Appears in *the Journal of Logic and Computation, 2 (3) 1992*

Logic Programming with Focusing Proofs in Linear Logic

Jean-Marc Andreoli
European Computer Industry Research Centre, Munich, Germany

Abstract

The deep symmetry of Linear Logic [18] makes it suitable for providing abstract models of computation, free from implementation details which are, by nature, oriented and non symmetrical. I propose here one such model, in the area of Logic Programming, where the basic computational principle is

Computation = Proof search.

Proofs considered here are those of the Gentzen style sequent calculus for Linear Logic. However, proofs in this system may be redundant, in that two proofs can be syntactically different although identical up to some irrelevant reordering or simplification of the applications of the inference rules. This leads to an untractable proof search where the search procedure is forced to make costly choices which turn out to be irrelevant. To overcome this problem, a subclass of proofs, called the "focusing" proofs, which is both complete (any derivable formula in Linear Logic has a focusing proof) and tractable (many irrelevant choices in the search are eliminated when aimed at focusing proofs) is identified. The main constraint underlying the specification of focusing proofs has been to preserve the symmetry of Linear Logic, which is its most salient feature. In particular, dual connectives have dual properties with respect to focusing proofs.

Then, a programming language, called LinLog, consisting of a fragment of Linear Logic, in which focusing proofs have a more compact form, is presented. LinLog deals with formulae which have a syntax similar to that of the definite clauses and goals of Horn logic, but the crucial difference here is that it allows clauses with multiple atoms in the head, connected by the "par" (multiplicative disjunction). It is then shown that the syntactic restriction induced by LinLog is not performed at the cost of any expressive power: a mapping from full Linear Logic to LinLog, preserving focusing proofs, and analogous to the normalization to clausal form for Classical Logic, is presented.

Contents

1 Introduction

Linear Logic [18] has been used in various areas of computing. Its computational appeal basically stems from its deep symmetry, expressed by the duality operator $^{\perp}$ and the De Morgan laws (which hold without precluding the logic from being constructive). In the case of Functional programming, for instance, this symmetry leads to a formulation of the cut elimination procedure (based on proof nets [22], or proof expressions [1]) in which the role of input and output of a computation are blurred. Instead, a computation is viewed as a manipulation of resources: "input" corresponds to a consumption of a resource and "output" to a production, and these two operations are strictly dual. This mechanism of consumption-production of resources, which is inherently concurrent, is analogous to applications of rewrite rules, formalized in the (non-equational) framework of rewrite logic [25]. Concurrent resource manipulations also lie at the core of the applications of Linear Logic to Petri nets [12, 23] and can be given a metaphorical interpretation in terms of chemical reactions [9, 10].

A similar, resource based, approach to computation applies in the area of Logic Programming, especially to concurrent logic programming systems. In this framework, a computation consists of a set of processes running in parallel, possibly exchanging information. Each process is entirely characterized, at any given time, by its state, which is a collection of available resources. State transitions, i.e. the elementary steps in a process evolution, consist of resource manipulations, such as production, consumption, duplication...

Sequent style proofs offer a satisfactory tool to formalize the history of an execution during a certain time interval. Indeed, the evolution of each computational process present at the beginning of the interval can be represented as a proof-tree, read bottom-up, whose root (conclusion) represents the state of the process at the beginning of the interval, and whose leaves (hypotheses) represent the resulting states at the end of the interval. The nodes inside the proof represent the intermediate states of the evolution of the process during the interval. There might be more than one resulting state in a transition, or none at all. This corresponds to the possibility for a process either to create several new processes or to terminate. A sequent system, which describes the correct inferences in proofs, can therefore be viewed as a formal specification of allowed process state transitions.

This process view of logic programming has already been taken in such systems as Concurrent Prolog [30, 29], Parlog [13, 15] or GHC [31]. However, these languages, based on Classical Logic (more precisely, its fragment restricted to Horn clauses), offer a very limited structure for representing process states. Indeed, a state must be encoded as a simple atom, i.e. a first-order term. It has been shown in [3, 4, 7, 6] that this kind of representation is inadequate both from the point of view of knowledge sharing and communication between processes, mainly because the tree structure of terms enforces an artificial hierarchical order on their components and this, in turn, results in unwanted sequentiality in the access to these components. On the other hand, Linear Logic offers a far richer structure for process state representation: a state is encoded as a sequent, i.e. a multiset of formulae (being unordered, multisets are more suited to concurrent access).

By precluding the possibility of freely duplicating or deleting formulae in sequents, by application of the structural rules of Contraction and Weakening, Linear Logic bestows on formulae the status of *restricted* resources. Hence, this status is explicit in Linear Logic, and does not derive, as in Prolog, from an interpretation of a fragment of the logic (Horn clauses and goals). Therefore, it becomes realistic to think that a proof search procedure, as "efficient" in terms of resource manipulations as SLD-resolution for Horn clauses, can be devised for *full* Linear Logic, without any syntactical restriction. This paper attempts to devise such a procedure.

The basic procedure could be described as follows: given an initial sequent σ_o, it incrementally builds a proof Π of σ_o, in a Prolog-like fashion. Initially Π is assigned to a single (open) node labeled with σ_o, and the process SEARCH(σ_o) is started.

Procedure SEARCH(σ: open node of Π)

1. Select an instance of an inference figure of the sequent system with bottom sequent equal to σ and top sequents $\sigma_1, \ldots, \sigma_n$ (with $n \geq 0$);
2. Expand proof Π at node σ with n branches to new open nodes labeled respectively with $\sigma_1, \ldots, \sigma_n$;
3. For each $k = 1 \ldots n$, Start SEARCH(σ_k);

At step 3 of procedure SEARCH, the newly created open nodes $\{\sigma_k\}_{k=1\ldots n}$ can be searched simultaneously. This form of parallelism is called "global", since it concerns all the branches of the proof, as opposed to "local" parallelism, introduced below, and which occurs within a single branch of proof.

Step 1 of procedure SEARCH is not completely determined, in that the criterion for the selection of an inference figure is not specified. But, if we assume that all the possible choices at this step are explored (by a choice enumeration procedure for example) then all the possible proofs of the initial sequent are generated.

However, it appears that many of these generated proofs are redundant. Of course, syntactically speaking, all of them are different, since, by construction, they correspond to different sequences of application of the inference figures. But it may happen that the order in which these inference figures are applied turns out to be irrelevant,

or could be simplified, so that two (syntactically different) proofs built by the procedure SEARCH may in fact be equivalent in the following sense:

Definition 1 *Two proofs are said to be P-equivalent if each of them can be obtained from the other by simple permutations of inference figures and elimination or introduction of useless loops.*

This equivalence relation is denoted $\Longleftrightarrow$.

- Permutations of inferences in a proof are characterized by a situation of the following kind:

$$\dfrac{\dfrac{\dfrac{\vdots}{\sigma_1}\ \cdots\ \dfrac{\vdots}{\sigma_n}}{\sigma}[S]}{\sigma'}[R] \quad\Longleftrightarrow\quad \dfrac{\dfrac{\vdots}{\sigma_1}\ \cdots\ \dfrac{\vdots}{\sigma_n}}{\sigma'}[S]\ [R]$$

  Here, the sequent σ' is derived in two different ways from the same premisses $(\sigma_i)_{i=1\ldots n}$ simply by permuting the application of inference rules R and S (this is not possible with any two inferences rules R, S). From the point of view of the search process, this means that the inference rules R and S could in fact be selected (and applied) *simultaneously* at step 1 of procedure SEARCH, instead of sequentially, as required by the definition of this procedure. This provides a new form of parallelism, called "local" parallelism, which complements the "global" parallelism already mentioned above.

- Similarly, useless loops in a proof are characterized by a situation of the following kind:

$$\dfrac{\dfrac{\vdots}{\sigma}\quad\vdots}{\sigma} \quad\Longleftrightarrow\quad \dfrac{\vdots}{\sigma}$$

  where one sequent inside a proof is identical to the root of the proof (and hence, the sub-proof starting from that internal sequent could replace the overall proof).

Typical instances of these two cases of P-equivalence, which disturb the search procedure, are presented and analyzed in the next sections.

The solution adopted here to deal with the problem of redundant P-equivalent proofs is to modify the procedure SEARCH in such a way that, instead of trying to build all the possible proofs of a sequent, it generates only a subset of these proofs, from which all the others could be mechanically derived. The idea is that, within this subset, the number of distinct P-equivalent proofs should be null (ideally) or at least reduced. In other words, proofs from this subset can be viewed as "normal" representatives of the classes of P-equivalent proofs, and only these normal proofs are searched. I propose here a complete subset of proofs for Linear Logic, called the "focusing" proofs, which does not rely on any syntactic restriction (unlike, for instance, resolution, which applies only to clauses in Classical Logic).

2 Focusing Proofs

The class of focusing proof is defined below using an indirect method. It starts with the standard sequent system of Linear Logic, as can be found in [18] (see Fig. 1), and called, here, the "Monadic" system Σ_1 (this terminology is justified below). A first step of proof normalization is defined using an other sequent system, called the "Dyadic" system Σ_2, together with a (possibly non deterministic) transformation from monadic to dyadic proofs such that:

> If two monadic proofs can be mapped into the same dyadic proof, then they are P-equivalent.

Hence, each dyadic proof Π represents a subset of a P-equivalence class of monadic proofs, consisting of all the monadic proofs which can be mapped into Π. It constitutes a first approximation of this P-equivalence class. A better approximation is obtained during the second step of proof normalization, which is specified in exactly the same way, i.e. using another inference system, called the "Triadic" system Σ_3, and a mapping from dyadic to triadic proofs verifying: if two dyadic proofs can be mapped into the same triadic proof, then they are P-equivalent.

The first normalization step is concerned with the inference rules of Contraction and Weakening, whereas the second normalization step deals with all the other inference rules. It could have been possible to merge the two steps and go directly from the Monadic to the Triadic sequent system, but at the price of some clarity.

The procedure SEARCH becomes much more tractable when applied to the Triadic sequent system, since it avoids to generate all the redundant P-equivalent monadic proofs which correspond to the same triadic proof.

F, G stand for formulae, Γ, Δ stand for multisets of formulae.

- Identity [I] and Cut [C]

$$[I]\ \frac{}{\vdash F, F^{\perp}} \qquad [\mathbf{C}]\ \frac{\vdash \Gamma, F \quad \vdash \Delta, F^{\perp}}{\vdash \Gamma, \Delta}$$

- Weakening [<] and Contraction [>]

$$[<]\ \frac{\vdash \Gamma}{\vdash \Gamma, ?F} \qquad [>]\ \frac{\vdash \Gamma, ?F, ?F}{\vdash \Gamma, ?F}$$

- Logical inference rules

$$[\perp]\ \frac{\vdash \Gamma}{\vdash \Gamma, \perp} \qquad [\parr]\ \frac{\vdash \Gamma, F, G}{\vdash \Gamma, F \parr G} \qquad [?]\ \frac{\vdash \Gamma, F}{\vdash \Gamma, ?F}$$

$$[1]\ \frac{}{\vdash 1} \qquad [\otimes]\ \frac{\vdash \Gamma, F \quad \vdash \Delta, G}{\vdash \Gamma, \Delta, F \otimes G} \qquad [!]\ \frac{\vdash ?\Gamma, F}{\vdash ?\Gamma, !F}$$

$$[\top]\ \frac{}{\vdash \Gamma, \top} \qquad [\&]\ \frac{\vdash \Gamma, F \quad \vdash \Gamma, G}{\vdash \Gamma, F \,\&\, G} \qquad [\forall]\ \frac{\vdash \Gamma, F[c/x]}{\vdash \Gamma, \forall x\, F}$$

$$[\oplus_l]\ \frac{\vdash \Gamma, F}{\vdash \Gamma, F \oplus G} \qquad [\oplus_r]\ \frac{\vdash \Gamma, G}{\vdash \Gamma, F \oplus G} \qquad [\exists]\ \frac{\vdash \Gamma, F[t/x]}{\vdash \Gamma, \exists x\, F}$$

Figure 1: The Monadic Sequent System Σ_1

2.1 The Dyadic System Σ_2

In the Monadic sequent system Σ_1, a sequent consists of a single multiset (i.e. unordered list, denoted with Greek uppercase letters Γ, Δ) of formulae (i.e. resources). Linear Logic is characterized by the fact that the two inference figures of Contraction and Weakening cannot be applied freely to any formula in a sequent. This means that formulae are viewed as *restricted* (bounded) resources. However, some resources may need to be unrestricted, so that they can be used in a proof an unbounded number of times (including 0). This is achieved by explicitly prefixing these formulae with the modality ?, and the inference figures of Contraction [>] and Weakening [<] apply only to such modalized formulae, allowing unbounded duplication or deletion of the corresponding resource.

These two rules are immediate sources of P-equivalence. Indeed, any proof of a sequent $\Gamma, ?F$ (i.e. with at least one modalized formula), is P-equivalent to the proof simply obtained as follows:

$$[>]\ \frac{[<]\ \dfrac{\dfrac{\vdots}{\vdash \Gamma, ?F}}{\vdash \Gamma, ?F, ?F}}{\vdash \Gamma, ?F} \iff \frac{\vdots}{\vdash \Gamma, ?F}$$

This is a typical case of a useless loop: the two steps of Contraction and Weakening above just cancel each other. Similarly, Contraction and Weakening lead to permutations of inference figures, for instance:

$$[\parr]\ \frac{[<]\ \dfrac{\dfrac{\vdots}{\vdash \Gamma, G, H}}{\vdash \Gamma, ?F, G, H}}{\vdash \Gamma, ?F, G \parr H} \iff [<]\ \frac{[\parr]\ \dfrac{\dfrac{\vdots}{\vdash \Gamma, G, H}}{\vdash \Gamma, G \parr H}}{\vdash \Gamma, ?F, G \parr H}$$

The proof normalization proposed here to deal with these problems can be summarized informally as follows:

> Applications of Contraction and Weakening should be delayed as much as possible in the search procedure and applied only when needed.

Given that the search is performed from the root of the proof towards its leaves, this means that, in normal proofs, occurrences of Contraction and Weakening should be permuted as much as possible towards the leaves. This is

F, G stand for formulae, Θ, Γ, Δ stand for multisets of formulae.

- Identity [**I**] and Cut [**C**]

$$[I]\ \frac{}{\vdash \Theta : F, F^{\perp}} \qquad [\mathbf{C}]\ \frac{\vdash \Theta : \Gamma, F \quad \vdash \Theta : \Delta, F^{\perp}}{\vdash \Theta : \Gamma, \Delta}$$

- Absorption

$$[\mathbf{A}]\ \frac{\vdash \Theta, F : \Gamma, F}{\vdash \Theta, F : \Gamma}$$

- Logical inference rules

$$[\perp]\ \frac{\vdash \Theta : \Gamma}{\vdash \Theta : \Gamma, \perp} \qquad [\parr]\ \frac{\vdash \Theta : \Gamma, F, G}{\vdash \Theta : \Gamma, F \parr G} \qquad [?]\ \frac{\vdash \Theta, F : \Gamma}{\vdash \Theta : \Gamma, ?F}$$

$$[1]\ \frac{}{\vdash \Theta : 1} \qquad [\otimes]\ \frac{\vdash \Theta : \Gamma, F \quad \vdash \Theta : \Delta, G}{\vdash \Theta : \Gamma, \Delta, F \otimes G} \qquad [!]\ \frac{\vdash \Theta : F}{\vdash \Theta : !F}$$

$$[\top]\ \frac{}{\vdash \Theta : \Gamma, \top} \qquad [\&]\ \frac{\vdash \Theta : \Gamma, F \quad \vdash \Theta : \Gamma, G}{\vdash \Theta : \Gamma, F \,\&\, G} \qquad [\forall]\ \frac{\vdash \Theta : \Gamma, F[c/x]}{\vdash \Theta : \Gamma, \forall x\ F}$$

$$[\oplus_l]\ \frac{\vdash \Theta : \Gamma, F}{\vdash \Theta : \Gamma, F \oplus G} \qquad [\oplus_r]\ \frac{\vdash \Theta : \Gamma, G}{\vdash \Theta : \Gamma, F \oplus G} \qquad [\exists]\ \frac{\vdash \Theta : \Gamma, F[t/x]}{\vdash \Theta : \Gamma, \exists x\ F}$$

Figure 2: The Dyadic Sequent System Σ_2

always possible with Weakening; Contraction can also be permuted, except when it appears immediately before[1] an occurrence of the inference figure [$\otimes$], [**C**] or [?]. In terms of resource manipulation, this means that an unrestricted resource should not be touched until it is actually required in the proof, or when it can be discarded because a terminal node has been reached and it was never used.

These notions are formalized in the Dyadic system.

Definition 2 *A dyadic sequent is a pair of multisets of formulae.*

The dyadic sequent made up of the pair of multisets Θ and Γ is written $\Theta : \Gamma$. In fact, it stands for the monadic sequent $?\Theta, \Gamma$ obtained by prefixing all the formulae of the first field Θ with the modality ?. In other words, Θ represents a tank of unrestricted resources in which the proof search process can help itself at any time.

By definition, Contraction and Weakening could be applied freely on the formulae of Θ, but, in normal proofs, they are allowed only at the leaves of a proof (for Weakening) and immediately before the inference figure with which they do not permute (for Contraction). This is captured in the Dyadic sequent system Σ_2, given in Fig. 2, which uses dyadic sequents. The fundamental relation between the Monadic and the Dyadic systems is captured by the following theorem, proved in appendix A.1.

Theorem 1 *Let Θ and Γ be multisets of formulae. The sequent $\Theta : \Gamma$ is derivable in Σ_2 if and only if the (corresponding) sequent $?\Theta, \Gamma$ is derivable in Σ_1. Formally,*

$$\vdash \Theta : \Gamma \text{ if and only if } \vdash ?\Theta, \Gamma$$

The demonstration of this theorem is given in a constructive way, so that it would be possible (if not easy) to extract from it the specification of a mechanical (possibly non deterministic) transformation, mapping monadic into dyadic proofs. It could then be shown that this mapping satisfies the property: if two monadic proofs can be mapped into the same dyadic proof, they are P-equivalent (simply because the demonstration of the theorem relies on trivial permutations and simplifications of inferences).

