An approach to innocent strategies as graphs

Pierre-Louis Curien, Claudia Faggian

An approach to innocent strategies as graphs

Pierre-Louis Curien Claudia Faggian

Abstract

This paper proposes an approach for extending to graphs the close relation between proofs and innocent strategies. We work in the setting of L-nets, introduced by Faggian and Maurel as a game model of concurrent interaction. We show how L-nets satisfying an additional condition, which we call $\mathrm{L_S}$-nets, can be sequentialized into traditional tree-like strategies. Conversely, sequential strategies can be relaxed into more asynchronous ones.

We develop an algebra of constructors and destructors that serve to build and decompose graph strategies, and to describe a class of minimally sequential graph strategies, which can be seen as an abstract kind of multiplicative-additive proof-nets.

1 Introduction

In the context of game semantics several proposals have emerged - with different motivations - towards strategies where sequentiality is relaxed to capture a more asynchronous form of interaction [AM99, Hyl01, Mel04, HS02, MW05, SPP05]. Such strategies often appear as graphs, contrarily to more traditional (sequential) strategies, as in particular Hyland-Ong innocent strategies [HO00], which appear as trees. Here we will consider the setting of L-nets, recently introduced by Faggian and Maurel [FM05] as a game model of concurrent interaction, based on Girard's ludics.

A strategy describes in an abstract way the operational behaviour of a proof (or program). An interaction between tree strategies produces a sequence of actions, which describes the trace of the computation. The idea underlying L-nets (as well as other closely related approaches) is to not completely specify the order in which the actions should be performed, while still being able to express constraints. Certain tasks may have to be performed before other tasks; other actions can be performed in parallel, or scheduled in any order. A strategy is now a *directed acyclic graph*. The interaction results into a partial order, allowing for parallelism.

In this paper we are interested in relating parallel strategies and sequential strategies. Working in the setting of L-nets, we show how strategies represented by graphs, with partial ordering information, can be sequentialized into tree-like strategies; conversely, sequential strategies can be relaxed into more asynchronous ones.

The tree strategies and the graph strategies that we will consider are Girard's designs [Gir01] and (a subset of) Faggian and Maurel's L-nets, respectively. Syntactically, designs are particular sorts of Curien's abstract Böhm trees [Cur98, Cur06]).

As a computational object, a design is a Hyland-Ong innocent strategy on a universal arena, as discussed in [FH02]. L-nets are (potentially infinite) graph strategies on this arena. They subsume sequential strategies as a special case: a tree is, in particular, a graph. On the other hand, it is possible to define a class of L-nets of minimal sequentiality, which we call *parallel* L-nets.

Hence we have a homogeneous space inside which we can move, adding or relaxing sequentiality (i.e., dependency between the actions). Between completely sequential and completely parallel strategies, we get a full range of intermediate strategies with decreasing sequentiality level.

The present paper builds on a preliminary extended abstract [CF05], and on a subsequent analysis of the basic constructors and destructors on (graph or tree) strategies that are at work in our sequentialization an desequentialization procedures [Fag].

Two flavours of views. It is known that (innocent) tree strategies can be presented as sets of views with certain properties. A view is a totally ordered sequence of moves (again with certain properties), and the set of views forms a tree. Any interaction results into a totally ordered set of moves.

An L-net is a set of partially ordered views, each of which is a partially ordered set of moves, where the partial order expresses an enabling relation, or a scheduling among moves. The set of such partially ordered views forms a directed acyclic graph. Any interaction results into a partially ordered set of moves.

The proof-net experience. Tree strategies can be seen as abstract sequent calculus derivations. Specifically, designs arise as abstract (untyped) versions of (focalized) sequent calculus proofs of multiplicative-additive linear logic. By contrast, the class of graph strategies of minimal sequentiality (the parallel L-nets) can be seen as abstract multiplicative-additive proof-nets. Indeed, there are two standard ways to handle proofs in linear logic: either as sequent calculus derivations, or as proof-nets, which are graph-like structures satisfying a so-called correctness criterion. Sequent calculus derivations can be mapped onto proof-nets, by forgetting some of the order between the rules, and conversely proof-nets can be sequentialized into proofs.

While the origins of game semantics are closely connected to the analysis of correct proof structures [AJ92], this paper is, to the best of our knowledge, the first attempt to transfer – so to say in the other direction – the use of proof-net techniques to the semantic setting of (innocent) games. In this respect, our contribution fits into a general research direction aiming at bringing closer together syntax and semantics.

Relating sequential and parallel strategies. As we have anticipated, L-nets are a conservative extension of innocent strategies (in the form used by ludics). This makes it possible to relate the two approaches. We are able to associate *a set of tree-strategies* to a parallel L-net (in fact, any "correct" L-net, see below) $\mathfrak{D}$, by saturating the order, i.e., we add sequentiality to the point that all choices on scheduling of the moves during an execution are determined. Conversely, given a tree strategy Π, we have a desequen-

tialization procedure, which returns a parallel strategy, forgetting some "inessential" information on the scheduling.

Sequentialization is not possible for an arbitrary L-net, as L-nets can be intrinsically parallel, in the sense that actions depend on each other in an essential way. It is easy to build a counter-example to sequentialization by taking inspiration from Gustave function (a well-known example of a non-sequential function, see e.g. [AC98]). For this reason, we introduce $\mathrm{L_S}$-nets, which are L-nets satisfying a condition called Cycles. This condition upgrades the Acyclicity condition of [FM05], which is sufficient for computation purposes (i.e., to guarantee that strategies compose), but not for our goals here. Condition Cycles can be considered as an abstract correctness criterion, and as a matter of fact, it is the adaptation to our setting of Hughes and Van Glabbeek's toggling condition [HvG05].

A correctness criterion for proof-nets has two roles: it guarantees that (i) normalization is possible (we are not stuck with cycles during normalization), and (ii) it is possible to associate a sequent calculus derivation to the graph. The Acyclicity condition is a minimal criterion which takes care of (i); the new Cycles conditions guarantees also (ii).

We present an algebra of constructors and destructors allowing us to build and decompose graph strategies. This in particular allows us to (co)inductively define the classes of strategies of maximal and minimal sequentiality (i.e. of sequential and parallel strategies, respectively).

The destructors and constructors are also used to define the sequentialization procedure. This procedure works as a stream-like, bottom-up process (coinductively) acting on potentially infinite $\mathrm{L_S}$-nets. Dually, the same operations serve us to define a desequentialization procedure that transforms L-forests into parallel L-nets. We shall show that sequentialization and desequentialization can be performed so as to be inverse to each other.

Plan of the paper. Section 2 and Appendix B provide some background on (focalized) linear logic and designs, i.e., we introduce the ingredients of our universal arena and the tree strategies on which we work. Section 3 introduces our graph strategies. We recall the definition of L-net [FM05], and we define our subclass of sequentializable L-nets, the $\mathrm{L_S}$-nets[1].

The key technical result about $\mathrm{L_S}$-nets is the Splitting Lemma, which we prove in Appendix C. The core of the paper lies in Sections 4 through 8: we present our basic (co)algebra of elementary operations on L-nets (Sections 4 and 7), we describe our sequentialization and desequentialization procedures (Sections 5 and 6, respectively), and we relate them (Section 8). In Section 9, we impose connectedness restrictions on both the source and target of the procedures, so as to adjust the picture to Girard's original designs. Section 10 is a concluding section.

[1] $\mathrm{L_S}$-nets were called logical L-nets in [CF05].

2 Tree strategies and sequent calculus derivations

Designs, introduced in [Gir01], have a twofold nature: they are at the same time semantic structures (an innocent strategy, presented as a set of views) and syntactic structures, which can be understood as abstract sequent calculus derivations (in a focusing calculus, which we will introduce next).

In the following, we review in which (intuitive) sense a tree strategy can be associated with a sequent calculus derivation, and vice versa. In Appendix B, we provide formal procedures.

2.1 Focalization and synthetic connectives

Multiplicative and additive connectives of linear logic separate into two families: synchronous (also called positive) connectives: $\otimes, \oplus, 1, 0$, and asynchronous (or negative) ones: $\parr, \&, \bot, \top$. A formula is positive (negative) if its outermost connective is positive (negative).

A cluster of connectives with the same polarity can be seen as a single connective (called a *synthetic* connective), and a "cascade" of decompositions with the same polarity as a single rule. This corresponds to a property known as focalization, discovered by Andreoli (see [And01]), and which provides a (complete) so-called *focusing* strategy in proof-search: (i) negative connectives, if any, are given priority for persistent decomposition, (ii) when a subgoal containing only positive formulas is reached, choose a positive focus, and persistently decompose it up to its negative sub-formulas.

The division of connectives into positive and negative ones is not only fundamental to proof-search in linear logic, but also corresponds to the Opponent/Player alternation in a strategy.

Shift. To these standard connectives, it is natural to add two new (dual) connectives, called *shift*[2]: $\downarrow$ (positive) and $\uparrow$ (negative). The role of the shift operators is to change the polarity of a formula: if N is negative, $\downarrow N$ is positive, and if P is positive, $\uparrow P$ is negative. When decomposing a positive connective into its negative subformulas (or viceversa), the shift marks the polarity change. As an example, the formula $(A \& B) \oplus (C \otimes D)$ should now be written $(\downarrow (A' \& B')) \oplus (C' \otimes D')$, where, say, A' is the result of recursively decorating A with shift operators. The shift is the connective which *captures "time"* (or sequentiality): it marks a step in computation.

Focusing calculus. Focalization is captured by the following sequent calculus, originally introduced by Girard in [Gir99], and closely related to Andreoli's focusing calculus (see [And01]). We refer to those papers for more details.

Axioms: $\vdash x^\bot, x$
We assume by convention that all atoms x are positive (hence $x^\bot$ is negative).

Any positive (resp. negative) cluster of connectives can be written as a $\oplus$ of $\otimes$ (resp. a $\&$ of $\parr$), modulo distributivity and associativity. The rules for synthetic connectives

[2]The shift operators have been introduced by Girard as part of the decomposition of the exponentials.

are as follows. Notice that each rule has labels; rather than more usual labels such as $\otimes L$, $\otimes R$, etc., we use formulas in the labels, as described below.

Positive connectives: Let $P(N_1, \ldots, N_n) = \oplus_{I \in \mathcal{N}}(\otimes_{i \in I}(\downarrow N_i))$, where $\mathcal{N}$ is a set of sets I of indices. Each $\otimes_{i \in I}(\downarrow N_i)$ is called an additive component. In the calculus, there is an introduction rule for each additive component:

$$\frac{\ldots \quad \vdash N_i, \Delta_i \quad \ldots \quad \vdash N_j, \Delta_j \quad \ldots}{\vdash P, \ldots, \Delta_i, \ldots, \Delta_j, \ldots} \ (P, N_I)$$

A positive rule is labelled with a pair of (i) the active formula (or focus) P of the conclusion, and (ii) the set $N_I = \{N_i : \ i \in I\}$ of the subformulas of the additive component to which the rule corresponds.

Note that we should rather speak of a rule scheme, because even when P and N_I have been fixed, there remains freedom in the way of splitting the rest of the sequent between the premises.

Negative connectives: Let $N(P_1, \ldots, P_n) = \&_{I \in \mathcal{N}}(\parr_{i \in I}(\uparrow P_i))$. It admits only one introduction rule, which has a premise for each additive component $\parr_{i \in I}(\uparrow P_i)$:

$$\frac{\ldots \quad \vdash P_I, \Delta \quad \vdash P_J, \Delta \ldots}{\vdash N, \Delta} \ \{\ldots, (N, P_I), (N, P_J), \ldots\}$$

where $P_I = \{P_i : \ i \in I\}$. A negative rule is labelled by a set of pairs, each of the form (focus, set of subformulas), for each premise.

We call each of the pairs we used in the labels an *action.* We call an action positive (resp. negative) if it appears in the label of a positive (resp. negative) rule. In a negative rule, there is an action for each additive component.

In the purely multiplicative case (no connectives $\oplus, \&$), all negative rules have a single premise, and hence are labelled by a single action, while only one rule can be applied to each positive connective.

It is important to notice the *duality* between positive and negative rules: each premise (encoded by the action) (N, P_I) of a negative rule corresponds to one positive rule $(N^\perp, P_I^\perp)$ (where $P_I^\perp = \{N_i^\perp : \ i \in I\}$).

Another observation is that, starting with a proof of a sequent $\vdash P$ or $\vdash N$ consisting of one formula only, the rules maintain the invariant that all sequents contain at most one negative formula, a fact that can be stressed by writing $N^\perp \vdash \Delta$ (resp. $N_i^\perp \vdash \Delta_i$, $N_j^\perp \vdash \Delta_j$,...) instead of $\vdash N, \Delta$ (resp. $\vdash N_i, \Delta_i, \vdash N_j, \Delta_j, \ldots$).

Finally we note the following two special cases of the positive and negative rules (when $\mathcal{N} = \{I\}$ is a singleton and I is a singleton):

$$\frac{N^\perp \vdash \Delta}{\vdash \downarrow N, \Delta} \ (\downarrow N, N) \qquad \frac{\vdash P, \Delta}{(\uparrow P)^\perp \vdash \Delta} \ (\uparrow P, P)$$

In the sequel, we shall keep the shift operators implicit, except in those special cases. As a matter of fact, all the other shift operators can be (uniquely) reconstructed.

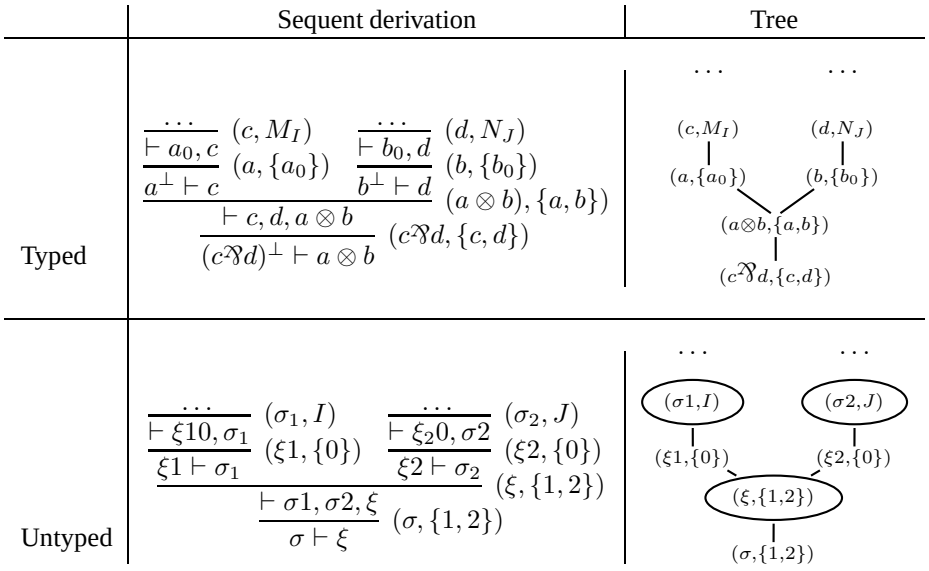


Figure 1: From focalized proofs to designs

2.2 Designs as (untyped) focusing proofs

Designs are an abstract version of focusing proofs. They are obtained in two steps. One transforms a sequent calculus proof into a tree whose nodes are labelled by actions, and one replaces all the formula occurrences by addresses. Conversely, given a design, we can build the "skeleton" of a sequent calculus derivation. Such a skeleton becomes a concrete (typed) derivation as soon as we are able to decorate it with types. Let us sketch this using an example.

First example. Consider the (purely multiplicative) derivation on the l.h.s. of Figure 1, where $a =\uparrow a_0$ and $b =\uparrow b_0$ are negative formulas and where c, d are positive formulas.

By forgetting everything in the sequent derivation but the labels of the rules, we obtain the tree depicted in the top right corner of Figure 1. This representation is more concise than the original sequent proof, but it still carries all relevant information, i.e., the sequents can be reconstructed. For example, when we apply the $\otimes$ rule, we know that the context of $a \otimes b$ is c, d, because they are used afterwards (above). After the decomposition of $a \otimes b$, we know that c (resp. d) is in the context of a (resp. b) because it is used after a (resp. b).

Addresses (loci). One of the essential features of ludics is that proofs do not manipulate formulas, but *addresses*. An address is a sequence of natural numbers, which could be thought of as a name, a channel, or as the address of a memory cell where an *occurrence of a formula* is stored. If we give address ξ to an occurrence of a formula, its (immediate) subformulas will receive addresses $\xi i, \xi j$, etc. Let $a = ((p_1 \parr p_2) \otimes q^\perp) \oplus r^\perp$.

If we locate a at the address ξ, we can locate $p_1 \parr p_2, q, r$ respectively at $\xi 1, \xi 2, \xi 3$ (the choice of addresses is arbitrary, as long as each occurrence receives a distinct immediate extension of ξ).

Let us consider an *action*, say, (P, N_I), where N_I corresponds to $\otimes_{i \in I}(\downarrow N_i)$. Its translation is (ξ, K), where ξ is the address of P, and K is the (finite) set of natural numbers corresponding to the relative addresses i of the subformulas N_i.

First example, continued. Coming back to our example (Figure 1), let us abstract from the type annotation (the formulas), and work with addresses. We locate $a \otimes b$ at the address ξ; for its subformulas a and b we choose the subaddresses $\xi 1$ and $\xi 2$. In the same way, we locate $c \parr d, c, d, a_0, b_0$ at the addresses $\sigma, \sigma 1, \sigma 2, \xi 10, \xi 20$, respectively. The result is depicted in the bottom right corner of Figure 1.

The two successive transformations are in fact independent. One can first transform formulas into addresses in the sequent calculus proof, yielding the bottom left abstract sequent derivation, and then keep only the tree of abstract labels. In Girard's terminology, the bottom left derivation and the bottom right tree are called *dessin* and *dessein*, respectively: they are the syntactic face and the semantic face of the same objects, which are called designs.

To indicate the polarity, in our pictures of designs and L-nets, we circle positive actions (to remind that they are clusters of $\otimes$ and $\oplus$).

