The Structure of Interaction*

Stéphane Gimenez and Georg Moser

Institute of Computer Science
University of Innsbruck, Austria
{stephane.gimenez,georg.moser}@uibk.ac.at

Abstract

Interaction nets form a local and strongly confluent model of computation that is per se parallel. We introduce a Curry–Howard correspondence between well-formed interaction nets and a deep-inference deduction system based on linear logic. In particular, linear logic itself is easily expressed in the system and its computational aspects materialise though the correspondence. The system of interaction nets obtained is a typed variant of already well-known sharing graphs. Due to a strong confluence property, strong normalisation for this system follows from weak normalisation. The latter is obtained via an adaptation of Girard's reducibility method. The approach is modular, readily gives rise to generalisations (e.g. second order, known as polymorphism to the programmer) and could therefore be extended to various systems of interaction nets.

1998 ACM Subject Classification F.1.2 Modes of Computation, F.4.1 Mathematical Logic

Keywords and phrases Interaction Nets, Linear Logic, Curry–Howard Correspondence, Deep Inference, Calculus of Structures, Strong Normalisation, Reducibility

Digital Object Identifier 10.4230/LIPIcs.CSL.2013.316

1 Introduction

We introduce a deep-inference deduction system based on multiplicative-exponential linear logic (MELL for short) [4] and provide a direct correspondence with interaction nets systems [11], among which sharing graphs [12, 5, 6] play a key role. Our calculus was directly inspired by, and is in large parts identical to, an earlier presentation of MELL in the calculus of structures given by Guglielmi and Straßburger [16, 15]. We thus unveil a Curry–Howard correspondence between deep-inference formalisms of linear logic and the simple, almost canonical, parallel computation model portrayed by interaction nets. On the one hand, the deep-inference deduction system fulfills the role of a long-awaited enhanced type system for nets: the additional structure conferred to nets through typing ensures correctness, and can furthermore, under some reasonable assumptions, guarantee termination. On the other hand, interaction nets provide an answer to the unsettled topic of a computational interpretation for proof normalization steps in deep-inference systems.

Interaction nets, introduced by Lafont in [11], form an abstract model of computation based on graph rewriting [18]. Reduction of interaction nets is *strongly confluent*: pairs of interacting agents can not only be contracted locally but also independently, and therefore any peak can be joined immediately. This gives rise to an elegant formalism to express parallel computations. These and other merits make interaction nets a promising programming paradigm, either as an execution platform for functional programs, or as a conceptional device for the (optimal) implementation of the λ-calculus [6, 13, 7]. However, the elegance

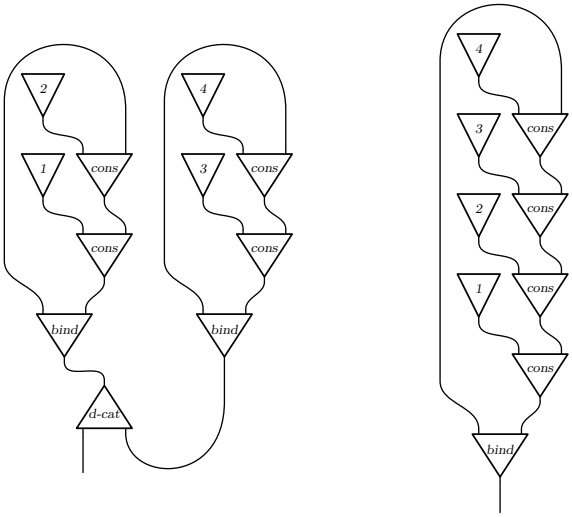
* This work is partially supported by FWF (Austrian Science Fund) project I-603-N18.

Computer Science Logic 2013 (CSL'13).
Editor: Simona Ronchi Della Rocca; pp. 316–331

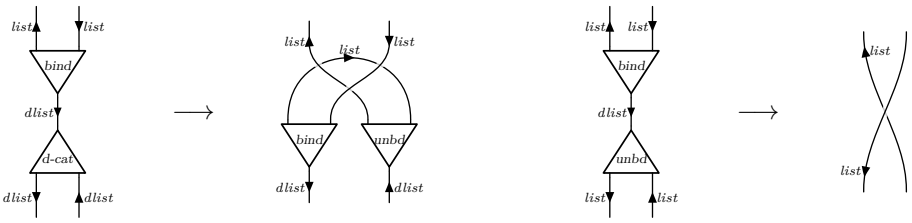
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

of nets has always been undermined by the lack of a versatile framework to guarantee basic correctness assumptions.

▶ **Example 1.1.** As an illustration, we reuse the difference lists example from [11], an implementation of lists that allows constant time concatenation. The two nets pictured below describe the concatenation of integer lists $[1, 2]$ and $[3, 4]$, before and after evaluation. Node *cons* is used as a standard list constructor, and *bind* creates a difference list from links to the tail and to the head of a standard list.



The computation is performed by means of the following rewriting rules:



After two reduction steps, reduction of the initial net yields the given normal form.

Such simple examples suffice to understand that interaction nets crucially miss structure, making reasoning about them difficult. Individual portions of nets found in left and right hand side of reduction rules are not standardly considered valid, even though they are subnets of valid nets. The reason being that usual net-construction rules available for this system, which are inherited from sequent-calculus inference rules, are too coarse-grained. Without preserving additional typing information, the inherent well-foundness of such rewriting rules could not be formulated directly [11]; a proof of a correctness preservation property would have to be conducted globally for the full reduction relation. The correspondence proposed in this paper allows for a finer analysis of nets, keeping track of their substantial structure, and is able to do so even in the most delicate case of sharing graphs.

Deep-inference deduction systems allow inference rules to modify formulas inside an arbitrary context. For most logics deep-inference presentations have been established [16, 9, 17, 8]. Their flexibility enables us to type and to ensure proper combination of individual cell-components of a net. We can fuse through this type system the essential properties of interaction nets and MELL. From nets we inherit strong confluence and from MELL

we inherit weak normalisation. Technically we adapt Girard's *reducibility method* to the context of deep-inference formalism to obtain weak normalisation. The proof is modular and can easily be extended, for instance, to additive connectives (which are used to type conditionals) or to second-order (polymorphism).

This work is related to very recent work on computational interpretations of deep-inference systems. In [10] Gundersen, Heijltjes and Parigot introduce the *atomic lambda calculus* as a typed λ-calculus that admits some particular form of sharing and preserves strong normalisation. The atomic lambda calculus provides (among others) a computational interpretation of the medial rule of calculi of structures [17]. Furthermore, recently, Pagani and Tortora de Falco established in [14] the very tedious and technical standardisation theorem required to prove strong normalisation of second-order linear logic. Its "delicate but boring" proof had been postponed but Girard's reducibility method however suffices in showing weak normalisation. When adapted to our framework, thanks to strong confluence, weak normalisation automatically promotes to a strong normalisation theorem.

This paper is structured as follows. In Section 2 we provide the proposed deep-inference presentation of MELL together with an embedding of its standard sequent calculus presentation. Section 3 defines the system of interaction nets which is of interest, picturing the direct correspondance with this presentation of MELL. In Section 4, we define reduction steps on the logic side, in a way that preserves strong confluence from nets. Section 5 adapts Girard's reducibility method to this setting and establishes weak normalisation (and thus strong normalisation). Finally, we conclude in Section 6 and mention potential future work.

2 A Convenient Presentation of Linear Logic with Structures

Structures. We define structures as follows:

$$\sigma \quad ::= \quad \star \;\mid\; \sigma \,;\, \sigma \;\mid\; \circ \;\mid\; \sigma \,,\, \sigma \;\mid\; \Box\sigma \;\mid\; \Diamond\sigma \;\mid\; A$$

This syntax is then quotiented so that binary connectives $\langle\ ;\ \rangle$ and $\langle\ ,\ \rangle$ are associative, commutative, respectively admit $\langle\ \star\ \rangle$ and $\langle\ \circ\ \rangle$ as neutral elements, and so that $\star = \Box\star$ and $\circ = \Diamond\circ$. Connectives $\star$ and $\circ$ can conveniently be thought of and referred to as "true" and "false" respectively. Last, A ranges over linear logic formulas:

$$A \quad ::= \quad 1 \;\mid\; A \otimes A \;\mid\; \bot \;\mid\; A \parr A \;\mid\; !A \;\mid\; ?A \;\mid\; \alpha \;\mid\; \bar{\alpha}$$

where α ranges over some arbitrary set of base types.