A reverse transformation from dyadic to monadic proofs is defined in Fig. 3 using simple rewrite rules on proofs. It shows that, in fact, the inference figures of the Dyadic system are simply obtained from those of the Monadic system by adding an extra field to each sequent, except for those dealing with the modalities.

[1] Proofs are built bottom-up from the root to the leaves; an inference is "before" another if it is closer to the root on the same branch.

$$[I]\ \frac{}{\vdash \Theta : F, F^{\perp}} \quad \mapsto \quad [<^{\star}]\ \frac{[I]\ \dfrac{}{\vdash F, F^{\perp}}}{\vdash ?\Theta, F, F^{\perp}}$$

$$[\mathbf{C}]\ \frac{\vdash \Theta : \Gamma, F \quad \vdash \Theta : \Delta, F^{\perp}}{\vdash \Theta : \Gamma, \Delta} \quad \mapsto \quad [>^{\star}]\ \frac{[\mathbf{C}]\ \dfrac{\vdash ?\Theta, \Gamma, F \quad \vdash ?\Theta, \Delta, F^{\perp}}{\vdash ?\Theta, ?\Theta, \Gamma, \Delta}}{\vdash ?\Theta, \Gamma, \Delta}$$

$$[\mathbf{A}]\ \frac{\vdash \Theta, F : \Gamma, F}{\vdash \Theta, F : \Gamma} \quad \mapsto \quad [>]\ \frac{[?]\ \dfrac{\vdash ?\Theta, ?F, \Gamma, F}{\vdash ?\Theta, ?F, ?F, \Gamma}}{\vdash ?\Theta, ?F, \Gamma}$$

$$[\perp]\ \frac{\vdash \Theta : \Gamma}{\vdash \Theta : \Gamma, \perp} \quad \mapsto \quad [\perp]\ \frac{\vdash ?\Theta, \Gamma}{\vdash ?\Theta, \Gamma, \perp}$$

$$[\parr]\ \frac{\vdash \Theta : \Gamma, F, G}{\vdash \Theta : \Gamma, F \parr G} \quad \mapsto \quad [\parr]\ \frac{\vdash ?\Theta, \Gamma, F, G}{\vdash ?\Theta, \Gamma, F \parr G}$$

$$[?]\ \frac{\vdash \Theta, F : \Gamma}{\vdash \Theta : \Gamma, ?F} \quad \mapsto \quad \vdash ?\Theta, ?F, \Gamma$$

$$[1]\ \frac{}{\vdash \Theta : 1} \quad \mapsto \quad [<^{\star}]\ \frac{[1]\ \dfrac{}{\vdash 1}}{\vdash ?\Theta, 1}$$

$$[\otimes]\ \frac{\vdash \Theta : \Gamma, F \quad \vdash \Theta : \Delta, G}{\vdash \Theta : \Gamma, \Delta, F \otimes G} \quad \mapsto \quad [> \star]\ \frac{[\otimes]\ \dfrac{\vdash ?\Theta, \Gamma, F \quad \vdash ?\Theta, \Delta, G}{\vdash ?\Theta, ?\Theta, \Gamma, \Delta, F \otimes G}}{\vdash ?\Theta, \Gamma, \Delta, F \otimes G}$$

$$[!]\ \frac{\vdash \Theta : F}{\vdash \Theta : !F} \quad \mapsto \quad [!]\ \frac{\vdash ?\Theta, F}{\vdash ?\Theta, !F}$$

For all the other inference figures ($[\top], [\&], [\forall], [\oplus_l], [\oplus_r], [\exists]$), the mapping simply replaces each dyadic sequent $\Theta : \Gamma$ in the inference figure by its corresponding monadic sequent $?\Theta, \Gamma$.

Figure 3: Canonical injection $\Sigma_2 \mapsto \Sigma_1$

- As expected, Contraction and Weakening have disappeared in the Dyadic system (at least as explicit rules). Weakening is implicit in the terminal inference rules $[1], [\top], [\mathbf{I}]$ (with no premiss) of the Dyadic system, which have been obtained as combination of the corresponding rule of the Monadic system and occurrences of Weakening. Contraction is implicit in the inference rules $[\otimes], [\mathbf{A}], [\mathbf{C}]$ of the Dyadic system which correspond to those inferences of the Monadic system with which Contraction does not permute.
- Thus, the rule $[?]$ of the Monadic system (a.k.a. Dereliction), which allows to effectively use a modalized formula (unrestricted resource) by stripping off its modality, is implicitly combined with a Contraction in the structural rule of Absorption $[\mathbf{A}]$ of the Dyadic system.

  $$[\mathbf{A}]\ \frac{\vdash \Theta, F : \Gamma, F}{\vdash \Theta, F : \Gamma}$$

  This just means that when an unrestricted resource is to be used, it can systematically be duplicated beforehand, using Contraction, even when this is not strictly needed (if the resource is not to be reused): anyway, the possibly useless Contraction thus introduced will automatically be canceled by a Weakening when a leaf is reached.
- On the other hand, the rule $[?]$ of the Dyadic system

  $$[?]\ \frac{\vdash \Theta, F .: \Gamma}{\vdash \Theta : \Gamma, ?F}$$

  corresponds to no rule in the Monadic system: mapped to monadic sequents, the top and bottom sequents of this inference figure are exactly the same. It allows to dynamically "fill the tank" of unrestricted resources, but locally only, i.e. in the branch of proof where it is applied.

2.2 The Triadic System Σ_3

The Dyadic system deals with P-equivalent proofs of the Monadic system caused by unrestricted use of the inference figures related to the modalities. But all the other inference figures are also sources of redundant P-equivalent proofs.

2.2.1 Cut and Identity

One typical case of useless loops in proofs (and hence inefficiency in the search) comes from the Cut rule [C]. Indeed, consider the following proofs:

$$[\parr]\ \dfrac{[\mathbf{C}]\ \dfrac{\dfrac{\vdots}{\vdash \Gamma, F \parr G}\quad [\otimes]\ \dfrac{[I]\ \dfrac{}{\vdash F, F^\perp}\quad [I]\ \dfrac{}{\vdash G, G^\perp}}{\vdash F^\perp \otimes G^\perp, F, G}}{\vdash \Gamma, F, G}}{\vdash \Gamma, F \parr G} \iff \dfrac{\vdots}{\vdash \Gamma, F \parr G}$$

The application of the Logical inference rule [$\parr$] at the root is immediately cancelled by the Cut [C] which returns the root sequent on its left premiss (hence a loop); the same problem occurs with the inference rule [&]. This problem can be dealt with by an already well known proof normalization result, namely the Cut-elimination theorem, proved, for the Monadic system, in [18]. This theorem also holds in the Dyadic system.

Furthermore, we can assume without loss of generality that the Identity rule [I] is always applied on atomic formulae. Indeed, non atomic occurrences of the Identity can be reduced to atomic one, by repeatedly applying simple transformations such as

$$[I]\ \dfrac{}{\vdash F \parr G, F^\perp \otimes G^\perp} \quad \mapsto \quad [\parr]\ \dfrac{[\otimes]\ \dfrac{[I]\ \dfrac{}{\vdash F, F^\perp}\quad [I]\ \dfrac{}{\vdash G, G^\perp}}{\vdash F, G, F^\perp \otimes G^\perp}}{\vdash F \parr G, F^\perp \otimes G^\perp}$$

Finally, we make the following convention:

> In the sequel, we consider only proofs in the Dyadic system which are Cut-free and contain only atomic occurrences of the Identity.

2.2.2 The Problem of the Principal Formula

The Logical inference rules lead to multiple cases of permutations of inferences, as for instance

$$[\otimes]\ \dfrac{[\parr]\ \dfrac{\dfrac{\vdots}{\vdash \Gamma, F, H, K}}{\vdash \Gamma, F, \underline{H \parr K}}\quad \dfrac{\vdots}{\vdash \Delta, G}}{\vdash \Gamma, \Delta, \underline{F \otimes G}, H \parr K} \iff [\parr]\ \dfrac{[\otimes]\ \dfrac{\dfrac{\vdots}{\vdash \Gamma, F, H, K}\quad \dfrac{\vdots}{\vdash \Delta, G}}{\vdash \Gamma, \Delta, \underline{F \otimes G}, H, K}}{\vdash \Gamma, \Delta, F \otimes G, \underline{H \parr K}}$$

From the point of view of the procedure SEARCH, the problem could be stated as follows. When a Logical inference is to be applied at a given node, two choices must be made: *(i)* choice of a (non atomic) "principal formula" (underlined in the proofs above) in the sequent at that node; *(ii)* choice of an instance of the Logical inference figure associated with the topmost connective of the selected principal formula. Although all forms of "don't know" non-determinism cannot be eliminated in these choices (a search procedure is intrinsically non deterministic), the permutation of inferences above shows that some of these choices are not significant and either need not be considered at all or could be treated deterministically ("don't care" non-determinism).

The linear connectives can be partitioned into two groups which behave differently with respect to the choice of the principal formula.

- The "asynchronous" connectives:
  - Multiplicative: $\perp$, $\parr$, ?
  - Additive: $\top$, &, $\forall$
- The "synchronous" connectives:
  - Multiplicative: 1, $\otimes$, !
  - Additive: 0, $\oplus$, $\exists$

This terminology is not standard and will be justified below. Notice that the dual of an asynchronous connective is synchronous and vice versa. A non-atomic formula whose top-most connective is synchronous (resp. asynchronous) is called a synchronous (resp. asynchronous) formula. The difference in search behavior between these two groups can be characterized as follows.

> If the principal formula which has been selected in a sequent is asynchronous, then there is one and only one applicable instance of the corresponding inference figure, whereas if it is synchronous, one among several (or sometimes no) instances has to be selected.

Thus, if the synchronous formula $F \otimes G$ is selected as principal formula in the sequent $\Gamma, F \otimes G$, many possible instances of the corresponding inference figure $[\otimes]$ can be applied, corresponding to the different partitions of Γ along the two branches. Similarly, a principal formula of the form $F \oplus G$ requires the choice between the left ($[\oplus_l]$) and right ($[\oplus_r]$) instances of the corresponding inference figure. On the other hand, when an asynchronous formula is selected as principal formula, there is a unique applicable instance of the corresponding inference figure and its application is therefore deterministic.

We can summarize these properties as

- Asynchronous $\mapsto$ Determinism
- Synchronous $\mapsto$ Non-determinism

which replaces the usual Prolog characterization where determinism is accounted for by the conjunction and non-determinism by the disjunction; Linear Logic features a "non deterministic" conjunction $\otimes$ and a "deterministic" disjunction $\parr$.

Another, related, characterization of the difference between synchronous and asynchronous connectives concerns the reversibility of the inference figures: the conclusion of the logical inference figure associated with an asynchronous connective is derivable if and only if all its premisses are derivable. This property does not hold for synchronous connectives. In fact, in the Monadic system, it does not even hold for the asynchronous modality ?: Dereliction is not reversible. However, it is interesting to notice that this exception disappears in the Dyadic system, where the inference rule [?] is reversible. This is not surprising since, in the Dyadic system, rule [?] simply moves a formula prefixed with the modality inside the tank of unbounded resources, thus *enabling* future Dereliction on that formula (implicit in the structural rule of Absorption), but it does not actually *perform* a Dereliction (so that it remains reversible).

2.2.3 Triadic Sequents

Potential permutations of inferences such as those mentioned in the previous section would induce limited perturbation in a proof search procedure, were it not allowed to select the principal formula anywhere in the sequent. The proof normalization proposed here precisely imposes constraints on the way such selection is performed. It can be summarized informally as follows:

- If the sequent contains some *asynchronous* formulae (at least one), then any one of them can be *immediately* and *randomly* selected as the principal formula ("don't care" non-determinism). Furthermore, as the formula thus selected is by hypothesis asynchronous, the instance of inference figure to apply is completely determined. Consequently, as long as the sequent contains an asynchronous formula, the search can be made completely deterministic.
- When all the asynchronous formulae have been decomposed, then a principal formula must be selected non deterministically. But, as soon as one formula has been selected, the search can *focus* on it, i.e. subsequently select systematically as principal formula the subformula stemming from the initial one, *and do so as long as this subformula is synchronous.*

Asynchronous formulae are decomposed immediately as soon as they appear in the sequent (hence their name "asynchronous"). Synchronous formulae are delayed until all the asynchronous formulae have been decomposed, and must be non deterministically selected to be processed: in other words, synchronous connectives synchronize the selection process and the decomposition process (hence their name "synchronous"). But once a synchronous formula starts being decomposed, it keeps on being decomposed till a non synchronous (i.e. atomic or asynchronous) formula is reached. This means that in a normal proof, each formula is viewed as a succession of layers of asynchronous connectives and of synchronous connectives; each synchronous layer is decomposed in a *critical section* (i.e. which cannot be interrupted), called a "critical focusing section" of the proof. For example, with this respect, the following

F, G stand for formulae, X stands for a positive atom, Γ, Δ, Θ stand for multisets of formulae (Γ, Δ containing no asynchronous formula) and L stand for an ordered list of formulae.

- Logical inference rules

$$[\bot]\ \frac{\vdash \Theta : \Gamma \Uparrow L}{\vdash \Theta : \Gamma \Uparrow L, \bot} \qquad [\parr]\ \frac{\vdash \Theta : \Gamma \Uparrow L, F, G}{\vdash \Theta : \Gamma \Uparrow L, F \parr G} \qquad [?]\ \frac{\vdash \Theta, F : \Gamma \Uparrow L}{\vdash \Theta : \Gamma \Uparrow L, ?F}$$

$$[\top]\ \frac{}{\vdash \Theta : \Gamma \Uparrow L, \top} \qquad [\&]\ \frac{\vdash \Theta : \Gamma \Uparrow L, F \quad \vdash \Theta : \Gamma \Uparrow L, G}{\vdash \Theta : \Gamma \Uparrow L, F \,\&\, G} \qquad [\forall]\ \frac{\vdash \Theta : \Gamma \Uparrow L, F[c/x]}{\vdash \Theta : \Gamma \Uparrow L, \forall x\, F}$$

$$[1]\ \frac{}{\vdash \Theta :\Downarrow 1} \qquad [\otimes]\ \frac{\vdash \Theta : \Gamma \Downarrow F \quad \vdash \Theta : \Delta \Downarrow G}{\vdash \Theta : \Gamma, \Delta \Downarrow F \otimes G} \qquad [!]\ \frac{\vdash \Theta :\Uparrow F}{\vdash \Theta :\Downarrow\, !F}$$

$$[\oplus_l]\ \frac{\vdash \Theta : \Gamma \Downarrow F}{\vdash \Theta : \Gamma \Downarrow F \oplus G} \qquad [\oplus_r]\ \frac{\vdash \Theta : \Gamma \Downarrow G}{\vdash \Theta : \Gamma \Downarrow F \oplus G} \qquad [\exists]\ \frac{\vdash \Theta : \Gamma \Downarrow F[t/x]}{\vdash \Theta : \Gamma \Downarrow \exists x\, F}$$

- Reaction $\Uparrow$: if F is not asynchronous

$$[R\Uparrow]\ \frac{\vdash \Theta : \Gamma, F \Uparrow L}{\vdash \Theta : \Gamma \Uparrow L, F}$$

- Reaction $\Downarrow$: if F is neither synchronous nor a negative atom

$$[R\Downarrow]\ \frac{\vdash \Theta : \Gamma \Uparrow F}{\vdash \Theta : \Gamma \Downarrow F}$$

- Identities

$$[I_1]\ \frac{}{\vdash \Theta : X \Downarrow X^\perp} \qquad [I_2]\ \frac{}{\vdash \Theta, X :\Downarrow X^\perp}$$

- Decisions : If F is not a positive atom

$$[D_1]\ \frac{\vdash \Theta : \Gamma \Downarrow F}{\vdash \Theta : \Gamma, F \Uparrow} \qquad [D_2]\ \frac{\vdash \Theta, F : \Gamma \Downarrow F}{\vdash \Theta, F : \Gamma \Uparrow}$$

Figure 4: The Triadic Sequent System Σ_3

proof is not normal

$$[\otimes]\ \frac{\dfrac{\vdots}{\vdash A, F} \quad [\oplus]\ \dfrac{[\otimes]\ \dfrac{\dfrac{\vdots}{\vdash B, D, G} \quad \dfrac{\vdots}{\vdash C, H}}{\vdash \underline{B \otimes C}, D, G, H}}{\vdash B \otimes C, \underline{D \oplus E}, G, H}}{\vdash \underline{A \otimes (B \otimes C)}, D \oplus E, F, G, H}$$

since, after the formula $A \otimes (B \otimes C)$, is selected as principal formula at the root, its synchronous subformula $B \otimes C$ should be immediately selected as principal formula in the right branch, instead of the formula $D \oplus E$. To normalize the previous proof, the inferences should at least be permuted as follows, so as to obtain a critical focusing section on the right branch (assuming B is not synchronous, otherwise, again, it should be selected as principal formula, instead of $D \oplus E$):

$$[\otimes]\ \frac{\dfrac{\vdots}{\vdash A, F} \quad [\otimes]\ \dfrac{[\oplus]\ \dfrac{\dfrac{\vdots}{\vdash B, D, G}}{\vdash B, \underline{D \oplus E}, G} \quad \dfrac{\vdots}{\vdash C, H}}{\vdash \underline{B \otimes C}, D \oplus E, G, H}}{\vdash \underline{A \otimes (B \otimes C)}, D \oplus E, F, G, H}$$

Furthermore, if one of the formulae F, G, H is asynchronous, then even the latter proof is not normal, since in this case, one asynchronous formula among F, G, H should be immediately selected as principal formula at the root.

