From focalization of logic to the logic of focalization *

Michele Basaldella¹, Alexis Saurin² and Kazushige Terui¹

¹ RIMS, Kyoto University, Japan. {mbasalde,terui}@kurims.kyoto-u.ac.jp
² PPS and INRIA πτ², France. alexis.saurin@pps.jussieu.fr

Abstract. Focalization property is a deep outcome of linear logic proof theory, putting to the foreground the role of polarity in logic. It resulted in important advances in various fields, ranging from linear logic programming to game theoretical analysis of linear logic proofs. The aim of this article is to propose an algebraic study of focalization property for the multiplicative-additive fragment of linear logic (MALL) in the framework of ludics.

1 Introduction

Focalization is a deep outcome of linear logic (LL) proof theory, putting to the foreground the role of polarity in logic. It resulted in important advances in various fields ranging from proof-search (the original motivation for Andreoli’s study [1] of focalization) and the ability to define synthetic connectives and hypersequentialized calculi [8,9] to game semantical analysis of logic.

In particular, focalization deeply influenced Girard’s ludics [10] which is a pre-logical framework which aims to analyze various logical and computational phenomena at a foundational level. For instance, the concluding results of loc. cit. are a full completeness theorem with respect to focalized multiplicative–additive linear logic MALL. Another characteristics of ludics is that types are built from untyped proofs (called designs). More specifically, types (called behaviours) are sets of designs closed under a certain closure operation. This view of types as sets of proofs opens a new possibility to discuss focalization and other properties of proofs at the level of types.

The purpose of this paper is to show that ludics is suitable for analyzing focalization and that this interactive analysis of focalization is fruitful. In particular, our study of focalization in ludics was primarily motivated by the concluding remarks of the third author’s paper on computational ludics [14] where focalization on data designs was conjectured to correspond to the tape compression theorem of Turing machines (see appendix A.2 for details).

Still, for the very reason that ludics abstracts over focalization (being built on hypersequentialized calculi) it is not clear whether an analysis of focalization can (or shall) be pursued in ludics: there seems to be no room to discuss and

* A preliminary version of this work will appear in [3].
prove focalization internally. This can however be settled by using a dummy shift operator. For instance, a compound formula \( L \oplus (M \otimes N) \) of LL can be expressed in ludics in two ways; either as a flat behaviour \( \oplus \otimes (L, M, N) \) built by a single synthetic connective \( \oplus \otimes \) from three sub–behaviours \( L, M, N \), or as a compound behaviour \( L \oplus \uparrow (M \otimes N) \), which consists of three layers: \( M \otimes N \) (positive), \( \uparrow (M \otimes N) \) (negative), and \( L \oplus \uparrow (M \otimes N) \) (positive).

Focalization can then be expressed as a mapping from the latter to the former behaviour. Hence we can deal with it as if it were an algebraic law, which may be compared with other logical isomorphisms such as associativity, distributivity, etc. To be precise, however, focalization is not an isomorphism but is an asymmetric relation. In this paper, we think of it as a retraction \( L \oplus \uparrow (M \otimes N) \rightarrow \oplus \otimes (L, M, N) \) which comes equipped with a section \( \oplus \otimes (L, M, N) \rightarrow L \oplus \uparrow (M \otimes N) \). (The latter is reminiscent of the CPS-translation in \( \lambda \)- calculus: \( A \rightarrow B \rightarrow \neg \neg A \rightarrow \neg \neg B \), as noticed by [6]. It might be possible to think of focalization as an abstract form of CPS-translation; this gives us another motivation to study focalization in depth.)

The aim of our current work is to promote this “algebraic” view of focalization in the setting of ludics. Furthermore, the pair of a retraction and the corresponding section can be naturally lifted by applications of logical connectives (Theorem 4). Hence we also have focalization inside a compound behaviour (or inside a context). This allows us to recover the original focalization theorem as a corollary to our “algebraic” focalization, via the full completeness theorem of Ludics with respect to MALL.

**Focalization in linear logic.** LL comes from a careful analysis of structural rules in sequent calculus resulting in a very structured proof theory, in particular regarding dualities. A fundamental outcome of those dualities is Andreoli’s discovery [1] of focalization, providing the first analysis of polarities in linear logic. Andreoli’s contribution lies mainly in the splitting of logical connectives in two groups – positive (\( \otimes, \oplus, 0, 1, \exists, \)!) and negative (\( \forall, \&, \top, \bot, \forall, ? \)) connectives.

The underlying meaning of this distinction comes from proof-search motivations. The introduction rules for negative connectives \( \forall, \&, \top, \bot, \forall \) are reversible: in the bottom–up reading, the rule is deterministic, i.e., there is no choice to make and provability of the conclusion implies provability of the premises. On the other hand, the introduction rules for positive connectives involve choices: e.g., splitting the context in \( \otimes \) rule, or choosing between \( \oplus_L \) and \( \oplus_R \) rules, resulting in possible erroneous choices during proof-search. Still, positive connectives satisfy a strong property called focalization [1]: let us consider a sequent \( \vdash F_0, \ldots, F_n \) containing no negative formulas, then there is (at least) one formula \( F_i \) which can be used as a focus for the search by hereditarily selecting \( F_i \) and its positive subformulas as principal formulas up to the first negative subformulas. This property induces the following strategy of proof–search called focalization discipline:
A sequent proof is called *focussing* if it respects the focalization discipline. It is proven in [1] that a provable sequent is provable with a focussing proof: the focalization discipline is therefore a complete proof-search strategy. Other approaches to focalization consider proof transformation techniques [12,13].

A very important consequence of focalization is the possibility to consider *synthetic connectives* [9,5]: a synthetic connective is a maximal cluster of connectives of the same polarity. They are built modulo commutativity and associativity of binary connectives and some syntactical isomorphism [11] of LL. For MALL, the underlying syntactical isomorphism in action is the *distributivity* of $\otimes$ with respect to $\oplus$, namely $(A \oplus B) \otimes C \cong (A \otimes C) \oplus (B \otimes C)$ and its dual.

Synthetic connectives give rise to hypersequentialized versions of LL (and more precisely of MALL) which are at the basis of Girard’s construction of Ludics.

## 2 Untyped designs

### 2.1 Syntax

We recall the term syntax for designs introduced in [14] which uses a process calculus notation inspired by the close relationship between ludics and linear $\pi$-calculus [7].

Designs are built over a given signature $\mathcal{A} = (A, \mathbf{ar})$, where $A$ is a set of names $a, b, c, \ldots$ and $\mathbf{ar}: A \rightarrow \mathbb{N}$ assigns an arity $\mathbf{ar}(a)$ to each name $a$. Let $\mathcal{V}$ be a countable set of variables $\mathcal{V} = \{x, y, z, \ldots\}$. Over a fixed signature $\mathcal{A}$, a (proper) positive action is $a$ with $a \in A$, and a (proper) negative action is $a(x_1, \ldots, x_n)$ where $x_1, \ldots, x_n$ are distinct variables and $\mathbf{ar}(a) = n$. In the sequel, an expression of the form $a(\vec{x})$ always stands for a negative action.

