Proof nets for additive linear logic with units

Willem Heijltjes
LFCS, School of Informatics, University of Edinburgh

***Abstract*—Additive linear logic, the fragment of linear logic concerning linear implication between strictly additive formulae, coincides with sum-product logic, the internal language of categories with free finite products and coproducts. Deciding equality of its proof terms, as imposed by the categorical laws, is complicated by the presence of the units (the initial and terminal objects of the category) and the fact that in a free setting products and coproducts do not distribute. The best known desicion algorithm, due to Cockett and Santocanale (CSL 2009), is highly involved, requiring an intricate case analysis on the syntax of terms.**

This paper provides canonical, graphical representations of the categorical morphisms, yielding a novel solution to this decision problem. Starting with (a modification of) existing proof nets, due to Hughes and Van Glabbeek, for additive linear logic without units, canonical forms are obtained by graph rewriting. The rewriting algorithm is remarkably simple. As a decision procedure for term equality it matches the known complexity of the problem. A main technical contribution of the paper is the substantial correctness proof of the algorithm.

I. Introduction

Proof nets, introduced by Girard in the seminal [6], are a beautiful, geometric description of linear logic proof. They aim to reproduce the qualities of the conjunction–implication fragment of intuitionistic natural deduction, that have made it into a prominent model of computation, via the Curry–Howard correspondence. The specifics of this motivation have been described in subtly different ways: proof nets are to remove the uninteresting, bureaucratic permutations from the cut-elimination procedure in the sequent calculus; to identify proofs that are 'morally' the same; or to obtain confluent cut-elimination, to name a few. A natural interpretation that subsumes those above, is that proof nets aim to be a canonical representation of proof in linear logic, with respect to its categorical semantics (see [15] and [1]). This means there is a 1–1 correspondence between proof nets and categorical morphisms, in the way that normal proofs in negative intuitionistic natural deduction uniquely describe morphisms in a free Cartesian closed category (see, e.g., [12]).

The original nets for linear logic were canonical, in this sense, only for the multiplicative connectives. It has proven exceedingly difficult to extend them to larger fragments, and in particular, to include the units, which are the neutral elements for the four binary connectives. A widely hailed success, after a partial result in [7], are the canonical proof nets for the combined multiplicative–additive fragment of linear logic, without the four units, in [9]. Successive approaches to including the multiplicative units are [2] and [13], and more recently [16] and [8]. Although some of the proposed representations have confluent normalisation, a treatment of the multiplicative units that is canonical with respect to the semantics provided by $*$-autonomous categories has remained out of reach.

This paper presents a new notion of proof net, for *additive linear logic*, the fragment of linear implication between additive formulae, including a canonical treatment of the two additive units, which have thus far not appeared in proof nets. To quote Girard, in [7, Appendix A.3]:

> There is still no satisfactory approach to additive neutrals [...].[1] The only way of handling $\top$ is by means of a box or, if one prefers, by means of a second order translation: on this Kamtchatka of linear logic, the old problems of sequent calculus are not fixed. The absence of a satisfactory treatment of $\top$ calls for another notion of proof-net...

Sum–product logic

Additive linear logic is also known as *sum–product logic*: it is the internal language of *sum–product categories*, categories that have all finite products and coproducts. Free sum–product categories are the finite, discrete versions of Joyal's free bicomplete categories [11], which are free completions with all limits and colimits. As such, free sum–product categories are characterised by the *softness* property described in that paper (and expanded on in [10]), a property related to cut-elimination and the subformula property in logic.

Cut-elimination for sum–product logic was investigated by Cockett and Seely in [4]. They show that it reduces the word problem—deciding equality of (proof) terms—for sum–product logic to a small set of simple permutations, making it decidable. However, this observation does not give a tractable algorithm, as the equivalence classes involved are exponentially sized. One factor complicating the search for an efficient decision procedure is the presence of the units, the categorical initial and terminal objects; omitting the units, a simple notion of proof identity via graphs is described in [5]. Another is the absence of distributivity; products distribute over coproducts in most situations where both occur, simplifying the problem considerably. In particular, no solution is provided by orientating the permutations as rewrites, a standard technique.

Recently, Cockett and Santocanale [3] have presented an intricate, polynomial-time decision algorithm for the word problem for sum–product logic.

[1]The original text reads, "...which are fortunately extremely uninteresting in practice." One can only guess at the reasons for questioning the significance of the additive units; after all, they are an integral part of linear logic, and in the opinion of the author, and probably that of others who have worked on them, pose a demanding challenge with interesting technical consequences.

$$\frac{a \in \hom(A,B)}{A \xrightarrow{a} B} \qquad \frac{X \xrightarrow{t_0} Y_0 \quad X \xrightarrow{t_1} Y_1}{X \xrightarrow{\langle t_0,t_1\rangle} Y_0 \times Y_1} \qquad \frac{X_i \xrightarrow{t} Y}{X_0 \times X_1 \xrightarrow{t\circ\pi_i} Y}$$

$$\frac{}{\mathbf{0} \xrightarrow{?} X} \qquad \frac{X_0 \xrightarrow{t_0} Y \quad X_1 \xrightarrow{t_1} Y}{X_0 + X_1 \xrightarrow{[t_0,t_1]} Y} \qquad \frac{X \xrightarrow{t} Y_i}{X \xrightarrow{\iota_i \circ t} Y_0 + Y_1}$$

$$\frac{}{X \xrightarrow{!} \mathbf{1}}$$

$$\frac{}{X \xrightarrow{id_X} X}\ \text{Id} \qquad \frac{X \xrightarrow{t} Y \quad Y \xrightarrow{s} Z}{X \xrightarrow{s\circ t} Z}\ \text{Cut}$$

Fig. 1. Sum–product logic

Sum–product nets

The proof nets presented in this paper provide graphical representations of proofs in additive linear logic, that are canonical with respect to the categorical semantics of free sum–product categories, constituting a novel solution to their word problem.

First, in Section II, the proof terms of sum–product logic are translated to *sum–product nets*, proof nets similar to the unit-free MALL-nets in [9]. These are canonical for the unit-free fragment, and factor out the permutations of that do not involve the units. This sharply isolates the challenge posed by the additive units, in the form of an equational theory over sum–product nets, induced by the remaining permutations.

This equational theory is addressed by rewriting to canonical forms. The rewrite algorithm, called *saturation*, is extremely simple, and is the first main contribution of the paper, presented in Section III. As a decision procedure—consisting of translating terms to proof nets, saturation, and testing for syntactic equality—it shares the polynomial time complexity of the decision algorithm by Cockett and Santocanale [3]. The correctness proof for the saturation algorithm is highly involved. This is the second main contribution of the paper, discussed in Section IV.

The nets obtained by saturation form a syntactic characterisation of free sum–product categories. Section V explores composition and identity morphisms in this context.

II. Sum–product nets

For the remainder, fix a category $\mathcal{C}$ and denote by $\Sigma\Pi(\mathcal{C})$ its free completion with products and coproducts (for an introduction to products and coproducts, see [14]). For the purposes of the present paper, $\Sigma\Pi(\mathcal{C})$ can be understood as a syntactic category generated by sum–product logic [4], which will be described below.

$$\iota_i \circ (t \circ \pi_j) = (\iota_i \circ t) \circ \pi_j \qquad ! = ! \circ \pi_i$$

$$\iota_i \circ [t,s] = [\iota_i \circ t, \iota_i \circ s] \qquad ! = [!,!]$$

$$\iota_i \circ ? = ?$$

$$\langle t \circ \pi_i, s \circ \pi_i\rangle = \langle t,s\rangle \circ \pi_i \qquad \langle ?,?\rangle = ?$$

$$\langle [t_0,t_1],[s_0,s_1]\rangle = [\langle t_0,s_0\rangle, \langle t_1,s_1\rangle] \qquad !_{\mathbf{0}} = ?_{\mathbf{1}}$$

Fig. 2. Permutations in sum–product logic

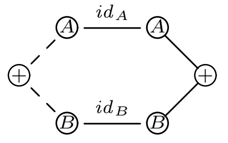
The objects in $\Sigma\Pi(\mathcal{C})$, ranged over by variables X, Y, Z, are given by the grammar

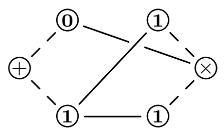
$$X \ ::= \ A \in \mathcal{C} \mid \mathbf{0} \mid \mathbf{1} \mid X + X \mid X \times X \ .$$

The sequent calculus presentation of sum–product logic, illustrated in Figure 1, provides a term calculus for the morphisms in $\Sigma\Pi(\mathcal{C})$, in which the cut-rule and identity-rule are admissible. The permutations in Figure 2 form an equational theory over proofs in sum–product logic. In the syntactic description of the category, the morphisms of $\Sigma\Pi(\mathcal{C})$ are the equivalence classes of (cut-free, identity-free) proofs [4].