Understanding the additives. A $\&$-rule must be thought of as the superposition of two unary rules on the same formula, corresponding to the two actions $(a \& b, a)$ and $(a \& b, b)$. Given a sequent calculus derivation in Multiplicative Additive Linear Logic (MALL), if for each $\&$-rule we select one of the premises, we obtain a derivation where all $\&$-rules are unary. This is called a *slice* [Gir87]. For example, the derivation on the l.h.s. below can be decomposed into the slices on the r.h.s.:

$$\dfrac{\dfrac{\overline{\vdash a, c} \quad \overline{\vdash b, c}}{\vdash a \& b, c}}{\vdash (a \& b) \oplus d, c} \quad \rightsquigarrow \quad \dfrac{\dfrac{\overline{\vdash a, c}}{\vdash a \& b, c}\,(a \& b, a)}{\vdash (a \& b) \oplus d, c} \quad \text{and} \quad \dfrac{\dfrac{\overline{\vdash b, c}}{\vdash a \& b, c}\,(a \& b, b)}{\vdash (a \& b) \oplus d, c}$$

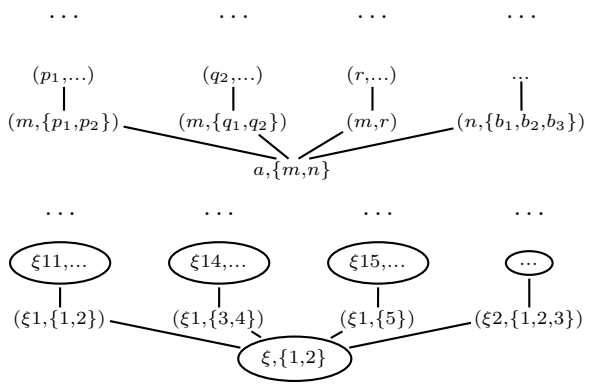
A more structured example. Let

$$a = (m \otimes n) \oplus c \ , \ m = (p_1 \parr p_2) \& (q_1 \parr q_2) \& r \ , \ n = b_1 \parr b_2 \parr b_3 \ ,$$

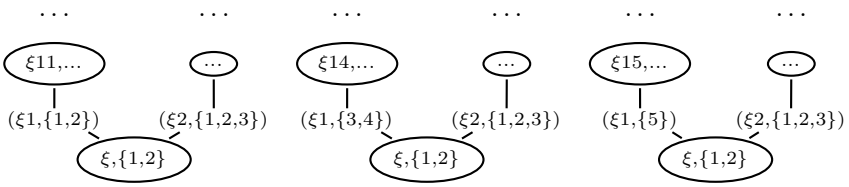
with r, p_i, q_i, b_i ($i = 1, 2$) positive formulas. Consider the following derivation:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dots}{\vdash p_1, p_2}\,(p_1, \dots) \quad \dfrac{\dots}{\vdash q_1, q_2}\,(q_2, \dots) \quad \dfrac{\dots}{\vdash r}\,(r, \dots)}{\vdash m}\,R_1 \quad \dfrac{\dfrac{\dots}{\vdash b_1, b_2, b_3}\,\dots}{\vdash n}\,R_2}{\vdash (m \otimes n) \oplus c}\,a, \{m, n\}$$

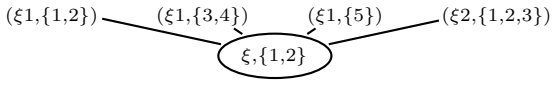
where $R_1 = \{(m, \{p_1, p_2\}), (m, \{q_1, q_2\}), (m, r)\}$ and $R_2 = \{(n, \{b_1, b_2, b_3\})\}$. The associated design is obtained as above in two steps:



It has three slices:



Bipoles. It is very natural to read a design (or an L-net) as built out of *bipoles*, which are the groups formed by a positive action (say, on address ξ) – the *root* of the bipole –, and all the negative actions which follow it (all being at immediate subaddresses ξi of ξ). The positive action corresponds to a positive connective. The negative actions are partitioned according to the addresses: each address corresponds to a formula occurrence, and each action on that address corresponds to an additive component. For example,

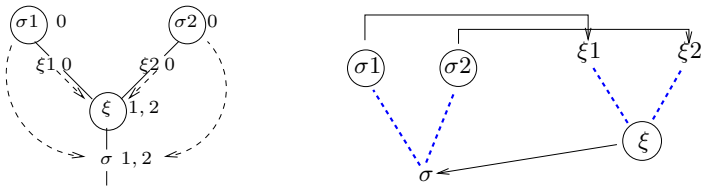


is a bipole, and the partition of the negative actions consists of the two sets

$$\{(\xi 2,\{1,2,3\})\} \quad \text{and} \quad \{(\xi 1,\{1,2\}),(\xi 1,\{3,4\}),(\xi 1,\{5\})\}\,.$$

Relating two orders: towards proof-nets. Let us consider a multiplicative design (or a slice). We are given *two partial orders*, which correspond to two kinds of information on each action $k = (\sigma, I)$: (i) a time relation (*sequential order*), specified by the tree structure of the design; (ii) a space relation (*prefix order*), corresponding to the relation of being subaddress (the arena dependency in game semantics).

Let us look again at our first example of design. We make explicit the relation of being a subaddress with a dashed arrow, as follows:



If we emphasize the prefix order rather than the sequential order, we recognize something similar to a proof-net (see [Fag02]), with some additional information on sequentialization. Taking forward this idea of proof-nets leads us to L-nets.

3 L-nets and $\mathrm{L_S}$-nets

In this section, we recall the notion of L-net of Faggian and Maurel [FM05]. We drop the Acyclicity condition[3], which we replace in Section 3.7 by a stronger condition. We will call $\mathrm{L_S}$-nets the class of L-nets which satisfy the stronger condition.

L-nets have an internal structure, described by a directed acyclic graph (d.a.g.) on polarized actions, and an interface, providing the names on which the L-net can communicate with the rest of the world.

3.1 Actions (moves).

An *action* is either the special symbol † (called daimon) or (cf. above) a pair $k = (\xi, I)$ given by an address ξ and a *finite* set I of indices (which are natural numbers). We say that k *uses* the address ξ.

The prefix relation induces a relation between the actions. We say that an action (ξ, I) *generates* the addresses ξi, for all $i \in I$, and that a is *parent* of b, if the action a generates the address of the action b (i.e., $a = (\xi, I)$ is parent of $b = (\xi i, K)$).

A *polarized action* is given by an action k together with a *polarity*, positive (k^+) or negative (k^-). The parent relation extends to polarized actions, with the condition that if a is parent of b, their polarity must be opposite. (Or, equivalently, all addresses of the same length have the same polarity.)

The action † is defined to be positive. When clear from the context, or not relevant, we omit the explicit indication of the polarity.

Arena. Actions together with the parent relation define what could be called a *universal arena*.

3.2 Interface

An *interface* (called base in [Gir01]) is a pair of finite sets Ξ, Λ of addresses, which we write as a sequent $\Xi \vdash \Lambda$, and which we call the negative and the positive addresses (or the inner and the outer names) of the interface, respectively. The addresses must be

[3]Thus L-nets stand here for the L-nets of [FM05] minus the Acyclicity condition.

pairwise disjoint, i.e., incomparable with respect to the prefix relation. We think of the inner names as passive, or receiving, and of the outer names as active or sending.

We impose that Ξ is either empty or a singleton (cf. Section 2.1).

3.3 Directed graphs (d.a.g.) and terminology

We recall that a directed acyclic graph (d.a.g.) G is an oriented graph without (oriented) cycles. In all our pictures, the edges are oriented downwards.

We consider any directed acyclic graphs G up to its transitive closure. In fact, we will only be interested in the properties of *non transitive edges*. An edge is called transitive if there exists another oriented path (of length > 1) from its source to its target.

We write $c \leftarrow b$ if there is a *non transitive* edge from b to c. We say that c is the *predecessor* of b We use $\overset{+}{\leftarrow}$ for the transitive closure of $\leftarrow$. Sometimes, when describing operations on graphs, it is convenient to uniformly add to the graph an edge with a certain property, without caring if this edge is transitive or not. In such a case, we write $c \longleftarrow b$.

A node n of G is called *minimal* (resp. *maximal*) if there is no node a such that $a \leftarrow n$ (resp. $n \leftarrow a$).

Downward closure. Given a node $n \in G$, we denote by $n^{\downarrow}$ (the downward closure of n) the sub-graph induced by restriction of G on $\{n\} \cup \{n' : \ n' \overset{+}{\leftarrow} n\}$.

D.a.g.'s and partial orders. It is standard to represent a strict partial order as a d.a.g., where we have $a \leftarrow b$ whenever $a <_1 b$ (i.e., there is no c such that $a < c$ and $c < b$.) Conversely, (the transitive closure of) a d.a.g. is a strict partial order on the nodes (or equivalently on the labels, if the nodes are labelled and all the labels are distinct).

3.4 L-nets

Nodes labelled by actions. We are going to work with nodes labelled by polarized actions. In the sequel, depending on the context, $k = (\xi, I)$ will read as either "k is a node labelled by (ξ, I)", or "k is an action equal to (ξ, I)". We shall let $k, a, b, c, \ldots$ range over nodes and actions. We extend to nodes the terminology we have introduced for the actions. We will say that a node is positive or negative, and that a node uses or generates an address, if it is the case for the labelling action.

Definition 3.1 (L-nets) *An* L-net $\mathfrak{D}$ *is given by:*

- *An interface* $\Xi \vdash \Lambda$. *If* Ξ *is empty (resp. non-empty),* $\mathfrak{D}$ *is called positive (resp. negative).*
- *A possibly infinite set* A *of nodes which are labelled by polarized actions*[4].

[4]Hence nodes are *occurrences* of actions.

- *A structure on A of* directed acyclic bipartite graph *(if $k \leftarrow k'$, the two nodes have opposite polarity) which satisfies conditions 1-4 and 5-6 below.*

  1. Views. *For each node k, all the addresses used in $k^{\downarrow}$ are distinct .*
  2. Parents.
     - *For each node a, using address σ, either σ belongs to the interface (and then the polarity of a is as indicated by the interface), or σ has been generated by (the action of) a preceding node $c \overset{+}{\leftarrow} a$ of opposite polarity, called the parent of a (this node c is uniquely determined by condtion Views).*
     - *If $a \leftarrow b$ and if a is positive, then b must use an address generated by a.*
     - *If a is negative, it has at most one predecessor.*

     (It follows that if a negative node is not minimal then its parent is its unique predecessor.)
  3. Negativity. *If $\Xi = \{\xi\}$ is not empty and $\mathfrak{D}$ is not empty, then at least one node uses ξ.*
  4. Positivity. *If a is maximal w.r.t. $\leftarrow$, then it is positive.*

  The conditions so far are enough if all addresses are distinct (i.e., if the structure is purely multiplicative). Conditions 5-6 below allow us to deal with the multiple use of addresses induced by the additive structure.

  Two distinct negative nodes are called sibling *– and we say that they form an* additive pair *– if they use the same address and if either they have the same predecessor, or they are both minimal.*

  5. Sibling. *Any two sibling nodes have distinct labels, i.e., of the form (σ, I_1), (σ, I_2), with $I_1 \neq I_2$.*
  6. Additives. *Given two distinct positive nodes k_1, k_2 which use the same address ξ (i.e., $k_1 = (\xi, K_1), k_2 = (\xi, K_2)$), there exists an additive pair w_1, w_2 such that $w_1 \overset{+}{\leftarrow} k_1$, and $w_2 \overset{+}{\leftarrow} k_2$.*

Remark 3.2 *Note that by condition Parents every minimal node uses an address in the basis. Conversely, the same condition also imposes an action using the negative address of the base (if any) to be minimal, but* actions using a positive address of the base need not be minimal.

In order to obtain a good computational behaviour of L-nets as strategies, and to be able to relate them to sequential innocent strategies, we still need a correctness criterion on graphs, which we give in Section 3.7. If we have in mind the theory of proof-nets, L-nets can be seen as "proof-structures."

The key role of condition Additives is to ensure a one-to-one correspondence between the nodes of an L-net and the sets of actions in their downward closure, that represent their history, or their preconditions.

Lemma 3.3 *For each pair of distinct nodes k, k' of an L-net $\mathfrak{D}$, the sets of actions of $k^{\downarrow}$ and $k'^{\downarrow}$ are different.*

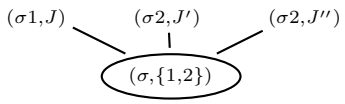
Proof. Suppose that the sets of actions of $k^{\downarrow}$ and $k'^{\downarrow}$ are the same. Then in particular there exists some $k_1 \in k'^{\downarrow}$ such that k and k_1 have the same label. We distinguish two cases:

- If $k_1 \neq k'$, then $k_1^{\downarrow}$ is strictly contained in $k'^{\downarrow}$, and hence by our assumption k_1 and k must be distinct.
- If $k_1 = k'$, then $k_1 \neq k$ by assumption.

Hence in both cases we have proved that k_1 and k are distinct. By condition Additives, there exists an additive pair w_1, w_2 such that $w_1 \in k^{\downarrow}$ and $w_2 \in k_1^{\downarrow}$ (and hence $w_2 \in k'^{\downarrow}$). Then, by our assumption, there is a node $w \in k^{\downarrow}$ that has the same label as w_2: this is impossible, as it would violate condition Views applied to k. □

3.5 Rules and conclusions.

A *negative rule* is a maximal set of (negative) sibling nodes. A *positive rule* consists of a single positive node. The positive nodes induce a partition of the d.a.g. into bipoles (cf. Section 2.2). Each bipole itself is partitioned into a positive rule (its root) and a set of negative rules. For example, the following bipole has two negative rules ($R_1 = \{(\sigma 1, J)\}$ and $R_2 = \{(\sigma 2, J'), (\sigma 2, J'')\}$) and one positive rule ($R = \{(\sigma, \{1, 2\})\}$).



We say that a rule is *unary* if it is a singleton (a positive rule is always unary). When a rule is not unary, we call it an *additive rule* (think of each action as an additive component, cf. Section 2.2). Note that an additive rule is necessarily a negative rule, but negative rules can be unary (see Section 3.6). Note also that if w_1, w_2 form an additive pair, then w_1, w_2 belong to the same negative rule.

By analogy with proof-nets, we call *conclusion* a rule whose nodes are all minimal.

We also observe that by condition Parents and Negativity, an L-net is positive if and only it has only positive conclusions, and a (non-empty) L-net is negative if and only if it has a negative conclusion. Note that an L-net can have at most one negative conclusion, by definition of a rule, and by the assumption that a basis contains at most one negative address.

3.6 Slices

We call *purely multiplicative* an L-net in which all used addresses are distinct, or, equivalently by condition Additives, in which all negative rules are unary (there are no additive pairs).

A *slice* $\mathfrak{S}$ of an L-net $\mathfrak{D}$ is a (downward closed) subgraph of $\mathfrak{D}$ which is a purely multiplicative.

In this paper, by slice we always mean a *maximal* slice.

3.7 Correctness criterion: $\mathrm{L_S}$-nets

We recall that we are only interested in the properties of *non transitive edges*. All conditions in this section are therefore on the skeleton of the graph.

Paths. The following notions are relative to some L-net $\mathfrak{D}$. An edge is an *entering edge* of the node a if it has a as target. If R is a negative rule and e an entering edge of an action $a \in R$, we call e a *switching edge* of R.

A *rule path* is a sequence of nodes $k_1, ...k_n$ belonging to distinct rules, and such that for each $i < n$ either $k_i \rightarrow k_{i+1}$ (the path is going down) or $k_i \leftarrow k_{i+1}$ (the path is going up). A rule cycle is defined similarly as a sequence of nodes $k_1, ...k_n, k_{n+1}$, where the k_i's ($i \leq n$) are distinct, where $k_1 = k_{n+1}$, and for each $i < n + 1$ either $k_i \rightarrow k_{i+1}$ or $k_i \leftarrow k_{i+1}$.

A *switching path* is a rule path which uses at most one switching edge for each negative rule, i.e., the path does not contain three successive nodes k_{i-1}, k_i, k_{i+1} such that k_i is negative, $k_i \leftarrow k_{i-1}$, and $k_i \leftarrow k_{i+1}$.

A *switching cycle* is a rule cycle which uses at most one switching edge for each negative rule.

Now we can complete the definition of $\mathrm{L_S}$-net. We want to be able to sequentialize our graphs. The following condition (which can be seen as a *correctness criterion*) guarantees that it is always possible to find a rule which does not depend on others.

Definition 3.4 ($\mathrm{L_S}$-nets) *An* $\mathrm{L_S}$-net *is an L-net such that the following condition holds:*

- Cycles. *Given a non-empty union C of switching cycles, there is an additive rule W not intersecting C, and a pair $w_1, w_2 \in W$ such that for some nodes $c_1, c_2 \in C$, $w_1 \overset{+}{\leftarrow} c_1$, and $w_2 \overset{+}{\leftarrow} c_2$.*

L-nets and $\mathrm{L_S}$-nets. The condition Cycles strengthens the Acyclicity condition of [FM05]. Acyclicity asserts that there are no switching cycles in a slice. It is immediate that the condition Cycles implies the acyclicity condition, and reduces to it in a purely mutliplicative framework (i.e., in the absence of any additive rule). Notice that while acyclicity is a property of a slice, the new condition speaks of cycles which traverse slices.

3.8 L-nets as sets of views / chronicles

Just as innocent strategies (and designs), an L-net can be presented as a set of views, with some properties. In this setting, a view is not a sequence of moves, but a partial order (with a top element).

Views/chronicles.

Definition 3.5 (Chronicles) *We call* chronicle *(or view) on $\Xi \vdash \Lambda$ a set $\mathfrak{c}$ of polarized actions equipped with a partial order, such that: $\mathfrak{c}$ has a top element, if $a <_1 b$ (cf. Section 3.3), they have opposite polarity, and $\mathfrak{c}$ satisfies (the analog of) condition Parents.*

A chronicle is positive or negative according to the polarity of its top element.