**Definition 1 (Designs).** The positive (resp. negative) designs $P$ (resp. $N$) are coinductively generated by the following grammar:

$$
\begin{align*}
P ::= & \quad \Omega \text{ (partiality)} \mid \mathbf{\mathcal{X}} \text{ (daimon)} \mid N_0 \mid \mathcal{X}(N_1, \ldots, N_n) \text{ (application)}, \\
N ::= & \quad x \text{ (variable)} \mid \sum a(\vec{x}).P_a \text{ (abstraction)},
\end{align*}
$$

where $\mathbf{ar}(a) = n$ and $\vec{x} = x_1, \ldots, x_n$. The formal sum $\sum a(\vec{x}).P_a$ is built from $|A|$-many components $\{a(\vec{x}).P_a\}_{a \in A}$.

Designs may be considered as infinitary $\lambda$-terms with named applications and superimposed abstractions. We use meta-variables $P, Q, \ldots$ (resp. $N, M, \ldots$, resp. $D, E, X, Y, \ldots$) to denote positive (resp. negative, resp. arbitrary) designs. Any subterm $E$ of $D$ is called a subdesign of $D$.  

<table>
<thead>
<tr>
<th>Sequent $\Gamma$ contains a negative formula</th>
<th>Sequent $\Gamma$ contains no negative formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>choose any negative formula (e.g. the leftmost one) and decompose it using the only possible negative rule</td>
<td>choose some positive formula and decompose it (and its subformulas) hereditarily until we get to atoms or negative subformulas</td>
</tr>
</tbody>
</table>
The design $\Omega$ is used to encode partial sums: given a set $\alpha = \{ a(x), b(y), \ldots \}$ of negative actions, we write $a(x).P_a + b(y).P_b + \cdots$ to denote the negative design $\sum a(x).R_a$, where $R_a = P_a$ if $a(x) \in \alpha$, and $R_a = \Omega$ otherwise.

A design $D$ may contain free and bound variables. An occurrence of subterm $a(x).P_a$ binds the free-variables $x$ in $P_a$. Variables which are not under the scope of any binder $a(x)$ are free. $fv(D)$ denotes the set of free variables occurring in $D$. Designs are always considered up to $\alpha$-equivalence, that is up to renaming of bound variables (see [14] for further details).

A positive design which is neither cut-free nor $\emptyset$ is either of the form $(\sum a(x).P_a)(\overline{x}(N_1, \ldots, N_n))$ and called a cut or of the form $x[\overline{x}(N_1, \ldots, N_n)]$ and called a head normal form. The head variable $x$ in the design above plays the same role as a pointer does in a strategy from Hyland-Ong’s games model and an address (or locus) in Girard’s ludics. On the other hand, a variable $x$ occurring in a bracket (as in $N_0[\overline{x}(N_1, \ldots, N, x, N_{+1}, \ldots, N_n)$) does not correspond to a pointer nor address but rather to an identity axiom (initial sequent) in sequent calculus, and for this reason is called an identity.

A design $D$ is said: total, if $D \neq \Omega$; linear (or affine), if for any subdesign of the form $N_0[\overline{x}(N_1, \ldots, N_n)$, the sets $fv(N_0), \ldots, fv(N_n)$ are pairwise disjoint.

The reduction relation $\rightarrow$ is defined on positive designs by:

$$(\sum a(x_1, \ldots, x_n).P_a)(\overline{x}(N_1, \ldots, N_n)) \rightarrow P_a[N_1/x_1, \ldots, N_n/x_n];$$

where $D[N_1/x_1, \ldots, N_n/x_n]$ denotes the simultaneous and capture-free substitution of $N_1, \ldots, N_n$ for $x_1, \ldots, x_n$ in $D$. We write $\rightarrow^*$ for transitive closure of $\rightarrow$. Given two positive designs $P, Q$, we write $P \Downarrow Q$ if $P \rightarrow^* Q$ and $Q$ is neither a cut nor $\emptyset$. We write $P \Downarrow$ if there is no $Q$ such that $P \Downarrow Q$.

Given a design $D$, its normal form $[D]$ is defined by corecursion as follows:

$$[[P]] = \emptyset \quad \text{if} \quad P \Downarrow \emptyset,$$

$$= x[\overline{x}(\overline{N})] \quad \text{if} \quad P \Downarrow x[\overline{x}(\overline{N})];$$

$$= \emptyset \quad \text{if} \quad P \Downarrow;$$

Normalization is associative:

$$[[D[N_1/x_1, \ldots, N_n/x_n]]] = [[[D]][[N_1]][[x_1, \ldots, [N_n]]/x_n]].$$

### 2.2 Functionals

In the rest of this work, we are mainly interested in the special subclass of total, cut-free, linear and identity-free designs, corresponding to the original notion of design [10]. We call standard any design which fulfils the above requirement. A very important subclass of standard designs is the one consisting of atomic designs.

A positive design $P$ is closed if $fv(P) = \emptyset$, atomic if $fv(P) \subseteq \{x_0\}$ for a certain fixed variable $x_0$. A negative design $N$ is atomic if $fv(N) = \emptyset$. The variable $x_0$ has to be thought to play the same role as the empty address “()” does in [10], i.e.,
it is a fixed and predetermined “location”. We denote by $\mathcal{D}$ the set consisting of all atomic standard designs, by $\mathcal{D}^+$ (resp. $\mathcal{D}^-$) its restriction to positive (resp. negative) designs.

We now introduce a class of designs of our main interest.

**Definition 2 (Functionals).** We call a negative standard design $N$ a **functional** whenever $fv(N) \subseteq \{x_0\}$.

We use meta-variables $f, g, h, \ldots$ to denote functionals.

Any functional $f$ can be thought as bi-directional function which sends atomic designs to atomic designs of the same polarity. Given an atomic positive design $P$, we can apply $f$ to $P$ by $f^+(P) := [P[f/x_0]]$. The result is either a positive atomic design or $\Omega$, which can be seen as a coding of “undefined”. So, the operation $f^+$ can be seen as a partial map $f^+ : \mathcal{D}^+ \rightarrow \mathcal{D}^+$. Similarly, given an atomic negative design $N$, $f^-(N) := [f[N/x_0]]$ is another atomic negative design. So, $f^* : \mathcal{D}^- \rightarrow \mathcal{D}^-$. 

