Introduction to linear logic and ludics, part II

Pierre-Louis Curien

Introduction to linear logic and ludics, Part II

Pierre-Louis Curien (CNRS & Université Paris VII)*

January 19, 2005

Abstract

This paper is the second part of an introduction to linear logic and ludics, both due to Girard. It is devoted to proof nets, in the limited, yet central, framework of multiplicative linear logic (section 1) and to ludics, which has been recently developped in an aim of further unveiling the fundamental interactive nature of computation and logic (sections 2, 3, 4, and 5). We hope to offer a few computer science insights into this new theory.

Keywords: Cut-elimination (03F05), Linear logic (03F52), Logic in computer science (03B70), Interactive modes of computation (68Q10), Semantics (68Q55), Programming languages (68N15).

Prerequisites

This part depends mostly on the first two sections of part I, and should therefore be accessible to any one who has been exposed to the very first steps of linear logic. We have used the following sources: [29, 23, 38] for proof nets, and and [35, 36] for ludics.

1 Proof nets

Proof nets are graphs that give a more economical presentation of proofs, abstracting from some irrelevant order of application of the rules of linear logic given in sequent calculus style (sections 2 and 3 of part I). We limit ourselves here to multiplicative linear logic (MLL), and we shall even begin with cut-free MLL. Proof nets for MLL are graphs which are "almost trees", and we stress this by our choice of presentation. We invite the reader to draw proof nets by himself while reading the section.

What plays here the role of a proof of a sequent $\vdash A_1, \ldots, A_n$ is the forest of the formulas $A_1, \ldots, A_n$ represented as trees, together with a partition of the leaves of the forest in pairs (corresponding to the application of an axiom $\vdash C, C^\perp$). Well, not quite. A proof may not decompose each formula completely, since an axiom $\vdash C, C^\perp$ can be applied to *any* formula. Hence we have to specify *partial* trees for the formulas $A_1, \ldots, A_n$. One way to do this is to specify the set of leaves of each partial tree as a set of (pairwise disjoint) occurrences. Formally, an occurrence is a word over $\{1, 2\}$, and the subformula A/u at occurrence u is defined by the following formal system:

$$A/\epsilon = A \qquad (A_1 \otimes A_2)/1u = (A_1 ⅋ A_2)/1u = A_1/u \qquad (A_1 \otimes A_2)/2u = (A_1 ⅋ A_2)/2u = A_2/u\ .$$

*Laboratoire *Preuves, Programmes et Systèmes*, Case 7014, 2 place Jussieu, 75251 Paris Cedex 05, France, curien@pps.jussieu.fr.

A *partial formula tree* A^U consists of a formula A together with a set U of pairwise disjoint occurrences such that $(\forall u \in U \ \ A/u$ is defined). Recall that a partition of a set E is a set X of non-empty subsets of E which are pairwise disjoint and whose union is E. If $X = \{Y_1, \ldots, Y_n\}$ we often simply write $X = Y_1, \ldots, Y_n$.

Definition 1.1 *A* proof structure *is given by*

$$\{A_1^{U_1}, \ldots, A_n^{U_n}\}[X] ,$$

where $\{A_1^{U_1}, \ldots, A_n^{U_n}\}$ *is a multiset of partial formula trees and where* X *is a partition of* $\{A_1/u \mid u \in U_1\} \cup \ldots \cup \{A_n/u \mid u \in U_n\}$ *(disjoint union) whose classes are pairs of dual formulas. We shall say that each class of the partition is an axiom of the proof structure, and that* $A_1, \ldots, A_n$ *are the conclusions of the proof structure.*

More generally, we shall manipulate graphs described as a forest plus a partition of its leaves, without any particular requirement on the partition. This notion is faithful to Girard's idea of paraproof, discussed in the next section.

Definition 1.2 *A* paraproof structure *is given by* $\{A_1^{U_1}, \ldots, A_n^{U_n}\}[X]$*, where* X *is a partition of* $\{A_1/u \mid u \in U_1\} \cup \ldots \cup \{A_n/u \mid u \in U_n\}$ *(disjoint union) . We shall say that each class of the partition is a generalized axiom , or* daimon *(anticipating on a terminology introduced in the following section), and that* $A_1, \ldots, A_n$ *are the conclusions of the paraproof structure.*

The actual graph associated to this description is obtained by:

- drawing the trees of the formulas $A_1, \ldots, A_n$ stopping at $U_1, \ldots, U_n$, respectively (i.e., there is a node corresponding to each subformula A_i/u, where $u < u_i$ for some $u_i \in U_i$); and
- associating a new node with each class of the partition and new edges from the new node to each of the leaves of the class (in the case of proof structures, one can dispense with the new node, and draw an edge between the matching dual formulas – such edges are usually drawn horizontally).

We next describe how to associate a proof structure with a proof. The following definition follows the rules of MLL.

Definition 1.3 *The* sequentializable *proof structures for (cut-free) MLL are the proof structures obtained by the following rules:*

$$\frac{}{\{C^{\{\epsilon\}}, (C^\perp)^{\{\epsilon\}}\}[\{\{C, C^\perp\}\}]} \qquad \frac{\{A_1^{U_1}, \ldots, A_n^{U_n}, B^V, C^W\}[X]}{\{A_1^{U_1}, \ldots, A_n^{U_n}, (B \parr C)^U\}[X]}$$

$$\frac{\{A_1^{U_1}, \ldots, A_n^{U_n}, B^V\}[X] \quad \{{A'}_1^{U'_1}, \ldots, {A'}_{n'}^{U'_{n'}}, C^W\}[Y]}{\{A_1^{U_1}, \ldots, A_n^{U_n}, {A'}_1^{U'_1}, \ldots, {A'}_{n'}^{U'_{n'}}, (B \otimes C)^U\}[X \cup Y]}$$

where $U = \{1v \mid v \in V\} \cup \{2w \mid w \in W\}$*. Sequentializable proof structures are called* proof nets.

It should be clear that there is a bijective correspondence between MLL proofs and the proofs of sequentialization. Let us check one direction. We show that if a proof structure with conclusions $A_1, \ldots, A_n$ is sequentializable, then a proof that it is so yields an MLL proof of the sequent $\vdash A_1, \ldots, A_n$. This is easily seen by induction: the proof associated with $\{C^{\{\epsilon\}}, (C^\perp)^{\{\epsilon\}}\}[\{\{C, C^\perp\}\}]$ is the axiom $\vdash C, C^\perp$, while the last steps of the proofs associated with $\{A_1^{U_1}, \ldots, A_n^{U_n}, (B \parr C)^U\}[X]$ and $\{A_1^{U_1}, \ldots, A_n^{U_n}, {A'}_1^{U'_1}, \ldots, {A'}_n^{U'_n}, (B \otimes C)^U\}[X]$ are, respectively:

$$\frac{\vdash A_1, \ldots, A_n, B, C}{\vdash A_1, \ldots, A_n, B \parr C} \qquad \frac{\vdash A_1, \ldots, A_n, B \quad \vdash A'_1, \ldots A'_{n'}, C}{\vdash A_1, \ldots, A_n, A'_1, \ldots, A'_{n'}, B \otimes C}$$

The question we shall address in the rest of the section is the following: given a proof structure, when is it the case that it is sequentializable? It turns out that the right level of generality for this question is to lift it to paraproof nets, which are defined next.

Definition 1.4 *The* sequentializable *paraproof structures for (cut-free) MLL are the paraproof structures obtained by the following rules:*

$$\frac{}{\{A_1^{\{\epsilon\}}, \ldots, A_n^{\{\epsilon\}}\}[\{\{A_1, \ldots, A_n\}\}]} \qquad \frac{\{A_1^{U_1}, \ldots, A_n^{U_n}, B^V, C^W\}[X]}{\{A_1^{U_1}, \ldots, A_n^{U_n}, (B \parr C)^U\}[X]}$$

$$\frac{\{A_1^{U_1}, \ldots, A_n^{U_n}, B^V\}[X] \quad \{{A'}_1^{U'_1}, \ldots, {A'}_{n'}^{U'_{n'}}, C^W\}[Y]}{\{A_1^{U_1}, \ldots, A_n^{U_n}, {A'}_1^{U'_1}, \ldots, {A'}_{n'}^{U'_{n'}}, (B \otimes C)^U\}[X \cup Y]}$$

where $U = \{1v \mid v \in V\} \cup \{2w \mid w \in W\}$. Sequentializable paraproof structures are called paraproof nets.

What is the proof-theoretic meaning of a paraproof net? Well, the above bijective correspondence extends to a correspondence between sequentialization proofs of paraproof structures and MLL *paraproofs*, which are defined by the following rules:

$$\frac{}{\vdash \Gamma} \qquad \frac{\vdash \Gamma_1, B \quad \vdash \Gamma_2, C}{\vdash \Gamma_1, \Gamma_2, B \otimes C} \qquad \frac{\vdash \Gamma, B, C}{\vdash \Gamma, B \parr C}$$

where in the first rule Γ is an arbitrary multiset of formulas. The first rule is called *generalized axiom*, or *daimon* rule. Starting in a proof search mode from an MLL formula A one may build absolutely freely a paraproof of A, making arbitrary decisions when splitting the context in a $\otimes$ rule. Any choice is as good as another, in the sense that the process will be successful at the end, i.e., we shall eventually reach generalized axioms. There is an interesting subset of paraproofs, which we call the *extreme* ones. An extreme paraproof is a paraproof in which in each application of the $\otimes$ rule we have $\Gamma_1 = \emptyset$ or $\Gamma_2 = \emptyset$. The following definition formalizes this notion.

Definition 1.5 *The* extreme *paraproof nets for (cut-free) MLL are the proof structures obtained by the following rules:*

$$\frac{}{\{A_1^{\{\epsilon\}},\ldots,A_n^{\{\epsilon\}}\}[\{\{A_1,\ldots,A_n\}\}]} \qquad \frac{\{A_1^{U_1},\ldots,A_n^{U_n},B^V,C^W\}[X]}{\{A_1^{U_1},\ldots,A_n^{U_n},(B⅋C)^U\}[X]}$$

$$\frac{\{A_1^{U_1},\ldots,A_n^{U_n},B^V\}[X] \quad \{C^W\}[Y]}{\{A_1^{U_1},\ldots,A_n^{U_n},(B\otimes C)^U\}[X\cup Y]} \qquad \frac{\{B^V\}[X] \quad \{A_1^{U_1},\ldots,A_n^{U_n},C^W\}[Y]}{\{A_1^{U_1},\ldots,A_n^{U_n},(B\otimes C)^U\}[X\cup Y]}$$

where $U=\{1v \mid v\in V\}\cup\{2w\mid w\in W\}$.

Extreme paraproofs will be soon put to use, but for the time being we are concerned with general paraproofs. Our first tool is the notion of switching. Let S be a paraproof stucture. A switching is a function from the set of all internal nodes of (the forest part of) S of the form $B_1⅋B_2$, to $\{L,R\}$. A switching induces a *correction* (sub)graph, defined as follows. At each internal node $B_1⅋B_2$, we cut exactly one of the two edges linking $B_1⅋B_2$ to its immediate subformulas, namely the edge between $B_1⅋B_2$ and B_2 if the switching is set to L, and the edge between $B_1⅋B_2$ and B_1 if the switching is set to R. The following definition is due to Danos and Regnier [23], and arose as a simplification of the original criterion proposed by Girard [29].

Definition 1.6 (DR) *We say that a paraproof stucture S satisfies the* DR-criterion *if all the correction graphs induced by a switching of S are connected and acyclic (that is, are trees).*

Proposition 1.7 *All sequentializable paraproof structures satisfy the DR-criterion. Moreover, in each correction graph, when a switch corresponding to a formula $B_1⅋B_2$ is, say, on the left, then the path from B_1 to B_2 does not go through $B_1⅋B_2$.*

Proof. We proceed by induction on the definition of paraproof nets. If N is a generalized axiom, then there is just one switching (the empty one) and the associated correction graph is the graph itself, which is obviously a tree. If N is obtained from N_1 and N_2 by a $\otimes$ rule acting on a conclusion B of N_1 and a conclusion C of N_2, then a switching of N is a pair of a switching of N_1 and a switching of N_2. We know by induction that the corresponding two correction graphs are trees. We can thus organize them in a tree form, with roots B, C respectively. Then the correction graph for N is obtained by adding a new root and edges between the new root and B and C, respectively: this is obviously a tree. Finally, suppose that N is obtained from N_1 by a ⅋ rule acting on two conclusions B and C of N_1, and consider a switching for N, which assigns, say L, to the new ⅋ node. The rest of the switching determines a correction graph for N_1 which is a tree by induction. We can organize the correction graph for N_1 in such a way that B is the root. Then the correction graph for N is obtained by adding a new root and an edge between the new root and B, and this is again obviously a tree. The second property of the statement is obvious to check, by induction on the sequentialization proof too. □

Our next tool is a parsing procedure, which takes a paraproof structure and progressively shrinks it, or contracts it. If the procedure is not blocked, then the paraproof structure is a paraproof net. This criterion was first discovered by Danos [21]. Guerrini explained the criterion as a successful parsing procedure [37]. Here we (straightforwardly) extend the procedure from proof structures to paraproof structures.

Definition 1.8 *We define the following rewriting system on paraproof structures:*

$$\frac{A/u = B_1 \parr B_2}{\{\Gamma, A^{U\cup\{u1,u2\}}\}[X, \{\Delta, B_1, B_2\}] \rightarrow_P \{\Gamma, A^{U\cup\{u\}}\}[X, \{\Delta, (B_1 \parr B_2)\}]}$$

$$\frac{A/u = B_1 \otimes B_2}{\{\Gamma, A^{U\cup\{u1,u2\}}\}[X, \{\Delta_1, B_1\}, \{\Delta_2, B_2\}] \rightarrow_P \{\Gamma, A^{U\cup\{u\}}\}[X, \{\Delta_1, \Delta_2, B_1 \otimes B_2\}]}$$

We say that a proof structure $S = \{A_1^{U_1}, \ldots, A_n^{U_n}\}[X]$ *satisfies the* weak Parsing criterion *if*

$$S \rightarrow_P^{\star} \{A_1^{\{\epsilon\}}, \ldots, A_n^{\{\epsilon\}}\}[\{A_1, \ldots, A_n\}]$$

and that it satisfies the strong Parsing criterion *if any reduction sequence* $S \rightarrow_P^{\star} S'$ *can be completed by a reduction* $S' \rightarrow_P^{\star} \{A_1^{\{\epsilon\}}, \ldots, A_n^{\{\epsilon\}}\}[\{A_1, \ldots, A_n\}]$.

Lemma 1.9 *If* $S \rightarrow_P S'$*, and if* S *satisfies the DR-criterion, then* S' *satisfies the DR-criterion.*

Proof. Suppose that $S \rightarrow_P S'$ by the $\parr$ rule, and that a switching for S' has been fixed. S is the same graph as S' except that S has two additional vertices B_1 and B_2 and that the edge connecting $B_1 \parr B_2$ with its class in S' is replaced by a diamond of edges between $B_1 \parr B_2$, B_1, B_2, and its class $\{\Delta, B_1, B_2\}$ in S. We extend the switching to S by assigning, say L, to $B_1 \parr B_2$ (which is internal in S). By assumption, the correction graph is a tree. We can take B_2 as a root, which has the class $\{\Delta, B_1, B_2\}$ as unique son, which has B_1 among his sons, which has $B_1 \parr B_2$ as unique son. Then the correction graph for the original switching relative to S' is obtained by collapsing B_2 with $\{\Delta, B_1, B_2\}$ and B_1 with $B_1 \parr B_2$, and is still a tree.

Suppose now that $S \rightarrow_P S'$ by the $\otimes$ rule. A switching of S' is also a switching for S, whose associated correction graph is thus a tree. Let us take $B_1 \otimes B_2$ as a root. Then the correction graph for S' is obtained as follows: collapse B_1, its unique son $\{\Delta_1, B_1\}$, B_2, and its unique son $\{\Delta_2, B_2\}$. This clearly yields a tree. □

Proposition 1.10 *If a proof structure satisfies the DR-criterion, then it satisfies the strong Parsing criterion.*

Proof. Let S be a paraproof structure with conclusions $A_1, \ldots, A_n$. Clearly, each $\rightarrow_P$ reduction strictly decreases the size of the underlying forest, hence all reductions terminate. Let $S \rightarrow_P^{\star} S'$, where S' cannot be further reduced. We know by Lemma 1.9 that S' also satisfies the DR-criterion. We show that S' must be a generalized axiom (which actually entails that S' must precisely be $\{A_1^{\{\epsilon\}}, \ldots, A_n^{\{\epsilon\}}\}[\{A_1, \ldots, A_n\}]$, since the set of conclusions remains invariant under reduction). Suppose that S' is not a generalized axiom, and consider an arbitrary class Γ of the partition of S'. We claim that Γ contains at least one formula whose father is a $\otimes$ node. Indeed, otherwise, each element of the class is either a conclusion or a formula B whose father is a $\parr$ node. Note that in the latter case the other child of the $\parr$ node cannot belong to the class as otherwise S' would not be in normal form. If the father of B is $B \parr C$ (resp. $C \parr B$), we set its switch to R (resp. L), and we extend the switching arbitrarily to all the other internal $\parr$ nodes of S'. Then the restriction of the correction graph G to Γ and its elements forms a connected component, which is strictly included in G since S' is not a generalized axiom. But then G is not connected, which is a contradiction.

We now construct a path in S' as follows. We start from a leaf B_1 whose father is a $\otimes$ node, say $B_1 \otimes C$, we go down to its father and then up through C to a leaf B_1', choosing a path of maximal length. All along the way, when we meet $⅋$ nodes, we set the switches in such a way that the path will remain in the correction graph. B_1' cannot have a $\otimes$ node as father, as otherwise by maximality the other son C_1' of this father would be a leaf too, that cannot belong to the same class as this would make a cycle, and cannot belong to another class because S' is in normal form. Hence, by the claim, we can pick B_2 different from B_1' in its class whose father is a $\otimes$ node. We continue our path by going up from B_1' to its class, and then down to B_2, down to its father, and we consider again a maximal path upwards from there. Then this path cannot meet any previously met vertice, as otherwise, setting the switches in the same way as above until this happens, we would get a cycle in a correction graph. Hence we can reach a leaf B_2' without forming a cycle, and we can continue like this forever. But S' is finite, which gives a contradiction. □

Proposition 1.11 *If a paraproof structure satisfies the weak Parsing criterion, then it is sequentializable.*

Proof. We claim that if $S \rightarrow_P^\star S'$, then S can be obtained from S' by replacing each generalized axiom of S' by an appropriate paraproof net. We proceed by induction on the length of the derivation. In the base case, we have $S' = S$, and we replace each generalized axiom by itself. Suppose that $S \rightarrow_P^\star S_1' \rightarrow_P S'$. We use the notation of Definition 1.8. Suppose that $S_1' \rightarrow_P S'$ by the $⅋$ reduction rule. By induction, we have a paraproof net N to substitute for $\{\Delta, B_1, B_2\}$. We can add a $⅋$ node to N and get a new paraproof net which when substituted for $\{\Delta, B_1 ⅋ B_2\}$ in S' achieves the same effect as the substitution of N in S_1', keeping the same assignment of paraproof nets for all the other generalized axioms. The case of the $\otimes$ rule is similar: we now have by induction two paraproof nets N_1 and N_2 to substitute for $\{\Delta_1, B_1\}$ and $\{\Delta_2, B_2\}$, and we form a new paraproof net by the $\otimes$ rule which when substituted for $\{\Delta_1, \Delta_2, B_1 \otimes B_2\}$ in S' achieves the same effect as the substitution of N_1 and N_2 in S_1'. Hence the claim is proved. We get the statement by applying the claim to S and $\{A_1^{\{\epsilon\}}, \ldots, A_n^{\{\epsilon\}}\}[\{A_1, \ldots, A_n\}]$. □

We can collect the results obtained so far.