Negation $\sigma^\perp$ of a structure σ is an involutive operation defined on linear logic formulas according to the usual de Morgan laws, and naturally extended to structures with:

$$\star^\perp = \circ \qquad (\sigma \,;\, \tau)^\perp = \sigma^\perp \,,\, \tau^\perp \qquad (\Box\sigma)^\perp = \Diamond(\sigma^\perp)$$

Our approach is similar in concept to other calculus of structures presentations [16, 1, 9, 10]. Among modifications introduced, a "computational layer" now appears underneath the "structural layer". This will allow for a direct correspondence with interaction nets systems.

Derivations. A derivation π from σ to τ, written $\pi : \sigma \to \tau$, is a sequence of structures whose first element is σ and last element is τ, such that every succession of two structures in this sequence is associated with a derivation rule. Basic derivation rules are split into three main categories.

- *Core structural rules*

$$\frac{\star}{\sigma^\perp \ , \ \sigma}\ \mathit{axiom} \qquad \frac{\sigma \ ; \ \sigma^\perp}{\circ}\ \mathit{cut} \qquad \frac{\omega \ ; \ (\sigma \ , \ \tau)}{(\omega \ ; \ \sigma) \ , \ \tau}\ \mathit{switch} \qquad \frac{\Box\sigma \ ; \ \Box\tau}{\Box(\sigma \ ; \ \tau)}\ \mathit{merge}$$

- *Administrative rules*

$$\frac{\Box\sigma}{\star}\ \mathit{erase}\uparrow \qquad \frac{\Box\sigma}{\Box\sigma \ ; \ \Box\sigma}\ \mathit{duplicate}\uparrow \qquad \frac{\Box\sigma}{\sigma}\ \mathit{open}\uparrow \qquad \frac{\Box\sigma}{\Box\Box\sigma}\ \mathit{nest}\uparrow$$

- *Computational rules* (we restrict here ourselves to the following set of rules which is sufficient to externalise linear logic)

$$\frac{\star}{1}\ \mathit{one}\downarrow \qquad \frac{\circ}{\bot}\ \mathit{bottom}\downarrow \qquad \frac{A \ ; \ B}{A \otimes B}\ \mathit{tensor}\downarrow \qquad \frac{A \ , \ B}{A \parr B}\ \mathit{par}\downarrow$$

$$\frac{\Box A}{!A}\ \mathit{of\text{-}course}\downarrow \qquad \frac{\Diamond A}{?A}\ \mathit{why\text{-}not}\downarrow$$

For reasons that follow, the following rule is admissible:

$$\frac{\Box(\sigma \ , \ \tau)}{\Box\sigma \ , \ \Diamond\tau}\ \mathit{select} \ := \ \cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\Box(\sigma \ , \ \tau)}{\Box(\sigma \ , \ \tau) \ ; \ (\Box\tau^\perp \ , \ \Diamond\tau)}\ \mathit{switch}}{(\Box(\sigma \ , \ \tau) \ ; \ \Box\tau^\perp) \ , \ \Diamond\tau}\ \mathit{merge}, -}{\Box((\sigma \ , \ \tau) \ ; \ \tau^\perp) \ , \ \Diamond\tau}\ \Box\mathit{switch}, -}{\Box(\sigma \ , \ (\tau \ ; \ \tau^\perp)) \ , \ \Diamond\tau}\ \Box(-, \mathit{cut}), -}{\Box\sigma \ , \ \Diamond\tau}$$

Deep inference. All basic deduction rules can be applied inside a structural context. If $\rho : \sigma \to \tau$ is a basic deduction rule and ν is a structural context (a structure with one hole), then $\nu[\rho] : \nu[\sigma] \to \nu[\tau]$ is also accepted as a deduction rule. For example, *tensor*$\downarrow$ can be applied inside disjunctive, conjunctive or exponential contexts, as well as any combination thereof, as follows:

$$\frac{\sigma \ , \ (A \ ; \ B)}{\sigma \ , \ A \otimes B}\ -, \mathit{tensor}\downarrow \qquad \frac{\sigma \ ; \ A \ ; \ B}{\sigma \ ; \ A \otimes B}\ -; \mathit{tensor}\downarrow \qquad \frac{\Box(A \ ; \ B)}{\Box(A \otimes B)}\ \Box\mathit{tensor}\downarrow$$

Symmetry. Besides, all rules can be turned upside-down: each rule $\rho : \sigma \to \tau$ is to be paired with a matching $\rho^\perp : \tau^\perp \to \sigma^\perp$ rule (calculus of structures implements contraposition natively). Core structural rules are paired with core structural rules, in particular *axiom* and *cut* are paired together and *switch* happens to be self-symmetric. Introduction rules (labeled with an arrow oriented downwards) are turned into elimination rules (labeled with an arrow oriented upwards) and vice versa. For example, symmetric variants of *tensor*$\downarrow$, *duplicate*$\uparrow$ and *merge* write as follows:

$$\frac{A \parr B}{A \ , \ B}\ \mathit{tensor}\uparrow \qquad \frac{\Diamond\sigma \ , \ \Diamond\sigma}{\Diamond\sigma}\ \mathit{duplicate}\downarrow \qquad \frac{\Diamond(\sigma \ , \ \tau)}{\Diamond\sigma \ , \ \Diamond\tau}\ \mathit{merge}$$

Given any structure ω, an empty sequence derives ω to ω itself. Such derivations are denoted by id_ω. Composition of two derivations $\pi_1 : \sigma \to \omega$ and $\pi_2 : \omega \to \tau$ is written $\pi_1 \cdot \pi_2 : \sigma \to \tau$. This defines a category whose objects are structures and whose morphisms are derivations.

Relaxed derivations. We quotient previously defined derivations by a few trivialities (and their symmetric variants, which are not listed), which are natural consequences of the quotient on structures.

$$\frac{\star}{\star \,,\, \circ}\,\mathit{axiom} \;\equiv\; \frac{\star}{\star}\,\mathit{id} \qquad \frac{\star \,;\, (\sigma \,,\, \tau)}{(\star \,;\, \sigma) \,,\, \tau}\,\mathit{switch} \;\equiv\; \frac{\sigma \,,\, \tau}{\sigma \,,\, \tau}\,\mathit{id} \qquad \frac{\Box\star \,;\, \Box\omega}{\Box(\star \,;\, \omega)}\,\mathit{merge} \;\equiv\; \frac{\Box\omega}{\Box\omega}\,\mathit{id}$$

$$\frac{\Box\star}{\star}\,\mathit{erase}\uparrow \;\equiv\; \frac{\Box\star}{\Box\star \,;\, \Box\star}\,\mathit{duplicate}\uparrow \;\equiv\; \frac{\Box\star}{\star}\,\mathit{open}\uparrow \;\equiv\; \frac{\Box\star}{\Box\Box\star}\,\mathit{nest}\uparrow \;\equiv\; \frac{\star}{\star}\,\mathit{id}$$

We moreover allow rules with disjoint scopes to freely pass each other. For example, given $\rho_1 : \sigma_1 \to \tau_1$ and $\rho_2 : \sigma_2 \to \tau_2$:

$$\frac{\dfrac{\sigma_1 \,;\, \sigma_2}{\tau_1 \,;\, \sigma_2}\,\rho_1 ; -}{\tau_1 \,;\, \tau_2}\,-\,;\rho_2 \;\equiv\; \frac{\dfrac{\sigma_1 \,;\, \sigma_2}{\sigma_1 \,;\, \tau_2}\,-\,;\rho_2}{\tau_1 \,;\, \tau_2}\,\rho_1 ; -$$

This, extended to arbitrary derivations $\pi_1 : \sigma_1 \to \tau_1$ and $\pi_2 : \sigma_2 \to \tau_2$, enables us to define contextual conjunction of derivations $\pi_1 \,;\, \pi_2 : \sigma_1 \,;\, \sigma_2 \to \tau_1 \,;\, \tau_2$, as well as disjunction $\pi_1 \,,\, \pi_2 : \sigma_1 \,,\, \sigma_2 \to \tau_1 \,,\, \tau_2$ unambiguously. Given $\pi : \sigma \to \tau$, we use similarly $\Box\pi : \Box\sigma \to \Box\tau$ and $\Diamond\pi : \Diamond\sigma \to \Diamond\tau$ to denote the use of a derivation π inside an exponential context.