**Lemma 1 (Duality).** For any $P \in \mathcal{D}^+$ and $N \in \mathcal{D}^-$,

$$[f^+(P)[N/x_0]] = [P[f^*(N)/x_0]].$$

(1)

**Proof.** Easy consequence of associativity: $[[f^+(P)[N/x_0]]] = [[P[f/x_0]][N/x_0]] = [P[f/x_0][N/x_0]] = [P[f[N/x_0]/x_0]] = [P[f^*(N)/x_0]].$ 

A very important functional is the *fax* [10] (or $\eta$-expanded identity, copycat strategy) recursively defined by the equation:

$$id := \sum a(x_1, \ldots, x_n) x_0 \mathfrak{π}(id(x_1), \ldots id(x_n))$$

(2)

where $id(x_k) := id[x_k/x_0]$ for any $1 \leq k \leq n$. The design $id$ plays the role of the identity function for (standard) designs, in particular: $id^+(P) = P$, for any $P \in \mathcal{D}^+$ and $id^-(N) = N$, for any $N \in \mathcal{D}^-$ (see [14] for a proof of this fact).

As said before, $f^* : \mathcal{D}^+ \rightarrow \mathcal{D}^+$ can be seen as a *partial* function, hence it makes sense to define its domain as $\text{Dom}(f) := \{P : f^+(P) \neq \Omega\}$.

For the identity functional $id$, it is immediate to check that $\text{Dom}(id) = \mathcal{D}^+$, hence $id$ is defined everywhere. On the other hand, let us consider the functional $id_a$ defined for any name $a$ by the equation:

$$id_a := a(x_1, \ldots, x_n) x_0 \mathfrak{π}(id(x_1), \ldots id(x_n))$$

(3)

Then $id_a^*(P) = P$ if $P$ of the form $x_0 \mathfrak{π}(N_1, \ldots, N_n)$ or $\bigstar$, and it is undefined i.e., $P \notin \text{Dom}(id_a)$ otherwise.

Functionals can be also composed together: for any $f, g$, we define the **positive composition** $g \circ^+ f := [g[f/x_0]]$ and the **negative composition** $g \circ^- f := [g[f/x_0]]$. Then, it is easy to see that $\circ^+$ and $\circ^-$ are associative and have the unit $id$. Furthermore, we have:

**Proposition 1 (Untyped composition).** For any $P \in \mathcal{D}^+$ and $N \in \mathcal{D}^-$,

$$(g \circ^+ f)^*(P) = g^*(f^+(P)), \quad (g \circ^- f)_*(N) = g_*(f_*(N)).$$
3 Synthetic connectives

3.1 Synthetic signature

Let \( A = (A, \text{ar}) \) be a signature. Let us denote by \( A^n \) the subset of \( A \) consisting of those names of arity \( n \), i.e., \( A^n := \{ a \in A : \text{ar}(a) = n \} \). We call a signature synthetic if it satisfies the following requirements:

- for any \( a \in A^n \), \( b \in A^m \) and \( 1 \leq i \leq n \) there exists a name, denoted by \( a[b/i] \), such that \( a[b/i] \in A^{n+m-1} \).
- \( a[b/i] = c[d/j] \) only if \( a = c \), \( b = d \) and \( i = j \).

From now on, we always assume that our signature \( A = (A, \text{ar}) \) is synthetic and equipped with a unary name \( \uparrow \), that we call the dummy shift operator. We denote by \( \downarrow \) the positive action \( \uparrow \), and abbreviate \( \uparrow(x).x\pi(N) \) by \( \pi(N) \).

As a convention, given disjoint sequences of variables \( \vec{x} = x_1, \ldots, x_n \) and \( \vec{y} \) and \( 1 \leq i \leq n \), we denote by the expression \( \vec{x}[\vec{y}/i] \) the sequence of variables \( x_1, \ldots, x_{i-1}, \vec{y}, x_{i+1}, \ldots, x_n \).

3.2 Logical and synthetic connectives

Informally, a logical connective is given by specifying (i) place holders for subformulas, and (ii) inference rules associated to the connective. In our setting, (i) is embodied by a sequence of variables and (ii) is by a set of negative actions.

**Definition 3 (Logical connective).** An \( n \)-ary logical connective \( \alpha \) is a pair \( (\mathcal{V}_\alpha, \{a_1(\vec{x}_1), \ldots, a_m(\vec{x}_m)\}) \) where \( \mathcal{V}_\alpha = z_1, \ldots, z_n \) is a finite sequence of distinct variables called the directory of \( \alpha \) and \( \{a_1(\vec{x}_1), \ldots, a_m(\vec{x}_m)\} \) is a finite set of negative actions, called the body of the connective, such that the names \( a_1, \ldots, a_m \) are distinct and each \( \vec{x}_i \) is a subsequence of \( \mathcal{V}_\alpha \).

In the sequel, we write \( a(\vec{x}) \in \alpha \) whenever \( a(\vec{x}) \) belongs to the body of \( \alpha \). We denote by \( \Lambda(A) \) the set of logical connectives over a (synthetic) signature \( A \) and we use meta-variables \( \alpha, \beta, \gamma, \ldots \) to denote logical connectives. We write \( \Lambda^n(A) \) for the subset of \( \Lambda(A) \) consisting of \( n \)-ary connectives.

Since variables are just used as place holders, we naturally identify two logical connectives if one is obtained from another by renaming the variables. Hence given two logical connectives \( \alpha \) and \( \beta \), we may always assume that the directories \( \mathcal{V}_\alpha \) and \( \mathcal{V}_\beta \) are disjoint, i.e., do not share a variable.

A synthetic signature allows us to synthesize two logical connectives.

**Definition 4 (Synthetic connective).** Let \( \alpha \) and \( \beta \) be logical connectives with \( \mathcal{V}_\alpha = z_1, \ldots, z_n \) and \( \mathcal{V}_\beta = w_1, \ldots, w_m \). Given \( 1 \leq i \leq n \), we define a new logical connective \( \gamma = \text{synth}(\alpha, \beta, i) \in \Lambda^{n+m-1}(A) \) which we call the synthetic connective associated to \( (\alpha, \beta, i) \) as follows.

- The directory of \( \gamma \) is \( \mathcal{V}_\alpha[\mathcal{V}_\beta/i] = z_1, \ldots, z_{i-1}, w_1, \ldots, w_m, z_{i+1}, \ldots, z_n \).
- The body of \( \gamma \) consists of the negative actions:
• \( a(\vec{x}) \in \alpha \) such that \( z_i \notin \{ \vec{x} \} \);
• \( a[b/j](\vec{x}[\vec{y}/j]) \) such that \( a(\vec{x}) \in \alpha \), \( b(\vec{y}) \in \beta \), \( \vec{x} = x_1, \ldots, x_k \) and \( z_i = x_j \).