Theorem 1.12 *The following are equivalent for a paraproof structure S:*

1. *S is a paraproof net, i.e., is sequentializable,*
2. *S satisfies the DR-criterion,*
3. *S satisfies the strong Parsing criterion,*
4. *S satisfies the weak Parsing criterion.*

Proof. We have proved (1) ⇒ (2) ⇒ (3) and (4) ⇒ (1), and (3) ⇒ (4) is obvious. □

These equivalences are easy to upgrade to "full MLL", that is, to structures that also contain cuts. We briefly explain how the definitions are extended. A paraproof structure can now be formalized as $[X']\{A_1^{U_1}, \ldots, A_n^{U_n}\}[X]$, where X' is a (possibly empty) collection of disjoint subsets of $\{A_1, \ldots, A_n\}$, of the form $\{B, B^\perp\}$. The conclusions of the paraproof structure are the formulas in $\{A_1, \ldots, A_n\} \setminus \bigcup X'$. The underlying graph is defined as above, with now in addition an edge between B and $B^\perp$ for each class of the partial partition X'. A paraproof is obtained as previously, with a new proof rule, the cut rule:

$$\frac{\vdash \Gamma_1, B \quad \vdash \Gamma_2, B^\perp}{\vdash \Gamma_1, \Gamma_2}$$

A sequentializable paraproof structure is now one which is obtained through the following formal system, which adapts and extends (last rule) the one in Definition 1.3.

$$\frac{}{[]\{A_1^{\{\epsilon\}}, \ldots, A_n^{\{\epsilon\}}\}[\{\{A_1, \ldots, A_n\}\}]} \qquad \frac{[X']\{A_1^{U_1}, \ldots, A_n^{U_n}, B^V, C^W\}[X]}{[X']\{A_1^{U_1}, \ldots, A_n^{U_n}, (B \parr C)^U\}[X]}$$

$$\frac{[X']\{A_1^{U_1}, \ldots, A_n^{U_n}, B^V\}[X] \quad [Y']\{A_1'^{U_1'}, \ldots, A_{n'}'^{U_{n'}'}, C^W\}[Y]}{[X' \cup Y']\{A_1^{U_1}, \ldots, A_n^{U_n}, A_1'^{U_1'}, \ldots, A_{n'}'^{U_{n'}'}, (B \otimes C)^U\}[X \cup Y]}$$

$$\frac{[X']\{A_1^{U_1}, \ldots, A_n^{U_n}, B^V\}[X] \quad [Y']\{A_1'^{U_1'}, \ldots, A_{n'}'^{U_{n'}'}, (B^\perp)^W\}[Y]}{[X' \cup Y' \cup \{\{B, B^\perp\}\}]\{A_1^{U_1}, \ldots, A_n^{U_n}, A_1'^{U_1'}, \ldots, A_{n'}'^{U_{n'}'}, B^V, (B^\perp)^W\}[X \cup Y]}$$

The parsing rewriting system is adapted and extended as follows:

$$\frac{A/u = B_1 \parr B_2}{[X']\{\Gamma, A^{U \cup \{u1,u2\}}\}[X, \{\Delta, B_1, B_2\}] \to_P [X']\{\Gamma, A^{U \cup \{u\}}\}[X, \{\Delta, (B_1 \parr B_2)\}]}$$

$$\frac{A/u = B_1 \otimes B_2}{[X']\{\Gamma, A^{U \cup \{u1,u2\}}\}[X, \{\Delta_1, B_1\}, \{\Delta_2, B_2\}] \to_P [X']\{\Gamma, A^{U \cup \{u\}}\}[X, \{\Delta_1, \Delta_2, B_1 \otimes B_2\}]}$$

$$[X', \{B, B^\perp\}]\{\Gamma, B^\epsilon, (B^\perp)^\epsilon\}[X, \{\Delta_1, B\}, \{\Delta_2, B^\perp\}] \to_P [X']\{\Gamma\}[X, \{\Delta_1, \Delta_2\}]$$

Of course, the formalization begins to become quite heavy. We encourage the reader to draw the corresponding graph transformations. Theorem 1.12 holds for paraproof structures with cuts. The cut rule is very similar to the $\otimes$ rule, and indeed the case of a cut in the proofs is treated exactly like that of the $\otimes$ rule.

We have yet one more caracterization of (cut-free, this time) paraproof structures ahead, but let us pause and take profit from cuts: once they are in the picture, we want to explain how to compute with them, that is, how to eliminate them. There is a single cut elimination rule, which transforms a cut between $B_1 \otimes B_2$ and $B_1^\perp \parr B_2^\perp$ into two cuts between B_1 and $B_1^\perp$, and between B_2 and $B_2^\perp$, and eliminates the vertices $B_1 \otimes B_2$ and $B_1^\perp \parr B_2^\perp$. Formally:

$$\begin{array}{ll} & [X', \{B_1 \otimes B_2, B_1^\perp \parr B_2^\perp\}]\{\Gamma, (B_1 \otimes B_2)^U, (B_1^\perp \parr B_2^\perp)^V\}[X] \\ \to & [X', \{B_1, B_1^\perp\}, \{B_2, B_2^\perp\}]\{\Gamma, B_1^{U_1}, B_2^{U_2}, (B_1^\perp)^{V_1}, (B_2^\perp)^{V_2}\}[X] \end{array}$$

where $U_1 = \{u \mid 1u \in U\}$ and U_2, V_1, V_2 are defined similarly. Let us stress the difference in nature between cut-elimination ($\rightarrow$) and parsing ($\rightarrow_P$): the former is of a dynamic nature (we compute something), while the latter is of a static nature (we check the correctness of something).

How are we sure that the resulting structure is a paraproof net, if we started from a paraproof net? This must be proved, and we do it next.

Proposition 1.13 *If S is a paraproof net and $S \rightarrow S'$, then S' is also a paraproof net.*

Proof. In this proof, we use the second part of Proposition 1.7. Let us fix a switching for S', and a switching of the $⅋$ eliminated by the cut rule, say, L. This induces a correction graph on S, which can be represented with $B_1^\perp ⅋ B_2^\perp$ as root, with two sons $B_1 \otimes B_2$ and $B_1^\perp$, where the former has in turn two sons B_1 and B_2. Let us call T_1, T_2 and T_1' the trees rooted at B_1, B_2 and $B_1^\perp$, respectively. The correction graph for S' is obtained by placing T_1 as a new immediate subtree of T_1' and T_2 as a new immediate subtree of the tree rooted at $B_2^\perp$. We use here the fact that we know that $B_2^\perp$ occurs in T_1' (the important point is to make sure that it does not occur in T_2, as adding a cut between B_2 and $B_2^\perp$ would result both in a cycle and in disconnectedness). This shows that S' satisfies the DR-criterion, and hence is a paraproof net. □

We now describe yet another characterization of paraproof nets, elaborating on a suggestion of Girard in [33]. This last criterion is interactive in nature, and as such is an anticipation on the style of things that will be studied in the subsequent sections. The following definition is taken from [21, 23] (where it is used to generalize the notion of multiplicative connective).

Definition 1.14 *Let E be a finite set. Any two partitions X and Y of E induce a (bipartite) graph defined as follows: the vertices are the classes of X and of Y, the edges are the elements of E and for each $e \in E$, e connects the class of e in X with the class of e in Y. We say that X and Y are orthogonal if the induced graph is connected and acyclic.*

Let S be a paraproof structure with conclusions $A_1^{U_1}, \ldots, A_n^{U_n}$, and consider a multiset of paraproof nets $N_1 = \{(A_1^\perp)^{U_1}\}[X_1], \ldots, N_n = \{(A_1^\perp)^{U_n}\}[X_n]$, which we call collectively a counter-proof for S. By taking $X_1 \cup \ldots \cup X_n$, we get a partition of the (duals of the) leaves of S, which we call the partition induced by the counter-proof. We are now ready to formulate the Counterproof criterion, or CP-criterion for short.

Definition 1.15 (CP) *Let S be a paraproof structure. We say that S satisfies the* CP-criterion *if its partition is orthogonal to all the partitions induced by all the counter-proofs of S.*

Proposition 1.16 *If a paraproof structure S satisfies the DR criterion, then it also satisfies the CP-criterion.*

Proof. Let $N_1, \ldots, N_n$ be a counter-proof of S. We can form a paraproof net by placing cuts between A_1 and $A_1^\perp$, etc..., which also satisfies the DR criterion. It is obvious to see that the cut-elimination ends exactly with the graph induced by the two partitions, and hence we conclude by Proposition 1.13. □

Proposition 1.17 *If a paraproof structure satisfies the CP-criterion, then it satisfies the DR-criterion.*

PROOF. Let $S = \{A_1^{U_1}, \dots, A_n^{U_n}\}[X]$ be a paraproof structure satisfying the CP-criterion and let us fix a switching for S. To this switching we associate a multiset Π of extreme paraproofs of $\vdash A_1^\perp, \dots, \vdash A_n^\perp$, defined as follows: the $\otimes$ rules of the counter-proof have the form

$$\frac{\vdash \Gamma, B_1 \quad \vdash B_2}{\vdash \Gamma, B_1 \otimes B_2} \quad \text{or} \quad \frac{\vdash B_1 \quad \vdash \Gamma, B_2}{\vdash \Gamma, B_1 \otimes B_2}$$

according to whether the corresponding $\mathbin{⅋}$ of S has its switch set to L or to R, respectively. By performing a postorder traversal of the counter-proof, we determine a sequence $S_1, \dots, S_p$ of paraproof structures and a sequence $\Pi_1, \dots, \Pi_p$ of multisets of extreme paraproofs, such that the roots of Π_i determine a partition of the conclusions of S_i, for all i. We start with $S_1 = S$ and $\Pi_1 = \Pi$. We construct Π_{i+1} by removing the last rule used on one of the trees of Π_i, and S_{i+1} by removing from S_i the formula (and its incident edges) dual to the formula decomposed by this rule (this is cut elimination!). We stop at stage p when Π_p consists of the generalized axioms of Π only, and when S_p consists of the partition of S only. Let G_i be the graph obtained by taking the union of S_i restricted according to the (induced) switching and of the set of the roots of Π_i, and by adding edges between a conclusion A of S_i and a root $\vdash \Gamma$ of Π_i if and only if $A^\perp$ occurs in Γ. We show by induction on $p - i$ that G_i is acyclic and connected. Then applying this to $i = p - 1$ we obtain the statement, since G_1 is the correction graph associated to our switching to which one has added (harmless) edges from the conclusions of S to new vertices $\vdash A_1^\perp, \dots, \vdash A_n^\perp$, respectively. The base case follows by our assumption that S satisfies the CP-criterion: indeed, the graph G_p is obtained from the graph G induced by X and the partition induced by Π by inserting a new node in the middle of each of its edges, and such a transformation yields a tree from a tree.

Suppose that one goes from G_i to G_{i+1} by removing

$$\frac{\vdash \Gamma, B_1, B_2}{\vdash \Gamma, B_1 \mathbin{⅋} B_2}$$

Then, setting $N_1 = \vdash \Gamma, B_1, B_2$ and $N_1' = \vdash \Gamma, B_1 \mathbin{⅋} B_2$, G_i is obtained from G_{i+1} by renaming N_1 as N_1', by removing the two edges between $B_1^\perp$ and N_1 and between $B_2^\perp$ and N_1, by adding a new vertex $B_1^\perp \otimes B_2^\perp$ and three new edges linking $B_1^\perp \otimes B_2^\perp$ with $B_1^\perp$, $B_2^\perp$, and N_1'. Considering G_{i+1} as a tree with root N_1, we obtain G_i by cutting the subtrees rooted at $B_1^\perp$ and $B_2^\perp$ and by removing the corresponding edges out of N_1, by adding a new son to N_1 and by regrafting the subtrees as the two sons of this new vertex. This clearly results in a new tree.

Suppose now that one goes from G_i to G_{i+1} by removing, say

$$\frac{\vdash \Gamma, B_1 \quad \vdash B_2}{\vdash \Gamma, B_1 \otimes B_2}$$

Then, setting $N_1 = \vdash \Gamma, B_1$ and $N_1' = \vdash \Gamma, B_1 \otimes B_2$, G_i is obtained from G_{i+1} by renaming N_1 as N_1', by removing the vertex $\vdash B_2$ and the two edges between $B_1^\perp$ and N_1 and between $B_2^\perp$ and $\vdash B_2$, and by adding a new vertex $B_1^\perp \mathbin{⅋} B_2^\perp$ and two new edges linking $B_1^\perp \mathbin{⅋} B_2^\perp$ with $B_1^\perp$ and N_1'. Considering G_{i+1} as a tree with root $\vdash B_2$, we obtain G_i by cutting the root $\vdash B_2$ (which had a unique son) and by inserting a new node "in the middle" of the edge between $B_1^\perp$ and N_1, which makes a new tree. □

Hence, we have proved the following theorem.

Theorem 1.18 *The following are equivalent for a paraproof structure S:*

1. *S satisfies the CP-criterion,*
2. *S satisfies the DR-criterion.*

But there is more to say about this characterization. Let us make the simplifying assumption that we work with paraproof structures with a single conclusion A (note that any paraproof structure can be brought to this form by inserting enough final $\parr$ nodes). Let us say that a paraproof structure of conclusion A is *orthogonal* to a paraproof structure of conclusion $A^\perp$ if their partitions are orthogonal. Given a set H of paraproof structures of conclusion B, we write $H^\perp$ for the set of paraproof structures of conclusion $B^\perp$ which are orthogonal to all the elements of H. We have (where "of" is shorthand for "of conclusion"):

$$\begin{array}{rcl}\{\text{paraproof nets of } A^\perp\}^\perp & = & \{\text{paraproof nets of } A\}\\ & = & \{\text{extreme paraproof nets of } A^\perp\}^\perp\ .\end{array}$$

Indeed, Proposition 1.16 and the proof of Proposition 1.17 say that:

$$\begin{array}{l}\{\text{paraproof nets of } A\} \subseteq \{\text{paraproof nets of } A^\perp\}^\perp \text{ and}\\ \{\text{extreme paraproof nets of } A^\perp\}^\perp \subseteq \{\text{paraproof nets of } A\}\ ,\end{array}$$

respectively, and the two equalities follow since

$$\{\text{paraproof nets of } A^\perp\}^\perp \subseteq \{\text{extreme paraproof nets of } A^\perp\}^\perp\ .$$

We also have:

$$\begin{array}{rcl}\{\text{paraproof nets of } A\} & = & \{\text{paraproof nets of } A\}^{\perp\perp}\\ & = & \{\text{extreme paraproof nets of } A\}^{\perp\perp}\ .\end{array}$$

Indeed, using the above equalities, we have:

$$\begin{array}{rcl}\{\text{paraproof nets of } A\}^{\perp\perp} & = & \{\text{extreme paraproof nets of } A\}^{\perp\perp}\\ & = & \{\text{paraproof nets of } A^\perp\}^\perp\\ & = & \{\text{paraproof nets of } A^{\perp\perp}\}\\ & = & \{\text{paraproof nets of } A\}\ ,\end{array}$$

from which the conclusion follows. Anticipating on a terminology introduced in section 5, we thus have that the set of paraproof nets of A forms a *behaviour*, which is generated by the set of extreme paraproof nets. The paraproof nets of conclusion A are those paraproof structures that "stand the test" against all the counter-proofs $A^\perp$, which can be thought of opponents. We refer to Exercise 1.21 for another criterion of a game-theoretic flavour.

Putting the two theorems together, we have obtained three equivalent characterizations of sequentializable paraproof structures: the DR-criterion, the Parsing criterion, and the CP-criterion. What about the characterization of sequentializable proof structures? All these equivalences cut down to proof structures, thanks to the following easy proposition.

Proposition 1.19 *A sequentializable paraproof structure which is a proof structure is also a sequentializable proof structure.*

Proof. Notice that any application of a generalized axiom in the construction of a paraproof net remains in the partition until the end of the construction. Hence the only possible axioms must be of the form $\{C, C^\perp\}$. □

We end the section by sketching how proof nets for MELL are defined, that is, how the proof nets can be extended to exponentials. Recall from section 3 that cut-elimination involves duplication of (sub)proofs. Therefore, one must be able to know exactly which parts of the proof net have to be copied. To this aim, Girard introduced *boxes*, which record some sequentialization information on the net. Boxes are introduced each time a promotion occurs. The promotion amounts to add a new node to the graph, corresponding to the formula $!A$ introduced by the rule. But at the same time, the part of the proof net that represents the subproof of conclusion $\vdash ?\Gamma, A$ is placed in a box. The formulas of $?\Gamma$ are called the auxiliary ports of the box, while $!A$ is called the principal port. We also add:

- *contraction nodes*, labelled with a formula $?A$, which have exactly two incoming edges also labelled with $?A$,
- *dereliction nodes*, also labelled with a formula $?A$, which have only one incoming edge labelled with A,
- *weakening nodes*, labelled with $?A$, with no incoming edge.

The corresponding cut-elimination rules are:

- the dereliction rule, which removes a cut between a dereliction node labelled $?A^\perp$ and a principal port $!A$ and removes the dereliction node and the box (and its principal port) and places a new cut between the sons $A^\perp$ of $?A^\perp$ and A of $!A$;
- the contraction rule, which removes a cut between a contraction node labelled $?A^\perp$ and a principal port $!A$ and removes the contraction node and duplicates the box and places two new cuts between the two copies of $!A$ and the two sons of $?A^\perp$ and adds contraction nodes connecting the corresponding auxiliary ports of the two copies;
- the weakening rule, which removes a cut between a weakening node labelled $?A^\perp$ and a principal port $!A$ and removes the weakening node and erases the box except for its auxiliary ports which are turned into weakening nodes.

All commutative conversions of sequent calculus (cf. part I, section 3) but one disappear in proof nets. The only one which remains is:

- a rule which concerns the cut between an auxiliary port $?A^\perp$ of a box and the principal port $!A$ of a second box, and which lets the second box enter the first one, removing the auxiliary port $?A^\perp$, which is now cut inside the box.

These rules, and most notably the contraction rule, are global, as opposed to the elimination rules for MLL, which involve only the cut and the immediately neighbouring nodes and edges. By performing copying in local, elementary steps, one could hope to copy only what is necessary for the computation to proceed, and continue to share subcomputations that do not depend on the specific context of a copy. Such a framework has been proposed by Gonthier and his coauthors in two insigthful papers [27, 28] bridging Lévy's optimality theory [50], Lamping's implementation of this theory [46], and proof nets. A book length account of this work can be found in [7]. It is related to an important research program, called the *geometry of interaction* (GOI) [30, 53, 21]. A detailed survey would need another paper. Here we shall only sketch what GOI is about. The idea is to look at the paths (in the graph-theoretic sense) in a given proof net and to sort out those which are "meaningful " computationally. A cut edge is such a path, but more generally, a path between two formulas which will be linked by a cut later (that is, after some computation) is meaningful. Such paths may be termed *virtual* (a terminology introduced by Danos and Regnier [22]) since they say something about the reduction of the proof net without actually reducing it. Several independent characterizations of meaningful paths have been given, and turn out to be all equivalent [6, 7]:

- *Legal* paths [8], whose definition draws from a careful study of the form of Lévy's labels, which provide descriptions of the history of reductions in the λ-calculus and were instrumental in his theory of optimality.
- *Regular* paths. One assigns weights (taken in some suitable algebras) to all the edges of the proof net, and the regular paths are those which have a non-null composed weight.
- *Persistent* paths. These are the paths that are preserved by all computations, i.e. that no reduction sequence can break apart. Typically, in a proof net containing a $\otimes/⅋$ cut

  $$\dfrac{\dfrac{A \quad B}{A \otimes B} \qquad \dfrac{A^{\perp} \quad B^{\perp}}{A^{\perp} ⅋ B^{\perp}}}{}$$

  a path going down to A and then up through $B^{\perp}$ gets disconnected when the cut has been replaced by the two cuts on A and B, and hence is not persistent.

Any further information on proof nets should be sought for in [29, 32], and for additive proof nets (which are not yet fully understood) in, say, [55, 42]. We just mention a beautiful complexity result of Guerrini: it can be decided in (quasi-)linear time whether an MLL proof structure is a proof net [37].

Exercise 1.20 (MIX rule) *We define the* Acylicity criterion *by removing the connectedness requirement in the DR-criterion. Show that this criterion characterizes the paraproof structures that can be sequentialized with the help of the following additional rule:*

$$\frac{\{A_1^{U_1}, \ldots, A_n^{U_n}\}[X] \quad \{{A'}_1^{U'_1}, \ldots, {A'}_{n'}^{U'_{n'}}\}[Y]}{\{A_1^{U_1}, \ldots, A_n^{U_n}, {A'}_1^{U'_1}, \ldots, {A'}_{n'}^{U'_{n'}}\}[X \cup Y]}$$

that corresponds to adding the following rule to the sequent calculus of MLL, known as the MIX rule:

$$\frac{\vdash \Gamma \qquad \vdash \Delta}{\vdash \Gamma, \Delta}$$

(Hint: adapt correspondingly the Parsing criteria.)

Exercise 1.21 (AJ-criterion) *In this exercise, we provide a criterion for cut-free proofs which is equivalent to the Acyclicity criterion (cf. Exercise 1.20) and thus characterizes MLL+MIX cut-free proof nets, due to Abramsky and Jagadeesan [2, 9]. Let $S = \{A_1^{U_1}, \ldots, A_n^{U_n}\}[X]$ be a proof structure, and let*

$$M = \{m^+ \mid m = (u,i), i \in \{1,\ldots,n\}, u \in U_i\} \cup \{m^- \mid m = (u,i), i \in \{1,\ldots,n\}, u \in U_i\} .$$

(M is a set of moves, one of each polarity for each conclusion of an axiom – think of these formulas as the atoms out of which the conclusions of the proof structure are built). Let $U = \{(v,i) \mid v$ is a prefix of some $u \in U_i\}$. Given $(v,i) \in U$ and a sequence $s \in M^\star$, we define $s\restriction^{(v,i)}$ as follows:

$$\epsilon\restriction^{(v,i)} = \epsilon \qquad (s\,(u,j)^\lambda)\restriction^{(v,i)} = \begin{cases} (s\restriction^{(v,i)})\,(u,j)^\lambda & \text{if } i = j \text{ and } v \text{ is a prefix of } u \\ s\restriction^{(v,i)} & \text{otherwise} \end{cases}$$

where $\lambda \in \{+,-\}$. A finite sequence $s = m_1^- m_2^+ m_3^- \ldots$ is called a play *if it satisfies the following three conditions:*

1. *for all $(v,i) \in U$, $s\restriction^{(v,i)}$ is alternating,*
2. *for all $(v,i) \in U$, if $A_i/v = B_1 \otimes B_2$, then only Opponent can switch (v,i)-components in $s\restriction^{(v,i)}$,*
3. *for all $(v,i) \in U$, if $A_i/v = B_1 \parr B_2$, then only Player can switch (v,i)-components in $s\restriction^{(v,i)}$,*

where component switching is defined as follows: Opponent (resp. Player) switches (v,i)-components if $s\restriction^{(v,i)}$ contains two consecutive moves $(u,i)^-(w,i)^+$ (resp. $(u,i)^+(w,i)^-$) where $u = v1u'$ and $w = v2w'$, or $u = v2v'$ and $w = v1w'$.

