Infinitary Completeness in Ludics

Michele Basaldella Kazushige Terui
RIMS, Kyoto University, Japan
{mbasalde,terui}@kurims.kyoto-u.ac.jp

***Abstract*—Traditional Gödel completeness holds between finite proofs and infinite models over formulas of finite depth, where proofs and models are heterogeneous. Our purpose is to provide an interactive form of completeness between infinite proofs and infinite models over formulas of infinite depth (that include recursive types), where proofs and models are homogenous.**

We work on a nonlinear extension of ludics, a monistic variant of game semantics which has the same expressive power as the propositional fragment of polarized linear logic. In order to extend the completeness theorem of the original ludics to the infinitary setting, we modify the notion of orthogonality by defining it via safety rather than termination of the interaction. Then the new completeness ensures that the universe of behaviours (interpretations of formulas) is Cauchy-complete, so that every recursive equation has a unique solution.

Our work arises from studies on recursive types in denotational and operational semantics, but is conceptually simpler, due to the purely logical setting of ludics, the completeness theorem, and use of coinductive techniques.

***Keywords*-linear logic; ludics; completeness; recursive types; coinduction.**

I. Introduction

A. Ludics and interactive completeness

The traditional logical completeness states that every proposition $\mathbf{A}$ has either a proof P or a countermodel M exclusively:

$$\exists P(P \vdash \mathbf{A}) \quad \mathsf{xor} \quad \exists M(M \not\models \mathbf{A}).$$

Here proofs and models are heterogeneous entities, so that there cannot be any interaction between them. It is quite contrary to the ordinary discussion, where Prover P and Refuter M do interact.

One aim of *ludics* [12], [8], [6], [21], a research program proposed by J.-Y. Girard, is to understand logical completeness in a more interactive way, closer to the ordinary discussion. In ludics, proofs and models are considered to be homogenous entities, called *designs*. The homogeneity allows us to obtain an *interactive completeness theorem*, stating that for every proposition $\mathbf{A}$ and every proof-attempt P, either P is a proof of $\mathbf{A}$ or there is a counter-argument M against $\mathbf{A}$ that defeats P:

$$P \vdash \mathbf{A} \quad \mathsf{xor} \quad \exists M(M \ \mathsf{against}\ \mathbf{A},\ M \ \mathsf{defeats}\ P). \qquad (1)$$

This work was supported by JSPS KAKENHI 2008803 and 21700041.

Since counter-arguments tend to be infinite, designs must be infinitary (coinductive) objects in general. Indeed, designs can be considered an abstract form of Böhm trees, which are possibly infinite (see [7] for a precise correspondence). Having a structure similar to Böhm trees, there is a sensible notion of normalization (cut-elimination), that defines the outcome of interaction: Player P defeats Refuter M ($P \perp M$) if the normalization of P applied to M succeeds, i.e., terminates without a type-error such as "`love`" + 1. In other words, M defeats P ($P \not\perp M$) if the normalization either leads to a "mismatch" (type-error), or gets into an "infinite chattering".

In ludics, types/formulas are identified with sets of designs closed under the biorthogonal operation $\mathbf{A} = \mathbf{A}^{\perp\perp}$, called *behaviours*, and "M against $\mathbf{A}$" simply means $M \in \mathbf{A}^{\perp}$. Thus, statement (1) amounts to the standard concept of *denotational completeness*: for every "logical" behaviour $\mathbf{A}$,

$$P \vdash \mathbf{A} \iff \neg\exists M \in \mathbf{A}^{\perp}(P \not\perp M) \iff P \in \mathbf{A}^{\perp\perp} = \mathbf{A}.$$

Although in ludics the denotation of a proof (the design corresponding to a proof) is very close to the proof itself, this does not mean that completeness is a trivial property. Notice that $P \vdash \mathbf{A}$ is defined by *induction on proofs* (syntax), $M \in \mathbf{A}^{\perp}$ is by *induction on types* (semantics), and the orthogonality relation $P \perp M$ is by *induction on reduction* (computation). The interactive completeness theorem establishes a harmony between these three fundamental inductions, that is far from being trivial.