We define a partial order on chronicles as follows: $\mathfrak{c} \sqsubseteq \mathfrak{c}'$ if $\mathfrak{c}$ is the restriction of $\mathfrak{c}'$ to $\{x : \ x \leq a\}$, for a certain $a \in \mathfrak{c}'$. A set S of chronicles is *closed under restriction* if $\mathfrak{c}' \in S$ and $\mathfrak{c} \sqsubseteq \mathfrak{c}'$ implies $\mathfrak{c} \in S$.

From L-nets to sets of chronicles. Any node k of an L-net $\mathfrak{D}$ defines a chronicle: indeed, $k^{\downarrow}$ induces a partial order on its nodes (cf. Section 3.3) and by the condition Views, there is a one-to-one correspondence between the nodes and the actions in $k^{\downarrow}$. Let $\overline{n}$ be the action labelling the node n. We set:

$$\ulcorner k \urcorner = \{\overline{n} : \ n \in k^{\downarrow}\}, \text{ with the order induced by } \leftarrow .$$

Hence we can associate to each L-net $\mathfrak{D}$ a set *Views*$(\mathfrak{D})$ of chronicles, as follows:

$$\textit{Views}(\mathfrak{D}) = \{\ulcorner n \urcorner : \ n \text{ is a node of } \mathfrak{D}\} \ .$$

The set *Views*$(\mathfrak{D})$ is closed under restriction.

From sets of chronicles to L-nets. Conversely, given a set Δ of chronicles which is closed under restriction, we define a directed graph *Graph*(Δ) as follows: the nodes are the elements of Δ, and $\mathfrak{c} \leftarrow \mathfrak{c}'$ iff $\mathfrak{c} \sqsubset_1 \mathfrak{c}'$.

Lemma 3.6 *Let Δ be a (possibly infinite) set of chronicles closed under restriction. Graph(Δ) is an L-net iff it satisfies conditions Positivity and Additives.*

Proof. The conditions Parents and Views hold obviously. Condition Sibling also holds: two negative chronicles with the same parent $\mathfrak{c}$ have the form $\mathfrak{c}_1 = \mathfrak{c} \cup a$ and $\mathfrak{c}_2 = \mathfrak{c} \cup b$. If $\mathfrak{c}_1 \neq \mathfrak{c}_2$, necessarily $a \neq b$. □

It is rather easy to express both Positivity and Additives in terms of chronicles. Hence we can also define an-L-net on a given interface as a set of chronicles closed under restriction, which satisfies (the analogue of) Positivity and Additives.

Relating the presentations. It is immediate that *Views*$(\textit{Graph}(\Delta)) = \Delta$, if Δ is a set of chronicles closed under restriction. Conversely, given (the skeleton of) an L-net $\mathfrak{D}$, we have that *Graph*$(\textit{Views}(\mathfrak{D}))$ is isomorphic to $\mathfrak{D}$ (easy consequence of Lemma 3.3).

Summarizing, we have shown that *Views* and *Graph* are inverse bijections.

We will use both presentations for L-nets. The presentation of L-nets as sets of chronicles, on which we will largely rely, allows us to compare nodes in different

graphs, by comparing the corresponding chronicles. (In particular, this makes it possible to treat easily the superposition of two L-nets, see Section 4.3 and Appendix A.2). Sometimes, the graph presentation is more intuitive. However, it is obvious that all notions and conditions can be expressed in either terms. Observe in particular that

$$k_1 \overset{+}{\leftarrow} k_2 \quad \text{iff} \quad \ulcorner k_1 \urcorner \sqsubset \ulcorner k_2 \urcorner \quad \text{iff} \quad \overline{k_1} < \overline{k_2} \text{ in } \ulcorner k_2 \urcorner .$$

Conventions. We will often not distinguish between isomorphic notions. Moreover, to keep notation simple, we will sometimes write $k \in \mathfrak{c}$ (for example, $k \in \ulcorner k \urcorner$) instead of $\overline{k} \in \mathfrak{c}$. We will also use the action for the node labelled by that action, when the node is uniquely determined.

When we want to make clear or stress the polarity, we explicitly decorate actions, nodes, chronicles and L-nets with their polarity (for example, $k^+, \mathfrak{c}^+, \mathfrak{D}^+$).

3.9 Designs and L-forests

If the chronicles are totally ordered (that is, if they are sequences of actions), the above definitions produce a forest, corresponding to a "standard" (sequential) innocent strategy. In particular, the designs of [Gir01] can be regarded as a special case of L-nets: they are those L-nets Π such that:

1. Π is a forest and branches only on positive nodes;
2. there is a unique conclusion.

(We speak here of L-nets rather than $\mathrm{L_S}$-nets, because condition Cycles is vacuous for forests.)

If we do not impose conditions on the branching into nodes, we obtain a more general notion.

Definition 3.7 *An L-forest is an L-net* Π *such that:*

1. Π *is a forest;*
2. *all conclusions have the same polarity (hence, if* Π *is negative, then it has only one conclusion, which is negative).*

In Appendix B, we will show that L-forests arise from adding a form of MIX rule to the sequent calculus underlying designs.

L-forests provide the natural target and source for our sequentialization and desequentialization procedures. In Section 9, we will restrict our setting to the L-nets which correspond to designs.

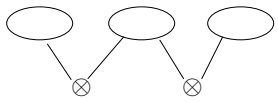
3.10 Sequential versus parallel strategies: an overview

Part of the process of abstraction leading from the syntax to strategies is that an action (a move) can be seen as a cluster of operations that can be performed together (thanks to focalization). However, in a tree strategy (L-forests, designs, innocent strategies...)

there remains a lot of artificial sequentiality, in the same way as in sequent calculus proofs for linear logic. In the case of proofs, the solution has been to develop proof-nets, a theory which has revealed itself extremely fruitful. The advantage of proof-nets is that information which is irrelevant to the "essence" of the proof is forgotten. More precisely, proof-nets allow us to identify sequent calculus derivations which only differ by some *permutations of rules*. Consider, for example, the two standard derivations

$$\dfrac{\dfrac{\vdash a, a^\perp \quad \vdash b, b^\perp \quad \vdash c, c^\perp}{\cdots}}{\vdash a^\perp \otimes b, b^\perp \otimes c, \ldots}\ \begin{matrix}(a^\perp \otimes b)\\ (b^\perp \otimes c)\end{matrix} \qquad \text{and} \qquad \dfrac{\dfrac{\vdash a, a^\perp \quad \vdash b, b^\perp \quad \vdash c, c^\perp}{\cdots}}{\vdash a^\perp \otimes b, b^\perp \otimes c, \ldots}\ \begin{matrix}(b^\perp \otimes c)\\ (a^\perp \otimes b)\end{matrix}$$

and compare them with the (unique) corresponding proof-net, which has the following shape:



The *different permutations* of the rules correspond to *different sequentializations* of the proof-net, that is, in our view, to *different schedulings* of the rules.

Similarly, sequential strategies (hence designs, in particular) distinguish proofs (or programs) which only differ by the order in which the operations are performed. The situation is in fact the same as for the sequent calculus, and we want to apply similar techniques.

We will define two procedures which we call *desequentialization* and *sequentialization*, that associate

- an $\mathrm{L_S}$-net *deseq* Π to an L-forest Π (Section 6), and
- a set of L-forests $\{\mathit{seq}\ \mathfrak{D}\}$ to an $\mathrm{L_S}$-net $\mathfrak{D}$ (Section 5), respectively.

All dependency which is taken away by desequentialization can be (non-deterministically) restored through sequentialization (Theorem 8.2). The non-determinism corresponds to the fact that several L-forests Π_i can be associated to the same $\mathrm{L_S}$-net $\mathfrak{D}$, each of which can be recovered by sequentialization of $\mathfrak{D}$. In Section 7, we make precise in which sense *deseq* Π has less sequentiality than Π, and give a description of the $\mathrm{L_S}$-nets of "minimal sequentiality" (which we call parallel L-nets), based on the operations presented in Section 4.

Why targetting L-forests. An L-net does not need to be connected (in the ordinary graph-theoretic sense). Non-connectedness is a natural and desirable feature if we want both parallelism and partial proofs, that is, proofs which can be completed into a proper proof. Actually, non-connectedness is an ingredient of Andreoli's concurrent proof construction [And02]. On the logical side, non-connectedness corresponds to the MIX rule (which is refused in [Gir01]).

We first prove a sequentialization (and desequentialization) result that holds for all $\mathrm{L_S}$-nets, and that has L-forests as target. In Section 9, we shall restrict this procedure so as to have designs as target.

4 Operations on $\mathrm{L_S}$-nets

In this section, we introduce operations that allow us to decompose and recompose L-nets and $\mathrm{L_S}$-nets. We shall make an intense use of these operations in Sections 4.2 through 4.6 where we shall describe sequentialization and desequentialization of $\mathrm{L_S}$-nets.

4.1 Preliminary properties

A convenient notion is that of partial strategy. We say that an L-net is *partial* if it does not satisfy condition Positivity.

We will make a repeated use of the following result.

Lemma 4.1 (Downward closure) *Let $\mathfrak{D}$ be an L-net. Assume $G \subseteq \mathfrak{D}$, and that G is downward closed (for any $k \in G$, if $a \in \mathfrak{D}$ and $a \overset{+}{\leftarrow} k$ in $\mathfrak{D}$, then $a \in G$). G is a (possibly partial) L-net. If $\mathfrak{D}$ satisfies condition Cycles, so does G.*

Proof. All properties are inherited from $\mathfrak{D}$. Any chronicle of G is also a chronicle of $\mathfrak{D}$. Let us check the preservation of conditions Additives and Cycles. If k_1, k_2 are two distinct nodes in G on the same address, then condition Additives for $\mathfrak{D}$ provides a pair of negative nodes w_1, w_2 such that $w_i \overset{+}{\leftarrow} k_i$, which (by downwards closure) belong to G.

Observe that any cycle in G is also a cycle in $\mathfrak{D}$. If we have a collection of switching cycles inside G, the condition Cycles for $\mathfrak{D}$ gives us an additive rule W that is not traversed by any of the cycles, and a pair $w_1, w_2 \in W \cap G$. Then, taking $W \cap G, w_1$, and w_2, the condition holds in G. □

Corollary 4.2 *If $\mathfrak{D}$ is an $\mathrm{L_S}$-net and K is a set of positive nodes of $\mathfrak{D}$, then the subgraph induced on $\bigcup\{k^{\downarrow} : k \in K\}$ is an $\mathrm{L_S}$-net.*

4.2 Rooting and boxing

The following constructions add a new *unary* conclusion.

Definition 4.3 (Rooting) *Let $\mathfrak{D}$ be a positive or negative L-net (of interface $\vdash \xi i, \xi j, \ldots, \Delta$ or $\xi i \vdash \Delta$). Let (ξ, I) be a negative or positive action, respectively. We indicate by $x \circ \mathfrak{D}$ the graph obtained as follows:*

1. *add a node $x = (\xi, I)$ to $\mathfrak{D}$;*
2. *add an edge $x \longleftarrow k$ for each node k which uses an address ξi (for some $i \in I$).*

If (ξ, I) is positive, the result is always an L-net (on the interface $\vdash \xi, \Delta$). If (ξ, I) is negative, the result is a possibly partial L-net (on the interface $\xi \vdash \Delta$). The condition Positivity is satisfied only if at least one of the addresses ξi is used in $\mathfrak{D}$.

Definition 4.4 (Boxing) *Let $\mathfrak{D}$ be a positive L-net (of interface $\vdash \xi i, \xi j, \ldots, \Delta$) and let (ξ, I) be a negative action, with $i, j, \ldots \in I$. We indicate by x . $\mathfrak{D}$ the graph obtained as follows:*

1. *add a node $x = (\xi, I)$ to $\mathfrak{D}$;*

2. *add an edge $x \leftarrow k$ for each node k which belongs to a conclusion of $\mathfrak{D}$.*

The result is clearly an L-net of interface $\xi \vdash \Delta$.

Remark 4.5 *Note that we have used $\longleftarrow$ and $\leftarrow$ in the definitions of rooting and boxing, respectively. This accounts for the fact that in the case of rooting an added edge $(\xi, I) \longleftarrow k$ might be non transitive if k uses an address ξi and is situated above a node using an address ξj.*

On positive L-nets, rooting or boxing give us two choices for adding a new negative node:

- *Rooting is a parallel operation,* in the sense that it only adds the minimum amount of sequentiality which is necessary for the satisfaction of condition Parents.

- *Boxing instead is a serial (sequential) operation*, which adds a maximal amount of sequentiality. If we think in terms of proof-nets, *boxing corresponds to enclosing $\mathfrak{D}$ in a box, which has x as principal port.*

As we shall see in Section 7, repetitive and consistent use of rooting and boxing will lead to (abstract versions of) proof-nets and sequent calculus derivations, respectively.

Remark 4.6 *Rooting and Boxing are two extremes. In between, we can define intermediate operators which add, on top of rooting, as much sequentiality as we wish: after rooting, we add any number of edges from positive nodes to x. Let us indicate this (generically) by $x \lhd \mathfrak{D}$. Hence, with respect to the order on the nodes, we have:*

$$Order(x \circ \mathfrak{D}) \subseteq Order(x \lhd \mathfrak{D}) \subseteq Order(x \centerdot \mathfrak{D}) \ .$$

The differences between the three L-nets is only the amount of order between x and the nodes of $\mathfrak{D}$.

4.3 Superposition of L-nets.

The union of a collection of L-nets is their union as sets of chronicles. The union is not disjoint in general; we also call this operation *superposition*. Under which conditions is a union of L-nets an L-net? By Lemma 3.6, if $\mathfrak{D}_1, \mathfrak{D}_2$ are L-nets (resp. $\mathrm{L_S}$-nets), $\mathfrak{D}_1 \cup \mathfrak{D}_2$ is an L-net (resp. an $\mathrm{L_S}$-net) iff it satisfies condition Additives (resp. conditions Additives and Cycles).

An especially relevant case is the following one, which we single out.

Proposition 4.7 *If two positive $\mathrm{L_S}$-nets are of the form $k^+ \circ \mathfrak{D}_1, k^+ \circ \mathfrak{D}_2$ and are such that the sets of addresses used by $\mathfrak{D}_1$ and $\mathfrak{D}_2$ are disjoint, then $(k^+ \circ \mathfrak{D}_1) \cup (k^+ \circ \mathfrak{D}_2)$ is an $\mathrm{L_S}$-net.*

Proof. Condition Additives is obviously inherited by the disjointness assumption. We show that this also true of condition Cycles. Suppose that there is a switching cycle traversing both $\mathfrak{D}_1$ and $\mathfrak{D}_2$, and consider two minimal portions of the cycle going from $\mathfrak{D}_1$ to $\mathfrak{D}_2$ and from $\mathfrak{D}_2$ to $\mathfrak{D}_1$, respectively. At most one of these portions can go through k. Thus, the other portion consists of two consecutive nodes $\mathfrak{c}_1$ and $\mathfrak{c}_2$, with, say, $\mathfrak{c}_1 \in \mathfrak{D}_1$, $\mathfrak{c}_2 \in \mathfrak{D}_2$, and $\mathfrak{c}_1 \sqsubset_1 \mathfrak{c}_2$, contradicting the disjointness assumption. □

Remark 4.8 *In defining the superposition of graphs it is crucial that we work not just with nodes, but with their view. This allows us to* compare nodes belonging to different graphs. *This fact plays a key role when defining additive union (below).*

4.4 Splitting

The following key lemma allows us to decompose a *positive* $\mathrm{L_S}$*-net* which satisfies the condition Cycles into disjoint components, where each component is itself an $\mathrm{L_S}$-net.

Definition 4.9 (Splitting rules) *1. A negative rule $W = \{\ldots, w_I, \ldots\}$ of an L-net $\mathfrak{D}$ is called splitting if either it is conclusion of $\mathfrak{D}$ (each w_I is a root), or if deleting the edges $w \leftarrow w_I$ to the root of W's bipole (for all $w_I \in W$), there is no more connection (i.e., no sequence of consecutive edges) between any of the w_I's and w.*

2. A positive rule of $\mathfrak{D}$ is called splitting if it is a conclusion and all negative rules just above it are splitting.

Lemma 4.10 (Splitting Lemma) *Every $\mathrm{L_S}$-net $\mathfrak{D}$ has a splitting conclusion. In particular, if all the conclusions are positive (i.e., if $\mathfrak{D}$ is positive), there is at least one positive splitting rule.*

The proof is given in Appendix C.

Splitting. As a consequence of the Splitting Lemma, we have the following property.

Proposition 4.11 (Splitting) *Let $\mathfrak{D}$ be an $\mathrm{L_S}$-net. If $\mathfrak{D}$ is positive, then there exists a positive conclusion $k = (\xi, I)$, which we call a splitting conclusion of $\mathfrak{D}$, such that*

$$\mathfrak{D} = (\bigcup_{i \in I} (k \circ \mathfrak{D}_{\xi i})) \biguplus \mathfrak{C} .$$

Moreover,

1. all $\mathfrak{D}_{\xi i}$'s and $\mathfrak{C}$ are $\mathrm{L_S}$-nets, and

2. they do not share addresses.

Proof. Let $k = (\xi, I)$ be a splitting positive conclusion. By deleting k, the graph splits into several connected components. Let us indicate by $\mathfrak{D}_{\xi i}$ the part of the graph which is connected to some nodes of address ξi, and let us indicate by $\mathfrak{C}$ the rest of the graph.

1. It is immediate that each $\mathfrak{C}$ and all G_i's are downward closed, and hence are $\mathrm{L_S}$-nets by Lemma 4.1. It follows readily that $\mathfrak{D}_{\xi i} = G_i \setminus k$ is an $\mathrm{L_S}$-net, for all i.