We need a last ingredient to formulate the criterion. To the partition X we associate a function (actually, a transposition) ϕ which maps (u,i) to (w,j) whenever $\{A_i/u, A_j/w\} \in X$.

We say that S satisfies the AJ-criterion *if whenever $m_1^-\,\phi(m_1)^+\,m_2^-\,\phi(m_2)^+\,\ldots\,m_n^-$ is a play, then $m_1^-\,\phi(m_1)^+\,m_2^-\,\phi(m_2)^+\,\ldots\,m_n^-\,\phi(m_n)^+$ is also a play. (It says that ϕ considered as a strategy is winning, i.e. can always reply, still abiding to the "rules of the game".)*

Show that an MLL+MIX sequentializable proof structure satisfies the AJ-criterion and that a proof structure satisfying the AJ-criterion satisfies the Acylicity criterion. (Hints: (1) Show using a minimality argument that if there exists a switching giving rise to a cycle, the cycle can be chosen such that no two tensors visited by the cycle are prefix one of another. (2) Fix a tensor on the cycle (there must be one, why?), walk on the cycle going up from there, and let $(u,i), \phi(u,i), \ldots, (w,j), \phi(w,j)$ be the sequence of axiom conclusions visited along the way. Consider $s = (u,i)^-\,\phi(u,i)^+\,\ldots\,(w,j)^-\,\phi(w,j)^+$. Show that if the AJ-criterion is satisfied, then all the strict prefixes of s are plays.) Conclude that the AJ-criterion characterizes MLL+MIX proof nets.

Exercise 1.22 *Reformulate the example of cut-elimination given in part I, section 3:*

$$\dfrac{\begin{matrix}\vdots \\ \vdash ?A^\perp \parr ?B^\perp, !(A\&B)\end{matrix} \qquad \begin{matrix}\vdots \\ \vdash !A \otimes !B, ?(A^\perp \oplus B^\perp)\end{matrix}}{\vdash !(A\&B), ?(A^\perp \oplus B^\perp)}$$

using proof nets instead of sequents. Which reductions rule(s) should be added in order to reduce this proof to (an η-expanded form of the) identity (cf. part I, Remark 3.1)? Same question for the elimination of the cut on $!(A\&B)$. We refer to [48] for a complete study of type isomorphisms in (polarized) linear logic.

2 Towards ludics

The rest of the paper is devoted to ludics, a new research program started by Girard in [36]. Ludics arises from forgetting the logical contents of formulas, just keeping their *locative* structure, i.e., their shape as a tree, or as a storage structure. For example, if $(A \parr B) \oplus A$ becomes an address ξ, then the (instances of) the subformulas $A \parr B$ (and its subformulas A and B) and A become $\xi 1$ (and $\xi 11$ and $\xi 12$) and $\xi 2$, respectively. The relation between formulas and addresses is quite close to the relation between typed and untyped λ-calculus, and this relation will be made clear in the following section (as a modest contribution of this paper).

Thus, another aspect of resource consciousness comes in: not only the control on the use of resources, which is the main theme of linear logic, but also the control on their storage in a shared memory. An important consequence of this is that ludics gives a logical status to subtyping, a feature related to object-oriented programming languages. It has been long observed that the semantics of subtyping, record types, intersection types, is not categorical, i.e. cannot be modelled naturally in the framework of category theory, unlike the simply typed λ-calculus, whose match with cartesian closed categories has been one of the cornerstones of denotational semantics over the last twenty years. Ludics points to the weak point of categories: everything there is up to isomorphism, up to renaming. The framework of ludics forces us to explicitly recognize that a product of two structures calls for the copying of their shape in two disjoint parts of the memory. When this is not done, we simply get intersection types, so the product, or additive conjunction, appears as a special case of a more general connective, called the intersection.

As a matter of fact, prior to the times of categorical semantics, denotational semantics took models of the untyped λ-calculus as its object of study. The whole subject started with Scott's solution to the domain equation $D = D \to D$, which allows us to give a meaning to self-application. The early days of semantics were concerned with questions such as the completeness of type assignment to untyped terms (as happens in programming languages like ML which have static type inference) [40, 10]. Types were interpreted as suitable subsets of the untyped model, the underlying intuition being that of types as properties: a type amounts to the collection of all (untyped) terms to which it can be assigned. In this framework, subtyping is interpreted by inclusion, and intersection types by ... intersection [52, 12]. Ludics invites us to revisit these ideas with new glasses, embodying locations, and interactivity: in ludics, the intuition is that of types as behaviours, which adds an interactive dimension to the old paradigm.

Another key ingredient of ludics is the *daimon*, which the author of this paper recognized as an *error* element. We mean here a recoverable error in the sense of Cardelli-Wegner [13]. Such an error is a rather radical form of output, that stops execution and gets propagated to the top level. If the language is endowed with error-handling facilities, then programs can call other programs called handlers upon receiving such error messages, whence the name "recoverable". The importance of errors in denotational semantics was first noted by Cartwright and Felleisen in (1991) (see [14]). If A is an algorithm of two arguments such that $A(err, \bot) = err$, this test, or *interaction* between A and $(err, \bot)$ tells us that A wants to know its first argument before anything else. This information reveals us a part of the computation strategy of A. Note that $A(err, \bot) = \bot$ holds for an algorithm A that will examine its second argument first. It may then seem that err and $\bot$ play symmetric rôles. This is not quite true, because $\bot$ means "undefined", with the meaning of "waiting for some information", that might never come. So err is definitely terminating, while we don't know for $\bot$: $\bot$ is a sort of error whose meaning is overloaded with that of non-termination.

In ludics, Girard introduces (independently) the *daimon*, denoted by ✠, with the following motivation. Ludics is an interactive account of logic. So the meaning of a (proof of a) formula A lies in its behavior against observers, which are the proofs of other formulas, among which the most essential one is its (linear) negation $A^\perp$ (not A, the negation of linear logic [29, 32]). This is just like the contextual observational semantics of a program, where the meaning of a piece of program M is given by all the observations of the results of the evaluation of $C[M]$, where C ranges over full (closed, basic type) program contexts. Now, there is a problem: if we have a proof of A, we hardly have a proof of $A^\perp$. There are not enough "proofs": let us admit more! This is the genesis of the daimon. Proofs are extended to paraproofs (cf. section 1): in a paraproof, one can place daimons to signal that one gives up, i.e., that one assumes some formula instead of proving it. Hence there is always a paraproof of $A^\perp$: the daimon itself, which stands for "assume $A^\perp$" without even attempting to start to write a proof of it. The interaction between any proof of A and this "proof" reduced to the daimon terminates immediately. There are now enough inhabitants to define interaction properly!

We borrow the following comparison from Girard. In linear algebra, one expresses that two vectors x and y are orthogonal by writing that their scalar product $\langle x \mid y\rangle$ is 0: we are happy that there is "enough space" to allow a vector space and its orthogonal to have a non-empty intersection. Like 0, the daimon inhabits all (interpretations of) formulas, and plays the role of an absorbing element (as we shall see, it is orthogonal to all its counter-paraproofs). It even inhabits – and is the only (defined) inhabitant of – the empty sequent, which in this comparison could be associated with the base field. But the comparison should not be taken too seriously.

In summary, errors or daimons help us to terminate computations and to explore the behavior of programs or proofs interactively. Moreover, computation is "*streamlike*", or *demand-driven.* The observer detains the prompt for further explorations. If he wants to know more, he has to further defer giving up, and hence to display more of his own behavior. This is related to lazy style in programming, where one can program, say, the infinite list of prime numbers in such a way that each new call of the program will disclose the next prime number. Coroutines come to mind here too, see [20] for a discussion.

A third ingredient of ludics is *focalization*, which we explain next. In section 2 of part I, we observed that the connective $\otimes$ distributes over $\oplus$ and that both connectives are irreversible, and that on the other hand $⅋$ distributes over & and that both connectives are reversible. We introduced there the terminology of positive connectives ($\otimes$, $\oplus$) and negative connectives ($⅋$, &). We extend the terminology to formulas as follows: a formula is positive (resp. negative) if its topmost connective is positive (resp. negative).

Andreoli shed further light on this division through his work on focalization [5]. His motivation was to reduce the search space for proofs in linear logic. Given, say, a positive formula, one groups its positive connectives from the root, and then its negative connectives, etc.... Each grouping is considered as a single synthetic connective. Let us illustrate this with an example. Consider $((N_1 \otimes N_2)\&Q)⅋R$, with Q, R, and $P = N_1 \otimes N_2$ positive and N_1, N_2 negative. The signs, or *polarities* of these formulas show evidence of the fact that maximal groupings of connectives of the same polarity have been made. We have two synthetic connectives, a negative and ternary one that associates $(P\&Q)⅋R$ with P, Q, R, and a positive one which is just the connective $\otimes$. The rule for the negative synthetic connective is:

$$\frac{\vdash P, R, \Lambda \qquad \vdash Q, R, \Lambda}{\vdash (P\&Q)⅋R, \Lambda}$$

Thus, a focused proof of $((N_1 \otimes N_2)\&Q)\parr R$ ends as follows:

$$\dfrac{\dfrac{\vdash N_1, R, \Lambda_1 \quad \vdash N_2, \Lambda_2}{\vdash N_1 \otimes N_2, R, \Lambda} \qquad \vdash Q, R, \Lambda}{\vdash ((N_1 \otimes N_2)\&Q)\parr R, \Lambda}$$

Notice the alternation of the active formulas in the proof: negative $((P\&Q)\parr R)$, then positive $(N_1 \otimes N_2)$, etc... (We recall that at each step the active formula is the formula whose topmost connective has just been introduced.) The same proof can be alternatively presented as follows:

$$\dfrac{\dfrac{N_1^\perp \vdash R, \Lambda_1 \quad N_2^\perp \vdash \Lambda_2}{\vdash N_1 \otimes N_2, R, \Lambda} \qquad \vdash Q, R, \Lambda}{((N_1 \otimes N_2)^\perp \oplus Q^\perp) \otimes R^\perp \vdash \Lambda}$$

The advantage of this formulation is that it displays only positive connectives. Notice also that there is at most one formula on the left of $\vdash$. This property is an invariant of focalization, and is a consequence of the following observation.

Remark 2.1 *In a bottom-up reading of the rules of MALL, only the $\parr$ rule augments the number of formulas in the sequent. In the other three rules for $\&$, $\otimes$, and $\oplus$, the active formula is replaced by a single subformula, and the context stays the same or gets shrinked. Thus, only a negative formula can give rise to more than one formula in the same sequent when it is active.*

Adapting this remark to synthetic connectives, we see that only a negative synthetic connective can give rise to more than one formula when it is active, and these formulas are positive by the maximal grouping.

The *focusing discipline* is defined as follows:

1. Once the proof-search starts the decomposition of a formula, it keeps decomposing its subformulas until a connective of the opposite polarity is met, that is, the proof uses the rules for synthetic connectives;

2. A negative formula if any is given priority for decomposition.

The focusing discipline preserves the invariant that a (monolateral) sequent contains at most one negative formula. Indeed, if the active formula is positive, then all the other formulas in the sequent are also positive (as if there existed a negative formula it would have to be the active one), and then all the premises in the rule have exactly one negative formula, each of which arises from the decomposition of the active formula (like N_1 or N_2 above). If the active formula is negative, then all the other formulas of the sequent are positive, and each premise of the rule is a sequent consisting of positive formulas only (cf. P, Q, R above). Initially, one wants to prove a sequent consisting of a single formula, and such a sequent satisfies the invariant.

From now on, we consider sequents consisting of positive formulas only and with at most one formula on the left of $\vdash$. We only have positive synthetic connectives, but we now have a left rule and a set of right rules for each of them: the right rules are irreversible, while the left rule is reversible.

Note that the left rule is just a reformulation of the right rule of the corresponding negative synthetic connective. Here are the rules for the ternary connective $(P^\perp \oplus Q^\perp) \otimes R^\perp$:

$$\{\{P,R\},\{Q,R\}\} \quad \frac{\vdash P,R,\Lambda \qquad \vdash Q,R,\Lambda}{(P^\perp \oplus Q^\perp) \otimes R^\perp \vdash \Lambda} \qquad \begin{array}{c} \{P,R\} \quad \dfrac{P \vdash \Gamma \qquad R \vdash \Delta}{\vdash (P^\perp \oplus Q^\perp) \otimes R^\perp, \Gamma, \Delta} \\ \{Q,R\} \quad \dfrac{Q \vdash \Gamma \qquad R \vdash \Delta}{\vdash (P^\perp \oplus Q^\perp) \otimes R^\perp, \Gamma, \Delta} \end{array}$$

Here is how the top right rule has been synthesized:

$$\dfrac{\dfrac{P \vdash \Gamma}{\vdash P^\perp \oplus Q^\perp, \Gamma} \qquad R \vdash \Delta}{\vdash (P^\perp \oplus Q^\perp) \otimes R^\perp, \Gamma, \Delta}$$

The synthesis of the other rules is similar. Note that the isomorphic connective $(P^\perp \otimes Q^\perp) \oplus (P^\perp \otimes R^\perp)$, considered as a ternary connective, gives rise to the same rules. For example the top right rule is now synthesized as follows:

$$\dfrac{\dfrac{P \vdash \Gamma \quad R \vdash \Delta}{\vdash P^\perp \otimes R^\perp, \Gamma}}{\vdash ((P^\perp \otimes Q^\perp) \oplus (P^\perp \otimes R^\perp), \Gamma, \Delta}$$

More generally, it is easily seen that any positive synthetic connective, viewed as a linear term over the connectives $\otimes$ and $\oplus$, can be written as a $\oplus$ of $\otimes$, modulo distributivity and associativity:

$$P(N_1, \ldots, N_k) = \cdots \oplus (N_{\phi(m,1)} \otimes \cdots \otimes N_{\phi(m,j_m)}) \oplus \cdots$$

where m ranges over $\{1, \ldots n\}$ for some n, the range of ϕ is $\{1, \ldots, k\}$, for each m the map $\phi(m, \_) = \lambda j.\phi(m,j)$ is injective, and the map $\lambda m.\phi(m, \_)$ is injective (for example, for $(N_1 \oplus N_2) \otimes N_3$, we have $n = 2$, $j_1 = j_2 = 2$, $\phi(1,1) = 1$, $\phi(1,2) = 3$, $\phi(2,1) = 2$, and $\phi(2,2) = 3$). Note that the connective is equivalently described by a (finite) set of finite subsets of $\{1, \ldots, k\}$. The rules for $P(\cdots)$ are as follows (one rule for each $m \in \{1, \ldots n\}$):

$$\frac{N_{\phi(m,1)} \vdash \Lambda_1 \quad \cdots \quad N_{\phi(m,j_m)} \vdash \Lambda_{j_m}}{\vdash P(N_1, \ldots, N_k), \Lambda_1, \ldots \Lambda_{j_m}}$$

There is one rule (scheme) corresponding to each component of the $\oplus$ that selects this component and splits its $\otimes$. Dually, a negative synthetic connective can be written as a & of $\parr$:

$$N(P_1, \ldots, P_k) = \cdots \;\&\; (P_{\phi(m,1)} \parr \cdots \parr P_{\phi(m,j_m)}) \;\&\; \cdots$$

Here is the rule for $N(\cdots)$.

$$\frac{\vdash P_{\phi(1,1)},\ldots,P_{\phi(1,j_1)},\Lambda \quad \cdots \quad \vdash P_{\phi(n,1)},\ldots,P_{\phi(n,j_n)},\Lambda}{N(P_1,\ldots,P_k)^\perp \vdash \Lambda}$$

There is one premise corresponding to each component of the &, and in each premise the ⅋'s have been dissolved into a sequence of positive formulas.

Remark 2.2 *Note that in the negative rule we have no choice for the active formula: it is the unique formula on the left. Thus, the negative rule is not only reversible at the level of the formula $N(\cdots)$, but also at the level of the sequent $N(\cdots)^\perp \vdash \Lambda$. In contrast, in a positive rule, one has to choose not only the m and the disjoint $\Lambda_1,\ldots,\Lambda_{j_m}$, as noted above, but also the formula $P(\cdots)$ in the sequent $\vdash P(\cdots),\Lambda$.*

Below, we display some examples of non-focusing proofs:

$$\dfrac{\dfrac{\dfrac{\vdots}{\vdash P\&Q, N_1, \Lambda_1} \quad \dfrac{\vdots}{\vdash R, N_2, \Lambda_2}}{\vdash P\&Q, R, N_1 \otimes N_2, \Lambda_1, \Lambda_2}}{\vdash (P\&Q)⅋R, N_1 \otimes N_2, \Lambda_1, \Lambda_2}$$

$$\dfrac{\dfrac{\vdots}{\vdash (P\&Q)⅋R, N_1, \Lambda_1} \quad \dfrac{\vdots}{\vdash N_2, \Lambda_2}}{\vdash (P\&Q)⅋R, N_1 \otimes N_2, \Lambda_1, \Lambda_2}$$

In the first proof, we did not respect the maximal grouping of negative connectives, and we abandoned our focus on $(P\&Q)⅋R$ to begin to work on another formula of the sequent. In the second proof, we did not respect the priority for the negative formula of the sequent. These examples show that the change of granularity induced by focusing is not innocuous, because the focusing discipline forbids some proofs. Of course, this was the whole point for Andreoli and Pareschi, who wanted to reduce the search space for proofs. But is the focusing discipline complete in terms of provability? The answer is yes [5], i.e., no provable sequents are lost. The result actually holds for the whole of linear logic, with ! (resp. ?) acting on a negative (resp. positive) formula to yield a positive (resp. negative) formula (see Remark 2.5).

Before we state and sketch the proof of the focalization theorem, we introduce a focusing sequent calculus [5, 35] which accepts only the focusing proofs. First, we note that a better setting for polarized formulas consists in insisting that a positive connective should connect positive formulas and a negative connective should connect negative formulas. This is possible if we make changes of polarities explicit with the help of change-of-polarity operators (cf., e.g., [47]). For example, $((N_1 \otimes N_2)\&Q)⅋R$ should be written as $(\uparrow((\downarrow N_1)) \otimes (\downarrow N_2))\&(\uparrow Q))⅋(\uparrow R)$. For MALL, we get the following syntax for positive and negative formulas (assuming by convention that the atoms X are all positive, which is no loss of generality since we get "as many" negative atoms $X^\perp$).

$$P ::= X \mid P \otimes P \mid P \oplus P \mid 1 \mid 0 \mid \downarrow N$$
$$N ::= X^\perp \mid N \invamp N \mid N \& N \mid \perp \mid \top \mid \uparrow P$$

The operators $\downarrow$ and $\uparrow$ are called the *shift* operations. There are two kinds of sequents: $\vdash \Gamma;$ and $\vdash \Gamma; P$ where in the latter the only negative formulas allowed in Δ are atoms. The zone in the sequent on the right of ";" is called the *stoup*, a terminology which goes back to an earlier paper of Girard on classical logic [31]. The stoup is thus either empty or contains exactly one formula. The rules of this sequent calculus are as follows (we omit the units):

$$\frac{}{\vdash P^\perp; P} \qquad \textbf{Focalization}\ \frac{\vdash \Gamma; P}{\vdash \Gamma, P;}$$

SHIFT

$$\frac{\vdash \Gamma, P;}{\vdash \Gamma, \uparrow P;} \qquad \frac{\vdash \Delta, N;}{\vdash \Delta; \downarrow N}$$

POSITIVE

$$\frac{\vdash \Gamma; P_1}{\vdash \Gamma; P_1 \oplus P_2} \quad \frac{\vdash \Gamma; P_2}{\vdash \Gamma; P_1 \oplus P_2} \qquad \frac{\vdash \Gamma_1; P_1 \quad \vdash \Gamma_2; P_2}{\vdash \Gamma_1, \Gamma_2; P_1 \otimes P_2}$$

NEGATIVE

$$\frac{\vdash N_1, N_2, \Gamma;}{\vdash N_1 \invamp N_2, \Gamma;} \qquad \frac{\vdash N_1, \Gamma; \quad \vdash N_2, \Gamma;}{\vdash N_1 \& N_2, \Gamma;}$$

In the focalization rule we require that the negative formulas of Γ (if any) are atomic. It should be clear that focusing proofs are in exact correspondence with the proofs of linear logic that respect the focusing discipline. In one direction, one inserts the shift operations at the places where polarity changes, like we have shown on a sample formula, and in the other direction one just forgets these operations. The synthetic connective view further abstracts from the order in which the negative subformulas of a negative connective are decomposed.