Uses of orthogonality to define "semantic types" as closed sets of terms are abundant in the literature; see, e.g., [11] for proving strong normalization, [14] for realizability, [20] for parametricity as well as more recent works [17], [18].

Since the orthogonality of the original ludics involves termination of reduction, which in turn implies consistency (neither $\vdash \mathbf{A}$ nor $\vdash \mathbf{A}^{\perp}$ is provable), there is an inherent logical limitation on possible behaviours (semantic types): the Russell's antinomy argument implies that there does not exist a behaviour $\mathbf{A}$ such that $\mathbf{A} = {\downarrow}\mathbf{A}^{\perp}$ (where $\downarrow$ adjusts the polarity; see Proposition III.6). Namely, *some recursive types do not exist due to logical consistency*.

B. From Induction to Coinduction

So far is the situation described in our previous work [3]. The purpose of this article is to illustrate the interactive completeness in the infinitary setting, based on *coinduction*

rather than induction. Our starting point is an observation that the termination requirement in the above definition of orthogonality is not an indispensable one (as in [14]): one can declare as well that $P \perp M$ when the normalization is just *safe*, i.e., never leads to a type-error [22], [17]. Thus, "infinite chattering" between P and M counts as a win of P (as sometimes happens in the ordinary discussion; a persistent Prover wins just by continuing discussion until an impatient Refuter gives up).

Since this new notion of orthogonality as safety is well suited for coinduction, it motivates us to make all concepts infinitary and coinductive. Thus we are led to *coinductive proofs* (infinitary trees made of inference rules) and *coinductive logical behaviours* (infinitary trees made of logical connectives). Although consistency is lost, it turns out that the interactive completeness theorem still holds with this uniform change of induction into coinduction (Theorem V.7).

Furthermore, behaviours are not only inductively built from atomic ones as before, but also coinductively built from *Cauchy sequences of logical behaviours*. Since orthogonality-as-safety is no more associated to termination and consistency, there is no logical limitation on possible behaviours. Indeed, we prove the Cauchy completeness of the universe of logical behaviours (Theorem IV.9), so that all recursive equations have unique solutions in it (Theorem IV.10). *Recursive types do exist, as a compensation for loss of consistency*.

C. Outline

In Section II, we recall the term syntax of [21] adapted for the nonlinear extension of ludics (in which duplication of actions is allowed) introduced in [3]. We also relativize the notion of orthogonality. In Section III we recall the notions of behaviour and logical connective and give a coinductive definition of logical behaviours. Section IV introduces Cauchy sequences, and outlines how to prove their convergence. Section V introduces a coinductive proof system and proves the interactive completeness theorem. It is then used in Section VI to prove a key property for Cauchy completeness. Finally Section VII mentions some of the key results in the vast literature on recursive types and compares our work with them.

II. Designs and Orthogonality

We follow the term syntax approach of [21] and define *designs* — the main entities of ludics — *as terms*, employing a process calculus notation inspired by the close relationship between ludics and linear π-calculus [9]. Although our syntax sacrifices a fundamental feature of locativity, it admits a simple, coinductive definition. The nonlinear extension has virtually the same expressive power as the propositional, variable-free fragment of polarized linear logic [15], [6] (see [3] for the correspondence); this leaves us a possibility to apply our ideas to game semantics [13] in future.

We first recall (the identity-free fragment of) the nonlinear extension of ludics [3] and then we relativize the notion of orthogonality so that we can deal with not just the standard orthogonality as termination but also the new orthogonality as safety.

A. Designs

Designs are built over a given *signature* $\mathcal{A} = (A, \mathsf{ar})$, where A is a set of *names* $a, b, c, \ldots$ and $\mathsf{ar} : A \longrightarrow \mathbb{N}$ is a function which assigns to each name a its *arity* $\mathsf{ar}(a)$. Let $\mathcal{V}$ be a countable set of variables $\mathcal{V} = \{x, y, z, \ldots\}$.

Over a fixed signature $\mathcal{A}$, a *positive action* is $\overline{a}$ with $a \in A$, and a *negative action* is $a(x_1, \ldots, x_n)$ where variables $x_1, \ldots, x_n$ are distinct and $\mathsf{ar}(a) = n$. In the sequel, we assume that an expression of the form $a(\vec{x})$ always stands for a negative action.