Other equivalences occur when the scope of one rule is captured within one structural variable of an adjacent core structural rule. We provide the following, given an arbitrary rule $\rho : \sigma \to \tau$, as an illustration:

$$\frac{\dfrac{\omega \,;\, (\sigma \,,\, \omega')}{\omega \,;\, (\tau \,,\, \omega')}\,-\,;(\rho\,,-)}{(\omega \,;\, \tau) \,,\, \omega'}\,\mathit{switch} \;\equiv\; \frac{\dfrac{\omega \,;\, (\sigma \,,\, \omega')}{(\omega \,;\, \sigma) \,,\, \omega'}\,\mathit{switch}}{(\omega \,;\, \tau) \,,\, \omega'}\,(-\,;\rho)\,,-$$

Given that the use of structural variables in rules *axiom* and *cut* is also linear in some sense, similar commutations are considered. Those are however slightly less straightforward as they reverse orientation of rules.

$$\frac{\dfrac{\star}{\sigma \,,\, \sigma^\perp}\,\mathit{axiom}}{\tau \,,\, \sigma^\perp}\,\rho\,,-} \;\equiv\; \frac{\dfrac{\star}{\tau \,,\, \tau^\perp}\,\mathit{axiom}}{\tau \,,\, \sigma^\perp}\,-\,,\rho^\perp} \qquad \frac{\dfrac{\sigma \,;\, \tau^\perp}{\tau \,;\, \tau^\perp}\,\rho ; -}{\circ}\,\mathit{cut} \;\equiv\; \frac{\dfrac{\sigma \,;\, \tau^\perp}{\sigma \,;\, \sigma^\perp}\,-\,;\rho^\perp}{\circ}\,\mathit{cut}$$

In fact, all definitions and properties about derivations mentioned in the sequel are compatible with these equivalences. From now on, the term *derivation* will be used to refer to equivalence classes, which abstract away the irrelevant sequentiality found in concrete syntactic derivations.

An embedding of the sequent calculus. Any sequent calculus proof Π of formula A in MELL can be translated to a derivation $\pi : \star \to A$, which we consider a proof of formula A in calculus of structures. The basic idea is to combine all formulas found inside hypothesis or conclusion sequents using $\langle \;,\; \rangle$ connectives and combine hypothesis sequents themselves with $\langle \;;\; \rangle$ connectives. Inference rules from the one-sided sequent calculus are of shape (a):

$$(a)\quad \frac{\vdash A_1, \ldots, A_n \quad \cdots \quad \vdash B_1, \ldots, B_m}{\vdash C_1, \ldots, C_k} \qquad (b)\quad \frac{(A_1 \,,\, \cdots \,,\, A_n) \,;\, \cdots \,;\, (B_1 \,,\, \cdots \,,\, B_m)}{\begin{array}{c}\vdots\\ C_1 \,,\, \cdots \,,\, C_k\end{array}}$$

Every specific inference rule will be encoded as a derivation of shape (b). Since derivations associated to branches of an arborescent proof may be combined by means of structural contexts, the tree structure of sequent-calculus proofs is then easily embedded inside a sequential derivation.

The encoding of individual rules goes as follows (whenever Γ or Δ is used to denote a disjunction of formulas in sequents, γ or δ denotes the same disjunction in structures):

- *Multiplicatives*

$$\frac{\vdash \Gamma, A \quad \vdash B, \Delta}{\vdash \Gamma, A \otimes B, \Delta}\ tensor \quad \mapsto \quad \frac{\dfrac{(\gamma\ ,\ A)\ ;\ (B\ ,\ \delta)}{\gamma\ ,\ (A\ ;\ B)\ ,\ \delta}\ switch(\times 2)}{\gamma\ ,\ A \otimes B\ ,\ \delta}\ -, tensor\downarrow, - \qquad \frac{}{\vdash 1}\ one \quad \mapsto \quad \frac{\star}{1}\ one\downarrow$$

$$\frac{\vdash A, B, \Gamma}{\vdash A \parr B, \Gamma}\ par \quad \mapsto \quad \frac{A\ ,\ B\ ,\ \gamma}{A \parr B\ ,\ \gamma}\ par\downarrow, - \qquad \frac{\vdash \Gamma}{\vdash \bot, \Gamma}\ bottom \quad \mapsto \quad \frac{\gamma}{\bot\ ,\ \gamma}\ bottom\downarrow, -$$

- *Identities*

$$\frac{}{\vdash A, A^\perp}\ axiom \qquad \mapsto \qquad \frac{\star}{A\ ,\ A^\perp}\ axiom$$

$$\frac{\vdash A, \Gamma \quad \vdash A^\perp, \Delta}{\vdash \Gamma, \Delta}\ cut \qquad \mapsto \qquad \frac{\dfrac{(\gamma\ ,\ A)\ ;\ (A^\perp\ ,\ \delta)}{\gamma\ ,\ (A\ ;\ A^\perp)\ ,\ \delta}\ switch(\times 2)}{\gamma\ ,\ \delta}\ -, cut, -$$

- *Exponentials*

$$\frac{\vdash \Gamma}{\vdash ?A, \Gamma}\ weakening \qquad \mapsto \qquad \frac{\dfrac{\gamma}{\Diamond A\ ,\ \gamma}\ erase\downarrow, -}{?A\ ,\ \gamma}\ why\text{-}not\downarrow, -$$

$$\frac{\vdash ?A, ?A, \Gamma}{\vdash ?A, \Gamma}\ contraction \qquad \mapsto \qquad \frac{\dfrac{\dfrac{?A\ ,\ ?A\ ,\ \gamma}{\Diamond A\ ,\ \Diamond A\ ,\ \gamma}\ of\text{-}course\uparrow, of\text{-}course\uparrow, -}{\Diamond A\ ,\ \gamma}\ duplicate\downarrow, -}{?A\ ,\ \gamma}\ why\text{-}not\downarrow, -$$

$$\frac{\vdash A, \Gamma}{\vdash ?A, \Gamma}\ dereliction \quad \mapsto \quad \frac{\dfrac{A\ ,\ \gamma}{\Diamond A\ ,\ \gamma}\ open\downarrow, -}{?A\ ,\ \gamma}\ why\text{-}not\downarrow, - \qquad \frac{\vdash ??A, \Gamma}{\vdash ?A, \Gamma}\ digging \quad \mapsto \quad \frac{\dfrac{\dfrac{\dfrac{??A\ ,\ \gamma}{\Diamond ?A\ ,\ \gamma}\ of\text{-}course\uparrow, -}{\Diamond\Diamond A\ ,\ \gamma}\ \Diamond of\text{-}course\uparrow, -}{\Diamond A\ ,\ \gamma}\ nest\downarrow, -}{?A\ ,\ \gamma}\ why\text{-}not\downarrow, -$$

Last, we give an encoding of functorial promotion; any non-functorial promotion can as usual be encoded as a composition of one functorial promotion and diggings. A functorial promotion is encoded in one step together with the whole proof of its premise Π. Assuming (inductively) that $\Pi \mapsto \pi$, the promoted branch is encoded as follows:

$$\dfrac{\dfrac{}{\vdash A, B_1, \ldots, B_n}\,\Pi}{\vdash\, !A, ?B_1, \ldots, ?B_n}\,\textit{functorial-promotion} \quad\mapsto\quad \dfrac{\dfrac{\dfrac{\dfrac{\star}{\Box(A\ ,\ B_1\ ,\ \cdots\ ,\ B_n)}\,\Box\pi}{\Box A\ ,\ \Diamond(B_1\ ,\ \cdots\ ,\ B_n)}\,\textit{select}}{\Box A\ ,\ \Diamond B_1\ ,\ \cdots\ ,\ \Diamond B_n}\,-\,,\textit{merge}(\times n)}{!A\ ,\ ?B_1\ ,\ \cdots\ ,\ ?B_n}\,\textit{of-course}\downarrow, \textit{why-not}\downarrow, \cdots, \textit{why-not}\downarrow$$

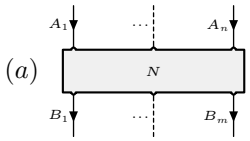
▶ **Lemma 2.1.** *A linear-logic formula A admits a proof Π in the standard sequent-calculus-style presentation of MELL if and only if there exists a derivation $\pi : \star \to A$ in the present calculus of structures.*

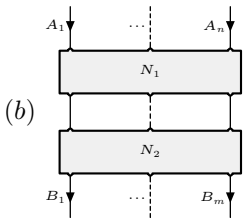
Proof. One direction follows directly from the provided encoding. For the other direction, one can reuse the (simple) inductive argument in [16]. ◀

Provability-wise this system is also equivalent to previous calculus of structure systems for linear logic which were studied by Straßburger [16].