Remark 2.3 *In view of the above sequent calculus, a weaker focusing discipline appears more natural, namely that obtained by removing the constraint that in a sequent with a non-empty stoup the context consists of positive formulas and negative atoms only. What then remains of the focusing discipline is to maintain the focus on the decomposition of* positive *formulas, which is enforced under the control of the stoup.*

Theorem 2.4 (Focalization) *The focusing discipline is complete, i.e., if $\vdash A$ is provable in linear logic, then it is provable by a cut-free proof respecting the focusing discipline.*

Proof (indication). The theorem says among other things that nothing is lost by giving priority to the negative connectives. In the two examples of non-focusing proofs, the decomposition of the $\otimes$ can be easily permuted with the decomposition of the $\&$, or of the $\invamp$. Conversely, the decomposition of $\invamp$ has given more possibilities of proofs for the decomposition of $\otimes$, allowing to send $P \& Q$ on the left and R on the right. But, more importantly, the theorem also says that nothing is lost by focusing on a positive formula and its topmost positive subformulas. More precisely, not any positive formula of the sequent to prove will do, but at least one of them can be focused on, as the following example shows:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash N_2, N_2^\perp} \quad \dfrac{}{\vdash N_3, N_3^\perp}}{\vdash N_2 \otimes N_3, N_2^\perp, N_3^\perp}}{\vdash N_2 \otimes N_3, N_2^\perp \parr N_3^\perp} \qquad \dfrac{}{\vdash N_4, N_4^\perp}}{\vdash N_2 \otimes N_3, N_4 \otimes (N_2^\perp \parr N_3^\perp), N_4^\perp}}{\vdash N_1 \oplus (N_2 \otimes N_3), N_4 \otimes (N_2^\perp \parr N_3^\perp), N_4^\perp}$$

This proof is not focused: the decomposition of the first positive formula $N_1 \oplus (N_2 \otimes N_3)$ of the conclusion sequent is blocked, because it needs to access negative subformulas of the second positive formula. But focusing on the second formula works:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash N_2, N_2^\perp} \quad \dfrac{}{\vdash N_3, N_3^\perp}}{\vdash N_2 \otimes N_3, N_2^\perp, N_3^\perp}}{\vdash N_1 \oplus (N_2 \otimes N_3), N_2^\perp, N_3^\perp}}{\vdash N_1 \oplus (N_2 \otimes N_3), N_2^\perp \parr N_3^\perp} \qquad \dfrac{}{\vdash N_4, N_4^\perp}}{\vdash N_1 \oplus (N_2 \otimes N_3), N_4 \otimes (N_2^\perp \parr N_3^\perp), N_4^\perp}$$

A simple proof of the focalization theorem can be obtained by putting these observations together (Saurin [54]). More precisely, the following properties of sequents of linear logic are easily proved:

(1) Any provable sequent $\vdash \Gamma, N$ has a proof in which N is active at the last step: this is an easy consequence of the fact that a configuration "negative rule followed by positive rule" can be permuted to a configuration "positive rule followed by negative rule" (and not conversely).

(2) Any provable sequent $\vdash \Gamma$ consisting of positive formulas only is such that there exists at least one formula P of Γ and a proof of $\vdash \Gamma$ which starts (from the root) with a complete decomposition of (the synthetic connective underlying) P. Let us sketch the proof. If the last step of the proof is a $\oplus$ introduction, say, $\Gamma = \Delta, A_1 \oplus A_2$ and $\vdash \Gamma$ follows from $\vdash \Delta, A_1$, induction applied to the latter sequent yields either A_1, in which case $A_1 \oplus A_2$ will do, or a formula P of Δ, in which case we take that formula, and we modify the proof of $\vdash \Delta, A_1$ obtained by induction, as follows: we spot the place where A_1 appears when the decomposition of P is completed, and insert the $\oplus$ introduction there, replacing systematically A_1 by $A_1 \oplus A_2$ from the insertion point down to the root, yielding a proof of $\vdash \Delta, A_1 \oplus A_2$ that satisfies the statement. The tensor case is similar.

Then a focalized proof can be obtained out of any MALL proof by repeatedly applying (1) to a cut-free proof so as to turn the sequent into a sequent of positive formulas only, and then (2), etc... The size of the sequents decreases at each step, so the procedure creates no infinite branch. The same property is likely to work in presence of exponentials, using the same kind of ordinal as in a cut-elimination proof.

Steps (1) and (2) may seem ad hoc manipulations, but in fact, they can be taken care of by cut-elimination. For example, setting $R' = P\&Q$, we can reshape the proof

$$\dfrac{\dfrac{\vdots}{\vdash R'\parr R, N_1, \Lambda_1} \qquad \dfrac{\vdots}{\vdash N_2, \Lambda_2}}{\vdash R'\parr R, N_1 \otimes N_2, \Lambda_1, \Lambda_2}$$

so as to get a proof ending with the decomposition of the negative connective $\parr$, as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{\vdash R'\parr R, N_1, \Lambda_1} \qquad \dfrac{\dfrac{}{\vdash R'^{\perp}, R'} \quad \dfrac{}{\vdash R^{\perp}, R}}{\vdash (R'\parr R)^{\perp}, R, R'}}{\vdash R', R, N_1, \Lambda_1} \qquad \dfrac{\vdots}{\vdash N_2, \Lambda_2}}{\vdash R', R, N_1 \otimes N_2, \Lambda_1, \Lambda_2}}{\vdash R'\parr R, N_1 \otimes N_2, \Lambda_1, \Lambda_2}$$

This kind of transformation can be performed systematically, so as to yield a proof which can be formalized in the focused sequent calculus *with cuts*. The resulting proof can then be run through the cut-elimination in the focused caclulus, yielding a focused and cut-free proof. This approach is closer to Andreoli's original proof (see [49] for details). □

Remark 2.5 *As a hint of how the focalization theorem extends to exponentials, let us mention that Girard has proposed to decompose* $!N$ *and* $?P$ *as* $\downarrow \sharp N$ *and* $\uparrow \flat P$ *, where* $\sharp$ *is a negative connective, taking a negative formula* N *to a negative formula* $\sharp N$ *and where* $\flat$ *is dually a positive connective. The change-of-polarity aspect of* $!$ *and* $?$ *is taken care of by the already introduced just-change-of-polarity operators* $\downarrow$ *and* $\uparrow$*. The rules for these connectives in the focusing system are as follows:*

$$\dfrac{\vdash ?\Gamma, N;}{\vdash ?\Gamma, \sharp N;} \qquad \dfrac{\vdash \Gamma; P}{\vdash \Gamma; \flat P} \qquad \dfrac{\vdash \Gamma;}{\vdash \Gamma, \flat P;} \qquad \dfrac{\vdash \Gamma, \flat P, \flat P;}{\vdash \Gamma, \flat P;}$$

Exercise 2.6 *Show that the focusing proof system (in its weaker form) enjoys the cut-elimination property.*

Exercise 2.7 *Show that* $\sharp$ *is reversible, and discuss the irreversibility of* $\flat$*.*

3 Designs

We now arrive at the basic objects of ludics: the *designs*, which are "untyped paraproofs" (cf. section 1). Designs are (incomplete) proofs from which the logical content has been almost erased. Formulas are replaced by their (absolute) addresses, which are sequences of relative addresses recorded as numbers. Thus addresses are words of natural numbers. We let $\zeta, \xi, \cdots \in \omega^\star$ range over addresses. We use $\xi \star J$ to denote $\{\xi j \mid j \in J\}$, and we sometimes write $\xi \star i$ for ξi. The empty address is written ϵ. Girard uses the terminology *locus* and *bias* for absolute address and relative address, respectively. Let us apply this forgetful transformation to our example $(P^\perp \oplus Q^\perp) \otimes R^\perp$ of the previous section, assigning relative addresses $1, 2, 3$ to $P^\perp, Q^\perp, R^\perp$, respectively.

$$(-, \xi, \{\{1,3\},\{2,3\}\}) \; \frac{\vdash \xi 1, \xi 3, \Lambda \qquad \vdash \xi 2, \xi 3, \Lambda}{\xi \vdash \Lambda} \qquad \begin{array}{c} (+, \xi, \{1,3\}) \; \dfrac{\xi 1 \vdash \Gamma \qquad \xi 3 \vdash \Delta}{\vdash \xi, \Gamma, \Delta} \\[2ex] (+, \xi, \{2,3\}) \; \dfrac{\xi 2 \vdash \Gamma \qquad \xi 3 \vdash \Delta}{\vdash \xi, \Gamma, \Delta} \end{array}$$

Notice the sets $\{1,3\}$ and $\{2,3\}$ appearing in the names of the rules. They can be thought of as a reservation of some (relative) addresses, to be used to store immediate subformulas of the current formula. Girard calls these sets *ramifications*.

We then consider a notion of "abstract sequent": a negative (resp. positive) *base* (also called *pitchfork* by Girard) has the form $\xi \vdash \Lambda$ (resp. $\vdash \Lambda$), where Λ is a finite set of addresses, i.e., a base is a sequent of addresses. We always assume a *well-formedness* conditions on bases: that all addresses in a base are pairwise disjoint (i.e. are not prefix one of the other). We spell out this condition in the key definitions.

The *designs* are trees built via the following rules (ω denotes the set of natural numbers, and $\mathcal{P}_f(\omega)$ denotes the set of finite subsets of ω):

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

$$✠ \; \frac{}{\vdash \Lambda}$$

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

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

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

$$(-, \xi, \mathcal{N}) \; \frac{\cdots \quad \vdash \xi \star J, \Lambda_J \cdots}{\xi \vdash \Lambda}$$

The first rule is the type-free version of the generalized axioms of Definition 1.2, while the two other rules are the type-free versions of the rules for the synthetic connectives given in section 2. The root of the tree is called the *base* of the design. A design is called negative or positive according to whether its base is positive or negative.

Here is how a generic negative design ends.

$$(-,\xi_1,\mathcal{N}_1)\;\dfrac{\cdots\quad (+,\xi_2,I_1)\;\dfrac{\cdots\quad (-,\xi_2 i_1,\mathcal{N}_2)\;\dfrac{\cdots\quad (+,\xi_3,I_2)\;\dfrac{\cdots\quad \xi_3 i_2 \vdash \Lambda_5\quad\cdots}{\vdash \xi_2 i_1 \star J_2,\Lambda_4}\quad\cdots}{\xi_2 i_1 \vdash \Lambda_3}\quad\cdots}{\vdash \xi_1 \star J_1,\Lambda_2}\quad\cdots}{\xi_1 \vdash \Lambda_1}$$

Note that the definition allows infinite designs, because the negative rule allows for infinite branching. But we also allow vertical infinity, i.e., infinite depth, and in particular we allow for recursive definitions of designs, as in the following fundamental example. The *fax* $\mathit{Fax}_{\xi,\xi'}$ based on $\xi \vdash \xi'$ is the infinite design which is recursively specified as follows:

$$\mathit{Fax}_{\xi,\xi'} \quad = \quad (-,\xi,\mathcal{P}_f(\omega))\;\dfrac{\cdots\quad (+,\xi',J_1)\;\dfrac{\cdots\quad \mathit{Fax}_{\xi' j_1,\xi j_1}\quad\cdots}{\vdash \xi \star J_1,\xi'}\quad\cdots}{\xi \vdash \xi'}$$

As its name suggests, the fax is a design for copying, which is the locative variant of identity, as it maps data to the "same" data, but copied elsewhere. It is indeed folklore from the work on game semantics (see e.g. [3]) – and actually from the earlier work of [11] – that computations are just mosaics of small bricks consisting of such copycat programs. Partial (finite or infinite) versions of the fax are obtained by deciding to place some $\mathcal{N}$'s instead of having systematically $\mathcal{P}_f(\omega)$ everywhere. How to define infinite designs of this kind and solve such recursive equations precisely is the matter of exercise 3.16. Other simple examples of designs are given in definition 3.4.

Remark 3.1 *Note that the three rules embody implicit potential weakening. Recall that weakening consists in deriving $\Gamma,\Gamma' \vdash \Delta,\Delta'$ from $\Gamma \vdash \Delta$ (for arbitrary Γ' and Δ'). Indeed, in the Daimon rule, Λ is arbitrary, and in the two other rules we only require that Λ contains the union of the Λ_i's (resp. the Λ_J's) (see also Remark 3.11).*

Remark 3.2 *It is immediate that the addresses appearing in a design based on $\vdash \Lambda$ (resp. on $\zeta \vdash \Lambda$) are extensions $\xi\xi'$ of addresses ξ of Λ (resp. $\{\zeta\} \cup \Lambda$). Also, it is easy to see that the property of well-formedness is forced upon designs whose base has the form $\vdash \xi$ or $\xi \vdash$, as well as the following one (the parity of an address refers to whether its length is even or odd):*

- *All addresses in each base on the right of $\vdash$ have the same parity, and then the formula on the left if any has the opposite parity.*

Girard requires this propertiy explicitly in the definition of a base. It does not appear to play an essential role, though.

Two flavors of designs. Girard also gives an alternative definition of designs, as strategies. Let us briefly recall the dialogue game interpretation of proofs (cf. part I, section 2). In this interpretation, proofs are considered as strategies for a Player playing against an Opponent. The Player plays inference rules, and the opponent chooses one of the premises in order to challenge the Player to disclose the last inference rule used to establish this premise, etc... . In a positive rule, Player chooses a ξ and an I. (Going back to the setting of section 2, he chooses which formula (ξ) to decompose, and which synthetic rule (I) to apply.) But in a negative rule $(-,\xi,\mathcal{N})$, Player does not choose the ξ (cf. Remark 2.2). Moreover, we can equivalently formulate the information conveyed by $\mathcal{N}$ by adding a premise Ω ("undefined") for each $I \notin \mathcal{N}$: intuitively, the non-presence of I in $\mathcal{N}$ is not a choice of Player, but rather a deficiency of her strategy: she has not enough information to answer an "attack" of Opponent that would interrogate this branch of the tree reduced to Ω. These remarks suggest a formulation where the negative rule disappears, or rather is merged with the positive rule:

$$(+,\xi,I)\ \frac{\overbrace{\cdots\quad \overbrace{\cdots\quad \vdash \xi i\star J,\Lambda_{i,J}\quad\cdots}^{J\in\mathcal{P}_f(\omega)}\quad\cdots}^{i\in I}}{\vdash \xi,\Lambda}$$

where Λ_{i_1,J_1} and Λ_{i_2,J_2} are disjoint as soon as $i_1 \neq i_2$. And we need to introduce Ω:

$$\Omega\ \frac{}{\vdash \Lambda}$$

This axiom is used for each $J' \in \mathcal{P}_f(\omega)\setminus\mathcal{N}$. Note that this change of point of view leads us to add a new tree that did not exist in the previous definition, namely the tree reduced to the Ω axiom. Girard calls it the *partial design*.

Remark 3.3 *Note that in the above formulation of the rule, there are infinitely many premises, that are split into a finite number of infinite sets of premises, one for each $i \in I$. The syntax that we shall introduce below respects this layering.*

Under this presentation of designs, an Opponent's move consists in picking an i (or ξi), and a J. We thus use names $(-,\zeta,J)$ to denote Opponent's moves after a rule $(+,\xi,I)$, with $\zeta = \xi i$ for some $i \in I$. Here is how the generic example above gets reformulated:

$$\cdots\quad \Omega\ \frac{}{\vdash \xi_1\star J_1',\Lambda_2'}\ \cdots\qquad (+,\xi_2,I_1)\ \frac{\cdots\quad \Omega\ \frac{}{\vdash \xi_2 i_1\star J_2',\Lambda_4'}\ \cdots\qquad (+,\xi_3,I_2)\ \frac{\cdots}{\vdash \xi_2 i_1\star J_2,\Lambda_4}\ \cdots}{\vdash \xi_1\star J_1,\Lambda_2}\quad\cdots$$

Finally, we can get rid of the "abstract sequents", and we obtain a tree (resp. a forest) out of a positive (resp. negative) design, whose nodes are labelled by alternating moves $(+,\xi,I)$ and $(-,\zeta,J)$, which are called positive and negative *actions*, respectively:

$$\cdots \quad \Omega\,\frac{\qquad}{(-,\xi_1,J_1')}\;\cdots \quad (+,\xi_2,I_1)\,\frac{\cdots\;\;\Omega\,\frac{\qquad}{(-,\xi_2 i_1,J_2')}\;\cdots\quad (+,\xi_3,I_2)\,\frac{\cdots}{(-,\xi_2 i_1,J_2)}\;\cdots}{(-,\xi_1,J_1)}\quad\cdots$$

By abuse of language (especially for Ω), Ω and $\maltese$ can also be called positive actions. We say that $(+,\xi,I)$ (resp. $(-,\zeta,J)$) is focalized in ξ (resp. ζ). The *opposite* (or dual) of an action $(+,\xi,I)$ (resp. $(-,\xi,I)$) is the action $(-,\xi,I)$ (resp. $(+,\xi,I)$).

Girard describes the tree or forest of actions as its set of branches, or *chronicles*, which are alternating sequences of actions (see Remark 3.8). A chronicle ending with a negative (resp. positive) action is called negative (resp. positive). Here, we choose a slightly different approach, and we present a *syntax* for designs. More precisely, we first present *pseudo-designs*, or raw designs, and we use a typing system (or, rather, a sorting system) to single out the designs as the well-typed pseudo-designs.

Here is the syntax:

Positive pseudo-designs	$\phi ::= \Omega \parallel \maltese \parallel (+,\xi,I)\cdot\{\psi_{\xi i} \mid i \in I\}$
Negative pseudo-designs	$\psi_\zeta ::= \{(-,\zeta,J)\phi_J \mid J \in \mathcal{P}_f(\omega)\}$

We shall use the notation $\psi_{\zeta,J}$ for ϕ_J. We remark:

- the rôle of I that commands the cardinality of the finite set of the $\psi_{\xi i}$'s;
- the uniform indexation by the set of all finite parts of ω of the set of trees of a pseudo-design ψ_ζ;
- the rôle of the index ζ in ψ_ζ, that commands the shape of its initial negative actions.

The syntax makes it clear that a positive design is a tree the root of which (if different from Ω and $\maltese$) is labelled by a positive action $(+,\xi,I)$ and has branches indexed by $i \in I$ and $J \in \mathcal{P}_f(\omega)$. The corresponding subtrees are grouped in a layered way (cf. Remark 3.3): for each i, the subtrees intexed by i, J, for J varying over $\mathcal{P}_f(\omega)$ form a negative design $\psi_{\xi i}$, each of whose trees has a root labelled with the corresponding negative action $(-,\xi i,J)$. Note that each negative root $(-,\xi i,J)$ has a unique son: designs are strategies, i.e., Player's answers to Opponent's moves are unique.

The following definition collects a few useful designs expressed in our syntax.

Definition 3.4 *We set:*

$$\begin{array}{l}
\mathit{Dai} = \maltese \\
\mathit{Dai}^-_\xi = \{(-,\xi,J)\maltese \mid J \in \mathcal{P}_f(\omega)\} \\
\mathit{Skunk}_\xi = \{(-,\xi,J)\Omega \mid J \in \mathcal{P}_f(\omega)\} \\
\mathit{Skunk}^+_{(\xi,I)} = (+,\xi,I)\cdot\{\{(-,\xi i,J)\Omega \mid J \in \mathcal{P}_f(\omega)\} \mid i \in I\} \\
\mathit{Ram}_{(\xi,I)} = (+,\xi,I)\cdot\{\{(-,\xi i,J)\maltese \mid J \in \mathcal{P}_f(\omega)\} \mid i \in I\} \\
\mathit{Dir}_{\mathcal{N}} = \{(-,\epsilon,I)\maltese \mid I \in \mathcal{N}\} \cup \{(-,\epsilon,I)\Omega \mid I \notin \mathcal{N}\}
\end{array}$$

The skunk is an animal that stinks and has therefore degree zero of sociability, where sociability is measured by the capacity for compromise, expressed by $\maltese$ in ludics (see Exercise 4.6).

Not every term of this syntax corresponds to a design, whence our terminology of pseudo-design. We can recover those pseudo-designs that come from designs as the “correctly typed” ones. More precisely, designs can be defined either as typing proofs of pseudo-designs (our first definition earlier in the section), or as typable pseudo-designs. Girard uses the French words “dessin” and “dessein” to name these two flavours, respectively. This is reminiscent of the distinction between typing “à la Church” and typing “à la Curry”, respectively. Here is the “type system”:

$$\frac{(\vdash \Lambda \text{ well formed})}{\Omega : (\vdash \Lambda)} \qquad \frac{(\vdash \Lambda \text{ well formed})}{\maltese : (\vdash \Lambda \qquad)}$$

$$\frac{\cdots\ \psi_{\xi i} : (\xi i \vdash \Lambda_i)\ \cdots \qquad \left(\begin{array}{l} i \in I \\ \forall\, i \;\; \Lambda_i \subseteq \Lambda \\ \forall\, i_1, i_2 \;\; (i_1 \neq i_2 \Rightarrow \Lambda_{i_1} \cap \Lambda_{i_2} = \emptyset) \\ \vdash \xi, \Lambda \text{ well formed} \end{array}\right)}{(+, \xi, I) \cdot \{\psi_{\xi i} \mid i \in I\} : (\vdash \xi, \Lambda)}$$

$$\frac{\cdots\ \phi_J : (\vdash \zeta \star J, \Lambda_J)\ \cdots \qquad \left(\begin{array}{l} J \in \mathcal{P}_f(\omega) \\ \forall\, J \;\; \Lambda_J \subseteq \Lambda \\ \zeta \vdash \Lambda \text{ well formed} \end{array}\right)}{\{(-, \zeta, J)\phi_J \mid J \in \mathcal{P}_f(\omega)\} : (\zeta \vdash \Lambda)}$$

We also write $\Psi : \{\ldots, (\zeta_i \vdash \Lambda_i), \ldots\}$ if $\Psi = \{\psi_{\zeta_1}, \ldots, \psi_{\zeta_n}\}$ and $\psi_{\zeta_i} : (\zeta_i \vdash \Lambda_i)$ for all i (finite number of bases).