Definition II.1 (Designs)**.** *For a fixed signature $\mathcal{A}$, the class of **positive designs** $P, Q, \ldots$, that of **predesigns** $S, T, \ldots$, and that of **negative designs** $N, M, \ldots$ are* coinductively *defined as follows:*

$$
\begin{array}{rcll}
P & ::= & \Omega & \textit{(syntactical error)} \\
 & | & \bigwedge\{S_i : i \in I\} & \textit{(nondeterministic conjunction)} \\
S & ::= & x|\overline{a}\langle N_1, \ldots, N_n\rangle & \textit{(head normal form)} \\
 & | & N|\overline{a}\langle N_1, \ldots, N_n\rangle & \textit{(cut)} \\
N & ::= & \sum a(\vec{x}).P_a & \textit{(abstraction)}
\end{array}
$$

where:

- $\mathsf{ar}(a) = n$*;*
- $\vec{x} = x_1, \ldots, x_n$ *and the formal sum* $\sum a(\vec{x}).P_a$ *is built from* $|A|$ *many components* $\{a(\vec{x}).P_a\}_{a\in A}$*;*
- $\bigwedge\{S_i : i \in I\}$ *is built from a set* $\{S_i : i \in I\}$ *of predesigns with I an arbitrary index set.*

Intuitively, designs may be considered as infinitary λ-terms with *named* applications, *superimposed* abstractions and a (universal) *nondeterministic* "choice" operator $\bigwedge$.

Specifically, a predesign $X|\overline{a}\langle N_1, \ldots, N_n\rangle$, where X is either a variable x or a negative design N, can be thought of as iterated application $XN_1 \cdots N_n$ of name $a \in A$.

For negative designs, one can think of $a(\vec{x}).P_a$ as iterated abstraction $\lambda\vec{x}.P_a$ of name $a \in A$. A family $\{a(\vec{x}).P_a\}_{a\in A}$ of abstractions indexed by A is then superimposed to form a negative design $\sum a(\vec{x}).P_a$.

Now the *cut* $(\sum a(\vec{x}).P_a)\,|\overline{b}\langle N_1, \ldots, N_n\rangle$ reduces as follows: since the application is of name b, the abstraction $b(\vec{x}).P_b$ of name b is chosen from the family $\{a(\vec{x}).P_a\}_{a\in A}$. Then the whole design reduces by "β-reduction" to $P_b[\vec{N}/\vec{x}]$ (see below).

So far the system is deterministic. However, our calculus is also equipped with the *conjunction* operator $\bigwedge$ that allows to express a certain sort of *universal nondeterminism* at the

level of the syntax. It is needed to maintain completeness in the nonlinear setting [3] (see also [2]), for the same reason as strategies in the Hyland-Ong games need to take into account *noninnocent* behaviour of Opponent [13] to get full completeness in the nonlinear case.

We write $\maltese$ (*daimon* [12]) for the empty conjunction $\bigwedge \emptyset$. A unary conjunction $\bigwedge\{S\}$ is simply written as S. This allows us to treat a predesign as a positive design. If $P = \bigwedge\{S_i : i \in I\}$, each S_i is called a *conjunct* of P.

Finally, the positive design Ω is a special design which denotes "syntactical error". It is also useful to encode partial sums: given a set $\alpha = \{a(\vec{x}), b(\vec{y}), \ldots\}$ of negative actions with distinct names $a, b, \ldots$, we write $\sum_\alpha a(\vec{x}).P_a$ to denote the negative design $\sum a(\vec{x}).R_a$, where $R_a = P_a$ if $a(\vec{x}) \in \alpha$, and $R_a = \Omega$ otherwise.

We denote arbitrary designs by $D, E, \ldots$.

A design D may contain free and bound variables. An occurrence of subterm $a(\vec{x}).P_a$ *binds* the free variables $\vec{x}$ in P_a. Variables which are not under the scope of the binder $a(\vec{x})$ are *free*. We denote by $\mathsf{fv}(D)$ the set of free variables occurring in D.