Usual and synthetic MALL connectives can be defined in a synthetic signature \( \mathcal{L} = (L, \alpha) \) containing unary names \( \uparrow, \pi_1, \pi_2 \) and a binary name \( \varphi \). We define:

\[ \top := \emptyset, \quad \forall := \{ x_1, x_2, \{ \varphi(x_1, x_2) \} \}, \quad \uparrow := \{ x, \{ \uparrow(x) \} \}, \quad \& := \{ y_1, y_2, \{ \pi_1(y_1), \pi_2(y_2) \} \}. \]

Let us now build a new synthetic connective \( \gamma = \text{synth}(\forall, \& , 1) = (y_1, y_2, x_2, \{ \varphi[\pi_1/1] (y_1, x_1), \varphi[\pi_2/1](y_2, x_1) \}). \) It is a logical connective with the following inference rules:

\[
\frac{\vdash \Gamma, P, R \quad \vdash \Gamma, Q, R}{\vdash \Gamma, \gamma(P, Q, R)} \quad \frac{\vdash \Gamma, N \quad \vdash \Delta, K}{\vdash \Gamma, \Delta, \pi(N, M, K)} \quad \frac{\vdash \Gamma, M \quad \vdash \Delta, K}{\vdash \Gamma, \Delta, \pi(N, M, K)} \quad \frac{\vdash \Gamma}{\vdash \Gamma, N}
\]

It is clear that the rule \( \gamma \) is a combination of the rules \( \& \) and \( \forall \), while \( \pi_1 \) and \( \pi_2 \) are combinations of \( \& \), \( \&_1 \) and \( \&_2 \).

### 4 Focalizing designs

In the sequel, given a sequence of variables \( \vec{z} = z_1, \ldots, z_k \) we denote by the expression \( \text{id}(\vec{z}) \) the sequence of functionals \( \text{id}(z_1), \ldots, \text{id}(z_k) \). With this notation, \( \text{id} \) can be succinctly expressed by \( \sum a(\vec{x}) . x_0 \pi(\text{id}(\vec{x})) \).

**Focalizing and inverting designs.** Given two logical connectives \( \alpha, \beta \) with \( \forall_{\alpha} = z_1, \ldots, z_n, \forall_{\beta} = w_1, \ldots, w_m \) and \( 1 \leq i \leq n \), we define two functionals: **focalizing design** \( f(\alpha, \beta, i) \) and **inverting design** \( u(\alpha, \beta, i) \). Below, we assume that \( a(\vec{x}) \in \alpha \) and \( \vec{x} = x_1, \ldots, x_k \).

\[
\begin{align*}
{f}(\alpha, \beta, i) & := \text{id}_a & \text{if } z_i \notin \{ \vec{x} \}, \\
{f}(\alpha, \beta, i) & := a(\vec{x}).z_i \downarrow (\sum b(\vec{y}) . x_0 [a[b/j](\text{id}(\vec{x}[\vec{y}/j]))]) & \text{if } z_i = x_j, \\
{f}(\alpha, \beta, i) & := \sum \alpha . f(\alpha, \beta, i); & \\
u(\alpha, \beta, i) & := \text{id}_a & \text{if } z_i \notin \{ \vec{x} \}, \\
u(\alpha, \beta, i) & := a[b/j](\vec{x}[\vec{y}/j]).x_0 \pi(\text{id}(\vec{x})) & \text{if } z_i = x_j, \\
u(\alpha, \beta, i) & := \sum \gamma . u(\alpha, \beta, i), & \quad (4)
\end{align*}
\]

where \( \vec{x} = \vec{x}_l, x_j, \vec{x}_r \) in the definition of \( u(\alpha, \beta, i) \) and \( \gamma = \text{synth}(\alpha, \beta, i) \) in that of \( u(\alpha, \beta, i) \).

To see how they work, consider:

\[
\begin{align*}
f &= f(\forall, \& , 1) = \varphi(x_1, x_2).x_1 \downarrow (\sum_{i=1,2} \pi_i(y_i).x_0 [\varphi[\pi_i/1](\text{id}(y_i), \text{id}(x_2))]), \\
u &= u(\forall, \& , 1) = \sum_{i=1,2} \varphi[\pi_i/1](y_i, x_2).x_0 \pi(\text{id}(y_i), \text{id}(x_2)).
\end{align*}
\]
Consider also the following atomic designs:

\[ P_1 := x_0[\varphi(\uparrow \pi_1(M), N)], \quad N_1 := \varphi(x_1, x_2).x_1\downarrow (\pi_1(y_1).P_1 + \pi_2(y_2).P_2), \]
\[ P_2 := x_0[\varphi(\pi_1/1)(M, N)], \quad N_2 := \varphi(\pi_1/1)(y_1, x_2).P_1 + \varphi(\pi_2/1)(y_2, x_2).P_2. \]

We can calculate \( f^*(P_1) \) by normalization:

\[
\begin{align*}
f^*(P_1) &= \left[ f \mid \varphi(\uparrow \pi_1(M), N) \right] \\
&= \left[ \left[ \downarrow \varphi(\pi_1(M)) \right] \left[ \downarrow (\pi_1(y_1).x_0[\varphi(\pi_1/1)(id(y_1), id(x_2)[N/x_2]))] \right] \right] \\
&= \left[ \left[ \pi_1(y_1).x_0[\varphi(\pi_1/1)(id(y_1), id(x_2)[N/x_2))] \right] \mid \pi_1(M) \right] \\
&= \left[ x_0[\varphi(\pi_1/1)(id(y_1)[M/y_1], id(x_2)[N/x_2])] \right] \\
&= x_0[\varphi(\pi_1/1)(id_*(M), id_*(N))] = P_2.
\end{align*}
\]

Similarly, we obtain

\[ u^*(P_2) = P_1, \quad u_*(N_1) = N_2, \quad f_*(N_2) = N_1. \]

Observe that \( f^* \) (resp. \( u_* \)) “collapses” three polarity layers into one when applied to a positive (resp. negative) design, while \( u^* \) (resp. \( f_* \)) “cancels” the effect of \( f^* \) (resp. \( u_* \)). Hence we could informally claim that \((f^*, u^*)\) internally accounts for focalization of positive connectives, while \((u_*, f_*)\) internally accounts for invertibility of negative connectives.

**Proposition 2 (Focalization-Inversion).** Let \( f = f_{(\alpha, \beta, i)} \) and \( u = u_{(\alpha, \beta, i)}. \)

\[
f \circ u = id_\gamma, \text{ where } \gamma = \text{synth}(\alpha, \beta, i) \text{ and } id_\gamma = \sum_{c(x) \in \gamma} id_c.
\]

- \( u \circ f \) is idempotent: \((u \circ f) \circ (u \circ f) = u \circ f.\)

The equation \( f \circ u = id_\gamma \) roughly states that \( f \) and \( u \) are opposite operations. Later we shall state more precisely that focalizing designs are retractions of inverting designs. That will formally verify the intuition that focalization of positive rules is dual to invertibility of negative rules \(\text{[10]}.\)

The idempotency of \( u \circ f \) also has a proof-theoretical reading: suppose that we have a (possibly not focalized) proof \( \pi \) in \text{MALL} built by usual connectives and a function \( f \) which transforms it into a focalized proof made by synthetic connectives \( \pi^f. \) Suppose also that we have a converse operation \( u \) which transforms a proof \( \rho \) made by synthetic connectives back into the usual one \( \rho^u \) in \text{MALL}. The idempotency of \( u \circ f \) says that even if \( \pi \neq (\pi^f)^u \) (as it happens in general) we have that \((\pi^f)^u = ((\pi^f)^u)^u \) and hence \((\pi^f)^u \) could be considered as a focalization of \( \pi \) which stays within \text{MALL}.

