 1990.

- 499 24 Stéphane Lengrand. *Normalisation and equivalence in proof theory and type theory*. PhD
500 thesis, University of St. Andrews, 2006.
- 501 25 Simon L. Peyton-Jones. *The implementation of functional programming languages*. Prentice
502 Hall, 1987.
- 503 26 Andrea Aler Tubella and Lutz Straßburger. Introduction to deep inference. Lecture notes for
504 ESLLI'19, 2019. URL: <https://hal.inria.fr/hal-02390267>.



505 **A Proofs for Section 4**

506 ▶ **Lemma 6 (restatement).** *For a combinatorial tree t with at least one edge, if no extremal*
 507 *port is open, then a reduction step applies.*

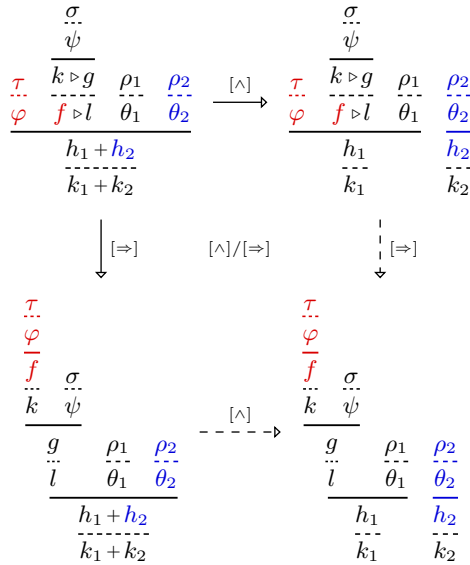
508 **Proof.** At first, we ignore constants. If t contains a node that is an axiom $1 \vdash 1$, or a port
 509 that is a contraction $[k, l]$ (with $k, l \neq \emptyset$) or a weakening $\emptyset_{[A]}$, then a reduction step applies.

510 Otherwise, all ports are of the forms $1, k \triangleright f$, and $f + g$. The $[\wedge, \Rightarrow]$ steps apply if both
 511 ports connected by the cut-edge are open. Let t consist of n nodes. By definition, every node
 512 has at least one open port, and by assumption it is internal. Then since there are only $n - 1$
 513 edges, at least one edge connects two open ports, and a $[\wedge]$ or $[\Rightarrow]$ reduction step applies.

514 To include constants in the argument, we consider c of type $P \Rightarrow A$ as a node with only an
 515 open conclusion port. If c is applied to a tree $s :: P$ of base type, we need to reduce this to a
 516 constant first (see the reduction rule for constants). We can do so by iterating the argument
 517 for the subtree s , since its extremal ports are closed (its conclusion must be $1 :: P$ and its
 518 premisses are also premisses of t). ◀

519 **B Diagrams for Section 5**

520 This section provides Figures 11, 12, 13 demonstrating the property of *single-node confluence*
 521 for the proof of local confluence, Proposition 9.

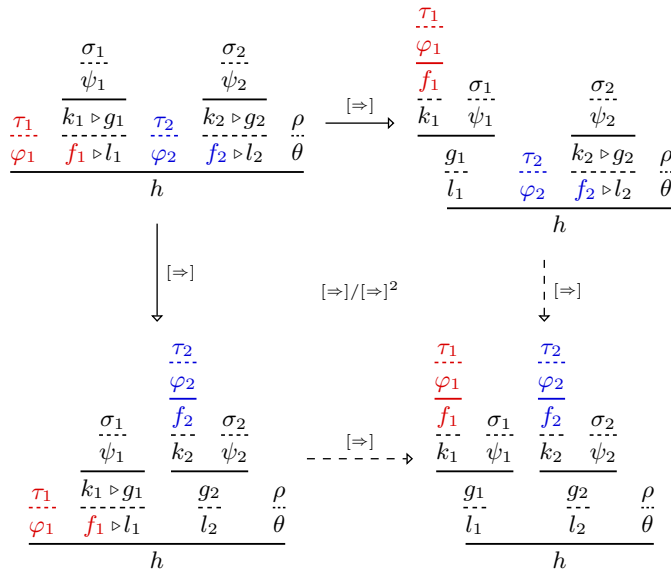


■ **Figure 11** Single-node confluence (1)