2. Suppose for a contradiction that there are two nodes k_1, k_2, say in $\mathfrak{D}_{\xi i}$ and in $\mathfrak{D}_{\xi j}$ using the same address. By condition Additives applied to $\mathfrak{D}$, there exists an additive pair w_1, w_2, with w_1, w_2 below k_1, k_2, respectively, which by downward closedness implies $w_1 \in \mathfrak{D}_{\xi i}$ and $w_2 \in \mathfrak{D}_{\xi j}$. This is impossible because all the nodes in any negative rule W of $\mathfrak{D}$ belong to the same connected component. □

4.5 Root removal

The following operation allows us to decompose a *negative L-net*, whose negative conclusion is *unary* (this is the only negative destructor we need in the case of a purely multiplicative L-net).

Definition 4.12 (Root removal) *Given an L-net $\mathfrak{D}$ (of interface $\xi \vdash \Delta$) with a negative unary conclusion $\{x\}$ with $x = (\xi, I)$, we indicate by $\mathfrak{D} \setminus x$ the graph obtained from $\mathfrak{D}$ by removing x.*

It is immediate that the result is an L-net on the interface $\vdash \ldots, \xi i, \ldots, \Delta$ ($i \in I$).

4.6 Additive structure

In a setting in which all negative rules are unary (as for slices or in multiplicative linear logic), root removal is all we need to decompose a negative L-net. In a general setting, the following constructions allow us to decompose, and to reconstruct, a *negative L-net*.

Definition 4.13 (Scoping) *Let $\mathfrak{D}$ be an L-net of negative conclusion $X = \{x_I : I \in \mathcal{N}\}$. For all $I \in \mathcal{N}$, we define the* scope of x_I in $\mathfrak{D}$ *as follows:*

$$\mathit{Scope}(x_I, \mathfrak{D}) = \{\mathfrak{c} : \mathfrak{c} \sqsubseteq \mathfrak{c}', \mathfrak{c}' \in \mathfrak{D}, \mathfrak{c}' \text{ positive and } x_J \notin \mathfrak{c}' \text{ for all } J \in \mathcal{N} \setminus \{I\}\} .$$

or equivalently:

$$\mathit{Scope}(x_I, \mathfrak{D}) = \bigcup \{k^{\downarrow} : k \text{ positive and } x_J \not\dashv k \text{ for all } J \in \mathcal{N} \setminus \{I\}\} .$$

By Lemma 4.1, if $\mathfrak{D}$ is an L-net (resp. an $\mathrm{L_S}$-net), $\mathit{Scope}(x_I, \mathfrak{D}) \subseteq \mathfrak{D}$ is an L-net (resp. an $\mathrm{L_S}$-net).

Definition 4.14 (Additive union) *Let $\mathfrak{D}_I, \mathfrak{D}_J, \ldots$ be a set of L-nets which all have negative unary conclusions $x_I = (\xi, I)$, $x_J = (\xi, J), \ldots$ on the same address ξ (with distinct $I, J, \ldots$). Their additive union $\Downarrow_I \mathfrak{D}_I$ is defined as*

$$\Downarrow_I \mathfrak{D}_I = \bigcup_I \Phi(\mathfrak{D}_I) ,$$

where each $\Phi(\mathfrak{D}_I)$ is obtained from $\mathfrak{D}_I$ by minimally adding edges in such a way that $x_I \stackrel{+}{\leftarrow} k$ (in $\Phi(\mathfrak{D}_I)$ for each positive node $k \in \mathfrak{D}_I$ such that $\ulcorner k \urcorner \notin \mathfrak{D}_J$ (for some $J \neq I$).

Intuitively, Φ is a function on chronicles which *marks* with an edge towards x_I the chronicles of $\mathfrak{D}_I$ which are specific to it, or more precisely those chronicles which are *not shared* by all $\mathfrak{D}_J$'s.

Remark 4.15 *Notice that if the set of $\mathfrak{D}_I$'s is a singleton, then $\Phi(\mathfrak{D}_I) = \mathfrak{D}_I$. This remark will allow us to treat both unary and non unary conclusions as a single case.*

The following two lemmas will play a crucial role in the decomposition of L-nets. Their statement refers to the notations in Definition 4.14.

Lemma 4.16 *1. The chronicles of $\Phi(\mathfrak{D}_I)$ can be partitioned into two disjoint sets:*

$$\Phi(\mathfrak{D}_I) = C \uplus B_I \ ,$$

where

$$C = \bigcap_J \mathfrak{D}_J = \{\mathfrak{c} : \ \mathfrak{c} \in \mathfrak{D}_J, \text{ for all } J\}$$
$$B_I = \{\ulcorner k \urcorner \in \Phi(\mathfrak{D}_I) : \ x_I \stackrel{+}{\leftarrow} k\} = \{\mathfrak{c} \in \Phi(\mathfrak{D}_I) : \ x_I \in \mathfrak{c}\}$$

2. If $\mathfrak{D} = \biguplus_I \mathfrak{D}_I$ then

$$\mathfrak{D} = C \uplus (\bigcup_I B_I) \ .$$

Proof.

1. If a chronicle $\mathfrak{c}$ of $\Phi(\mathfrak{D}_I)$ does not belong to B_I, then, by construction, no edge has been added, which implies both that $\mathfrak{c}$ is a chronicle of $\mathfrak{D}_I$ and that it belongs to all $\mathfrak{D}_J$'s ($J \neq I$). Thus $\Phi(\mathfrak{D}_I) = C \cup B_I$. Moreover, if $x_I \in \mathfrak{c}$, then $\mathfrak{c}$ cannot belong to any $\mathfrak{D}_J$ ($J \neq I$), as this would entail $x_I \in \mathfrak{D}_J$, contradicting the assumption that $\mathfrak{D}_J$ has a *unary* conclusion. Hence the union is disjoint.
2. If $\mathfrak{D} = \biguplus_I \mathfrak{D}_I = \bigcup_I \Phi(\mathfrak{D}_I)$, we can write $\mathfrak{D} = \bigcup_I (C \uplus B_I) = C \uplus (\bigcup_I B_I)$.

□

Proposition 4.17 *$\biguplus_I \mathfrak{D}_I$ is an L-net. Moreover the construction preserves condition Cycles.*

Proof. It is immediate that each $\Phi(\mathfrak{D}_I)$ is an L-net. All properties are inherited from $\mathfrak{D}$. As for condition Cycles, notice that all the newly added edges enter x_I, and no switching path which uses the new edges to x_I can continue to form a cycle. Hence all $\Phi(\mathfrak{D}_I)$'s are $\mathrm{L_S}$-nets.

We are left to show (cf. Section 4.3) that $\biguplus_I \mathfrak{D}_I = \bigcup_J \Phi(\mathfrak{D}_J)$ satisfies conditions Additives and Cycles. We just check condition Cycles. It is convenient to partition the nodes of $\biguplus_I \mathfrak{D}_I$ as in Lemma 4.16.

Given a collection of switching cycles, assume it is contained inside one of the $\Phi(\mathfrak{D}_J)$'s: in such a case the additive pair is given by the condition applied to $\Phi(\mathfrak{D}_J)$.

Otherwise, we have at least a node $k_1 \in B_I$ and a node $k_2 \in B_J$ (with $I \neq J$) traversed by the cycles. By construction, $x_I \overset{+}{\leftarrow} k_1$ and $x_J \overset{+}{\leftarrow} k_2$, and condition Cycles is satisfied. □

Remark 4.18 *Notice that $\bigcap \mathfrak{D}_J$ is not an L-net in general , because its maximal chronicles do not need to be positive.*

Lemma 4.19 *If $\mathfrak{D} = \biguplus_J \mathfrak{D}_J$, then:*

1. *Scope$(x_I, \mathfrak{D}) = \Phi(\mathfrak{D}_I)$.*
2. *Assume $\mathfrak{D}_I = x_I \circ \mathfrak{C}_I$, or $\mathfrak{D}_I = x_I \bullet \mathfrak{C}_I$. Then:*

$$\textit{Scope}(x_I, \mathfrak{D}) \setminus x_I \;=\; \mathfrak{C}_I \;.$$

Proof.

1. By Lemma 4.16, we can write $\mathfrak{D} = C \uplus (\bigcup_J B_J)$. We compute *Scope*$(x_I, \mathfrak{D})$:
   - by definition of scoping, the B_J's ($J \neq I$) are left out;
   - no chronicle of C contains an x_J, since C is disjoint from each B_I;
   - every chronicle of B_I contains x_I, and hence no chronicle of B_I contains any x_J ($J \neq I$), by condition Views.

   It follows that *Scope*$(x_I, \mathfrak{D}) = C \uplus B_I = \Phi(\mathfrak{D}_I)$.
2. This follows immediately from 1, since all what Φ does to $\mathfrak{D}_I$ is undone when x_I is removed.

□

4.7 Constructors and destructors

Summing up the content of this section, the operators we have presented can be grouped into two families:

- Rooting, boxing, union and additive union are *constructors*.
- Splitting, root removal, and scoping are *destructors*. The decomposition of an $\mathrm{L_S}$-net goes as follows:

$\mathfrak{D}$ *is positive.* All conclusions are positive. If there are no negative rules, we are finished: our L-net is reduced to its conclusions.
Otherwise, by splitting, $\mathfrak{D} = (\bigcup_i \ (x \circ \mathfrak{D}_i)) \uplus \mathfrak{C}$. Hence

$$\mathfrak{D} \rightsquigarrow \ldots \ , \ \mathfrak{D}_i \ , \ \ldots \ , \ \mathfrak{C} \ .$$

$\mathfrak{D}$ *is negative* (and non empty). Let X be the unique negative conclusion of $\mathfrak{D}$.

i. If X is unary, we decompose $\mathfrak{D}$ by root removal.

ii. Otherwise, we reduce $\mathfrak{D}$ to the previous case by scoping.

In general terms, assuming $X = \{\ldots, x_I, \ldots\}$ we have:

$$\mathfrak{D} \rightsquigarrow \ldots \ , \ \mathit{Scope}(x_I, \mathfrak{D}) \setminus x_I \ , \ \ldots \ .$$

These constructors and destructors are put to use in the following sections.

5 Sequentializing a graph strategy

An edge of an L-net states a dependency, an enabling relation, or a precedence among actions. The aim of this section is to provide a procedure, which takes an $\mathrm{L_S}$-net and adds sequentiality in such a way that the constraints specified by the L-net are respected.

Let us consider a very simple example: a chronicle $\mathfrak{c}$, i.e., a partially ordered view. A sequentialization of $\mathfrak{c}$ is a linear extension of the partial order. That is, we add sequentiality (edges) to obtain a total order. A total order which extends c will define a complete scheduling of the tasks, respecting the constraint that each action is performed only after all of its constraints are satisfied.

Dependency between the actions of a slice, and of sets of slices (L-nets) is more subtle, as there are also global constraints. The key point in the sequentialization is to select a rule which does not depend on others. This is exactly the role of the Splitting Lemma, and the reason for the condition Cycles.

The process of sequentialization is non-deterministic, as one can expect, i.e., there are several tree strategies which can be associated to the same $\mathrm{L_S}$-net.

As we have both multiplicative and additive structure, when sequentializing we will perform two tasks:

1. add sequentiality (sequential links) until the order in each chronicle is completely determined;

2. separate slices which are shared through additive superposition.

Sequentialization procedure. The following procedure *progressively* transforms an $\mathrm{L_S}$-net $\mathfrak{D}$ into an L-forest on the same interface as $\mathfrak{D}$. It works bottom-up and follows the paradigm of lazy, stream-like computation.

The procedure is non-deterministic. In what follows, $\mathfrak{D}' = \mathit{seq}\ \mathfrak{D}$ should be read as: "$\mathfrak{D}'$ is a possible sequentialization of $\mathfrak{D}$".

$\mathfrak{D}$ is negative. Let $X = \{\ldots, x_I, \ldots\}$ be the unique negative conclusion of $\mathfrak{D}$. Let $\mathfrak{D}_I = \mathit{Scope}(x_I, \mathfrak{D}) \setminus x_I$, for all I. Then:

- $\mathit{seq}\ \mathfrak{D} = \bigcup_I\ (x_I\ \centerdot\ \mathit{seq}\ \mathfrak{D}_I)$.

$\mathfrak{D}$ is positive.

1. Assume $\mathfrak{D}$ is connected (in the ordinary graph-theoretic sense).

If $\mathfrak{D}$ consists of a single positive node, we are finished.

Otherwise we select a positive splitting rule $x = (\xi, I)$ and proceed as follows. By Proposition 4.11, each of the components $\mathfrak{D}_i$ obtained by splitting is an $\mathrm{L_S}$-net whith a negative conclusion on an address ξi. Then:

- $\mathit{seq}\ \mathfrak{D} = \bigcup_i\ (x \circ \mathit{seq}\ \mathfrak{D}_i)$.

2. Assume $\mathfrak{D} = \biguplus_i\ \mathfrak{C}_i$, where the $\mathfrak{C}_i$'s are the connected components of $\mathfrak{D}$. Then:

- $\mathit{seq}\ \mathfrak{D} = \biguplus_i\ (\mathit{seq}\ \mathfrak{C}_i)$.

Proposition 5.1 *If $\mathfrak{D}$ is an $\mathrm{L_S}$-net on the interface $\Xi \vdash \Delta$, seq $\mathfrak{D}$ is an L-forest on the same interface.*

Proof. We have already established all partial results needed to prove this. □

This procedure applies to infinite L-nets, by coinduction. Indeed, one can formally show that L-forests form a final coalgebra and the $\mathrm{L_S}$-nets form a coalgebra for a functor F on sets, and that *seq* is the associated unique coalgebra morphism. We only sketch the construction below. The reader unfamiliar with final coalgebra semantics can get the necessary background from [JR97].

- One considers the functor F in the category of sets and functions that takes a set X to the disjoint union of the set of all finite sets whose elements are of the form $((\xi, I), \{\ldots, a_i, \ldots\})$, where the a_i's form a collection of elements of X indexed by a subset of I, and of the set of all $\{\ldots, ((\zeta, J), a_J), \ldots\}$, where the a_J's form a collection of elements of X indexed by some $\mathcal{N} \subseteq \mathcal{P}_f(\omega)$.

- One proves that the collection of all L-forests forms a final coalgebra for this functor. The situation is similar to that of, say, Böhm trees. The coalgebra structure takes a positive (respectively negative) L-forest and decomposes it into its root(s) and its immediate subforests.

- Thanks to the Splitting Lemma, one can choose a decomposition for each positive $\mathrm{L_S}$-net, and codify this "oracle" in the form of a coalgebra structure on the collection of all $\mathrm{L_S}$-nets.

- Then *seq* is the unique coalgebra morphism from this coalgebra to the final coalgebra. That it is a coalgebra morphism amounts to the equations given above to define *seq* $\mathfrak{D}$.

6 Desequentializing a tree strategy

We are now looking for a *desequentialization* transformation in the opposite direction, from L-forests to $\mathrm{L_S}$-nets. This transformation should take as input the output of the sequentialization procedure. We have already seen that the output of *seq* is only of a potential (or coinductive) nature. More precisely, its progressive construction yields at any step an actual finite part Π – the part of the forest that has been already recognized –, and a collection of $\mathrm{L_S}$-nets to sequentialize, each associated with a leaf of Π.

Dually, we shall desequentialize *truncations* of L-forests (up to an arbitrary finite level). The output of the procedure will be a finite $\mathrm{L_S}$-net $\mathfrak{D}$, on which one could graft appropriately the (possibly infinite) subforests that have been taken away by the truncation.

In order to define the desequentialization procedure, we need to introduce a new notion, that of decoration.

Making the axioms explicit: decorations. In Section 3.10, we have illustrated the purpose of desequentialization by taking as example the relation between proof-nets and sequent calculus derivations. Our aim is to remove some artificial sequentialization, while preserving essential information:

1. *axioms* (multiplicative proof-net = formula tree + axioms [Gir87]);
2. *dependency due to additive rules*: some nodes must not be shared.

The second issue is addressed by our definition of additive union (cf. Section 4.6). As for axioms, such information is present in the source L-forest (or design), but is *implicit* (and not univoque). To make the information on the axioms explicit (and univoque), we introduce an auxiliary notion, that of decorated node. Essentially, we decorate each leaf k with a set of addresses, which we denote by $\mathit{link}(k)$; this information codes the axioms. For example, a leaf $k = (\xi, \{0\})$ which is decorated with the address σ will correspond to the axiom $\vdash \xi 0, \sigma$. The decoration is closely related to the sequent calculus presentation of an L-forest, and we describe it in detail in Appendix B.

In Section 2.2, we already exemplified how to move from an L-forest to an explicit sequent calculus style presentation. Essentially, given an L-forest Π, we associate to each node k a sequent of addresses. Each leaf $k = (\xi, I)$ in the forest will correspond to a *generalized axiom* in the sequent calculus derivation, of either (see Section B) of the two forms

$$\frac{}{\vdash \xi, \Gamma}\ k = (\xi, I)^+ \qquad \frac{}{\vdash \Gamma}\ k = \dagger$$

The decoration we will use captures this information: we decorate each leaf k of the L-forest with the set Γ of addresses which appear in the sequent associated to k.

Decorated leaves as boxes. The two ideas of truncation of an L-forest and of decorating the leaves of an L-forest are related, and complement each other. Suppose that we truncate the tree Π after the node k, leaving out the subtrees Π_i, above k. The sequent associated to the node k, which is now a leaf, is the interface of the subtree

$\bigcup\ (k \circ \Pi_i)$. Hence, the addresses in *link*(k) are meant as the addresses which are used in the Π_i's. In this sense, a decorated leaf acts as a sort of (black) box: we hide the content of the box (i.e., $\bigcup\ (k \circ \Pi_i)$) and only keep memory of the interface (the conclusion of the box).

Link sets versus infinitary expansions. In ludics, identity axioms are interpreted by *infinitary* strategies, called faxes in [Gir01]. These strategies are an instance of the copy-cat strategies of game semantics. In such infinitary strategies, every generated action is eventually used. Faxes are the typical example of what we want to enclose in a box. Actually, even if we are establishing general results, the kind of strategies we are really interested in are those corresponding to proofs. Morally, the (real) use of link sets is to deal with finite truncations of these strategies. In other words, link sets are a way to express (in a finitary way) the axioms.