"Don't know" non-determinism appears in the search only during the critical focusing sections, which involve synchronous connectives (asynchronous connectives generate only "don't care" non-determinism). However, non-determinism can be considerably reduced by the following condition imposed on normal proofs. Let's partition arbitrarily the atomic formulae into two dual disjoint classes: positive atoms X and negative atoms $X^{\perp}$. In a normal proof, when a critical focusing section reaches a negative atom, then the inference figure of Identity [**I**] *must* be applied. This condition lies at the core of the language LinLog, and will be discussed in the next section.

These notions are formalized in the Triadic system.

Definition 3 *A triadic sequent is of one of the following two forms:*

- $\Theta : \Gamma \Uparrow L$ *where* L *is an* ordered *list of formulae;*
- $\Theta : \Gamma \Downarrow F$ *where* F *is a formula;*

and where Θ *and* Γ *are multisets of formulae,* Γ *containing no asynchronous formula.*

Thus, a triadic sequent is a dyadic sequent in which either an ordered list of formulae ($\Uparrow$-case) or one formula ($\Downarrow$-case) has been singled out in the second field. In fact, $\Theta : \Gamma \Uparrow L$ stands for the dyadic sequent $\Theta : \Gamma, L$ (in which the order of L is "forgotten") and $\Theta : \Gamma \Downarrow F$ stands for the dyadic sequent $\Theta : \Gamma, F$. The role of the arrows is explicited in the Triadic system Σ_3, given in Fig. 4, which uses triadic sequents:

- A sequent $\Theta : \Gamma \Uparrow L$ corresponds to the case where the sequent possibly contains an asynchronous formula (in L). The third field of the sequent L acts then as a stack. Each formula from L is popped and, if it is asynchronous, it is immediately decomposed, and its components are pushed back into the stack; otherwise it is simply added to the second field of the sequent Γ which, consequently, never contains any asynchronous formula. Thus, $\Uparrow$-sequents handle the layers of asynchronous connectives, which involve only "don't care" non-determinism treated deterministically.
- A sequent $\Theta : \Gamma \Downarrow F$ corresponds to the case where all the asynchronous formulae have been decomposed and a formula F has been selected as principal formula. The subformulae of F are then systematically selected as principal formulae (since they are put back in the third field of the sequent) till a non synchronous formula is reached. Thus, $\Downarrow$-sequents handle the layers of synchronous connectives which are processed during the critical focusing sections. Real "don't know" non-determinism occurs only in such sections.

The fundamental relation between the Dyadic and the Triadic systems is captured by the following theorem, proved in appendix A.2.

Theorem 2 *Let* Θ *and* Γ *be multisets of formulae,* Γ *containing no asynchronous formulae, and let* L *be an ordered list of formulae. The sequent* $\Theta : \Gamma \Uparrow L$ *is derivable in* Σ_3 *if and only if the (corresponding) sequent* $\Theta : \Gamma, L$ *is derivable in* Σ_2. *Formally,*

$$\vdash \Theta : \Gamma \Uparrow L \text{ if and only if } \vdash \Theta : \Gamma, L$$

The demonstration of this theorem is given in a constructive way, so that it is theoretically possible to extract from it the specification of a possibly non-deterministic transformation, mapping dyadic proofs into triadic proofs and satisfying the property: if two dyadic proofs can be mapped into the same triadic proof, they are P-equivalent.

A reverse transformation from triadic to dyadic proofs is defined in Fig. 5 using simple rewrite rules on proofs. It shows that, in fact, the *Logical* inference figures of the Triadic system are simply obtained from those of the Dyadic system by splitting the sequents with an arrow ($\Uparrow$ for inference figures corresponding to asynchronous connectives and $\Downarrow$ for the synchronous case). But the Triadic system also contains some specific structural rules which become trivial when mapped into the Dyadic system and which handle the initialization and termination of critical focusing sections.

- The "Reaction" rule [$R \Uparrow$] is triggered from a sequent $\Theta : \Gamma \Uparrow L, F$ when the last formula F is not asynchronous. In this case, F is just added to the second field of the sequent, Γ, for future use, when all the remaining asynchronous formulae of L will have been decomposed. Had F been asynchronous, then it would have immediately been decomposed using the Logical rule corresponding to its topmost connective.
- When all the asynchronous formulae have been decomposed in an $\Uparrow$-sequent (i.e. L is empty) a principal formula must be non deterministically selected, starting a new critical focusing section (of $\Downarrow$-sequents). This is the role of the "Decision" rules. The principal formula may be picked either inside the second field of the sequent, containing the non-asynchronous formulae which have been delayed by the Reaction rule [$R \Uparrow$] above (Decision [D_1]), or in the first field, i.e. the "reserve tank" of unrestricted formulae (Decision [D_2]) where it is not discarded. The search then proceeds with a critical section focusing on the selected formula.

$$[R\Uparrow]\ \frac{\vdash \Theta : \Gamma, F \Uparrow L}{\vdash \Theta : \Gamma \Uparrow L, F} \quad \mapsto \quad \vdash \Theta : \Gamma, L, F$$

$$[R\Downarrow]\ \frac{\vdash \Theta : \Gamma \Uparrow F}{\vdash \Theta : \Gamma \Downarrow F} \quad \mapsto \quad \vdash \Theta : \Gamma, F$$

$$[I_1]\ \frac{}{\vdash \Theta : X \Downarrow X^\perp} \quad \mapsto \quad [I]\ \frac{}{\vdash \Theta : X, X^\perp}$$

$$[I_2]\ \frac{}{\vdash \Theta, X : \Downarrow X^\perp} \quad \mapsto \quad [\mathbf{A}]\ \frac{[I]\ \dfrac{}{\vdash \Theta, X : X, X^\perp}}{\vdash \Theta, X : X^\perp}$$

$$[D_1]\ \frac{\vdash \Theta : \Gamma \Downarrow F}{\vdash \Theta : \Gamma, F \Uparrow} \quad \mapsto \quad \vdash \Theta, \Gamma, F$$

$$[D_2]\ \frac{\vdash \Theta, F : \Gamma \Downarrow F}{\vdash \Theta, F : \Gamma \Uparrow} \quad \mapsto \quad [\mathbf{A}]\ \frac{\vdash \Theta, F : \Gamma, F}{\vdash \Theta, F : \Gamma}$$

For all the other (Logical) inference figures ($[\bot], [\parr], [?], [\top], [\&], [\forall], [1], [\otimes], [!], [\oplus_l], [\oplus_r], [\exists]$) the mapping simply replaces each triadic sequent (resp. $\Theta : \Gamma \Uparrow L$ or $\Theta : \Gamma \Downarrow F$) in the inference figure by its corresponding dyadic sequent (resp. $\Theta : \Gamma, L$ or $\Theta : \Gamma, F$).

Figure 5: Canonical injection $\Sigma_3 \mapsto \Sigma_2$

- The "Reaction" rule $[R\Downarrow]$ is triggered when an asynchronous formula, or a *positive* atom, is reached at the end of a critical focusing section. The arrow is just turned upside down, which means that the critical section is terminated and the asynchronous formulae must be decomposed.
- When a *negative* atom $X^\perp$ is reached at the end of a critical focusing section, the Identity must be used, so that X must be found in the rest of the sequent, either as a restricted resource in the second field (Identity $[I_1]$), or as an unrestricted resource in the first field (Identity $[I_2]$). This considerably reduces the amount of non-determinism involved in the critical sections.

Finally, the modalities have a peculiar behavior in the triadic system:

- When a formula prefixed with the asynchronous modality ? is encountered, it is immediately stored in the first field of the sequent for possible future (unbounded) use, instead of being put back in the third field of the sequent for further decomposition of asynchronous connectives, as in the standard asynchronous case.
- The modality !, unlike standard synchronous connectives, terminates a critical focusing section (see rule $[!]$), but it enforces that at the moment of the interruption, the second field of the sequent is empty (i.e. the sequent contains only unbounded resources).

2.3 Summary

In this section, we have introduced the two sequent systems Σ_2, Σ_3 together with:

1. An injection from Σ_2-proofs into Σ_1-proofs (Fig. 3) which is a canonical morphism, in that it works at the inference level (the image of a combination of two inferences is a combination of the images of these inferences).
2. A projection from Σ_1-proofs into Σ_2-proofs, which is a mechanical, non-deterministic, transformation (specifiable from the demonstration of theorem 1) and is a reverse of the previous canonical injection. It is defined in terms of some more or less complex rearrangement of the inferences in the proofs (permutations of inferences and eliminations or introductions of useless loops), so that if two Σ_1-proofs are mapped into the same Σ_2-proof, then they are necessarily P-equivalent.
3. A similar canonical injection from Σ_3-proofs into Σ_2-proofs (Fig. 5).
4. A similar reverse projection from Σ_2-proofs into Σ_3-proofs (obtained from the demonstration of theorem 2).

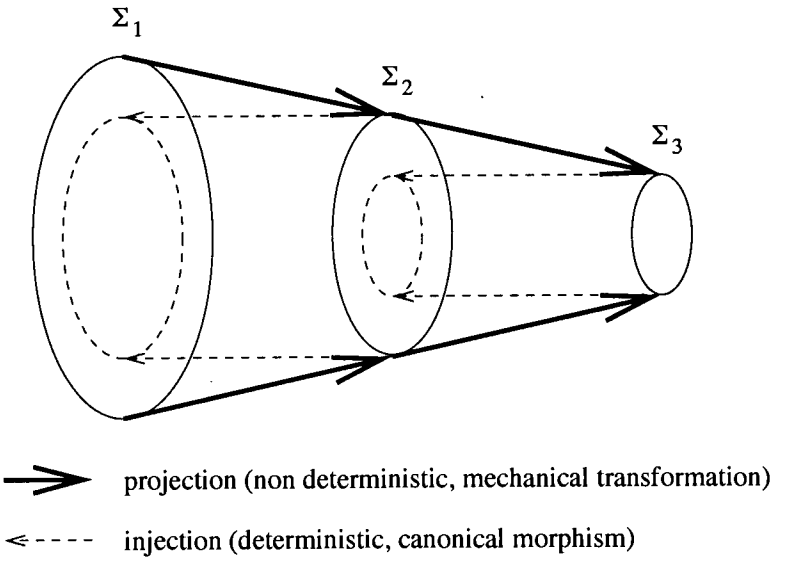

Figure 6: The sequent systems and their relations.

This situation is summarized in Fig. 6. The term "focusing" proof denotes, depending on the context, either any triadic proof (in Σ_3), or the canonical image of a triadic proof in Σ_2, or the canonical image of such a dyadic proof in Σ_1 (in other words, the canonical injections are treated as identities).

The notion of focusing proofs is characterized by the following properties.

- Focusing proofs form a complete subset of proofs for Linear Logic, i.e. each derivable formula in this logic has a focusing proof. This is a direct consequence of theorems 1 and 2.
- Focusing proofs respect the overall symmetry of Linear Logic, in that dual connectives have dual focusing properties. This duality is best visualized in the structure of the sequents of the Triadic system (resp. $\Uparrow$ and $\Downarrow$), which handle the dual groups of linear connectives (resp. called here asynchronous and synchronous).

Given that many P-equivalent proofs become identical when mapped into focusing proofs, the procedure SEARCH defined in the introduction becomes much more tractable when constrained to aim at focusing proofs. Furthermore, given the completeness of focusing proofs, all the other proofs which could have been generated by the procedure SEARCH if it had not been thus constrained, could be obtained from the generated focusing proofs by application of the reverse of the projections from Σ_1 to Σ_3.

However, notice that not all P-equivalent proofs become identical when mapped to focusing proofs. For example, the following (monadic) proofs are distinct focusing proofs, although they are P-equivalent.

$$\dfrac{\dfrac{\vdots}{\vdash a^\perp, a} \quad \dfrac{\dfrac{\vdots}{\vdash b^\perp, b} \quad \vdash \underline{c^\perp \otimes d^\perp}, c, d}{\vdash c^\perp \otimes d^\perp, c, \underline{b^\perp \otimes d}, b}}{\vdash c^\perp \otimes d^\perp, \underline{a^\perp \otimes c}, b^\perp \otimes d, a, b} \quad \Longleftrightarrow \quad \dfrac{\dfrac{\vdots}{\vdash b^\perp, b} \quad \dfrac{\dfrac{\vdots}{\vdash a^\perp, a} \quad \vdash \underline{c^\perp \otimes d^\perp}, d, c}{\vdash c^\perp \otimes d^\perp, d, \underline{a^\perp \otimes c}, a}}{\vdash c^\perp \otimes d^\perp, a^\perp \otimes c, \underline{b^\perp \otimes d}, a, b}$$

They just differ in the order of application of the inference rule [$\otimes$]. The cause of this problem can be identified in the Triadic system: when one of the Decision rules is used, to select a principal formula to focus on, this choice is completely free and is not influenced by previous steps in the proof. In fact, it may happen (as in the proofs above) that the order in which these choices are made is irrelevant and could be permuted. The Triadic system is not able, in its current stage, to handle this case of permutation of inferences.

3 The Logic Programming Language LinLog

Focusing proofs have a very simple and computationally significant interpretation, which appears clearly when they are applied to a fragment of Linear Logic, called LinLog, and presented below. Furthermore, the syntactic restrictions defining this fragment do not induce any restriction on the expressiveness of the language. Indeed,

it is shown that any formula of Linear Logic can be mapped into LinLog. This mapping can be viewed as a normalization procedure, analogous to the transformation to clausal form for Classical Logic. The main difference is that the latter transformation preserves only provability whereas the former preserves also the structure of the (focusing) proofs.

3.1 LinLog Syntax and Operational Semantics

3.1.1 Methods and Goals

LinLog is based on two classes of formulae, called the "methods" and the "goals". Goals have a two-layer structure:

- Elementary goals are combinations of positive atoms connected with asynchronous connectives, and where the modality ? applies only to atoms. The class g of elementary goals can thus be specified by

$$g = X \mid ?X \mid \bot \mid g \parr g \mid \top \mid g \,\&\, g \mid \forall x\, g$$

- Goals are combinations of elementary goals connected with synchronous connectives, and where the modality ! applies only to elementary goals. The class G of goals can thus be specified by

$$G = g \mid !g \mid 1 \mid G \otimes G \mid 0 \mid G \oplus G \mid \exists x\, G$$

In other words, a goal is a positive formula (containing no negative atom) in which a synchronous connective may never occur in the scope of an asynchronous one, and no connective may occur in the scope of the modality of the same class (asynchronous or synchronous). Methods are ground formulae of the form

$$\forall \vec{x}\,(G \multimap X_1 \parr \cdots \parr X_r)$$

where $X_1, \ldots, X_r$ are positive atoms ($r \geq 1$) called the *head* of the method, and G is a goal called the *body* of the method. $\multimap$ is the Linear implication defined as $A \multimap B =_{\text{def}} A^\perp \parr B$. As with Horn clauses, the universal quantifier $\forall$ in front of methods is often omitted (assuming an implicit typographical convention for variable names) and the implication is written in reverse notation:

$$X_1 \parr \cdots \parr X_r \circ\!\!- G$$

Definition 4 *A* LinLog *query is a pair* $\langle \mathcal{P}; g \rangle$ *where* $\mathcal{P}$ *is a set of methods (called the program) and* g *is an elementary goal.*

3.1.2 Triadic Sequents in LinLog

Let $\langle \mathcal{P}; g \rangle$ be a LinLog query. A LinLog execution consists of searching proofs of the sequent $!\mathcal{P} \vdash g$, i.e., using one sided sequents (Monadic system), $\vdash ?\mathcal{P}^\perp, g$. Using the Triadic system and theorems 1 and 2, this can be achieved by applying procedure SEARCH to the sequent $\vdash \mathcal{P}^\perp :\Uparrow g$. In other words, the program $\mathcal{P}$ (or, more precisely, its dual) acts as a set of unrestricted resources which provides the executing process with an everlasting source of computing energy, and the goal g acts as the initial state of the process in terms of restricted resources.

The syntax of goals and methods has been designed in such a way that triadic proofs of LinLog queries have a characteristic structure: they are repeatedly composed of

1. one layer of Logical inference rules mixed with the Reaction rules $[R \Uparrow]$ and $[R \Downarrow]$, and containing only sequents of the form

$$\vdash \mathcal{P}^\perp, \Psi : \Phi \Downarrow G \text{ or } \vdash \mathcal{P}^\perp, \Psi : \Phi \Uparrow \mathcal{G}$$

   where Ψ and Φ are multisets of positive atoms, G is a goal and $\mathcal{G}$ an ordered list of elementary goals;

2. one layer of inference rules starting with a sequent of the form

$$\vdash \mathcal{P}^\perp, \Psi : \Phi \Uparrow$$

   where Ψ and Φ are multisets of positive atoms.

In fact, the second layer, which is the only one in which the methods of the program play a role, can be given a compact representation consisting of a single inference step, called a Progression step, described below.

Definition 5 *Let* M *be the method* $X_1 \parr \cdots \parr X_r \circ\!\!- G$. *An instance of* M *is a pair written* $\Phi_o \circ\!\!- G_o$ *where* Φ_o *is a multiset of ground positive atoms and* G_o *is a ground goal, such that there exists a ground substitution* σ *verifying*

$$\sigma.\{X_1, \ldots, X_r\} = \Phi_o \text{ and } \sigma.G = G_o$$

Let $\mathcal{P}$ be a LinLog program. $\Psi, \Psi_o, \Phi, \Phi_1, \Phi_2$ stand for multisets of positive atoms, g, g_1, g_2 stand for elementary goals, $\mathcal{G}$ stands for an ordered list of elementary goals and G, G_1, G_2 stand for goals. X stands for a positive atom.