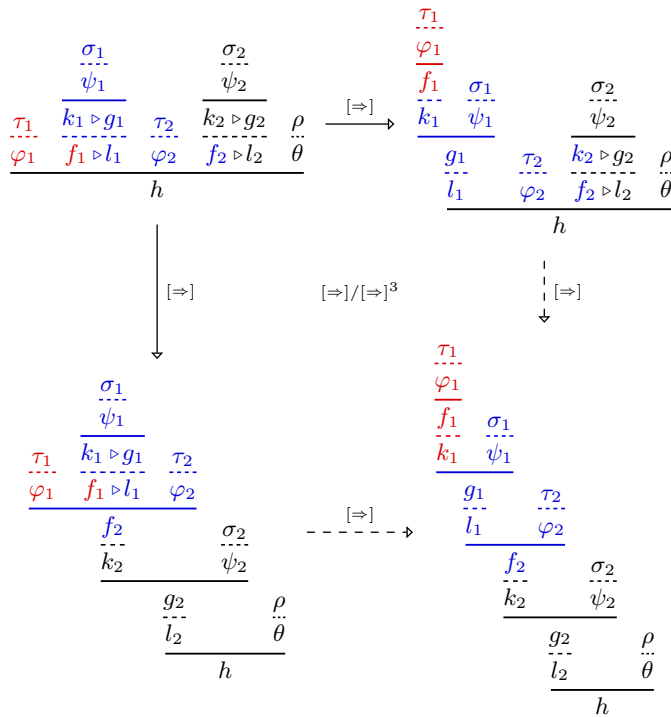
522 **C Strong normalization**

523 **C.1 Annotated reduction**

524 While local confluence holds, the interaction of symmetry and associativity with the interchange
 525 gives rise to some intricacy, such as in the example below. Each different term rendering of an



■ Figure 12 Single-node confluence (2)



■ Figure 13 Single-node confluence (3)

XX:20 Normalization Without Syntax

526 antecedent port k gives rise to a different reduction sequence of $[\wedge]$, $[c]$, and $[w]$ steps (and
527 $[\wedge L]$).

$$528 \quad [k_1, l_1] + [k_2, l_2] + [k_3, l_3] = [[k_1 + k_2, l_2] + \emptyset, l_1 + \emptyset + [k_3, l_3]]$$

529 More concretely, without the condition that k, l be non-empty, the reduction step $[c]$ would
530 create an infinite reduction through the equation $k = [k, \emptyset]$. While this particular case is
531 easily ruled out, the equation still rears its head in the third diagram in Figure 9, whose
532 equation is justified by:

$$533 \quad [k_1, k_2] + l = [k_1, k_2] + [l, \emptyset] = [k_1 + l, k_2 + \emptyset]$$

534 These equivalences, and in particular the diagram $[\wedge]/[c]^2$, create a problem for proving strong
535 normalization (SN). For our proof, we would like to show that infinite reductions are preserved
536 under non-deleting steps; but this becomes problematic if the commutation of reductions
537 introduces a $[w]$ step. Clearly, the weakening step in the diagram $[\wedge]/[c]^2$ should be harmless,
538 since it reduces a tree that was just duplicated, and any infinite reduction that might be
539 removed can be simulated in the remaining tree. Formalizing this, however, is delicate.

540 Our solution is to amend reduction to always introduce the maximal number of copies of
541 weakened subtrees, so that it is confluent without introducing $[w]$ steps. To map back onto
542 regular reduction, the superfluous copies may safely be removed. In this way, we separate the
543 idea that duplicate weakened subtrees may be removed, from other issues.

544 To obtain the maximal number of copies of weakened subtrees, we annotate a port k with
545 the “missing” number of copies i , as k^i . Let the **width** $\#k$ of a port k be the largest n for
546 which $k = [k_1, \dots, k_n]$ with $k_i \neq \emptyset$, where $\#\emptyset = 0$. That is, $\#k$ is the number of connected
547 components in the source graph $\mathcal{G} = \mathcal{G}_1 + \dots + \mathcal{G}_n$ of the arena morphism $k: \mathcal{G} \rightarrow \llbracket A \rrbracket$.

548 A port $k + \emptyset :: A \wedge B$ of width $\#(k + \emptyset) = \#k = n$, connected to a subtree $t + s$, may
549 generate a maximum of n weakened copies of s , by preferring the reduction $[c]$ over $[\wedge]$, and a
550 minimum of one the other way around. To obtain the same number of copies, we annotate
551 the reduction rules according to the following equations (with base cases below).

$$552 \quad (k+l)^i = k^{i+\#l} + l^{i+\#k} \quad [k, l]^{i+j} = [k^i, l^j]$$

$$\emptyset^1 = \emptyset \quad 1^0 = 1 \quad (f \triangleright k)^0 = f \triangleright k \quad (f, k \neq \emptyset)$$

553 For instance, if $\#k = \#l = 1$ and $k, l \neq \emptyset$ then we have:

$$554 \quad [k + \emptyset, l + \emptyset] = [k^0 + \emptyset^1, l^0 + \emptyset^1]$$

$$555 \quad = [(k + \emptyset)^0, (l + \emptyset)^0]$$

$$556 \quad = [k + \emptyset, l + \emptyset]^0$$

$$557 \quad = ([k, l] + [\emptyset, \emptyset])^0$$

$$558 \quad = [k, l]^0 + [\emptyset, \emptyset]^2$$

$$559 \quad = [k^0, l^0] + [\emptyset^1, \emptyset^1] = [k, l] + [\emptyset, \emptyset]$$

561 But note that $[\emptyset, \emptyset] = \emptyset$ is not derivable with annotations. While we can't restrict the
562 equations themselves in this way, we *can* restrict the reduction rules accordingly.

