UNIVERSITY OF
BATH

Citation for published version:
Heijltjes, W & Hughes, D 2015, Complexity bounds for sum-product logic via additive proof nets and Petri nets. in Proceedings of the 30th ACM/IEEE Symposium on Logic in Computer Science (LICS), 2015. IEEE, pp. 80-91, 30th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), 2015 , Kyoto, Japan, 6/07/15. https://doi.org/10.1109/LICS.2015.18

DOI:
10.1109/LICS.2015.18

Publication date:
2015

Document Version
Early version, also known as pre-print

Link to publication

University of Bath

Complexity Bounds for Sum-Product Logic
via Additive Proof Nets and Petri Nets

Willem Heijltjes & Dominic Hughes

Abstract—We investigate efficient algorithms for the additive fragment of linear logic. This logic is an internal language for categories with finite sums and products, and describes concurrent two-player games of finite choice. In the context of session types, typing disciplines for communication along channels, the logic describes the communication of finite choice along a single channel.

We give a simple linear time correctness criterion for unit-free propositional additive proof nets via a natural construction on Petri nets. This is an essential ingredient to linear time complexity of the second author's *combinatorial proofs* for classical logic.

For full propositional additive linear logic, including the units, we give a proof search algorithm that is linear-time in the product of the source and target formula, and an algorithm for proof net correctness that is of the same time complexity. We prove that proof search in first-order additive linear logic is NP-complete.

I. Introduction

Additive proof nets, as formulated in [19], provide an intuitive yet completely formal diagrammatic presentation of canonical maps such as:
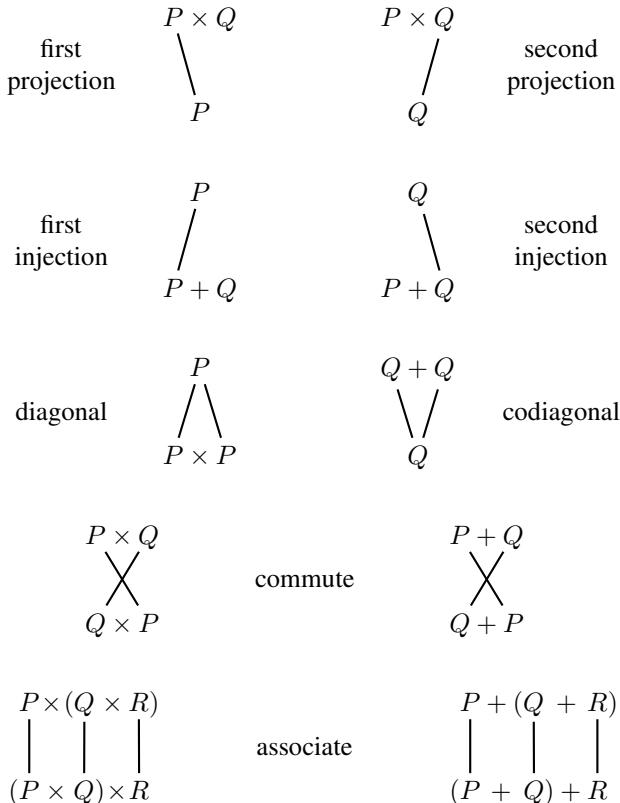
Composition is simply path composition (the standard composition of the underlying binary relations):
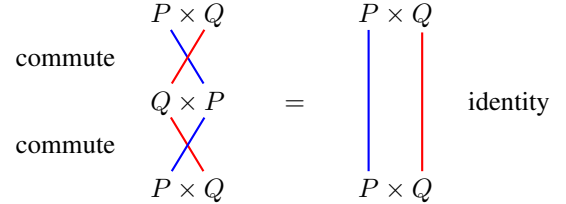

A proof net from a formula A to a formula B is a binary relation from the leaves (occurrences of atoms P, Q, R,...) of A to the leaves of B, satisfying the *resolution* condition [19], which is related to totality [24] and maximality [15].

In this paper we present a new correctness criterion for additive proof nets which can be verified in linear time (in the number of edges or *links* in the binary relation), via a natural construction on Petri nets (cartesian product). We illustrate the criterion on commutativity $P \times Q \to Q \times P$:

Since $P \times Q$ is a source, it is implicitly dualized. We unfold this duality and place the two formulas side by side:
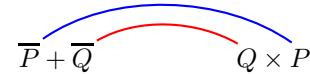
Each formula determines a Petri net. The Petri net of $\overline{P} + \overline{Q}$ has three places ◯ (one per symbol "$\overline{P}$", "$+$", "$\overline{Q}$") and two transitions ▪ :
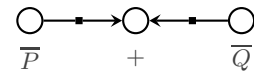
This Petri net $\mathcal{N}(\overline{P} + \overline{Q})$ captures the disjunctive essence of $+$ in the way it fires. Starting with a single token on $\overline{P}$,
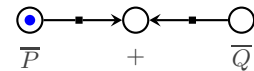
the Petri net can fire, moving the token to the central $+$:
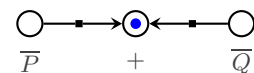
This firing sequence captures the logical disjunction rule "from $\overline{P}$ infer $\overline{P} + \overline{Q}$", The alternative firing sequence, starting with a token on the $\overline{Q}$ place,
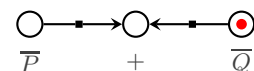
captures "from $\overline{Q}$ infer $\overline{P} + \overline{Q}$", as the token moves to the central $+$:
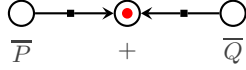
The Petri net $\mathcal{N}(Q \times P)$ has three places (one per symbol) and one transition:

Starting from two tokens, one on each of Q and P,

the firing results in a central token on the $\times$ place:

This corresponds to the logical conjunction rule "from Q and P infer $Q \times P$". Note that if we start with only one of the two tokens, the Petri net is deadlocked: we cannot infer $Q \times P$ ("Q AND P") from P alone, or from Q alone.

The next step in checking correctness is to build the cartesian product $\mathcal{N}(\overline{P} + \overline{Q}) \; * \; \mathcal{N}(Q \times P)$:
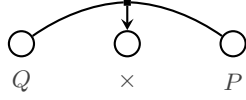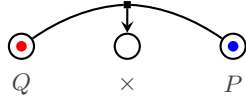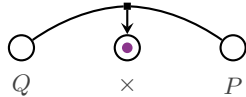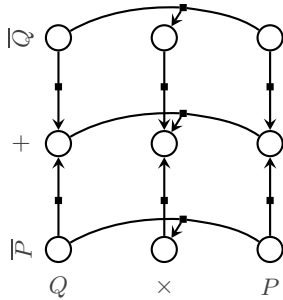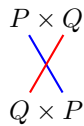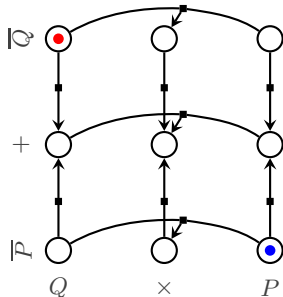

Each row is a copy of $\mathcal{N}(Q \times P)$ and each column a copy of $\mathcal{N}(\overline{P} + \overline{Q})$. Each link in the commutativity proof net

determines a token in the initial state:

The top-left token, in the "$\overline{Q}$ row" and "Q column", represents the link Q——Q, and the bottom-left token, in the "$\overline{P}$ row" and "P column", represents the link P——P. The proof net is correct if, upon exhaustively firing the Petri net, we end up with a single token in the center (and no other tokens).

An example firing sequence is shown in the left column of Fig. 1, from top to bottom. First the top-right token fires (downwards); then the bottom-left token fires (upwards); finally the pair in the middle row fires (horizontally). Because there is a central token, and no other tokens remaining on the Petri net, we have verified that commutativity is a proof net.

Since in practice we do not have to write out the entire grid of the cartesian product at the outset, the algorithm runs in linear time in the number of tokens (i.e., linear time in the number of links in the binary relation).

A. Relationship to Danos contractibility and top-down sequentialization

Danos' contractibility criterion for multiplicative linear logic [6] gradually produces a sequentialization by starting with the axiom links (as axiom rules) at the top then flowing downwards rule by rule until reaching the conclusion. Naively quadratic, Guerrini observed that the criterion can be checked in linear time [11]. Our Petri net condition is similar in spirit: the initial tokens (corresponding links) provide axiom rules at the top of a sequentialization; every Petri net firing yields a proof rule, top-down from the axiom rules.

Fig. 1 shows the how the sequentialization emerges from the firing sequence presented above. It begins with two axiom rules, corresponding to the two tokens in the initial state of the Petri net (in turn, corresponding to the two edges of the proof net).

When the first token fires, up the rightmost column, it corresponds to "from $\overline{P}$ infer $\overline{P} + \overline{Q}$" (as discussed earlier in the Introduction), and we write down the corresponding rule of two-sided additive linear logic (the internal logic of sum-product categories): "from $P \vdash A$ infer $P \times Q \vdash A$". When the second token fires, down the leftmost column, it corresponds to "from $\overline{Q}$ infer $\overline{P} + \overline{Q}$", and we write down the corresponding rule. When the final pair fires, along the central row, we write down the final $\times$-rule.