Definition 3.5 *A design is a typable pseudo-design $\neq \Omega$. The pseudo-design Ω, which is obviously typable ($\Omega : (\vdash \Lambda)$, for any Λ), also denoted by Fid, is called the partial design.*

Conversely, the following algorithm takes as input a finite positive pseudo-design ϕ and returns a base $\vdash \Lambda$ such that $\phi : (\vdash \Lambda)$ if it is a design and otherwise returns *FAIL*. Actually, the inductive load requires taking as arguments ϕ together with a (well-formed) base $\vdash \Gamma$. Initially, we take Γ empty. The algorithm is described by a formal system using the following notation:

$$\phi : (\vdash \Gamma, ?) \longrightarrow \phi : (\vdash \Lambda)$$

which reads as follows: the algorithm, when given ϕ and Γ as inputs, finds Λ such that $\vdash \Lambda$ (cf. Remark 3.2), $\Gamma \subseteq \Lambda$, and $\phi : (\vdash \Lambda)$. The failure case is written as $\phi : (\vdash \Gamma, ?) \longrightarrow$ *FAIL*. The induction is on ϕ. If the design is infinite, we apply the algorithm to finite approximations – we can obviously not decide whether an infinite pseudo-design is a design.

$$\frac{}{\Omega : (\vdash \Gamma, ?) \longrightarrow \Omega : (\vdash \Gamma)} \qquad \frac{}{\maltese : (\vdash \Gamma, ?) \longrightarrow \maltese : (\vdash \Gamma)}$$

In the following rules, the notation $\Psi_{\xi i,J}$ stands for ϕ_J where $\Psi = \{\psi_{\xi i} \mid i \in I\}$ and $\psi_{\xi i} = \{(-,\zeta,J)\phi_J \mid J \in \mathcal{P}_f(\omega)\}$.

$$\frac{\cdots\ \Psi_{\xi i,J} : (\vdash \xi i \star J, ?) \longrightarrow \Psi_{\xi i,J} : (\vdash \xi i \star J, \Lambda_{i,J})\ \cdots \quad \left(\begin{array}{l} i \in I, J \in \mathcal{P}_f(\omega) \\ \forall\, i_1, i_2\ \ (i_1 \neq i_2 \Rightarrow \Lambda_{i_1,J_1} \cap \Lambda_{i_2,J_2} = \emptyset) \\ \vdash \xi, \Gamma \setminus \{\xi\} \bigcup_{i \in I, J \in \mathcal{P}_f(\omega)} \Lambda_{i,J} \text{ well formed} \end{array}\right)}{(+,\xi,I)\cdot\Psi : (\vdash \Gamma, ?) \longrightarrow (+,\xi,I)\cdot\Psi : (\vdash \xi, \Gamma \setminus \{\xi\} \bigcup_{i \in I, J \in \mathcal{P}_f(\omega)} \Lambda_{i,J})}$$

$$\frac{\Psi_{\xi i,J} : (\vdash \xi i \star J, ?) \longrightarrow \mathit{FAIL}}{(+,\xi,I)\cdot\Psi : (\vdash \Gamma, ?) \longrightarrow \mathit{FAIL}}$$

$$\frac{\begin{array}{l} \Psi_{\xi i_1,J_1} : (\vdash \xi i_1 \star J_1, ?) \longrightarrow \Psi_{\xi i_1,J_1} : (\vdash \xi i_1 \star J_1, \Lambda_{i_1,J_1}) \\ \Psi_{\xi i_2,J_2} : (\vdash \xi i_2 \star J_2, ?) \longrightarrow \Psi_{\xi i_2,J_2} : (\vdash \xi i_2 \star J_2, \Lambda_{i_2,J_2}) \end{array} \quad (i_1 \neq i_2 \text{ and } \Lambda_{i_1,J_1} \cap \Lambda_{i_2,J_2} \neq \emptyset)}{(+,\xi,I)\cdot\Psi : (\vdash \Gamma, ?) \longrightarrow \mathit{FAIL}}$$

$$\frac{\cdots\ \Psi_{\xi i,J} : (\vdash \xi i \star J, ?) \longrightarrow \Psi_{\xi i,J} : (\vdash \xi i \star J, \Lambda_{i,J})\ \cdots \quad (\vdash \xi, \Gamma \setminus \{\xi\} \bigcup_{i \in I, J \in \mathcal{P}_f(\omega)} \Lambda_{i,J} \text{ not well formed})}{(+,\xi,I)\cdot\Psi : (\vdash \Gamma, ?) \longrightarrow \mathit{FAIL}}$$

The first rule of failure, which has a unique premise (for a given $i \in I$ and a given J), expresses failure propagation. The second rule says that failure occurs when linearity (actually, affinity) is violated. The third rule controls the well-formedness of bases. The following statement is easy to prove.

Proposition 3.6 *The above algorithm, with ϕ, Γ as arguments, terminates if and only if there exists Λ' such that $\phi : (\vdash \Lambda')$ and $\Gamma \subseteq \Lambda'$. The base $\vdash \Lambda$ returned by the algorithm is the smallest base that satisfies these conditions.*

Proof (indication). That the algorithm returns the minimum base follows from the fact that it collects unions of bases, which are themselves least upper bounds. $\square$

Remark 3.7 *The relation between the two flavours of designs (dessins and desseins, or typing proofs and typable pseudo-designs) is of the same nature as the relation between proofs in natural deduction expressed in terms of sequents and proofs in natural deduction expressed as trees of formulas with marked hypotheses, or as the relation between linear logic proofs in the sequent calculus formulation (cf. part I, section 2) and proof nets (cf. section 1). While proof nets allow us to identify many sequent calculus proofs, here, as we have just seen, the two points of view (explicit sequents or not) happen to be essentially the same, up to weakening. This is due to the focalization, which already eliminated many sequent calculus proofs!*

Remark 3.8 *We can define an injection* $^\bullet$ *mapping pseudo-designs to sets of chronicles ending with a positive action as follows:*

$$\Omega^\bullet = \emptyset \qquad ((+,\xi,I)\cdot\{\psi_{\xi i} \mid i\in I\})^\bullet = \{\epsilon\}\bigcup_{i\in I}\{(+,\xi,I)r \mid r\in\psi^\bullet_{\xi i}\}$$
$$\maltese^\bullet = \{\maltese\} \qquad \{(-,\zeta,J)\phi_J \mid J\in\mathcal{P}_f(\omega)\}^\bullet = \bigcup_{J\in\mathcal{P}_f(\omega)}\{(-,\zeta,J)r \mid r\in\phi^\bullet_J\}$$

Note that the Ω*'s disappear during this transformation. In [36], the designs-desseins are defined as sets of chronicles, and properties characterizing the image of the set of designs by the function* $^\bullet$ *are given. These properties include:*

- coherence *: the maximal prefix of any two chronicles of the design which are not prefix of each other ends with a positive action;*
- focalization*: a negative action which is not at the root must be of the form* $(-,\xi i,J)$ *and its father must be of the form* $(+,\xi,I)$*, with* $i\in I$*;*
- subaddress*: a positive action* $(+,\xi,I)$ *is either such that* ξ *is an address on the positive side of the base, or there exist* $\xi', i,$ *and* J *such that* $\xi=\xi' i$*,* $i\in J$*, and* $(-,\xi',J)$ *occurs on the chronicle between* $(+,\xi,I)$ *and the root.*

Designs have an implicit pointer structure: for each action whose focus is some ξi, one looks for the action of focus ξ below it in the forest or tree representation of the design, if any, and we say that the former *points* to the latter, or is *bound* to the latter. Otherwise, we say that the action of focus ξi occurs *free* in the design. It is easily seen that for well-typed designs the two actions must have opposite signs, and that a free action is necessarily positive and that its focus belongs to the right hand side of the base of the design (cf. Remark 3.2). Moreover, negative actions are always bound and always point to the immediately preceding positive action. These are typical features of Hyland and Ong (HO) games [43] (see [25] for precise comparisons). In [16, 17], a general formalism of such (untyped) HO style strategies over a given alphabet of moves, called *abstract Böhm trees*, has been developped. Opponent's moves are letters of the alphabet, while Player's moves are pairs noted $[a,\overset{\kappa}{\hookleftarrow}]$, where a is a letter of the alphabet and where κ is either the symbol $\_$ that indicates that the move is free, or a natural number that counts the number of Opponent's moves encountered before reaching the binder. The designs are abstract Böhm trees constructed on the alphabet consisting of all the pairs (i,I) of a natural number and a finite set of natural numbers. The general shape is as follows (rotating the tree clockwise by 90 degrees):

$$\left\{\begin{array}{l} \vdots \\ (j_1,J_1')\Omega \\ \vdots \\ (j_1,J_1)[(j_2,I_1),\overset{\kappa_2}{\hookleftarrow}] \left\{\begin{array}{l}\vdots\\ (i_1,J_2')\Omega\\ \vdots \\ (i_1,J_2)[(j_3,I_2),\overset{\kappa_3}{\hookleftarrow}]\left\{\ \vdots\right. \\ \vdots\end{array}\right. \\ \vdots\end{array}\right.$$

The difference with the (raw) designs is that we use relative addressses rather than absolute ones. The pointers allow us to reconstruct the full addresses. Here is how the fax looks like, first, as a design-dessein:

$$\left\{\begin{array}{l}\vdots\\ \\ (-,\xi,J_1)(+,\xi',J_1)\left\{\begin{array}{l}\vdots\\ \\ (-,\xi' j_1,J_2)(+,\xi j_1,J_2)\left\{\begin{array}{l}\vdots\\ (-,\xi j_1 j_2,J_3)(+,\xi' j_1 j_2,J_3)\left\{\ \vdots\right.\\ \vdots\end{array}\right.\\ \\ \vdots\end{array}\right.\\ \\ \vdots\end{array}\right.$$

and then as abstract Böhm tree (where ξ,ξ' end with i,i', respectively):

$$\left\{\begin{array}{l}\vdots\\ \\ (i,J_1)[(i',J_1),\overset{-}{\hookleftarrow}]\left\{\begin{array}{l}\vdots\\ \\ (j_1,J_2)[(j_1,J_2),\overset{1}{\hookleftarrow}]\left\{\begin{array}{l}\vdots\\ (j_2,J_3)[(j_2,J_3),\overset{1}{\hookleftarrow}]\left\{\ \vdots\right.\\ \vdots\end{array}\right.\\ \\ \vdots\end{array}\right.\\ \\ \vdots\end{array}\right.$$

The pointer 1 in, say, $[(j_1,J_2),\overset{1}{\hookleftarrow}]$, formalizes the fact that the positive action $(+,\xi j_1,J_2)$ is not bound to the negative action that immediately precedes it, but, one level up, to $(-,\xi,J_1)$: Girard calls this subfocalization.

Designs and λ-calculus. If we restrict both designs and (a variant of) λ-calculus, then we can get a bijective correspondence, as we show now. The interest is twofold. First, this correspondence allows us to connect designs to the well-established tradition of the λ-calculus. But also, it has suggested us to extend the correspondence to all designs, giving rise to a term language for designs. We start with the restricted correspondence, which applies to slices, defined next.

Definition 3.9 (Slice) *A slice is a design in which each application of the rule $(-,\xi,\mathcal{N})$ is such that $\mathcal{N}$ is a singleton. In terms of designs as sets of chronicles, this rule says that if the design contains two chronicles $r(-,\xi,I_1)$ and $r(-,\xi,I_2)$, then $I_1 = I_2$.*

(It is easy to check that slices arise from purely multiplicative proofs.) We introduce variables and proof terms for slices simultaneously, as follows:

$$\frac{}{\Omega : (\vdash \Lambda)} \qquad \frac{}{\maltese : (\vdash \Lambda)}$$

$$\frac{P : (\vdash \{x_j : \zeta j \mid j \in J\}, \Lambda') \quad (\Lambda' \subseteq \Lambda)}{\lambda\{x_j \mid j \in J\}.P : (\zeta \vdash \Lambda)}$$

$$\frac{\cdots\ M_i : (\xi i \vdash \Lambda_i)\ \cdots \qquad \left(\begin{array}{l} i \in I \\ \forall\, i\ \ \Lambda_i \subseteq \Lambda \\ \forall\, i_1, i_2\ \ (i_1 \neq i_2 \Rightarrow \Lambda_{i_1} \cap \Lambda_{i_2} = \emptyset) \\ x \text{ fresh} \end{array}\right)}{x\{M_i \mid i \in I\} : (\vdash x : \xi, \Lambda)}$$

In the first (resp. second) rule, the cardinal of J (resp. I) gives the number of head λ's (resp. the number of arguments of the variable). The bases $\xi \vdash \Lambda$ and $\vdash \Lambda$ are now such that Λ consists of a set of variable declarations of the form $x : \zeta$. Let us place the λ-calculus style syntax that we just introduced (on the left below) in perspective with the syntax of normal forms of the λ-calculus (on the right below):

$$\begin{array}{ll} M ::= \lambda\{x_j \mid j \in J\}.P & \qquad M ::= \lambda x_1 \ldots x_m.P \\ P ::= x\{M_i \mid i \in I\} \mid \Omega \mid \maltese & \qquad P ::= x M_1 \ldots M_n \end{array}$$

The difference lies in the fact that we have freed ourselves from the sequential order of application of the ordinary λ-calculus, and that we now have explicit addresses for the arguments of the application: i is the (relative) address of M_i.

Remark 3.10 *Combining the bijection from slices to terms just given with the representation of designs as HO style trees with pointers given above, what we have obtained is a correspondence of the same nature as the bijection between λ-terms and λ-terms in De Bruijn notation (see e.g. [1]). More precisely, it is exactly a bijection of this kind, replacing De Bruijn indices by other indices that we have called Böhm indices in [17] (and that have appeared in previous works of Danos-Regnier and Huet), which are pairs of indices of the form $[i, \overset{j}{\hookleftarrow}]$ where j counts the number of λ's separating the variable from its binder and where i serves to identify which among the variables collectively bound by the binder it is. For example, the Böhm indices of x_1 and x_2 in $\lambda\{x_1, x_2\}.x_1\{\lambda\{y_1\}.x_2\{\}\}$ are $[1, \overset{0}{\hookleftarrow}]$ and $[2, \overset{1}{\hookleftarrow}]$, respectively.*

Remark 3.11 *It is easily checked that the typable terms are exactly the affine terms, i.e., those in which any variable occurs at most once.*

We next move on and give terms for *arbitrary designs.*

$$\frac{}{\Omega : (\vdash \Lambda)} \qquad \frac{}{\maltese : (\vdash \Lambda)}$$

$$\frac{\cdots\ M_i : (\xi i \vdash \Lambda_i)\ \cdots \qquad \left(\begin{array}{l} i \in I \\ \forall i\ \ \Lambda_i \subseteq \Lambda \\ \forall i_1, i_2\ \ (i_1 \neq i_2 \Rightarrow \Lambda_{i_1} \cap \Lambda_{i_2} = \emptyset) \\ x \text{ fresh} \end{array}\right)}{(x.I) \cdot \{M_i \mid i \in I\} : (\vdash x : \xi, \Lambda)}$$

$$\frac{\cdots\ P_J : (\vdash \{x_j : \zeta j \mid j \in J\}, \Lambda_J)\ \cdots \qquad \left(\begin{array}{l} J \in \mathcal{P}_f(\omega) \\ \forall J\ \ \Lambda_J \subseteq \Lambda \end{array}\right)}{\{J = \lambda\{x_j \mid j \in J\}.P_J \mid J \in \mathcal{P}_f(\omega)\} : (\zeta \vdash \Lambda)}$$

Forgetting now about typing rules, we have the following (untyped) syntax:

$$\begin{array}{l} M ::= \{J = \lambda\{x_j \mid j \in J\}.P_J \mid J \in \mathcal{P}_f(\omega)\} \\ P ::= (x \cdot I)\{M_i \mid i \in I\} \mid\mid \Omega \mid\mid \maltese \end{array}$$

Since we now have two syntaxes for (pseudo-)designs, we shall refer to the two syntaxes as the abstract syntax and the concrete syntax, respectively (in the order in which they have appeared in the paper). As an illustration, here is the fax (recursively) expressed in concrete syntax:

$$Fax_{\xi, x:\xi'} \quad = \quad \{I = \lambda\{y_i \mid i \in I\}.(x \cdot I)\{Fax_{\xi' i, y_i : \xi i} \mid i \in I\} \mid I \in \mathcal{P}_f(\omega)\}\ .$$

In this form, the fax appears as the variable x, in all its possible (hereditarily) η-expansions. Note the remarkable use of the additive flavour of designs: the fax is ready for *any* ramification I, i.e. for any possible η-expansion (see Exercise 4.10).

Ordering designs. Two rather natural partial orders can be defined on designs:

• The usual partial information ordering on trees: to increase information, one replaces an Ω with a tree $\neq \Omega$ (in the first representation of designs, this amounts to extend a set $\mathcal{N}$ in a negative rule). This is reminiscent of the ordering between stable functions defined by the inclusion of their traces (see e.g. [4, chapter 12]).

• The second ordering $\sqsubseteq$ is obtained by adding a second rule for increasing: replace a subtree by a $\maltese$. We shall see that this ordering characterizes the observational order between designs, so we shall call it the observational ordering. It is reminiscent of the pointwise ordering between Scott continuous functions. This ordering is defined formally as follows:

$$\frac{}{\Omega \sqsubseteq \phi} \qquad\qquad \frac{}{\phi \sqsubseteq \maltese}$$

$$\frac{\cdots\quad \psi_{\xi i} \sqsubseteq \psi'_{\xi i} \quad \cdots \qquad (i \in I)}{(+, \xi, I) \cdot \{\psi_{\xi i} \mid i \in I\} \sqsubseteq (+, \xi, I) \cdot \{\psi'_{\xi i} \mid i \in I\}} \qquad \frac{\cdots\quad \phi_J \sqsubseteq \phi'_J \quad \cdots \qquad J \in \mathcal{P}_f(\omega)}{\{\ldots, (-, \zeta, J)\phi_J, \ldots\} \sqsubseteq \{\ldots, (-, \zeta, J)\phi'_J, \ldots\}}$$

$$\frac{}{\phi \sqsubseteq \phi} \qquad \frac{\phi_1 \sqsubseteq \phi_2 \quad \phi_2 \sqsubseteq \phi_3}{\phi_1 \sqsubseteq \phi_3}$$

We write $\phi_1 \leq^R \phi_2$ (resp. $\phi_1 \leq^L \phi_2$) if $\phi_1 \sqsubseteq \phi_2$ has been proved without using the axiom $\phi \sqsubseteq \maltese$ (resp. $\Omega \sqsubseteq \phi$). Thus, $\phi_1 \leq^R \phi_2$ is the stable ordering, which we also denote as $\phi_1 \subseteq \phi_2$ (see Exercise 3.15).

Remark 3.12 *We have the following decomposition property: if $\phi_1 \sqsubseteq \phi_2$, then there exists a ϕ such that $\phi_1 \leq^L \phi \leq^R \phi_2$, obtained by first cutting some subtrees and replacing them by a $\maltese$, and then by adding some subtrees at some Ω's. Note that this is a quite natural way of proceeding, since one might do useless work otherwise, by adding subtrees using axiom $\Omega \sqsubseteq \phi$ that could then be removed while using axiom $\phi \sqsubseteq \Omega$. The decomposition is not unique, as we have both $\Omega \leq^L \Omega \leq^R \maltese$ and $\Omega \leq^L \maltese \leq^R \maltese$ (see Exercise 3.17). Remarkably enough, similar decompositions have been discovered and exploited elsewhere:*

- *by Winskel in his analysis of Berry's stable ordering and bi-domains (see [18]), and*
- *by Laird in his extensional account of the model of sequential algorithms [45, 19].*

Exercise 3.13 *Complete the characterization of designs as sets of chronicles (cf. Remark 3.8). (Hint: The only missing conditions are the ones ensuring that the designs are affine.)*

Exercise 3.14 *Is there a sense in which the concrete syntax and the abstract syntax of pseudo-designs or of designs are in bijective correspondence? (Hints: Pseudo-designs in abstract syntax are more permissive, e.g., they do not exclude the possibility of a move $(+, \xi i, J)$ "pointing" to a move $(+, \xi, I)$. On the other hand, "raw" terms in the concrete syntax are actually all "typable", in a system where the affinity constraint is relaxed.)*

Exercise 3.15 *Show that, on designs as sets of chronicles (cf. Remark 3.8), the stable ordering is set inclusion, and that the extensional ordering can be characterized by the following property: $\phi_1 \sqsubseteq \phi_2$ if and only if whenever $r_1 = r\,(-, \xi, I) r'_1 \in \phi_1$ and $r_2 = r\,(-, \xi, I) r'_2 \in \phi_2$ diverge after some action $(-, \xi, I)$, then either $r'_1 = \Omega$ or $r'_2 = \maltese$.*

Exercise 3.16 *We define formally (infinite-depth) designs as ideals (i.e., downwards closed directed subsets of designs) with respect to $\subseteq$. Show that the equation defining the fax admits a least fixed point which is such an ideal.*

Exercise 3.17 *Let $\leq$ be the intersection of the partial orders $\leq^L$ and $\leq^R$. Show the following more precise formulation of the decomposition property: if $\phi_1 \sqsubseteq \phi_2$, then there exist ϕ_{min} and ϕ_{max} such that:*

$$\forall \phi \ \ ((\phi_1 \leq^L \phi \leq^R \phi_2) \Leftrightarrow \phi_{min} \leq \phi \leq \phi_{max}) \,.$$

4 Normalization of designs

Normalization is a general machinery that applies to all abstract Böhm trees, hence to designs in particular. It can be described in various equivalent ways, see [16]. In the affine case, the execution is simpler (no copying involved), so we shall describe here (in three disguises) an abstract machine for designs taking the affine nature of designs into account. The first two presentations of the normalization engine are based on the abstract and concrete syntax for designs, respectively, and use the framework of environment machines, whose states have two components: the code to be run, and the environment or context or counter-design. The third version is more visual, and features a simple token game. Its main advantage is that it makes the very useful notion of the part of a design visited during normalization explicit.