563 **► Definition 19. Annotated reduction** $\rightarrow_{\#}$ replaces the reduction steps $[\wedge]$, $[\Rightarrow]$, and $[c]$ of
564 Figure 6 by those in SonFigure 14. Hybrid rules $[\wedge L]$ and $[\Rightarrow R]$ are adapted analogously to $[\wedge]$
565 and $[\Rightarrow]$ respectively. Annotated rules apply to un-annotated ports by an initial annotation
566 with $i = j = 0$, while the rules $[1]$, $[\Rightarrow]$, and $[w]$ apply through the base-case equations.

567 The main point of annotated reduction is to obtain the local confluence diagram $[\wedge]/[c]$
568 in Figure 15. Next, we show that regular and annotated reduction are interchangeable for
569 proving strong normalization.

$$\begin{array}{c}
\frac{s+t}{(k+l)^i} \xrightarrow{[\wedge]} \# \frac{s}{k^{i+\#l}} \frac{t}{l^{i+\#k}} \\
\frac{s}{[k,l]^{i+j}} \xrightarrow{[c]} \# \frac{s}{k^i} \frac{s}{l^j} \quad \begin{array}{l} (k=\emptyset \Rightarrow i \geq 1) \\ (l=\emptyset \Rightarrow j \geq 1) \end{array} \\
\frac{\frac{\frac{\tau}{\varphi} \quad \frac{\frac{\sigma}{\psi} \quad (k_1+k_2)^i \triangleright g}{(f_1+f_2) \triangleright l} \quad \frac{\rho}{\theta}}{h}}{\frac{[\Rightarrow]}{\#} \frac{\frac{\tau}{\varphi} \quad \frac{\frac{\sigma}{\psi} \quad f_1}{k_1^{i+\#k_2}} \quad \frac{\rho}{\theta}}{k_2^{i+\#k_1} \triangleright g} \quad \frac{\rho}{\theta}}{f_2 \triangleright l} \quad \frac{\rho}{\theta}}{h}}
\end{array}$$

(ϕ, f₁ || f₂, l, θ, h)
(g ≠ k' ▷ g')

■ **Figure 14** Annotated reduction rules

$$\begin{array}{c}
\frac{t+s}{[k_1+l_1, k_2+l_2]^{i+j}} \xrightarrow{[\wedge]} \frac{t}{[k_1, k_2]^{n_1+n_2}} \frac{s}{[l_1, l_2]^{m_1+m_2}} \\
\downarrow [c] \quad \quad \quad [\wedge]/[c] \quad \quad \quad \downarrow [c] \\
\frac{t+s}{(k_1+l_1)^i} \frac{t+s}{(k_2+l_2)^j} \xrightarrow{[\wedge]} \frac{t}{k_1^{n_1}} \frac{t}{k_2^{n_2}} \frac{s}{l_1^{m_1}} \frac{s}{l_2^{m_2}} \\
n_1 = i + \#l_1 \quad n_2 = j + \#l_2 \quad m_1 = i + \#k_1 \quad m_2 = j + \#k_2
\end{array}$$

■ **Figure 15** Confluence for annotated reduction

570 ▶ **Lemma 20.** *A tree has an infinite \rightarrow reduction if and only if it has an infinite $\rightarrow_{\#}$*
571 *reduction.*

572 **Proof.** From left to right, a \rightarrow reduction can be simulated by a $\rightarrow_{\#}$ reduction by inserting
573 $[w]$ steps. From right to left, any reduction step $t \rightarrow_{\#} r$ has an equivalent in \rightarrow except the
574 following.

$$575 \quad \frac{s}{[k, \emptyset]^{i+j}} \xrightarrow{[c]} \# \frac{s}{k^i} \frac{s}{\emptyset^j}$$

576 Then if r , containing the duplicate subtrees s as above right, has an infinite reduction inside s ,
577 so does t . Otherwise the weakened s may be removed from r in a $[w]$ step, while the remaining
578 reduction from r is still infinite. The infinite $\rightarrow_{\#}$ reduction may then be simulated by \rightarrow . ◀

579 C.2 Auxiliary reduction

580 We separate reduction into *implicative* $[\Rightarrow]$ and *auxiliary* $[\neq] = [1, \wedge, c, w, \star, \wedge R, \wedge L, \Rightarrow R, \Rightarrow L]$.