- Progression

$$[\circ\!\!-] \frac{\vdash \Psi, \Psi_o : \Phi \Downarrow G_o}{\vdash \Psi, \Psi_o : \Phi, \Phi_o \Uparrow}$$

  where $\Psi_o, \Phi_o \circ\!\!- G_o$ is an instance of a method in $\mathcal{P}$

- Logical inference rules

$$[\bot] \frac{\vdash \Psi : \Phi \Uparrow \mathcal{G}}{\vdash \Psi : \Phi \Uparrow \mathcal{G}, \bot} \qquad [\parr] \frac{\vdash \Psi : \Phi \Uparrow \mathcal{G}, g_1, g_2}{\vdash \Psi : \Phi \Uparrow \mathcal{G}, g_1 \parr g_2} \qquad [?] \frac{\vdash \Psi, X : \Phi \Uparrow \mathcal{G}}{\vdash \Psi : \Phi \Uparrow \mathcal{G}, ?X}$$

$$[\top] \frac{}{\vdash \Psi : \Phi \Uparrow \mathcal{G}, \top} \qquad [\&] \frac{\vdash \Psi : \Phi \Uparrow \mathcal{G}, g_1 \quad \vdash \Psi : \Phi \Uparrow \mathcal{G}, g_2}{\vdash \Psi : \Phi \Uparrow \mathcal{G}, g_1 \,\&\, g_2} \qquad [\forall] \frac{\vdash \Psi : \Phi \Uparrow \mathcal{G}, g[c/x]}{\vdash \Psi : \Phi \Uparrow \mathcal{G}, \forall x\, g}$$

$$[1] \frac{}{\vdash \Psi :\Downarrow 1} \qquad [\otimes] \frac{\vdash \Psi : \Phi_1 \Downarrow G_1 \quad \vdash \Psi : \Phi_2 \Downarrow G_2}{\vdash \Psi : \Phi_1, \Phi_2 \Downarrow G_1 \otimes G_2} \qquad [!] \frac{\vdash \Psi :\Uparrow g}{\vdash \Psi :\Downarrow !g}$$

$$[\oplus_l] \frac{\vdash \Psi : \Phi \Downarrow G_1}{\vdash \Psi : \Phi \Downarrow G_1 \oplus G_2} \qquad [\oplus_r] \frac{\vdash \Psi : \Phi \Downarrow G_2}{\vdash \Psi : \Phi \Downarrow G_1 \oplus G_2} \qquad [\exists] \frac{\vdash \Psi : \Phi \Downarrow G[t/x]}{\vdash \Psi : \Phi \Downarrow \exists x\, G}$$

- Reactions

$$[R\Uparrow] \frac{\vdash \Psi : \Phi, X \Uparrow L}{\vdash \Psi : \Phi \Uparrow L, X} \qquad [R\Downarrow] \frac{\vdash \Psi : \Phi \Uparrow g}{\vdash \Psi : \Phi \Downarrow g}$$

Figure 7: The Sequent System of LinLog $\Sigma_3[\mathcal{P}]$

The Progression inference rule is defined by

$$[\circ\!\!-] \frac{\vdash \mathcal{P}^\perp, \Psi, \Psi_o : \Phi \Downarrow G_o}{\vdash \mathcal{P}^\perp, \Psi, \Psi_o : \Phi, \Phi_o \Uparrow}$$

where $\Psi_o, \Phi_o \circ\!\!- G_o$ is an instance of a method in $\mathcal{P}$.

Therefore, it is possible to specialize the inference system Σ_3 into an inference system $\Sigma_3[\mathcal{P}]$, given in Fig. 7, where the single inference rule of Progression $[\circ\!\!-]$ above replaces those of Decision $[D_1], [D_2]$ and Identity $[I_1], [I_2]$ of the general system. In $\Sigma_3[\mathcal{P}]$-proofs, the occurrences of $\mathcal{P}^\perp$, which appear in the first field of all the sequents, are omitted (they are implicit).

$\Sigma_3[\mathcal{P}]$ is the inference system of the programming language LinLog, and is justified by the following theorem, which derives quite straightforwardly from theorems 1 and 2 (see appendix A.3 for a demonstration).

Theorem 3 *Let $\langle \mathcal{P}; g \rangle$ be a* LinLog *query. The sequent $!\mathcal{P} \vdash g$ is derivable in Linear Logic if and only if the sequent $:\Uparrow g$ is derivable in $\Sigma_3[\mathcal{P}]$. Formally,*

$$!\mathcal{P} \vdash g \text{ if and only if } \vdash_{\mathcal{P}} :\Uparrow g$$

Given a triadic sequent σ in LinLog, $\vdash_{\mathcal{P}} \sigma$ is taken to mean that σ is derivable in $\Sigma_3[\mathcal{P}]$.

3.1.3 Computational Interpretation

The inference system of LinLog has a very natural computational interpretation. It manipulates sequents which, when mapped to monadic sequents, are of the form $\vdash ?\mathcal{P}^\perp, \mathcal{C}$, where $\mathcal{C}$, called the context, is a multiset of ground goals containing at most one non elementary goal [2].

Definition 6 *A context is said to be flat if it contains only flat goals, i.e. positive atoms, possibly prefixed with the modality ?.*

The inference mechanism of LinLog can then be informally characterized in two clauses:

[2] More precisely, it can be shown that if $\mathcal{C}$ does contain a non elementary goal, then the rest of the context is flat

- If the context is not flat, then select at random a non flat goal in the context and decompose it using the inference rule corresponding to its topmost connective. The selection step involves "don't care" non-determinism. The decomposition step may involve "don't know" non-determinism if the selected goal is not elementary (hence synchronous).
- If the context is flat, then find an instance of a method in the program such that its head matches exactly a submultiset of the atoms (modalized or not) of the context. Then replace in the context those (and only those) atoms of this submultiset which are not modalized, by the body of the method. Then proceed with the context thus updated. The selection of the method, as well as that of the submultiset of the context to match its head, involve "don't know" non-determinism.

From that point of view, standard Prolog appears as the degenerated case of LinLog in which the contexts are singletons and the heads of the methods contain themselves only one atom; in this case, the distinction between asynchronous and synchronous connectives in the goals becomes meaningless (as well as the modalities).

The computational model of LinLog is especially suited for concurrency. First, of course, the different branches of proof can be searched concurrently ("global" parallelism). Furthermore, on one branch, several strategies can be devised for the selection of the method to apply at each step of the computation. Once the set of candidate methods (and matching subcontext) has been determined, one of them can be picked at random and applied (committed choice strategy); an other solution is to apply *simultaneously* all the candidate methods whose heads match *disjoint* submultisets of the context ("local" parallelism); intermediate solutions are also possible. Once the method(s) is (are) applied, the goals introduced in the context can, in turn, be decomposed in parallel. The determination of the strategy can be made at compile time, so as to make optimal use of the available parallel processing facilities.

LinLog supports various inter-process communication mechanisms. Communication is viewed here as exchange of resources between processes. Goals connected by the multiplicative connectives $\parr$ and $\otimes$ yield processes which compete for resources from their context, whereas the additive connectives & and $\oplus$ support resource sharing (in which each process gets a separate copy of the resources). On the other hand, the asynchronous connectives $\parr$ and & support deterministic and immediate communication whereas synchronous connectives $\otimes$ and $\oplus$ lead to non deterministic, possibly deferred, communication.

These different computational behaviors are directly significant in the framework of concurrent object-oriented systems, especially in the actors tradition. With this perspective in mind, the subset of LinLog called LO (for Linear Objects), in which goals are built only from the asynchronous connectives $\parr$ and &, has been studied in [3, 7, 4, 6, 8], where computational examples can be found. For instance, dynamic programming techniques find a very natural concurrent implementation in LO, as shown in [6, 8]. LO has also been applied in [7] to the optimization of Horn clause programs which have an exponential complexity when executed by the standard Prolog strategy. Using program transformation techniques, they can be converted into more tractable LO programs, switching from backward to forward chaining strategies. Various toy applications of full LinLog (including all the connectives) have been devised, but a clear characterization of the class of applications thus covered has yet to be stated.

3.2 Normalization to LinLog Form

Let F be a formula. The procedure SEARCH could be used to build the focusing proofs of F. However, its specialized LinLog version, which has been tailored for computational efficiency, cannot be used as such if F does not satisfy the syntax of a LinLog query. It is shown here that this syntactical restriction is not bought at the price of generality. Indeed, this section presents a mechanical transformation, analogous to the transformation to clausal form for Classical Logic, and which maps any formula F into a LinLog query such that the focusing proofs of F are isomorphic to the LinLog proofs of the corresponding query, up to some irrelevant name conventions.

3.2.1 An Example

Let $F = (a^\perp \otimes b) \parr a \parr b^\perp$. Formula F violates the syntax of LinLog elementary goals since the synchronous subformula $F_1 = a^\perp \otimes b$ and the negative atom $F_2 = b^\perp$ occur in the scope of an asynchronous connective. However, were F_1 and F_2 positive atoms, F would be an elementary goal, and the first steps of its LinLog proof would be the same as the first steps of the focusing proof of F (there is only one focusing proof here), i.e. a sequence of simple applications of the inference rules $[\parr]$ and $[R \Uparrow]$:

$$\dfrac{\dfrac{\vdots}{\vdash : F_1, a, F_2 \Uparrow}}{\dfrac{\vdots}{\vdash : \Uparrow F_1 \parr a \parr F_2}}$$

In the focusing proof of F, the next step consists of the selection of F_1 as principal formula, using Decision rule $[D_1]$, followed by a (short) critical focusing section:

$$[D_1]\ \dfrac{[\otimes]\ \dfrac{[I_1]\ \overline{\vdash: a \Downarrow a^\perp} \quad \dfrac{\vdots}{\vdash: F_2 \Downarrow b}}{\vdash: a, F_2 \Downarrow F_1}}{\vdash: F_1, a, F_2 \Uparrow}$$

We now want to map this critical section into a LinLog Progression step $[\circ\!\!-]$. To achieve this, let's replace F_1 by a *new* positive atom, say u_1, and let's try to view the formula $u_1 \circ\!\!- F_1$ as a method M_1. Thus, the selection of formula F_1 by the decision rule $[D_1]$ above could as well be seen as the triggering of method M_1 by u_1. Here, we simply have $M_1 = u_1 \parr a \circ\!\!- b$ and the critical section above can be mapped into the LinLog single Progression step using method M_1 (doing as if F_2 were a positive atom):

$$[\circ\!\!-]\ \dfrac{\dfrac{\vdots}{\vdash: F_2 \Downarrow b}}{\vdash: u_1, a, F_2 \Uparrow}$$

Then, the focusing proof as well as the LinLog proof proceed in the same way:

$$[R\Downarrow]\ \dfrac{[R\Uparrow]\ \dfrac{\dfrac{\vdots}{\vdash: F_2, b \Uparrow}}{\vdash: F_2 \Uparrow b}}{\vdash: F_2 \Downarrow b}$$

At this point, the focusing proof of F proceeds in selecting formula F_2 using decision rule $[D_1]$ and starting a new critical focusing section (which completes the proof):

$$[D_1]\ \dfrac{[I_1]\ \overline{\vdash: b \Downarrow F_2}}{\vdash: F_2, b \Uparrow}$$

There again, we can map this critical section into a LinLog Progression step $[\circ\!\!-]$: let's introduce a *new* positive atom u_2 and let's try to view $u_2 \circ\!\!- F_2$ as a method M_2. Here $M_2 = u_2 \parr b \circ\!\!- 1$ and the critical section above is mapped into a LinLog Progression step using method M_2:

$$[\circ\!\!-]\ \dfrac{[1]\ \overline{\vdash: \Downarrow 1}}{\vdash: u_2, b \Uparrow}$$

Finally, let g be the formula obtained by replacing in F its subformula F_1 by u_1 and its subformula F_2 by u_2. As intended, the formula $g = u_1 \parr a \parr u_2$ is now a LinLog elementary goal and we have mapped the focusing proof of $:\Uparrow F$ (in Σ_3) into the following LinLog proof of $:\Uparrow g$ (in $\Sigma_3[\mathcal{P}]$) where $\mathcal{P}$ is the LinLog program containing the methods M_1 and M_2.

$$[\ldots]\ \dfrac{[\circ\!\!-]\ \dfrac{[\ldots]\ \dfrac{[\ldots]\ \overline{\vdash: \Downarrow 1}}{\vdash: u_2, b \Uparrow}}{\vdash: u_2 \Downarrow b}}{\vdash: u_1, a, u_2 \Uparrow} \qquad \dfrac{}{\vdash: \Uparrow u_1 \parr a \parr u_2}$$

The fact that atoms u_1, u_2 are "new" (i.e. do not occur anywhere else in F when they are introduced) ensures that, conversely, any LinLog proof of $:\Uparrow g$ in $\Sigma_3[\mathcal{P}]$ can be mapped back into a focusing proof of $:\Uparrow F$. Indeed, it ensures that the methods of $\mathcal{P}$ can only be used in the situations above, so that the proof of $:\Uparrow F$ can be reconstructed. In fact, the atoms u_1, u_2 are identifiers of the syntactic occurrences of, respectively, subformulae F_1, F_2 in F, and, by definition, syntactic occurrences are unique.

The operation of "naming" F_1 by u_1 (or F_2 by u_2) is strictly similar to the Skolemization step in the classical clausal transformation; only, Skolemization occurs at the term level whereas here, naming occurs at the subformula level.

3.2.2 The Algorithm

We present the LinLog normalization algorithm here in the propositional case (no variables nor quantification), but it can easily be extended to the first-order case (see below). Let F_o be a formula. The computation of its LinLog normal form, which is a LinLog query, can be informally described as follows:

1. Try to view F_o as an elementary goal, i.e. determine the deepest subformulae in F_o which preclude it from being an elementary goal in LinLog syntax.
2. Let g be the formula obtained by replacing in F_o each of the subformulae computed at step 1 by a *fresh* positive atom (a different one for each subformula). By construction, g is an elementary goal, which defines the goal component of the normal form.
3. For each atom u introduced in F_o to replace a subformula F, try to view (as in step 1) the formula $u \circ\!\!- F$ as a set of LinLog methods (connected by &). This may lead to new replacements of subformulae by new atoms, which are recursively treated in the same way and which may produce more methods.
4. The set of all the methods produced at step 3 forms the program component of the normal form.

The formula $u \circ\!\!- F$ can be interpreted as the "definition" of the atom u by the formula F (which u replaces in F_o). The algorithm attempts to translate this definition into a LinLog program, possibly recursively introducing new definitions. Notice that such definitions are unrestricted resources with global scope: they are just naming conventions which can be reused as many times as needed anywhere. This is why they are mapped into the program component of the query.

To specify the algorithm formally, we introduce "quasi-methods", which are like methods, except that they may have an empty head, and their body is a multiset of goals[3] (implicitly connected by $\otimes$).

Definition 7 *A quasi-method is a pair* $[\Phi;\Upsilon]$ *where* Φ *is a multiset of positive atoms and* Υ *a multiset of goals. A quasi-method is said to be strict if* Φ *is non-empty.*

The LinLog normalization algorithm is given by the function NORMALIZE in Fig. 8. This algorithm uses three side functions, NGOAL, NATOM and NMETH, which try to view the formula passed in their argument as, respectively, an elementary goal, a positive atom and the dual of a set of quasi-methods (in the last case, the function returns this set, whose elements are taken to be implicitly connected by &).

- The function NGOAL is trivial: it recursively scans its argument formula F till it reaches the occurrences where, for F to be an elementary goal, the subformulae at these occurrences have to be positive atoms. These subformulae are then replaced by positive atoms obtained by applying the function NATOM to each of them.
- The function NATOM is also elementary: if its argument F is already a positive atom, it simply returns it; otherwise, it introduces a new Skolem constant X, which it returns, and maps the definition of X by F, i.e. the formula $X \circ\!\!- F$, into a set of methods, which are added to $\mathcal{P}$. In fact, to avoid multiple dualization inside F, it is easier to try to map the dual formula $X^{\perp} \otimes F$ into the dual of a set of methods (obtained by calling the function NMETH).
- The definition of the function NMETH must be understood as follows: any critical section of proof focusing on F can be mapped into a LinLog Progression step (extended trivially to deal with non necessarily strict quasi-methods) using one of the quasi-methods of NMETH(F). Thus, for example, if F is of the form $F_1 \oplus F_2$, a critical section focusing on F necessarily goes on with a critical section focusing either on F_1 or on F_2. Therefore, if we assume that, in both cases $i = 1, 2$, the critical section focusing on F_i can be mapped into a Progression step using a quasi-method in NMETH(F_i), then the critical section focusing on F can be mapped into a Progression step using a quasi-method in NMETH(F_1) $\cup$ NMETH(F_2). This justifies the definition of NMETH(F) in that case. The treatment of the connective $\otimes$ is a bit more complex and makes use of the operator $\star$ on sets of quasi-methods, defined as follows. Let $\mathcal{S}_1, \mathcal{S}_2$ be sets of quasi-methods; $\mathcal{S}_1 \star \mathcal{S}_2$ denotes the set of quasi-methods defined by pairwise combining the elements from each set:

$$\mathcal{S}_1 \star \mathcal{S}_2 = \bigcup_{[\Phi_1;\Upsilon_1] \in \mathcal{S}_1 \text{ and } [\Phi_2;\Upsilon_2] \in \mathcal{S}_2} \{[\Phi_1, \Phi_2 \,;\, \Upsilon_1, \Upsilon_2]\}$$

  Combining two quasi-methods yields a quasi-method obtained by respectively merging their heads and their bodies.

[3] In fact, elementary goals, possibly prefixed with the modality !, are sufficient.