The notation for the connectives was chosen to agree with that of [4], and is natural given the intended interpretation as categorical products and coproducts. To retrieve the notation of linear logic, interpret the unit $\mathbf{1}$ as $\top$, and the connectives $+$ and $\times$ respectively as $\oplus$ and $\&$. The case of additive linear logic is given by choosing a discrete base category $\mathcal{C}$ (one with only identity morphisms).

A sum–product net will consist of a source object and a target object from the category $\Sigma\Pi(\mathcal{C})$, and a collection of *links* connecting vertices in the syntax tree of the former to vertices in that of the latter. Two example nets are drawn below, together with the terms they represent.





$[(\iota_0 \circ id_A), (\iota_1 \circ id_B)]$ $\qquad$ $[?, \langle !,! \rangle]$

Interpreted informally, nets are to be read from left to right. Solid edges in the object trees correspond to projections and injections, while dashed edges correspond, roughly, to (the application of) the inference rules $\langle -,- \rangle$ and $[-,-]$. Links correspond to axioms, and are distinguished from solid edges in the object trees by being slightly detached from vertices.

For the formal definition, the *vertices* (or *positions*) in the syntax tree of an object X are given as binary words, with ε the empty word and $(\leq)$ the standard prefix ordering, and collected in the set $\mathrm{pos}(X)$. The subformula of X at a vertex v is denoted X_v, and 'v is Y' will mean $X_v = Y$ when X is understood. In this definition, if v is a product or coproduct it has children $v0$ and $v1$, and none otherwise.

Definition 1 (Pre-nets)**.** A $\Sigma\Pi(\mathcal{C})$*-pre-net* $(X, Y, \mathcal{R})$ consists of a *source* object X, a *target* object Y and a relation

$$\mathcal{R} \quad \subseteq \quad \mathrm{pos}(X) \ \times \ \big(\hom(\mathcal{C}) \uplus \{*\}\big) \ \times \ \mathrm{pos}(Y)$$

such that for any $\langle v, l, w\rangle \in \mathcal{R}$, if $l = *$ then $X_v = \mathbf{0}$ or $Y_w = \mathbf{1}$, and otherwise $l \in \mathcal{C}(X_v, Y_w)$.

Variables f, g, h and k are used for pre-nets. The *links* in a pre-net are the elements $\langle v, l, w\rangle$ of $\mathcal{R}$, which may be rendered $\langle v, w\rangle$ when the label l is understood or irrelevant. A link $\langle v, *, w\rangle$ (the label $*$ will be omitted from diagrams) is a *unit* link; if v is $\mathbf{0}$ it is an *initial* link, if w is $\mathbf{1}$ a *terminal* link. A link labelled with a $\mathcal{C}$-morphism is *atomic*.

A *switching* ς of an object X is a function choosing one branch of each product vertex: $\varsigma(v) \in \{0,1\}$ if X_v is a product, while otherwise $\varsigma(v)$ is undefined. The dual notion of a *co-switching* is a function choosing branches on coproduct vertices. A vertex w is *switched on* by a [co-]switching ς, written $\varsigma \mathrel{\text{⏻}} w$, if for any ancestor (i.e. prefix) of w that is a [co]product, ς selects the branch containing w;

$$\varsigma \mathrel{\text{⏻}} w \quad \overset{\triangle}{\iff} \quad \big(vi \leq w \;\wedge\; v \in \mathrm{dom}(\varsigma)\big) \;\Rightarrow\; \varsigma(v) = i \ .$$

A switching for a pre-net $(X, Y, \mathcal{R})$ is a pair (ς, τ) of a co-switching ς of X and a switching τ of Y. A link $\langle v, w\rangle$ is *switched on* by (ς, τ) if $\varsigma \mathrel{\text{⏻}} v$ and $\tau \mathrel{\text{⏻}} w$.

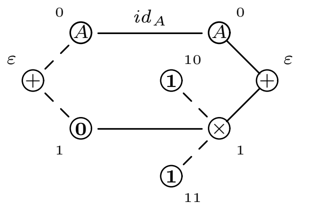
Definition 2 (Nets)**.** A $\Sigma\Pi(\mathcal{C})$*-net* is a pre-net f that satisfies the following correctness criterion (the *switching condition*).

- Any switching (ς, τ) for f switches on precisely one link.

Let NET denote the set of all $\Sigma\Pi(\mathcal{C})$-nets.

Sum–product nets without units form the purely additive fragment of the MALL-nets by Hughes and Van Glabbeek [9, Section 4.10]. The addition of unit links has only minor technical consequences, which are due to the fact that unlike atomic links, unit links may connect to non-leaf nodes.

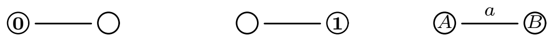
How diagrams and definitions relate is illustrated below.



$$(A + \mathbf{0},\ A + (\mathbf{1} \times \mathbf{1}),\ \{\, \langle 0, id_A, 0\rangle,\ \langle 1, *, 1\rangle \,\}\ )$$

The edges of nodes subject to (co-)switchings (in the switching condition) are dashed. The switching condition can be separated into an at–least–one and an at–most–one part. A pre-net satisfying the latter, i.e. one for which any switching (ς, τ) switches on at most one link, is called a *partial net*. (An insightful way of gaining familiarity with the switching condition is by convincing oneself that there are no natural nets from $A \times (B + C)$ to $(A \times B) + (A \times C)$, showing that sum and product do not distribute.)

The proof terms of sum–product logic suggest an inductive construction method for sum–product nets. Using the abbreviation (X, Y, l) for $(X, Y, \{\langle \varepsilon, l, \varepsilon\rangle\})$, there are *basic* nets



$$?_Y \ \triangleq\ (\mathbf{0}, Y, *) \qquad !_X \ \triangleq\ (X, \mathbf{1}, *) \qquad (A, B, a)$$

for each $X, Y \in \Sigma\Pi(\mathcal{C})$ and $a \in \mathcal{C}(A, B)$, corresponding to the axioms—note the upright font, to contrast with terms. Inference rules coincide with the four *constructors*,

$$(\pi_i(X);-) \qquad [-,-] \qquad \langle -,-\rangle \qquad (-;\iota_j(Y))\ ,$$

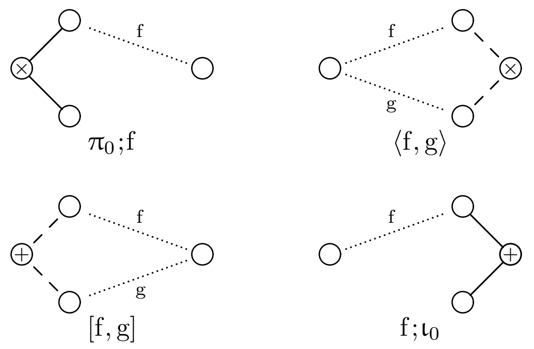


Fig. 3. Net constructors

illustrated in Figure 3. Using the notation

$$\begin{aligned} u \cdot \mathcal{R} \ &\triangleq\ \{\langle uv, l, w\rangle \mid \langle v, l, w\rangle \in \mathcal{R}\} \\ \mathcal{R} \cdot u \ &\triangleq\ \{\langle v, l, uw\rangle \mid \langle v, l, w\rangle \in \mathcal{R}\}\ , \end{aligned}$$

the constructors are defined, on pre-nets, by

$$\pi_i(X_0 \times X_1);(X_i, Y, \mathcal{R}) \triangleq (X_0 \times X_1, Y,\ i \cdot \mathcal{R})$$

$$[(X, Z, \mathcal{R}), (Y, Z, \mathcal{S})] \triangleq (X + Y, Z, (0 \cdot \mathcal{R}) \cup (1 \cdot \mathcal{S}))$$

$$\langle (X, Y, \mathcal{R}), (X, Z, \mathcal{S})\rangle \triangleq (X, Y \times Z, (\mathcal{R} \cdot 0) \cup (\mathcal{S} \cdot 1))$$

$$(X, Y_i, \mathcal{R});\iota_i(Y_0 + Y_1) \triangleq (X, Y_0 + Y_1,\ \mathcal{R} \cdot 0)\ .$$

The translation from (cut-free) proof terms to nets, implicit in the naming of constructors, is made explicit as $[\![-]\!]$ below.

$$[\![?_Y]\!] = ?_Y \qquad [\![!_X]\!] = !_X \qquad [\![a : A \to B]\!] = (A, B, a)$$

$$[\![t \circ \pi_i]\!] = \pi_i;[\![t]\!] \qquad [\![\langle t, s\rangle]\!] = \langle [\![t]\!], [\![s]\!]\rangle$$

$$[\![[t, s]]\!] = [\,[\![t]\!], [\![s]\!]\,] \qquad [\![\iota_j \circ t]\!] = [\![t]\!];\iota_j$$

Applying a constructor is called *construction*, the reverse *deconstruction*. A pre-net of the form π_i;f or [f, g] is *left-constructible*; one of the form $\langle$f, g$\rangle$ or f$;\iota_i$ *right-constructible*; one that is either, *constructible*; and one that is both, *bi-constructible*. Both construction and deconstruction preserve the switching condition, and moreover, all nets are basic or constructible. This gives the *sequentialisation* result below, which states that all nets correspond to some term.