B. The coalescence criterion

As tokens fire (the first column of Fig. 1), or rules are written (the middle column), an alternative view is to shuffle links, as presented in the last column of Fig. 1. The first move shifts the source of the link P——P from P to the whole formula $P \times Q$, i.e., to the symbol $\times$, or equivalently, to the root vertex $\times$ of the parse tree of the formula:

Similarly, the second move (from the second to third row in the last column of Fig. 1) shifts the source of the link Q——Q from Q to the whole formula $P \times Q$. Finally, the two links
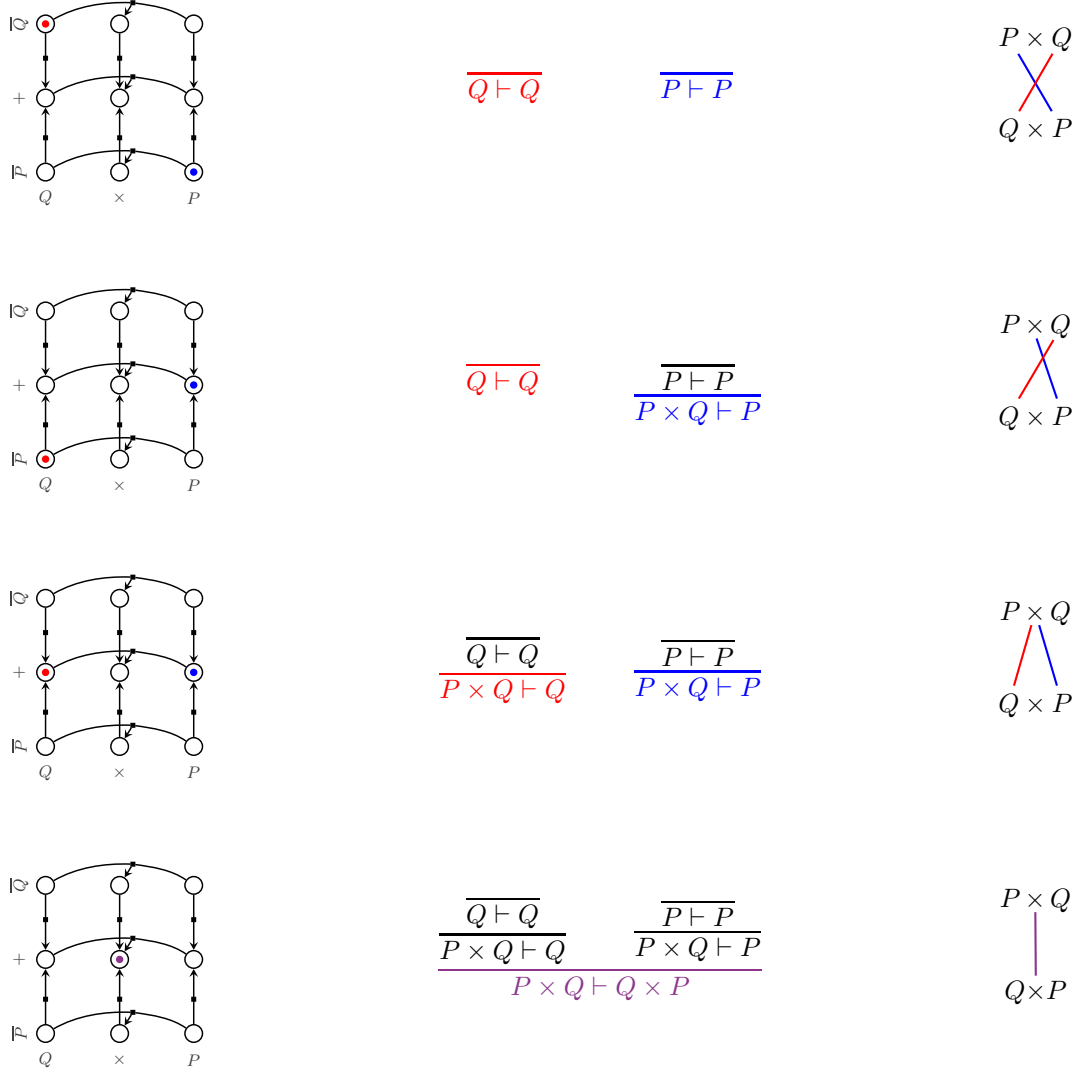
Fig. 1. A tight three-way correspondence: firing the cartesian product Petri net, sequentializing top-down, and coalescing.

from $P \times Q$, one to P and one to Q, coalesce into a single link from $P \times Q$ to $Q \times P$.

This forms the basis of the *coalescence criterion*: a proof net from A to B is correct if, upon carrying out moves such as those described above (each corresponding to a Petri net fire, or to writing down a proof rule), the final result is a single link from the root vertex of A to the root vertex of B.

Given the tight correspondence with additive top-down sequentialization, one may consider the coalescence criterion an additive analogue of Danos' multiplicative contractibility.

C. Grid notation

To save space, we abbreviate cartesian product Petri nets in a grid notation. For example, to verify associativity

$$P \times (Q \times R)$$
$$| \quad | \quad |$$
$$(P \times Q) \times R$$

we construct the cartesian product Petri net $\mathcal{N}(P \times (Q \times R)) * \mathcal{N}((P \times Q) \times R)$ initialized with three tokens, one per link:
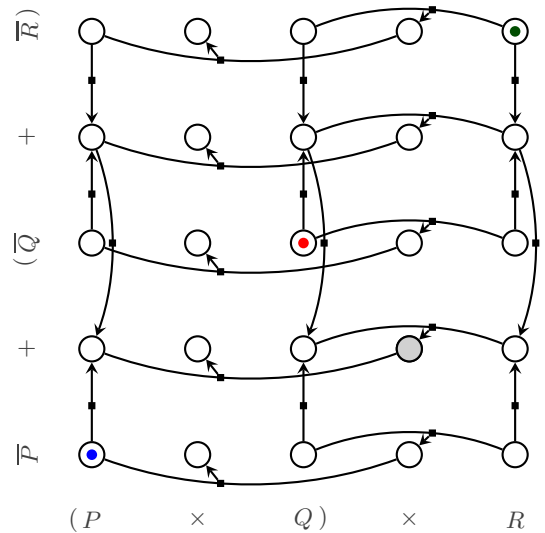
The goal place has been highlighted in grey. The reader may verify that, upon exhaustively firing the Petri net, a single token remains, on the goal (irrespective of firing order). We

abbreviate the Petri net to the following grid
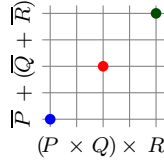
The parse trees of the two labelling formulas determine the legal firings. For example, the following firing is legal based on the structure of $(P \times Q) \times R$:
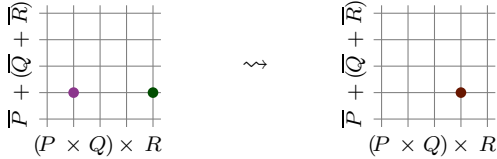
D. Efficient provability and proof search

Define *spawning* in a Petri net as the variant of firing obtained by leaving consumed tokens in place. Here is an example spawning sequence (in compact grid notation):
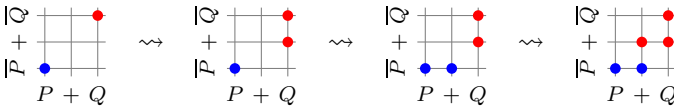
Spawning provides an efficient search for the provability of a sequent $A \vdash B$: initialize the Petri net $\mathcal{N}(A) * \mathcal{N}(B)$ with a token on every place which has dual atomic labels ("row $\overline{P}$ and column P"); spawn repeatedly; the sequent is provable iff we spawn onto the goal place. The above spawning sequence shows the probability of $P \times Q \vdash P + Q$. Upon reversing a successful spawning sequence one can (non-deterministically) extract a proof.

E. Units

The categorical initial object 0 and terminal object 1, the nullary coproduct and product, are characterised by unique initial maps $0 \to A$ and terminal maps $A \to 1$. These are naturally rendered in proof nets as follows.

$$
\begin{array}{ccc}
 & 0 & A \\
\text{initial} & | & | & \text{terminal} \\
\text{map} & & & \text{map} \\
 & A & 1
\end{array}
$$
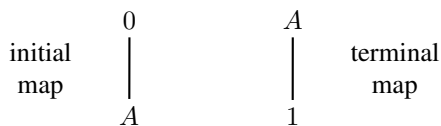
However, in this representation proof nets no longer correspond 1–1 to categorical maps, and the uniqueness property of initial and terminal maps forces an equivalence ($\Leftrightarrow$) on proof nets. For terminal maps it is generated by:
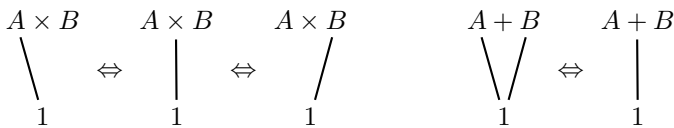
To decide whether two proof nets with units are equivalent is non-trivial, due to the interaction of initial maps and terminal maps via occurrences of the map $0 \to 1$, which is both; e.g.:
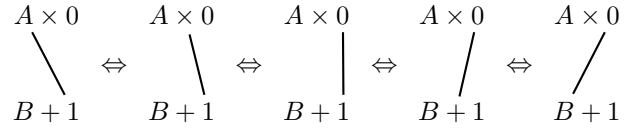
Proof net equivalence can be decided by *saturating* proof nets [12], [13]: instead of replacing one link by another in a rewrite step, both links are kept. For example:
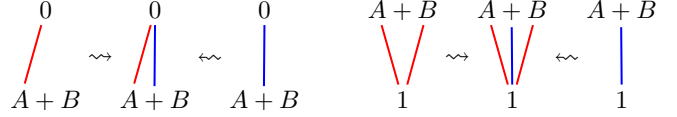
The resulting *saturated nets* correspond 1–1 to categorical maps. Below are two example saturated nets, representing the two morphisms from $0*(0+P)$ to $1+(1*P)$ (in grid notation, to manage the link density).
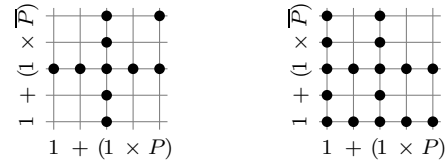
The main result we present for sum–product logic with units, in Section IV, is that correctness of a saturated net for a sequent $A \vdash B$ can be decided in time linear in $|A| \times |B|$.

F. Quantifiers

There is an interesting interaction between universal and existential quantifiers, even in the absence of any other logical connectives, or weakening and contraction [26]. We take a brief look at first-order additive linear logic, which combines the quantifiers with products and coproducts (but without the units). We show that, unlike in the propositional fragment, proof search is NP-hard.