Global $\mathcal{P}$: LinLog program;

Function NORMALIZE(F: formula) **returns** a LinLog query

$\mathcal{P} := \emptyset$;
$g :=$ NGOAL(F);
Return $\langle \mathcal{P}; g \rangle$;

Function NGOAL(F: formula) **returns** an elementary goal

Select

Case $F =$	Return
$F_1 \parr F_2$	NGOAL(F_1) $\parr$ NGOAL(F_2)
F_1 & F_2	NGOAL(F_1) & NGOAL(F_2)
$\bot$	$\bot$
$\top$	$\top$
$?F'$	$?$NATOM(F')
Otherwise	NATOM(F)

Function NATOM(F: formula) **returns** a positive atom

If F is a positive atom **Then**
  Return F;
Else
  Let X be a new positive atom (Skolem constant);
  $\mathcal{P} := \mathcal{P} \cup \|$NMETH($X^\bot \otimes F$)$\|$;
  Return X;

Function NMETH(F: formula) **returns** a set of quasi-methods

If F is a negative atom $X^\bot$ **Then**
  Return $\{[X; \emptyset]\}$;
Else
  Select

Case $F =$	Return
$F_1 \otimes F_2$	NMETH(F_1) $\star$ NMETH(F_2)
$F_1 \oplus F_2$	NMETH(F_1) $\cup$ NMETH(F_2)
1	$\{[\emptyset; \emptyset]\}$
0	$\emptyset$
$!F'$	$\{[\emptyset; !$NGOAL(F')$]\}$
Otherwise	$\{[\emptyset;$NGOAL(F)$]\}$

Figure 8: The LinLog normalization algorithm

Function NATOM may add, when called, new elements (methods) to $\mathcal{P}$ (which works as an accumulator). These methods are obtained using the operator $\|-\|$ defined as follows: let $\mathcal{S}$ be a set of *strict* quasi-methods; $\|\mathcal{S}\|$ denotes the set of corresponding methods, trivially defined as

$$\|\mathcal{S}\| = \bigcup_{[X_1,\ldots,X_r\ ;\ G_1,\ldots,G_s]\ \in\ \mathcal{S}} \{X_1 \parr \cdots \parr X_r \multimap G_1 \otimes \cdots \otimes G_s\}$$

When $s = 0$ in this definition, the body of the method is reduced to the logical constant 1. We have to make sure that when this operator is used in the definition of the function NATOM, its argument $\mathcal{S} = \text{NMETH}(X^\perp \otimes F)$ contains only *strict* quasi-methods (otherwise the definition above does not make sense). This directly results from a simple application of the definition of NMETH in this case:

$$\mathcal{S} = \{[X, \Phi\ ;\ \Upsilon] \text{ such that } [\Phi\ ;\ \Upsilon] \in \text{NMETH}(F)\}$$

Hence, the head of each quasi-method in $\mathcal{S}$ contains at least X (i.e. the defined atom) and is not empty.

The normalization algorithm is justified by the following result, proved in appendix A.4.

Theorem 4 *Let F be a formula and $\langle \mathcal{P}; g\rangle$ be its* LinLog *normal form (computed by the function* NORMALIZE*). F is derivable if and only if g is derivable in* LinLog *using program $\mathcal{P}$.*

$$\vdash F \text{ if and only if } \vdash_{\mathcal{P}} \Uparrow g$$

The demonstration of this theorem shows that, furthermore, not only is provability preserved in the normalization, but also the structure of the focusing proofs.

The algorithm can easily be adapted to the non propositional case. Only, when evaluating $\text{NATOM}(F)$ in the case where F is not already a positive atom, instead of simply introducing a new constant, we introduce a new functor f and return the positive atom $f(\vec{x})$ where $\vec{x}$ is the set of free variables in F. This is strictly analogous to the way Skolemization works, at the term level, in the clausal transformation. Furthermore, we have

$$\text{NGOAL}(\forall x\ F) = \forall x\ \text{NGOAL}(F)$$
$$\text{NMETH}(\exists x\ F) = \text{NMETH}(F)$$

3.3 Focusing and Cut reduction

Focusing has been described here in a Cut-free system. However, it can easily be extended to a system with Cut. Indeed, consider a proof Π of a sequent $\Theta_o : \Gamma_o$ (in the Dyadic system), possibly containing Cuts. Let Θ_c be the set of all the formulae of the form $H \otimes H^\perp$ such that H is a Cut formula in Π. By appending Θ_c to the first field of all the sequents in Π, we obtain a proof of $\Theta_o, \Theta_c : \Gamma_o$. Now, each occurrence of a Cut in this proof can be replaced as follows:

$$[\mathbf{C}]\ \frac{\vdash \Theta : \Gamma, H \quad \vdash \Theta : \Delta, H^\perp}{\vdash \Theta : \Gamma, \Delta} \quad \mapsto \quad [\mathbf{A}]\ \frac{[\otimes]\ \dfrac{\vdash \Theta : \Gamma, H \quad \vdash \Theta : \Delta, H^\perp}{\vdash \Theta : \Gamma, \Delta, H \otimes H^\perp}}{\vdash \Theta : \Gamma, \Delta}$$

The occurrence of the Absorption rule [**A**] here is justified, since the formula $H \otimes H^\perp$ which is absorbed is, by construction, in Θ_c and hence in Θ. Thus, we obtain a Cut-free proof of $\Theta_o, \Theta_c : \Gamma_o$, which can in turn be focused. We can now go back to the original proof by removing Θ_c from the first field of each sequent. This is always possible except in steps where a formula in Θ_c is selected by the Decision rule $[D_2]$ for decomposition. Given that the formulae of Θ_c are synchronous, and that decision rules start critical focusing sections, such steps are necessarily of the form

$$[D_2]\ \frac{[\otimes]\ \dfrac{\vdash \Theta : \Gamma \Downarrow H \quad \vdash \Theta : \Delta \Downarrow H^\perp}{\vdash \Theta : \Gamma, \Delta \Downarrow H \otimes H^\perp}}{\vdash \Theta : \Gamma, \Delta \Uparrow}$$

They can be replaced by introducing the following Cut rule for the Triadic system:

$$[\mathbf{C}]\ \frac{\vdash \Theta : \Gamma \Downarrow H \quad \vdash \Theta : \Delta \Downarrow H^\perp}{\vdash \Theta : \Gamma, \Delta \Uparrow}$$

In other words, in the Triadic system, the Cut-rule starts a critical section focusing on the Cut-formula in each of its premisses. This is one aspect of the analogy between the Cut rule and the Logical rule for the connective $\otimes$ mentioned in [18]. Now, from the discussion above, we can straightforwardly generalize the main result of this paper, namely that any proof (possibly containing Cuts) can be mapped into a focusing proof (possibly containing focused Cuts).

Naturally, the Cut elimination theorem holds in the Triadic system, since it already holds in the Dyadic system. Basically, the direct demonstration of this result, which is given in appendix A.5, is adapted from the classical one (see [16] or [20], for example). It mainly consists of a sequence of interleaved permutation steps and reduction steps on Cut formulae. In the usual case, the reduction step is possible only when the Cut formula is selected as principal formula in both premisses of a Cut, and the permutation steps are meant to obtain this condition. In the case of a focused proof, this condition is already ensured, but it might be violated once a reduction is performed. In fact, it can be shown that, given that the Cut formula is necessarily synchronous in exactly one of the premisses (except in the trivial case where it is atomic), the reduction of its whole topmost synchronous layer can be performed "in one step" in that premiss (following the critical focusing section). Matching permutations must be performed in the other premiss, but when the critical section is terminated in the first premiss, then it is the Cut formula in the other premiss which becomes synchronous, and can in turn be reduced.

But there is another interesting property relating Cut reduction and Focusing. Indeed, the crucial part of Focusing, captured by the Progression rule [o—] of the inference system of LinLog, can be expressed in terms of a Cut reduction property. Consider a method $M = X_1 \parr \cdots \parr X_r \multimap G$ in LinLog. A Progression step using this method can be mapped into a Cut using a non logical axiom equivalent to M, namely $G^\perp, X_1, \ldots, X_r$:

$$[\multimap]\ \frac{\vdots}{\vdash \mathcal{C}, G}{\vdash \mathcal{C}, X_1, \ldots, X_r} \quad \mapsto \quad [\mathrm{C}]\ \frac{\vdash G^\perp, X_1, \ldots, X_r \qquad \dfrac{\vdots}{\vdash \mathcal{C}, G}}{\vdash \mathcal{C}, X_1, \ldots, X_r}$$

Thus, we obtain a new justification of the Progression rule[4] of LinLog through the following result, which is a variant of Cut reduction:

> Any proof using (non logical) axioms of the form $X_1, \ldots, X_r, G^\perp$ (where $X_1, \ldots, X_r$ are positive atoms and G is a goal) can be transformed into a proof where each occurrence of the Cut satisfies the following property: one of the premiss is a non logical axiom and the cut formula is necessarily $G^\perp$.

This result is shown in [5] (in the framework of Classical logic, but it can be straightforwardly adapted to Linear logic), as a special case of a more general property: when Cut reduction is performed in a proof containing non logical axioms, all the Cuts cannot be eliminated, but they can be reduced so as to appear in "stacks" initiated by a non-logical axiom and where each Cut formula in a stack comes from its initial non logical axiom (and is not a positive atom); it can be shown that there is a direct correspondence between such stacks of Cuts and critical focusing sections.

4 Conclusion and Related Works

It has been mentioned above that Prolog is a fragment of LinLog. Other attempts have been made to extend Prolog using sequent systems. For instance, [26] introduces the notion of "uniform" proofs in Intuitionistic Logic (extended to the system of Intuitionistic Linear Logic in [21]). The computational efficiency of uniform proofs basically stems from a property similar to that of LinLog proofs (in the search, a method is applied only after all the non flat goals in the current context have been decomposed). Uniform proofs are defined for an implicational fragment of Intuitionistic Logic, known as Hereditary Harrop Logic[5], whereas it has been shown here that the notion of LinLog proofs is a degenerated but fully representative case of the notion of focusing proofs, which apply to full Linear Logic. Furthermore, using the translation from Intuitionistic to Linear logic described in [18], uniform proofs are precisely mapped into focusing proofs.

But the "definite clauses" of [26] (which correspond to the methods here) are characterized, as in Prolog, by a single atomic head; this feature simplifies considerably the mechanism of clause selection, especially in the framework of a sequential computation (although the use of lambda-terms instead of first-order terms, and the fact that clauses may be dynamically loaded in the proof, make things more complex than in the classical case). On the other hand, in a parallel environment, it has been shown in [3, 4, 6] and also in [27, 14, 11], that the formalism of multi-headed formulae (e.g. methods here) is better suited, especially for synchronization purposes. Contextual Horn clauses [27], or the logical objects of [14], or Shared Prolog [11], basically correspond to the fragment of LinLog

[4] In fact, further refinements would be needed to account for the fact that Progression can only be triggered when the context $\mathcal{C}$ is flat, and also for the fact that the head of a method may match atoms prefixed with the modality ?, which are not removed by the Progression step.

[5] In fact, this fragment allows all the connectives, but in such a way that it can directly be mapped into the pure implicative fragment of Intuitionistic Logic, without modifying substantially the shape of the uniform proofs. Thus, conjunction $\wedge$ can be mapped into stacked implications; similarly, a definite clause containing a disjunction $\vee$ in its body can be mapped into two separate clauses connected by conjunction. The mapping would not apply, were disjunction allowed to appear in the head of a clause; but this case is precisely forbidden.

where only the connective ⅋ is used in goals (but of course, each of these systems also have specific features which are not accounted for in LinLog). It leads to an "actors" model of computation [2, 32] where multiple independent agents perform concurrent tasks, communicating via a shared "blackboard" [28], or tuple space [17, 24]. In [6], we propose a more refined notion of blackboard (called "forum") which exploits not only the connective ⅋, but also &, in goals. It provides a notion of locality for the operations of consumption and production of messages in the blackboard.

Interaction nets [22] also provide a model for parallel computation. Although the theoretical justification of the system is proof reduction (Cut elimination) instead of proof search, the resulting computational model is remarkably related to that of LinLog. This may be a consequence of the similarity between focusing and Cut reduction mentioned in the previous section. In fact, the computational mechanism of Interaction nets is, again, that of LinLog where only the connective ⅋ is used. But Interaction nets are furthermore equipped with a strong type discipline, which prevents some forms of deadlocks, and ensures strong normalization (a typical requirement in functional programming). LinLog on the other hand was designed in the perspective of possibly unsafe distributed environments, where such requirements are not realistic.

Proof nets [18] offer a desequentialized representation of proofs. They are therefore directly relevant to the problem, addressed here, of eliminating in a search redundant P-equivalent proofs, which precisely differ only by their sequentialization. However, I could not devise a simple algorithm, such as procedure SEARCH given in the introduction, which could incrementally generate correct proof nets (satisfying their validity criterion: no short path), other than obtaining them from sequent proofs (but then their advantage is lost). Therefore, I preferred to stick to the sequent system approach, where sequents have a direct intuitive computational interpretation as process states, although the use of proof nets would have made the demonstration of the theorems much shorter.

The Triadic system described here features sequents split into three fields in which the formulae have different behaviors, respectively, classical, linear and non-commutative linear, corresponding to various levels of restriction on the use of the structural rules of Contraction, Weakening and Exchange. This triadic structure, which has been introduced here as an operational tool for specifying the class of focusing proofs, can also be used to provide a unified framework for various logics, as done in [19] with Classical, Intuitionistic and Linear logics.

Acknowledgement

The material presented here is contained in a thesis of the university of Paris 6 by the author [3], which was prepared at the European Computer Industry Research Center (ECRC) in Munich. I am indebted to the members of the jury for their helpful comments and suggestions. I also thank G. Comyn and A. Herold for their encouragement and support at ECRC. Special thanks are due to R. Pareschi for the many crucial discussions which helped shape the main ideas contained in this paper.

References

[1] S. Abramsky. Computational interpretations of linear logic. *Theoretical Computer Science*, 111(1–2):3–57, 1993.

[2] G. Agha and C. Hewitt. Actors: a conceptual foundation for concurrent object-oriented programming. In B. Shriver and P. Wegner, editors, *Research Directions in Object-Oriented Programming*. MIT Press, Cambridge, Ma, U.S.A., 1987.

[3] J-M. Andreoli. Proposition pour une synthèse des paradigmes de la programmation logique et de la programmation par objets, 1990. Thèse d'Informatique de l'Université de Paris 6.

[4] J-M. Andreoli and R. Pareschi. LO and behold! concurrent structured processes. In *Proc. of OOPSLA/ECOOP'90*, Ottawa, Canada, 1990.

[5] J-M. Andreoli and R. Pareschi. Logic programming with sequent systems: a linear logic approach. In *Proc. of the Workshop on Extensions of Logic Programming*, Lecture Notes in Artificial Intelligence 475, (Springer Verlag), Tübingen, Germany, 1990.

[6] J-M. Andreoli and R. Pareschi. Communication as fair distribution of knowledge. In *Proc. of OOPSLA'91*, Phoenix, Az, U.S.A., 1991.

[7] J-M. Andreoli and R. Pareschi. Linear objects: Logical processes with built-in inheritance. *New Generation Computing*, 9(3+4):445–473, 1991.

[8] J-M. Andreoli, R. Pareschi, and M. Bourgois. Dynamic programming as multi-agent programming. In *Proc. of the OOPSLA'90/ECOOP'91 workshop on Object-based concurrent computing*, Lecture Notes in Computer Science 612 (Springer Verlag), Genève, Switzerland, 1991.

[9] J-P. Banâtre, A. Coutant, and D. Le Metayer. A parallel machine for multiset transformation and its programming style. *Future Generation Computer Systems*, 4(2):133–145, 1988.

[10] G. Berry and G. Boudol. The chemical abstract machine. In *Proc. of the 17th ACM Symposium on Principles of Programming Languages*, San Francisco, Ca, U.S.A., 1990.

[11] A. Brogi and P. Ciancarini. The concurrent language shared prolog. *ACM Transactions on Programming Languages and Systems*, 13(1):99–123, 1991.

[12] C. Brown. Relating petri nets to formulae of linear logic. Technical report, University of Edinburgh, Edinburgh, U.K., 1989.

[13] K. Clark and S. Gregory. Parlog: Parallel programming in logic. *ACM Transactions on Programming Languages and Systems*, 8(1):1–50, 1986.

[14] J.S. Conery. Logical objects. In *Proc. of the 5th International Conference on Logic Programming*, Seattle, Wa, U.S.A., 1988.

[15] T. Conlon. *Programming in Parlog.* Addison-Wesley, Reading, Ma, U.S.A., 1989.

[16] J. Gallier. *Logic for Computer Science.* Harper & Row, New-York, NY, U.S.A., 1986.

[17] D. Gelernter. Generative communication in Linda. *ACM Transactions on Programming Languages and Systems*, 7(1):80–112, 1985.

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

[19] J-Y. Girard. On the unity of logic, 1991. Preprint, Université de Paris 7.

[20] J-Y. Girard, Y. Lafont, and P. Taylor. *Proofs and Types.* Cambridge University Press, 1989.

[21] J.S. Hodas and D. Miller. Logic programming in a fragment of intuitionistic linear logic. *Journal of Information and Computation*, 110(2):327–365, 1994.

[22] Y. Lafont. Interaction nets. In *Proc. of 17th ACM Symposium on Principles of Programming Languages*, San Francisco, Ca, U.S.A., 1990.

[23] N. Marti-Olliet and J. Meseguer. From petri-nets to linear logic. *Mathematical Structures in Computer Sience*, 1:69–101, 1991. Also SRI Technical Report.

[24] A. Matsuoka and S. Kawai. Using tuple space communication in distributed object oriented languages. In *Proc. of OOPSLA'88*, San Diego, Ca, U.S.A., 1988.

[25] J. Meseguer. Conditional rewriting logic as a unified model of concurrency. *Theoretical Computer Science*, 93:73–155, 1992.

[26] D. Miller, G. Nadathur, F. Pfenning, and A. Scedrov. Uniform proofs as a foundation for logic programming. *Annals of Pure and Applied Logic*, 51:125–157, 1991.