3 The Underlying Computational Model

Translation from structures to interaction nets. Any derivation $\pi : \sigma \to \tau$ projects into a net, as shown in (a):

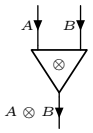




Input wires are typed with formulas $A_1, \ldots, A_n$ found in σ and output wires are typed with formulas $B_1, \ldots, B_m$ found in τ. The composition of any two derivations $\pi_1 : \sigma \to \omega$ and $\pi_2 : \omega \to \tau$, whose respective projections are assumed to be N_1 and N_2, projects as (b).

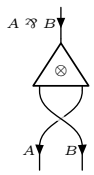
Computational rules are translated by single-cell nets, as described below. These rules may be used inside structural contexts, in which case one wire has to be added to their translations for every formula that appears in the context.

$$\dfrac{A\ ;\ B}{A \otimes B}\,\textit{tensor}\downarrow \quad\mapsto\quad$$



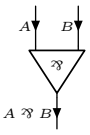
$$\dfrac{\star}{1}\,\textit{one}\downarrow \quad\mapsto\quad$$

$$\dfrac{A \parr B}{A\ ,\ B}\,\textit{tensor}\uparrow \quad\mapsto\quad$$



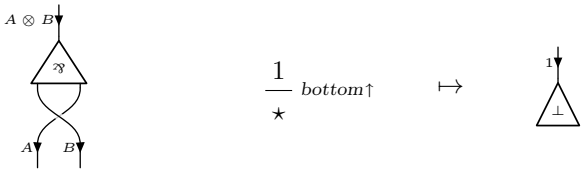
$$\dfrac{\bot}{\circ}\,\textit{one}\uparrow \quad\mapsto\quad$$

$$\dfrac{A\ ,\ B}{A \parr B}\,\textit{par}\downarrow \quad\mapsto\quad$$



$$\dfrac{\circ}{\bot}\,\textit{bottom}\downarrow \quad\mapsto\quad$$

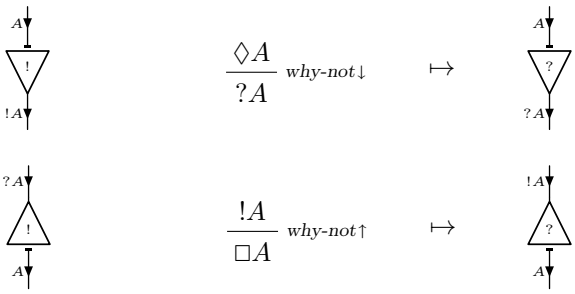
$$\frac{A \otimes B}{A \ ; \ B}\ par\uparrow \quad \mapsto \quad \frac{1}{\star}\ bottom\uparrow \quad \mapsto$$



We use a special notation for exponential cells, to distinguish them from standard dereliction (and co-dereliction) notation. A special marker is added to ports which were originally typed with a structural exponential connective.

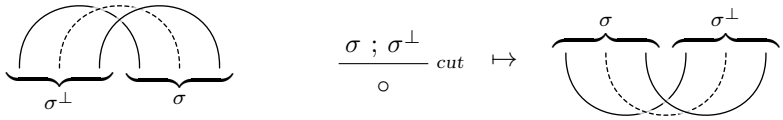
$$\frac{\Box A}{!A}\ of\text{-}course\downarrow \quad \mapsto \quad \frac{\Diamond A}{?A}\ why\text{-}not\downarrow \quad \mapsto$$

$$\frac{?A}{\Diamond A}\ of\text{-}course\uparrow \quad \mapsto \quad \frac{!A}{\Box A}\ why\text{-}not\uparrow \quad \mapsto$$



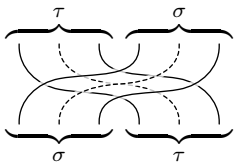
Rules from the identity fragment allow reversing the direction of wires:

$$\frac{\star}{\sigma^\perp \ , \ \sigma}\ axiom \quad \mapsto \quad \frac{\sigma \ ; \ \sigma^\perp}{\circ}\ cut \quad \mapsto$$

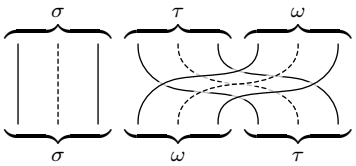


Other core structural rules are simple wirings that reflect implicit permutations of formulas. For example:

$$\frac{\tau \ ; \ \sigma}{\sigma \ ; \ \tau}\ id \quad \mapsto$$

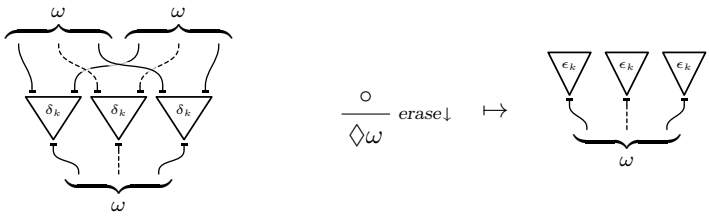


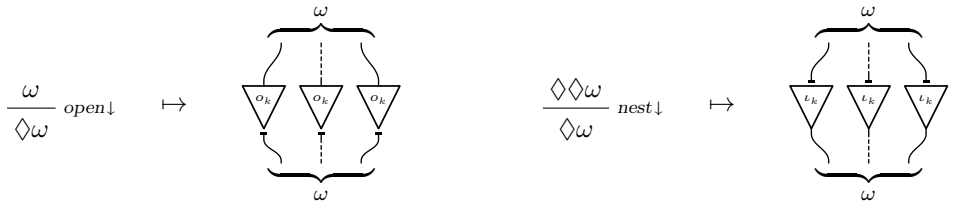
$$\frac{(\sigma \ , \ \tau) \ ; \ \omega}{(\sigma \ ; \ \omega) \ , \ \tau}\ switch \quad \mapsto$$



Only the translation of administrative rules *duplicate*, *erase*, *open* and *nest* requires particular cells, which are indexed by the number k of structural exponential connectives these derivation rules are applied beneath:

$$\frac{\Diamond\omega \ , \ \Diamond\omega}{\Diamond\omega}\ duplicate\downarrow \quad \mapsto \quad \frac{\circ}{\Diamond\omega}\ erase\downarrow \quad \mapsto$$





All structural connectives are lost during the projection. Formulas that appear inside structures can however be used to label wires in accordance with traditional "shallow" typing practices in interaction nets. Derivations appear as an "enriched" form of typing for nets and expose additional information about their structure.

A reduction for the interaction nets system we thus obtained could be defined directly and would be very similar to sharing graphs' reduction. The choice of an appropriate reduction rule when two δ-cells (which correspond to "fan" cells in sharing-graphs terminology) interact would rely on their assigned indexes, which are in fact the only piece of data extracted from the structure of nets that is really needed for computation purposes. But, more conveniently, all the necessary reduction rules can be enriched with structure and written within the deep-inference framework itself, as presented in Section 4. In particular, this automatically ensures the preservation of "enriched" type information after every reduction step.