### 5 Types and interactive functions

We recall the notion of type (behaviour) and then we move on to the discussion about functionals in the typed setting.
5.1 Orthogonality and behaviours

Two atomic designs $P, N$ of opposite polarities are said orthogonal (written $P \perp N$) when $[P[N/x_0]] = \mathcal{X}$. If $X$ is a set of atomic designs of the same polarity, then its orthogonal set is defined by $X^\perp := \{E : \forall D \in X, D \perp E\}$.

In terms of orthogonality, Equation (1) can be nicely expressed as:

$$f^*(P) \perp N \text{ if and only if } P \perp f_*(N).$$

A behaviour is a set $X$ of atomic designs of the same polarity such that $X^\perp = X$. A behaviour is positive or negative according to the polarity of its designs. We denote positive behaviours by $P, Q, R, \ldots$ and negative behaviours by $N, M, K, \ldots$.

There are the least and the greatest behaviours among all positive (resp. negative) behaviours with respect to set inclusion:

$$0 := \{\mathcal{X}^+\}, \quad 0 := \{\mathcal{X}^-\}, \quad \top := \mathcal{X}^+ = \mathcal{D}^+, \quad \bot := \mathcal{X}^- = \mathcal{D}^-$$

where $\mathcal{X}^- := \sum a(f)\mathcal{X}$ is called negative daimon in [10]. Notice also that $P \in D^+, P \neq \Omega$ and $P \perp \mathcal{X}^-$ are perfectly equivalent expressions.

**Theorem 1.** For any functional $f$, $\text{Dom}(f)$ is a behaviour.

*Proof.* The inclusion $\text{Dom}(f) \subseteq \text{Dom}(f)^\perp$ is immediate. For the other inclusion, it is sufficient to prove that for any $Q \in \text{Dom}(f)^\perp$, $f^*(Q) \perp \mathcal{X}^-$. Let $P \in \text{Dom}(f)$. Since $f^*(P) \perp \mathcal{X}^-$, by Equation (3), $P \perp f_*(\mathcal{X}^-)$. This holds for any $P \in \text{Dom}(f)$, hence $f_*(\mathcal{X}^-) \in \text{Dom}(f)^\perp$. For $Q \in \text{Dom}(f)^\perp$, we have by orthogonality $Q \perp f_*(\mathcal{X}^-)$ and again by Equation (3), $f^*(Q) \perp \mathcal{X}^-$. □

We are now ready to assign “types” to functionals.

**Definition 5 (Function space).** Let $P, Q$ be positive behaviours. We define the positive function space as the set of functionals $P \rightarrow Q := \{f : \forall P \in P, f^*(P) \in Q\}$. Analogously, given negative behaviours, we define $N \rightarrow M := \{f : \forall N \in N, f_*(N) \in M\}$. We write $f : X \rightarrow Y$ whenever $f \in X \rightarrow Y$.

For instance, we have $id : P \rightarrow P$ and $id : N \rightarrow N$ for any behaviours $P, N$.

In practice, when we want to calculate $f \in X \rightarrow Y$, we can use the following:

$$f : P \rightarrow Q \iff \forall N \in Q^+, \forall P \in P, [P[f[N/x_0]/x_0]] = \mathcal{X};$$
$$f : N \rightarrow M \iff \forall P \in M^+, \forall N \in N, [P[f[N/x_0]/x_0]] = \mathcal{X}.\quad (6)\ (7)$$

As an immediate consequence, we have that $f : P \rightarrow Q \iff f : Q^\perp \rightarrow P^\perp$.

Composition of two typed functionals is naturally typed. Let us consider $f : P \rightarrow Q, g : Q \rightarrow R$ and $f' : N \rightarrow M, g' : M \rightarrow K$. By Proposition II we have the following:

**Proposition 3 (Composition).** $g \circ^* f : P \rightarrow R$ and $g' \circ^* f' : N \rightarrow K$.\quad 9
Let us now consider the “minimal” positive function space \( \mathbf{0} \rightarrow \mathbf{0} \). In the usual sense, there is only one (total) function which relates \( \mathbf{0} \) to itself, namely the one which sends \( \ddagger \) to \( \ddagger \).

On the other hand, the function space \( \mathbf{0} \rightarrow \mathbf{0} \) contains all the functionals:
\[
f^*(\ddagger) = \ddagger \text{ for any functional } f. \quad \text{All of them play the same role: they send } \ddagger \text{ to } \ddagger \quad \text{and hence they are equivalent from the point of view of } \mathbf{0} \rightarrow \mathbf{0}. \quad \text{For this reason, it seems to be quite natural to introduce an equivalence relation, depending on } \mathbf{X} \text{ and } \mathbf{Y}, \text{ in order to identify such functionals which are “equivalent” in } \mathbf{X} \rightarrow \mathbf{Y}. \quad \text{Similarly, elements of a behaviour } \mathbf{X} \text{ can be equipped with such a relation: } D \text{ and } E \text{ are equivalent in } \mathbf{X} \text{ whenever they only differ by useless parts i.e., subdesigns which do not play any active role when normalizing against designs of } \mathbf{X}^\perp. \quad \text{Interestingly, ludics is already equipped with such a relation which is called “equality up to materiality” [10], which we recall and adapt to functionals in the next section.}

5.2 Materiality, section, retraction and isomorphism

Roughly speaking, two functionals \( f, g \in \mathbf{X} \rightarrow \mathbf{Y} \), are “equal up to materiality” in \( \mathbf{X} \rightarrow \mathbf{Y} \) if they share that “minimal” part \( h \) which is really necessary during any computation with designs of \( \mathbf{X} \) and \( \mathbf{Y} \). For example, given \( f : \mathbf{0} \rightarrow \mathbf{0} \) observe that no part of \( f \) is necessary for computations, because \( [\ddagger[f/x_0]] \) immediately gives \( \ddagger \), whatever \( f \) is.

Before introducing materiality, we first define the concept of inclusion and intersection of two designs.