We embark on the first description, for which we shall need the following terminology.

Notation 4.1 *We denote by* Ψ *a finite set* $\{\psi_{\zeta_1}, \ldots, \psi_{\zeta_n}\}$ *of negative pseudo-designs, with the* ζ_i*'s pairwise disjoint. We say that* Ψ accepts *(resp. does not accept) an address* ζ *if* $\zeta \in \{\zeta_1, \ldots, \zeta_n\}$ *(resp.* $\zeta \notin \{\zeta_1, \ldots, \zeta_n\}$*). If* Ψ *accepts* ζ*, i.e.,* $\zeta = \zeta_j$ *for some* j*, we use the notation* Ψ_ζ *for* ψ_{ζ_j}*. We also note simply* ψ *for* $\{\psi\}$*. Finally, if* Ψ *accepts* ζ*, we note* $\Psi \setminus \zeta = \Psi \setminus \{\psi_\zeta\}$*.*

We start with a design $\phi : (\vdash \xi, \Lambda_1)$ to be normalized against a design $\psi : (\xi \vdash \Lambda_2)$ (with Λ_1 and Λ_2 pairwise disjoint), which we call its *counter-design*. The normalization should produce a design based on $\vdash \Lambda_1, \Lambda_2$ (think of a cut rule).

The machine has states of the form $\langle \phi \mid \Psi \rangle$, and execution consists in applying the following state transformation, which we call the *weak reduction* rule (by contrast to strong reduction, introduced below):

$$\text{(R)} \quad \langle (+,\xi,I)\cdot\Psi' \mid \Psi \rangle \longrightarrow \langle \Psi_{\xi,I} \mid \Psi' \cup (\Psi \setminus \xi) \rangle \qquad (\Psi \text{ accepts } \xi)$$

Notice that we throw away all of Ψ_ξ except for $\Psi_{\xi,I}$. In particular, the negative actions $(-,\xi,J)$ of Ψ are not needed. That we can do this safely will be justified by Proposition 4.18.

The initial state is $\langle \phi \mid \{\psi\} \rangle$. It is important to keep $\Psi \setminus \xi$ in order to handle subfocalizations, as illustrated by the example below (where $\zeta = \xi i_1 i$, for some $i \in I_1$):

$$\phi \quad = \quad \dfrac{\cdots \quad \dfrac{\dfrac{\cdots}{(+,\zeta,J)}}{(-,\xi i_1,I_1)} \quad \cdots \quad \dfrac{\cdots}{(-,\xi i_2,I_2)} \quad \cdots}{(+,\xi,I)} \qquad\qquad \psi \quad = \quad \cdots \quad \dfrac{\dfrac{\cdots \quad \dfrac{\dfrac{\cdots}{(+,\xi i_2,I_2)}}{(-,\zeta,J)} \quad \cdots}{(+,\xi i_1,I_1)}}{(-,\xi,I)} \quad \cdots$$

or:

$\phi = (+,\xi,I)\cdot\Psi'$
  with $\Psi' = \{\ldots,(-,\xi i_1,I_1)\phi_2,\ldots,(-,\xi i_2,I_2)\cdots,\ldots\}$
  with $\phi_2 = (+,\zeta,J)\cdot\Psi_2$

$\psi = \{\ldots,(-,\xi,I)\phi_1,\ldots\}$
  with $\phi_1 = (+,\xi i_1,I_1)\cdot\Psi_1$
  with $\Psi_1 = \{\ldots,(-,\zeta,J)\phi_3,\ldots\}$
  with $\phi_3 = (+,\xi i_2,I_2)\cdots$

Execution goes as follows:

$$\begin{array}{lll} \langle \phi \mid \psi \rangle & \longrightarrow & \langle \phi_1 \mid \Psi' \cup (\psi \setminus \xi) \rangle = \langle \phi_1 \mid \Psi' \rangle \\ & \longrightarrow & \langle \phi_2 \mid \Psi_1 \cup (\Psi' \setminus \xi i_1) \rangle \\ & \longrightarrow & \langle \phi_3 \mid \Psi_2 \cup (\Psi_1 \setminus \xi) \cup (\Psi' \setminus \xi i_1) \rangle \end{array}$$

At state $\langle \phi_3 \mid \Psi_2 \cup (\Psi_1 \setminus \xi) \cup (\Psi' \setminus \xi i_1) \rangle$, it is essential to have Ψ' available, as the head action of ϕ_3 corresponds to a head action of Ψ' and not of Ψ_2, i.e., when normalization reaches $(+, \xi i_2, I_2)$, the corresponding negative action on the left is not found above $(+, \zeta, J)$, but above $(+, \xi, I)$.

The machine stops when it reaches an Ω (which one should interpret as waiting for more information from the context), a $\maltese$ (that indicates convergence), or a state $\langle (+, \xi_n, I_n) \cdot \Psi'_n \mid \Psi_n \rangle$ such that Ψ_n does not accept ξ_n (in λ-calculus terminology, this means reaching a head variable). In all three cases, we have found the root of the normal form of $\langle \phi \mid \psi \rangle$ (see also Remark 4.4).

We can go on, and perform "strong reduction", by relaunching the machine in order to get progressively, on demand, the chronicles of the normal form, and not only the first positive action of the normal form. This can be formalized by indexing the state by the chronicle q of the normal form under exploration. The index remains invariant in the rule (R), which is thus reformulated as:

$$\text{(R)} \quad \langle (+, \xi, I) \cdot \Psi' \mid \Psi \rangle_q \longrightarrow \langle \Psi_{\xi, I} \mid \Psi' \cup (\Psi \setminus \xi) \rangle_q \qquad (\Psi \text{ accepts } \xi)$$

Initially, the index is the empty chronicle ϵ. The chronicle is extended using the following rule (S stands for strong reduction):

$$\text{(S)} \quad \langle (+, \xi, I) \cdot \Psi' \mid \Psi \rangle_q \longrightarrow \langle \Psi'_{\xi i, J} \mid \Psi \rangle_{q(+,\xi,I)(-,\xi i,J)} \qquad (\Psi \text{ does not accept } \xi \text{ and } i \in I)$$

Note that this rule is non-deterministic: the choice of i, J is left to the Opponent of the normal form, he has to say which branch of the normal form he intends to explore.

The two other termination cases remain termination cases in this setting, and are formalized as follows:

$$\begin{array}{l} \langle \Omega \mid \Psi \rangle_q \longrightarrow\ !\, q\, \Omega \\ \langle \maltese \mid \Psi \rangle_q \longrightarrow\ !\, q \maltese \end{array}$$

Read $!\, q\, \Omega$ (resp. $!\, q \maltese$) as an output, saying that the chronicle $q\, \Omega$ (resp. $q \maltese$) belongs to the normal form. As for rule (S), we can read that $q(+, \xi, I)$ belongs to the normal form, but in addition we know that the normal form contains a chronicle of the form $q(+, \xi, I)(-, \xi i, J)\kappa_{i,J}$ for all $i \in I$ and J, and may start an exploration in order to find any of these actions $\kappa_{i,J}$.

We denote the normal form of $\langle \phi \mid \{\psi\} \rangle$ by $[\![\phi, \{\psi\}]\!]$, or simply $[\![\phi, \psi]\!]$, and more generally, we denote the normal form of $\langle \phi \mid \Psi \rangle$ (see Definition 4.2) by $[\![\phi, \Psi]\!]$. As a set of chronicles, it is obtained by collecting all the results of execution:

$$\begin{array}{lll} [\![\phi, \Psi]\!] & = & \{q(+, \xi, I) \mid \exists\, \Psi', \Psi''\ \langle \phi \mid \Psi \rangle_\epsilon \longrightarrow^\star \langle (+, \xi, I) \cdot \Psi' \mid \Psi'' \rangle_q\} \\ & & \cup\, \{q\, \Omega \mid \langle \phi \mid \Psi \rangle_\epsilon \longrightarrow^\star\ !\, q\, \Omega\} \\ & & \cup\, \{q \maltese \mid \langle \phi \mid \Psi \rangle_\epsilon \longrightarrow^\star\ !\, q \maltese\} \\ & & \cup\, \{q\, \Omega \mid \text{computation does not terminate and } q \text{ is the maximal index reached}\} \end{array}$$

Note that we have added a second case of divergence: the divergence due to the cases where the machine does not terminate. To be more precise, we are speaking of non-termination due to rule

(R) only (at index q), not of non-termination due to a potentially infinite branch of the normal form (recall that we are in a demand-driven setting).

Therefore, there are two sorts of Ω in the normal form: those that are "created" (by the magic of a mathematical formula, not by some computation observed in a finite time), and those that come from the $\Omega's$ explicitly present in ϕ or Ψ. The overloading of the two situations is really what makes the difference between Ω and ✠ that otherwise behave in a dual way (cf. e.g. the definition of $\leq^L$ and $\leq^R$). It is also the reason for the name *Fid* given by Girard to the partial design Ω: you must have faith (*Fides* in Latin) to wait for a result that might never come.

We still have to establish that $[\![\phi, \Psi]\!]$ is a design, not only a pseudo-design, and hence we should exhibit a typing proof of $[\![\phi, \Psi]\!]$. We first have to type the states of the machine, which we can also call nets, using Girard's terminology inspired from proof nets. We have seen that a state, or net, consists of a positive design ϕ and a collection Ψ' of negative designs. We stick to this situation in the following definition.

Definition 4.2 *A (positive) net is given by a positive design $\phi : (\vdash \Lambda)$, called the principal design of the net, and a finite set Ψ of negative designs such that:*

1. *the addresses appearing in the bases of ϕ and of the elements of Ψ are pairwise disjoint or equal;*
2. *an address may appear at most twice in these bases;*
3. *if an address appears twice, then it appears once on the left of $\vdash$ and once on the right of $\vdash$, thus forming a* cut*;*
4. *The graph whose set of vertices is $\{\phi\} \cup \Psi$ (or the associated set of bases) and whose edges are the cuts is acyclic.*

We denote the net by $\langle \phi \mid \Psi \rangle$.

Girard requires moreover connectivity. We do not make this requirement explicit here, but it is clear that what interests us is the connected component of ϕ. In this connected component, all the ξ's on the left of $\vdash$ are cut. This is easily shown by induction on the length of a path starting at $\vdash \Lambda$. The only interesting case is when the path has reached a node of the form $\xi' \vdash \Lambda'$ and proceeds from there to a node of the form $\xi'' \vdash \Lambda''$ with a cut on ξ'. But then by induction and by condition 2, the latter node must have been visited by the path earlier and must be the one that witnesses that ξ' is cut. Therefore, we know by induction that ξ'' is cut.

Hence all the non-cut addresses of the designs of the connected component stand to the right of $\vdash$, in ϕ or in one of the elements of Ψ. Let Λ be the set of these addresses. We set $\langle \phi \mid \Psi \rangle : (\vdash \Lambda)$. (It is thus the connected component that we actually type, but it is more convenient not to throw away explicitly the elements of the other connected components.)

Proposition 4.3 *If $\langle \phi \mid \Psi \rangle$ is a net and if $\langle \phi \mid \Psi \rangle : (\vdash \Lambda)$, then $[\![\phi, \Psi]\!] : (\vdash \Lambda)$.*

PROOF. We only prove that rule (R) preserves typing, which is enough to prove the statement in the case of a net of type $\vdash$ (see Remark 4.4). Suppose that $\langle (+, \xi, I) \cdot \Psi' \mid \Psi \rangle : (\vdash \Lambda)$. We have to prove $\langle \Psi_{\xi,I} \mid \Psi' \cup (\Psi \setminus \xi) \rangle : (\vdash \Lambda)$.

Condition 4. We have $(+, \xi, I) \cdot \Psi' : (\vdash \xi, \ldots)$, and $\Psi : (\ldots, (\xi \vdash \ldots), \ldots)$, hence $\Psi' : \{(\xi i \vdash \ldots) \mid i \in I\}$ and $(\Psi)_{\xi,I} : (\vdash \xi \star I, \ldots)$. Thus we remove the bases $(\vdash \xi, \ldots)$ and $(\xi \vdash \ldots)$ from the graph and replace them by the base $(\vdash \xi \star I, \ldots)$ and the bases $(\xi i \vdash \ldots)$. The edge that connected $(\vdash \xi, \ldots)$ to $(\xi \vdash \ldots)$ is replaced by edges that connect $(\vdash \xi \star I, \ldots)$ to each of the $(\xi i \vdash \ldots)$'s. Note that these new edges form a connnected configuration. The edges that once arrived to $(\xi \vdash \ldots)$ now arrive to $(\vdash \xi \star I, \ldots)$, and those that arrived to $(\vdash \xi, \ldots)$ are now either disconnected or arrive to one of the $(\xi i \vdash \ldots)$'s. If there was a cycle in the new graph, one would then easily construct one in the former graph (to which we can go back by amalgamating the bases $(\xi i \vdash \ldots)$).

Conditions 2 and 3. The two occurrences of ξ disappear, and are replaced by pairs of occurrences of each ξi. $\square$

Remark 4.4 *A consequence of the previous proposition is that when the weak reduction machine starting from $\langle \phi \mid \Psi \rangle : (\vdash \Lambda)$ stops in a state $\langle (+, \xi_n, I_n) \cdot \Psi'_n \mid \Psi_n \rangle$, where Ψ_n does not accept ξ_n, then $\xi_n \in \Lambda$. Indeed, by type preservation we have $\langle (+, \xi_n, I_n) \cdot \Psi'_n \mid \Psi_n \rangle : (\vdash \Lambda)$. On the other hand we have $(+, \xi_n, I_n) \cdot \Psi_n : (\vdash \Lambda_0)$ and $\Psi'_n : \{\zeta_1 \vdash \Lambda_1, \ldots, \zeta_m \vdash \Lambda_m\}$ with $\xi_n \in \Lambda_0$ and $\xi_n \notin \{\zeta_1, \ldots, \zeta_m\}$, that is, ξ_n is not cut. Hence $\xi_n \in \Lambda$. In particular, when Λ is empty, then the weak reduction machine can only end with an Ω or a ✠, and there is no need for strong reduction – this situation is quite similar to the situation in a functional programming language like CAML, whose implementation is based on weak reduction, which is complete for evaluating observable results of basic types. In the sequel, we shall often place ourselves in such a situation, with $\phi : (\vdash \xi)$, $\Psi = \{\psi\}$, and $\psi : (\xi \vdash)$, for some ξ.*

Definition 4.5 *For $\phi : (\vdash \xi)$ and $\psi : (\xi \vdash)$, we write $\phi \perp \psi$ (or $\psi \perp \phi$) if $[\![\phi, \psi]\!] =$ ✠ and we say that ϕ is* orthogonal *to ψ. For a set A of designs on the same base $\vdash \xi$ (resp. $\xi \vdash$), we write $A^\perp = \{\psi \mid \forall \phi \in A \;\; \phi \perp \psi\}$ (resp. $A^\perp = \{\phi \mid \forall \psi \in A \;\; \phi \perp \psi\}$), and we write simply $\phi^\perp$ for $\{\phi\}^\perp$.*

The definition of orthogonality is not limited to bases of the form $\vdash \xi$ and $\xi \vdash$: more generally, if $\phi : (\vdash \xi_1, \ldots, \xi_n)$, $\psi_1 : (\xi_1 \vdash), \ldots, \psi_n : (\xi_n \vdash)$, and $[\![\phi, \{\psi_1, \ldots, \psi_n\}]\!] =$ ✠, then we say that ϕ is orthogonal to $\{\psi_1, \ldots, \psi_n\}$.

Exercise 4.6 *Show that ✠ is the only design which is orthogonal to the skunk (cf. Definition 3.4).*

Exercise 4.7 *Show that if $I \subseteq J$, $\phi : (\vdash I)$ and $\psi_j : (j \vdash)$ for all $j \in J$, then $[\![\phi, \{\psi_i \mid i \in I\}]\!] = [\![\phi, \{\psi_j \mid j \in J\}]\!]$ (taking ϕ to be of type $\vdash J$ on the right side of this equality).*

Exercise 4.8 *Rule (R) does not make explicit the alternation between (the subdesigns of) ϕ and ψ: the positive actions that guide computation are indeed alternating. The following variant of rule (R) (to be applied systematically after the second step) makes this alternation explicit:*

$$\text{(R')}\quad \frac{\langle (+, \xi, I) \cdot \Psi' \mid \Psi \rangle \longrightarrow \langle (+, \xi_1, I_1) \cdot \Psi'_1 \mid \Psi_1 \rangle \qquad (\Psi_1 \text{ accepts } \xi_1)}{\langle (+, \xi_1, I_1) \cdot \Psi'_1 \mid \Psi_1 \rangle \longrightarrow \langle (\Psi_1)_{\xi_1, I_1} \mid \Psi'_1 \cup (\Psi \setminus \xi) \rangle}$$

(1)Using this rule, check that the execution of the above example unrolls now as follows:

$$\begin{array}{rcl} \langle \phi \mid \psi \rangle & \longrightarrow & \langle \phi_1 \mid \Psi' \rangle \\ & \longrightarrow & \langle \phi_2 \mid \Psi_1 \rangle \\ & \longrightarrow & \langle \phi_3 \mid (\Psi' \setminus \xi i_1) \cup \Psi_2 \rangle \end{array}$$

(2) Show formally that the machine obtained with (R') is equivalent to the one defined with (R).

Exercise 4.9 *What about normalizing a negative design against a negative design? (Hint: Use strong reduction.)*

Normalization in concrete syntax. How does normalization look like, when described in terms of the concrete rather than the abstract syntax? We let the reader convince himself that the affine (weak reduction) machine given above gets mapped through the bijective correspondence between the concrete syntax and the abstract syntax to the following one:

$$\langle (x\cdot I)\{M_i \mid i\in I\} \mid \rho \cup (x \leftarrow \{\ldots, I = \lambda\{x_i \mid i \in I\}.P_I, \ldots\}\rangle \longrightarrow \langle P_I \mid \rho \cup (\bigcup_{i\in I}(x_i \leftarrow M_i)\rangle$$

In this rule, ρ stands for an environment which is a function from a finite set of variables to terms, described as a set of bindings of the form $(x \leftarrow M)$, and the union symbols stand for disjoint unions. The initial state corresponding to $\langle \phi \,|\, \psi\rangle$ is here of the form $\langle P \,|\, \{(x \leftarrow M)\}\rangle$, for some $P : (\vdash x : \xi, \Lambda_1)$ and $M : (\xi \vdash \Lambda_2)$.

The crucial use of the affine restriction lies in the fact that x does not occur in any of the M_i's, and hence that the binding for x is consumed after the application of the rule. But the general case of non necessarily affine pseudo-designs in concrete syntax is not much more difficult to describe, by means of a very similar abstract machine, which is an instance of the (stack-free) Krivine machine described in [16]:

$$\langle (x\cdot I)\{M_i \mid i\in I\} \mid \rho\rangle \longrightarrow \langle P_I \mid \rho_I \cup (\bigcup_{i\in I}(x_i \leftarrow \langle M_i \mid \rho\rangle)\rangle$$

where $\rho(x) = \{\ldots, I = \langle \lambda\{x_i \mid i\in I\}.P_I \mid \rho_I\rangle, \ldots\}$.

The main change with respect to the affine machine is that now an environment maps variables to closures, which are used in the implementation of functional programming languages to keep track of a code together with the environment for its free variables. Note that we keep now all the environment to be available for the M_i's, even the binding for x, as x may appear in any of the M_i's.

Exercise 4.10 *Show that, for any positive P of appropriate type, the (strong version of) the machine reduces $\langle P \mid \{x' \leftarrow Fax_{\xi,x:\xi'}\}\rangle$ to $P[x' \leftarrow x]$ (α-renaming).*

Token machine: normalization as a visit. We now give our third version of normalization, which is more intuitive. It is described in terms of pushing a token through the designs (see also [24]). Under this interpretation, the net formed of ϕ and ψ remains fixed, and information (in the form of a mark or token placed on a single node) flows through it. Initially, the token is placed at the root of ϕ. In order to identify the nodes of ϕ and ψ, we introduce the following notation for occurrrences of actions:

$$(\Omega)_\epsilon = \Omega \qquad\qquad (\maltese)_\epsilon = \maltese$$

$$((+,\xi,I)\cdot\{\psi_{\xi i} \mid i\in I\})_\epsilon = (+,\xi,I) \qquad\qquad ((+,\xi,I)\cdot\{\psi_{\xi i} \mid i\in I\})_{iu} = (\psi_{\xi i})_u$$

$$(\{(-,\zeta,J)\phi_J \mid J\in\mathcal{P}_f(\omega)\})_J = (-,\zeta,J) \qquad\qquad (\{(-,\zeta,J)\phi_J \mid J\in\mathcal{P}_f(\omega)\})_{J1u} = (\phi_J)_u$$

Thus an occurrence is a word over an alphabet whose letters are either i, I, or 1, and which appear in the order $i_1I_11i_2I_21i_3\ldots$. The token is formalized as an occurrence u of either ϕ or ψ, which we write (L,u) (resp. (R,u)) if the occurrence is in ϕ (resp. ψ). The token machine maintains a set of pairs $((L,u),(R,v))$ where the actions $(\phi)_u$ and $(\psi)_v$ are opposite, and where each u,v occurs at most once. These pairs are called *bindings* (in a sequent calculus description of cut-elimination (cf. part I, section 3), they represent the successive cuts). Initially, the token is at occurrence (L,ϵ), and the set of bindings is empty. The token game follows the tree structure and the pointer structure of ϕ and ψ, and the bindings. The rules are as follows:

- from (L,ϵ), with $(\phi)_\epsilon = (+,\xi,I)$, move to (R,I) and place $(((L,\epsilon),(R,I))$ in the list of bindings;
- from (R,u) such that $(\psi)_u$ is a negative action, move to $(R,u1)$;
- from (R,u) such that $(\psi)_u = (+,\zeta i,I)$, u points to v, and $((L,v'),(R,v))$ is in the set of bindings, then move to $(L,v'\,i\,I)$ and add $((L,v'\,i\,I),(R,u))$ to the set of bindings;
- from (L,u) such that $(\phi)_u = (+,\zeta i,I)$, u points to v, and $((L,v),(R,v'))$ is in the set of bindings, then move to $(R,v'\,i\,I)$ and add $((L,u),(R,v'\,i\,I))$ to the set of bindings;
- from (L,u) such that $(\phi)_u$ is a negative action, move to $(L,u1)$.