G. Linear time combinatorial proofs for classical logic

Our result that an additive proof net can be verified in linear time implies that combinatorial proofs for propositional classical logic (*Proofs Without Syntax* [16]) can be verified in linear time. A combinatorial proof of Peirce's Law

$$\big((P \Rightarrow Q) \Rightarrow P\big) \Rightarrow P \quad = \quad \big((\overline{P} \vee Q) \wedge \overline{P}\big) \vee P$$

is shown below:
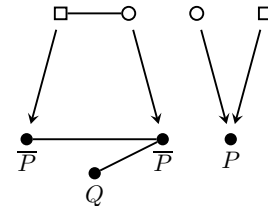
A combinatorial proof of A is a *skew fibration* (a particular kind of undirected graph homomorphism) from a coloured graph into the graph $\mathcal{G}(A)$ associated with A. In the example above, each graph has four vertices, and the upper graph has two colours. Such skew fibrations correspond [17] to homomorphisms which preserve maximal cliques, hence a skew fibration corresponds to a special kind of additive proof

net: one which is *functional* in the sense that every source leaf is in one and only one link. Our theorem in this paper that an additive proof net $\lambda : C \vdash A$ can be verified in linear time in $|\lambda|$ implies that a combinatorial proof, where $|\lambda|$ is linear in the size of C, can be verified in linear time. Work in progress to extend combinatorial proofs to first-order classical logic builds on the same linear time complexity result.

H. Related work

An early result is Whitman's Theorem [29] on free lattices. The theorem, which gives a factorisation of lattice inequalities, corresponds closely to cut-elimination in sum–product logic.

The advent of linear logic [9], where categorical products and coproducts are captured by the additive fragment, sparked a wave of syntactic and semantic approaches, to capture the multiplicatives and exponentials, and sometimes also the additives. Joyal generalised Whitman's Theorem to bicomplete categories [20], categories with all limits and colimits, and gave an interpretation of cut-elimination in game semantics [21]. A canonical treatment of the additives, without the units, was first given by Hu in terms of *contractible* coherence spaces [15]. Canonical proof nets are a fragment of the MALL proof nets of the second author and van Glabbeek [19].

In the presence of the units, proof equivalence was investigated first by Cockett and Seely [5], and later by Cockett and Santocanale [4]. The latter give an effective, intricate decision procedure, based on a careful analysis of the structure of finite sum–product categories. A canonical syntax for additive linear logic with units, *saturated nets*, was given by the first author [12], [13].

From the perspective of game semantics, the problem of canonically representing additive proofs surfaced as the problem that the games defined by Blass [3] were not associative [1]. The problem is one of concurrency: in games, a map from A to B interpreted as a parallel game on A and B, where the coproducts and products in $\overline{A}$ and B represent binary choice for Player and Opponent respectively. This problem was addressed in detail by Abramsky and Melliès [2].

A related interpretation of an additive sequent is as a protocol for concurrent communication along a single channel [4], which is finding its way into practial use in the idea of *session types* (see e.g. [28]).

Many results in the paper have close analogues in the multiplicative fragment, or other fragments of linear logic. The coalescence condition is related to Danos's contractibility [6]. Like the latter, which was used by Guerrini to show linear-time correctness of MLL proof nets [11], it provides an effective correctness algorithm. Provability for MLL was first shown to be NP-complete by Kanovich [22]. This and other complexity results were compounded in an early overview [23], which interestingly does not include any results for the additive fragment. The correctness of MALL proof nets was found to be NL-complete [7], but this appears not to impact the (much more restricted) purely additive fragment. Proof equivalence for MLL with units was recently shown to be PSPACE-complete, by Robin Houston and the first author [14].

In one version of proof nets, the additives are treated using *monomial weights* [10]. These have computational advantages,

but they are not canonical. While Danos's contractibility has been extended to these proof nets [25], the difference in structure between weighted and non-weighted proof nets means that this approach is not related to our coalescence.

II. ADDITIVE LINEAR LOGIC

We will consider three fragments of additive linear logic: propositional without units (ALL), propositional with units (ALLU), and first-order without units (FOALL). Fix a set of **atoms** $\{P, Q, R, \ldots\}$, which in the first-order case includes predicates $P(t_1, \ldots, t_n)$ over a first-order term language. We take t, u, v to range over first-order terms and x, y, z to range over term variables.

Formulas are generated by the following grammars:

$$A, B, C \quad ::= \quad P \mid A + B \mid A \times B$$
$$\mid \quad 0 \quad \mid \quad 1 \qquad \text{ALLU only}$$
$$\mid \quad \exists x.A \quad \mid \quad \forall x.A \qquad \text{FOALL only}$$

A **sequent** $A \vdash B$ comprises a **source** formula A and a **target** formula B. Sequent calculi for the three fragments are given in Fig. 2. each fragment includes the inference rules for ALL, while ALLU adds the unit rules and FOALL adds the quantifier rules. The rules for composition and general identity are admissible in each fragment.

Theorem 1. *Cut-elimination and identity-elimination holds for* ALL*,* ALLU*, and* FOALL*.*

This result goes back to Whitman's Theorem for free lattices [29], which essentially states that the cut-free sequent calculus for ALLU is complete; see also [21, Appendix].

III. ALL PROOF NETS

By a **subformula** of a formula A we mean an *occurrence*, as distinguished by a rooted path in the formula tree of A. We denote the size of A by $|A|$, measured in the number of subformulae, or equivalently, connectives and atoms. Given a sequent $A \vdash B$, a **link** $C \!\!-\!\! D$ connects a **source** subformula C in A to a **target** subformula D in B. An **axiom link** is a link between occurrences of the same atom.

Definition 2. A **linking** on a sequent $A \vdash B$ is a set of links on $A \vdash B$. An **axiom linking** is a linking whose every link is an axiom link.

We write $\lambda : A \vdash B$ to indicate that λ is a linking on $A \vdash B$.

A **resolution** r for an additive formula A is a function choosing one child for each subformula that is a product, i.e. either B or C for each subformula $B \times C$ [19]. A subformula C of A is **retained** (opposite: **discarded**) by r if whenever C is a subformula of B_i in a subformula $B_0 \times B_1$ of A, then r chooses B_i. Dually, a **co-resolution** for A chooses on coproducts, and a resolution for a sequent $A \vdash B$ is a pair $r = (r_A, r_B)$ where r_A is a co-resolution for A and r_B is a resolution for B. A link in a linking $\lambda : A \vdash B$ is retained by r if both its source is retained by r_A and its target is retained by r_B.

ALL:

$$\frac{A \vdash B_i}{A \vdash B_0 + B_1} \qquad \frac{A \vdash B \quad A \vdash C}{A \vdash B \times C}$$

$$\overline{P \vdash P}$$

$$\frac{A \vdash C \quad B \vdash C}{A + B \vdash C} \qquad \frac{A_i \vdash B}{A_0 \times A_1 \vdash B}$$

Units:

$$\overline{0 \vdash A} \qquad\qquad \overline{A \vdash 1}$$

Quantifiers:

$$\frac{A \vdash B[t/x]}{A \vdash \exists x.B} \qquad \frac{A \vdash B[y/x]}{A \vdash \forall x.B} \quad y \text{ not free in } A$$

$$\frac{A[t/x] \vdash B}{\forall x.A \vdash B} \qquad \frac{A[y/x] \vdash B}{\exists x.A \vdash B} \quad y \text{ not free in } B$$

Identity & composition:

$$\overline{A \vdash A} \qquad\qquad \frac{A \vdash B \quad B \vdash C}{A \vdash C}$$

Fig. 2. Additive linear logic.

Definition 3. A linking $\lambda : A \vdash B$ is **discrete** if every resolution for $A \vdash B$ retains exactly one link in λ.
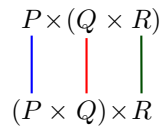
A **proof net** is a discrete axiom linking.

Definition 4. A proof Π of $A \vdash B$ translates to the axiom linking $[\![\Pi]\!] = \lambda : A \vdash B$ where λ collects the axioms in Π as axiom links.

The following proof of $\times$-associativity will provide a running example.

$$\frac{\dfrac{\overline{P \vdash P}}{P \times (Q \times R) \vdash P} \quad \dfrac{\dfrac{\overline{Q \vdash Q}}{Q \times R \vdash Q}}{P \times (Q \times R) \vdash Q}}{\dfrac{P \times (Q \times R) \vdash P \times Q}{P \times (Q \times R) \vdash (P \times Q) \times R}} \quad \dfrac{\dfrac{\overline{R \vdash R}}{Q \times R \vdash R}}{P \times (Q \times R) \vdash R}$$

It translates to the following proof net:

$$P \times (Q \times R)$$
$$\mid \quad \mid \quad \mid$$
$$(P \times Q) \times R$$

Proposition 5 ([18], [19]). *The translation $[\![\Pi]\!]$ of a proof Π is a proof net. For any proof net $\lambda : A \vdash B$ there is a proof Π of $A \vdash B$ such that $[\![\Pi]\!] = \lambda : A \vdash B$.*

The proof Π is a **sequentialisation** of λ.

A. Petri Net criterion

A **transition** on a set $\mathcal{P}$ is pair $\langle s, t \rangle$ whose **source** s and **target** t are subsets of $\mathcal{P}$.[1] A **Petri net** [27] $(\mathcal{P}, {\twoheadrightarrow})$ is a set $\mathcal{P}$ of **places** and a set ${\twoheadrightarrow}$ of transitions on $\mathcal{P}$. We abbreviate $\langle \{p_1, \ldots, p_m\}, \{q_2, \ldots, q_n\} \rangle \in {\twoheadrightarrow}$ to $p_1, \ldots, p_m \twoheadrightarrow q_1, \ldots, q_n$. Example Petri nets were drawn in the Introduction with places as circles $\bigcirc$ and a transition $\langle s, t \rangle$ as a black square $\blacksquare$ with an undirected edge from each place in s and a directed edge to each place in t.
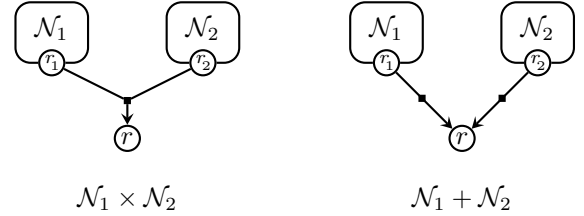
A **marking** is a subset M of $\mathcal{P}$. Elements of M are **tokens**. **Firing** is the rewrite relation on markings defined by

$$M \quad \rightsquigarrow \quad (M \setminus s) \cup t$$

whenever $s \subseteq M$, $s \twoheadrightarrow t$, and t and M are disjoint. The left column of Fig. 1 shows a firing sequence, with tokens $\bullet$.