[27] L. Monteiro and A. Porto. Contextual logic programming. In *Proc. of the 6th International Logic Programming Conference*, Lisboa, Portugal, 1989.

[28] H. Penny Nii. Blackboard systems. In A. Barr, Cohen P., and Feigenbaum E., editors, *The Handbook of Artificial Intelligence, vol. 4*, pages 1–82. Addison-Wesley, Reading, Ma, U.S.A., 1989.

[29] E. Shapiro. The family of concurrent logic programming languages. Technical report, The Weizmann Institute of Science, Rehovot, Israel, 1989.

[30] E. Shapiro and A. Takeuchi. Object oriented programming in concurrent prolog. *New Generation Computing*, 1(1):25–48, 1983.

[31] K. Ueda. *Guarded Horn Clauses.* PhD thesis, Dept of Information Engineering, University of Tokyo, Tokyo, Japan, 1986.

[32] A. Yonezawa, E. Shibayama, Y. Honda, T. Takada, and J-P. Briot. An object-oriented concurrent computation model ABCM/1 and its description language ABCL/1. In A. Yonezawa, editor, *ABCL, an Object-Oriented Concurrent System*, pages 13–45. MIT Press, Cambridge, Ma, U.S.A., 1990.

A Demonstrations of the Theorems

The demonstrations are generally based on inductive reasoning, using the depth of sequent proofs as the induction counter. If Π is the proof

$$\Pi \;=\; \frac{\overbrace{\bigvee}^{\Pi_1} \quad \cdots \quad \overbrace{\bigvee}^{\Pi_n}}{\sigma_1 \qquad\quad \sigma_n \qquad\quad \sigma}$$

then the depth of Π, written $\mu(\Pi)$, is defined inductively by

$$\mu(\Pi) = 1 + \max\{\mu(\Pi_1), \ldots, \mu(\Pi_n)\}$$

The number of premisses n can be null, in that case the depth is simply 1. The induction steps of the demonstrations generally consist of a case-by-case analysis of the last inference step of a proof (the one which occurs at the root). In the sequel, we generally mention the induction hypothesis and only a few of these cases (the most significant).

Furthermore, we make the following convention: when we simply say that a sequent is derivable, or has a proof of a certain form, it is implicitly assumed to be in the sequent system (Monadic, Dyadic or Triadic) to which the sequent belongs.

A.1 Projection $\Sigma_1 \mapsto \Sigma_2$

We have to show the result stated in theorem 1, i.e.

$$\text{if } \vdash ?\Theta, \Gamma \text{ then } \vdash \Theta : \Gamma \tag{1}$$

(the reverse implication is immediate from Fig. 3). We make use of the following lemma.

Lemma 5 *If $\Theta \sqsubseteq \Theta'$ and $\Theta : \Gamma$ has a proof then $\Theta' : \Gamma$ has a proof of same depth.*

where $\Theta \sqsubseteq \Theta'$ means that all the elements of Θ are also elements of Θ' (possibly with a different number of occurrences). This lemma results immediately from the following remark: in a proof of $\Theta : \Gamma$, the multiset Θ appears in the first field of all the sequents. A proof of $\Theta' : \Gamma$ can therefore be obtained by simply replacing everywhere Θ by Θ'. We now come to the demonstration of the property (1).

Demonstration: Let $\mathcal{R}(n)$ be the following induction hypothesis

If $?\Theta, \Gamma$ has a proof of depth at most n then $\Theta : \Gamma$ is derivable.

Let's assume $\mathcal{R}(n)$ and let's show $\mathcal{R}(n+1)$. Let Π be a proof of depth $n+1$ of $?\Theta, \Gamma$.

1. If the last step of Π is an instance of the Logical inference rule [⅋], then the principal formula at this step is necessarily in Γ (since the formulae in $?\Theta$ are prefixed with the modality $?$) and we have

$$\Pi \;=\; [⅋]\,\frac{\overbrace{\bigvee}^{\Pi'}}{\vdash ?\Theta, \Gamma', F, G \;/\; \vdash ?\Theta, \Gamma', F ⅋ G}$$

   where $\Gamma = \Gamma', F ⅋ G$. The sub-proof Π' is of depth at most n. By the induction hypothesis $\mathcal{R}(n)$, we obtain that $\Theta : \Gamma', F, G$ is derivable, and hence, so is $\Theta : \Gamma$ by application of [⅋].

   The inference figure [⊥], [1], [⊤], [&], [∀], [⊕], [∃], [**I**] are treated in the same way.

2. If the last step of Π is an instance of the Logical inference rule [⊗] then, as above, the principal formula is necessarily in Γ and we have

$$\Pi \;=\; [\otimes]\,\frac{\overbrace{\bigvee}^{\Pi_a} \qquad \overbrace{\bigvee}^{\Pi_b}}{\vdash ?\Theta_a, \Gamma_a, F \quad \vdash ?\Theta_b, \Gamma_b, F \;/\; \vdash ?\Theta_a, ?\Theta_b, \Gamma_a, \Gamma_b, F \otimes G}$$

   where $\Theta = \Theta_a, \Theta_b$ and $\Gamma = \Gamma_a, \Gamma_b, F \otimes G$. By the induction hypothesis applied to Π_a (resp. Π_b), we obtain that $\Theta_a : \Gamma_a, F$ and $\Theta_b : \Gamma_b, G$ are derivable. By hypothesis, $\Theta_a \sqsubseteq \Theta$ and $\Theta_b \sqsubseteq \Theta$. By lemma 5 we obtain that $\Theta : \Gamma_a, F$ and $\Theta : \Gamma_b, G$ are derivable, and hence, so is $\Theta : \Gamma$ by application of [⊗].

   The Cut inference figure [**C**] is treated in the same way.

3. If the last step of Π is a Dereliction [?], two cases must be considered

   (a) If the derelicted formula is in $?\Theta$, we have

$$\Pi \;=\; [?]\,\frac{\overbrace{\bigvee}^{\Pi'}}{\vdash ?\Theta', F, \Gamma \;/\; \vdash ?\Theta', ?F, \Gamma}$$

where $\Theta = \Theta', F$. By the induction hypothesis applied to Π', we obtain that $\Theta' : \Gamma, F$ is derivable. By lemma 5 we obtain that $\Theta', F : \Gamma, F$ is derivable, and hence, so is $\Theta : \Gamma$ by

$$[\mathbf{A}]\ \frac{\dfrac{\vdots}{\vdash \Theta', F : \Gamma, F}}{\vdash \Theta', F : \Gamma}$$

(b) If the derelicted formula is in Γ, we have

$$\Pi \ = \ [?]\ \frac{\overset{\Pi'}{\bigvee}\ \vdash ?\Theta, \Gamma', F}{\vdash ?\Theta, \Gamma', ?F}$$

where $\Gamma = \Gamma', ?F$. By the induction hypothesis applied to Π', we obtain that $\Theta : \Gamma', F$ is derivable. By lemma 5, we obtain that $\Theta, F : \Gamma', F$ is derivable, and hence, so is $\Theta : \Gamma$ by

$$[?]\ \frac{[\mathbf{A}]\ \dfrac{\dfrac{\vdots}{\vdash \Theta, F : \Gamma', F}}{\vdash \Theta, F : \Gamma'}}{\vdash \Theta : \Gamma', ?F}$$

The inference figures $[!], [>], [<]$ are treated in a similar way.

Therefore $\mathcal{R}(n+1)$ holds. By induction $\mathcal{R}(n)$ holds for all n. □

A.2 Projection $\Sigma_2 \mapsto \Sigma_3$

We have to show the result stated in theorem 2, i.e.

$$\text{if } \vdash \Theta : \Gamma, L \text{ then } \vdash \Theta : \Gamma \Uparrow L \tag{2}$$

where Γ contains no asynchronous formula (the reverse implication is immediate from Fig. 5).

A.2.1 The main property

First, let's prove that property (2) is equivalent to

$$\text{if } \vdash \Theta : L \text{ then } \vdash \Theta :\Uparrow L \tag{3}$$

Demonstration:

- Clearly, property (3) derives from property (2) by substituting Γ with the empty multiset.
- Conversely, assume property (3) holds, and let $\Theta : \Gamma, L$ be derivable; let L' be any ordered list of the elements of Γ. By hypothesis, $\vdash \Theta : L, L'$ and, by property (3), we obtain that $\vdash \Theta :\Uparrow L, L'$. Given that L' (as Γ) contains no asynchronous formula, the last steps of a proof of $\Theta :\Uparrow L, L'$ can only be a sequence of applications of the Reaction rule $[R\Uparrow]$.

$$[R\Uparrow]\ \frac{\dfrac{\vdots}{\vdash \Theta : \Gamma \Uparrow L}}{\vdash \Theta :\Uparrow L, L'}$$

Therefore, $\Theta : \Gamma \Uparrow L$ is derivable.

Hence, property (2) and (3) are equivalent. □

Now, to show that property (3) holds, we need the following "inversion" lemmas, which will be shown in the next section.

Lemma 6 *Let Θ, Γ, Δ be multisets of formulae (Γ and Δ containing no asynchronous formula), let L, M be ordered lists of formulae and let F, G be formulae.*

$\mathcal{L}\otimes$: *If* $\vdash \Theta : \Gamma \Uparrow F, L$ *and* $\vdash \Theta : \Delta \Uparrow G, M$ *then* $\vdash \Theta : \Gamma, \Delta, F \otimes G \Uparrow L, M$

$\mathcal{L}\oplus$: *If* $\vdash \Theta : \Gamma \Uparrow F, L$ *then* $\vdash \Theta : \Gamma, F \oplus G \Uparrow L$

$\mathcal{L}\exists$: *If* $\vdash \Theta : \Gamma \Uparrow F[t/x], L$ *then* $\vdash \Theta : \Gamma, \exists x\ F \Uparrow L$

$\mathcal{L}\mathbf{A}$: *If* $\vdash \Theta, F : \Gamma \Uparrow F, L$ *then* $\vdash \Theta, F : \Gamma \Uparrow L$

$\mathcal{L}\equiv$: *If* $\vdash \Theta : \Gamma \Uparrow L$ *and* $L \equiv M$ *then* $\vdash \Theta : \Gamma \Uparrow M$

where $L \equiv M$ means that lists L and M differ only by the order of their elements. We now come to the demonstration of the property (3).

Demonstration: Let $\mathcal{R}(n)$ be the following induction hypothesis

If $\Theta : L$ has a proof of depth at most n then $\Theta :\Uparrow L$ is derivable.

Let's assume $\mathcal{R}(n)$ and let's show $\mathcal{R}(n+1)$. Let Π be a proof of depth $n+1$ of $\Theta : L$.

1. If the last step of Π is an instance of the Logical inference rule $[\parr]$, we have

$$\Pi \;=\; [\parr]\,\frac{\overset{\pi'}{\vdots}\;\vdash \Theta : L', F, G}{\vdash \Theta : L', F \parr G}$$

   where $L \equiv L', F \parr G$. The sub-proof Π' is of depth at most n. By the induction hypothesis $\mathcal{R}(n)$, we obtain that $\Theta :\Uparrow L', F, G$ is derivable, and hence, so is $\Theta :\Uparrow L', F \parr G$ by application of $[\parr]$. Given that $L \equiv L', F \parr G$, by property $\mathcal{L} \equiv$ of lemma 6, we obtain that $\Theta :\Uparrow L$ is derivable.

   The inference figures $[\bot], [?], [\top], [\&], [\forall]$ are treated in the same way.

2. If the last step of Π is an instance of the Logical inference rule $[\otimes]$, we have

$$\Pi \;=\; [\otimes]\,\frac{\overset{\pi_a}{\vdots}\;\vdash \Theta : L_a, F \quad \overset{\pi_b}{\vdots}\;\vdash \Theta : L_b, G}{\vdash \Theta : L_a, L_b, F \otimes G}$$

   where $L \equiv L_a, L_b, F \otimes G$. By the induction hypothesis applied to Π_a (resp. Π_b), we obtain that $\Theta :\Uparrow F, L_a$ and $\Theta :\Uparrow G, L_b$ are derivable. By property $\mathcal{L}\otimes$ of lemma 6, we obtain that $\Theta : F \otimes G \Uparrow L_a, L_b$ is derivable, and hence, so is $\Theta :\Uparrow L_a, L_b, F \otimes G$ by application of $[R\Uparrow]$. Finally, given that $L \equiv L_a, L_b, F \otimes G$, by property $\mathcal{L} \equiv$ of lemma 6, we obtain that $\Theta :\Uparrow L$ is derivable.

   The inference figures $[\oplus_l], [\oplus_r], [\exists]$ are treated in the same way, using properties $\mathcal{L}\oplus$ and $\mathcal{L}\exists$ of lemma 6.

3. If Π is simply

$$\Pi \;=\; [1]\,\frac{}{\vdash \Theta : 1}$$

   then $L = 1$ and $\Theta :\Uparrow L$ is derivable by

$$[R\Uparrow]\,\frac{[D_1]\,\dfrac{[1]\,\dfrac{}{\vdash \Theta :\Downarrow 1}}{\vdash \Theta : 1 \Uparrow}}{\vdash \Theta :\Uparrow 1}$$

4. If the last step of Π is an instance of the Logical inference rule $[!]$, we have

$$\Pi \;=\; [!]\,\frac{\overset{\pi'}{\vdots}\;\vdash \Theta : F}{\vdash \Theta : !F}$$

   and $L = !F$. By the induction hypothesis applied to Π', we obtain that $\Theta :\Uparrow F$ is derivable and hence, so is $\Theta :\Uparrow L$ by

$$[R\Uparrow]\,\frac{[D_1]\,\dfrac{[!]\,\dfrac{\dfrac{\vdots}{\vdash \Theta :\Uparrow F}}{\vdash \Theta :\Downarrow !F}}{\vdash \Theta : !F \Uparrow}}{\vdash \Theta :\Uparrow !F}$$

5. If the last step of Π is an instance of the Absorption rule $[\mathbf{A}]$, we have

$$\Pi \;=\; [\mathbf{A}]\,\frac{\overset{\pi'}{\vdots}\;\vdash \Theta', F : L, F}{\vdash \Theta', F : L}$$

   where $\Theta = \Theta', F$. By the induction hypothesis applied to Π', we obtain that $\Theta', F :\Uparrow F, L$ is derivable. By property $\mathcal{L}\mathbf{A}$ of lemma 6, we obtain that $\Theta', F :\Uparrow L$, i.e. $\Theta :\Uparrow L$, is derivable.

6. Finally, if Π is simply

$$\Pi \;=\; [I]\,\frac{}{\vdash \Theta : X, X^{\perp}}$$

then $L = X, X^{\perp}$ and $\Theta :\Uparrow L$ is derivable by

$$[R\Uparrow^2]\,\frac{[D_1]\,\dfrac{[I_1]\,\dfrac{}{\vdash \Theta : X \Downarrow X^{\perp}}}{\vdash \Theta : X, X^{\perp}\Uparrow}}{\vdash \Theta :\Uparrow X, X^{\perp}}$$

Therefore $\mathcal{R}(n+1)$ holds. By induction $\mathcal{R}(n)$ holds for all n. □

A.2.2 The Inversion Lemma $\mathcal{L}\equiv$

We show here property $\mathcal{L}\equiv$ of lemma 6, i.e.

$$\text{if } \vdash \Theta : \Gamma \Uparrow L \text{ and } L \equiv M \text{ then } \vdash \Theta : \Gamma \Uparrow M$$

Demonstration: The proof is long but not difficult. It is sketched here.

- It is first shown that the Logical inference rules concerning $\Uparrow$-sequents in the triadic system apply anywhere in the last field of these sequents (not only to the last formula). In other words,

$$\left\{\begin{array}{lll}
\vdash \Theta : \Gamma \Uparrow L, \bot, M & \text{if} & \vdash \Theta : \Gamma \Uparrow L, M \\
\vdash \Theta : \Gamma \Uparrow L, F \parr G, M & \text{if} & \vdash \Theta : \Gamma \Uparrow L, F, G, M \\
\vdash \Theta : \Gamma \Uparrow L, ?F, M & \text{if} & \vdash \Theta, F : \Gamma \Uparrow L, M \\
\vdash \Theta : \Gamma \Uparrow L, \top, M & & \\
\vdash \Theta : \Gamma \Uparrow L, F \,\&\, G, M & \text{if} & \vdash \Theta : \Gamma \Uparrow L, F, M \text{ and } \vdash \Theta : \Gamma \Uparrow L, G, M \\
\vdash \Theta : \Gamma \Uparrow L, \forall x\, F, M & \text{if} & \vdash \Theta : \Gamma \Uparrow L, F[c/x], M \\
\vdash \Theta : \Gamma, F \Uparrow L, M & \text{if} & \vdash \Gamma \Uparrow L, F, M \text{ and } F \text{ is not asynchronous}
\end{array}\right. \tag{4}$$

  All these properties are shown by induction of the complexity of M (i.e. the sum of the complexities of its elements).

- Then it is shown that

$$\text{if } \vdash \Theta : \Gamma \Uparrow L, F, M \text{ then } \vdash \Theta : \Gamma \Uparrow L, M, F \tag{5}$$

  This is also shown by induction on the complexity of M using, for each case of the last formula of M, the corresponding case in property (4).

- From property (5), it can easily be shown by induction that provability in the Triadic system is preserved under any transposition in the last field of the sequent. More generally, provability is preserved under any permutation. This is precisely the content of property $\mathcal{L}\equiv$ of lemma 6.

□

If $\Theta : \Gamma, F \Uparrow L$ is derivable (where F is not asynchronous), then so is $\Theta : \Gamma \Uparrow L, F$ by application of $[R\Uparrow]$, and, by property $\mathcal{L}\equiv$, so is $\Theta : \Gamma \Uparrow F, L$. But we can state a more specific property, which is used below:

Lemma 7 *Let Θ, Γ be multisets of formulae, L be an ordered list of formulae, and F be a formula (Γ, F containing no asynchronous formula). If $\Theta : \Gamma, F \Uparrow L$ has a proof of depth n then $\Theta : \Gamma \Uparrow F, L$ has a proof of depth at most $n+1$.*

This is shown by a simple induction on the complexity of L.