Proposition 3. NET *is the smallest set containing all basic nets, closed under construction.*

(This is a minor variant on the analogous result by Hughes and Van Glabbeek in [9].)

For bi-constructible pre-nets the following equations are immediate from the definitions, as illustrated in Figure 4.

Proposition 4. *Construction of pre-nets satisfies:*

$$(\pi_i;\mathrm{f});\iota_j \;=\; \pi_i;(\mathrm{f};\iota_j) \qquad \langle [\mathrm{f}, \mathrm{h}], [\mathrm{g}, \mathrm{k}]\rangle \;=\; [\langle \mathrm{f}, \mathrm{g}\rangle, \langle \mathrm{h}, \mathrm{k}\rangle]$$

$$[\mathrm{f}, \mathrm{g}];\iota_j \;=\; [(\mathrm{f};\iota_j), (\mathrm{g};\iota_j)] \qquad \langle (\pi_i;\mathrm{f}), (\pi_i;\mathrm{g})\rangle \;=\; \pi_i;\langle \mathrm{f}, \mathrm{g}\rangle$$

These equations correspond to the four equations on the left in Figure 2; precisely those not involving the units.

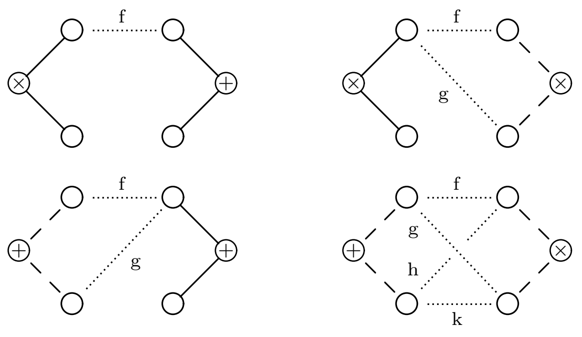


Fig. 4. Bi-constructible nets

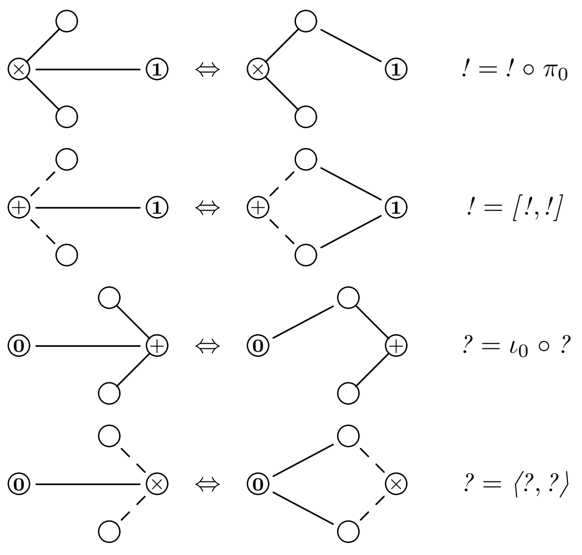


Fig. 5. The unit laws force an equational theory over nets

Of the remaining equations in Figure 2, those involving the units, only $?_{\mathbf{1}} = !_{\mathbf{0}}$ is factored out (by labelling initial and terminal links uniformly). The other four will form an equational theory over nets, *equivalence* ($\Leftrightarrow$), illustrated in Figure 5. The natural way of defining it is via graph-rewriting, as rewrite rules that replace one subnet with another—which firstly requires a notion of subnet. In the general notion a sub-pre-net of $(X, Y, \mathcal{R})$ will mean a pre-net between subformulae of X and Y, with a subcollection of the links between them: a pre-net $(X_v, Y_w, \mathcal{S})$ such that $v \cdot \mathcal{S} \cdot w \subseteq \mathcal{R}$. Call two pre-nets *parallel* if they have the same source objects and the same target objects, and define, on parallel pre-nets,

$$(X, Y, \mathcal{S}) \subseteq (X, Y, \mathcal{R}) \quad \stackrel{\Delta}{\iff} \quad \mathcal{S} \subseteq \mathcal{R} \ ,$$

and, for a pre-net $\mathrm{f} = (X, Y, \mathcal{R})$,

$$\begin{aligned} \mathcal{R}_{v,w} \quad &\triangleq \quad \{\langle v', l, w' \rangle \mid \langle vv', l, ww' \rangle \in \mathcal{R}\} \\ \mathrm{f}_{v,w} \quad &\triangleq \quad (X_v, Y_w,\ \mathcal{R}_{v,w}) \ . \end{aligned}$$

Definition 5 (Subnets)**.** A *sub-pre-net* of a pre-net f is a pre-net $\mathrm{g} \subseteq \mathrm{f}_{v,w}$. If g is a net, it is a *subnet* of f.

The notation $\mathrm{f}\{\mathrm{g}\}_{v,w}$ denotes a pre-net f with the sub-pre-net $\mathrm{f}_{v,w}$ replaced by a parallel pre-net g. Formally, for pre-nets $\mathrm{f} = (X, Y, \mathcal{R})$ and $\mathrm{g} = (X_v, Y_w, \mathcal{S})$, define the following.

$$\mathcal{R}\{\mathcal{S}\}_{v,w} \triangleq \{\langle v', l, w' \rangle \in \mathcal{R} \mid v \not\leq v' \vee w \not\leq w'\} \cup (v \cdot \mathcal{S} \cdot w)$$
$$\mathrm{f}\{\mathrm{g}\}_{v,w} \triangleq (X, Y,\ \mathcal{R}\{\mathcal{S}\}_{v,w})$$

The general form of rewriting in context is given by the following relation.

$$\mathrm{f}\{\mathrm{g}\}_{v,w} \ \ \dashv\!\![\, \mathrm{g} \mid \mathrm{h} \,]\!\!\Rightarrow_{v,w} \ \mathrm{f}\{\mathrm{h}\}_{v,w}$$

The relation $\dashv\!\![\, \mathrm{g} \mid \mathrm{h} \,]\!\!\Rightarrow_{v,w}$ replaces the pre-net between vertices v and w, which is required to be g, with the parallel pre-net h, leaving the context intact. An equivalent formulation would be $\mathrm{f} \dashv\!\![\, \mathrm{f}_{v,w} \mid \mathrm{h} \,]\!\!\Rightarrow_{v,w} \mathrm{f}\{\mathrm{h}\}_{v,w}$. Dropping the subscript v, w indicates the union over all v and w.

Definition 6 (Equivalence)**.** The equational theory $\Leftrightarrow$ (*equivalence*) on $\Sigma\Pi$-nets is the equivalence relation generated by the following four relations.

$$\dashv\!\![\, ! \mid \pi_i ; ! \,]\!\!\Rightarrow \quad \dashv\!\![\, ! \mid [!, !] \,]\!\!\Rightarrow \quad \dashv\!\![\, ? \mid \langle ?, ? \rangle \,]\!\!\Rightarrow \quad \dashv\!\![\, ? \mid ? ; \iota_j \,]\!\!\Rightarrow$$

These four rewrite rules are the equivalences illustrated in Figure 5, interpreted as rewrite steps from left to right, on subnets. With some effort, it then follows from the description of free sum–product categories by Cockett and Seely in [4], that $\Sigma\Pi$-nets up to equivalence, too, characterise free sum–product categories.

Proposition 7. *For cut-free proof terms t and s of sum–product logic,*

$$\Sigma\Pi(C) \models t = s \quad \iff \quad [\![t]\!] \Leftrightarrow [\![s]\!] \ .$$

III. Deciding equivalence of nets

The equivalence relation over nets will be decided by rewriting equivalent nets to a common canonical form. A natural first question is whether a suitable, confluent rewrite relation can be obtained by orientating the equivalence rewrites, i.e. by restricting them to one direction. Two straightforward candidates are to rewrite towards the leaves or towards the the roots of the trees. In fact, neither option is confluent. For the first, an example of non-confluence is illustrated in Figure 6. For the second option, the situation is more delicate. The non-confluent example in Figure 7 could in principle be resolved by introducing a novel kind of link connecting both root nodes, while preserving the switching condition as the correctness criterion for nets. However, the non-confluence of the example in Figure 8 has no solution along these lines.