A **root** is a place r which is not in the source of any transition. A Petri net is **rooted** if it has a unique root. Henceforth assume every Petri net is rooted.

Every ALL formula A determines a Petri net $\mathcal{N}(A)$ upon interpeting the symbols $\times$ and $+$ as operations on Petri nets. Let $\mathcal{N}_1 = (\mathcal{P}_1, {\twoheadrightarrow}_1)$ and $\mathcal{N}_2 = (\mathcal{P}_2, {\twoheadrightarrow}_2)$ be Petri nets with respective roots r_1 and r_2. Define $\mathcal{N}_1 \times \mathcal{N}_2$ as disjoint union plus a transition from the two roots to a new root r, and define $\mathcal{N}_1 + \mathcal{N}_2$ as disjoint union plus two transitions to a new root r, one from r_1 and the other from r_2.

$$\mathcal{N}_1 \times \mathcal{N}_2 \qquad\qquad \mathcal{N}_1 + \mathcal{N}_2$$

Formally:

$$\mathcal{N}_1 \times \mathcal{N}_2 \;=\; (\,\mathcal{P}_1 \sqcup \mathcal{P}_1 \sqcup \{r\}, \; {\twoheadrightarrow}_1 \sqcup {\twoheadrightarrow}_2 \sqcup {\twoheadrightarrow}_\times \,)$$

$$\mathcal{N}_1 + \mathcal{N}_2 \;=\; (\,\mathcal{P}_1 \sqcup \mathcal{P}_1 \sqcup \{r\}, \; {\twoheadrightarrow}_1 \sqcup {\twoheadrightarrow}_2 \sqcup {\twoheadrightarrow}_+ \,)$$

where $\sqcup$ denotes disjoint union and ${\twoheadrightarrow}_\times$ and ${\twoheadrightarrow}_+$ are defined by:

$$r_1, r_2 \;\twoheadrightarrow_\times\; r \qquad\qquad \begin{aligned} r_1 &\;\twoheadrightarrow_+\; r \\ r_2 &\;\twoheadrightarrow_+\; r \end{aligned}$$

Both Petri nets are rooted, with root r. Define the Petri net $\mathcal{N}(P)$ of an atom P as a single place with no transitions, define $\mathcal{N}(A \times B) = \mathcal{N}(A) \times \mathcal{N}(B)$ and define $\mathcal{N}(A + B) = \mathcal{N}(A) + \mathcal{N}(B)$. Examples can be found in the Introduction. By induction the places of $\mathcal{N}(A)$ are in bijection with the subformulas of A, and A itself corresponds to the root place.

The **cartesian product** $\mathcal{N}_1 * \mathcal{N}_2$ is $(\mathcal{P}_1 \times \mathcal{P}_2, {\twoheadrightarrow})$ where ${\twoheadrightarrow}$ is defined by

$$\{p_1\} \times s_2 \;\twoheadrightarrow\; \{p_1\} \times t_2 \quad \text{for all } p_1 \in \mathcal{P}_1 \text{ and } s_2 \twoheadrightarrow_2 t_2$$

$$s_1 \times \{p_2\} \;\twoheadrightarrow\; t_1 \times \{p_2\} \quad \text{for all } p_2 \in \mathcal{P}_2 \text{ and } s_1 \twoheadrightarrow_1 t_1$$

---

[1] Some more general definitions take s and t to be multisets, and markings to be multisets. We shall not require this level of generality.
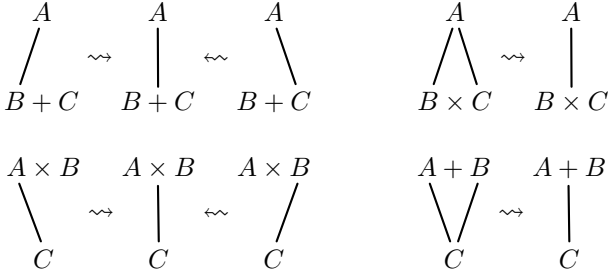
$$A \rightsquigarrow A \leftsquigarrow A \qquad A \rightsquigarrow A$$
$$B+C \quad B+C \quad B+C \qquad B\times C \quad B\times C$$

$$A\times B \rightsquigarrow A\times B \leftsquigarrow A\times B \qquad A+B \rightsquigarrow A+B$$
$$C \qquad C \qquad C \qquad C \qquad C$$

Fig. 3. Coalescence rules.

$$P\times(Q\times R) \rightsquigarrow P\times(Q\times R) \rightsquigarrow P\times(Q\times R)$$
$$(P\times Q)\times R \qquad (P\times Q)\times R \qquad (P\times Q)\times R$$

$$\updownarrow$$

$$P\times(Q\times R) \leftsquigarrow P\times(Q\times R) \leftsquigarrow P\times(Q\times R)$$
$$(P\times Q)\times R \qquad (P\times Q)\times R \qquad (P\times Q)\times R$$

$$\updownarrow$$

$$P\times(Q\times R) \rightsquigarrow P\times(Q\times R)$$
$$(P\times Q)\times R \qquad (P\times Q)\times R$$

Fig. 4. Coalescing the associativity example.

Define $\mathcal{N}(A \vdash B) = \mathcal{N}(A) * \mathcal{N}(B)$. (The Introduction has examples.) Via the subformula-to-place bijections, every link A'—B' on $A \vdash B$ corresponds to a distinct place $\mathcal{P}(A'$—$B')$ in $\mathcal{N}(A \vdash B)$, and $\mathcal{P}(A$—$B)$ is the root.

The **root marking** contains only the root. Let $\lambda : A \vdash B$ be a linking. The **link marking** of λ is the marking of $\mathcal{N}(A \vdash B)$ determined by the links of λ: $\{ \mathcal{P}(A'$—$B') : A'$—$B' \in \lambda \}$. A **run** of λ is a maximal firing sequence starting from the link marking of λ; the last marking of the sequence is the **result**.

Definition 6. The **Petri net criterion** is the following function of a linking: choose a run; if the result is the root marking return PASS, otherwise return FAIL.

The results below follow from the next section, in which we recast the Petri net criterion as the coalescence criterion.

Theorem 7. *The Petri net criterion is deterministic: its* PASS/FAIL *output is independent of the choice of run.*

Theorem 8. *An axiom linking is a proof net iff it satisfies the Petri net criterion.*

B. Coalescence criterion

We recast the Petri net criterion directly in terms of linkings to define the *coalescence criterion*. Coalescence is analogous to Danos' *contractibility* for multiplicative proof nets [6]: it is a simple rewriting procedure whose rewrite steps correspond directly to ALL inference rules, which is confluent on proof nets and reduces them to trivial form. The rewrite rules, illustrated in Fig. 3, are as follows.

Definition 9. *Coalescence* is the rewrite relation on linkings generated by the following rewrite rules:

- replace a link A—B or one A—C by one A—$B+C$;
- replace two links A—B and A—C by one A—$B\times C$;
- replace two links A—C and B—C by one $A+B$—C;
- replace a link A—C or one B—C by one $A\times B$—C.

A linking $\lambda : A \vdash B$ **weakly coalesces** if there is a sequence of coalescence steps starting from λ and ending in a single link A—B. A linking $\lambda : A \vdash B$ **coalesces** if any coalescence sequence terminates in A—B.

Fig. 4 illustrates the coalescence of our running example, the associativity proof net. The coalescence rewrite steps of Definition 9 correspond one-to-one to the inference rules of ALL in Fig. 2, where the links in the left- and right-hand side of the coalescence step correspond, respectively, to the premises and conclusion of the inference rule. It is readily observed that inference rules and coalescence steps preserve discreteness in both directions. This leads to the following proposition.
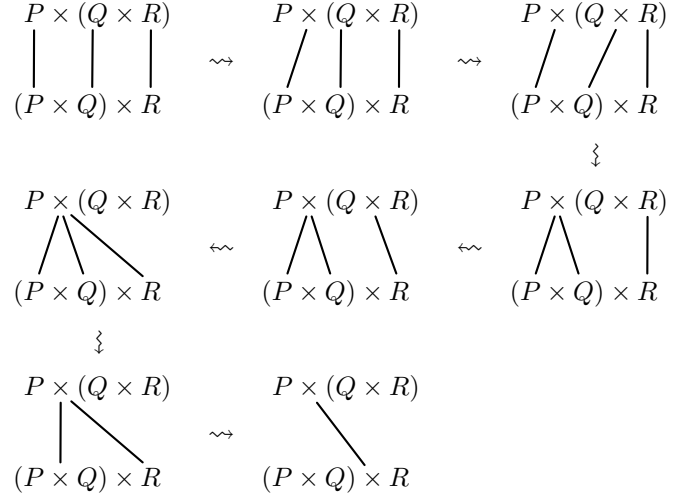
Proposition 10. *A linking is discrete if and only if it weakly coalesces.*

Proof. By a minor generalisation of Proposition 5, a linking $\lambda : A \vdash B$ is discrete if and only if it sequentialises to a derivation in ALL extended with arbitrary axioms $\overline{C \vdash D}$ (which correspond to links). Using the correspondence between inference rules and coalescence steps, this sequentialisation may be turned into a coalescence sequence terminating in the single link A—B, and vice versa. $\qquad\square$

Proposition 11. *Coalescence is confluent on discrete linkings.*

Proof. The critical pairs of the coalescence relation are the pairs of rewrite steps where both replace the same link. There are at most two rewrite rules that apply to a link, one determined by the parent formula of the source, and one by that of the target. It will be shown that every critical pair can be resolved.