Type systems for specialised interaction nets. Interaction nets from Example 1.1 can similarly be assigned a type system. This system is simple in the sense that concatenation of difference lists does not require duplication of data. Typing will therefore in this case not rely on structural exponential connectives. Adapting [11], it can be enriched with the following structure (where *dlist*, *list* and *int* are used as base types):

$$\frac{dlist}{\bar{list}\ ,\ list}\ unbd\uparrow \qquad \frac{dlist}{dlist\ ,\ d\bar{list}}\ d\text{-}cat\uparrow$$

$$\frac{\bar{list}\ ,\ list}{dlist}\ bind\downarrow \qquad \frac{int\ ;\ list}{list}\ cons\downarrow \qquad \frac{\star}{int}\ n\downarrow$$

The initial net from our example types as follows:

$$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\star}{(\bar{list}\ ,\ list)\ ;\ (\bar{list}\ ,\ list)}\ axiom;axiom}{(\bar{list}\ ,\ list)\ ;\ (\bar{list}\ ,\ list)}\ (-,cons\_2\downarrow);(-,cons\_4\downarrow)}{(\bar{list}\ ,\ list)\ ;\ (\bar{list}\ ,\ list)}\ (-,cons\_1\downarrow);(-,cons\_3\downarrow)}{dlist\ ;\ dlist}\ bind\downarrow;bind\downarrow}{(dlist\ ,\ d\bar{list})\ ;\ dlist}\ d\text{-}cat\uparrow;-}{dlist\ ,\ (d\bar{list}\ ;\ dlist)}\ switch}{dlist}\ -;cut$$

where:

$$\frac{list}{list}\ cons\_n\downarrow\ :=\ \cfrac{\cfrac{list}{int\ ;\ list}\ n\downarrow;-}{list}\ cons\downarrow$$

Projections of derivations built with the multiplicative structural fragment and the above-provided five rules are correctly built, meaning they are subnets of nets standardly considered as valid. Moreover, it can be shown that any valid net accepts such a derivation.

The fact that reduction preserves correctness of nets is automatically deduced from the fact that known reduction rules for this system of interaction nets can be enriched with

structure as well. For example, the first reduction rule types as follows:

$$\dfrac{\dfrac{\bar{list}\ ,\ list}{dlist}\ bind{\downarrow}}{dlist\ ,\ d\bar{list}}\ d\text{-}cat{\uparrow} \quad\longrightarrow\quad \dfrac{\dfrac{\dfrac{\bar{list}\ ,\ list}{(\bar{list}\ ,\ list)\ ;\ (\bar{list}\ ,\ list)}\ -;axiom}{\bar{list}\ ,\ list\ ,\ (list\ ;\ \bar{list})}\ switch(\times 2)}{dlist\ ,\ d\bar{list}}\ bind{\downarrow}, unbd{\downarrow}$$

4 Reduction within the Deep-Inference Formalism

Considering Curry–Howard correspondences between natural-deduction proof formalisms and several λ-calculus variants, it is natural to understand introduction rules as data constructions and elimination rules as data deconstructions; also to notice that computation arises from the interaction of the former with the latter.

Computation steps. Reduction can be defined in a similar fashion within our calculus by solving every potential interaction between an introduction (a downward-oriented rule) and a matching elimination (an upward-oriented rule) according to rewriting rules of this shape:

$$\dfrac{\dfrac{\sigma}{\omega}\downarrow}{\tau}\uparrow \quad\longrightarrow\quad \dfrac{\sigma}{\vdots}{\tau}$$

In particular, interfaces (hypothesis and conclusion) of derivations are preserved during reduction, which means that computation can be performed deep inside a structural context as well as inside a derivation context.

Our system for linear logic includes the following simple computation steps (symmetric variants are, and will always be, omitted):

$$\dfrac{\dfrac{\star}{1}\ one{\downarrow}}{\star}\ bottom{\uparrow} \quad\longrightarrow\quad \dfrac{\star}{\star}\ id \qquad \dfrac{\dfrac{A\ ;\ B}{A\otimes B}\ tensor{\downarrow}}{A\ ;\ B}\ par{\uparrow} \quad\longrightarrow\quad \dfrac{A\ ;\ B}{A\ ;\ B}\ id$$

$$\dfrac{\dfrac{\Box A}{!A}\ of\text{-}course{\downarrow}}{\Box A}\ why\text{-}not{\uparrow} \quad\longrightarrow\quad \dfrac{\Box A}{\Box A}\ id$$

Administrative reduction steps. In order to concretise any potential interaction between two computational rules, we will force introduction rules downwards and elimination rules upwards, relying on a few reduction rules which are described hereafter and which are to be considered additionally to commutations already assumed by the equivalence on derivations.

First, we explicit direct interactions of administrative deduction rules with *merge*:

$$\dfrac{\dfrac{\Box\sigma\ ;\ \Box\tau}{\Box(\sigma\ ;\ \tau)}\ merge}{\star}\ erase{\uparrow} \quad\longrightarrow\quad \dfrac{\Box\sigma\ ;\ \Box\tau}{\star}\ erase{\uparrow};erase{\uparrow}$$

$$\dfrac{\dfrac{\dfrac{\Box\sigma\ ;\ \Box\tau}{\Box(\sigma\ ;\ \tau)}\,merge}{\Box(\sigma\ ;\ \tau)\ ;\ \Box(\sigma\ ;\ \tau)}\,duplicate\uparrow}\quad\longrightarrow\quad\dfrac{\dfrac{\dfrac{\Box\sigma\ ;\ \Box\tau}{\Box\sigma\ ;\ \Box\sigma\ ;\ \Box\tau\ ;\ \Box\tau}\,id}{\Box\sigma\ ;\ \Box\tau\ ;\ \Box\sigma\ ;\ \Box\tau}\,duplicate\uparrow;duplicate\uparrow}{\Box(\sigma\ ;\ \tau)\ ;\ \Box(\sigma\ ;\ \tau)}\,merge;merge$$

$$\dfrac{\dfrac{\dfrac{\Box\sigma\ ;\ \Box\tau}{\Box(\sigma\ ;\ \tau)}\,merge}{\sigma\ ;\ \tau}\,open\uparrow}\quad\longrightarrow\quad\dfrac{\Box\sigma\ ;\ \Box\tau}{\sigma\ ;\ \tau}\,open\uparrow;open\uparrow$$

$$\dfrac{\dfrac{\dfrac{\Box\sigma\ ;\ \Box\tau}{\Box(\sigma\ ;\ \tau)}\,merge}{\Box\Box(\sigma\ ;\ \tau)}\,nest\uparrow}\quad\longrightarrow\quad\dfrac{\dfrac{\dfrac{\Box\sigma\ ;\ \Box\tau}{\Box\Box\sigma\ ;\ \Box\Box\tau}\,nest\uparrow;nest\uparrow}{\Box(\Box\sigma\ ;\ \Box\tau)}\,merge}{\Box\Box(\sigma\ ;\ \tau)}\,\Box merge$$

Remaining administrative steps take care of promoted content found under exponential contexts. We use the symbol $\check{\rho} : \sigma \to \tau$ to range over blocks of several structural rules whose projections as nets do not link together two inputs with a wire (for instance a single *cut* rule is excluded) or to range over a single introduction rule. Labels $\Box\check{\rho}$ used in left-hand sides of the following reduction steps denote any such deduction pattern used in a context that admits a box connective as top-level connective:

$$\dfrac{\dfrac{\Box\sigma}{\Box\tau}\,\Box\check{\rho}}{\star}\,erase\uparrow\quad\longrightarrow\quad\dfrac{\Box\sigma}{\star}\,erase\uparrow\qquad\dfrac{\dfrac{\Box\sigma}{\Box\tau}\,\Box\check{\rho}}{\Box\tau\ ;\ \Box\tau}\,duplicate\uparrow\quad\longrightarrow\quad\dfrac{\Box\sigma}{\Box\tau\ ;\ \Box\tau}\,\Box\check{\rho};\Box\check{\rho}\,duplicate\uparrow$$

$$\dfrac{\dfrac{\Box\sigma}{\Box\tau}\,\Box\check{\rho}}{\tau}\,open\uparrow\quad\longrightarrow\quad\dfrac{\dfrac{\Box\sigma}{\sigma}\,open\uparrow}{\tau}\,\check{\rho}\qquad\dfrac{\dfrac{\Box\sigma}{\Box\tau}\,\Box\check{\rho}}{\Box\Box\tau}\,nest\uparrow\quad\longrightarrow\quad\dfrac{\Box\sigma}{\Box\Box\sigma}\,nest\uparrow\,\Box\Box\check{\rho}$$

When $\sigma = \star$, the above reduction steps applied to blocks of core structural rules correspond, in the interaction-net formalism, to final steps of administrative reductions performed on nets that are purely made of wires.