Definition 6.1 *A* decorated $\mathrm{L_S}$*-net is an* $\mathrm{L_S}$*-net* $\mathfrak{D}$ *in which all leaves* k *are equipped with a finite set* $link(k)$ *of addresses (called the link set of* k*), in such a way that the conditions on* $\mathrm{L_S}$*-nets hold with respect to all addresses (thus, including those in the link sets).*

We still use $\mathfrak{D}$ *to denote a decorated* $\mathrm{L_S}$*-net.*

Observe that if $\mathfrak{D}$ is an L-net, and given an assignment of link sets to the leaves of $\mathfrak{D}$, all what we have to check for it to yield a decorated $\mathrm{L_S}$-net are conditions Parents and Additives.

From now on, we upgrade the definition of "a node uses an address" as follows:

Definition 6.2 (used addresses) *Let* k *be a node labelled by an action and possibly a link set. We say that the node* k *uses an address* ξ *if either*

- ξ *is the address of the action, or*
- ξ *appears in the link set.*

The extension of the operators we have introduced for L-nets to decorated L-nets is immediate. The upgraded definition plays a role only when rooting an L-net on a negative action: now (with the notation of Definition 4.3), we add an edge $(\xi, I) \longleftarrow k$ for each node k such that k is generated by (ξ, I), or k is a leaf such that $\xi i \in link(k)$. We maintain the same notations as in Section 4.

Remark 6.3 *The new edges* $(\xi, I) \longleftarrow k$ *where* k *is a leaf such that* $\xi i \in link(k)$ *are close in spirit to the* μ*-pointers recently introduced by Laurent in his investigations on game semantics for first-order (classical) logic [Lau].*

Observe that a label of a node is now a a decorated action, i.e., an action possibly together with a link set . *Actions which are decorated in a different way are different, and we do not identify them.* This is natural if we consider that they correspond to different axioms.

In the following, we define a desequentialization procedure, which takes as input finite decorated L-forests. More precisely we choose a special decoration discipline,

which corresponds to the idea of making the axioms explicit. In Appendix B, we prove that it is always possible to choose a decoration which satisfies the following property.

Definition 6.4 (well-decorated) *A well decorated L-net is a decorated L-net $\mathfrak{D}$ such that all addresses of the interface, and all addresses generated by a negative action of $\mathfrak{D}$ are used in $\mathfrak{D}$ (in the sense of Definition 6.2).*

Lemma 6.5 *Every L-forest can be well decorated.*

Proof. See Corollary B.5. □

Desequentialization procedure.

Π **is negative.** Let $X = \{x_I, x_J, \ldots\}$ be the conclusion of Π. Let us call Π_I the subforest above x_I (i.e., $\Pi = \bigcup_I \ (x_I \centerdot \Pi_I)$). Then:

- *deseq* $\Pi = \bigcup\!\!\!\bigcup_I \ (x_I \circ$ *deseq* $\Pi_I)$.

Π **is positive.**

1. Assume Π is a tree of conclusion x, using address ξ. If the tree is reduced to a single node, then we are done (base case). Otherwise, it has the form $\Pi = \bigcup_i \ (x \circ \Pi_i)$, where each Π_i is the subforest of all the trees on the address ξi ($i \in I$). Then:

  - *deseq* $\Pi = \bigcup_i \ (x \circ$ *deseq* $\Pi_i)$.

2. Assume $\Pi = \biguplus_i \ \Pi_i$. Then:

  - *deseq* $\Pi = \biguplus_i \ ($*deseq* $\Pi_i)$.

Proposition 6.6 *If Π is a well decorated L-forest on the interface $\Xi \vdash \Delta$, then deseq Π is an $\mathrm{L_S}$-net on the same interface.*

Proof. All steps needed to show that *deseq* Π is a partial $\mathrm{L_S}$-net are immediate. Notice that in the positive case, $\bigcup_i \ (x \circ$ *deseq* $\Pi_i)$ is an $\mathrm{L_S}$-net by Proposition 4.7 (since all the addresses used in different Π_i's are disjoint, so are the addresses of the *deseq* Π_i's).

Condition Positivity follows from Lemma 6.7 below, and by (the upgraded) definition of rooting. □

Notice that decorations play a role only to prove that *deseq* $\mathfrak{D}$ satisfies condition Positivity.

Lemma 6.7 *Let Π be a well decorated L-forest on the interface $\Xi \vdash \Delta$. All the addresses of the interface are used in deseq Π.*

Proof. Similar to the proof of Lemma 9.6. □

Remark 6.8 *At any step of the desequentialization:*

- *the sets of labels of Π and deseq Π are the same (intuitively, no node is deleted);*
- *if l is the label of a leaf in Π, it also labels a leaf in deseq Π.*

7 An algebraic presentation

In this section, we focus on the $\mathrm{L_S}$-nets generated by the constructors (rooting, boxing, union and additive union), and we single out two important classes of $\mathrm{L_S}$-nets, obtained by consistently using rooting, or consistently using boxing, respectively (cf. Section 4.2). In the first case, we speak of *parallel* L-nets, which we regard as *abstract proof-nets*. In the second case, we get the *L-forests*, which correspond to *abstract sequent calculus derivations* (see Appendix B).

In the following, we denote with $\mathfrak{D}^+$ a positive L-net, and with $\mathfrak{D}^-_\sigma$ a negative L-net whose negative conclusion has address σ. We denote by k^+ a (possibly decorated) positive action.

Abstract proof-nets. A *parallel L-net* is an $\mathrm{L_S}$-net generated by the following grammar:

$$\begin{array}{lcl} \mathfrak{D} & := & \mathfrak{D}^+ \mid \mathfrak{D}^-_\sigma \\ \mathfrak{D}^+ & := & \mathfrak{E}^+ \uplus \ldots \uplus \mathfrak{E}^+ \\ \mathfrak{E}^+ & := & k^+ \mid \bigcup_{i\in I} ((\xi, I)^+ \circ \mathfrak{D}^-_{\xi_i}) \\ \mathfrak{D}^-_\sigma & := & \biguplus_J (\sigma, J)^- \circ \mathfrak{D}^+ \end{array}$$

Such an $\mathrm{L_S}$-net has *minimal sequentiality*, in the sense that the we use constructors of minimal sequentiality.

Abstract sequent calculus derivations. The *sequential* L-nets are the L-nets generated by the following grammar:

$$\begin{array}{lcl} \mathfrak{D} & := & \mathfrak{D}^+ \mid \mathfrak{D}^-_\sigma \\ \mathfrak{D}^+ & := & \mathfrak{E}^+ \uplus \ldots \uplus \mathfrak{E}^+ \\ \mathfrak{E}^+ & := & k^+ \mid \bigcup_{i\in I} ((\xi, I)^+ \circ \mathfrak{D}^-_{\xi_i}) \\ \mathfrak{D}^-_\sigma & := & \bigcup_J ((\sigma, J)^- \centerdot \mathfrak{D}^+) \end{array}$$

It is clear that sequential L-nets and L-forests are one and the same thing. Notice that, by construction, both classes of L-nets hereditarily admit splitting.

Remark 7.1 *1. We write $\bigcup_{i\in I} ((\xi, I) \circ \mathfrak{D}_{\xi i})$ instead of $(\xi, I) \circ (\bigcup_{i\in I} \mathfrak{D}_{\xi i})$, because $(\bigcup_{i\in I} \mathfrak{D}_{\xi i})$* is not *an L-net, according to our definition.*

2. The production $\mathfrak{D}^+ := \mathfrak{E}^+ \uplus \ldots \uplus \mathfrak{E}^+$ takes care of graphs which are not connected (i.e., that a re built using the MIX rule).

Remark 7.2 *If $\mathfrak{D}_I, \mathfrak{D}_J, \ldots$ are L-forests, then*

- $\biguplus_I \mathfrak{D}_I = \bigcup_I \mathfrak{D}_I$, *and*
- *positive rooting behaves in a "boxing-like" fashion in $x^+ \circ \mathfrak{D}^-_i$.*

8 Relating sequential and parallel strategies

In this section, we study the relation between L-forests (sequential strategies) and parallel L-nets (parallel strategies). We have already proved (Proposition 5.1) that for every $\mathrm{L_S}$-net, *seq* $\mathfrak{D}$ is an L-forest. Conversely, the following is an immediate consequence of the definition of parallel L-nets.

Proposition 8.1 *For every L-forest* Π*, deseq* Π *is a parallel L-net.*

Every time we desequentialize an L-forest Π, there is a sequentialization procedure *seq* such that *seq* $(\mathit{deseq}\ \Pi) = \Pi$.

Theorem 8.2 *Given an L-forest* Π*, there exists a strategy of sequentialization such that* $\Pi = \mathit{seq}\ (\mathit{deseq}\ \Pi)$.

Proof. We only consider the interesting cases.

$\mathfrak{D}$ **is negative.** Let us denote by $\{x_I, \dots\}$ the conclusion of the tree. Since $\Pi = \bigcup_I\ (x_I . \Pi_i)$, its desequentialization is $\mathit{deseq}\ \Pi = \biguplus_I\ (x_I \circ \mathit{deseq}\ \Pi_I)$. To sequentialize, we use scoping. By Lemma 4.19 (ii), $\mathit{Scope}(x_I, (\mathit{deseq}\ \Pi)) \setminus x_I = \mathit{deseq}\ \Pi_I$. Hence

$$\mathit{seq}\ (\mathit{deseq}\ \Pi) = \bigcup_I\ (x_I\ .\ \mathit{seq}\ (\mathit{deseq}\ \Pi_I))\ .$$

$\mathfrak{D}$ **is positive.** If the root is x, $\Pi = \bigcup_i\ (x.\Pi_i)$. Since $\mathit{deseq}\ \Pi = \bigcup_i\ (x \circ \mathit{deseq}\ \Pi_i)$, to sequentialize it we select x as splitting rule. Removing x gets us back to the set of all $\mathit{deseq}\ \Pi_i$'s. Hence:

$$\mathit{seq}\ (\mathit{deseq}\ \Pi) = \bigcup_i\ (x \circ \mathit{seq}\ (\mathit{deseq}\ \Pi_i))\ .$$

□

Theorem 8.2 says that in the desequentialization there is no essential loss of information. All dependency (sequentialization) which is taken away can be restored.

Establishing a result in the opposite direction (i.e., $\mathit{deseq}\ (\mathit{seq}\ \mathfrak{D}) = \mathfrak{D}$) only makes sense starting from a parallel L-net, because as *deseq* Π reduces sequentiality to a "minimal" amount, if $\mathfrak{D}$ is not parallel there is no hope that $\mathit{deseq}\ (\mathit{seq}\ \mathfrak{D}) = \mathfrak{D}$.

Theorem 8.3 *If* $\mathfrak{R}$ *is a parallel L-net, it admits a sequentialization procedure such that* $\mathit{deseq}\ (\mathit{seq}\ \mathfrak{R}) = \mathfrak{R}$.

Proof. Following the destructors, we are guaranteed (i) to have splitting, and (ii) that when we use scoping, we are in the situation described by Lemma 4.19. We just spell out the definitions.

- If $\mathfrak{R}$ *is negative*, we have $\mathfrak{R}^- = \bigdownarrow_I (x_I \circ \mathfrak{R}_I)$. By definition of sequentialization, we have

$$\mathit{seq}\ \mathfrak{R} = \bigcup_I (x_I \ \centerdot\ (\mathit{Scope}(x_I, \mathfrak{R}) \setminus x_I)) \ .$$

But by Lemma, 4.19, we have $\mathit{Scope}(x_I, \mathfrak{R}) \setminus x_I = \mathfrak{R}_I$. Hence we have in fact

$$\mathit{seq}\ \mathfrak{R} = \bigcup_I (x_I \ \centerdot\ (\mathit{seq}\ \mathfrak{R}_I) \ ,$$

from which

$$\mathit{deseq}\ (\mathit{seq}\ \mathfrak{R}^-) = \bigdownarrow_I (x_I \circ (\mathit{deseq}\ (\mathit{seq}\ \mathfrak{R}_I)))$$

follows

- If $\mathfrak{R}$ *is positive*, assume $\mathfrak{R}^+ = \bigcup_i (x \circ \mathfrak{R}_i)$ (all others cases are immediate). By construction, x is a splitting positive rule, and we select it. We have that $\mathit{seq}\ \mathfrak{R} = \bigcup_i (x \circ \mathit{seq}\ \mathfrak{R}_i)$. Hence we have:

$$\mathit{deseq}\ (\mathit{seq}\ \mathfrak{R}^+) = \bigcup_i (x \circ (\mathit{deseq}\ (\mathit{seq}\ \mathfrak{R}_i))) \ .$$

□

Corollary 8.4 (Completeness) *An* $\mathrm{L_S}$*-net* $\mathfrak{D}$ *is a parallel L-net if and only if there is an L-forest* Π *such that* $\mathfrak{D} = \mathit{deseq}\ \Pi$*. In particular, parallel L-nets are* $\mathrm{L_S}$*-nets.*

Remark 8.5 *The crucial point in the proof of Theorem 8.3 is that the following holds for a parallel L-net:*

$$\mathfrak{R}^- = \bigdownarrow_I (x_I \circ (\mathit{Scope}(x_I, \mathfrak{R}) \setminus x_I)) \ ,$$

i.e. we can decompose (or destruct) a negative parallel L-net (scoping) and then reconstruct it (rooting and additive union). This does not hold in general for an $\mathrm{L_S}$*-net* $\mathfrak{D}$*.*

Remark 8.6 *We have omited decorations in this section for simplicity. To be perfectly rigorous, one should maintain decorations through all the sequentialization and desequentialization process. For example, the sequentialization of a well decorated* $\mathrm{L_S}$*-net is defined just as the sequentialization of an* $\mathrm{L_S}$*-net (the only difference concerns the base case, where the decorations are kept).*

9 Restricting the picture to designs

As mentioned earlier (and as proved in Appendix B), L-forests correspond to designs with MIX. In this section, we show that we can get rid of this rule by (unsurprisingly) imposing an additional connectedness assumption on L_S-nets.

Given an L-net $\mathfrak{D}$ and a slice $\mathfrak{S} \subseteq \mathfrak{D}$, a *switching graph* of $\mathfrak{S}$ is a subgraph obtained from $\mathfrak{S}$ by choosing a single entering edge for each negative node, and deleting all the other ones. A slice is *S-connected* if all its switching graphs are connected. Finally, we call an L-net S-connected if all its maximal slices are.

An S-connected L-forest is obviously a tree, and in fact it is a design.

Lemma 9.1 *An L-forest* Π *is S-connected iff it is a design (in the sense of [Gir01]).*

Sequentialization and desequentialization preserve S-connectedness, and hence by restriction to S-connected $\mathrm{L_S}$-nets our results specialize to designs, rather than arbitrary L-forests. We shall give details only for the desequentialization. The proofs concerning the sequentialization are similar and simpler.

Lemma 9.2 (Slices) *Let* $\mathfrak{D}^+ = \bigcup_i (x^+ \circ \mathfrak{D}_i)$. *All slices of* $\mathfrak{D}$ *have the form* $\mathfrak{S} = \bigcup_i (x^+ \circ \mathfrak{S}_i)$, *where each* $\mathfrak{S}_i$ *is a slice of* $\mathfrak{D}_i$.

Let $\mathfrak{D}^- = \bigcup\!\!\!\bigcup_I (x_I \circ \mathfrak{D}_I) = \bigcup_I \Phi(x_I \circ \mathfrak{D}_I)$. *If* $\mathfrak{S}$ *is a slice of* $\mathfrak{D}$, *then* $\mathfrak{S}$ *is a slice of* $\Phi(x_I \circ \mathfrak{D}_I)$ *(for some* I, *and conversely. Moreover,* $\mathfrak{S}_I = \mathfrak{S} \setminus x_I$ *is a slice of* $\mathfrak{D}_I$. *On can recover* $\mathfrak{S}$ *from* $x_I \circ \mathfrak{S}_I$ *by adding appropriate edges from some nodes of* $\mathfrak{S}_I$ *to* x_I.

Proposition 9.3 *If the L-net* $\mathfrak{D}$ *is S-connected, seq* Π *is S-connected, and hence it is a design.*

In order to restrict the converse transformation, we need a strengthening of Lemma 6.7.

Definition 9.4 *A* uniformly decorated *L-forest is an L-forest that is well decorated slicewise, i.e., each slice* $\mathfrak{S}$ *uses all the addresses of the interface, and all the addresses generated by a negative action of* $\mathfrak{S}$.

Lemma 9.5 *Every L-forest can be uniformly decorated.*

Proof. See Corollary B.5. □

There is a bijective correspondence between uniformly decorated L-forests, and their sequent calculus representation (Proposition B.4).

Lemma 9.6 (Used addresses) *Let* Π *be a uniformly decorated L-forest on the interface* $\Xi \vdash \Delta$. *If* $\mathfrak{S}$ *is a maximal slice of the decorated L-net deseq* Π, *all the addresses of the interface are used in* $\mathfrak{S}$.

Proof. The claim is true if Π consists of a single decorated action on $\vdash \Gamma$ ($\underline{\Pi}$ is essentially reduced to an axiom).

Assume $\Pi = \bigcup_i ((\xi, I)^+ \circ \Pi_i)$ *is positive and has interface* $\vdash \xi, \Delta$. Each Π_i is an L-forest of interface $\xi i \vdash \Delta$. For each i, any slice $\mathfrak{S}_i \subseteq$ *deseq* Π_i uses all the addresses in $\xi i \vdash \Delta$. Hence $\bigcup_i ((\xi i) \circ$ *deseq* $\mathfrak{S}_i)$ is a slice which uses all the addresses in $\vdash \xi, \Delta$.