For a source with parent $A\times B$ and target with parent $C+D$, a critical pair

$$A\times B \leftsquigarrow A\times B \rightsquigarrow A\times B$$
$$C+D \qquad C+D \qquad C+D$$
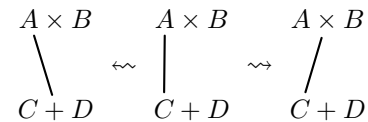
resolves immediately:

$$A\times B \rightsquigarrow A\times B \leftsquigarrow A\times B$$
$$C+D \qquad C+D \qquad C+D$$
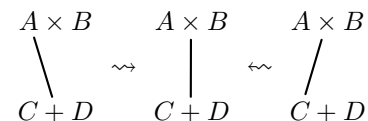
The other cases, for links A—C, B—C, and B—D, are similar. For $A\times B$ and $C\times D$, a critical pair

resolves as follows.

Other cases are similar, as are those for $A + B$ and $C + D$.

For $A + B$ and $C \times D$, a critical pair is

To resolve the pair, a link $B \!-\! D$ is needed. We show that further coalescence exposes this link.

Let the configuration above centre occur in a linking $\lambda : X \vdash Y$. Since coalescence preserves discreteness, this linking is discrete, and any resolution r of $X \vdash Y$ retaining $B \vdash D$ must retain exactly one link. This link must lie within $B \vdash D$, for the following reason. By changing r to choose A rather than B in X, or C rather than D in Y, one of the links $A \!-\! C$, $A \!-\! D$, or $B \!-\! C$, is retained. Then by discreteness, no link outside $A + B \vdash C \times D$ may be retained by r. The links in $B \vdash D$ thus form a discrete linking, which by Proposition 10 coalesces to a single link $B \!-\! D$.

With $B \!-\! D$ present, the critical pair is resolved as follows.

The other cases are similar. $\qquad\qquad\qquad\qquad\square$

Our main theorem on coalescence is:

Theorem 12. *A linking is discrete if and only if it coalesces.*

Proof. By Proposition 10 a linking is discrete if and only if it weakly coalesces, and by Proposition 11 it weakly coalesces if and only if it coalesces. $\qquad\qquad\qquad\qquad\square$

Corollary 13. *Correctness of an* ALL *proof net* $\lambda : A \vdash B$ *is decidable in time* $\mathcal{O}(|A| \times |B|)$.

Theorem 14. *Correctness of an* ALL *proof net* $\lambda : A \vdash B$ *is decidable in time* $\mathcal{O}\big(|\lambda| \times (\mathsf{d}A + \mathsf{d}B) \times \max(\log|A| + \log|B|)\big)$.

Here $|\lambda|$ is the number of links in λ, $|C|$ is the number size of a formula C, and $\mathsf{d}C$ is the depth of C.

Proof. During coalescence the maximum number of times a token (link) can fire before termination is $\mathsf{d}A + \mathsf{d}B$, since each firing moves one step closer to the root of one of A or B. After each firing we must check if a new two-token firing becomes enabled. This requires maintaining a data structure to retrieve candidates for the other token. Employing a balanced binary tree (for example) provides worst case $\log n$ complexity for looking up among n candidates. Since every candidate is from either A or B, the look-up complexity is bounded by $\max(\log|A| + \log|B|)$. $\qquad\qquad\square$
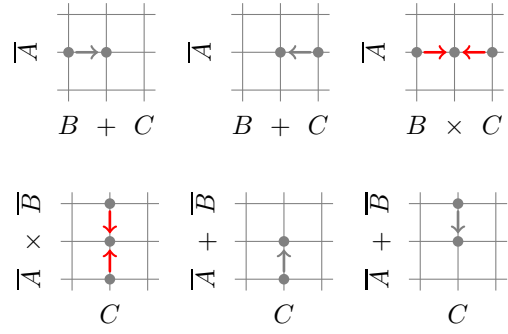
Fig. 5.   Inference rules in grid notation
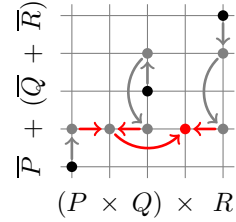
Fig. 6.   The associativity proof in grid notation

C. Grid notation

In a discrete linking $\lambda : A \vdash B$ links are sparse, i.e. λ is small relative to its potential maximal size of $|A| \times |B|$. For non-sparse sets of links, where the proof net representation used thus far is not convenient, we will introduce the representation used below right.
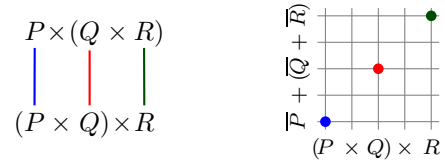
The links of a linking are displayed as tokens on a grid. The source formula is displayed up the left side and the target formula along the bottom. Horizontal grid lines correspond to source subformulas, and vertical gridlines to target subformulas, so each crossing is a potential link.

To concisely represent proofs in a grid, an inference will be drawn as one or two arrows between link tokens for its premises and conclusion. Inference rules in grid notation are given in Fig. 5, and the example associativity proof is given in Fig. 6. Axiom links are drawn as black tokens, the root link is highlighted in red, and other links are in grey.

D. Efficient proof search: spawning

We define *spawning* as a variant of coalescence which performs efficient proof search. To establish provability of $A \vdash B$, a linking is generated in which a link $C \!-\! D$ indicates the provability of a sub-sequent $C \vdash D$ of $A \vdash B$. The initial configuration is the linking $\lambda_{A,B} : A \vdash B$ comprising every possible axiom link on $A \vdash B$. The spawning rewrite relation that propagates provability is the variant of coalescence in which links are only added, but never deleted.
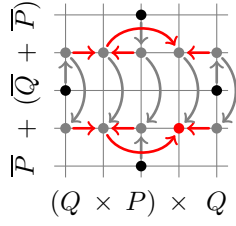
Fig. 7. Spawning a provability grid
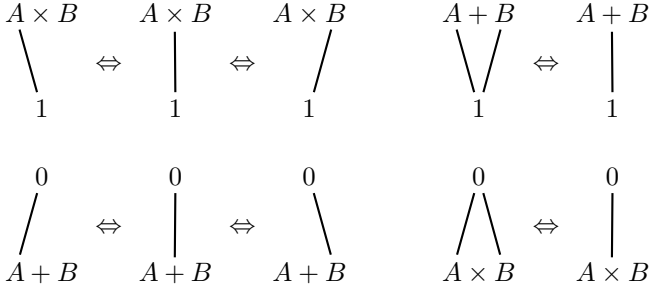
Fig. 8. Equivalence of ALLU proof nets

Definition 15. *Spawning* is the rewrite relation generated by the following rewrite steps:

- given a link $A - B$ or one $A - C$ add $A - B + C$;
- given two links $A - B$ and $A - C$ add $A - B \times C$;
- given two links $A - C$ and $B - C$ add $A + B - C$;
- given a link $A - C$ or one $B - C$ add $A \times B - C$.

The *provability grid* of a sequent $A \vdash B$ is the result of exhaustive spawning on $\lambda_{A,B} A \vdash B$.

Theorem 16. *A sequent $A \vdash B$ is provable in* ALL *if and only if its provability grid contains the root link $A - B$.*

Proof. By induction on the spawning relation, a link $C - D$ exists in the provability grid if and only if there is a proof of $C \vdash D$. □

An algorithm implementing spawning need perform only a single pass over the grid of $A \vdash B$, by respecting the product order over the subformula relation—i.e. $C - D \leq C' - D'$ if and only if C is a subformula of C', and D one of D'.

Corollary 17. *Provability for an* ALL *sequent $A \vdash B$ is decidable in time $\mathcal{O}(|A| \times |B|)$.*

Fig. 7 shows an example provability grid, as it is generated by spawning, for the sequent $P \times (Q \times P) \vdash (Q \times P)Q$. As the illustration suggests, to obtain a witness—an actual proof—from a provability grid, it is sufficient to (non-deterministically) retrace the steps taken by the spawning relation.

IV. ALLU

Canonical nets for ALLU, *saturated nets*, were introduced by the first author in [12]. They are obtained by a *saturation* rewrite procedure, from ALL proof nets extended with ***unit links***, links $0 - A$ and $A - 1$ for any A.
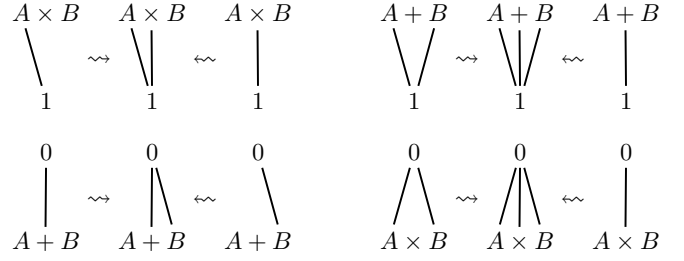
Fig. 9. Saturation (symmetric cases omitted)

Definition 18. An ALLU ***proof net*** is a discrete linking in which every link is an axiom link or a unit link.

Definition 19. *Equivalence* ($\Leftrightarrow$) of ALLU proof nets is the equivalence relation generated by the rewrite steps:

- replace two links $A - 1$ and $B - 1$ by one $A + B - 1$;
- replace a link $A - 1$ or one $B - 1$ by one $A \times B - 1$;
- replace a link $0 - B$ or one $0 - C$ by one $0 - B + C$;
- replace two links $0 - B$ and $0 - C$ by one $0 - B \times C$.

Proof net equivalence is illustrated in Fig. 8.

Definition 20. *Saturation* is the rewrite relation on ALLU linkings generated by:

- given $A - 1$ and $B - 1$, add $A + B - 1$, and vice versa;
- given $A - 1$ or $B - 1$, add $A \times B - 1$, and vice versa;
- given $0 - B$ or $0 - C$, add $0 - B + C$, and vice versa;
- given $0 - B$ and $0 - C$, add $0 - B \times C$, and vice versa.

Saturation is illustrated in Fig. 9.

Definition 21. A *saturated net* is the normal form of a proof net under saturation.

Two other characterisations of saturated nets are as follows.