Normal forms. A derivation in normal form is a derivation which can be written $\hat{\pi} \cdot \bar{\pi} \cdot \check{\pi}$, where $\hat{\pi}$, $\bar{\pi}$ and $\check{\pi}$ respectively contain eliminations, core structural rules and introductions only (all other configurations can be reduced).

▶ Remark. Although potential interaction between rules *axiom* and *cut* may be considered an artefact (because it leaves the underlying computation model unaffected, wirings are normal forms in interaction nets), we have reasons to believe they could be substituted with oriented variants *axiom*↓ and *cut*↑ to induce a call for further-simplified derivations. Those two rules were however left in the core structural fragment as we unfortunately do not know yet how to properly resolve those interactions when other structural rules interpose. For example, in the following simple case, reduction is possible given that both hand sides project to the same net:

$$\dfrac{\dfrac{\dfrac{\omega}{\omega\ ;\ (\omega^{\perp}\ ,\ \omega)}\,-;axiom\downarrow}{(\omega\ ;\ \omega^{\perp})\ ,\ \omega}\,switch}{\omega}\,cut\uparrow,-\quad\longrightarrow\quad\dfrac{\omega}{\omega}\,id$$

But generalisation seems difficult and it would require introduction of additional core structural rules, would we ever want to perform the reduction through local steps. Specifically, the following pairs (and their symmetric variants) cannot be commuted directly:

$$\dfrac{\dfrac{\omega \;;\; (\omega^{\perp} \;,\; \tau)}{(\omega \;;\; \omega^{\perp}) \;,\; \tau}\, switch}{\tau}\, cut\uparrow, - \qquad\qquad \dfrac{\dfrac{\Box\omega \;;\; \Box\omega^{\perp}}{\Box(\omega \;;\; \omega^{\perp})}\, merge}{\circ}\, \Box cut\uparrow$$

Daimons. For technical purposes, we introduce a new pair of basic deduction rules used to wrap up computations, together with an associated computation residual. Those may be understood computationally as an input/output mechanism.

$$\dfrac{\star}{\tau}\, daimon\downarrow \qquad\qquad \dfrac{\sigma}{\circ}\, daimon\uparrow \qquad\qquad \dfrac{\star}{\circ}\, done$$

They obviously break the consistency of the logic and are therefore not expected to be found in any acceptable proof. They are only used as a tool to prove normalisation of our system. Reduction-wise, daimons "collect" interacting rules as follows:

$$\dfrac{\dfrac{\sigma}{\tau}\,\check{\rho}}{\circ}\, daimon\uparrow \longrightarrow \dfrac{\sigma}{\circ}\, daimon\uparrow \qquad\qquad \dfrac{\dfrac{\star}{\omega}\, daimon\downarrow}{\circ}\, daimon\uparrow \longrightarrow \dfrac{\star}{\circ}\, done$$

Ultimately, given any derivation $\pi : \sigma \to \tau$, if the reduction process terminates, $daimon\downarrow \cdot \pi \cdot daimon\uparrow$ is expected to reduce to *done*, a normal form which marks the completion of the evaluation.

Strong confluence. Notice that in this system, redexes for administrative reductions may overlap with one another (for example, a single *duplication* or a *daimon* rule may have distinct interactions with each of several rules that commute), however, just like in the interaction-net formalism, peaks can be joined immediately and this system therefore enjoys the diamond property as well. In such a strong confluence setting, existence of a reduction path from a given derivation to a normal form (weak normalisation property) is equivalent to all rewriting paths from this derivation leading to a normal form (strong normalisation property). The normal form is moreover unique.

▶ **Lemma 4.1.** *Assuming $\pi : \star \to \omega$ admits π' as normal form:*

- $\Box\pi \cdot duplicate\uparrow$ *reduces to normal form* $\Box\pi' \;;\; \Box\pi'$
- $\Box\pi \cdot erase\uparrow$ *reduces to normal form* $id_{\star}$
- $\Box\pi \cdot open\uparrow$ *reduces to normal form* π'
- $\Box\pi \cdot nest\uparrow$ *reduces to normal form* $\Box\Box\pi'$

Proof. Inner reduction of π is followed by successive reduction steps on π' which, as a normal form, can be written $\bar{\pi} \cdot \check{\pi}$, where $\bar{\pi} : \star \to \sigma$ contains core structural rules only and $\check{\pi} : \sigma \to \omega$ contains introduction rules only. Introduction rules are handled inductively. The final reduction against $\bar{\pi}$ is an instance of an administrative reduction step applied to a block of core structural rules without hypothesis formulas. ◀

5 An Extensible Normalisation Proof via the Reducibility Technique

Let $\mathcal{O}$ be the set of normalisable derivations. A set of derivations from $\star$ to ω is called an *ω-initialiser set* and a set of derivations from ω to $\circ$ is called an *ω-finaliser set*. The *orthogonal* of a given ω-initialiser set $\mathcal{I}$ is the ω-finaliser set defined as follows:

$$\mathcal{I}^{\vee} \;:=\; \{\, \psi : \omega \to \circ \mid \forall \pi \in \mathcal{I},\, \pi \cdot \psi \in \mathcal{O} \,\}$$

Symmetrically, to any *ω-finaliser set* $\mathcal{F}$, we associate an orthogonal *ω-initialiser set* $\mathcal{F}^{\wedge}$:

$$\mathcal{F}^{\wedge} \;:=\; \{\, \phi : \star \to \omega \mid \forall \pi \in \mathcal{F},\, \phi \cdot \pi \in \mathcal{O} \,\}$$

These definitions come with the following properties:

$$\mathcal{I}^{\vee\wedge\vee} \;=\; \mathcal{I}^{\vee} \qquad\qquad \mathcal{F}^{\wedge\vee\wedge} \;=\; \mathcal{F}^{\wedge}$$
$$\mathcal{I} \;\subseteq\; \mathcal{I}^{\vee\wedge} \qquad\qquad \mathcal{F} \;\subseteq\; \mathcal{F}^{\wedge\vee}$$

By definition, an *ω-initialiser behaviour* is an ω-initialiser set $\mathcal{I}$ such that $\mathcal{I} = \mathcal{I}^{\vee\wedge}$, and an *$\omega$-finaliser behaviour* is an ω-finaliser set $\mathcal{F}$ such that $\mathcal{F} = \mathcal{F}^{\wedge\vee}$. *Initialiser candidates* $\lfloor \omega \rfloor$ and *finaliser candidates* $\lceil \omega \rceil$ are respectively ω-initialiser and ω-finaliser behaviours which are defined inductively, altogether, over ω. Hereafter, we provide inductive definition bodies for initialiser candidates in the case of top-level positive connectives.