Assume $\Pi = \bigcup_I ((\xi, I)^- \centerdot \Pi_I)$ *is negative and has interface* $\xi \vdash \Delta$. By Lemma 9.2, $\mathfrak{T}$ is a slice in $\biguplus_I ((\xi, I)^- \circ$ *deseq* $\Pi_I)$ iff $\mathfrak{T}$ is a slice of $\Phi((\xi, I) \circ$ *deseq* $\Pi_I)$, for some I. Moreover, $\mathfrak{T}_I = \mathfrak{T} \setminus x_I$ is a slice of *deseq* Π_I.

Each Π_I is an L-forest of interface $\vdash \xi * I, \Delta$ (where $\xi * I = \{\xi i, i \in I\}$). Hence $\mathfrak{T}_I$ uses all the addresses in such a interface, and we can obtain the slice $\mathfrak{T}' = (\xi, I)^- \circ \mathfrak{S}_I$ which uses all the addresses in $\xi \vdash \Delta$. Finally, since $\mathfrak{T}$ is obtained from $\mathfrak{T}'$ by adding some edges, this operation does not change the nodes, and hence does not change the set of used addresses. □

Proposition 9.7 *If* Π *is a design, and if we choose a uniform decoration for* Π*, then* *deseq* Π *is S-connected.*

Proof. By assumption, Π is an S-connected L-forest.

Π *is negative.* By Lemma 9.2, $\mathfrak{S}$ is a slice of *deseq* $\Pi = \bigcup_I \Phi(x_I \circ$ *deseq* $\Pi_I)$ iff $\mathfrak{S}$ is a slice of $\Phi(x_I \circ \mathfrak{D}_I)$, for some I. We have that $\mathfrak{S}_I = \mathfrak{S} \setminus x_I$ is a slice of *deseq* Π_I. By hypothesis, $\mathfrak{S}_I$ is S-connected. Let $x_I = (\xi, I)$. By Lemma 9.6, in $x_I \circ \mathfrak{S}_I$ there are some edges connecting x_I to the nodes of $\mathfrak{S}_I$, those using some ξi. We obtain $\mathfrak{S}$ by adding some more edges.

We conclude by observing that (i) any choice of an edge entering x_I leaves x_I connected to a node of $\mathfrak{S}_I$, (ii) any switching S of $\mathfrak{S}$ restricted to $\mathfrak{S}_I$ is a switching of $\mathfrak{S}_I$, and (iii) by hypothesis, any two nodes of $\mathfrak{S}_I$ are connected in S.

Π *is positive.* By Lemma 9.2, $\mathfrak{S}$ is a slice of *deseq* $\Pi = \bigcup_i (x \circ$ *deseq* $\Pi_i)$ iff $\mathfrak{S} = \bigcup_i (x \circ \mathfrak{S}_i)$, and $\mathfrak{S}_i$ is a slice of *deseq* Π_i, for all i. By induction, all the $\mathfrak{S}_i$'s are S-connected, and hence $\mathfrak{S}$ is S-connected. □

10 Discussion and further work

Graduating sequentiality. In our setting, if we have minimal sequentiality, we have "parallel" strategies. At the other extreme, if we have maximal sequentiality, we have designs. The tools we have defined allow us to vary between these extremes, and hence provide us with a framework in which we can graduate sequentiality.

We are currently investigating this gradient of sequentiality, in particular along the following two directions. (i) In this paper we saturate L-nets to maximal sequentiality. We are studying how to perform sequentialization gradually, by adding sequential edges progressively. (ii) We would like to have a more precise understanding of what it means to have maximal or minimal sequentiality, and to investigate the extent of our desequentialization.

Actually, we already have answers to both questions in the multiplicative case, but the extension to the additive case needs further investigation.

Proof-nets. If tree strategies can be seen as *abstract sequent calculus derivations*, L-nets correspond to *abstract proof-nets.* We are currently investigating a typed setting, which also means to define a new syntax for proof-nets.

The typed counter-part of $\mathrm{L_S}$-nets should be an extension of focusing proof-nets [And02]. While previous work on focusing proof-nets was limited to multiplicative linear logic, our framework extends to additive connectives.

We expect also to study the gradient of sequentiality in a typed setting, and then, mirroring the treatment of strategies, we expect to be able to move from MALL proof-nets to sequent calculus derivations in a continuum. More precisely, using the semantical experience, we treat the graphs as orders. Then, we vary the amount of sequentiality (order) on the graphs (proof-nets) from most parallel to most sequential, where the most-sequential proof-nets can be seen as sequent calculus derivations.

This setting would realize a goal which was first proposed by Girard. Preliminary steps in this direction have already produced an extremely simple new proof of sequentialization for multiplicative proof-nets [DGF06].

Further questions. We have singled out two classes of L-nets, those of maximal sequentiality (which are idempotent with respect to *seq*) and those of minimal sequentiality. Notice that while *seq* applies to arbitrary L-nets, here we have defined *deseq* only on trees. We expect to be able to define the desequentialization of arbitrary L-nets, by using the Splitting Lemma.

Moreover, we are able to give a direct characterization of L-forests, while we only have an inductive definition of parallel L-nets. Beside the problem of expressing minimal sequentiality, we also need to capture the fact that an L-net admits decomposition. The condition Acyclicity captures the L-nets which hereditarily admit splitting (i.e., positive decomposition). To describe the L-nets which hereditarily admit negative decomposition (scoping with good properties), we need some weak form of typing, such as the typing provided by an arena. We postpone the definition of such a setting to future work.

Equational theory, canonicity. An underlying idea, which needs further study, is that a parallel strategy has not only an interest as an "asynchronous" model of computation, but also could play the same role that proof-nets have in providing an equational theory for proofs.

We expect to be able to use the parallel L-nets as an equivalence class on L-forests, i.e.,

$$\Pi_1 \cong \Pi_2 \iff \mathit{deseq}\,\Pi_1 = \mathit{deseq}\,\Pi_2 \ .$$

If we restrict to purely multiplicative L-nets, it is easy to show that given two parallel L-nets $\mathfrak{R}_1, \mathfrak{R}_2$ $(\mathit{seq}\,\mathfrak{R}_1 = \mathit{seq}\,\mathfrak{R}_2) \implies (\mathfrak{R}_1 = \mathfrak{R}_2)$ (*canonicity*). We do not know if this is true also in the additive case.

Acknowledgments. We would like to thank Olivier Laurent for numerous discussions on MALL proof-nets, and also Dominic Hughes and Rob van Glabbeek for fruitful exchanges on the technique of domination.

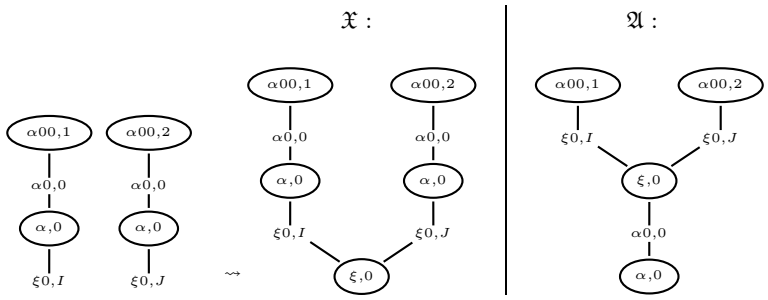


Figure 2:

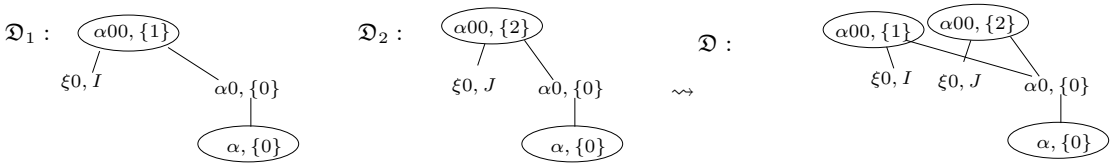
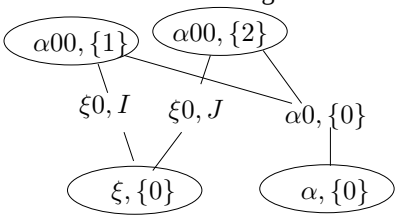


Figure 3:

A Examples

A.1 Sequentialization

Let us consider the following L-net $\mathfrak{R}$:



We have two negative rules ($\{(\xi 0, I), (\xi 0, J)\}$ and $\{(\alpha 0, \{0\})\}$), and two positive conclusions, both splitting. To sequentialize, we choose one of them. If we choose $(\xi, \{0\})$, we obtain the two trees on the left-hand side of Figure 2, and then the design $\mathfrak{X}$. Instead, by choosing $(\alpha, \{0\})$ we obtain the design $\mathfrak{A}$ (on the r.h.s.).

A.2 Superposition

The superposition of two L-nets is their union as sets of chronicles. Let us see an example. Consider the two L-nets $\mathfrak{D}_1, \mathfrak{D}_2$ in Figure 3. The superposition of $\mathfrak{D}_1$ and $\mathfrak{D}_2$ produces the L-net $\mathfrak{D} = \mathfrak{D}_1 \bigcup \mathfrak{D}_2$.

In fact, the set of chronicles of $\mathfrak{D}_1$ is the set of chronicles defined by each of its nodes

k, that is:

$$\{\overset{\alpha,0}{\bigcirc}, \overset{\alpha0,0}{\overset{|}{\bigcirc}}, (\xi 0, I), \ulcorner(\alpha 00, \{1\})\urcorner = \mathfrak{D}_1\}.$$

The set of chronicles of $\mathfrak{D}_2$ is:

$$\{\overset{\alpha,0}{\bigcirc}, \overset{\alpha0,0}{\overset{|}{\bigcirc}}, (\xi 0, J), \ulcorner(\alpha 00, \{2\})\urcorner = \mathfrak{D}_2\}.$$

The resulting union is:

$$\{\overset{\alpha,0}{\bigcirc}, \overset{\alpha0,0}{\overset{|}{\bigcirc}}, (\xi 0, I), (\xi 0, J), \mathfrak{D}_1, \mathfrak{D}_2\},$$

which corresponds to $\mathfrak{D}$.

A.3 Desequentialization

Example 1. Desequentializing either of the designs $\mathfrak{A}$ or $\mathfrak{X}$ in our previous example A.1, equipped with the only possible uniform decoration, yields the original L-net $\mathfrak{R}$. The only uniform decoration is, for both designs:

- $link(\alpha 00, 1) = \{\xi 0 * I\}$,
- $link(\alpha 00, 2) = \{\xi 0 * J\}$.

Example 2. Let us consider the design in Figure 4, where we just omit an obvious negative action at the place of .... The only uniform decoration is:

- $link(b) = \{\alpha 001, \xi 0 * I\}$,
- $link(c) = \{\alpha 002, \xi 0 * J\}$.

Following the desequentialization procedure, a few easy steps produce the two L-nets $\mathfrak{D}_1, \mathfrak{D}_2$, represented in Figure 5. Observe that we have a chronicle for each node; $\mathfrak{D}_1 \bigcap \mathfrak{D}_2$ is equal to $\{\ulcorner(\alpha, \{0\})\urcorner, \ulcorner(\alpha 0, \{0\})\urcorner\}$. We obtain $\mathfrak{D}'_1$ by adding the relation $(\xi 0, I) \leftarrow (\alpha 00, \{1\})$, and $\mathfrak{D}'_2$ in a similar way. Remember that we do not explicitly write the (non transitive) edge $\xi 0 \longleftarrow b$. The union $\mathfrak{D}'_1 \bigcup \mathfrak{D}'_2$ produces the L-net on the right-hand side of Figure 5.

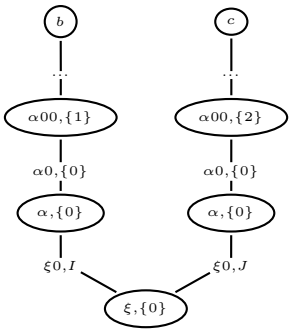


Figure 4:

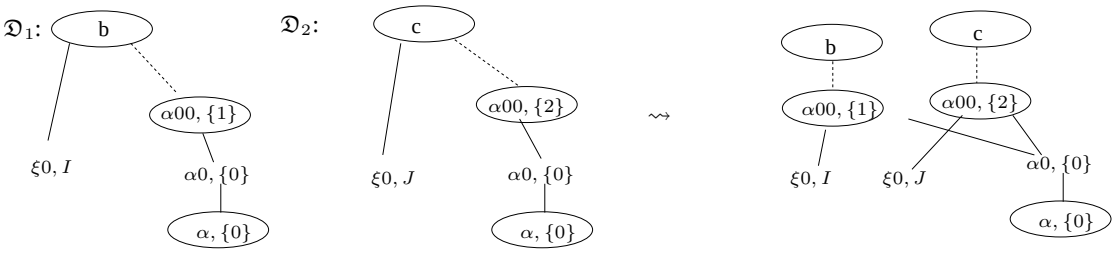


Figure 5:

A.4 A typed example: additives

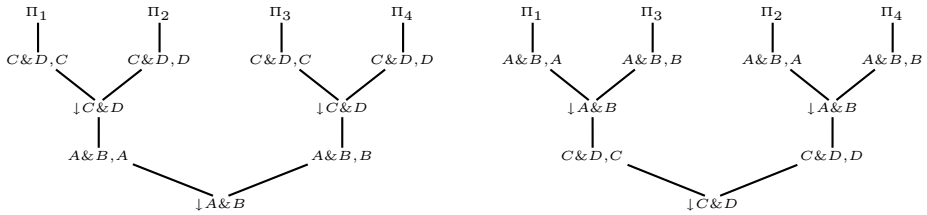
The following (typical) example with additives illustrates the relation between tree strategies and (parallel) L-nets.

Assume we have derivations $\Pi_1, \Pi_2, \Pi_3, \Pi_4$ of (respectively) $\vdash A, C$, $\vdash A, D$, $\vdash B, C$, $\vdash B, D$. In the sequent calculus (and in proof-nets with boxes [Gir87]) there are two distinct ways to derive $\vdash A\&B, C\&D$, and the two derivations differ only by commutations of the rules.

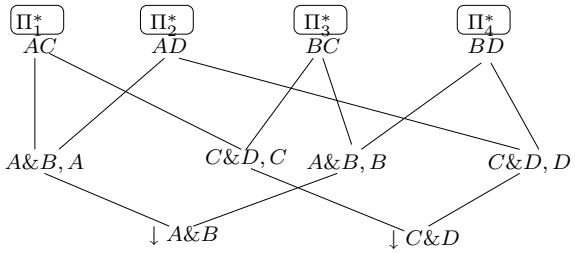
$$\dfrac{\dfrac{\dfrac{\Pi_1}{\vdash A, C}\quad\dfrac{\Pi_2}{\vdash A, D}}{\vdash A, C\&D}\,C\&D\qquad\dfrac{\dfrac{\Pi_3}{\vdash B, C}\quad\dfrac{\Pi_4}{\vdash B, D}}{\vdash B, C\&D}\,C\&D}{\vdash A\&B, C\&D}\,A\&B$$

$$\dfrac{\dfrac{\dfrac{\Pi_1}{\vdash A, C}\quad\dfrac{\Pi_2}{\vdash A, D}}{\vdash A\&B, C}\,A\&B\qquad\dfrac{\dfrac{\Pi_3}{\vdash B, C}\quad\dfrac{\Pi_4}{\vdash B, D}}{\vdash A\&B, D}\,A\&B}{\vdash A\&B, C\&D}\,C\&D$$

The same phenomenon can be reproduced in the setting of designs, or in the setting of polarized linear logic [Lau02]. Very similar to the above derivations are the two following (typed) designs, where we introduce some $\downarrow$'s in order to fit into our polarized setting. We write formulas instead of addresses, to make the example easier to grasp.



The desequentialization of either of the trees above is the following $\mathrm{L_S}$-net $\mathfrak{R}$:



Conversely, when sequentializing $\mathfrak{R}$, we get either one or the other tree back, depending on whether we choose to start from $A\&B$ or from $C\&D$. Notice that both $A\&B$ and $C\&D$ are splitting.

B Sequent calculus presentation of L-forests and decoration

In this section, we recall the sequent calculus for designs [Gir01] (see also [Cur06]). We add a rule which corresponds to the MIX rule, and we examine the correspondence between such extended designs and L-forests. *In this section,* Π *will range over sequent calculus proofs.* This does not conflict with our use of Π to denote L-forests, since we show here in some detail that they are essentially one and the same thing.

Girard's original sequent calculus for designs is the following (an interface is called well-formed if it consists of pairwise disjoint addresses with respect to the prefix ordering):

Daimon: ($\vdash \Lambda$ well formed)

$$\frac{}{\vdash \Lambda}\ \dagger$$

Positive rule ($I \subseteq \omega$ finite, one premise for each $i \in I$, all Λ_i's pairwise disjoint and included in Λ, $\vdash \xi, \Lambda$ well formed):

$$\frac{\cdots \quad \xi i \vdash \Lambda_i \quad \cdots}{\vdash \xi, \Lambda}\ (\xi, I)^+$$

Negative rule ($\mathcal{N} \subseteq \mathcal{P}_f(\omega)$ possibly infinite, one premise for each $J \in \mathcal{N}$, all Λ_J's included in Λ, $\xi \vdash \Lambda$ well formed):

$$\frac{\cdots \quad \vdash \xi * J, \Lambda_J \quad \cdots}{\xi \vdash \Lambda}\ \{(\xi, J)^- :\ J \in \mathcal{N}\}$$

xhere $\xi * J$ stands for the set of the ξj's ($j \in J$).