The inclusion of design is the largest binary relation \( \subseteq \) on standard designs such that:

1. if \( X \subseteq D \), then \( D = X \);
2. if \( \Omega \subseteq D \), then \( D \) is a positive design;
3. if \( x[\pi](N_1, \ldots, N_n) \subseteq D \), then \( D = x[\pi](M_1, \ldots, M_n) \) and \( N_i \subseteq M_i \), for any \( 1 \leq i \leq n \);
4. if \( \sum a(\bar{x}).P_a \subseteq D \) then \( D = \sum a(\bar{x}).Q_a \) and \( P_a \subseteq Q_a \) for every \( a \in A \).

The order \( \subseteq \) is also called the stable order [10, 5, 14]. Informally, \( D \subseteq E \) whenever \( D \) is obtained from \( E \) by replacing some positive subterms with \( \Omega \). An important property of the stable order is monotonicity: normal form function is monotone w.r.t. \( \subseteq \). More precisely, if \( P, N_1, \ldots, N_n \) and \( Q, M_1, \ldots, M_n \) are standard designs such that \( P \subseteq Q, N_1 \subseteq M_1, \ldots, N_n \subseteq M_n \) then \( [P[N_1/x_1, \ldots, N_n/x_n]] \subseteq [Q[M_1/x_1, \ldots, M_n/x_n]] \). A proof this property can be found in [14].

Given two standard designs \( D \) and \( E \), we define their intersection \( D \cap E \) by corecursion as follows:

1. \( P \cap \Omega = \Omega \cap P = \Omega \); \( \ddagger \cap \ddagger = \ddagger \);
2. \( x[\pi](N_1, \ldots, N_n) \cap x[\pi](M_1, \ldots, M_n) = x[\pi](N_1 \cap M_1, \ldots, N_n \cap M_n) \) if \( N_i \cap M_i \) are defined for every \( 1 \leq i \leq n \);
Let $A$ be a behaviour $X$ or a function space $X \to Y$. We define the material part of $D$ in $A$ as $|D|_A := \bigcap \{ E \subseteq D : E \in A \}$. Two designs $D, E$ are said equal up to materiality in $A$, noted by $D \sim_A E$, whenever $|D|_A = |E|_A$.

The definition of materiality is justified by the fact that $|D|_A$ is the smallest design of $A$ such that $|D|_A \subseteq D$ (see [10][11] for a proof of this facts). So, each equivalence class induced by $\sim_A$ has a canonical and unique representative in $A$.

For the function space $0 \to 0$, it is immediate to show that for any pair of functionals $f, g$, we have that $f \sim_0 g$, because the minimal functional in $0 \to 0$ is $0 := \sum a(x)O$, which has the property that $0 \subseteq N$ for any standard design $N$. The design $0$ can be thought as the empty negative one (it is called negative skunk in [10]).

**Lemma 2.** $D \sim_A E$ if and only if $\exists F \in A$ such that $F \subseteq D$ and $F \subseteq E$.

**Proof.** For the “if” part, observe that $F \subseteq D$ implies $|F|_A = |D|_A$ and $|F|_A = |E|_A$ i.e., $D \sim_A E$. For the “only if” part, take $F = |D|_A = |E|_A$. $\square$

**Theorem 2 (Preservation of $\sim$).**

1. If $P \sim_Q Q$ and $f : P \to Q$ then $f^*(P) \sim_Q f^*(Q)$;
2. If $P \in P$ and $f \sim_Q g$ then $f^*(P) \sim_Q g^*(P)$.

Similarly for negative behaviours and negative function spaces.

**Proof.** 1. By Lemma 2 if $P \sim_Q Q$ then $\exists R \in P$ such that $R \subseteq P$ and $R \subseteq Q$. Applying $f$, we have that $f^*(R) \subseteq P$ and $f^*(R) \subseteq f^*(Q)$ and by Lemma 2 again, we conclude $f^*(P) \sim_Q f^*(Q)$. For 2. and for negatives we use a similar reasoning. $\square$

We define an interactive notion of section, retraction and isomorphism.

**Definition 6 (Section, Retraction, Isomorphism).** Let $P, Q$ be positive behaviours and $r, s$ be functionals such that $r : P \to Q$ and $s : Q \to P$. When $r \circ s \sim_Q \sim_Q \sim_Q r \circ s$, we say that $s$ is a section of $r$ and $r$ is a retraction of $s$. If $s \circ r \sim_P \sim_P s \circ r$ id holds in addition, $s$ and $r$ are called isomorphisms. Similarly for negatives.

For a simple example of isomorphism, let us consider unary names $a, b$ and the negative design $0$ above. We claim that $P_a = \{ x_0 \bar{a}(0) \} \perp \perp$ and $P_b = \{ x_0 \bar{b}(0) \} \perp \perp$ are isomorphic. In fact, define $f = a(x).x_0 \bar{a}(0)$ and $g = b(x).x_0 \bar{b}(0)$ and we have that $f : P_a \to P_b$ and $g : P_b \to P_a$. Composing them, we have that $g \circ f = a(x).x_0 \bar{a}(0) \subseteq id$ and similarly $f \circ g = b(x).x_0 \bar{b}(0) \subseteq id$. By Lemma 2 we conclude $g \circ f \sim_{P_a} \sim_{P_a} id$ and $f \circ g \sim_{P_b} \sim_{P_b} id$.

In Section 7 we shall show that $f_{(a, \beta, i)}$ is a retraction of $u_{(a, \beta, i)}$, but in order to state it precisely, we have to clarify in which function-spaces these functionals live.
6 Logical behaviours

Let \( \alpha \) be an \( n \)-ary logical connective \((\mathcal{V}_\alpha, \{a_1(\bar{x}_1), \ldots, a_m(\bar{x}_m)\}) \in \mathcal{A} \) with \( \mathcal{V}_\alpha = x_1, \ldots, x_n \). Since each \( \bar{x}_i \) is a subsequence of \( \mathcal{V}_\alpha \), it is of the form \( x_{i_1}, \ldots, x_{i_k} \) with \( k = \text{ar}(a_i) \) and \( i_1, \ldots, i_k \in \{1, \ldots, n\} \). Given behaviours \( N_1, \ldots, N_n, P_1, \ldots, P_n \) we define:

\[ \begin{align*}
\overline{\alpha}(N_1, \ldots, N_n) &:= \left( \bigcup_{1 \leq i \leq m} \overline{a_i}(N_i, \ldots, N_{i_k}) \right)^{\perp}, \text{ where indices } i_1, \ldots, i_k \\
\alpha(P_1, \ldots, P_n) &:= \overline{\prod_{i=1}^{n} P_i}^{\perp}.
\end{align*} \]

A remarkable property of logical connectives is internal completeness [10]. It means that we can give a precise and direct description to the elements in logical behaviours without using the orthogonality and without referring to any proof system:

\[ \begin{align*}
\overline{\pi}(N_1, \ldots, N_n) &= \bigcup_{1 \leq i \leq m} \overline{a_i}(N_{i_1}, \ldots, N_{i_k}) \cup \{\mathcal{X}\}, \\
\alpha(P_1, \ldots, P_n) &= \{\sum_{i=1}^{n} a_i(\bar{x}_i) \mid P_i \models x_{i_1} : P_{i_1}, \ldots, x_{i_k} : P_{i_k} \text{ for every } 1 \leq i \leq m\},
\end{align*} \]

where the expression \( P_i \models x_{i_1} : P_{i_1}, \ldots, x_{i_k} : P_{i_k} \) is a short for \( \forall N_1 \in P_{i_1}, \ldots, \forall N_k \in P_{i_k}, [P_i[N_{i_1}/x_{i_1}, \ldots, N_k/x_{i_k}]] = \mathcal{X} \). Notice that additive components \( P_i \)’s can be arbitrary when \( b(\bar{y}) \notin \alpha \) (see [14] for the detail).