A.2.3 The Other Inversion Lemmas

We now show property $\mathcal{L}\otimes$ of lemma 6. The remaining properties in this lemma ($\mathcal{L}\oplus$, $\mathcal{L}\exists$ and $\mathcal{L}\mathbf{A}$) are shown in the same way. Thus, we have to show:

$$\text{if } \vdash \Theta : \Gamma \Uparrow F, L \text{ and } \vdash \Theta : \Delta \Uparrow G, M \text{ then } \vdash \Theta : \Gamma, \Delta, F \otimes G \Uparrow L, M$$

where Γ and Δ contain no asynchronous formula. This is shown by induction, but the problem with this induction is that it has to deal with both $\Uparrow$ and $\Downarrow$-sequents in an interleaved way (whereas the proof of $\mathcal{L}\equiv$ was only concerned with $\Uparrow$-sequents).

Demonstration: Let $\mathcal{R}(n)$ be the following induction hypothesis.

$$\mathcal{R}(n) = \mathcal{R}^{\Uparrow}(n) \text{ and } \mathcal{R}^{\Downarrow}(n-1)$$

where

- $\mathcal{R}^{\Uparrow}(n)$ is the property:
  If $\Theta : \Gamma \Uparrow F, L$ and $\Theta : \Delta \Uparrow G, M$ have proofs, whose total depth does not exceed n, then $\Theta : \Gamma, \Delta, F \otimes G \Uparrow L, M$ is derivable.
- $\mathcal{R}^{\Downarrow}(n)$ is the property:
  If F is a synchronous formula or a negative atom and if $\Theta : \Gamma, F \Downarrow H$ and $\Theta : \Delta \Uparrow G$ have proofs, whose total depth does not exceed n, then $\Theta : \Gamma, \Delta, F \otimes G \Downarrow H$ is derivable.

By "total" depth of several proofs, we mean the sum of their depths. Let's assume $\mathcal{R}(n)$, i.e. $\mathcal{R}^{\Uparrow}(n)$ and $\mathcal{R}^{\Downarrow}(n-1)$, and let's show $\mathcal{R}(n+1)$, i.e. $\mathcal{R}^{\Uparrow}(n+1)$ and $\mathcal{R}^{\Downarrow}(n)$

- First, let's show $\mathcal{R}^{\Uparrow}(n+1)$. Let Π_1 and Π_2 be proofs of, respectively, $\Theta : \Gamma \Uparrow F, L$ and $\Theta : \Delta \Uparrow G, M$ such that $\mu(\Pi_1) + \mu(\Pi_2) = n+1$.

  1. If L or M is not empty: say $L = L', H \parr K$ (the other cases for the last formula of L are similar). Therefore

$$\Pi_1 \;=\; [\parr] \dfrac{\overset{\Pi_1'}{\vdots}\; \vdash \Theta : \Gamma \Uparrow F, L', H, K}{\vdash \Theta : \Gamma \Uparrow F, L', H \parr K}$$

  By construction, $\mu(\Pi_1')+\mu(\Pi_2) \leq n$. By the induction hypothesis $\mathcal{R}^{\Uparrow}(n)$ we obtain that $\Theta : \Gamma, \Delta, F \otimes G \Uparrow L', H, K, M$ is derivable, and hence so is $\Theta : \Gamma, \Delta, F \otimes G \Uparrow L, M$ by application of property $\mathcal{L} \equiv$ and the inference rule $[\parr]$.

  2. If both L and M are empty and neither F nor G is a synchronous formula or a negative atom, then $\Theta : \Gamma, \Delta, F \otimes G \Uparrow$ is derivable by

$$[D_1] \dfrac{[\otimes] \dfrac{[R\Downarrow] \dfrac{\overset{\Pi_1}{\vdots}\; \vdash \Theta : \Gamma \Uparrow F}{\vdash \Theta : \Gamma \Downarrow F} \quad [R\Downarrow] \dfrac{\overset{\Pi_2}{\vdots}\; \vdash \Theta : \Delta \Uparrow G}{\vdash \Theta : \Delta \Downarrow G}}{\vdash \Theta : \Gamma, \Delta \Downarrow F \otimes G}}{\vdash \Theta : \Gamma, \Delta, F \otimes G \Uparrow}$$

  The inferences $[R\Downarrow]$ above are valid only because of the hypotheses on F and G.

  3. If both L and M are empty and F is a synchronous formula or a negative atom, then the last inference step of Π_1 is an instance of the Reaction rule $[R\Uparrow]$, preceded by an instance of a Decision rule $[D_1]$ or $[D_2]$. Three cases are possible:

     (a) If

$$\Pi_1 \;=\; [R\Uparrow] \dfrac{[D_1] \dfrac{\overset{\Pi_1'}{\vdots}\; \vdash \Theta : \Gamma', F \Downarrow H}{\vdash \Theta : \Gamma', H, F \Uparrow}}{\vdash \Theta : \Gamma', H \Uparrow F}$$

     where $\Gamma = \Gamma', H$, then $\mu(\Pi_1') + \mu(\Pi_2) \leq n-1$. By the induction hypothesis $\mathcal{R}^{\Downarrow}(n-1)$ we obtain that $\Theta : \Gamma', \Delta, F \otimes G \Downarrow H$ is derivable, and hence, so is $\Theta : \Gamma, \Delta, F \otimes G \Uparrow$ by application of $[D_1]$.

     (b) If

$$\Pi_1 \;=\; [R\Uparrow] \dfrac{[D_2] \dfrac{\overset{\Pi_1'}{\vdots}\; \vdash \Theta', H : \Gamma, F \Downarrow H}{\vdash \Theta', H : \Gamma, F \Uparrow}}{\vdash \Theta', H : \Gamma \Uparrow F}$$

     where $\Theta = \Theta', H$, then $\mu(\Pi_1') + \mu(\Pi_2) \leq n-1$. By the induction hypothesis $\mathcal{R}^{\Downarrow}(n-1)$ we obtain that $\Theta : \Gamma, \Delta, F \otimes G \Downarrow H$ is derivable, and hence, so is $\Theta : \Gamma, \Delta, F \otimes G \Uparrow$ by application of $[D_2]$.

     (c) If

$$\Pi_1 \;=\; [R\Uparrow] \dfrac{[D_1] \dfrac{\overset{\Pi_1'}{\vdots}\; \vdash \Theta : \Gamma \Downarrow F}{\vdash \Theta : \Gamma, F \Uparrow}}{\vdash \Theta : \Gamma \Uparrow F}$$

     then, by symmetry, we can assume that G is a synchronous formula or a negative atom and that

$$\Pi_2 \;=\; [R\Uparrow] \dfrac{[D_1] \dfrac{\overset{\Pi_2'}{\vdots}\; \vdash \Theta : \Delta \Downarrow G}{\vdash \Theta : \Delta, G \Uparrow}}{\vdash \Theta : \Delta \Uparrow G}$$

Therefore, $\Theta : \Gamma, \Delta, F \otimes G \Uparrow$ is derivable by

$$[D_1]\ \dfrac{[\otimes]\ \dfrac{\overbrace{\vdots}^{\Pi'_1} \quad \overbrace{\vdots}^{\Pi'_2}}{\vdash \Theta : \Gamma \Downarrow F \quad \vdash \Theta : \Delta \Downarrow G}}{\vdash \Theta : \Gamma, \Delta \Downarrow F \otimes G} \qquad \vdash \Theta : \Gamma, \Delta, F \otimes G \Uparrow$$

Therefore $\mathcal{R}^{\Uparrow}(n+1)$ holds.

- Now, let's show $\mathcal{R}^{\Downarrow}(n)$. Let F be a synchronous formula or a negative atom and let Π_1 and Π_2 be proofs of, respectively, $\Theta : \Gamma, F \Downarrow H$ and $\Theta : \Delta \Uparrow G$ such that $\mu(\Pi_1) + \mu(\Pi_2) = n$.

  1. If H is synchronous: say $H = H_a \oplus H_b$ (the other cases for H are similar). Therefore, for example,

$$\Pi_1 \ = \ [\oplus_l]\ \dfrac{\overbrace{\vdots}^{\Pi'_1}}{\vdash \Theta : \Gamma, F \Downarrow H_a}}{\vdash \Theta : \Gamma, F \Downarrow H_a \oplus H_b}$$

     By construction, $\mu(\Pi'_1)+\mu(\Pi_2) \leq n-1$. By the induction hypothesis $\mathcal{R}^{\Downarrow}(n-1)$ we obtain that $\Theta : \Gamma, \Delta, F \otimes G \Downarrow H_a$ is derivable, and hence so is $\Theta : \Gamma, \Delta, F \otimes G \Downarrow H$ by application of $[\oplus_l]$.

  2. If H is not synchronous, then H cannot be a negative atom $X^{\perp}$, since, in that case, we would have necessarily

$$\Pi_1 \ = \ [I_1]\ \dfrac{}{\vdash \Theta : X \Downarrow X^{\perp}}$$

     This would require $\Gamma = \emptyset$ and $F = X$. So F would be a positive atom, in contradiction with the hypothesis that F is a synchronous formula or a negative atom. Therefore, H being neither synchronous nor a negative atom, we have

$$\Pi_1 \ = \ [R \Downarrow]\ \dfrac{\overbrace{\vdots}^{\Pi'_1}}{\vdash \Theta : \Gamma, F \Uparrow H}}{\vdash \Theta : \Gamma, F \Downarrow H}$$

     Therefore, $\Theta : \Gamma, F \Uparrow H$ has a proof Π'_1. By lemma 7, given that F is not asynchronous, we obtain that $\Theta : \Gamma \Uparrow F, H$ has a proof of depth at most $\mu(\Pi'_1) + 1$, i.e. $\mu(\Pi_1)$. Therefore, $\Theta : \Gamma \Uparrow F, H$ and $\Theta : \Delta \Uparrow G$ have proofs, whose total depth does not exceed $\mu(\Pi_1) + \mu(\Pi_2)$, which is equal to n, by hypothesis. By the induction hypothesis $\mathcal{R}^{\Uparrow}(n)$ we obtain that $\Theta : \Gamma, \Delta, F \otimes G \Uparrow H$ is derivable, and hence so is $\Theta : \Gamma, \Delta, F \otimes G \Downarrow H$ by application of $[R \Downarrow]$.

  Therefore $\mathcal{R}^{\Downarrow}(n)$ holds.

Therefore, both $\mathcal{R}^{\Uparrow}(n+1)$ and $\mathcal{R}^{\Downarrow}(n)$ hold. Therefore $\mathcal{R}(n+1)$ holds. By induction $\mathcal{R}(n)$ holds for all n. □

A.3 Soundness and Completeness of LinLog

We have to show the result stated in theorem 3, i.e.

$$!\mathcal{P} \vdash g \ \text{ iff } \ \vdash_{\mathcal{P}} :\Uparrow g$$

We have

- $!\mathcal{P} \vdash g$ if and only if [transposition to monadic sequents]
- $\vdash\ ?\mathcal{P}^{\perp}, g$ if and only if [projection to dyadic sequents, theorem 1]
- $\vdash \mathcal{P}^{\perp} : g$ if and only if [projection to triadic sequents, theorem 2]
- $\vdash \mathcal{P}^{\perp} :\Uparrow g$

Therefore, we have to show that

$$\vdash \mathcal{P}^{\perp} :\Uparrow g \ \text{ iff } \ \vdash_{\mathcal{P}} :\Uparrow g$$

This is achieved by a trivial induction on the depth of the proof. The induction hypothesis is

> If $\mathcal{P}^{\perp}, \Psi : \Phi \Uparrow \mathcal{G}$ (resp. $\mathcal{P}^{\perp}, \Psi : \Phi \Downarrow G$) has a proof of depth at most n in Σ_3 then $\Psi : \Phi \Uparrow \mathcal{G}$ (resp. $\Psi : \Phi \Downarrow G$) is derivable in $\Sigma_3[\mathcal{P}]$.

(the hypothesis for the reverse implication is similar). The only non obvious case of the induction comes from the Progression rule $[\circ\!\!-]$ and is solved by the following lemma.

Lemma 8 *Let Ψ, Φ be multisets of positive atoms.*

- *If $\Psi_o, \Phi_o \multimap G_o$ is an instance of a method in $\mathcal{P}$, and $\vdash \mathcal{P}^\perp, \Psi, \Psi_o : \Phi \Downarrow G_o$, then $\vdash \mathcal{P}^\perp, \Psi, \Psi_o : \Phi, \Phi_o \Uparrow$.*
- *Conversely, if $\vdash \mathcal{P}^\perp, \Psi : \Phi \Uparrow$ then there exists an instance $\Psi_o, \Phi_o \multimap G_o$ of a method in $\mathcal{P}$, and multisets of positive atoms Ψ' and Φ' such that*

$$\Psi = \Psi_o, \Psi' \text{ and } \Phi = \Phi_o, \Phi' \text{ and } \vdash \mathcal{P}^\perp, \Psi', \Psi_o : \Phi' \Downarrow G_o$$

Demonstration:

- Let M be a method in $\mathcal{P}$, say

$$M \quad = \quad \forall x\, p(x) \parr q(x) \multimap G$$

and let $\Psi_o, \Phi_o \multimap G_o$ be an instance of M, say

$$\Psi_o = p(t) \qquad \Phi_o = q(t) \qquad G_o = G[t/x]$$

where t is a ground term. Let's assume that $\mathcal{P}^\perp, \Psi, \Psi_o : \Phi \Downarrow G_o$ is derivable. Then so is $\mathcal{P}^\perp, \Psi, \Psi_o : \Phi, \Phi_o \Uparrow$ by

$$[D_2]\ \dfrac{[\exists]\ \dfrac{\overset{\Pi'}{\vdots}}{\vdash \mathcal{P}^\perp, \Psi, \Psi_o : \Phi, \Phi_o \Downarrow p(t)^\perp \otimes q(t)^\perp \otimes G_o}}{\vdash \mathcal{P}^\perp, \Psi, \Psi_o : \Phi, \Phi_o \Downarrow M^\perp} \Big/ \vdash \mathcal{P}^\perp, \Psi, \Psi_o : \Phi, \Phi_o \Uparrow$$

where Π' is the following proof:

$$[\otimes]\ \dfrac{[I_2]\ \dfrac{}{\vdash \mathcal{P}^\perp, \Psi, p(t) :\Downarrow p(t)^\perp} \quad [\otimes]\ \dfrac{[I_1]\ \dfrac{}{\vdash \Theta : q(t) \Downarrow q(t)^\perp} \quad \dfrac{\vdots}{\vdash \Theta : \Phi \Downarrow G_o}}{\vdash \Theta : \Phi, q(t) \Downarrow q(t)^\perp \otimes G_o}}{\vdash \mathcal{P}^\perp, \Psi, p(t) : \Phi, q(t) \Downarrow p(t)^\perp \otimes q(t)^\perp \otimes G_o}$$

where Θ stands for $\mathcal{P}^\perp, \Psi, p(t)$.

- Conversely, let's assume that $\mathcal{P}^\perp, \Psi : \Phi \Uparrow$ has a proof Π. The last step of Π is necessarily an instance of a Decision rule $[D_1]$ or $[D_2]$. As the selected formula in these rules must not be a positive atom and Ψ, Φ contain only positive atoms, we have necessarily

$$\Pi = [D_2]\ \dfrac{\overset{\Pi'}{\vdots}}{\vdash \mathcal{P}^\perp, \Psi : \Phi \Downarrow M^\perp} \Big/ \vdash \mathcal{P}^\perp, \Psi : \Phi \Uparrow$$

where M is a method in $\mathcal{P}$, say

$$M \quad = \quad \forall x\, p(x) \parr q(x) \multimap G$$

By construction, Π' starts a critical focusing section, and is therefore necessarily of the form

$$\Pi' \quad = \quad [\exists]\ \dfrac{\overset{\Pi''}{\vdots}}{\vdash \mathcal{P}^\perp, \Psi : \Phi \Downarrow p(t)^\perp \otimes q(t)^\perp \otimes G_o} \Big/ \vdash \mathcal{P}^\perp, \Psi : \Phi \Downarrow M^\perp$$

where t is a ground term (selected at step $[\exists]$) and $G_o = G[t/x]$. Let Θ be $\mathcal{P}^\perp, \Psi$. As the critical section proceeds, we have necessarily

$$\Pi'' \quad = \quad [\otimes]\ \dfrac{\overset{\Pi_p}{\vdots}\ \vdash \Theta : \Phi_p \Downarrow p(t)^\perp \quad [\otimes]\ \dfrac{\overset{\Pi_q}{\vdots}\ \vdash \Theta : \Phi_q \Downarrow q(t)^\perp \quad \overset{\Pi_o}{\vdots}\ \vdash \Theta : \Phi' \Downarrow G_o}{\vdash \Theta : \Phi_q, \Phi' \Downarrow q(t)^\perp \otimes G_o}}{\vdash \Theta : \Phi_p, \Phi_q, \Phi' \Downarrow p(t)^\perp \otimes q(t)^\perp \otimes G_o}$$

where $\Phi = \Phi_p, \Phi_q, \Phi'$. Now, on each of the two leftmost branches, a negative atom is reached which terminates the critical section. Hence one of the Identities ($[I_1]$ or $[I_2]$) must be used, say

$$\Pi_p \quad = \quad [I_2]\ \dfrac{}{\vdash \Theta :\Downarrow p(t)^\perp} \qquad \Pi_q \quad = \quad [I_1]\ \dfrac{}{\vdash \Theta : q(t) \Downarrow q(t)^\perp}$$

where $p(t) \in \Theta$. Therefore, $\Phi_p = \emptyset$ and $\Phi_q = q(t)$, and hence, $\Phi = q(t), \Phi'$. Furthermore, since $p(t)$ is in $\mathcal{P}^\perp, \Psi$ and $\mathcal{P}^\perp$ cannot contain positive atoms, $p(t)$ must be in Ψ and we obtain that $\Psi = p(t), \Psi'$. Finally we have

$$\Phi = \Phi_o, \Phi'$$
$$\Psi = \Psi_o, \Psi'$$

where $\Psi_o = p(t)$ and $\Phi_o = q(t)$. Therefore $\Psi_o, \Phi_o \circ\!\!- G_o$ is an instance of M. Furthermore, Π_o is a proof of $\mathcal{P}^\perp, \Psi : \Phi' \Downarrow G_o$.