581 ▶ **Lemma 21.** *Auxiliary reduction $[\neq]$ is strongly normalizing and confluent.*

582 **Proof.** For SN, observe that 1) all $[\neq]$ -reduction steps preserve or reduce the depth of a
583 combinatorial tree, and 2) the only step that increases the size of the tree is $[c]$, which
584 duplicates a subtree but reduces the node it is attached to. The following measure then
585 strictly decreases.

586 Let the the **depth** of a node be its longest path to a leaf, and its **size** the sum number of
587 steps in every way of sequentializing it as an ICP. We measure a node as the ordered pair of
588 its depth and its size, and a tree as the multiset over its nodes (ordered in the standard
589 multiset ordering). A normalization step reduces a node's size, and may only duplicate
590 nodes of smaller depth, so that the overall measure reduces. (Note that the argument works

XX:22 Normalization Without Syntax

interchangeably for regular and annotated reduction, using the regular or annotated hybrid rules $[\wedge L, \Rightarrow R]$ for sequentialization.)

Confluence then follows from local confluence, which is by the convergence of the critical pairs in Figures 9 and 15. \blacktriangleleft

We identify a class of **safe** reduction steps, which are non-deleting, and guaranteed to reflect SN (if $t \rightarrow s$ and $s \in \text{SN}$, then $t \in \text{SN}$).

► **Definition 22.** A reduction step is **safe**, denoted as $t \rightarrow_s s$, if it is not a weakening step $[w]$, or if it is a weakening step where the deleted subtree is SN.

$$\frac{\frac{\tau \quad \dots r \quad \dots}{\varphi \quad \emptyset :: A}}{f :: B} \xrightarrow[r \in \text{SN}]{[w]} \frac{\tau}{f :: B}$$

► **Lemma 23** (Safe reduction reflects SN). If $t \rightarrow_s s$ and $s \in \text{SN}$ then $t \in \text{SN}$.

Proof. We assume an infinite reduction in t and find a corresponding one in s to reach a contradiction. Using Lemma 20, we will work with $\rightarrow_{\#}$ instead of \rightarrow . We will discuss each normalization rule $[1, \wedge, \Rightarrow, c, w]$; for simplicity we will ignore the hybrid rules $[\star, \wedge R, \wedge L, \Rightarrow R, \Rightarrow L]$ as they behave similarly to their counterparts $[1, \wedge, \Rightarrow]$.

• $[w]$ Consider the reduction step $t \rightarrow_s s$ below.

$$\frac{\frac{\tau \quad r}{\varphi \quad \emptyset}}{f} \xrightarrow[r \in \text{SN}]{[w]} \frac{\tau}{f}$$

The only reduction step in t where r and s interact is the above, which deletes r . Then if t has an infinite reduction, it must either have infinitely many steps in r , a contradiction, or infinitely many steps not in r , in which case s has an infinite reduction, a contradiction.

• $[1]$ Consider a rewrite sequence of $[1]$ steps $t \twoheadrightarrow s$. An infinite reduction from t must contain infinitely many steps in $[\wedge, \Rightarrow, c, w]$, since a sequence of $[1]$ (and auxiliary) steps will strictly shrink the tree t , and so must be finite. We map the infinite reduction from t onto one from s along the $[1]$ reduction $t \twoheadrightarrow s$, as follows. A $[1]$ -sequence commutes with individual $[\wedge, \Rightarrow, c, w]$ steps as below left. Note that the length of the $[1]$ sequence $t' \twoheadrightarrow s'$ may be longer or shorter than $t \twoheadrightarrow s$ when commuting with $[c]$ respectively $[w]$ steps. The sequence $t \twoheadrightarrow s$ commutes with $[1]$ steps as below right, where the $[1]$ reduction $s \twoheadrightarrow s'$ consists of zero steps if the step $t \rightarrow t'$ is absorbed in $t \twoheadrightarrow s$ (i.e. the same step occurs in $t \twoheadrightarrow s$), and one step otherwise.

$$\begin{array}{ccc} t \xrightarrow{[1]} s & & t \xrightarrow{[1]} s \\ [\wedge, \Rightarrow, c, w] \downarrow & \downarrow [\wedge, \Rightarrow, c, w] & [1] \downarrow & \downarrow [1] \\ t' \dashrightarrow s' & & t' \dashrightarrow s' \end{array}$$

The above diagrams then map the infinite reduction from t , containing infinitely many $[\wedge, \Rightarrow, c, w]$ steps, onto one from s , a contradiction.