For example, the execution of our running example goes as follows, with the corresponding actions (recall that $\zeta = \xi i_1 i$):

$$\begin{array}{ccccccccc}
(L,\epsilon) & (R,I) & (R,I1) & (L,i_1I_1) & (L,i_1I_11) & (R,I1iJ) & (R,I1iJ1) & (L,i_2\,I_2) & \ldots \\
 & & & & & & & & \\
(\phi)_\epsilon & (\psi)_I & \psi_{I1} & (\phi)_{i_1I_1} & (\phi)_{i_1I_11} & (\psi)_{I1iJ} & (\psi)_{I1iJ1} & (\phi)_{i_2I_2} & \ldots \\
= & = & = & = & = & = & = & = & \ldots \\
(+,\xi,I) & (-,\xi,I) & (+,\xi i_1,I_1) & (-,\xi i_1,I_1) & (+,\zeta,J) & (-,\zeta,J) & (+,\xi i_2,I_2) & (-,\xi i_2,I_2) & \ldots
\end{array}$$

The actions *visited* during normalization are all the actions $(\phi)_u$ such that the token reaches position (L,u) and all the actions $(\psi)_u$ such that the token reaches position (R,u). They determine two designs $\phi_1 \subseteq \phi$ and $\psi_1 \subseteq \psi$ (the *pull-back*, see Theorem 4.13) which form a *balanced* pair (i.e., their sets of actions are dual). Remarkably, there is a converse to this (see Proposition 4.19).

Exercise 4.11 *Show that the first and third versions are equivalent, i.e., that they yield the same result of normalization for every pair of designs $\phi : (\vdash \xi)$ and $\psi : (\xi \vdash)$. (Hint: At some point, Proposition 4.18 has to be used.)*

Analytical theorems. These are the following theorems:

1. the *associativity theorem*, that corresponds to Church-Rosser property;
2. the *separation theorem*, that corresponds to (an affine version of) Böhm's theorem;
3. the *monotonicity theorem*, that corresponds to the syntactic continuity theorem in the λ-calculus, due to Welch and Lévy, which states that Böhm trees commute with contexts (see e.g. [4, section 2.3]);

4. the *stability theorem*, that corresponds to the syntactic stability theorem of the λ-calculus, due to Berry, which states that the Böhm tree function from partial terms to partial Böhm trees is stable (it enjoys actually the stronger property of being sequential, see e.g. [4, section 2.3]).

Remark 4.12 *The tradition behind the third and the fourth theorems is more widely known through their outsprings in denotational semantics [4]: Scott 's continuous semantics, Berry 's stable semantics. Berry's semantics was clearly motivated by the stability theorem, while the syntactic continuity theorem seems rather to have given an additional a posteriori confirmation of the pertinence of Scott's theory, which was initially suggested by results in recursion theory (theorems of Rice and of Myhill-Shepherdson).*

We start with the stability theorem.

Theorem 4.13 (Stability) *If r is a chronicle of $[\![\phi, \psi]\!]$, then there exist $\phi_0 \subseteq \phi, \psi_0 \subseteq \psi$ minimum, called the* pull-back *of r along normalization, such that $r \in [\![\phi_0, \psi_0]\!]$.*

Proof. We mark all the nodes visited during the (deterministic) normalization in order to obtain r: all these nodes must be present, otherwise normalization would diverge, moreover they suffice (just adding Ω's above non visited negative actions following visited positive actions, in order to obtain pseudo-designs). One also has to check that these pseudo-designs are actually designs (omitted). $\square$

It is well-known that stability described in terms of pull-backs as above entails stability in algebraic terms: if $\phi_1, \phi_2 \subseteq \phi$ and $\psi_1, \psi_2 \subseteq \psi$, then

$$[\![\phi_1 \cap \phi_2, \psi_1 \cap \psi_2]\!] = [\![\phi_1, \psi_1]\!] \cap [\![\phi_2, \psi_2]\!]$$

(set intersection of the designs as sets of chronicles). In particular, for $\phi_1, \phi_2 \subseteq \phi$ based on, say, $\vdash \xi$, and ψ based on $\xi \vdash$, we have:

$$(\phi_1 \cap \phi_2) \perp \psi \quad \Leftrightarrow \quad \phi_1 \perp \psi \text{ and } \phi_2 \perp \psi \,.$$

The same properties are true for any bounded intersection of ϕ_k's, ψ_k's.

Theorem 4.14 (Separation) *The following equivalence holds, for all designs ϕ_1, ϕ_2 on the same base: $\phi_1 \sqsubseteq \phi_2$ if and only if $\phi_1^\perp \subseteq \phi_2^\perp$.*

Proof. Let $\phi_1 \sqsubseteq \phi_2$. By transitivity, we may restrict our attention to $\leq^R$ and to $\leq^L$. Let ψ be such that $\phi_1 \perp \psi$. Let ϕ be the part of ϕ_1 that is effectively visited during normalization (cf. the stability theorem). If $\phi_1 \leq^R \phi_2$, ϕ is also included in ϕ_2, and we thus also have $\phi_2 \perp \psi$. If $\phi_1 \leq^L \phi_2$, and if ϕ is not included in ϕ_2, then some subtree of ϕ has been replaced by ✠, but then the normalization will meet this ✠, i.e., normalization is more rapid with ϕ_2 than with ϕ_1.

Reciprocally, suppose that $\phi_1^\perp \subseteq \phi_2^\perp$, and that there exists a pair of chronicles, one in ϕ_1, the other in ϕ_2, that are not prefix of each other. Their intersection q ends with a negative action, since after a positive action, the set of negative actions depends only on the previous positive action, not on the design. By Exercise 3.15, it suffices to check that the only possible configurations are the following: either the branch that continues in ϕ_1 is reduced to Ω, or the branch that continues in ϕ_2 is reduced to ✠. We use the following construction, that allows us to explore a chronicle interactively. We first define the (dual) *view* of a chronicle (supposing that the design is based on $\vdash \epsilon$, for simplicity):

$$\begin{aligned}&view(r(-,\zeta,J)) = view(r)(+,\zeta,J)\\&view((+,\epsilon,I)) = (-,\epsilon,I)\\&view(r(-,\xi,J)\cdots(+\xi j,I)) = view(r)(+,\xi,J)(-,\xi j,I))\end{aligned}$$

We next define the following designs (as sets of chronicles):

- Opp_r (for a positive chronicle r) consists of the set of the views of the prefixes of r, plus $view(r)$ ✠; we have that Opp_r is the minimum design such that $r \perp Opp_r$, by construction;
- Opp_q (for a negative chronicle) consists of the set of the views of the prefixes of q, plus all the chronicles of the form $view(q)(-,\zeta,J)\Omega$; we have that Opp_q is the minimum design such that $(q$ ✠$) \perp Opp_q$, by construction.

Back to the proof. Let q be the maximum common prefix. We distinguish three cases:

1. q ✠ $\in \phi_1$. Then we have $(q$ ✠$)\perp Opp_q$, hence $\phi_1 \perp Opp_q$. We should then have $\phi_2 \perp Opp_q$, but the interaction between ϕ_2 and Opp_q follows q, and then has not "enough fuel" to continue in ϕ_2. This case is thus impossible.
2. $r = q(+,\xi,I) \in \phi_1$. We reason similarly. We have $r \perp Opp_r$, and the interaction has not "enough fuel" to continue in ϕ_2, except if q ✠ $\in \phi_2$ (second allowed configuration).
3. $q\,\Omega \in \phi_1$. This is the first allowed configuration. □

Remark 4.15 *We have made a crucial use of the affinity condition in the proof of the separation theorem. Consider a chronicle of the form $q(+,\xi)(-,\xi i_1)\ldots(+,\xi)(-,\xi i_2)\ldots$, forgetting ramifications for simplicity. Then we collect two views*

$$view(q)\,(-,\xi)\,(+,\xi i_1) \text{ and } view(q)\,(-,\xi)\,(+,\xi i_2)$$

that do not fit together in a design, since a strategy answers an Opponent's move uniquely.

Theorem 4.16 (Associativity) *Let ϕ, Ψ_1, and Ψ_2 be such that ϕ and $\Psi_1 \cup \Psi_2$ form a net. Then we have:*

$$[\![[\![\phi,\Psi_1]\!],\Psi_2]\!] = [\![\phi,\Psi_1\cup\Psi_2]\!]\ .$$

Proof (indication). Let r be a chonicle of $[\![\phi,\Psi_1\cup\Psi_2]\!]$, with pullback ϕ',Ψ_1',Ψ_2'. One shows that r can be pulled back along normalization of $\langle[\![\phi,\Psi_1]\!] \mid \Psi_2\rangle$, with a pullback ϕ',Ψ_2'' such that $\Psi_2' = \Psi_2''$ and the pullback of ϕ' along normalization of $[\![\phi,\Psi_1]\!]$ is ϕ',Ψ_1'. A similar statement can be proved in the other direction. These statements are best proved using a strong reduction version of the token machine. Intuitively, the machine execution corresponding to the left hand side does some extra work with respect to the execution on the right hand side net: as the execution proceeds, it records the part of the normal form of the net $\langle\phi \mid \Psi_1\rangle$ that is being built. □

Theorem 4.17 (Monotonicity) *If $\phi_1 \sqsubseteq \phi_2$ and $\psi_{1,1}\sqsubseteq\psi_{2,1},\ldots,\psi_{1,n}\sqsubseteq\psi_{2,n}$, then*

$$[\![\phi_1,\{\psi_{1,1},\ldots,\psi_{1,n}\}]\!] \sqsubseteq [\![\phi_2,\{\psi_{2,1},\ldots,\psi_{2,n}\}]\!]\ .$$

Proof. We have to show that $[\![\phi_1, \{\psi_{1,1}, \ldots, \psi_{1,n}\}]\!] \perp \Psi$ implies $[\![\phi_2, \{\psi_{2,1}, \ldots, \psi_{2,n}\}]\!] \perp \Psi$, for all Ψ. By associativity, this amounts to deduce $[\![\phi_2, \{\psi_{1,1}, \ldots, \psi_{1,n}\} \cup \Psi]\!] = \maltese$ from $[\![\phi_1, \{\psi_{2,1}, \ldots, \psi_{2,n}\} \cup \Psi]\!] = \maltese$, which is a consequence of the assumptions (recall that any increase in the order $\sqsubseteq$ only fastens normalization). $\square$

We end the section by showing that normalization explores only multiplicative parts.

Proposition 4.18 *The pull-back of a chronicle is always a slice (cf. Definition 3.9). More generally, the pullback of a slice is a slice. Moreover, each node of this slice is visited only once.*

Proof. We limit ourselves to a normalization between $\phi : (\vdash \xi)$ and $\psi : (\xi \vdash)$ (and hence to the pullback of $\maltese$). For two negative actions $(-, \xi i, I_1)$ and $(-, \xi i, I_2)$ with the same focus and above the same positive action $(+, \xi, I)$ to be visited, we need on the other side two positive actions $(+, \xi i, I_1)$ and $(+, \xi i, I_2)$ pointing to the same negative action $(-, \xi, I)$. We reason by minimal counter-example, where minimality is taken with respect to the time at which the hypothetical second visit takes place. Typing excludes that these two positive actions be one above another (the address is consumed after the first visit). Thus they must appear in incompatible positions in the tree, and the divergence starts from a positive action. Then typing implies that the first two negative actions on the two diverging paths must have the same focus (disjointness of contexts). These two actions must have been visited before, a contradiction to minimality. Note that the reasoning applies a fortiori with $I_1 = I_2$, which proves the last part of the statement. $\square$

Notice that Proposition 4.18 implies that the token machine terminates on finite designs (a property which is obvious with our first formalization of normalization).

Proposition 4.19 *If two finite slices ϕ and ψ are such that their underlying set of actions are opposite, then the normalization of $\langle \phi \mid \psi \rangle$ visits all of ϕ and ψ.*

Proof. (A more general statement is proved in [35, Proposition 1]). Note that since we deal with slices, we can omit the ramifications, and name the actions simply with their sign and their focus. We say that two slices satisfying the condition in the statement form a *balanced pair*: $(+, \xi)$ (resp. $(-, \xi)$) occurs in ϕ if and only if $(-, \xi)$ (resp. $(+, \xi)$) occurs in ψ.

The proof is by contradiction. If the normalization stops before having explored ϕ and ψ entirely, it stops having visited $\phi' \subseteq \phi$ and $\psi' \subseteq \psi$, where ϕ' and ψ' form a balanced pair, and we have, say, $\phi' \neq \phi$, but then also $\psi' \neq \psi$ since ϕ' and ψ' have the same addresses. Consider the actions on the border, i.e., which are not in ϕ' nor in ψ', but whose father is in ϕ' or ψ': they must be negative, since otherwise the normalization procedure would have visited them. So let us pick an action $(-, \xi_1)$ on the border, say in $\phi \setminus \phi'$. Then also $(+, \xi_1)$ occurs somewhere in $\psi \setminus \psi'$. Consider the chronicle (determined by) $(+, \xi_1)$, and let $(-, \xi_2)$ be the action of this chronicle which lies on the border. We can continue this and build an infinite sequence of actions $(-, \xi_n)$. Now, for all n, we have $\xi_n = \xi'_n i_n$ and $(+, \xi'_n) \in \phi' \cup \psi'$, for some ξ'_n and i_n (focalization condition, cf. Remark 3.8). We also have that $(-, \xi'_n) \in \phi' \cup \psi'$, since ϕ' and ψ' form a balanced pair. The action $(-, \xi'_n)$ appears on the chronicle $(+, \xi_n)$ (subaddress condition), and hence by construction $(-, \xi'_n)$ occurs before $(+, \xi'_{n+1})$ on this chronicle. It follows that $(-, \xi'_n)$ is visited before $(+, \xi'_{n+1})$ during normalization. Since $(+, \xi'_n)$ is visited right before $(-, \xi'_n)$, we have that $(+, \xi'_{n+1})$ is visited strictly after $(+, \xi'_n)$. But this contradicts the termination of the normalization procedure, which as we have seen is a consequence of Proposition 4.18. $\square$

Several characterizations of those designs ϕ which can be entirely visited during normalization against a counter-design ψ are given in [26].

Exercise 4.20 *Show that in presence of the separation property, associativity is equivalent to (a general formulation of) the following statement (called* closure principle*). If $\phi : (\vdash \xi)$ and $\psi : (\xi \vdash \lambda)$, then $[\![\phi, \psi]\!]$ is the unique design ϕ' such that $[\![\phi', \psi']\!] = [\![\phi, \{\psi, \psi'\}]\!]$ for all $\psi' : (\lambda \vdash)$.*

5 Behaviours

A positive *behaviour* on a base $\vdash \Lambda$ (resp. a negative behaviour on a base $\xi \vdash \Lambda$) is a set $\mathbf{G}$ of designs of type $(\vdash \Lambda)$ (resp. $(\xi \vdash \Lambda)$) closed by taking the bi-orthogonal, i.e., $\mathbf{G} = \mathbf{G}^{\perp\perp}$. In most cases, we suppose that the base is of the form $\vdash \xi$ or $\xi \vdash$. Equivalently, a behaviour, say, on the base $\vdash \xi$ (resp. $(\xi \vdash)$) is a set of the form $A^\perp$, where A is an arbitrary set of designs of type $(\xi \vdash)$ (resp. $(\vdash \xi)$). Throughout the section, we take $\xi = \epsilon$. Here are some examples of behaviours:

- If $A = \emptyset$, then $A^\perp$ is the set of all the designs of type $(\epsilon \vdash)$ (resp. $(\vdash \epsilon)$): this behaviour is denoted $\top$.
- If $A = \top$ is negative (resp. positive), it is easy to see that $A^\perp = \{Dai\}$ (resp. $A^\perp = \{Dai^-\}$). Hence a behaviour is never empty, it always contains *Dai* or Dai^-.
- If ϕ is a design, the smallest behaviour that contains it is:

$$\{\phi\}^{\perp\perp} = \{\phi' \mid \phi \sqsubseteq \phi'\} \ .$$

  Indeed, for any A, we have $A^\perp = \{\phi' \mid A \subseteq \phi'^\perp\}$, so in particular $\{\phi\}^{\perp\perp} = \{\phi' \mid \phi^\perp \subseteq \phi'^\perp\} = \{\phi' \mid \phi \sqsubseteq \phi'\}$.

Below, we list a few closure properties of behaviours, which are easily checked:

- If $\mathbf{G}$ is a behaviour and if $\phi \in \mathbf{G}$ and $\phi \sqsubseteq \phi'$, then $\phi' \in \mathbf{G}$ (by the separation theorem).
- If $\mathbf{G}$ is a behaviour and if ϕ_k is a bounded family of designs (considered as sets of chronicles) of $\mathbf{G}$, i.e., $\exists \phi \ (\forall k \ \ \phi_k \subseteq \phi)$), then their intersection is a design of $\mathbf{G}$ (by the stability theorem).
- Every intersection of behaviours is a behaviour (as, say, $A^\perp \cap B^\perp = (A \cup B)^\perp$).

Definition 5.1 (incarnation) *If $\mathbf{G}$ is a behaviour and if $\phi \in \mathbf{G}$, we call the incarnation of ϕ in $\mathbf{G}$ the smallest design $\subseteq \phi$ of $\mathbf{G}$ (whose existence follows from the second closure property listed above). We denote it as $|\phi|_{\mathbf{G}}$ or simply $|\phi|$. An incarnated design is a design ϕ such that $\phi = |\phi|$. We set $|\mathbf{G}| = \{\phi \in \mathbf{G} \mid \phi = |\phi|_{\mathbf{G}}\}$.*

The stability theorem and the definition of behaviour allow us to give a more operational characterization of the incarnation: $|\phi|_{\mathbf{G}}$ is the set of chronicles that are visited during a normalization of ϕ against some design ψ of $\mathbf{G}^\perp$ (different designs may be used for witnessing the visit of different chronicles).

Incarnation is clearly contravariant: if $\mathbf{G} \subseteq \mathbf{H}$ and $\phi \in \mathbf{G}$, then $|\phi|_{\mathbf{H}} \subseteq |\phi|_{\mathbf{G}}$. The extreme case is $|\psi|_\top = Skunk$ (negative base).

Lemma 5.2 *For any behaviour, we have* $\mathbf{G}^\perp = |\mathbf{G}|^\perp$.

Proof. We get $\mathbf{G}^\perp \subseteq |\mathbf{G}|^\perp$ by contravariance. Now let $\psi \in |\mathbf{G}|^\perp$ and let $\phi \in G$. Then $\psi \perp |\phi|_{\mathbf{G}}$, and hence $\psi \perp \phi_G$ by monotonicity. □

Additives. We now have enough material to return to types and logic. The paradigm is that of behaviours as types. The rest of the section is devoted to constructions on behaviours, corresponding to those of linear logic, and more. Some of these constructions are reminiscent of phase semantics (cf. part I, section 5), but the present framework is obviously much richer.

Definition 5.3 (Intersection, Union) *If* $\mathbf{G}_k$ *is a family of behaviours on the same base, the set intersection of this familly is a behaviour (cf. above), that we call the intersection of these behaviours, notation* $\bigcap_k \mathbf{G}_k$. *We define the union of behaviours* $\bigsqcup_k \mathbf{G}_k$ *as the bi-orthogonal of their set union:* $\bigsqcup_k \mathbf{G}_k = (\bigcup_k \mathbf{G}_k)^{\perp\perp}$.

The ordinary additives correspond to the particular case where the connectives are applied to disjoint behaviours – a notion that we define now (using some of the designs of Definition 3.4).

Definition 5.4 (Directory) *A directory is a set of ramifications (i.e., a subset of* $\mathcal{P}_f(\omega)$*). If* $\mathbf{G}$ *is a positive behaviour on* $\vdash \epsilon$*, we define the directory* $Dir(\mathbf{G})$ *as follows:* $Dir(\mathbf{G}) = \{I \mid Ram_{(\epsilon,I)} \in \mathbf{G}\}$. *If* $\mathbf{G}$ *is negative, we define* $Dir(\mathbf{G})$ *by the equation* $Dir_{Dir(\mathbf{G})} = |Dai^-|_{\mathbf{G}}$.

The following properties show the relevance of this definition:

- If $\mathbf{G}$ is positive, then $Dir(\mathbf{G})$ is the set of the I's such that $\mathbf{G}$ contains a design beginning with $(+,\epsilon,I)$. Let us check this. If $\phi \in \mathbf{G}$ begins with $(+,\epsilon,I)$, then $\phi \sqsubseteq Ram_{(\epsilon,I)}$, hence $Ram_{(\epsilon,I)} \in \mathbf{G}$, i.e., $I \in Dir(\mathbf{G})$. The reciprocal is immediate.