Recall that we have expressed the standard MALL connectives \( \top, \uparrow, \otimes, \& \) as logical connectives in Subsection 3.2 With these logical connectives we can build (semantic versions of) usual linear logic types. For sake of readability, we use the notation \( \downarrow = \top, \mathbf{1} = \pi, \otimes = \mathcal{X}, \mathbf{1} = \mathcal{X}, \text{ and } \boxplus = \mathcal{X}. \) By using the infix notation and taking into account the internal completeness, we have:

<table>
<thead>
<tr>
<th>( N \otimes M = \bullet(N, M) \cup {\mathcal{X}} )</th>
<th>( P \uparrow Q = {\varphi(x_1, x_2), P + \cdots : P \models x_1 : P, x_2 : Q} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( N \oplus M = \mathbf{1}(N) \cup \mathbf{2}(M) \cup {\mathcal{X}} )</td>
<td>( P &amp; Q = {\pi_1(x_0), P + \pi_2(x_0), Q + \cdots : P \in P, Q \in Q} )</td>
</tr>
<tr>
<td>( \downarrow N = \downarrow(N) \cup {\mathcal{X}} )</td>
<td>( \uparrow P = {\tau(x_0), P + \cdots : P \in P} )</td>
</tr>
<tr>
<td>( 0 = {\mathcal{X}} )</td>
<td>( \top = \mathcal{D}^{-} )</td>
</tr>
</tbody>
</table>

where irrelevant components of sums are surpressed by “…”.

As to the synthetic connective \( \gamma = \text{synth}(\otimes, \&), 1 \), we have:

\[ \begin{align*}
\overline{\pi}(N, M, K) &= \overline{\varphi}[\pi_1/1](N, K) \cup \overline{\varphi}[\pi_2/1](M, K) \cup \{\mathcal{X}\}, \\
\gamma(P, Q, R) &= \{\varphi[\pi_1/1](y_1, x_2), P + \varphi[\pi_2/1](y_2, x_2), Q + \cdots : P \models y_1 : P, x_2 : R \text{ and } Q \models y_2 : Q, x_2 : R\}.
\end{align*} \]

7 An analysis of focalization in ludics

In this section, we collect all we have done so far in order to show our main results.
7.1 Focalization in behaviours

Suppose that we are given logical connectives \( \alpha \in \Lambda(A)^n, \beta \in \Lambda(A)^m, \) a natural number \( 1 \leq i \leq n, \) and behaviours \( N_1, \ldots, N_{i-1}, M_1, \ldots, M_m, N_{i+1}, \ldots, N_n. \)

We can then form an “unsynthesized” behaviour \( P \) and a “synthesized” one \( Q \) by

\[
P = \gamma(N_1, \ldots, N_{i-1}, \beta(M), N_{i+1}, \ldots, N_n),
Q = \gamma(N_1, \ldots, N_{i-1}, M, N_{i+1}, \ldots, N_n),
\]

where \( \gamma = \text{synth}(\alpha, \beta, i). \) In this setting, we have the following:

**Proposition 4.** For \( P \) and \( Q \) as above, we have that

\[
f_{(\alpha, \beta, i)} : P \rightarrow Q, \quad u_{(\alpha, \beta, i)} : Q \rightarrow P.
\]

Hence, if \( P \sim P' \) then \( f_{(\alpha, \beta, i)}(P) \sim Q f_{(\alpha, \beta, i)}(P') \). The same for \( u_{(\alpha, \beta, i)} \).

**Proof.** We have to show that for any \( P \in \mathcal{P} \) and \( N \in \mathcal{Q} \), \( f_{(\alpha, \beta, i)}(P) \downarrow N \). The result follows by internal completeness. \( \square \)

**Theorem 3 (Section-retraction).** For \( P \) and \( Q \) as above, \( f_{(\alpha, \beta, i)} : P \rightarrow Q \) is a retraction of \( u_{(\alpha, \beta, i)} : Q \rightarrow P \).

**Proof.** By Proposition 4 we have that \( f = f_{(\alpha, \beta, i)} : P \rightarrow Q \) and \( u = u_{(\alpha, \beta, i)} : Q \rightarrow P \) and by Proposition 3 that \( f \circ u : Q \rightarrow Q \). By Proposition 4

\[
f \circ u = id, \text{ and since } id \subseteq id, f \circ u \sim id.
\]

We have thus arrived at a formal statement of our slogan: **focalization is a retraction of invertibility.**

It is also interesting to analyze how focalizing and inverting designs work in context. Given a sequence of positive behaviours \( P_1, \ldots, P_n \), it makes sense to consider them in “context” i.e., as components of a negative behaviour \( \alpha(P_1, \ldots, P_n) \).

Interestingly, focalization in context can be automatically obtained from the one-step focalization by the following general principle: **sections and retractions can be naturally lifted along arbitrary logical connectives.** Precisely:

**Theorem 4 (Lifting).**

Let \( \{s_i : Q_i \rightarrow P_i\}_{1 \leq i \leq n} \) be a family of sections and \( \{r_i : P_i \rightarrow Q_i\}_{1 \leq i \leq n} \) the family of corresponding retractions. For any logical connective \( \alpha \), there exists a section \( s : \alpha(Q_1, \ldots, Q_n) \rightarrow \alpha(P_1, \ldots, P_n) \) with a corresponding retraction \( r \).