If confluent rewriting is impossible without breaking the switching condition, the obvious next step is to break it. Then when two nets rewrite into each other, the easiest way to obtain confluence is to combine the links of both, as in the example of Figure 9. This gives a simple rewrite relation called *saturation*.

To define the saturation relation a different form of rewriting is required, whereby links are added to a net, rather than

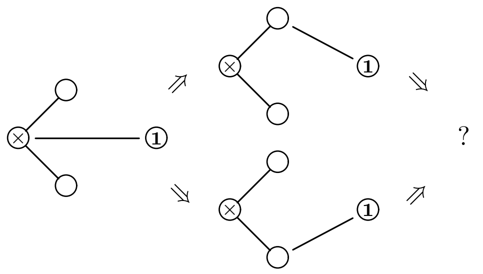


Fig. 6. Rewriting towards the leaves is non-confluent

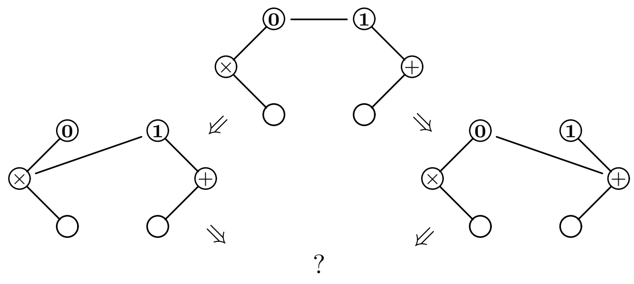


Fig. 7. Rewriting towards the roots is non-confluent (1)

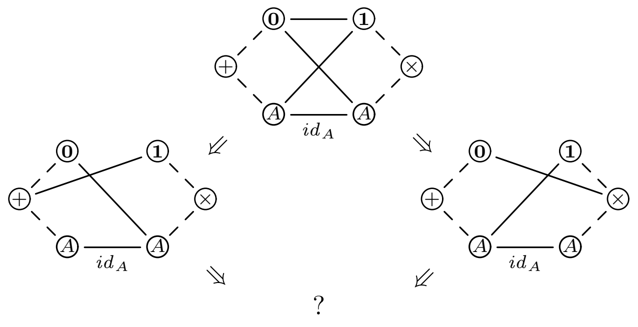


Fig. 8. Rewriting towards the roots is non-confluent (2)

replaced. Let the *union* of two parallel pre-nets be the union of their collections of links,

$$(X, Y, \mathcal{R}) \cup (X, Y, \mathcal{S}) \;\triangleq\; (X, Y, \mathcal{R} \cup \mathcal{S}) \; .$$

A statement $\mathrm{f} = \mathrm{f}\{\mathrm{f}' \cup \mathrm{g}\}_{v,w}$ then expresses the condition that a pre-net f must have g as a subnet, $\mathrm{g} \subseteq \mathrm{f}_{v,w}$ (where f' holds the other links in $\mathrm{f}_{v,w}$). Define a second rewrite relation:

$$\mathrm{f} \;\; \prec(\mathrm{g} \,|\, \mathrm{h})\!\rightarrowtail_{v,w} \;\; \mathrm{f}\{\mathrm{f}_{v,w} \cup \mathrm{h}\}_{v,w} \qquad \text{if } \mathrm{g} \subseteq \mathrm{f}_{v,w} \; .$$

Definition 8. The *saturation* relation $\rightarrowtail$ on pre-nets is the union of the following eight relations.

$$\begin{array}{cccc}
\prec(\pi_i;!\,|\,!)\!\rightarrowtail & \prec([!,!]\,|\,!)\!\rightarrowtail & \prec(\langle ?,? \rangle\,|\,?)\!\rightarrowtail & \prec(?;\iota_j\,|\,?)\!\rightarrowtail \\
\prec(!\,|\,\pi_i;!)\!\rightarrowtail & \prec(!\,|\,[!,!])\!\rightarrowtail & \prec(?\,|\,\langle ?,? \rangle)\!\rightarrowtail & \prec(?\,|\,?;\iota_j)\!\rightarrowtail
\end{array}$$

The relation $\rightarrowtail^{-}$ is the irreflexive restriction of $\rightarrowtail$.

The eight *saturation steps* in Definition 8 are illustrated in Figure 10. In general, the relation $\prec(\mathrm{g}\,|\,\mathrm{h})\!\rightarrowtail_{v,w}$ is reflexive for nets that already have h (and g) as a subnet between vertices

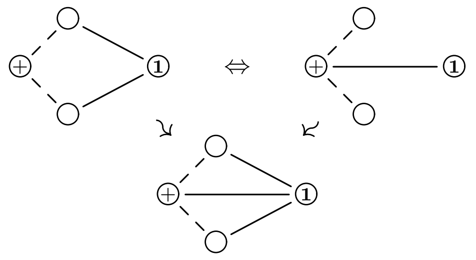


Fig. 9. Saturation

v and w. In order to provide saturation with a standard notion of termination, the irreflexive variant $\rightarrowtail^{-}$ is defined. Both $\rightarrowtail$ and $\rightarrowtail^{-}$ will be referred to as saturation, with the distinction only made when necessary.

Proposition 9. *The saturation relation ($\rightarrowtail^{-}$) is confluent and strongly normalising.*

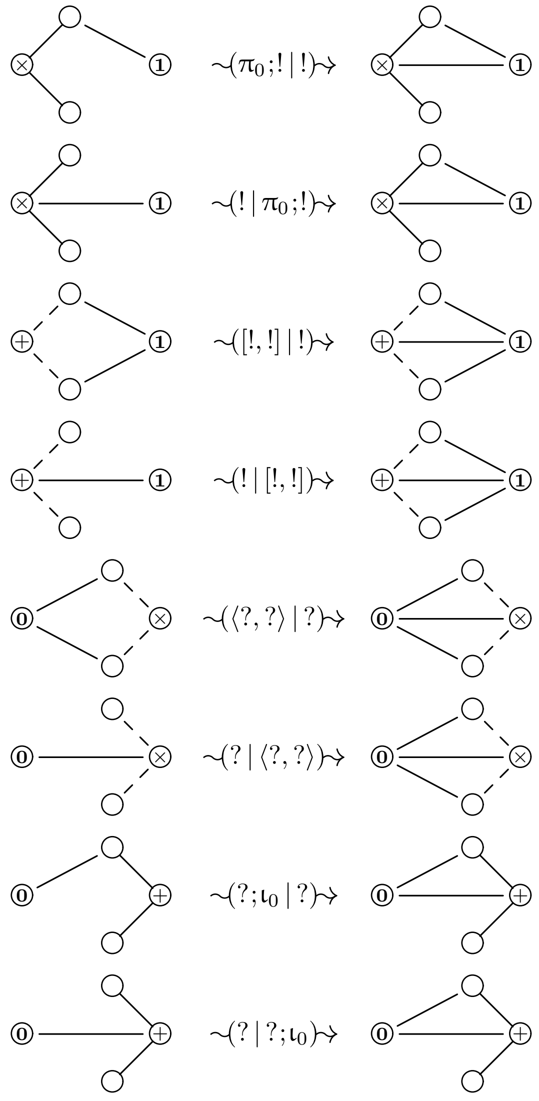

$\prec(\pi_0;!\,|\,!)\!\rightarrowtail$
$\prec(!\,|\,\pi_0;!)\!\rightarrowtail$
$\prec([!,!]\,|\,!)\!\rightarrowtail$
$\prec(!\,|\,[!,!])\!\rightarrowtail$
$\prec(\langle ?,? \rangle\,|\,?)\!\rightarrowtail$
$\prec(?\,|\,\langle ?,? \rangle)\!\rightarrowtail$
$\prec(?;\iota_0\,|\,?)\!\rightarrowtail$
$\prec(?\,|\,?;\iota_0)\!\rightarrowtail$

Fig. 10. Saturation steps

Proof: For strong normalisation it is sufficient to observe that each step in $\rightsquigarrow^{-}$ adds one or two unit links to a pre-net, while the number of unit links in a pre-net $(X, Y, \mathcal{R})$ is bounded by the size of $\mathrm{pos}(X) \times \mathrm{pos}(Y)$.