Proposition 22 ([13, Proposition 3.3.2]). *Saturated nets are the linkings $\lambda : A \vdash B$ such that for some* ALLU *proof net $\theta : A \vdash B$,*

$$\lambda = \bigcup \{\phi \mid \phi : A \vdash B \Leftrightarrow \theta : A \vdash B\} \ .$$

Definition 23. The symmetric *neighbouring* relation $\frown$ connects links $A - B \frown C - D$ where $A = C = 0$ and D is a child of B, or $B = D = 1$ and C is a child of A.

Proposition 24 ([13, Theorem 3.4.3]). *Saturated nets are the saturated linkings such that for every resolution the retained links form exactly one connected component under $\frown$.*

The main theorem of the first author's previous work on additive linear logic is that saturated nets are canonical for finite sum–product categories [12], [13].

Theorem 25 ([13, Theorem 3.3.1]). ALLU *proof nets are equivalent if and only if they have the same saturation.*

Time complexity of saturation is as follows. a single step in the saturation process can be performed in constant time, since placing a link token requires the inspection of only direct neighbours (on the grid). A saturation algorithm that keeps a stack of potential links to place, for each link placed pushing

(some of) its neighbours onto the stack, visits only a constant number of new link positions for each link in the eventual saturated net. This gives the following proposition.

Proposition 26 ([13, Section 3.5])**.** *Saturation is linear-time in the size of the saturated net.*

Before moving on, we briefly remark on ALLU proof search.

A. Proof search

The proof search algorithm for ALL can be used to find proof nets for ALLU simply by extending it with unit links as well as axiom links. To find a saturated net it is sufficient to saturate the proof net found by proof search.

Proposition 27. *Provability of a sequent $A \vdash B$ in* ALLU *is decidable in time $\mathcal{O}(|A| \times |B|)$.*

B. Correctness

Neither of the three characterisations of saturated nets directly offers an effective algorithm to determine their correctness. Here, we will introduce such an algorithm: a *desaturation* procedure that given a linking $\sigma : A \vdash B$, in time $\mathcal{O}(|A| \times |B|)$ returns a proof net $\lambda : A \vdash B$ whose saturation is σ if and only if $\sigma : A \vdash B$ is a saturated net. The key ingredient to the algorithm is a *factorisation* grid similar to the provability grid used for ALL proof search, which makes the factorisation of a saturated net (through projections, injections etc.) accessible without costly backtracking.

Note that it is not sufficient for desaturation to simply pick a subnet (a subset of its links, forming a net) of the saturated net. This is illustrated by the two saturated nets below, where the second is contained in the first.

The dynamics of saturated nets are dominated by the interaction between initial and terminal maps, via the unique map $0 \to 1$. A *copointed* map is one that factors through 0, a *pointed* map one that factors through 1, and for any $A \to B$ there is at most one *bipointed* map, that factors through both.

Definition 28. The *pointed* respectively *copointed* formulae of ALLU are given by:

$$X \quad ::= \quad 1 \quad | \quad X + A \quad | \quad A + X \quad | \quad X \times X$$

$$Y \quad ::= \quad 0 \quad | \quad Y + Y \quad | \quad Y \times A \quad | \quad A \times Y$$

A sequent $X \vdash Y$ is *bipointed*.

Definition 29. A linking $\lambda : A \vdash B$ is *pointed* if every resolution of B retains a link $A \longrightarrow 1$ in λ, and *copointed* if every co-resolution of A retains a link $0 \longrightarrow B$. A linking that is both pointed and copointed is *bipointed*.

Given a pointed formula X there is a *canonical* pointed net, which is biased to factor through the first injection of an object $X + X$ when both subformulae are pointed. Dually, a *canonical* copointed net is biased to factor through the first projection.

Definition 30. A linking $\lambda : A \vdash B$ is *full* if λ contains every unit link in $A \vdash B$, but no axiom links (or other links).

The following proposition characterises the behaviour of pointed and copointed proof nets. The uniqueness of bipointed maps is captured simply by the saturated net being full.

Proposition 31 ([13, Lemma 4.2.3, Lemma 4.3.7])**.** *The saturation of the following nets is full:*
- *any net on a sequent $0 \vdash A$ or $A \vdash 1$;*
- *a pointed net $\lambda : X \vdash Y$ with X copointed;*
- *a copointed net $\lambda : X \vdash Y$ with Y pointed.*

To allow inductive reasoning on saturated nets we formalise when a linking factors through a projection or injection, or as a pair or copair. Definition 33 is chosen to be uniform for both proof nets and saturated nets. Given a linking $\lambda : A \vdash B$ and subformulae C, D of A, B respectively, let $\lambda|_{C \vdash D}$ be the sub-linking obtained by restricting λ to the sequent $C \vdash D$.

Definition 32. A linking $\lambda : A \vdash B$ is *connected* if it has a sub-linking $\phi \subseteq \lambda$ such that $\phi : A \vdash B$ is discrete.

Definition 33. A linking $\lambda : A \vdash B$ factors through:
- a *projection* π_i if $A = A_0 \times A_1$ and $\lambda|_{A_i \vdash B}$ is connected,
- an *injection* ι_i if $B = B_0 + B_1$ and $\lambda|_{A \vdash B_i}$ is connected,
- a *pair* if $B = B_0 \times B_1$ and both $\lambda|_{A \vdash B_i}$ are connected,
- a *copair* if $A = A_0 + A_1$ and both $\lambda|_{A_i \vdash B}$ are connected.

Proposition 34. *If a saturated net factors through a projection / injection / pair / copair, it is the saturation of a proof net that factors similarly.*

Proof. The case for pairs is immediate: a net into $A \times B$ always has an equivalent net that factors through a pair, obtained by replacing each link $0 \longrightarrow A \times B$ by $0 \longrightarrow A$ and $0 \longrightarrow B$. Dually, also the case for copairs is immediate.

For a saturated net that factors both through a pair and a projection, we split into a pair as above and reason by induction on the components. The case for a copair and injection is dual.

This leaves the case of a saturated net σ from $A \times B$ into $C + D$. If it factors through only one projection or injection, any net λ of which it is the saturation must factor similarly, since $\lambda \subseteq \sigma$. The remaining case, where Σ factors in multiple ways, is exactly Lemma 4.5.2 of [13]. $\qquad\square$

The above gives us the prerequisites to define the *desaturation* procedure. It is non-deterministic, to allow for the fact that a saturated net may factor in multiple ways.

Definition 35. A *desaturation* of a linking $\lambda : A \vdash B$ is a net $d : A \vdash B$ obtained by the following procedure:

1) if λ is full, and A is copointed or B is pointed, let $d : A \vdash B$ be the canonical copointed or pointed net;

otherwise, d is generated by one of the following steps:

2) if A and B are atomic, d is $\{A \longrightarrow B\}$;

3) if $A = A_0 + A_1$ then $d = d_0 \cup d_1$, where d_i is a desaturation of $\lambda|_{A_i \vdash B}$ for $i = 0, 1$;
4) if $B = B_0 \times B_1$ then $d = d_0 \cup d_1$, where d_i is a desaturation of $\lambda|_{A \vdash B_i}$ for $i = 0, 1$;
5) if $A = A_0 \times A_1$ and λ factors through a projection π_i, then d is a desaturation of $\lambda|_{A_i \vdash B}$;
6) if $B = B_0 + B_1$ and λ factors through an injection ι_i, then d is a desaturation of $\lambda|_{A \vdash B_i}$.

The desaturation procedure is an inverse to saturation (modulo equivalence). This is expressed by the following theorem.

Theorem 36. *A saturated net $\sigma : A \vdash B$ has a desaturation, and any desaturation of it saturates to σ.*

Proof. See appendix. $\qquad\square$

Two crucial properties of the desaturation procedure are that 1) it recurses only by restricting links to a sub-sequent, without deleting any, and 2) it relies only on how a linking factorises, whether it is full, and whether its source and target are (co)pointed. The point of 2) is that all three properties can be pre-computed in a single pass of the proof grid or the formula, and the point of 1) is that the pre-computed data remain correct when recursively finding a desaturation.

Definition 37. The *factorisation grid* for a linking $\lambda : A \vdash B$ is the linking obtained by exhaustive application of spawning (Definition 15) on λ.

Definition 38. The *fullness grid* for a linking $\lambda : A \vdash B$ contains the link $C \text{---} D$ if and only if $\lambda|_{C \vdash D}$ is full.

As with provability search, by respecting the product order over the subformula relation both grids may be computed in a single pass. This brings us to the main result for ALLU.

Theorem 39. *Correctness of a saturated net $\sigma : A \vdash B$ is decidable in time $\mathcal{O}(|A| \times |B|)$.*

Proof. Desaturation gives a net d saturating to σ if and only if $\sigma : A \vdash B$ is correct: from left to right is immediate (if any net saturates to σ it is correct), the other direction is Theorem 36. Desaturation can be performed by a single simultaneous walk on the factorisation grid of σ, its fullness grid, and A and B annotated with copoint / point information. Generating both grids is in time $\mathcal{O}(|A| \times |B|)$, as is saturating the desaturation that is found (Proposition 26). $\qquad\square$

V. FOALL

The main result of this section is that the provability problem for FOALL is NP-complete. Membership of NP is immediate by the size of proofs. We will show NP-hardness by a reduction from Boolean satisfyability (SAT; see [8]).

First, we will sketch how a Boolean formula A and an assignment γ for A, a function from the propositional atoms in A to truth values $\{\bot, \top\}$, may be interpreted in FOALL. The formula A is assumed to be in negation-normal form. We will encode a Boolean atom (propositional variable) p and its negation $\neg p$ by two distinct, unrelated atomic formulae, $P(\top)$ and $P(\bot)$, constructed by instantiating a predicate $P(x)$ with two distinct first-order constants $\top$ and $\bot$. Let the formula $\lfloor A \rfloor$ be the direct additive interpretation of A, as follows.

$$\lfloor p \rfloor = P(\top) \qquad \lfloor A \vee B \rfloor = \lfloor A \rfloor + \lfloor B \rfloor$$

$$\lfloor \neg p \rfloor = P(\bot) \qquad \lfloor A \wedge B \rfloor = \lfloor A \rfloor \times \lfloor B \rfloor$$

The interpretation $\lfloor \gamma \rfloor$ of the assignment γ is the product over each atomic formula, instantiated with its assigned value:

$$\lfloor \gamma \rfloor = \prod \{ P(x) \mid \gamma(p) = x \}$$

Provability of the sequent $\lfloor \gamma \rfloor \vdash \lfloor A \rfloor$ then encodes the evaluation of A under the assignment γ.