• $[\wedge, \Rightarrow]$ Consider a $[\wedge, \Rightarrow]$ reduction $t \twoheadrightarrow_{\#} s$. By the confluence diagrams $[\wedge]/[\wedge]$ and $[\Rightarrow]/[\Rightarrow]^1$ through $[\Rightarrow]/[\Rightarrow]^3$ in Figures 9 and 11–13 in the appendix it commutes with $[1, \wedge, \Rightarrow, w]$ steps as below.

$$\begin{array}{ccc} t \xrightarrow{[\wedge, \Rightarrow]} s & & \\ [1, \wedge, \wedge, w] \downarrow & \downarrow [1, \wedge, \wedge, w] & \\ t' \dashrightarrow s' & & \end{array}$$

626 Here, the reduction $s \rightarrow_{\#} s'$ is a single step, unless the step $t \rightarrow_{\#} t'$ is absorbed in (i.e. the
 627 same step occurs in) the reduction $t \rightarrow_{\#} s$; then $s = s'$.

628 Next, the reduction $t \rightarrow_{\#} s$ commutes with a $[c]$ step as below, by the diagram $[\wedge]/[c]$ in
 629 Figure 15. The reduction $s \rightarrow_{\#} s'$ contains at least one step.

$$\begin{array}{ccc} t & \xrightarrow{[\wedge, \Rightarrow]} & s \\ [c] \downarrow & & \downarrow [c] \\ t' & \xrightarrow{[\wedge, \Rightarrow]} & s' \end{array}$$

630

631 Then since the $[\wedge, \Rightarrow]$ reduction $t \rightarrow_{\#} s$ may absorb only a finite number of consecutive $[\Rightarrow, \wedge]$
 632 steps, it maps an infinite reduction from t onto an infinite reduction from s , a contradiction.

633 • $[c]$ Consider a $[\wedge, c]$ reduction $t \rightarrow_{\#} s$. It commutes with $[1, \Rightarrow, w]$ steps as below left,
 634 where $s \rightarrow_{\#} s'$ contains at least one step. By the diagrams $[\wedge]/[\wedge]$ and $[\wedge]/[c]$ the relation
 635 $[\wedge, c]$ commutes with itself as below right (by Lemma 21 these auxiliary reductions are finite).

$$\begin{array}{ccc} t & \xrightarrow{[\wedge, c]} & s \\ [1, \Rightarrow, w] \downarrow & & \downarrow [1, \Rightarrow, w] \\ t' & \xrightarrow{[\wedge, c]} & s' \end{array} \quad \begin{array}{ccc} t & \xrightarrow{[\wedge, c]} & s \\ [\wedge, c] \downarrow & & \downarrow [\wedge, c] \\ t' & \xrightarrow{[\wedge, c]} & s' \end{array}$$

636

637 Since $[\wedge, c]$ reduction is SN, the infinite reduction from t contains an infinite number of
 638 $[1, \Rightarrow, w]$ steps. Then the corresponding reduction from s is infinite, a contradiction. ◀

639 C.3 Reducibility

640 We complete the strong normalization proof by abstract reducibility. The **reducibility set**
 641 $\|A\|$ of a formula A is the set of combinatorial trees defined as follows.

$$\begin{aligned} \|P\| &= \text{SN} \\ \|A \Rightarrow B\| &= \{\star :: A \Rightarrow B\} \cup \left\{ \frac{\tau}{k \triangleright f :: A \Rightarrow B} \mid \forall s \in \|A\|. \frac{\tau \quad \dots s}{f :: B} \in \|B\| \right\} \\ \|A \wedge B\| &= \{\star :: A \wedge B\} \cup \{t :: A \wedge B \mid \exists t_1 \in \|A\|. \exists t_2 \in \|B\|. t \rightarrow_s t_1 + t_2\} \end{aligned}$$

643 We write $\tau \in \|\Gamma\|$ if $\tau :: \Gamma = t_1 :: A_1, \dots, t_n :: A_n$ and $t_i \in \|A_i\|$ for all $i \leq n$. We establish the
 644 standard lemmata.

645 ▶ **Lemma 24.** $\|A\| \subseteq \text{SN}$.

646 **Proof of Lemma 24.** By induction on A . The case $\|P\|$ is immediate. For $\|A \Rightarrow B\|$, let t be
 647 the following tree.

$$648 \quad t = \frac{\tau}{k \triangleright f :: A \Rightarrow B} \in \|A \Rightarrow B\|$$

649 Observe that $\star \in \|A\|$. By definition of $\|A \Rightarrow B\|$ then

$$650 \quad t' = \frac{\tau \quad \dots \star}{f :: B} \in \|B\| .$$

651 By the inductive hypothesis, $t' \in \text{SN}$. Since the $[\Rightarrow]$ -reduction step $t \rightarrow_s t'$ is safe, then
 652 $t \in \text{SN}$ by Lemma 23.