For confluence, let $\mathrm{f} = (X, Y, \mathcal{R})$, let $\mathrm{g}' = (X_v, Y_w, \mathcal{S})$, and let $\mathrm{h}' = (X_x, Y_y, \mathcal{T})$. Observe that the result of applying a saturation step $\prec\!(\mathrm{g} \,|\, \mathrm{g}')\!\rightsquigarrow_{v,w}$ to f is just

$$\mathrm{f}\{\mathrm{f}_{v,w} \cup \mathrm{g}'\}_{v,w} \quad = \quad (X, Y,\ \mathcal{R}\ \cup\ v \cdot \mathcal{S} \cdot w)\ .$$

The following diagram shows local confluence for $\rightsquigarrow$.

$$\begin{tikzcd}
 & (X,Y,\mathcal{R}) \arrow[dl, rightsquigarrow, "(\mathrm{g}\,|\,\mathrm{g}')_{v,w}"'] \arrow[dr, rightsquigarrow, "(\mathrm{h}\,|\,\mathrm{h}')_{x,y}"] & \\
(X,Y,\mathcal{R}\ \cup\ v\cdot\mathcal{S}\cdot w) \arrow[dr, rightsquigarrow, "(\mathrm{h}\,|\,\mathrm{h}')_{x,y}"'] & & (X,Y,\mathcal{R}\ \cup\ x\cdot\mathcal{T}\cdot y) \arrow[dl, rightsquigarrow, "(\mathrm{g}\,|\,\mathrm{g}')_{v,w}"] \\
 & (X,Y,\mathcal{R}\ \cup\ v\cdot\mathcal{S}\cdot w\ \cup\ x\cdot\mathcal{T}\cdot y) &
\end{tikzcd}$$

Then also $\rightsquigarrow^{-}$ is locally confluent, and in the context of strong normalisation this implies $\rightsquigarrow^{-}$ is confluent. ■

The normal form of a pre-net f w.r.t. $\rightsquigarrow^{-}$ is denoted $\sigma\mathrm{f}$, and, if f is a net, is called a *saturated net*. The idea is that saturation provides a decision procedure by comparing saturated nets, i.e. $\mathrm{f} \Leftrightarrow \mathrm{g}$ if and only if $\sigma\mathrm{f} = \sigma\mathrm{g}$.

Theorem 10 (Completeness)**.** *For nets* f *and* g*, if* $\mathrm{f} \Leftrightarrow \mathrm{g}$ *then* $\sigma\mathrm{f} = \sigma\mathrm{g}$.

The completeness proof of this decision procedure is straightforward: two nets $\mathrm{f} \Leftrightarrow \mathrm{g}$ that are equivalent by a single rewrite step have saturation steps $\mathrm{f} \rightsquigarrow \mathrm{h}$ and $\mathrm{g} \rightsquigarrow \mathrm{h}$ with a common target h; the statement then follows from confluence. The soundness theorem is stated below; its elaborate proof will be the subject of the next section.

Theorem 11 (Soundness)**.** *For nets* f *and* g*, if* $\sigma\mathrm{f} = \sigma\mathrm{g}$ *then* $\mathrm{f} \Leftrightarrow \mathrm{g}$.

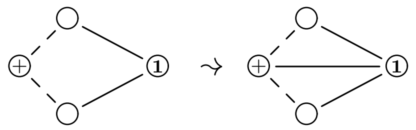
IV. The soundness proof

A natural approach to proving the soundness theorem would be by induction on the saturation paths of $\sigma\mathrm{f}$ and $\sigma\mathrm{g}$; e.g. for f this is a sequence

$$\mathrm{f} \ \rightsquigarrow\ \mathrm{f}_1 \ \rightsquigarrow\ \mathrm{f}_2 \ \rightsquigarrow\ \dots \ \rightsquigarrow\ \sigma\mathrm{f}\ .$$

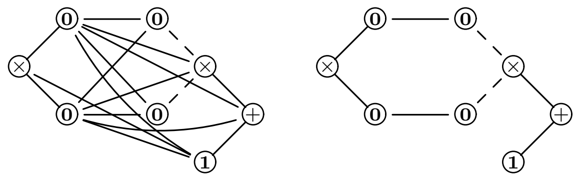
One could imagine a proof to proceed as follows. Each stage f_k in the saturation path would be taken to represent a collection of equivalent nets, by their union, with f representing just itself. A saturation step $\mathrm{f}_k \ \prec\!(\mathrm{g} \,|\, \mathrm{h})\!\rightsquigarrow_{v,w}\ \mathrm{f}_{k+1}$ would then create the collection of f_{k+1} by closing that of f_k under the corresponding rewrite step, $=\!\![\,\mathrm{g} \,|\, \mathrm{h}\,]\!\!\Rightarrow_{v,w}$. To complete the argument, it would suffice to show that if $\sigma\mathrm{f} = \sigma\mathrm{g}$ then the collections of equivalent nets that both saturations represent, which contain f and g respectively, share at least one net.

A proof along these lines faces several obstacles, mainly in the form of statements that are true, but hard to prove in the proposed way. The first is that a saturated net $\sigma\mathrm{f}$ is the union of all nets equivalent to f—a fact that will follow from aspects of the eventual soundness proof, and will have a separate use in characterising composition of saturated nets. Despite being naturally suggested by the proof idea above, the statement is difficult to prove in the way suggested. In particular, that every link in $\sigma\mathrm{f}$ belongs to some net equivalent to f would follow by induction on the saturation path, if not for the saturation step below (and its dual).



The left-hand side of this step contains two links; even if each occurs in some net equivalent to f, for the corresponding equivalence step to apply, both links must occur together in a single net, and it is not obvious how to show this is the case.

One approach to the above problem would be to characterise, for a saturated net, the nets whose saturation it is; call such nets *representatives* of the saturated net. A simple potential answer, that any subnet of a saturated net would be a representative, turns out to be false: the saturated net below left (the saturation of $!;\iota_1$) has that on the right as a subnet, but the latter is already saturated.



In fact, no representative of the saturated net on the left contains both links from the net on the right.

The main difficulty of this proof idea, however, is the final step it suggests, of showing that the equivalence classes represented by $\sigma\mathrm{f}$ and $\sigma\mathrm{g}$ overlap. Without a characterisation of representatives, there are not many immediate indications left on how this should be approached.

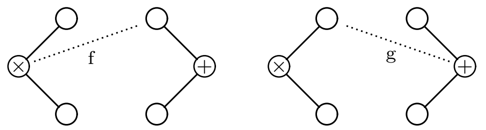
The previous served to illustrate how saturation is difficult to characterise, and prove properties of, via saturation paths alone. Instead, therefore, the soundness proof will proceed by induction on the source and target objects of nets, and rely on a different description of saturation, which follows the construction of a net. For space reasons many technical details are necessarily omitted.

As a first overview, there will be three cases, for nets f and g with the same saturation $(X, Y, \mathcal{R})$:

- one of X and Y is an atom or unit,
- X is a coproduct or Y is a product, and
- X is a product and Y a coproduct.

The first two cases are relatively straightforward, and will be treated in the next subsection. The main body of the proof is concerned with the third case, which is that of nets $\mathrm{f};\iota_j$ and

$\pi_i;g$ as illustrated below.



There are three main obstacles to overcome.

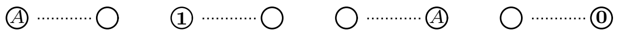
1) Inductive saturation: To apply the induction hypothesis it must be possible to relate, e.g., a saturated net $\sigma(f;\iota_0)$, to the saturation of its component net, σf. This will be addressed by Lemma 18, which describes saturated nets σf inductively, on the construction of f. Subsection IV-B presents supporting material for the lemma, whose main content will be discussed in Subsection IV-C.

2) Representatives: The second obstacle is that nets constructed over different projections and injections, e.g. $f;\iota_0$ and $\pi_0;g$, but also $f;\iota_0$ and $g;\iota_1$, may have the same saturation; naturally, in such a case the induction hypothesis cannot be applied to σf and σg. This will be solved by Lemma 19, which, for a saturated net σf, finds a representative g equivalent to f containing given initial links $\langle v, \varepsilon\rangle$, or terminal links $\langle \varepsilon, w\rangle$, from σf. From the presence of these links it can then be deduced that g is left-constructible or right-constructible, respectively, and over which projection or injection it is constructed. This is described in Subsection IV-D.

3) Deconstruction alone may not suffice: The third obstacle is that nets constructed over the same projection or injection, e.g. $f;\iota_0$ and $g;\iota_0$, may have the same saturation, while their components, f and g, do not. By isolating the exact cause of this discrepancy it will be possible to transform the net $f;\iota_0$ into an equivalent net $h;\iota_0$, such that h does have the same saturation as g. This will be treated in Subsection IV-E.

A. The first two cases

The first case of the soundness proof concerns parallel nets whose source or target is an atom or unit. For nets with source X and target Y, this gives six, pairwise dual, possibilities. Four are immediate: if X is an atom or $\mathbf{1}$, or dually if Y is an atom or $\mathbf{0}$, illustrated below, it is easily observed that no rewrite or saturation steps apply.