Proposition 40. *An assignment γ is satisfying for a Boolean formula A if and only if $\lfloor \gamma \rfloor \vdash \lfloor A \rfloor$ is provable in FOALL.*

Proof. For an atom p in A, the sequent $\lfloor \gamma \rfloor \vdash P(\top)$ is provable if and only if $\gamma(p) = \top$, and $\lfloor \gamma \rfloor \vdash P(\bot)$ is provable if and only if $\gamma(p) = \bot$. By induction on A it follows that $\lfloor \gamma \rfloor \vdash \lfloor A \rfloor$ is provable if and only if γ is satisfying for A. $\qquad\square$

The next step is to encode the possibility of assigning mutually exclusive truth values to atomic formulae. The chosen encoding of Boolean atoms p and $\neg p$, as a predicate $P(x)$ over distinct constants $\bot$ and $\top$, means this can be expressed via the formula $\forall x. P(x)$, which quantifies over both truth values. For example, the Boolean formula $p \vee \neg p$ may be encoded as

$$\forall x. P(x) \vdash P(\top) + P(\bot)$$

However, in this naive formulation the interpretation of the contradiction $p \wedge \neg p$ would become:

$$\forall x. P(x) \vdash P(\top) \times P(\bot)$$

which is provable, by the following proof.

$$\frac{\dfrac{\overline{P(\top) \vdash P(\top)}}{\forall x. P(x) \vdash P(\top)} \quad \dfrac{\overline{P(\bot) \vdash P(\bot)}}{\forall x. P(x) \vdash P(\bot)}}{\forall x. P(x) \vdash P(\top) \times P(\bot)}$$

The problem is that the product rule appears below the universal quantifier rule in the proof, which means the quantifier may be instantiated differently for both branches of the product. To remedy this, we introduce a "lock" construction that forces the universal quantifier to be instantiated first. The main lock mechanism consists of two existential quantifiers, inserted after the universal quantifier and before the product. Each binds the variable y in a special predicate $\mathsf{lock}(y)$, with occurrences in the antecedent and in the conclusion that must in a proof become linked by an axiom. Then the existential quantifier in the antecedent must be instantiated before that in the consequent, as both must take the same value. The interpretation of $p \wedge \neg p$ then becomes as follows.

$$\forall x \exists y. \mathsf{lock}(y) \times P(x) \vdash \exists y. \mathsf{lock}(y) \times P(\top) \times P(\bot)$$

The full construction is then the following.

Definition 41. Let $\|A\|$ be the following sequent, for a Boolean formula A with atoms $p_1, \ldots, p_n$.

$$\forall x_1 \ldots \forall x_n \exists y. \mathsf{lock}(y) \times \prod_{1 \leq i \leq n} P_i(x_i) \vdash \exists y. \mathsf{lock}(y) \times \lfloor A \rfloor$$

Proposition 42. *The Boolean formula A is satisfiable if and only if $\|A\|$ is provable in* FOALL.

Proof. From left to right, let γ be a satisfying assignment for A. A proof of $\|A\|$ is constructed by first instantiating each universally quantified variable x_i with the constant $\gamma(a_i) \in \{\bot, \top\}$, so that the antecedent becomes the formula $\exists y.\, \text{lock}(y) \times \lfloor\gamma\rfloor$. Next, there is the following derivation for the lock construction.

$$
\dfrac{\dfrac{\dfrac{\dfrac{\overline{\text{lock}(a) \vdash \text{lock}(a)}}{\text{lock}(a) \times \lfloor\gamma\rfloor \vdash \text{lock}(a)} \quad \dfrac{\vdots}{\dfrac{\lfloor\gamma\rfloor \vdash \lfloor A\rfloor}{\text{lock}(a) \times \lfloor\gamma\rfloor \vdash \lfloor A\rfloor}}}{\text{lock}(a) \times \lfloor\gamma\rfloor \vdash \text{lock}(a) \times \lfloor A\rfloor}}{\text{lock}(a) \times \lfloor\gamma\rfloor \vdash \exists y.\, \text{lock}(y) \times \lfloor A\rfloor}}{\exists y.\, \text{lock}(y) \times \lfloor\gamma\rfloor \vdash \exists y.\, \text{lock}(y) \times \lfloor A\rfloor}
$$

The remaining sequent, $\lfloor\gamma\rfloor \vdash \lfloor A\rfloor$, is provable by Proposition 40, since γ is a satisfying assignment for A.

From right to left, let Π be a cut-free proof of $\|A\|$. Since Π proves the consequent $\exists y.\, \text{lock}(y) \times \lfloor A\rfloor$, it must contain an axiom $\text{lock}(a) \vdash \text{lock}(a)$. By construction, there is only one occurrence of the predicate lock in the antecedent of $\|A\|$, so that Π must contain the following sub-derivation.

$$
\dfrac{\dfrac{\dfrac{\vdots}{\text{lock}(a) \times B \vdash} \quad \text{lock}(a) \times \lfloor A\rfloor}{\text{lock}(a) \times B \vdash \exists y.\, \text{lock}(y) \times \lfloor A\rfloor}}{\exists y.\, \text{lock}(y) \times B \vdash \exists y.\, \text{lock}(y) \times \lfloor A\rfloor}
$$

Here, $\text{lock}(a) \times B$ is the (fully instantiated) propositional part of the antecedent of $\|A\|$.

The side-condition on the left-introduction rule for the existential quantifier, that the eigenvariable a may not occur in the consequent $\exists y.\, \text{lock}(y) \times \lfloor A\rfloor$, means that the two inferences above may not permute. The proof Π must then instantiate each universal quantification below the existential one, which means each is instantiated exactly once. Define an assignment γ for A as follows: $\gamma(a_i) = \top$ if x_i is instantiated to $\top$ in Π, and $\gamma(a_i) = \bot$ otherwise (note that this includes those x_i that are instantiated to anything other than $\bot$ or $\top$).

As Π is cut-free, each inference in its propositional part must either decompose $\lfloor A\rfloor$, or project onto a smaller fragment of the product B. By induction on this sub-proof it follows that γ is a satisfying assignment for A. $\square$

The above proposition then allows to prove the main theorem of this section.

Theorem 43. *The provability problem for* FOALL *is NP-complete.*

Proof. Since a proof for a sequent $A \vdash B$ has size no larger than $\mathcal{O}(|A| \times |B|)$, FOALL provability is in NP. Proposition 42 gives a polynomial-time reduction from the Boolean satisfiability problem, proving that FOALL provability is NP-hard. $\square$

REFERENCES

[1] Samson Abramsky and Radha Jagadeesan. Games and full completeness for multiplicative linear logic. *Journal of Symbolic Logic*, 59(2):543–574, 1994.
[2] Samson Abramsky and Paul-André Melliès. Concurrent games and full completeness. In *Proc. 14th Annual IEEE Symposium on Logic in Computer Science (LiCS'99)*, 1999.
[3] Andreas Blass. A game semantics for linear logic. *Ann. Pure Appl. Logic*, 56:183–220, 1992.
[4] Robin Cockett and Luigi Santocanale. On the word problem for ΣΠ-categories, and the properties of two-way communication. In *Proc. 18th EACSL Annual Conference on Computer Science Logic (CSL'09)*, volume 5771, pages 194–208, 2009.
[5] Robin Cockett and Robert Seely. Finite sum-product logic. *Theory and Applications of Categories*, 8(5):63–99, 2001.
[6] Vincent Danos. *La Logique Linéaire appliquée à l'étude de divers processus de normalisation (principalement du Lambda-calcul)*. PhD thesis, Université Paris 7, 1990.
[7] Paulin Jacobé De Naurois and Virgile Mogbil. Correctness of multiplicative additive proof structures is nl-complete. In *LICS*, 2008.
[8] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.
[9] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50(1):1–102, 1987.
[10] Jean-Yves Girard. Proof-nets: the parallel syntax for proof-theory. *Logic and Algebra*, pages 97–124, 1996.
[11] Stefano Guerrini. A linear algorithm for mll proof net correctness and sequentialization. *Theoretical Computer Science*, 412(20):1958–1978, 2011.
[12] Willem Heijltjes. Proof nets for additive linear logic with units. In *Proc. 26th Annual IEEE Symposium on Logic in Computer Science (LiCS'11)*, pages 207–216, 2011.
[13] Willem Heijltjes. *Graphical representation of canonical proof: Two case studies*. PhD thesis, School of Informatics, University of Edinburgh, 2012.
[14] Willem Heijltjes and Robin Houston. No proof nets for MLL with units: Proof equivalence in MLL is PSPACE-complete. In *CSL-LICS*, 2014.
[15] H. Hu. Contractible coherence spaces and maximal maps. *Elec. Notes in Theor. Comp. Sci.*, 20, 1999.
[16] D. J. D. Hughes. Proofs Without Syntax. *Annals of Mathematics*, 143:1065–1076, 2006.
[17] D. J. D. Hughes. Towards Hilbert's 24th Problem: Combinatorial Proof Invariants. In *Proc. WOLLiC'06*, volume 165 of *Lec. Notes in Comp. Sci.*, 2006.
[18] Dominic J. D. Hughes. A canonical graphical syntax for non-empty finite products and sums. Technical report, Stanford University, 2002.
[19] Dominic J. D. Hughes and Rob van Glabbeek. Proof nets for unit-free multiplicative-additive linear logic. *ACM Transactions on Computational Logic*, 6(4), 2005.
[20] Andre Joyal. Free bicomplete categories. *C.R. Math. Rep. Acad. Sci. Canada*, XVII(5):219–224, 1995.
[21] Andre Joyal. Free lattices, communication and money games. *Proc. 10th Int. Cong. of Logic, Methodology and Philosophy of Science*, 1995.
[22] Max I. Kanovich. Horn programming in linear logic is np-complete. In *LICS*, 1992.
[23] Patrick Lincoln, John Mitchell, Andre Scedrov, and Natarajan Shankar. Decision problems for propositional linear logic. *Annals of Pure and Applied Logic*, 56:239–311, 1992.
[24] R. Loader. Linear logic, totality and full completeness. In *Proc. 9th Annual IEEE Symp. on Logic in Computer Science, Paris, July 1994*, pages 292–298. IEEE Computer Society Press, 1994.
[25] Roberto Maieli. Retractile proof nets of the purely multiplicative and additive fragment of linear logic. In *LPAR*, 2007.
[26] Samuel Mimram. The structure of first-order causality. *Mathematical Structures in Computer Science*, 21(1):65–110, 2011.
[27] C. A. Petri. *Kommunikation mit Automaten*. PhD thesis, University of Bonn, 1962.
[28] Philip Wadler. Propositions as sessions. *ACM SIGPLAN Notices*, 47(9), 2012.
[29] P.M. Whitman. Free lattices. *Annals of Mathematics*, 42(1):325–330, 1941.