**Proof.** Let \( z_1, \ldots, z_n \) be the directory of \( \alpha \) and let \( a(\bar{x}) \in \alpha \) with \( \bar{x} = x_1, \ldots, x_k \). Let us denote by \( s^\alpha \) the sequence of sections \( s_{x_1}[x_1/x_0], \ldots, s_{x_k}[x_k/x_0] \), where \( s_{x_j} = s_i \) for \( z_i = x_j \) and analogously for \( r^\alpha \). We define \( s \) and \( r \) as follows:

\[
s := \sum_{\alpha} a(\bar{x})x_0[\overline{s^\alpha}], \quad r := \sum_{\alpha} a(\bar{x})x_0[\overline{r^\alpha}].
\]
It is easy to show \( s : \alpha(Q_1, \ldots, Q_n) \rightarrow \alpha(P_1, \ldots, P_n) \) and \( r : \alpha(P_1, \ldots, P_n) \rightarrow \alpha(Q_1, \ldots, Q_n) \). Finally, we have to prove \( r \circ s \sim_{\alpha(Q_1, \ldots, Q_n)} \rightarrow_{\alpha(Q_1, \ldots, Q_n)} \text{id} \):

\[
(\sum_\alpha a(\vec{x}).(\sum_\alpha a(\vec{x}).x_0[\pi(s_n')])\{\pi(\vec{r}_n')\}) = \\
(\sum_\alpha a(\vec{x}).(\sum_\alpha a(\vec{x}).x_0[\pi(s_n')])\{\pi(\vec{r}_n')\}) = \\
(\sum_\alpha a(\vec{x}).x_0[\pi(id(\vec{x}))]) \subseteq \text{id},
\]

where \( r_n \circ s_n' \) denotes the componentwise composition. \( \square \)

### 7.2 A proof of focalization for MALL

We now sketch how to combine our treatment of focalization together with full completeness theorem for MALL \[10\] to derive a focalization theorem.

We consider the constant-only fragment of MALL, where formulas are generated by the following grammar:

\[
F ::= 0 | T | F \otimes F | F \oplus F | F \mathcal{H} F | F \& F \quad \text{(Rules are given in Appendix A.1)}.
\]

Inductively, we define the following interpretation function \( \cdot \) which sends a formula into a negative behaviour:

\[
0^* := \uparrow 0, \\
T^* := T, \\
(F \otimes F)^* := (F^* \otimes F^*), \\
(F \oplus F)^* := (F^* \oplus F^*), \\
(F \mathcal{H} F)^* := \downarrow F^* \mathcal{H} F^*, \\
(F \& F)^* := \downarrow F^* \& F^*.
\]

Let \( \pi \) be a cut-free proof of a formula \( F \) of MALL. By soundness theorem, we can get a design \( D \in F^* \) (or a class of designs which are equal up to materiality in \( F^* \)).

Now, we can repeatedly apply focalizing designs \( f_1, \ldots, f_n \) (in context) in order to get a design \( (f_n \circ \ldots \circ f_1)^*(D) \) in which positive layers are maximally synthesized. For negative layers, we can apply sequences of inverting designs. We finally get a design \( D^f \) in a behaviour \( F^*_s \) which is maximally synthesized. For an example of positive layer, the formula \( (F \oplus G) \& H \) is sent to \( \uparrow(\oplus(\mathcal{H}(\uparrow(F^*), \mathcal{H}(G^*), \mathcal{H}(H^*)))) \), from which we can find (the maximal) synthetic connective \( \mathcal{H}(F^*, G^*, H^*) \).

Now, we can apply the corresponding \( u_n, \ldots, u_1 \), where \( u_i \) are respectively the sections of \( f_i \). We get a new design \( D^{fu} \) in \( F^* \) built by usual connectives and focalized. It is clear that this procedure preserves \( \mathcal{H} \)-freeness (a characteristic of designs corresponding to proofs). Hence by full completeness, from \( D^{fu} \) we can get a proof \( \pi' \) of \( F \).

### 8 Conclusion and future works

We have considered in this paper how focalization can be considered from an interactive point of view, that is how the process of focalizing proofs could be achieved in Ludics, in an internal way. In order to do so, we considered particular designs which can be seen as interactive functions and which have the
ability to make a cluster of two positive logical connectives being separated by a single trivial \( \uparrow \) logical connective: that if they merge them in a single synthetic connective. Our approach allowed us to recover Focalization theorem for MALL thanks to Ludics full completeness results.

Our work naturally leads to several directions for extension and we wish to stress one of them which seems particularly promising. It is related with an analysis of usual computability and complexity theory by logical means. Indeed, our analysis of Focalization in Ludics was primarily motivated by an analogy with the proof of the Tape Compression theorem for Turing Machines. In [14], the third author showed a correspondence between Turing machines and some classes of designs (see appendix for more details). The words over \( \Sigma \) are encoded as data-designs: \( \epsilon^* = \uparrow \text{nil} \) \( (lw)^* = \uparrow \langle w^* \rangle \).

The focalization property on these data-designs allows to group (or synthesize) letters of those words in a larger alphabet, which is basically the way to obtain Tape Compression theorem. Then, the focalization process would correspond to the translation between the Turing machine working on alphabet \( \Sigma \) and Turing machines working on the “synthetic alphabet”.

References

A Appendix

A.1 Rules of constant-only MALL

For reader convenience, we report here the sequent calculus rules:

\[
\begin{align*}
\Gamma, F & \vdash \Gamma_1, F, \Gamma_2, \perp \\
\Gamma, & \vdash \Gamma_1, F, \Gamma_2 \\
\Gamma, F_1 & \vdash \Gamma, F_1 \oplus F_2 \\
\Gamma, F, G & \vdash \Gamma, F \otimes G \\
\Gamma, F, & \vdash \Gamma, F, G \\
\end{align*}
\]

A.2 Focalization and Tape Compression.

Data-designs are negative designs coinductively generated as:

\[
d := \uparrow \langle d_1, \ldots, d_n \rangle
\]

Words on an alphabet \( \Sigma \) are instances of Data-designs:

\[
e^* = \uparrow \text{nil}, \quad (lw)^* = \uparrow T(w^*)
\]

Two results of [14] relate Designs with standard computational objects:

**Theorem 5.** For every deterministic finite automaton \( \mathcal{M} \), there exists a finitely generated positive standard design \( \mathcal{D} \) such that for any \( w \in \Sigma^* \), \( \mathcal{M} \) accepts \( w \) iff \( \mathcal{D}[w^*/x] \Downarrow \mathcal{X} \) (and conversely).

**Theorem 6.** For every Turing machine \( \mathcal{M} \), there exists a finitely generated positive linear and identity-free design \( \mathcal{D} \) such that for any \( w \in \Sigma^* \), \( \mathcal{M} \) accepts \( w \) iff \( \mathcal{D}[w^*/x] \Downarrow \mathcal{X} \) (and conversely).

Thanks to theorem 6 it is natural to connect Focalization and Tape Compression theorem we recall below:

**Theorem 7 (Tape Compression).** Let \( c \) be any constant greater than 0. If \( \mathcal{L} \) is a language accepted by a deterministic Turing machine \( \mathcal{M} \) in space complexity \( S(n) \), then \( \mathcal{L} \) is accepted by a deterministic Turing machine \( \mathcal{M}^c \) with space complexity \( cS(n) \).

Indeed, the proof of the Tape Compression theorem relies on the fact of considering an alphabet made of large enough words. The operation of synthetizing letters in words precisely corresponds, thanks to the structure of data-designs, to the operation of Focalization.