For such nets f and g, it follows that if $\sigma f = \sigma g$ then $f = g$.

For the remaining two cases, nets with source object $\mathbf{0}$ will be called *initial*, and with target $\mathbf{1}$, *terminal*. The links in an initial net $(\mathbf{0}, Y, \mathcal{R})$ can move up and down the syntax tree of Y essentially without hindrance. From this Lemma 12 and Lemma 13, below, follow.

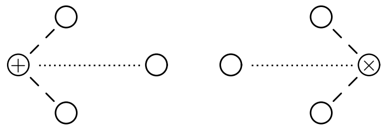
Lemma 12. *All parallel initial nets are equivalent, as are all parallel terminal nets.*

This proves soundness of saturation for initial and terminal nets, and although it need not refer to saturated nets, their characterisation will be useful. Call a pre-net *full* if it contains all possible unit links (but no atomic links), i.e. one of the form

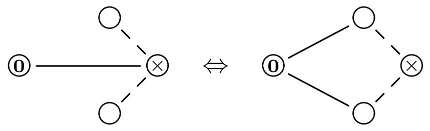
$$(X, Y, \{\langle v, *, w\rangle \mid X_v = \mathbf{0} \text{ or } Y_w = \mathbf{1}\})\ .$$

Lemma 13. *The saturation of initial and terminal nets is full.*

The second case of the soundness proof concerns nets whose source is a coproduct or whose target is a product; call these *coproduct nets* and *product nets*, respectively.

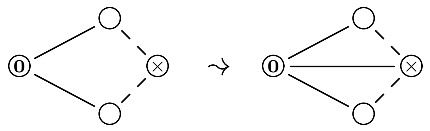


A product net is right-constructible unless it contains links $\langle v, \varepsilon\rangle$ connecting to the root of its target. Such a link must be an initial link, and can be moved away from the root by the following equivalence step. This gives the lemma below.

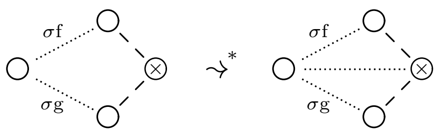


Lemma 14. *A product net* g *is equivalent to a net* $\langle g_0, g_1\rangle$. *A coproduct net* f *is equivalent to a net* $[f_0, f_1]$.

A similar result holds for the saturation of (co)product nets. For a net $\langle f, g\rangle$, if first the saturation steps in f and g are applied, the remaining steps to be applied to $\langle \sigma f, \sigma g\rangle$ are of the following kind.



Applying these steps completes the saturation of $\langle f, g\rangle$: after a step of the kind above, to the newly added link no saturation steps apply, as the only possible step would be the reverse. Moreover, the links added by these steps are all of the form $\langle v, \varepsilon\rangle$ and thus, in $\sigma\langle f, g\rangle$, illustrated below right, separate from links in σf and σg.



Lemma 15. *Saturation of (co)product nets satisfies:*

$$(\sigma[f_0, f_1])_{i,\varepsilon} = \sigma f_i \qquad (\sigma\langle g_0, g_1\rangle)_{\varepsilon,i} = \sigma g_i\ .$$

Using the two lemmata on (co)product nets, the present case in the soundness proof will be completed. For parallel product nets f and g with the same saturation, Lemma 14 gives equivalent nets $\langle f_0, f_1\rangle$ and $\langle g_0, g_1\rangle$ respectively. By Lemma 15

$$\sigma f_i = (\sigma\langle f_0, f_1\rangle)_{\varepsilon,i} = (\sigma\langle g_0, g_1\rangle)_{\varepsilon,i} = \sigma g_i$$

for $i \in \{0, 1\}$. The induction hypothesis of the soundness proof gives $f_i \Leftrightarrow g_i$, and the equivalences below follow.

$$f \Leftrightarrow \langle f_0, f_1\rangle \Leftrightarrow \langle g_0, g_1\rangle \Leftrightarrow g$$

B. Pointed and copointed nets

A *point* is a map out of a terminal object, a *copoint* one into an initial object. An object that has a [co]point is [*co*]*pointed*. In free sum–product categories the pointed and copointed objects are given by the following grammars, respectively.

$$P \ := \ \mathbf{1} \mid P + X \mid X + P \mid P \times P$$

$$Q \ := \ \mathbf{0} \mid Q + Q \mid Q \times X \mid X \times Q$$

Here, X may be any object: pointed, copointed, or neither. Note that an object is never both pointed and copointed, and that in $\Sigma\Pi(\varnothing)$, the free sum–product completion of the empty category, where atoms are absent, every object is either pointed or copointed.

A categorical map that factors through a point, i.e. one of the form $p \circ !$, where p is a point, is called *pointed*; one that factors through a copoint, $? \circ q$, *copointed*. *Pointed* and *copointed* nets are defined slightly more narrowly, by the following grammars over the net constructors.

$$\mathrm{p} \ := \ ! \mid \mathrm{p};\iota_j \mid \langle \mathrm{p}, \mathrm{p} \rangle \qquad\qquad \mathrm{q} \ := \ ? \mid [\mathrm{q}, \mathrm{q}] \mid \pi_i;\mathrm{q} \ .$$

Up to equivalence, the definition corresponds to the categorical one, but it requires pointed and copointed nets to have the following syntactic form. Call a terminal link $\langle \varepsilon, *, w \rangle$ and an initial link $\langle v, *, \varepsilon \rangle$ *rooted*; pointed nets are those consisting entirely of rooted terminal links, and copointed nets those consisting of rooted initial links.

A map that is both pointed and copointed will be called *bipointed*. Bipointed maps feature heavily in the decision procedure of Cockett and Santocanale [3]—where they are called *disconnects*—because of the following property: there is precisely one bipointed map from a copointed object Q to a pointed object P, and none between other objects. The uniqueness property is easily observed from the fact that in the diagram below the copoint q and the point p are arbitrary.

$$\begin{tikzcd}
Q \arrow[r, "q"'] \arrow[rr, bend left, "!"] & \mathbf{0} \arrow[r, "!"] \arrow[r, "?"'] \arrow[rr, bend right, "?"'] & \mathbf{1} \arrow[r, "p"] & P
\end{tikzcd}$$

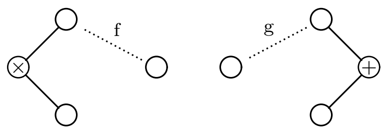
The corresponding notion for nets will again be restricted to a useful syntactic form: let a *bipointed* net be a net from a copointed to a pointed object that is itself pointed or copointed. The properties of bipointed maps carry over, by the following two lemmata.

Lemma 16. *Any two parallel bipointed nets are equivalent.*

Lemma 17. *The saturation of a bipointed net is full.*

C. Saturation via construction

Previous lemmata showed that the saturation of initial, terminal, and bipointed nets is full (Lemma 13 and 17). The next result will be that saturation is completely described by this dynamic, i.e. by the 'filling' of initial, terminal, and bipointed subnets. This enables, and is proved via, a characterisation of saturation by induction on the construction of a net. For the base cases, saturation of the basic nets $(\mathbf{0}, Y, *)$ and $(X, \mathbf{1}, *)$ is full, by Lemma 13, while no saturation steps apply to a net (A, B, a). The saturation of nets $\langle \mathrm{g}, \mathrm{h} \rangle$ and $[\mathrm{g}, \mathrm{h}]$ was described informally in Subsection IV-A, leaving the cases $\pi_i;\mathrm{f}$ and $\mathrm{g};\iota_j$ (illustrated below for $i = j = 0$).



As an example, the saturation of $\mathrm{g};\iota_0 = (X, Y, \mathcal{R})$ from $\sigma\mathrm{g}$ can be described as follows. Firstly, any rooted initial link $\langle v, \varepsilon \rangle$ in $\sigma\mathrm{g}$ forms an initial subnet between v and the root of Y in $(\sigma\mathrm{g});\iota_0$, which will be filled. Secondly, since copointed nets consist of initial links, if $\sigma\mathrm{g}$ contains a copointed subnet $\mathrm{q} \subseteq (\sigma\mathrm{g})_{v,\varepsilon}$, the duplication of initial links will produce a copointed subnet, in the saturation of $\mathrm{g};\iota_0$, between v and any w in Y. Then if w is pointed, such a copointed subnet is bipointed. Lemma 18, below, states that filling these bipointed subnets completes the saturation of $\mathrm{g};\iota_0$.