APPENDIX

This appendix contains the proof and supporting definitions and lemmata of Theorem 36.

Proposition 44. *The saturation* $\sigma : A \vdash B$ *of an* ALLU *proof net* $\lambda : A \vdash B$ *is as follows.*

- *If* λ *is a singleton axiom link* $\sigma = \lambda$; *if* λ *is a singleton unit link* σ *is full.*
- *If* $A = A_0 + A_1$ *and* λ *factors as a copair of* $\lambda_0 : A_0 \vdash B$ *and* $\lambda_1 : A_1 \vdash B$, *then* σ *is the union of the two saturations* σ_i *of* λ_i *plus the set*
$$\{A \text{ --- } 1 \mid A_0 \text{ --- } 1 \in \sigma_0 \text{ and } A_1 \text{ --- } 1 \in \sigma_1 \} .$$
- *If* $B = B_0 \times B_1$ *and* λ *factors as a pair of* $\lambda_0 : A \vdash B_0$ *and* $\lambda_1 : A \vdash B_1$, *then* σ *is the union of the two saturations* σ_i *of* λ_i *plus the set*
$$\{0 \text{ --- } B \mid 0 \text{ --- } B_0 \in \sigma_0 \text{ and } 0 \text{ --- } B_1 \in \sigma_1 \} .$$
- *If* $A = A_0 \times A_1$ *and* λ *factors through a projection* π_i, *then* σ *is the union of the saturation* σ_i *of* $\lambda : A_i \vdash B$ *plus:*
$$\Gamma = \{C \text{ --- } 1 \mid C \text{ is a subformula of } A, \ A_i \text{ --- } 1 \in \sigma_i\}$$
$$\Delta = \{C \text{ --- } D \mid C \text{ --- } D \text{ is a unit link in some } X \vdash Y \text{ in }$$
$$A \vdash B \text{ with } X \text{ copointed and } \sigma_i|_{A_i \vdash Y} \text{ pointed } \} .$$
- *If* $B = B_0 + B_1$ *and* λ *factors through an injection* ι_i, *then* σ *is the union of the saturation* σ_i *of* $\lambda : A \vdash B_i$ *plus:*
$$\Gamma = \{0 \text{ --- } D \mid D \text{ is a subformula of } B, \ 0 \text{ --- } B_i \in \sigma_i\}$$
$$\Delta = \{C \text{ --- } D \mid C \text{ --- } D \text{ is a unit link in some } X \vdash Y \text{ in }$$
$$A \vdash B \text{ with } \sigma_i|_{X \vdash B_i} \text{ copointed and } Y \text{ pointed } \} .$$

Proof. Cases 2 and 3 are [13, Lemma 4.2.6]; cases 4 and 5 are [13, Lemma 4.4.1]. $\qquad\square$

Definition 45. A *maximal copoint* of a linking $\lambda : A \vdash B$ is a largest copointed linking $\lambda|_{X \vdash B}$ in λ. Dually, a *maximal point* is a largest pointed linking $\lambda|_{A \vdash Y}$.

Definition 46. The *P, Q-super-saturation* of a saturated net $\sigma : A \vdash B$ is a linking obtained by replacing a set P of compatible rooted points and a set Q of compatible rooted copoints by corresponding full linkings. The super-saturation of a net is the super-saturation of its saturation.

Lemma 47. *A P, Q-super-saturated net $\tau : A \vdash B$ has at least one desaturation $d : A \vdash B$, and each desaturation d is such that 1) τ is a super-saturation of d, and 2) the saturation of d contains the canonical point for each P and the canonical copoint for each Q.*

Proof. The proof is by induction on A and B, following the desaturation steps in Definition 35. Let $\tau : A \vdash B$ be the super-saturation of the net $\lambda : A \vdash B$. First, note that at least one desaturation step applies: step 2 if λ is the single axiom link $A \text{ --- } B$, and step 1 if λ is a single unit link. If λ factors through a copair, pair, projection or injection, then so does σ (Propositon 34), and so does τ, since the saturation of a copointed net (into a product or sum) already factors on the right, and dually for pointed nets. Then step 3, 4, 5, or 6 applies, respectively.

Case 1: τ is full, A is copointed or B is pointed, and $d : A \vdash B$ is the canonical copointed or pointed net. It is immediate that τ a super-saturation of d. If A is copointed, the saturation of d is full for any $A \vdash Y$ with Y pointed, and contains any

canonical point; and if Q is not a singleton $q : A \vdash B$, since τ is full the canonical copoint of $A \vdash B$ is uniquely determined by those of Q.

Case 2: A and B are atomic and d is $\{A \text{ --- } B\}$. Then τ must be $\{A \text{ --- } B\}$ and P and Q must be empty.

Case 3: $A = A_0 + A_1$ and $d = d_0 \cup d_1$, where d_i is a desaturation of $\tau_i = \tau|_{A_i \vdash B}$ for $i = 0, 1$. Let τ be a super-saturation of a net $\lambda : A \vdash B$. The net λ is equivalent to a copair of λ_0 and λ_1, by replacing each $A \text{---} 1$ with $A_0 \text{---} 1$ and $A_1 \text{ --- } 1$. By Proposition 44, the saturation σ of λ is obtained from the saturations of $\lambda_0 : A_0 \vdash B$ and $\lambda_1 : A_1 \vdash B$ by adding a link $A \text{ --- } 1$ whenever $A_0 \text{ --- } 1$ and $A_1 \text{ --- } 1$ are present. Since a link $A \text{ --- } 1$ is in neither σ_i, it follows that σ_i is the saturation of λ_i.

For each $p : A \vdash Y$ in P a there is a corresponding $p_i : A_i \vdash Y$ in σ_i, generating P_0 and P_1. Each $q : X \vdash B$ in Q is in one of σ_0 or σ_1, splitting Q into Q_0 and Q_1 (this is because $X \neq A$; otherwise τ would be copointed and full, and desaturation step 1 would apply instead of the present one). It follows that each $\tau_i = \tau|_{A_i \vdash B}$ is the P_i, Q_i-super-saturation of σ_i, and by induction we have desaturations d_0 and d_1. The saturation of $d = d_0 \cup d_1$ trivially contains the canonical copoints of $Q = Q_0 \cup Q_1$. That it contains the canonical point $p : A \vdash Y$ for each point in P follows by saturating the the copair $p_0 \cup p_1 : A \vdash Y$ of the canonical points $p_i : A_i \vdash Y$ in P_i.

Case 4: dual to case 3.

Case 5: $A = A_0 \times A_1$, τ factors through a projection π_i, and d is a desaturation d_i of $\tau|_{A_i \vdash B}$. Since also the saturation σ of λ factors through π_i, by Proposition 34 there is an equivalent net to λ that factors through π_i. Let this net factor as $\lambda_i : A_i \vdash B$, and have saturation σ_i.

Let σ be the union of σ_i, Γ, and Δ, as in Proposition 44. If A_i is copointed or A is not copointed then $\sigma_i = \sigma|_{A_i \vdash B}$, since neither Γ nor Δ introduces new links into the sequent $A_i \vdash B$ (see also [13, Lemma 4.6.1]). Next, all rooted copoints in Q must be in $A_i \vdash B$, since σ is not copointed and full (since σ is not pointed and full, it factors only through one projection). For each rooted point $p : A \vdash Y$ there is a corresponding $p_i : A_i \vdash Y$, forming P_i. Then $\tau|_{A_i \vdash B}$ is the Q, P_i-super-saturation of λ_i, giving the desaturation $d_i : A_i \vdash B$ by induction. It is immediate that the saturation of $d : A \vdash B$ contains the canonical copoint for each $q \in Q$, and by saturating the canonical point for p_i in P_i it contains also the canonical point for each p in P.

Finally, let A be copointed, but not A_i. Let P_i be the set of maximal pointed subnets in σ_i; then $\sigma|_{A_i \vdash B}$ is the union of σ_i with the full linking for each $p : A \vdash Y$ in P_i. Since each rooted copoint in Q is in σ_i, and each rooted point in P is dominated by a maximal one, $\tau_i = \tau|_{A_i \vdash B}$ is the Q, P_i-super-saturation of σ_i. (It can be shown by induction on σ_i that, since A_i is not copointed, the rooted points in P_i must be compatible, since they are maximal—see also [13, Lemma 4.6.3].) The induction hypothesis then gives the desaturation $d_i : A_i \vdash B$. The saturation of $d : A \vdash B$ contains the canonical copoints of Q, since they are contained in the saturation of d_i. It also contains the canonical points of P, since each rooted point in P is dominated by a full linking

$A \vdash Y$ generated by a maximal point $p_i : A_i Y$ in P_i, since A is copointed.

Case 6: dual to case 5. $\qquad\qquad\qquad\square$

Theorem 36 (restatement) A saturated net $\sigma : A \vdash B$ has a desaturation, and any desaturation of it saturates to σ.

Proof. By Lemma 47, taking the $\varnothing, \varnothing$-super-saturation of $\sigma : A \vdash B$. $\qquad\qquad\qquad\square$