- *Structural layers*

$$\lfloor \star \rfloor \;:=\; \{\, \mathit{daimon}{\downarrow} \,\}^{\vee\wedge} \qquad \lfloor \sigma_1 \,;\, \sigma_2 \rfloor \;:=\; \{\, \pi_1 \,;\, \pi_2 \mid \pi_1 \in \lfloor \sigma_1 \rfloor, \pi_2 \in \lfloor \sigma_2 \rfloor \,\}^{\vee\wedge}$$
$$\lfloor \Box\sigma \rfloor \;:=\; \{\, \Box\pi \mid \pi \in \lfloor \sigma \rfloor \,\}^{\vee\wedge}$$

- *Computational layers*

$$\lfloor 1 \rfloor \;:=\; \{\, \pi \cdot \mathit{one}{\downarrow} \mid \pi \in \lfloor \star \rfloor \,\}^{\vee\wedge} \qquad \lfloor A \otimes B \rfloor \;:=\; \{\, \pi \cdot \mathit{tensor}{\downarrow} \mid \pi \in \lfloor A \,;\, B \rfloor \,\}^{\vee\wedge}$$
$$\lfloor !A \rfloor \;:=\; \{\, \pi \cdot \mathit{of\text{-}course}{\downarrow} \mid \pi \in \lfloor A \rfloor \,\}^{\vee\wedge}$$

- *Base types*

$$\lfloor \alpha \rfloor \;:=\; \{\, \mathit{daimon}{\downarrow} \,\}^{\vee\wedge}$$

Finaliser candidates definitions for negative connectives (and base types) are handled symmetrically. Initialiser candidates for negative connectives and finaliser candidates for positive connectives are then defined as orthogonals to the former, so that $\lfloor \omega \rfloor = \lceil \omega \rceil^{\wedge}$ and $\lceil \omega \rceil = \lfloor \omega \rfloor^{\vee}$ globally holds. This inductive definition is well founded.

▶ **Definition 5.1.** A derivation $\pi : \sigma \to \tau$ is said to be *reducible* when:

$$\forall \phi \in \lfloor \sigma \rfloor,\, \forall \psi \in \lceil \tau \rceil,\, \phi \cdot \pi \cdot \psi \in \mathcal{O}$$

Notice that equivalently, $\pi : \sigma \to \tau$ is reducible iff $\forall \phi \in \lfloor \sigma \rfloor$, $\phi \cdot \pi \in \lfloor \tau \rfloor$, or, symmetrically, iff $\forall \psi \in \lceil \tau \rceil$, $\pi \cdot \psi \in \lceil \sigma \rceil$.

▶ **Lemma 5.2.** *For any structure ω, candidates $\lceil \omega \rceil$ and $\lfloor \omega \rfloor$ contain normalisable nets only, and moreover $\mathit{daimon}{\uparrow} \in \lceil \omega \rceil$ and $\mathit{daimon}{\downarrow} \in \lfloor \omega \rfloor$.*

Proof. Valid at base types, this combination of statements propagates to all structures by induction. ◀

▶ **Lemma 5.3.** *Every derivation is reducible.*

Proof. By induction on the derivation; we address all derivation constructions separately. The definition of reducibility being symmetric, the number of cases to consider is reduced.

Identity. Given $\phi \in \lfloor \sigma \rfloor$ and $\psi \in \lceil \sigma \rceil$, the derivation $\phi \cdot id_\sigma \cdot \psi = \phi \cdot \psi$ normalises as a composition of two derivations which belong to dual candidates.

Composition. Given two reducible derivations $\pi_1 : \sigma \to \omega$ and $\pi_2 : \omega \to \tau$, using properties $\forall \phi \in \lfloor \sigma \rfloor, \phi \cdot \pi_1 \in \lfloor \omega \rfloor$ and $\forall \psi \in \lceil \tau \rceil, \pi_2 \cdot \psi \in \lceil \omega \rceil$, we obtain that any derivation $\phi \cdot \pi_1 \cdot \pi_2 \cdot \psi$ such that $\phi \in \lfloor \sigma \rfloor$ and $\psi \in \lceil \tau \rceil$ normalises. We have thus shown reducibility of $\pi_1 \cdot \pi_2$.

Contexts

- *Conjunctions.* Given reducible derivations $\pi_1 : \sigma_1 \to \tau_1$ and $\pi_2 : \sigma_2 \to \tau_2$, we show that $\pi_1 \,;\, \pi_2 : \sigma_1 \,;\, \sigma_2 \to \tau_1 \,;\, \tau_2$ is reducible, written as follows: for all $\psi \in \lceil \tau_1 \,;\, \tau_2 \rceil$ we have $(\pi_1 \,;\, \pi_2) \cdot \psi \in \lceil \sigma_1 \,;\, \sigma_2 \rceil$. Given that $\lceil \sigma_1 \,;\, \sigma_2 \rceil = \{\, \pi_1 \,;\, \pi_2 \mid \pi_1 \in \lfloor \sigma_1 \rfloor, \pi_2 \in \lfloor \sigma_2 \rfloor \,\}^\vee$, according to orthogonality's definition, this is equivalent to showing that for all $\phi_1 \in \lfloor \sigma_1 \rfloor$ and $\phi_2 \in \lfloor \sigma_2 \rfloor$ we have $(\phi_1 \,;\, \phi_2) \cdot (\pi_1 \,;\, \pi_2) = (\phi_1 \cdot \pi_1) \,;\, (\phi_2 \cdot \pi_2) \in \lfloor \tau_1 \,;\, \tau_2 \rfloor$. This holds by definition since by reducibility of π_1 and π_2 we have $\phi_1 \cdot \pi_1 \in \lfloor \tau_1 \rfloor$ and $\phi_2 \cdot \pi_2 \in \lfloor \tau_2 \rfloor$.
- *Box.* Given a reducible derivation $\pi : \sigma \to \tau$, we show that $\Box\pi : \Box\sigma \to \Box\tau$ is reducible, written as follows: for all $\psi \in \lceil \Box\tau \rceil$ we have $\Box\pi \cdot \psi \in \lceil \Box\sigma \rceil$. Given that $\lceil \Box\sigma \rceil = \{\, \Box\pi \mid \pi \in \lfloor \sigma \rfloor \,\}^\vee$, this is equivalent to showing that for all $\phi_0 \in \lfloor \sigma \rfloor$, we have $\Box\phi_0 \cdot \Box\pi = \Box(\phi_0 \cdot \pi) \in \lfloor \Box\tau \rfloor$. This holds by definition since $\phi_0 \cdot \pi \in \lfloor \tau \rfloor$, by reducibility of π.

Core structural rules

- *Axiom.* We show for all $\phi \in \lfloor \star \rfloor$ that $\phi \cdot axiom \in \lfloor \sigma^\perp \,,\, \sigma \rfloor$, or equivalently that derivation $axiom \cdot (\psi_1^\perp \,,\, \psi_2)$ normalises for all $\psi_1 \in \lfloor \sigma \rfloor$ and $\psi_2 \in \lceil \sigma \rceil$. This derivation equivalently writes $\psi_1 \cdot \psi_2$, which normalises.
- *Switch.* We show for all $\phi \in \lfloor \omega \,;\, (\sigma \,,\, \tau) \rfloor$ that $\phi \cdot switch \in \lfloor (\omega \,;\, \sigma) \,,\, \tau \rfloor$, or equivalently that $switch \cdot (\psi_1 \,,\, \psi_2) \in \lceil \omega \,;\, (\sigma \,,\, \tau) \rceil$ for all $\psi_1 \in \lceil \omega \,;\, \sigma \rceil$ and $\psi_2 \in \lceil \tau \rceil$, which in turn is equivalent to showing that $(\phi_1 \,;\, \phi_2) \cdot switch \cdot (\psi_1 \,,\, \psi_2)$ normalises for all $\phi_1 \in \lfloor \omega \rfloor$, $\phi_2 \in \lfloor \sigma \,,\, \tau \rfloor$, $\psi_1 \in \lceil \omega \,;\, \sigma \rceil$ and $\psi_2 \in \lceil \tau \rceil$. Given that $\phi_1 : \star \to \omega$ and $\psi_2 : \tau \to \circ$, this last derivation equivalently writes $\pi_1 \cdot \pi_2$ where $\pi_1 = \phi_2 \cdot (id_\sigma \,,\, \psi_2)$ and $\pi_2 = (\phi_1 \,;\, id_\sigma) \cdot \psi_1$; it normalises because $\pi_1 \in \lfloor \sigma \rfloor$ and $\pi_2 \in \lceil \sigma \rceil$.
- *Merge.* We show for all $\psi \in \lceil \Box(\sigma \,;\, \tau) \rceil$ that $merge \cdot \psi \in \lceil \Box\sigma \,;\, \Box\tau \rceil$, or equivalently that:
  - $(\phi_1 \,;\, \phi_2) \cdot merge \cdot \psi$ normalises for all $\phi_1 \in \lfloor \Box\sigma \rfloor$, $\phi_2 \in \lfloor \Box\tau \rfloor$, $\psi \in \lceil \Box(\sigma \,;\, \tau) \rceil$
  - $(\phi_1 \,;\, id_{\Box\tau}) \cdot merge \cdot \psi \in \lceil \Box\tau \rceil$ for all $\phi_1 \in \lfloor \Box\sigma \rfloor$, $\psi \in \lceil \Box(\sigma \,;\, \tau) \rceil$
  - $(\phi_1 \,;\, \Box\phi_2') \cdot merge \cdot \psi$ normalises for all $\phi_1 \in \lfloor \Box\sigma \rfloor$, $\phi_2' \in \lfloor \tau \rfloor$, $\psi \in \lceil \Box(\sigma \,;\, \tau) \rceil$
  - $(id_{\Box\sigma} \,;\, \Box\phi_2') \cdot merge \cdot \psi \in \lceil \Box\sigma \rceil$ for all $\phi_2' \in \lfloor \tau \rfloor$, $\psi \in \lceil \Box(\sigma \,;\, \tau) \rceil$
  - $(\Box\phi_1' \,;\, \Box\phi_2') \cdot merge \cdot \psi$ normalises for all $\phi_1' \in \lfloor \sigma \rfloor$, $\phi_2' \in \lfloor \tau \rfloor$, $\psi \in \lceil \Box(\sigma \,;\, \tau) \rceil$.