In a pre-net $\mathrm{f} = (X, Y, \mathcal{R})$, say that a vertex v in X has a *rooted copointed subnet* if there is a copointed net $\mathrm{q} \subseteq \mathrm{f}_{v,\varepsilon}$. If v is minimal among the vertices in X that have rooted copointed subnets in f, then v is said to have a *maximal* copointed subnet; let $\textsc{maxcp}(\mathrm{f})$ denote the set of such variables in f. Dually, let $\textsc{maxp}(\mathrm{f})$ be the set of variables in Y that have *maximal pointed subnets*, i.e. are minimal among the vertices that have *rooted* pointed subnets.

Lemma 18. *For a net* $\mathrm{g};\iota_j$ *let* $\sigma\mathrm{g} = (X, Y_j, \mathcal{R})$ *and let* $\sigma(\mathrm{g};\iota_j) = (X, Y, \mathcal{S})$*. Then* $\mathcal{S} = (\mathcal{R} \cdot j) \cup \Gamma \cup \Delta$*, where*

$$\begin{aligned}
\Gamma \ &= \ \{\langle v, *, w \rangle \mid X_v = \mathbf{0},\ \langle v, *, \varepsilon \rangle \in \mathcal{R}\} \\
\Delta \ &= \ \{\langle v, *, w \rangle \mid X_v = \mathbf{0} \text{ or } Y_w = \mathbf{1}, \\
&\qquad\qquad \exists v' \leq v.\ v' \in \textsc{maxcp}(\sigma\mathrm{g}), \\
&\qquad\qquad \exists w' \leq w.\ Y_{w'} \text{ is pointed } \}
\end{aligned}$$

The case $\pi_i;\mathrm{f}$ is dual. The lemma is proved by showing that its description of a saturated net, as $(X,\, Y,\, (\mathcal{R} \cdot j) \cup \Gamma \cup \Delta)$, is closed under $\rightsquigarrow$.

D. Deconstruction of saturated nets

The previous lemma (Lemma 18) illustrates that retrieving the saturation of f from that of $\mathrm{f};\iota_0$ is easy in some cases, but not in others. A simple case that follows immediately, for example, is that $\sigma(\mathrm{f};\iota_0) = (\sigma\mathrm{f});\iota_0$ if and only if $\sigma\mathrm{f}$ contains no rooted initial links. For the remaining part of the soundness proof, on nets from products into coproducts, this solves the case for saturated nets that are constructible. However, this need not be the case; and moreover, parallel nets constructed over different projections and injections, as illustrated in Figure 11, may have the same saturation (lower right). For an inductive proof this is clearly problematic: for nets $\mathrm{f};\iota_j$ and $\pi_i;\mathrm{g}$, with the same saturation, equivalent nets must be found that are constructed over the same projection or injection. In other words, equivalent nets must be found that

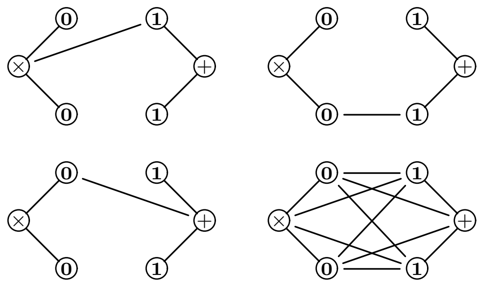


Fig. 11. Differently constructed nets with the same saturation

allow the deconstruction of a saturated net along a certain projection or injection.

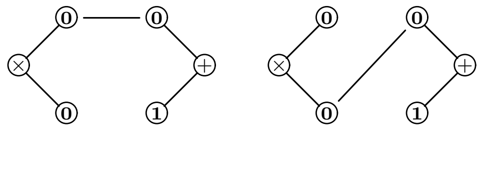
Fortunately the examples in Figure 11 also suggest a solution. The two nets on the left are both bipointed, and thus equivalent (by Lemma 16). Then for the one top right, it needs to be shown that from the fact that its saturation is full, it follows that it is equivalent to a bipointed net. Since not all saturated nets are between pointed and copointed objects, a generalisation is needed. Recall that a partial net is a pre-net that has at most one link for each switching. Call a partial net [*co*]*pointed* if it consists entirely of rooted terminal [initial] links—note that in this definition the target of a partial pointed net need not be pointed.

Lemma 19. *If* f *is a net and* $\mathrm{q} \subseteq \sigma\mathrm{f}$ *is a partial pointed or copointed net, then there is a net* g *s.t.* $\mathrm{q} \subseteq \mathrm{g}$ *and* $\mathrm{f} \Leftrightarrow \mathrm{g}$.

This solves the deconstruction problem, for nets $\mathrm{f};\iota_j$ and $\pi_i;\mathrm{g}$, as follows. Suppose the saturation of $\mathrm{f};\iota_j$ is non-constructible (i.e. not of the form $\pi_i;\mathrm{h}$ or $\mathrm{h};\iota_j$, nor $\langle \mathrm{h},\mathrm{k}\rangle$ or $[\mathrm{h},\mathrm{k}]$, for any pre-nets h and k). This can only be the case if the saturation of f contains rooted initial links, which rewrite from $\langle v,j\rangle$ in $\sigma(\mathrm{f};\iota_j)$ to $\langle v,\varepsilon\rangle$. Then the above lemma provides a net h equivalent to $\mathrm{f};\iota_j$ containing $\langle v,\varepsilon\rangle$, since this link constitutes, on its own, a partial copointed subnet of $\sigma(\mathrm{f};\iota_j)$. Now h, containing $\langle v,\varepsilon\rangle$, is not right-constructible, and so must be left-constructible; moreover, if $0 \leq v$ then h is constructed over π_0, and if $1 \leq v$ then over π_1. Since the saturation of $\pi_i;\mathrm{g}$ contains the same link $\langle v,\varepsilon\rangle$, it has an equivalent k constructed over the same projection as h.

E. Completing the proof

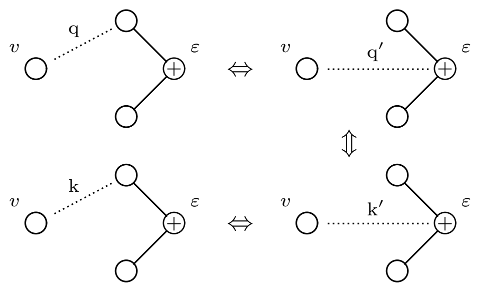
The final case in the soundness proof, for nets f and g between a product and a coproduct, is nearly complete. It was shown that if their (common) saturation is constructible, the induction hypothesis can be applied immediately, and that if it is not, there are equivalent nets $\mathrm{f}';\iota_j$ and $\mathrm{g}';\iota_j$ constructed over the same injection (or projection). A final obstacle is the fact that their components f' and g' need not have the same saturation, and indeed need not be equivalent. The simple example below illustrates the idea.



These two nets, $\pi_0;?;\iota_0$ and $\pi_1;?;\iota_0$, have the same saturation, which is full. However, their components $\pi_0;?$ and $\pi_1;?$ do not: they are already saturated. That this is a general problem can be observed from Lemma 18. Consider the saturation of a net $(\mathrm{f};\iota_0)$, with a pointed target Y, described by

$$(\sigma\mathrm{f});\iota_0 \;\cup\; \Gamma \;\cup\; \Delta$$

(abusing notation). With Y pointed, if a vertex v has a maximal copointed subnet q in $\sigma\mathrm{f}$, the subnet between v and ε is filled, by Δ. Now suppose g is identical to f, except that v has a different maximal copointed subnet k in the saturation $\sigma\mathrm{g}$. Then $\mathrm{f};\iota_0$ and $\mathrm{g};\iota_0$ have the same saturation, but f and g might not. The solution is illustrated below.



The subnets q and k need not be equivalent, but $\mathrm{q};\iota_0$ and $\mathrm{k};\iota_0$ are. They are equivalent to q' and k' by moving their links up to the root (each $\langle u,0\rangle$ becomes $\langle u,\varepsilon\rangle$). Because Y is pointed, q' and k' are bipointed, and hence equivalent by Lemma 16.

The generalised application of this idea is as follows. If nets $\mathrm{f};\iota_0$ and $\mathrm{g};\iota_0$ have the same saturation, it can be shown that the same vertices v have maximal copointed subnets in $\sigma\mathrm{f}$ as in $\sigma\mathrm{g}$. It may be assumed, by Lemma 19, that a maximal copointed subnet of $\sigma\mathrm{f}$ is also a subnet of f. Then a net h is formed from f as follows: for every vertex v that has a maximal copointed subnet q in f and one k in g, replace q in f with k. Then $\mathrm{h};\iota_0$ is equivalent to $\mathrm{f};\iota_0$ by the above reasoning (the equivalence of $\mathrm{q};\iota_0$ and $\mathrm{k};\iota_0$), while h and g have the same saturation, allowing the induction hypothesis to be applied.