653 For $\|A \wedge B\|$, let $t \rightarrow_s t_1 + t_2$ with $t_1 \in \|A\|$ and $t_2 \in \|B\|$. By the inductive hypothesis
 654 $t_1, t_2 \in \text{SN}$. Then $t_1 + t_2 \in \text{SN}$, and $t \in \text{SN}$ by Lemma 23. ◀

XX:24 Normalization Without Syntax

655 ▶ **Lemma 25.** *If $t_1 :: A \rightarrow_s t_2 :: A$ and $t_2 \in \|A\|$ then $t_1 \in \|A\|$.*

656 **Proof of Lemma 25.** By induction on A . The case $\|P\|$ is Lemma 23. For $\|A \Rightarrow B\|$, let

$$657 \quad t_1 = \frac{\frac{\tau_1}{\varphi_1}}{k \triangleright f :: A \Rightarrow B} \xrightarrow{[x]} \frac{\frac{\tau_2}{\varphi_2}}{k \triangleright f :: A \Rightarrow B} = t_2 \in \|A \Rightarrow B\| .$$

658 For any $s \in \|A\|$ we get the corresponding reduction step

$$659 \quad t_3 = \frac{\frac{\tau_1}{\varphi_1} \quad \frac{s}{k :: A}}{f :: B} \xrightarrow{[x]} \frac{\frac{\tau_2}{\varphi_2} \quad \frac{s}{k :: A}}{f :: B} = t_4 .$$

660 By definition of $\|A \Rightarrow B\|$ we have $t_4 \in \|B\|$, by the inductive hypothesis we have $t_3 \in \|B\|$,
661 and again by definition of $\|A \Rightarrow B\|$ we have $t_1 \in \|A \Rightarrow B\|$.

662 For $t_2 \in \|A \wedge B\|$, there are $t_3 \in \|A\|$ and $t_4 \in \|B\|$ such that we have the following
663 reductions.

$$664 \quad t_1 \rightarrow_s t_2 \rightarrow_s t_3 + t_4$$

665 Then $t_1 \in \|A \wedge B\|$ by definition of $\|A \wedge B\|$. ◀

666 ▶ **Lemma 26.** *For any tree t for a sequent $\Gamma \vdash B$, we have*

$$667 \quad \forall \sigma \in \|\Gamma\| . \frac{\sigma}{t} \in \|B\| .$$

668 **Proof of Lemma 26.** By induction on the construction of t . We cover a selection of cases,
669 relegating the others to the appendix.

670 ■ $t = \star$

671 We need to show $s \in \|A\|$ for any $s \in \|A\|$, which is immediate.

$$672 \quad \text{■ } t = \frac{\frac{\star :: \Gamma_1}{t_1 :: A} \quad \star :: \Gamma_2}{t_2 :: B}$$

673 We need to show for t' as below left that $t' \in \|B\|$ for any $\sigma_1 \in \|\Gamma_1\|$ and $\sigma_2 \in \|\Gamma_2\|$.

$$674 \quad t' = \frac{\frac{\sigma_1}{t_1 :: A} \quad \sigma_2}{t_2 :: B} \quad t'_1 = \frac{\sigma_1}{t_1 :: A}$$

675 We apply the inductive hypothesis twice, first on t_1 to get $t'_1 \in \|A\|$ with t'_1 above right,
676 and then on t_2 to get $t' \in \|B\|$.

$$677 \quad \text{■ } t = \frac{\frac{\star}{\varphi :: \Gamma}}{k \triangleright f :: A \Rightarrow B}$$

678 We need to show for t' as below left that $t' \in \|A \Rightarrow B\|$ for any $\sigma \in \|\Gamma\|$. We apply the
679 inductive hypothesis to t_1 , below centre, which gives that $t'_1 \in \|B\|$ for any $r \in \|A\|$ for t'_1
680 as below right.

$$681 \quad t' = \frac{\frac{\sigma}{\varphi :: \Gamma}}{k \triangleright f :: A \Rightarrow B} \quad t_1 = \frac{\frac{\star}{\varphi :: \Gamma} \quad \frac{\star}{k :: A}}{f :: B} \quad t'_1 = \frac{\frac{\sigma}{\varphi :: \Gamma} \quad \frac{r}{k :: A}}{f :: B}$$

682 By the definition of $\|A \Rightarrow B\|$ then $t' \in \|A \Rightarrow B\|$.

$$683 \quad t = \frac{\frac{\frac{\star}{\varphi::\Gamma} \quad \frac{\star}{f \triangleright l::A \Rightarrow B} \quad \frac{\star}{\theta::\Delta}}{h::C}}{(\varphi, f \parallel l, \psi, h)}$$