We add the following rule (which is a type-free counterpart of the MIX rule from linear logic [Gir87]):

MIX ($\vdash \Lambda_1, \ldots, \Lambda_n$ well formed, all Λ_m's pairwise disjoint)

$$\frac{\vdash \Lambda_1 \quad \ldots \quad \vdash \Lambda_n}{\vdash \Lambda_1, \ldots, \Lambda_n}\ \mathit{MIX}$$

Remark B.1 *The negative rule conveys some inherent weakening. Each action $(\xi, J)^-$ creates simultaneously all the addresses ξj ($j \in J$), which are recorded in the sequent, regardless of whether they will be used or not.*

Applications of the rule Daimon yield positive leaves in a proof tree. We shall also consider as a positive leaf any proof tree of the following form:

$$\frac{\ldots \quad \dfrac{}{\xi i \vdash \Lambda_i}\ \emptyset \quad \ldots}{\vdash \xi, \Lambda}\ (\xi, I)^+$$

where all negative rules are applied with $\mathcal{N}$ empty. We shall write simply:

$$\frac{}{\vdash \xi, \Lambda}\ (\xi, I)^+$$

We now briefly review how we can translate (in this extended setting) a sequent calculus proof Π into an L-forest $\overline{\Pi}$. The translation satisfies the following invariant. A proof of a sequent $\vdash \Lambda$ translates to a forest whose roots (or conclusions) are on distinct addresses of Λ, and a proof of a sequent $\xi \vdash \Lambda$ translates to a forest all of whose conclusions are of address ξ. The definition is by induction, according to the last rule of the proof. We use the syntax introduced in Section 7. We omit the (easy) proof that $\overline{\Pi}$ is an L-forest.

- Daimon. Then $\overline{\Pi} = \dagger$.
- $(\xi, I)^+$. Then $\overline{\Pi} = \bigcup_{i \in I} ((\xi, I)^+ \circ \overline{\Pi_i})$, where the Π_i's are the proofs of the sequents $\xi i \vdash \Lambda_i$ ($i \in I$). Note that $\overline{\Pi}$ is a tree.
- $\{(\xi, J)^- : J \in \mathcal{N}\}$. Then $\overline{\Pi} = \bigcup_J ((\xi, J)^- \centerdot \overline{\Pi_J})$, where the Π_J's are the proofs of the sequents $\vdash \xi * J, \Lambda_J$.
- MIX. Then $\overline{\Pi} = \biguplus_m \overline{\Pi_m}$, where the Π_m's are the proofs of the sequents $\vdash \Lambda_m$'s.

It should be clear that the rules of $\overline{\Pi}$ (as defined in Section 3) correspond univoquely to the occurrences of rules in Π. By going from Π to $\overline{\Pi}$, we have just forgotten all sequent informations except at the root. We now examine the converse direction, from L-forests to sequent calculus proofs.

Proposition B.2 *The mapping $\Pi \mapsto \overline{\Pi}$ is onto. More precisely, for every L-forest $\mathfrak{D}$ there exists a* uniform *sequent calculus proof Π such that $\mathfrak{D} = \overline{\Pi}$, where a uniform proof is a proof in which the positive and negative rules are constrained as follows:*

$$\frac{\cdots \quad \xi i \vdash \Lambda_i \quad \cdots}{\vdash \xi, \bigcup_i \Lambda_i} \ (\xi, I)^+$$

$$\frac{\cdots \quad \vdash \xi * J, \Lambda \quad \cdots}{\xi \vdash \Lambda} \ \{(\xi, J)^- : J \in \mathcal{N}\}$$

i.e., we require that in the positive rule, no address is lost ($\Lambda = \bigcup_i \Lambda_i$), and, in the negative rule, all Λ_J's are chosen maximal (and equal to Λ).

Proof. We have to extend the setting of [Gir01] (see also [Cur06]) from designs to L-forests, and to make sure that the target is restricted to uniform proofs. Let $\mathfrak{D}$ be an L-forest. There are four cases. In each case, we sketch how to (coinductively) generate the final rule of the proof (we give more details for the quite similar proof of Proposition B.4 (2) below).

1. If $\mathfrak{D}$ is a leaf, then the associated proof is its interface.
2. If $\mathfrak{D}$ is a positive L-forest with more than one root, then we transform each of the trees $\mathfrak{E}_j$ of $\mathfrak{D}$ into a proof of $\vdash \Lambda'_j$, where each Λ'_j consists of the minimal addresses used in $\mathfrak{E}_j$, and then we accommodate the constraint $\Lambda = \bigcup_j \Lambda_j$ by dispatching arbitrarily any $\xi \in \Lambda \setminus (\bigcup_i \Lambda'_i)$ to exactly one of the Λ'_j's, yielding suitable Λ_j's.
3. If $\mathfrak{D}$ is positive and is a tree, then we proceed essentially as in the previous case.
4. If $\mathfrak{D}$ is negative on interface $\xi \vdash \Lambda$, then it is easily seen that each $\vdash \xi * J, \Lambda$ is an interface for the corresponding subtree of $\mathfrak{D}$, so that we can carry on the construction.

□

Remark B.3 *The uniform proof discipline described in the statement of Proposition B.2 corresponds to pushing weakening maximally to the leaves.*

The assignment of a sequent calculus proof to an L-forest is non-deterministic, i.e., the map $\Pi \to \overline{\Pi}$ is not injective. With decorations (cf. Section 6), we get a bijective correspondence. We recall (cf. Definitions 6.1, 6.4, and 9.4) that

- a decorated L-forest is an L-forest $\mathfrak{D}$ in which all leaves k are equipped with a finite set $link(k)$ of addresses (called the link set of k), in such a way that the conditions on L-nets hold with respect to all addresses (including those in the link sets);
- a well decorated L-net is a decorated L-net $\mathfrak{D}$ such that all addresses of the interface, and all addresses generated by a negative action of $\mathfrak{D}$ are used in $\mathfrak{D}$, i.e., appear as a label of the underlying L-net or in a link set;
- a uniformly decorated L-forest is a decorated L-forest $\mathfrak{D}$ such that every (maximal) slice of $\mathfrak{D}$ is well decorated.

Proposition B.4 *1. Well decorated L-forests are in one-to-one correspondence with sequent calculus proofs subject to the restriction that in all applications of the positive rule (resp. negative rule) we have* $\bigcup_{i\in I} \Lambda_i = \Lambda$ *(resp.* $\bigcup_{J\in\mathcal{N}} \Lambda_J = \Lambda$*).*

2. Uniformly decorated L-forests are in one-to-one correspondence with the proofs subject to the further restriction of uniformity (cf. Proposition B.2).

Proof. 1. The correspondence in one direction is obtained by adapting $\overline{\Pi}$, as follows: for each leaf with conclusion $\vdash \Lambda$ (resp. $\vdash \xi, \Lambda$) obtained by an application of $\dagger$ (resp. $(\xi, I)^+$), the translation is now $\dagger$ (resp. $(\xi, I)^+$) with $link(\dagger) = \Lambda$ (resp. $link((\xi, I)^+) = \Lambda$). It is easily checked that the respective restrictions on the construction of proofs ensure that $\overline{\Pi}$ is a well decorated or a uniformly decorated L-forest.

Conversely, given a well decorated L-forest $\mathfrak{D}$, we associate (deterministically) a proof, as follows. An invariant of the construction is that the minimal addresses (in the prefix ordering) used in $\mathfrak{D}$ form the final sequent of the associated proof $\underline{\mathfrak{D}}$.

- $\mathfrak{D} = \biguplus_i \mathfrak{C}_i$ has several positive conclusions. Then we translate each of the trees $\mathfrak{C}_1, \ldots, \mathfrak{C}_n$ of $\mathfrak{D}$, yielding proofs of sequents $\vdash \Lambda_1, \ldots, \vdash \Lambda_n$. By the invariant, we know that all addresses in Λ_m are used in $\mathfrak{C}_i$, and then by condition Additives we are sure that the Λ_m's are distinct. Therefore we can apply the MIX rule, and we define $\underline{\mathfrak{D}}$ as
$$\frac{\underline{\mathfrak{C}_1} \quad \ldots \quad \underline{\mathfrak{C}_n}}{\vdash \Lambda_1, \ldots, \Lambda_n}\ MIX$$
- $\mathfrak{D}$ has conclusion $k = \dagger$ with $link(k) = \Lambda$. Then $\underline{\mathfrak{D}}$ is
$$\frac{}{\vdash \Lambda}\ \dagger$$

- $\mathfrak{D}$ has only one positive conclusion $(\xi, I)^+$. If $\mathfrak{D}$ is reduced to a leaf, then we proceed as in the previous case. If $\mathfrak{D} = \bigcup_{\{j \in J\}} ((\xi, I)^+ \circ \mathfrak{D}_j)$, for some $J \subseteq I$, by the same reasoning as in the first case, we have that the $\underline{\mathfrak{D}_j}$'s are proofs of sequents $\vdash \Lambda_j$, for paiwise disjoint Λ_j's. Then we define $\underline{\mathfrak{D}}$ as

$$\frac{\ldots \quad \underline{\mathfrak{D}_j} \quad \ldots \quad \overline{\xi k \vdash}^{\ \emptyset} \quad \ldots}{\vdash \Lambda} \ (\xi, I)^+$$

  where j (resp. k) ranges over J (resp. $I \setminus J$), and where Λ is the union of the Λ_i's.

- $\mathfrak{D} = \bigcup_J ((\xi, J) \ . \ \mathfrak{D}_J)$. By construction of the decoration, all addresses ξj ($j \in J$) are minimal in $\mathfrak{D}_J$. By this remark, and applying induction, we get that $\underline{\mathfrak{D}_J}$ is a proof of a sequent of the form $\vdash \xi * J, \Lambda_J$. Then we define $\underline{\mathfrak{D}}$ as

$$\frac{\ldots \quad \underline{\mathfrak{D}_J} \quad \ldots}{\xi \vdash \Lambda} \ \{(\xi, J)^- : \ J \in \mathcal{N}\}$$

  where $\mathcal{N} = \{J : \ (\xi, J)^-$ is a root of $\mathfrak{D}\}$ and Λ is the union of the Λ_J's.

The final sequent of $\mathfrak{D}$ is its interface, by the invariant of the translation, and by the initial construction of the decoration, that takes the basis of $\mathfrak{D}$ into account.

It is straightforward to prove that this transformation is inverse to the transformation $\Pi \mapsto \overline{\Pi}$.

2. This correspondence is simply obtained by restricting the correspondence to uniformly decorated L-nets and to uniform proofs. □

Corollary B.5 *Every L-forest can be uniformly decorated, and hence a fortiori well decorated.*

Proof. To an L-forest $\mathfrak{D}$, we can associate a uniform proof by Proposition B.2, and then a uniform decoration, by Proposition B.4. □

Remark B.6 *Note that the bijective correspondences of Proposition B.4 induce a bijective correspondence between the link sets used in the decoration of an L-forest and the generalized axioms used in the corresponding sequent calculus proof.*

C Proof of the Splitting Lemma

In this section, we prove Lemma 4.11, namely that every $\mathrm{L_S}$-net $\mathfrak{D}$ has a splitting conclusion.

We recall from section 4.4 that a negative rule $W = \{\ldots, w_I, \ldots\}$ of an L-net $\mathfrak{D}$ is called splitting if either it is conclusion of the $\mathrm{L_S}$-net (each w_I is a root), or if deleting all the edges $w_I \to w$ there is no more connection (i.e., no sequence of consecutive edges) between any of the w_I's and w, and that a positive conclusion of $\mathfrak{D}$ is called splitting if all negative rules just above it are splitting.

If $\mathfrak{D}$ is negative, then the Splitting Lemma holds vacuously. For positive $\mathfrak{D}$'s, we first establish the following Negative Splitting Lemma.

Lemma C.1 (Negative Splitting Lemma) *Every positive* $\mathrm{L_S}$*-net* $\mathfrak{D}$ *which has a negative rule has a splitting negative rule among all the negative rules of level 1 (i.e., located just above a conclusion).*

Our proof is an adaptation to our setting of the proof of the similar lemma in [HvG05]. A switching path $x_0 \dots x_n$ is called *strong* (and denoted $x_0 \Leftarrow x_n$) if either its last node is positive or if it ends upwards in the last node. Strong switching paths satisfy the following concatenation property: if γ_1 is a strong switching path and γ_2 is a switching path such that their concatenation $\gamma_1\gamma_2$ is a rule path (cf. section 3.7), then $\gamma_1\gamma_2$ is switching, and if moreover γ_2 is strong, then $\gamma_1\gamma_2$ is strong.

Definition C.2 (Domination) *Given an* $\mathrm{L_S}$*-net* $\mathfrak{D}$*, a negative rule* X *and a finite set of nodes* G*, we say that* G *is an* X*-zone if for every* $z \in G$ *there are nodes* $x \in X$ *and* x' *such that* $x \leftarrow x' \Leftarrow z$*, where the path* $x' \Leftarrow z$ *is included in* G*. Given a node* z *of* $\mathfrak{D}$*, we say that* X dominates z*, denoted* $X \trianglelefteq_{\mathfrak{D}} z$ *(or simply* $X \trianglelefteq z$*), if there exists an* X*-zone* G *in* $\mathfrak{D}$ *such that* $z \in G$*. We say that the zone* G *and the sequence* $x \leftarrow x' \Leftarrow z$ *witness* $X \trianglelefteq z$*.*

The following statement lists some simple consequences of the definition of domination.

Lemma C.3 *1.* X*-zones are closed under unions.*

2. *If* $X \trianglelefteq z$ *is witnessed by a sequence* $x \leftarrow x' \Leftarrow z$*, then* X *dominates every node of the path* $x' \Leftarrow z$*.*
3. *If* $x \leftarrow y$ *for some* $x \in X$*, then* $X \trianglelefteq y$*.*
4. *Given a negative rule* W*, if* X *dominates a node* $w \in W$ *then* X *dominates all* $w' \in W$*.*
5. *If* $X \trianglelefteq y$*, and if* $y \leftarrow z$*, or if* $z \leftarrow y$ *and* z *is not negative, then* $X \trianglelefteq z$*.*

Proof. The first three parts of the statement are obvious. Let $X \trianglelefteq w$ be witnessed by G and $x \leftarrow x' \Leftarrow w$. By definition of strong, the path $x' \Leftarrow w$ terminates with $k \leftarrow w$. Then we obtain a strong path to any $w' \in W$ by just replacing the last edge with $k \leftarrow w'$. It follows that $G \cup W$ is an X-zone, and therefore $(\forall\, w' \in W \;\; X \trianglelefteq w')$.

We now prove the last assertion of the statement. Let G and $x \leftarrow a \Leftarrow y$ be a witness of $X \trianglelefteq y$. If z does not belong to a rule that intersects the sequence $a \dots y$, then the sequence $a \dots yz$ is a path, that is switching by the assumptions. Hence $G \cup \{z\}$ and $x \leftarrow a \Leftarrow z$ are a witness for $X \trianglelefteq z$. If z intersects the sequence $a \dots y$, then we conclude using the second and fourth parts of the statement. □

Thanks to Lemma C.3, we shall henceforth safely assume that X-zones are *rule-saturated*, i.e. are unions of rules.

The notion of domination extends to rules. Let W be a rule. We write $W_1 \trianglelefteq W_2$ if there exists $w_2 \in W_2$ such that $W_1 \trianglelefteq w_2$ (or, equivalently, if $W_1 \trianglelefteq w_2$ for all $w_2 \in W_2$, by Lemma C.3). If X is not dominated, we say that it is *free*.

Lemma C.4 *Domination is transitive.*

Proof. Assume $X \trianglelefteq Y$ and $Y \trianglelefteq Z$, witnessed by (rule-saturated) G and $x \leftarrow a \Leftarrow y$, and by G' and $y' \leftarrow b \Leftarrow z$ ($z \in Z$, $y, y' \in Y$, $x \in X$). By Lemma C.3, we can assume $y = y'$. It is enough to show that $G' \cup G$ is an X-zone, and for this we only have to consider $z' \in G' \setminus G$, if any, witnessed by $y'' \leftarrow b' \Leftarrow z'$. Let z'' be the last node in the sequence $y''b' \dots z'$ which is in G, witnessed by $x' \leftarrow a' \Leftarrow z''$. Then, concatenating with the rest of the sequence from (the successor of) z'' to z', we obtain a path (by construction, and because G is rule-saturated). This path is strong and switching because its constituents are. □

Lemma C.5 *Let W be a negative rule. If $w \in W$ is below a node of a switching cycle C, then W dominates all nodes of the cycle. If $w_1, w_2 \in W$ are such that $w_i \overset{+}{\leftarrow} z_0$ and $w_j \overset{+}{\leftarrow} z_n$, then W dominates every node in a switching path from z_0 to z_n.*

Proof. We prove the second part of the statement (the reasoning is the same for the first part). Let G_1 (resp. G_2) be the set of nodes on a path going up from w_1 to z_0 (resp. from w_2 to z_n). We shall show that $G_1 \cup G_2 \cup C$ is a W-zone. That G_1 and G_2 are W-zones follows readily from Lemma C.3. Let $z \in C$. Because C is switching, we have either $z_0 \Leftarrow z$ or $z_n \Leftarrow z$. Suppose that we have, say, $z_0 \Leftarrow z$. Let z_1' be the first node on the way up from w_0 to z_0 that belongs to a rule intersecting $z_0 \Leftarrow z$ at some z_2'. Then the sequence obtained by going up from w_1 to z_2' and then to z is witnessing $W \trianglelefteq z$. □

Lemma C.6 *Let $\mathfrak{D}$ be a* finite $\mathrm{L_S}$*-net. If a rule X intersects a switching cycle, then X is dominated by an additive rule W which intersects no switching cycle.*

Proof. We construct a sequence of negative rules W_i as follows. We set $X = W_0$. If W_i intersects a switching cycle, then applying the condition Cycles gives us a rule W_{i+1}. We have $W_{i+1} \trianglelefteq W_i$, by Lemma C.5. At each iteration the union of the cycles increases strictly, and hence by finiteness of $\mathfrak{D}$ we eventually reach some negative rule $W_n = W$ which intersects no switching cycle. Moreover, we have $W \trianglelefteq X$ by Lemma C.4. □

Due to the finiteness condition in the previous lemma, we shall first establish the Splitting Lemma in the finite case, and then show how to lift the result also to infinite $\mathrm{L_S}$-nets.