V. The Category of Saturated Nets

The soundness result, together with completeness, means that saturated nets are in one-to-one correspondence with morphisms in $\Sigma\Pi(\mathcal{C})$. A complete description of this category requires also composition and identities to be defined. A useful result in this respect is the characterisation of saturated nets as unions over equivalence classes of nets.

Proposition 20. *The saturation of a net* f *is* $\bigcup\{\mathrm{g} \mid \mathrm{f} \Leftrightarrow \mathrm{g}\}$.

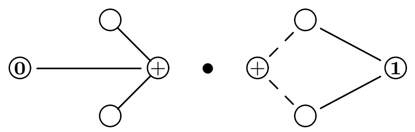
Identity nets are the translation and saturation of identity proofs in sum–product logic: nets $\sigma(\mathrm{id}_X)$ where

$$\begin{array}{lcllcl} \mathrm{id}_{X+Y} & \triangleq & [(\mathrm{id}_X;\iota_0),(\mathrm{id}_Y;\iota_1)] & \mathrm{id}_{\mathbf{0}} & \triangleq & ?_{\mathbf{0}} \\ \mathrm{id}_{X\times Y} & \triangleq & \langle(\pi_0;\mathrm{id}_X),(\pi_1;\mathrm{id}_Y)\rangle & \mathrm{id}_{\mathbf{1}} & \triangleq & !_{\mathbf{1}} \;. \end{array}$$

Hughes and Van Glabbeek established composition for unit-free nets as relational composition [9]. Define, for pre-nets,

$$(X,Y,\mathcal{R}) \bullet (Y,Z,\mathcal{S}) \quad\triangleq\quad (X,Z,\mathcal{S}\circ\mathcal{R})$$

(denoting relational composition by $(\circ)$; labels l and k may be composed as $k \circ l$ if both are morphisms in $\mathcal{C}$, and $*$ otherwise). In the presence of the units, this does not work immediately: the following composition would be empty.



As is clear from this simple example, relational composition does work for nets whose links only connect to leaves—call these *composable*—and, by moving unit links up towards the leaves, any net is equivalent to a composable one. Furthermore, that composition preserves equivalence follows by a comparison with the cut-elimination procedure for sum–product logic in [4], which it closely resembles.

Lemma 21. *For composable nets, if* $\mathrm{f} \Leftrightarrow \mathrm{f}'$ *and* $\mathrm{g} \Leftrightarrow \mathrm{g}'$ *then* $(\mathrm{f} \bullet \mathrm{g}) \Leftrightarrow (\mathrm{f}' \bullet \mathrm{g}')$.

For non-composable nets f and g the composition $\mathrm{f}' \bullet \mathrm{g}'$ of equivalent, composable nets $\mathrm{f}' \Leftrightarrow \mathrm{f}$ and $\mathrm{g}' \Leftrightarrow \mathrm{g}$ may be used; this does not define a unique result, but by the above lemma the possible outcomes are equivalent. Consequently, in the category of saturated nets, the composition of $\sigma\mathrm{f}$ and $\sigma\mathrm{g}$ must be the saturation $\sigma(\mathrm{f}' \bullet \mathrm{g}')$. It is obtained from $\sigma\mathrm{f}$ and $\sigma\mathrm{g}$ as follows.

Proposition 22. *Composition of saturated nets is relational composition followed by saturation.*

Since saturated nets are unions over equivalence classes, by Proposition 20, their relational composition is given by

$$\sigma\mathrm{f} \bullet \sigma\mathrm{g} \quad = \quad \bigcup \{\mathrm{h} \bullet \mathrm{k} \mid \mathrm{h} \Leftrightarrow \mathrm{f},\ \mathrm{k} \Leftrightarrow \mathrm{g}\} \ .$$

It is easily shown that this is a sub-pre-net of the desired solution $\sigma(\mathrm{f}' \bullet \mathrm{g}')$ mentioned above—in particular, the inclusion of pre-nets $\mathrm{h} \bullet \mathrm{k}$ for non-composable h and k is harmless. In many cases, it will be strictly smaller: $\sigma(\mathrm{f}' \bullet \mathrm{g}')$ is the union over all nets equivalent to $\mathrm{f}' \bullet \mathrm{g}'$, which are not necessarily of the form $\mathrm{h} \bullet \mathrm{k}$ with $\mathrm{h} \Leftrightarrow \mathrm{f}$ and $\mathrm{k} \Leftrightarrow \mathrm{g}$. Fortunately, the remaining nets can be captured by saturating $\sigma\mathrm{f} \bullet \sigma\mathrm{g}$.

VI. Notes, conclusions and further work

The saturated nets presented in this paper are canonical representations of proofs in additive linear logic with units, or equivalently of morphisms in free sum–product categories. The saturation algorithm, by which they are obtained from sum–product nets, is simple and tractable, which makes comparing saturated nets an effective decision procedure. Finally, some smaller issues will be picked up here.

Time complexity: The time complexity of saturation is as follows. Using an appropriate representation of nets, a saturation step may be performed in constant time. Bounded by the maximum number of unit links in a net $(X, Y, \mathcal{R})$, saturation has a time complexity bound of $\mathcal{O}(|X| \times |Y|)$ (where $|X|$ denotes the number of vertices in X). By comparison, the algorithm by Cockett and Santocanale [3] has a time complexity bound of

$$\mathcal{O}((hgt(X) + hgt(Y)) \times |X| \times |Y|)$$

(where $hgt(X)$ is the height of the syntax tree of X).

Correctness and sequentialisation: A tractable algorithm to find representatives of saturated nets, or *sequentialisation*, can be obtained from the soundness proof, using, in particular, the inductive characterisation of saturated nets. Such an algorithm also constitutes a correctness criterion, separating saturated nets from arbitrary pre-nets. A useful addition would be an elegant combinatorial correctness criterion, such as, possibly, a modification of the switching condition.

Games semantics: A fruitful branch of research into logic, linear or otherwise, is that of game-theoretic semantics, which interprets formulae as two-player games and proofs as (winning) strategies. Sum–product nets admit a simple game-interpretation, where two games are played in parallel, one on the source object and one on the target object. It appears that saturated nets, viewed as strategies, exhibit interesting game-theoretic properties concerning this parallelism, opening an intriguing angle for future work.

Acknowledgements

Many thanks to Alex Simpson for guidance and support. Thanks also to the anonymous referees for helpful, constructive comments. The author was supported by a Ph.D. studentship on EPSRC research grant EP/F042043/1.

References

[1] M. Barr: $*$-Autonomous categories and linear logic. Mathematical Structures in Computer Science 1:159–178 (1991)
[2] R.F. Blute, J.R.B. Cockett, R.A.G. Seely, T.H. Trimble: Natural deduction and coherence for weakly distributive categories. Journal of Pure and Applied Algebra 113:229-296 (1996)
[3] J.R.B. Cockett, L. Santocanale: On the word problem for $\Sigma\Pi$-categories, and the properties of two-way communication. LNCS 5771:194–208 (2009)
[4] J.R.B. Cockett, R.A.G. Seely: Finite sum–product logic. Theory and Applications of Categories 8(5):63–99 (2001)
[5] K. Došen, Z. Petrić: Bicartesian Coherence. Studia Logica 71(3):331–353 (2002)
[6] J.-Y. Girard: Linear Logic. Theor. Comput. Sci. 50 (1987)
[7] J.-Y. Girard: Proof-nets: The parallel syntax for proof-theory. Logic and Algebra, 97–124 (1996)
[8] D.J.D. Hughes: Simple free star-autonomous categories and full coherence. Preprint, available from the author's website (2005)
[9] D.J.D. Hughes, R.J. van Glabbeek: Proof nets for unit-free multiplicative-additive linear logic. ACM Trans. Comput. Log. 6(4):784–842 (2005)
[10] H. Hu, A. Joyal: Coherence completions of categories. Theor. Comput. Sci. 227(1-2):153–184 (1999)
[11] A. Joyal: Free bicomplete categories. C.R. Math. Rep. Acad. Sci. Canada 17(5):219–224 (1995)
[12] J. Lambek, P. Scott: Introduction to Higher-Order Categorical Logic. Cambridge University Press (1988)
[13] T.-W. Koh, C.-H.L. Ong: Explicit Substitution Internal Languages for Autonomous and $*$-Autonomous Categories. Electronic Notes in Theoretical Computer Science 29 (1999)
[14] S. Mac Lane: Categories for the working mathematician. Second edition. Graduate Texts in Mathematics, Vol. 5, Springer-Verlag, New York (1998)
[15] R.A.G. Seely: Linear logic, $*$-autonomous categories and cofree coalgebras. Contemporary Mathematics 92 (1989)
[16] L. Straßburger, F. Lamarche: On Proof Nets for Multiplicative Linear Logic with Units. CSL 2004:145-159