- We have $Dir(\mathbf{G}^\perp) = Dir(\mathbf{G})$. The proof goes as follows. Since $\mathbf{G}^{\perp\perp} = \mathbf{G}$, we may suppose that $\mathbf{G}$ is positive. We have $I \in Dir(\mathbf{G}^\perp)$ if and only if there exists $\phi \in \mathbf{G}$ such that the normalization of $\langle \phi \mid Dai^- \rangle$ explores the branch $(-,\epsilon,I)$ of Dai^-, i.e., such that ϕ begins with $(+,\epsilon,I)$. But by the previous property, this amounts to $I \in Dir(\mathbf{G})$.

- If $\mathbf{G}$ is a negative behaviour and if ψ is incarnated in $\mathbf{G}$ on base $\xi \vdash$, then $Dir(\mathbf{G}) = \{I \mid (-,\xi,I)$ is an initial action of $\psi\}$. Indeed, each ϕ of $\mathbf{G}^\perp$ induces a visit on ψ, which starts with a $(-,\xi,I)$ matching ϕ's initial action: collecting all these visits together (which is the definition of incarnation), we get the statement.

Definition 5.5 (Disjoint behaviours) *We say that two behaviours* $\mathbf{G}$ *and* $\mathbf{G}'$ *on the same base are disjoint if* $Dir(\mathbf{G})$ *and* $Dir(\mathbf{G}')$ *are disjoint sets.*

If two negative behaviours $\mathbf{G}$ and $\mathbf{H}$ are disjoint (that is, $Dir(\mathbf{G}) \cap Dir(\mathbf{H}) = \emptyset$), and if ψ_1 and ψ_2 are respective incarnated designs of $\mathbf{G}$ and $\mathbf{H}$, their union is well-defined. Moreover, it is obviously a design of $\mathbf{G} \cap \mathbf{H}$. This actually defines a bijection between $|\mathbf{G}| \times |\mathbf{H}|$ and $|\mathbf{G} \cap \mathbf{H}|$. Hence in the disjoint case, intersection is product! Girard calls this striking property the “mystery of incarnation”.

Lemma 5.6 *If* $\mathbf{G}$ *and* $\mathbf{H}$ *are disjoint negative behaviours, then* $(\mathbf{G} \cap \mathbf{H})^\perp = \mathbf{G}^\perp \cup \mathbf{H}^\perp$.

Proof. By contravariance, we have $\mathbf{G}^\perp \subseteq (\mathbf{G}\cap\mathbf{H})^\perp$ and $\mathbf{H}^\perp \subseteq (\mathbf{G}\cap\mathbf{H})^\perp$. It remains to show $(\mathbf{G}\cap\mathbf{H})^\perp \subseteq \mathbf{G}^\perp \cup \mathbf{H}^\perp$. Let $\phi \in (\mathbf{G}\cap\mathbf{H})^\perp$. If $\phi = \maltese$, then ϕ belongs to any behaviour, and hence a fortiori to $\mathbf{G}^\perp \cup \mathbf{H}^\perp$. So we can suppose that ϕ starts with a positive action. For any pair of incarnated designs $\psi_1 \in \mathbf{G}$ and $\psi_2 \in \mathbf{H}$, we have (cf. above) $\psi_1 \cup \psi_2 \in \mathbf{G}\cap\mathbf{H}$, and hence $\phi \perp (\psi_1 \cup \psi_2)$. Then ϕ's initial action $(+, \xi, I)$ matches an initial action of (exclusively) either ψ_1 or ψ_2, say, not of ψ_2: then all normalizatiojn takes place in ψ_1, so we have in fact $\phi \perp \psi_1$. Now we can let ψ_1 vary over $|\mathbf{G}|$ while keeping ψ_2 fixed. Then we have $\phi \perp \psi$ for all $\psi \in |\mathbf{G}|$, that is, $\phi \in |\mathbf{G}|^\perp$, and we conclude by Lemma 5.2. □

Definition 5.7 (Additive connectives) *If* $\mathbf{G}$ *and* $\mathbf{H}$ *are two disjoint negative (resp. positive) behaviours, we rebaptize their intersection (resp. union) in the sense of definition 5.3 as follows:* $\mathbf{G}\cap\mathbf{H} = \mathbf{G}\&\mathbf{H}$ *(resp.* $\mathbf{G}\bigsqcup\mathbf{H} = \mathbf{G}\oplus\mathbf{H}$*).*

Proposition 5.8 *If* $\mathbf{G}$ *and* $\mathbf{H}$ *are negative and disjoint, then* $|\mathbf{G}\&\mathbf{H}| \approx |\mathbf{G}| \times |\mathbf{H}|$.

Proof. We have constructed above a mapping from $|\mathbf{G}| \times |\mathbf{H}|$ to $\mathbf{G}\cap\mathbf{H}$, which takes ψ_1 and ψ_2 and returns $\psi_1 \cup \psi_2$. This design is incarnated, since the whole of ψ_1 is visited by normalisation against the designs of $\mathbf{G}^\perp$ (which are a fortiori designs of $(\mathbf{G}\cap\mathbf{H})^\perp$), and similarly for ψ_2. Hence we have defined a map from $|\mathbf{G}| \times |\mathbf{H}|$ to $|\mathbf{G}\cap\mathbf{H}|$. This map is injective by the disjointness assumption. We are left to show that it is surjective. Let ψ be an incarnated design of $\mathbf{G}\cap\mathbf{H}$, Then, by Lemma 5.6, we can write $\psi = \psi_1 \cup \psi_2$, where, say, $\psi_1 = \{\psi_\phi \mid \phi \in \mathbf{G}^\perp\}$ (where ψ_ϕ is the part of ψ visited during the normalization of $\langle \phi \mid \psi \rangle$), which is incarnated in $\mathbf{G}$. □

Remark 5.9 *Note that the cartesian product is associative only up to isomorphism, while intersection is isomorphic up to equality. Girard points out that a way to get a better match is to redefine the product of two disjoint sets* X *and* Y *as follows:* $X \times Y = \{x \cup y \mid x \in X \text{ and } y \in Y\}$. *Note that this definition makes also sense when the condition on* X *and* Y *is not satisfied, but one then does not obtain a product (in the category of sets) of* X *and* Y *anymore.*

The $\oplus$ connective has also a remarkable property: one can get rid of the bi-orthogonal. Moreover, it is also a union at the level of incarnations (see Exercise 5.13).

Proposition 5.10 *Let* $\mathbf{G}$ *and* $\mathbf{H}$ *be positive and disjoint behaviours. Then* $\mathbf{G}\oplus\mathbf{H}$ *is simply the set union of* $\mathbf{G}$ *and* $\mathbf{H}$.

Proof. This is an immediate consequence of Lemma 5.6:

$$\mathbf{G}\cup\mathbf{H} = \mathbf{G}^{\perp\perp} \cup \mathbf{H}^{\perp\perp} = (\mathbf{G}^\perp \cap \mathbf{H}^\perp)^\perp = (\mathbf{G}\cup\mathbf{H})^{\perp\perp}\ .$$

□

Remark 5.11 *Girard calls* internal completeness *the situation when a new behaviour can be defined from other behaviours without using a bi-orthogonal. Indeed, given a set* A *of designs, the set* $A^\perp$ *can be viewed as the set of the (counter-)models of* A*, and then* A *being closed under bi-orthogonal means that everything valid was already there.*

What if two behaviours are not disjoint? Then we can still form a & or a $\oplus$, provided we force disjunction by copying the behaviours, or *delocating* them, as Girard says (see Exercises 5.14 and 5.15).

But intersections are not less useful than products. They have been introduced long ago in the study of models of untyped λ-calculi [15, 44], and have been used to give semantic foundations to object-oriented programming (see, e.g. [39]). We show here a simple example of how records can be encoded as designs, and how they can be observed by suitable behaviours. Let us consider three fields: `radius`, `angle`, `colour`. Assuming that we have fixed an origin and two orthogonal vectors of equal norm, then giving a positive number r as radius and a positive number $\phi \pmod{360}$ gives us a point in the plane, while giving an r and a coulour, say, blue, gives us a blue circle. If the three fields are given a value, then we get a couloured point. To encode these simple data, we shall use negative designs (recall that & is a negative connective), on base $\epsilon \vdash$. Let I_1, I_2, I_3 be arbitrary ramifications (i.e., finite subsets of ω), which are pairwise distinct, for example, $I_1 = \{i_1\}, I_2 = \{i_2\}, I_3 = \{i_3\}$ with i_1, i_2, i_3 distinct. We shall use them to encode the three fields `radius`, `angle`, and `colour`, respectively. Here is a red point on the horizontal axis, negative side, at distance 2 from the origin (yes, we encode here only denumerably many points..., and, say, red is encoded by 9):

$$\begin{array}{rcl} \psi & = & \{(-,\epsilon,\{i_1\})\,(+,i_1,\{2\})\cdot\{\mathit{Skunk}_{i_1 2}\}, \\ & & (-,\epsilon,\{i_2\})\,(+,i_2,\{180\})\cdot\{\mathit{Skunk}_{i_2\star(180)}\}, \\ & & (-,\epsilon,\{i_3\})\,(+,i_3,\{9\})\cdot\{\mathit{Skunk}_{i_3 9}\}\} \end{array}$$

Suppose that we are only interested in the underlying red circle (centered at the origin) – yes, with this representation, the type of points is a subtype of the type of circles centered at the origin: we simply forget the `angle` component. Formally, we define the following behaviour $\mathbf{G}$ of coloured circles:

$$\mathbf{G} \quad = \quad \{(+,\epsilon,\{i_1\})\cdot\{\mathit{Dai}^-_{i_1}\}\ ,\ (+,\epsilon,\{i_3\})\cdot\{\mathit{Dai}^-_{i_3}\}\}^\perp\ .$$

We have $|\psi|_{\mathbf{G}} = \{(-,\epsilon,\{i_1\})\,(+,i_1,\{2\})\cdot\{\mathit{Skunk}_{i_1 2}\}\ ,\ (-,\epsilon,\{i_3\})\,(+,i_3,\{9\})\cdot\{\mathit{Skunk}_{i_3 9}\}\}$. One could similarly define the behaviour $\mathbf{G}'$ of (uncoloured) points. Then $\mathbf{G}\cap\mathbf{G}'$ is the behaviour of coloured points, and this behaviour is not a product. We could also define the behaviours $\mathbf{G_1}$ and $\mathbf{G_2}$ of circles and colours, respectively, and then we recover $\mathbf{G}$ as the intersection of $\mathbf{G_1}$ and $\mathbf{G_2}$, which in this case is a product.

There is more material in [35] and in [51], to which we refer for further reading:

- In ludics, one can define different sorts of commutative and non-commutative tensor products of designs and of behaviours. The basic idea is to glue two positive designs ϕ_1 and ϕ_2 starting respectively with $(+,\xi,I)$ and $(+,\xi,J)$ into a design starting with $(+,\xi,I\cup J)$, the problem being what to do on $I\cap J$: the non-commutative versions give priority to one of the designs, while the commutative versions place uniformly $\perp$ (resp. ✠) after all actions $(-,\xi k,L)$, where $k\in I\cap J$. The degenerate cases of the definition (tensoring with *Fid*) are a bit tricky, and have led Maurel to introduce an additional design, called negative divergence (see [51]).
- Ludics suggests new kinds of quantifications, in a vein similar to what we have seen for additives.
- Girard proves a full completeness result for a polarized version of MALL (cf. section 2) with quantifiers. To this aim, behaviours must be enriched with partial equivalence relations, to cope with the uniform, or parametric nature of proofs (think of axiom $\vdash X, X^\perp$ as parametric in the atom X).

- Exponentials can be handled by introducing a quantitative framework, where positive actions in designs are combined through formal probability trees: probabilities allow us to count repetitions, and to lift the separation property to a framework of designs with repetition and pointers (see [51]).

Exercise 5.12 *Show that two positive behaviours are disjoint if and only if their set intersection is* $\{\maltese\}$*, and that two negative behaviours* $\mathbf{G}$ *and* $\mathbf{G}'$ *are disjoint if and only if for all* $\psi \in \mathbf{G}$ *and* $\psi' \in \mathbf{G}'$ *we have* $|\psi|_{\mathbf{G}} \cap |\psi'|_{\mathbf{G}'} = \emptyset$ *(as sets of chronicles).*

Exercise 5.13 *Show that if* $\mathbf{G}$ *and* $\mathbf{H}$ *are positive and disjoint, then* $|\mathbf{G} \oplus \mathbf{H}| = |\mathbf{G}| \cup |\mathbf{H}|$.

Exercise 5.14 *What conditions should a function* θ *from addresses to addresses satisfy to be such that it induces a well defined transformation on designs, replacing every action, say,* $(+, \xi, I)$ *with* $(+, \theta(\xi), J)$ *such that* $\theta(\xi) \star J = \{\theta(\xi i) \mid i \in I\}$*? Such a function is called a* delocation *function.*

Exercise 5.15 *Show that for any two behaviours* $\mathbf{G_1}$ *and* $\mathbf{G_2}$ *on the same base, one can find two delocation functions* θ_1 *and* θ_2 *such that* $\theta_1(\mathbf{G_1})$ *and* $\theta_2(\mathbf{G_2})$ *are disjoint.*

Exercise 5.16 *Show that as operations on behaviours* $\otimes$ *and* $\oplus$ *are such that* $\otimes$ *distributes over* $\oplus$.

Acknowledgements

I wish to thank Song Fangmin and the University of Nanjing for an invitation in October 2001, which offered me the occasion to give a few lectures on linear logic, and Aldo Ursini and the University of Siena for providing most hospitable conditions for me to write this survey article and to give the corresponding course.

References

[1] M. Abadi, L. Cardelli, P.-L. Curien, and J.-J. Lévy, Explicit substitutions, Journal of Functional Programming 1(4), 375-416 (1992).

[2] S. Abramsky and R. Jagadeesan, Games and full completeness for multiplicative linear logic, Journal of Symbolic Logic 59, 543-574 (1994).

[3] S. Abramsky and G. McCusker, Game semantics, in Computational Logic, U. Berger and H. Schwichtenberg eds, Springer Verlag (1999).

[4] R. Amadio and P.-L. Curien, Domains and lambda-calculi, Cambridge University Press (1998).

[5] J.-M. Andreoli: Logic Programming with Focusing Proofs in Linear Logic. Journal of Logic and Compuation 2(3), 297-347 (1992).

J.-M. Andreoli and R. Pareschi, Linear objects: logical processes with built-in inheritance, New Generation Computing 9(3-4), 445-473 (1991).

[6] A. Asperti, V. Danos, C. Laneve, and L. Regnier, Paths in the lambda-calculus, in Proc. Logic in Computer Science 1994.

[7] A. Asperti and S. Guerrini, The optimal implementation of functional programming languages, Cambridge University Press (1998).

[8] A. Asperti and C. Laneve, Paths, computations and labels in the λ-calculus, Thoretical Computer Science 142(2), 277-297 (1993).

[9] P. Baillot, Approches dynamiques en sémantique de la logique linéaire: jeux et géométrie de l'interaction, Thèse de Doctorat, Université Aix-Marseille II (1999).

[10] H. Barendregt, M. Coppo, and M. Dezani, A filter lambda model and the completeness of type assignment, Journal of Symbolic Logic 48, 931-940 (1983).

[11] G. Berry and P.-L. Curien, Sequential algorithms on concrete data structures, Theoretical Computer Science 20, 265-321 (1982).

[12] K. Bruce and G. Longo, A modest model of records, inheritance and bounded quantification, Information and Computation 87, 196-240 (1990).

[13] L. Cardelli and P. Wegner, On understanding types, data abstraction, and polymorphism, Computing Surveys 17(4), 471-522 (1985).

[14] R. Cartwright, P.-L. Curien, and M. Felleisen, Fully abstract models of observably sequential languages, Information and Computation 111 (2), 297-401 (1994).

[15] Coppo, M., Dezani, M.A., and Sallé, P., Functional Characterization of some Semantic Equalities inside lambda-calculus, ICALP Graz, Lecture Notes in Computer Science, Springer Verlag, 1979.

[16] P.-L. Curien and H. Herbelin, Computing with abstract Böhm trees, Third Fuji International Symposium on Functional and Logic Programming, April 1998, Kyoto, World Scientific (1998)

[17] P.-L. Curien, Abstract Böhm trees, Mathematical Structures in Computer Science 8(6), 559-591 (1998).

[18] P.-L. Curien, G. Plotkin, and G. Winskel, Bistructure models of linear logic, Milner Festschrift, MIT Press (1999).

[19] P.-L. Curien, Sequential algorithms as bistable maps, to appear (2004).

[20] P.-L. Curien, Symmetry and interactivity in programming, Bulletin of Symbolic Logic 9, 169-180 (2003).

[21] V. Danos, La logique linéaire appliquée à l'étude de divers processus de normalization (principalement du λ-calcul), Thèse de Doctorat, Université Paris 7 (1990).

[22] V. Danos and L. Regnier, Local and asynchronous beta-reduction, Proc. of Logic in Computer Science 1993.

[23] V. Danos and L. Regnier, The structure of multiplicatives, Archive for Mathematical Logic 28, 181-203 (1989).

[24] C. Faggian, Travelling on designs: ludics dynamics, in Proc. CSL 2002, Lecture Notes in Computer Science 2471 (2002).

[25] C. Faggian and M. Hyland, Designs, disputes and strategies, in Proc. CSL 2002, Lecture Notes in Computer Science 2471 (2002).

[26] C. Faggian, Interactive observability in ludics, in Proc. of ICALP 2004, Lecture Notes in Computer Science (2004).

[27] G. Gonthier, M. Abadi and J.-J. Lévy, The geometry of optimal lambda reduction, in Proc. Principles of Programming Languages 1992.

[28] G. Gonthier, M. Abadi and J.-J. Lévy, Linear logic without boxes, in Proc. Logic in Computer Science 1992.

[29] J.-Y. Girard, Linear logic, Theoretical Computer Science 50, 1-102 (1987).

[30] J.-Y. Girard, Geometry of interaction I: interpretation of system F, in Proc. Logic Colloquium '88, 221-260, North Holland (1989).

[31] J.-Y. Girard, A new constructive logic: classical logic, Mathematical Structures in Computer Science 1, 255-296 (1991).

[32] J.-Y. Girard, Linear logic: it's syntax and semantics, in Advances of Linear Logic, J.-Y. Girard, Y. Lafont, and L. Regnier eds, Cambridge University Press, 1-42 (1995).

[33] J.-Y. Girard, On the meaning of logical rules I: syntax vs. semantics, in Computational Logic, U. Berger and H. Schwichtenberg eds, 215-272, Nato Series F 165, Springer (1999).

[34] J.-Y. Girard, On the meaning of logical rules II: multiplicative/additive case, in Foundation of Secure Computation, NATO series F 175, 183-212, IOS Press (2000).

[35] J.-Y. Girard, Locus solum: from the rules of logic to the logic of rules, Mathematical Structures in Computer Science 11(3), 301-506 (2001).

[36] J.-Y. Girard, From foundations to ludics, Bulletin of Symbolic Logic 9, 131-168 (2003).

[37] S. Guerrini, Correctness of multiplicative proof nets is linear, in Proc. Logic in Computer Science 1999.

[38] S. Guerrini and A. Masini, Parsing MELL proof nets, Theoretical Computer Science 254 (1-2), 317-335 (2001)..

[39] Theoretical aspects of object-oriented proigramming: types, semantics, and language design, C. Gunter and J. Mitchell eds., MIT Press (1994).

[40] R. Hindley, The completeness theorem for typing lambda-terms, Theoretical Computer Science 22, 1-17 (1983).

[41] G. Huet and D. Oppen, Equations and Rewrite Rules: a Survey, in Formal Language Theory: Perspectives and Open Problems, R. Book (ed.), Academic Press, 349-405 (1980).

[42] D. Hughes and R. van Glabbeek, Proof nets for unit-free multiplicative-additive linear logic, in Proc. Logic in Computer Science (2003).

[43] M. Hyland and L. Ong, On Full Abstraction for PCF, Information and Computation 163, 285-408 (2000).

[44] J.-L. Krivine, Lambda-calculus, types and models, Ellis Horwood (1993).

[45] J. Laird, Bistability: an extensional characterization of sequentiality, in Proc. Computer Science Logic 2003, Springer LNCS 2803 (2003).

[46] J. Lamping, An algorithm for optimal lambda calculus reduction. Proc. Principles of Programming Languages 1990, 16-30.

[47] O. Laurent, Etude de la polarisation en logique, PhD Thesis, Université Aix-Marseille II (2002).

[48] O. Laurent, Classical isomorphisms of types, to appear in Mathematical Structures in Computer Science (2004).

[49] O. Laurent, A proof of the focalization property of linear logic, draft (2003).

[50] J.-J. Lévy, Optimal reductions in the lambda-calculus, in To H.B. Curry: Essays in Combinatory Logic, Lambda Calculus and Formalism, J. Seldin and R. Hindley eds, 159-191, Academic Press (1980).

[51] F. Maurel, Un cadre quantitatif pour la ludique, PhD Thesis, Paris 7 Univ. (2004).

[52] J. Mitchell, Polymorphic type inference and containment, Information and Computation 76, 211-249 (1988).

[53] L. Regnier, Lambda-calcul et réseaux, Thèse de Doctorat, Université Paris 7 (1992).

[54] A. Saurin, A note on focalization property and synthetic connectives, draft (2004).

[55] L. Tortora de Falco, Additives of linear logic and normalization – Part I: a (restricted) Church-Rosser property, Theoretical Computer Science 294 (3), 489-524 (2003).