Lemma C.7 *If $X \trianglelefteq X$, then X is in a switching cycle.*

Proof. If $X \trianglelefteq X$, we have $x \leftarrow a \Leftarrow x$ for some $x \in X$. Then we can close the path from a to x with the edge $x \leftarrow a$. Because the path from a to x is strong, the cycle is switching. □

Proposition C.8 *Let $\mathfrak{D}$ be a finite* $\mathrm{L_S}$*-net. Every negative rule is either free or dominated by a free negative rule. As a consequence, if there are negative rules, there exists a free negative rule.*

Proof. The proof is by contradiction. Let X be a negative rule that is neither free nor dominated by a free negative rule. We shall build an infinite sequence of rules X_i which are all distinct, are all not free, and are such that

$$\dots X_{i+1} \trianglelefteq X_i \trianglelefteq \dots X_1 \trianglelefteq X_0 \trianglelefteq X \ ,$$

contradicting the finiteness of $\mathfrak{D}$. We take $X_0 = X$. By assumption, X_0 is not free. Suppose that we have constructed the sequence up to X_i. We distinguish two cases:

1. If X_i is not in a switching cycle, we choose any X_{i+1} such that $X_{i+1} \trianglelefteq X_i$ (this is possible since X_i is not free by induction hypothesis). This X_{i+1} is fresh as otherwise we would have by transitivity $X_i \trianglelefteq X_i$, contradicting our assumption on X_i, by Lemma C.7.

2. If X_i is in a switching cycle, then by Lemma C.6 we can choose a rule X_{i+1} such that X_{i+1} intersects no switching cycle. This X_{i+1} is fresh as otherwise we would have by transitivity $X_{i+1} \trianglelefteq X_{i+1}$, and this contradicts our assumption about the choice of X_{i+1}, by Lemma C.7.

In both cases, we have constructed a fresh X_{i+1} such that $X_{i+1} \trianglelefteq X_i$. Moreover, by transitivity, $X_{i+1} \trianglelefteq X$, from which it follows that X_{i+1} is not free. □

Let X, Y be distinct negative rules.

- We write $X \longleftrightarrow Y$ if there is a switching path $z_0 \dots z_n$ (called witnessing path) such that $x \leftarrow z_0$ and $z_n \rightarrow y$, for some $x \in X$ and $y \in Y$.
- We write $X \rightarrow\!\!\leftarrow Y$ if X, Y belong to the same bipole, hence $x \rightarrow k$ and $y \rightarrow k$, for some k and all $x \in X$ and $y \in Y$.

Lemma C.9 *If X, Y and Z are negative rules such that $X \neq Y$, $X \trianglelefteq Z$ and $Y \trianglelefteq Z$, then $X \longleftrightarrow Y$.*

Proof. Consider $x \leftarrow a \trianglelefteq z$ (for some $x \in X$ and $z \in Z$). Let z' be the first node on the path from a to z such that $Y \trianglelefteq z'$ (and hence $y \leftarrow b \Leftarrow z'$ for some $y \in Y$). Then we get a path witnessing $X \longleftrightarrow Y$ by going from a to z' and then from z' to b. This sequence of nodes is a rule path since if it were not, there would be z_1 in the first portion and z_2 in the second portion belonging to the same rule, but we have that $Y \trianglelefteq z'$ implies $Y \trianglelefteq z_2$ which in turn implies $Y \trianglelefteq z_1$, contradicting the minimality of z'. It is switching since the path from b to z' is strong. □

Lemma C.10 *If X is a free negative rule of $\mathfrak{D}$ and does not split, then there exist free negative rules Y, Z of $\mathfrak{D}$ such that $X \rightarrow\!\!\leftarrow Y$ and $X \longleftrightarrow Z$.*

Proof. Let c be the node just below X. Since X does not split, for some $x \in X$ we can form a cycle (in the ordinary sense of graph theory, i.e. without the disjoint rules assumption) $xc \dots ax$, without using any edge between c and X other than $c \leftarrow x$. Since X is free, c is a conclusion of the net, and the next node on the cycle must be

some y such that $c \leftarrow y$. By construction, y belongs to a rule Y distinct from X, and thus we have $X \rightarrow\!\!\leftarrow Y$.

Next we observe that we cannot have $X \trianglelefteq x$, because X is free, and that $X \trianglelefteq a$ because $x \leftarrow a$ (since the only edge of $\mathfrak{D}$ out of x is already used). Let b be the first node, following the cycle in the direction $xc\ldots$, such that $X \trianglelefteq b$. The node z' before b must be negative and we must have $z' \leftarrow b$ as otherwise we would have $X \trianglelefteq z'$ by Lemma C.3. Let Z' be the rule to which z' belongs. Then we have $X \longleftrightarrow Z'$. If Z' is free, we can set $Z = Z'$. If Z' is not free, it dominates some free Z, by Proposition C.8, and we conclude by Lemma C.9 (since $X \trianglelefteq \{b\}$, and $Z \trianglelefteq \{b\}$ by transitivity). □

We are now able to prove the Negative Splitting Lemma for *finite* $\mathrm{L_S}$-nets, i.e. for $\mathrm{L_S}$-nets having finitely many nodes.

Proof (Negative Splitting Lemma, finite case). If the $\mathrm{L_S}$-net $\mathfrak{D}$ has no splitting negative rules, then all its conclusions must be positive, and starting from a free negative rule X_0 (whose existence is guaranteed by Proposition C.8), and using again and again Lemma C.10, we can build an infinite sequence $X_0 \rightarrow\!\!\leftarrow X_1 \longleftrightarrow X_2 \rightarrow\!\!\leftarrow \ldots$ where X_{i+1} is a free negative rule and $X_{i+1} \neq X_i$, for all i. Since there are only finitely many free negative rules, we have $X_i = X_j = X$ for some $i < j$. By the definition of the $\rightarrow\!\!\leftarrow$ and $\longleftrightarrow$ relations, we can form a switching sequence of nodes starting in $X_i = X$ and ending in $X_j = X$ which is nondegenerate (i.e. of length at least 2) since $X_i \neq X_{i+1}$. But this sequence is not guaranteed to be a rule path. To build a rule path, we take two nodes z_1 and z_2 at minimal distance in the sequence such that z_1 and z_2 belong to the same rule. Again, this distance is non-degenerate, as z_1 and z_2 cannot belong to the same path witnessing some $X_{2k+1} \longleftrightarrow X_{2k+2}$ ($i \leq 2k+1 < j$), and moreover, by the same reason, the path $z_1 z'_1 \ldots z'_2 z_2$ from z_1 to z_2 must cross some X_n. We distinguish two cases:

1. If $z'_1 \leftarrow z_1$ or $z'_2 \leftarrow z_2$, say, $z'_1 \leftarrow z_1$, then we also have $z'_1 \leftarrow z_2$, and adding this (reversed) edge to the path from z'_1 to z_2 yields a switching cycle. Then, by a (weakened form of) Lemma C.6, we obtain that X_n is dominated.

2. If $z_1 \leftarrow z'_1$ and $z_2 \leftarrow z'_2$, then we are in the situation of the (second part of the statement of) Lemma C.5, and we also obtain that X_n is dominated.

We have reached a contradiction, since X_n is free by construction.

The Negative Splitting Lemma holds actually for arbitrary $\mathrm{L_S}$-nets. The following definition and lemma ensure a finiteness condition, even if our $\mathrm{L_S}$-net $\mathfrak{D}$ is infinite.

Definition C.11 *Let $\mathfrak{D}$ be an $\mathrm{L_S}$-net whose conclusions are all positive. We denote by $Neg_1(\mathfrak{D})$ the set of negative rules that are just above a conclusion (i.e., the set of rules of level 1).*

Lemma C.12 *1. The set $Neg_1(\mathfrak{D})$ is finite.*

2. Every free (negative) rule is in $Neg_1(\mathfrak{D})$.

Proof. By the finiteness of the interface, there are finitely many conclusions, and since there are are only finitely many rules just above a positive rule, it follows that $Neg_1(\mathfrak{D})$ is finite. Suppose that W is a rule of level > 1, then, going down from W, we reach a rule W' that dominates W (that $W' \trianglelefteq W$ follows by repeated use of Lemma C.3). The second part of the statement follows. □

Proof (Negative Splitting Lemma, infinite case). Let $\mathfrak{D}$ be an infinite $\mathrm{L_S}$-net. We concentrate on $Neg_1(\mathfrak{D})$. For each pair $X, X' \in Neg_1(\mathfrak{D})$ such that $X \trianglelefteq X'$, we take an X-zone witnessing this domination. Let $\mathfrak{F}$ be a minimal (finite) $\mathrm{L_S}$-net containing these zones, obtained by possibly adding in a minimal way positive chronicles to the set $\mathfrak{F}'$ of all chronicles $\ulcorner k \urcorner$, where k ranges over the union of the zones (note that $\mathfrak{F}'$ is already a partial L-net by Lemma 4.1). By construction, $Neg_1(\mathfrak{F})$ consists of all sets $X \cap \mathfrak{F}$ such that $X \in Neg_1(\mathfrak{D})$ and $X \cap \mathfrak{F} \neq \emptyset$. Moreover, for any two $X, X' \in Neg_1(\mathfrak{D})$, we have, by construction of $\mathfrak{F}$:

$$X \trianglelefteq_{\mathfrak{D}} X' \qquad \Leftrightarrow \qquad (X \cap \mathfrak{F} \neq \emptyset\ ,\ X' \cap \mathfrak{F} \neq \emptyset\ ,\ \text{and}\ \ (X \cap \mathfrak{F}) \trianglelefteq_{\mathfrak{F}} (X' \cap \mathfrak{F}))\ .$$

It follows that if $(X \cap \mathfrak{F}) \in Neg_1(\mathfrak{F})$ is free (in $\mathfrak{F}$), then X is free (in $\mathfrak{D}$), using the fact that whenever a rule is dominated, it is dominated by a rule in $Neg_1(\mathfrak{D})$.

Now, suppose that there exists a negative rule X of $\mathfrak{D}$ that is neither free nor dominated by a free rule. We can assume $X \in Neg_1(\mathfrak{D})$ since this property is a fortiori true of any negative rule below. Then, as we noted above, $X \cap \mathfrak{F}$ is not free. Neither can $X \cap \mathfrak{F}$ be dominated by a free rule $X' \cap \mathfrak{F}$, because then we would have that X' is free and $X' \trianglelefteq_{\mathfrak{D}} X$. Therefore $X \cap \mathfrak{F}$ is neither free nor dominated by a free rule in $\mathfrak{F}$, contradicting the Negative Splitting Lemma (finite case).

We now prove the Splitting Lemma, as a consequence of the Negative Splitting Lemma.

Proof (Splitting Lemma). Let $\mathfrak{D}$ be an $\mathrm{L_S}$-net that has only positive conclusions. We define $size(\mathfrak{D})$ as:

- 0 if at least one of the positive conclusions of $\mathfrak{D}$ is a leaf, and otherwise as
- the cardinal of the set of level 1 negative rules of $\mathfrak{D}$.

Since $\mathfrak{D}$ has finitely many positive conclusions, the size of $\mathfrak{D}$ is finite, even if $\mathfrak{D}$ is not finite.

We apply the Negative Splitting Lemma to $\mathfrak{D}$. We select a splitting negative rule X. Since $\mathfrak{D}$ has no negative conclusion, X is just above a conclusion k of $\mathfrak{D}$. We delete the edges from x to k, for all $x \in X$.

Let us call $\mathfrak{D}_X$ the union of the connected components (in the ordinary unoriented graph-theoretic sense) of the elements of X, and $\mathfrak{D}_k$ the rest of the graph, which contains k. We prove that $\mathfrak{D}_X$ and $\mathfrak{D}_k$ are $\mathrm{L_S}$-nets. Let $\mathfrak{D}'$ stand for either $\mathfrak{D}_X$ or $\mathfrak{D}_k$. We note that if $c \in \mathfrak{D}'$, then $\mathfrak{D}'$ contains every path of $\mathfrak{D}$ starting from c that does not go through one of the deleted edges. It follows that $\ulcorner c \urcorner_{\mathfrak{D}'}$ possibly differs from $\ulcorner c \urcorner_{\mathfrak{D}}$

only by not containing k. We are thus almost in the situation of Lemma 4.2, modulo straightforward adaptations. Let us look for example at the condition Additives: the path leading down from k_1 to w_1 (resp. from k_2 to w_2) does not go through k, and hence belongs to $\ulcorner k_1 \urcorner_{\mathfrak{D}'}$ (resp. $\ulcorner k_2 \urcorner_{\mathfrak{D}'}$), so condition Additives in $\mathfrak{D}'$ is inherited from condition Additives in $\mathfrak{D}$.

We have that $size(\mathfrak{D}_k) < size(\mathfrak{D})$, because every conclusion k' of $\mathfrak{D}_k$ is a conclusion of $\mathfrak{D}$, and every negative rule of $\mathfrak{D}_k$ is a negative rule of $\mathfrak{D}$. Moreover, every free splitting negative rule of $\mathfrak{D}_k$ is a splitting negative rule of $\mathfrak{D}$: indeed, if $\mathfrak{D}_k$ splits into $\mathfrak{D}_{k'}$ and $\mathfrak{D}_{X'}$, then $\mathfrak{D}$ splits into $(\mathfrak{D}_{k'} \cup \mathfrak{D}_X)$ and $\mathfrak{D}_{X'}$. We are now ready to prove that $\mathfrak{D}$ has a positive splitting conclusion, by induction on $size(\mathfrak{D})$:

- Base case. Obvious. Since one positive conlusion is a leaf, it is splitting vacuously.
- Induction case. Let k' be a splitting positive conclusion of $\mathfrak{D}_k$. If $k' \neq k$, then it is also a splitting positive conclusion of $\mathfrak{D}$, since every negative rule just above k' is splitting in $\mathfrak{D}_k$, hence in $\mathfrak{D}$. If $k' = k$, let Y be a negative rule just above k. If $Y = X$, it is splitting by construction; if $Y \neq X$, Y belongs to $\mathfrak{D}_k$ by construction, and hence Y is splitting in $\mathfrak{D}_k$, and hence also in $\mathfrak{D}$, so that k is a positive splitting conclusion. This completes the proof.

References

[AC98] R. Amadio and P.-L. Curien. *Domains and Lambda-calculi*. Cambridge University Press, 1998.

[AJ92] S. Abramsky and R. Jagadeesan. Games and full completeness for multiplicative linear logic. In *Proc. FST-TCS, Springer Lect. Notes in Comp. Sci. 652*, 1992.

[AM99] S. Abramsky and P.-A. Melliès. Concurrent games and full completeness. In *Proc. of LICS'99 (Logic in Computer Science)*. IEEE Computer Society Press, 1999.

[And01] J.-M. Andreoli. Focussing and proof construction. *Annals of Pure and Applied Logic*, 2001.

[And02] J.-M. Andreoli. Focussing proof-net construction as a middleware paradigm. In *Proceedings of CADE'02 (Conference on Automated Deduction)*, 2002.

[CF05] P.-L. Curien and C. Faggian. L-nets, strategies and proof-nets. In *CSL'05 (Computer Science Logic)*, volume 3634 of *LNCS*. Springer, 2005.

[Cur98] P.-L. Curien. Abstract Böhm trees. *Mathematical Structures in Computer Science*, 8(6), 1998.

[Cur06] P.-L. Curien. Introduction to linear logic and ludics, part II. *Advances of Mathematics, China*, 35(1), 2006.

[DGF06] P. Di Giamberardino and C. Faggian. A jump from parallel to sequential proofs. Multiplicatives. In *CSL'06 (Computer Science Logic)*, volume 4207 of *LNCS*, 2006.

[Fag] C. Faggian. A common language for (sequential and parallel) strategies and proof-nets. draft.

[Fag02] C. Faggian. Travelling on designs: ludics dynamics. In *CSL'02*, volume 2471 of *LNCS*. Springer Verlag, 2002.

[FH02] C. Faggian and M. Hyland. Designs, disputes and strategies. In *CSL'02*, volume 2471 of *LNCS*. Springer Verlag, 2002.

[FM05] C. Faggian and F. Maurel. Ludics nets, a game model of concurrent interaction. In *Proc. of LICS'05*. IEEE Computer Society Press, 2005.

[Gir87] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[Gir99] J.-Y. Girard. On the meaning of logical rules I: syntax vs. semantics. In Berger and Schwichtenberg, editors, *Computational logic*, NATO series F 165, pages 215–272. Springer, 1999.

[Gir01] J.-Y. Girard. Locus solum. *Mathematical Structures in Computer Science*, 11:301–506, 2001.

[HO00] M. Hyland and L. Ong. On full abstraction for PCF. *Information and Computation*, 2000.

[HS02] M. Hyland and A. Schalk. Games on graphs and sequentially realizable functionals. In *Proc. of LICS'02*, pages 257–264. IEEE Computer Society Press, 2002.

[HvG05] D. Hughes and R. van Glabbeek. Proof nets for unit-free multiplicative-additive linear logic. *ACM Transactions on Computational Logic*, 6(4):784–842, 2005.

[Hyl01] M. Hyland. A category of partial orders and merging. Oral communication at Rencontres Franco-Américaines de Mathématiques (AMS-SMF), July 2001.

[JR97] B. Jacobs and J.J.M.M. Rutten. A tutorial on (co)algebras and (co)induction. *Bulletin of EATCS*, 62:222–259, 1997.

[Lau] O. Laurent. Game semantics for first-order logic. draft.

[Lau02] O. Laurent. *Etude de la polarisation en logique*. PhD thesis, Université Aix-Marseille II, 2002.

[Mel04] P.-A. Melliès. Asynchronous games 2 : The true concurrency of innocence. In *Proc. CONCUR'04*, volume 3170 of *LNCS*. Springer Verlag, 2004.

[MW05] G. McCusker and M. Wall. Categorical and game semantics for SCIR. In *Proc. Galop'05 (Games for Logic and Programming Languages), workshop affiliated with Etaps'05*, pages 157–178, 2005.

[SPP05] A. Schalk and J.J. Palacios-Perez. Concrete data structures as games. In *CTCS'04 (Category Theory and Computer Science*, volume 122 of *Electr. Notes Theor. Comput. Sci.*, 2005.