  The last derivation writes $\Box(\phi_1' \,;\, \phi_2') \cdot \psi$ and normalises since $\Box(\phi_1' \,;\, \phi_2') \in \lfloor \Box(\sigma \,;\, \tau) \rfloor$.

Administrative rules

- *Duplicate.* We show for all $\psi \in \lceil \Box\sigma \,;\, \Box\sigma \rceil$ that $duplicate{\uparrow} \cdot \psi \in \lceil \Box\sigma \rceil$, or equivalently that $\Box\phi_0 \cdot duplicate{\uparrow} \cdot \psi$ normalises for all $\phi_0 \in \lfloor \sigma \rfloor$. By Lemma 4.1, this derivation reduces to $(\Box\phi_0 \,;\, \Box\phi_0) \cdot \psi$, which normalises since $\Box\phi_0 \,;\, \Box\phi_0 \in \lfloor \Box\sigma \,;\, \Box\sigma \rfloor$.
- *Erase, Open, Nest.* Similar to the *Duplicate* case.

Computational rules

- *Tensor.* We obtain $\phi \cdot tensor{\downarrow} \in \lfloor A \otimes B \rfloor$ for all $\phi \in \lfloor A \; ; \; B \rfloor$ directly from the candidate definition. This shows reducibility of $tensor{\downarrow}$ derivations.
- *One*, *Of-course.* Similar to the *Tensor* case.
- *Par.* Reducibility of $par{\downarrow}$ holds iff for all $\phi \in \lfloor A \; , \; B \rfloor$, we have $\phi \cdot par{\downarrow} \in \lfloor A \parr B \rfloor$. Unfolding the candidate definition, it suffices to show that $\phi \cdot par{\downarrow} \cdot tensor{\uparrow} \cdot \psi$ normalises for all $\psi \in \lceil A \; , \; B \rceil$. This holds because reduced derivations $\phi \cdot \psi$ normalise.
- *Bottom*, *Why-not.* Similar to the *Par* case. ◀

▶ **Theorem 5.4.** *All derivations normalise.*

Proof. By the previous lemma, any derivation $\pi : \sigma \to \tau$ is reducible, hence $daimon{\downarrow} \cdot \pi \cdot daimon{\uparrow}$ normalises and so does π. ◀

Discussion about modularity and possible extensions. In the present development, we handled base types as purely abstract types which cannot be introduced or eliminated, these could only be passed around. In practice, we may want to use base types to represent primitive types which come together with associated primitive operations. Our developments could be extended to take those into account by adding related data constructions to respective candidate definitions. Normalisation remains of course ultimately dependent on primitives' behaviour, since associated inductive cases appear in the proof. Base types can also implement type variables used in second-order quantifiers (the reader is referred to the original proof by Girard [4] as for how to handle these).

6 Conclusion

In this paper we have introduced a Curry–Howard correspondence between well-formed interaction nets and a deep-inference deduction system based on multiplicative-exponential linear logic. Linear logic itself can be expressed in the system and its computational aspects materialise though the correspondence as a system of sharing graphs. The enriched type system for nets that stems from this correspondence not only sheds additional light upon the structure of multiplicatives [2], but moreover encompasses the exponential layer and could easily be extended further.

Our approach fuses the essential properties of strong confluence (interaction nets) and weak normalisation (via Girard's reducibility method) to obtain a consice, modular and extensible proof of strong normalisation. However, it currently relies on a somewhat unorthodox notion of normal form, which does not consider interactions between identity rules. As the sought computational interpretation is unaffected this seems to be negligible, but we will study this peculiarity in more detail in future work.

Furthermore, we are interested in extending the method to stronger logics: additive, second-order, inductive constructions, etc. Extensions towards differential linear logic, which was introduced by Ehrhard and Regnier and features further symmetry [3], will also be investigated in these studies.

References

1 Kai Brünnler. Deep inference and its normal form of derivations. In *CiE*, volume 3988 of *LNCS*, pages 65–74, 2006.

2 Vincent Danos and Laurent Regnier. The structure of multiplicatives. *Arch. Math. Logic*, 28:181–203, 1989.

3 Thomas Ehrhard and Laurent Regnier. Differential interaction nets. *Theor. Comput. Sci.*, 364(2):166–195, 2006.

4 Jean-Yves Girard. Linear logic. *Theor. Comput. Sci.*, 50:1–102, 1987.

5 G. Gonthier, M. Abadi, and J.-J. Lévy. The geometry of optimal lambda reduction. In *Proc. 19th POPL*, pages 15–26. ACM Press, 1992.

6 Georges Gonthier, Martín Abadi, and Jean-Jacques Lévy. Linear logic without boxes. In *Proc. 7th LICS*, pages 223–34, 1992.

7 S. Guerrini, S. Martini, and A. Masini. Coherence for sharing proof-nets. *Theor. Comput. Sci.*, 294(3):379–409, 2003.

8 A. Guglielmi. A system of interaction and structure. *ACM Trans. Comput. Log.*, 8(1), 2007.

9 A. Guglielmi and L. Straßburger. Non-commutativity and MELL in the calculus of structures. In *Proc. 10th CSL*, volume 2142 of *LNCS*, pages 54–68, 2001.

10 T. Gundersen, W. Heijltjes, and M. Parigo. Atomic lambda calculus – a typed lambda calculus with explicit sharing. In *Proc. 28th LICS*, 2013. To appear.

11 Yves Lafont. Interaction nets. *Proc. 17th POPL*, pages 95–108, 1990.

12 J. Lamping. An algorithm for optimal lambda calculus reduction. In *Proc. 7th POPL*, pages 16–30. ACM Press, 1990.

13 I. Mackie. Interaction nets for linear logic. *Theor. Comput. Sci.*, 247(1-2):83–140, 2000.

14 Michele Pagani and Lorenzo Tortora de Falco. Strong normalization property for second order linear logic. *Theor. Comput. Sci.*, 411(2):410–444, 2010.

15 Lutz Straßburger and Alessio Guglielmi. A system of interaction and structure IV: The exponentials and decomposition. *ACM Trans. Comput. Log.*, 12(4):23, 2011.

16 Lutz Straßburger. MELL in the calculus of structures. *Theor. Comput. Sci.*, 309:213–285, 2003.

17 A. Tiu. A local system for intuitionistic logic. In *Proc. 13th LPAR*, volume 4246 of *LNCS*, pages 242–256, 2006.

18 C.P. Wadsworth. *Semantics and Pragmatics of the Lambda-Calculus*. PhD thesis, University of Oxford, 1971.