684 We need to show for t' as below left that $t' \in \llbracket C \rrbracket$, for any $\tau \in \llbracket \Gamma \rrbracket$, $s \in \llbracket A \Rightarrow B \rrbracket$, and
 685 $\rho \in \llbracket \Delta \rrbracket$. Let s be as below right.

$$686 \quad t' = \frac{\frac{\tau}{\varphi} \quad \frac{s}{f \triangleright l::A \Rightarrow B} \quad \frac{\rho}{\theta}}{h::C} \quad s = \frac{\frac{\sigma}{\psi}}{k \triangleright g::A \Rightarrow B}$$

687 We will apply the inductive hypothesis to t_1 and t_2 as given below.

$$688 \quad t_1 = \frac{\frac{\tau}{\varphi::\Gamma}}{f::A} \quad t_2 = \frac{\frac{l::B \quad \theta::\Delta}{h::C}}$$

689 The induction hypothesis for t_1 gives us $t'_1 \in \llbracket A \rrbracket$ as below left. By definition of $\llbracket A \Rightarrow B \rrbracket$
 690 we then get $s' \in \llbracket B \rrbracket$ as below centre. Then induction hypothesis for t_2 gives us $t'_2 \in \llbracket C \rrbracket$
 691 as below right.

$$692 \quad t'_1 = \frac{\frac{\tau}{\varphi}}{f::A} \quad s' = \frac{\frac{t'_1}{k::A} \quad \frac{\sigma}{\psi}}{g::B} \quad t'_2 = \frac{\frac{s'}{l::B} \quad \frac{\rho}{\theta}}{h::C}$$

693 By Lemma 25 and the below reduction $t' \rightarrow t'_2$ then $t' \in \llbracket C \rrbracket$.

$$694 \quad t' = \frac{\frac{\frac{\tau}{\varphi} \quad \frac{\frac{\frac{\sigma}{\psi}}{k \triangleright g::A \Rightarrow B} \quad \frac{\rho}{\theta}}{h::C}}{f \triangleright l::A \Rightarrow B}}{h::C} \xrightarrow{[\Rightarrow]} \frac{\frac{\frac{\frac{\sigma}{\psi}}{f::A} \quad \frac{\sigma}{\psi}}{k::A} \quad \frac{\rho}{\theta}}{g::B} \quad \frac{\rho}{\theta}}{l::B} \quad \frac{\rho}{\theta}}{h::C} = t'_2$$

$$695 \quad t = \frac{\frac{\star}{1::P}}{1::P}$$

696 We need to show for t' as below that $t' \in \llbracket P \rrbracket$ for any $s \in \llbracket P \rrbracket$.

$$697 \quad t' = \frac{\frac{s}{1::P}}{1::P} \xrightarrow{[1]} s$$

698 Since $t' \rightarrow s$ as above, we get $t' \in \llbracket P \rrbracket$ by Lemma 25.

$$699 \quad t = \frac{\frac{\frac{\star}{\varphi::\Gamma} \quad \frac{\star}{\psi::\Delta}}{f+g::A \wedge B}}{(f, \varphi \parallel g, \psi)}$$

700 We need to show that $t' \in \llbracket A \wedge B \rrbracket$ for t' as below left, for any $\tau \in \llbracket \Gamma \rrbracket$ and $\sigma \in \llbracket \Delta \rrbracket$. We
 701 apply the inductive hypothesis to t_1 and t_2 , as below right,

$$702 \quad t' = \frac{\frac{\tau}{\varphi::\Gamma} \quad \frac{\sigma}{\psi::\Delta}}{f+g::A \wedge B} \quad t_1 = \frac{\frac{\star}{\varphi}}{f::A} \quad t_2 = \frac{\frac{\star}{\psi}}{f::B}$$

703 which gives $t'_1 \in \llbracket A \rrbracket$ and $t'_2 \in \llbracket B \rrbracket$, as below.

$$704 \quad t'_1 = \frac{\frac{\tau}{\varphi}}{f::A} \in \llbracket A \rrbracket \quad t'_2 = \frac{\frac{\sigma}{\psi}}{f::B} \in \llbracket B \rrbracket$$

705 Since $t' = t'_1 + t'_2$, we have $t' \rightarrow_s t'_1 + t'_2$ by the empty reduction, and by definition
 706 $t' \in \llbracket A \wedge B \rrbracket$.