□

A.4 Normalization to LinLog Form

We have to show the result stated in theorem 4, i.e.

$$\vdash F \text{ iff } \vdash_{\mathcal{P}} g$$

where $\langle \mathcal{P}; g \rangle = \text{NORMALIZE}(F)$.

Let's extend the definition of the function NORMALIZE to triadic sequents $\Theta : \Gamma \Uparrow L$ (where Γ contains no asynchronous formula). It returns an extended LinLog query consisting of a LinLog program and a sequent $\Psi : \Phi \Uparrow \mathcal{G}$ where Ψ and Φ are multisets of atoms and $\mathcal{G}$ is an ordered list of elementary goals.

Function NORMALIZE($\Theta : \Gamma \Uparrow L$) **returns** an extended LinLog query

$\mathcal{P} := \emptyset$;
$\mathcal{G} :=$ NGOAL(L); $\Phi :=$ NATOM(Γ); $\Psi :=$ NATOM(Θ);
Return $\langle \mathcal{P} \ ; \ \Psi : \Phi \Uparrow \mathcal{G} \rangle$

Remark: in this definition, when the function NATOM or NGOAL is applied to a list (resp. multiset) of formulae, it returns the list (resp. multiset) of images of these formulae. We now have to show the more general property

$$\vdash \Theta : \Gamma \Uparrow L \text{ iff } \vdash_{\mathcal{P}} \Psi : \Phi \Uparrow \mathcal{G}$$

where $\langle \mathcal{P} \ ; \ \Psi : \Phi \Uparrow \mathcal{G} \rangle = \text{NORMALIZE}(\Theta : \Gamma \Uparrow L)$. Let's show here the direct implication (the converse is similar).

Demonstration: Let $\mathcal{R}(n)$ be the induction hypothesis

If $\Theta : \Gamma \Uparrow L$ has a proof of depth at most n in Σ_3 then $\Psi : \Phi \Uparrow \mathcal{G}$ is derivable in $\Sigma_3[\mathcal{P}]$.

Let's assume $\mathcal{R}(n)$ and let's show $\mathcal{R}(n+1)$. Let Π be a proof of depth $n+1$ of $\Theta : \Gamma \Uparrow L$.

1. If L is not empty: say $L = L', F_1 \parr F_2$ (the other cases for the last formula of L are similar). Therefore

$$\Pi \ = \ [\parr] \ \frac{\overbrace{\vdots}^{\Pi'}}{\vdash \Theta : \Gamma \Uparrow L', F_1, F_2} \\ \vdash \Theta : \Gamma \Uparrow L', F_1 \parr F_2$$

Let $\mathcal{G}' = \text{NGOAL}(L')$ and $g_1 = \text{NGOAL}(F_1)$ and $g_2 = \text{NGOAL}(F_2)$. By definition of the function NGOAL, we obtain that $\text{NGOAL}(F_1 \parr F_2) = g_1 \parr g_2$. Therefore, $\mathcal{G} = \mathcal{G}', g_1 \parr g_2$. By the induction hypothesis applied to Π' we obtain that $\Psi : \Phi \Uparrow \mathcal{G}', g_1, g_2$ is derivable in $\Sigma_3[\mathcal{P}]$ and hence, so is $\Psi : \Phi \Uparrow \mathcal{G}$ by application of $[\parr]$.

2. If L is empty, the last step of Π is necessarily an instance of a Decision rule $[D_1]$ or $[D_2]$, say $[D_1]$

$$\Pi \ = \ [D_1] \ \frac{\overbrace{\vdots}^{\Pi'}}{\vdash \Theta : \Gamma' \Downarrow F} \\ \vdash \Theta : \Gamma', F \Uparrow$$

where $\Gamma = \Gamma', F$ and F is not a positive atom. Let $u = \text{NATOM}(F)$. As F is not a positive atom, u is a Skolem constant and $\|\text{NMETH}(u^\perp \otimes F)\|$ is included in $\mathcal{P}$. Now, let's apply to the topmost layer of synchronous connectives of F the following transformations, which express associativity, commutativity and distributivity properties of the synchronous connectives.

$(F_1 \otimes F_2) \otimes F_3$	$\mapsto$	$F_1 \otimes (F_2 \otimes F_3)$	$(F_1 \oplus F_2) \oplus F_3$	$\mapsto$	$F_1 \oplus (F_2 \oplus F_3)$
$F_1 \otimes F_2$	$\mapsto$	$F_2 \otimes F_1$	$F_1 \oplus F_2$	$\mapsto$	$F_2 \oplus F_1$
$1 \otimes F$	$\mapsto$	F	$0 \oplus F$	$\mapsto$	F
$F_1 \otimes (F_2 \oplus F_3)$	$\mapsto$	$(F_1 \otimes F_3) \oplus (F_2 \otimes F_3)$			
$0 \otimes F$	$\mapsto$	0			

Notice that these transformations do not modify NMETH(F) since the operators $\star$ and $\cup$ on sets of quasi-clauses are associative commutative, have a neutral element (resp. $\{[\emptyset;\emptyset]\}$ and $\emptyset$) and $\star$ is distributive over $\cup$. Furthermore,

provability is also preserved, and even the focusing proofs, up to some irrelevant local reordering of the inferences. Thus, for the distributivity property, we have

$$[\otimes]\ \frac{\dfrac{\vdots}{\vdash \Theta : \Gamma_1 \Downarrow F_1}\quad [\oplus_l]\ \dfrac{\dfrac{\vdots}{\vdash \Theta : \Gamma_2 \Downarrow F_2}}{\vdash \Theta : \Gamma_2 \Downarrow F_2 \oplus F_3}}{\vdash \Theta : \Gamma_1, \Gamma_2 \Downarrow F_1 \otimes (F_2 \oplus F_3)} \quad \mapsto \quad [\oplus_l]\ \frac{[\otimes]\ \dfrac{\dfrac{\vdots}{\vdash \Theta : \Gamma_1 \Downarrow F_1}\quad \dfrac{\vdots}{\vdash \Theta : \Gamma_2 \Downarrow F_2}}{\vdash \Theta : \Gamma_1, \Gamma_2 \Downarrow F_1 \otimes F_2}}{\vdash \Theta : \Gamma_1, \Gamma_2 \Downarrow (F_1 \otimes F_2) \oplus (F_1 \otimes F_3)}$$

Hence, we may assume with no loss of generality that F is in normal form with respect to the transformation system above, i.e.

$$F = F_1 \oplus \cdots \oplus F_n$$

where $n \geq 0$ (if $n = 0$ then $F = 0$) and each F_k is of the form

$$F_k = H_1 \otimes \cdots \otimes H_m$$

where $m \geq 0$ (if $m = 0$ then $F_k = 1$) and the topmost connective of each H_j is neither of $0, 1, \oplus, \otimes$.
By construction, Π' starts a critical focusing section, and is therefore necessarily of the form

$$\Pi' \ = \ [\oplus]\ \frac{\overbrace{\vdots}^{\Pi''}}{\vdash \Theta : \Gamma' \Downarrow F_k}}{\vdash \Theta : \Gamma' \Downarrow F}$$

where $k \in \{1 \ldots n\}$. Let's make explicit among the component formulae H_j of F_k those which are negative atoms, say

$$F_k = p^\perp \otimes q^\perp \otimes H$$

where $p^\perp, q^\perp$ are negative atoms and H is a tensor product of formulae which either are positive atoms, or start with an asynchronous connective, or with the modality ! (since the other connectives are precluded). Let's take for instance

$$H = A \otimes {!}B$$

where A is a positive atom or an asynchronous formula and B is a formula.
As the critical section proceeds, we have

$$\Pi'' \ = \ [\otimes]\ \frac{\overbrace{\vdots}^{\Pi_p} \quad \overbrace{\vdots}^{\Pi_q} \quad \overbrace{\vdots}^{\Pi_A} \quad \overbrace{\vdots}^{\Pi_B}}{\vdash \Theta : \Gamma_p \Downarrow p^\perp \quad \vdash \Theta : \Gamma_q \Downarrow q^\perp \quad \vdash \Theta : \Gamma_A \Downarrow A \quad \vdash \Theta : \Gamma_B \Downarrow {!}B}{\vdash \Theta : \Gamma_p, \Gamma_q, \Gamma_A, \Gamma_B \Downarrow F_k}$$

where $\Gamma' = \Gamma_p, \Gamma_q, \Gamma_A, \Gamma_B$. Now, in Π_p and Π_q, a negative atom is reached which terminates the critical section. Hence, the identities must be used, say

$$\Pi_p \ = \ [I_2]\ \frac{}{\Theta :\Downarrow p^\perp} \qquad \Pi_q \ = \ [I_1]\ \frac{}{\Theta : q \Downarrow q^\perp}$$

where $p \in \Theta$. Therefore $\Gamma_p = \emptyset$ and $\Gamma_q = q$. Furthermore, given that A is an asynchronous formula or a positive atom, we have

$$\Pi_A \ = \ [R\Downarrow]\ \frac{\dfrac{\vdots}{\vdash \Theta : \Gamma_A \Uparrow A}}{\vdash \Theta : \Gamma_A \Downarrow A} \qquad \Pi_B \ = \ [!]\ \frac{\dfrac{\vdots}{\vdash \Theta :\Uparrow B}}{\vdash \Theta :\Downarrow {!}B}$$

Therefore $\Gamma_B = \emptyset$ and hence, $\Gamma = q, \Gamma_A, F$. Let $g_A = \text{NGOAL}(A)$ and $g_B = \text{NGOAL}(B)$ and $\Phi_A = \text{NATOM}(\Gamma_A)$. By the induction hypothesis we obtain that $\Psi : \Phi_A \Uparrow g_A$ and $\Psi :\Uparrow g_B$ are derivable in $\Sigma_3[\mathcal{P}]$.
Now, by definition of the function NMETH, we obtain that

$$\text{NMETH}(F_k) = \{[p, q \ ; \ g_A, {!}g_B]\}$$

and hence, the method $M = u \parr p \parr q \multimap g_A \otimes {!}g_B$ is in $\|\text{NMETH}(u^\perp \otimes F)\|$, and hence in $\mathcal{P}$. Since $\text{NATOM}(p) = p$ and $p \in \Theta$, we obtain that $p \in \Psi$. Therefore, $\Psi : \Phi_A, u, q \Uparrow$ is derivable in $\Sigma_3[\mathcal{P}]$ by triggering method M:

$$[\multimap]\ \frac{[\otimes]\ \dfrac{[R\Downarrow]\ \dfrac{\dfrac{\vdots}{\vdash \Psi : \Phi_A \Uparrow g_A}}{\vdash \Psi : \Phi_A \Downarrow g_A}\quad [!]\ \dfrac{\dfrac{\vdots}{\vdash \Psi :\Uparrow g_B}}{\vdash \Psi :\Downarrow {!}g_B}}{\vdash \Psi : \Phi_A \Downarrow g_A \otimes {!}g_b}}{\vdash \Psi : \Phi_A, u, q \Uparrow}$$

Now, since $\Gamma = \Gamma_A, q, F$, we obtain that $\Phi = \Phi_A, q, u$, and therefore $\Psi : \Phi \Uparrow$ is derivable in $\Sigma_3[\mathcal{P}]$.
Therefore $\mathcal{R}(n+1)$ holds. By induction $\mathcal{R}(n)$ holds for all n. □

A.5 Cut elimination in the Triadic system

A formula H is taken to be "reducible" if any instance of the Cut rule using H as Cut formula can be eliminated from any proof. We have to show that all formulae are reducible. For simplification purpose, we only show here that formulae containing no modalities are reducible; the reasoning below could be adapted to avoid this restriction but it would become much more complicated.

Demonstration: To prove that any (modality-free) formula H is reducible, we reason by induction on H. Consider an occurrence of the Cut rule

$$[\mathbf{C}]\ \frac{\overset{\Pi}{\vdots}\ \vdash \Theta : \Gamma \Downarrow H \qquad \overset{\Pi'}{\vdots}\ \vdash \Theta : \Delta \Downarrow H^{\perp}}{\vdash \Theta : \Gamma, \Delta \Uparrow}$$

where Cuts have already been eliminated from Π and Π'. There are two cases in the induction.

1. If H is atomic, then, by symmetry, we may consider that H is a positive atom X. Therefore, Π is necessarily of the form

$$\Pi \ = \ [R\Downarrow, R\Uparrow]\ \frac{\overset{\vdots}{\vdash \Theta : \Gamma, X \Uparrow}}{\vdash \Theta : \Gamma \Downarrow X}$$

   Hence, $\Theta : \Gamma, X \Uparrow$ is derivable. Moreover, as Π' is focusing, it is necessarily reduced to an identity. The two forms are possible:

   - If
   $$\Pi' \ = \ [I_1]\ \frac{}{\vdash \Theta : X \Downarrow X^{\perp}}$$
   then $\Delta = X$ and $\Theta : \Gamma, \Delta \Uparrow$ is identical to $\Theta : \Gamma, X \Uparrow$ which is derivable.
   - If
   $$\Pi' \ = \ [I_2]\ \frac{}{\vdash \Theta :\Downarrow X^{\perp}}$$
   then Δ is empty and X must be in Θ. But then, we can obtain a proof of $\Theta : \Gamma \Uparrow$ simply by discarding X from the proof of $\Theta : \Gamma, X \Uparrow$. The only case where this would not be possible is in a step where the Identity rule $[I_1]$ is applied to match X and $X^{\perp}$, but such steps can be replaced by the Identity rule $[I_2]$, which does not make use of X (since it is already in Θ). Thus, we obtain a proof of $\Theta : \Gamma \Uparrow$, which is identical to $\Theta : \Gamma, \Delta \Uparrow$ (since Δ is empty).

   In both case, we obtain a (Cut-free) proof of $\Theta : \Gamma, \Delta \Uparrow$. Therefore, atomic formulae are always reducible.

2. Let's now assume that H is not atomic and that all its strict sub-formulae are reducible. We want to show that H itself is reducible. By symmetry, we may assume that H is asynchronous. Let, for example, $H = A \parr (B \,\&\, C)$ be its topmost asynchronous layer (i.e. A, B, C are not asynchronous). The idea here is that the reduction of this topmost layer can be performed "in one step". Since Π and Π' are focusing, and A, B, C are not asynchronous, the two proofs are necessarily of the form

$$\Pi \ = \ [\parr, \&]\ \frac{\overset{\vdots}{\vdash \Theta : \Gamma, A, B \Uparrow} \qquad \overset{\vdots}{\vdash \Theta : \Gamma, A, C \Uparrow}}{\vdash \Theta : \Gamma \Downarrow A \parr (B \,\&\, C)} \quad \text{and} \quad \Pi' \ = \ [\otimes, \oplus_l]\ \frac{\overset{\vdots}{\vdash \Theta : \Delta_A \Downarrow A^{\perp}} \qquad \overset{\vdots}{\vdash \Theta : \Delta_B \Downarrow B^{\perp}}}{\vdash \Theta : \Delta_A, \Delta_B \Downarrow A^{\perp} \otimes (B^{\perp} \oplus C^{\perp})}$$

   where $\Delta = \Delta_A, \Delta_B$ (notice that we have assumed here that the rule $[\oplus_l]$ is used to decompose $B^{\perp} \oplus C^{\perp}$; the other possibility $[\oplus_r]$ would be treated in the same way). Thus, $\Theta : \Delta_A \Downarrow A^{\perp}$ and $\Theta : \Delta_B \Downarrow B^{\perp}$ are derivable. Now, in the proof of $\Theta : \Gamma, A, B \Uparrow$, let's replace the occurrences of A by Δ_A and those of B by Δ_B. This is always possible except in two cases (given here for A but B is treated in the same way):

   - The Decision rule $[D_1]$ is used to select A:
   $$[D_1]\ \frac{\overset{\vdots}{\vdash \Theta' : \Gamma' \Downarrow A}}{\vdash \Theta' : \Gamma', A \Uparrow}$$
   In this case, A can still be replaced by Δ_A, introducing a Cut on A:
   $$[\mathbf{C}]\ \frac{\overset{\vdots}{\vdash \Theta' : \Gamma' \Downarrow A} \qquad \overset{\vdots}{\vdash \Theta' : \Delta_A \Downarrow A^{\perp}}}{\vdash \Theta' : \Gamma', \Delta_A \Uparrow}$$

- The Identity rule $[I_1]$ is used to match A and $A^\perp$ (assuming A is a positive atom):

$$[I_1]\ \frac{}{\vdash \Theta' : A \Downarrow A^\perp}$$

In this case also, A can be replaced by Δ_A, yielding $\Theta' : \Delta_A \Downarrow A^\perp$ which is derivable.

Therefore, by replacing A by Δ_A and B by Δ_B, we obtain a proof of $\Theta : \Gamma, \Delta_A, \Delta_B \Uparrow$ possibly containing Cuts on A and B. But, as A and B are subformulae of H, by the induction hypothesis, they are reducible. Hence we can eliminate Cuts on these formulae and obtain a Cut-free proof of $\Theta : \Gamma, \Delta \Uparrow$. Therefore, H is itself reducible.

□