XX:26 Normalization Without Syntax

$$707 \quad \dashv\!\!\dashv t = \frac{\frac{\overset{\star}{k+l} :: A \wedge B \quad \overset{\star}{\theta} :: \Gamma}{h :: C}}$$

708 We need to show for t' as below left that $t' \in \llbracket C \rrbracket$ for any $\rho \in \llbracket \Gamma \rrbracket$ and $s \in \llbracket A \wedge B \rrbracket$. By
709 definition of $\llbracket A \wedge B \rrbracket$ there is a reduction as below right, with $s_1 \in \llbracket A \rrbracket$ and $s_2 \in \llbracket B \rrbracket$.

$$710 \quad t' = \frac{\frac{\overset{s}{k+l} :: A \wedge B \quad \overset{\rho}{\theta}}{h :: C}}{s \rightarrow_s s_1 + s_2 = \frac{\frac{\overset{\tau}{\varphi} \quad \overset{\sigma}{\psi}}{f+g :: A \wedge B}}$$

711 This gives the reduction below.

$$712 \quad t' \rightarrow_s \frac{\frac{\frac{\overset{\tau}{\varphi} \quad \overset{\sigma}{\psi}}{f+g :: A \wedge B} \quad \overset{\rho}{\theta}}{h :: C}}{\frac{\frac{\overset{\tau}{f} :: A \quad \overset{\sigma}{g} :: B}{k :: A \quad l :: B} \quad \overset{\rho}{\theta}}{h :: C}} \xrightarrow{[\wedge]} \frac{\frac{\overset{\tau}{f} :: A \quad \overset{\sigma}{g} :: B}{k :: A \quad l :: B} \quad \overset{\rho}{\theta}}{h :: C} = t'_1 .$$

713 The inductive hypothesis for t_1 below (with $\rho \in \llbracket \Gamma \rrbracket$, $s_1 \in \llbracket A \rrbracket$, $s_2 \in \llbracket B \rrbracket$), gives $t'_1 \in \llbracket C \rrbracket$
714 . Then $t' \in \llbracket C \rrbracket$ by the above reduction and Lemma 25.

$$715 \quad t_1 = \frac{\frac{\overset{\star}{k} :: A \quad \overset{\star}{l} :: B \quad \overset{\star}{\theta} :: \Gamma}{h :: C}}$$

$$716 \quad \dashv\!\!\dashv t = \frac{\frac{\overset{\star}{\varphi} :: \Gamma \quad \overset{\star}{[k,l]} :: A}{f :: B}}{(k, l \neq \emptyset)}$$

717 We need to show that $t' \in \llbracket C \rrbracket$ for t' as in the reduction step below

$$718 \quad t' = \frac{\frac{\overset{\tau}{\varphi} \quad \overset{s}{[k,l]} :: A}{f :: B}}{\xrightarrow{[c]} \frac{\frac{\overset{\tau}{\varphi} \quad \overset{s}{k} :: A \quad \overset{s}{l} :: A}{f :: B}} = t'_1$$

719 for any $\tau \in \llbracket \Gamma \rrbracket$ and $s \in \llbracket A \rrbracket$. The inductive hypothesis on t_1 below gives $t'_1 \in \llbracket C \rrbracket$, and
720 by Lemma 25 then $t' \in \llbracket C \rrbracket$.

$$721 \quad t_1 = \frac{\frac{\overset{\star}{\varphi} \quad \overset{\star}{k} :: A \quad \overset{\star}{l} :: A}{f :: B}}$$

$$722 \quad \dashv\!\!\dashv t = \frac{\frac{\overset{\star}{\varphi} :: \Gamma \quad \overset{\star}{\emptyset} :: A}{f :: B}}$$

723 We need to show that $t' \in \llbracket C \rrbracket$ for t' as in the reduction step below left, for any $\tau \in \llbracket \Gamma \rrbracket$
724 and $s \in \llbracket A \rrbracket$. The inductive hypothesis on t_1 (below right) gives $t'_1 \in \llbracket C \rrbracket$. Next, $s \in \text{SN}$
725 by Lemma 24, so that the reduction step is safe. Then $t' \in \llbracket C \rrbracket$ by Lemma 25.

$$726 \quad t' = \frac{\frac{\overset{\tau}{\varphi} \quad \overset{s}{\emptyset} :: A}{f :: B}}{\xrightarrow{[w]} \frac{\overset{\tau}{\varphi}}{f :: B}} = t'_1 \quad t_1 = \frac{\overset{\star}{\varphi}}{f :: B}$$

728 **► Theorem 10 (restatement).** *Combinatorial-tree reduction is strongly normalizing.*

729 **Proof.** Let t be an arbitrary combinatorial tree for $\Gamma \vdash B$. Note that $\overset{\star}{\cdot} :: \Gamma$ is in $\llbracket \Gamma \rrbracket$. Then
730 $t = \frac{\overset{\star}{\cdot} :: \Gamma}{t} \in \llbracket B \rrbracket$ by Lemma 26, and $t \in \text{SN}$ by Lemma 24. ◀