In analogy with λ-calculus, we always consider designs *up to α-equivalence*, i.e., up to renaming of bound variables (see [21], [3] for more detail).

The conjunction operator can be extended to positive and negative designs:

$$\begin{array}{rcl} \bigwedge\{S_i : i \in I\} \wedge \bigwedge\{S_j : j \in J\} & = & \bigwedge\{S_k : k \in I \cup J\}, \\ \Omega \wedge P & = & \Omega, \\ \sum a(\vec{x}).P_a \wedge \sum a(\vec{x}).Q_a & = & \sum a(\vec{x}).(P_a \wedge Q_a). \end{array}$$

In particular, we have $P \wedge \maltese = P$.

A design D is said:

- *closed*, if it does not contain any free variable;
- *cut-free*, if it does not contain a cut as subdesign.

B. Reduction and Orthogonality

In ludics, designs *interact* together via the following notion of reduction.

We use the standard notation $D[\vec{N}/\vec{x}]$ to denote the design obtained by the simultaneous and capture-free substitution of negative designs $\vec{N} = N_1, \ldots, N_n$ for free variables $\vec{x} = x_1, \ldots, x_n$ in D.

Definition II.2 (Reduction relation $\longrightarrow$)**.** *Given positive designs P, Q, we write $P \longrightarrow Q$ if P has a conjunct $\sum a(\vec{x}).P_a \mid \overline{b}\langle\vec{N}\rangle$ and $Q = P_b[\vec{N}/\vec{x}]$.*

Notice that Ω and $\maltese$ do not reduce to anything, and that $\longrightarrow$ is *nondeterministic* as the following example illustrates.

Example II.3. *Let us consider*

$$P = (a(x).\maltese) \,|\overline{a}\langle N\rangle \;\wedge\; (b(y).\Omega) \,|\overline{b}\langle M\rangle \;\wedge\; (c(z).Q) \,|\overline{c}\langle K\rangle.$$

We have that $P \longrightarrow \maltese$, $P \longrightarrow \Omega$, and $P \longrightarrow Q[K/z]$.

We now focus on positive and closed designs. Every such design P has one of the following forms: $\maltese$, Ω or $\bigwedge\{S_i : i \in I\}$, where each conjunct S_i is a cut. In the third case, $P \longrightarrow Q$ and Q is again a positive and closed design. Hence, any sequence of reductions from P eventually ends with $\maltese$, or Ω or *diverges*. While we always think of $\maltese$ as "win of Prover" and Ω as "win of Refuter" according to the intuition given in the introduction, there are various possibilities to interpret divergence. It is reflected in the definition of orthogonality below.

We write $r(P)$ for the set of immediate *reducts* of P, i.e., $r(P) = \{Q : P \longrightarrow Q\}$, and $\mathcal{D}_0$ for the set of positive and closed designs other than Ω.

Definition II.4 (Orthogonality)**.** *An **orthogonality** $\perp$ is a subset of $\mathcal{D}_0$ such that:*

($\mathcal{O}_1$) **(Reduction)** : $P \in \perp \Longrightarrow r(P) \subseteq \perp$*;*

($\mathcal{O}_2$) **(Expansion)** : $r(P) \subseteq \perp \Longrightarrow P \in \perp$.

The definition implies that $\maltese \in \perp$ and $\Omega \notin \perp$ as intended.

Any orthogonality $\perp$ is completely determined by its predesign members:

Proposition II.5. *$\bigwedge\{S_i : i \in I\} \in \perp$ if and only if $\{S_i : i \in I\} \subseteq \perp$. In particular, $\maltese \in \perp$.*

Proof: Suppose $\{S_i : i \in I\} \subseteq \perp$. By ($\mathcal{O}_1$) we have $\bigcup_{i\in I} r(S_i) \subseteq \perp$. Since $r\left(\bigwedge\{S_i : i \in I\}\right) = \bigcup_{i\in I} r(S_i)$, we conclude $\bigwedge\{S_i : i \in I\} \in \perp$ by ($\mathcal{O}_2$). The converse is similar. ∎

The above proposition also shows the universal (rather than existential) nature of our nondeterministic operator $\bigwedge$.

We end this section introducing some orthogonalities relevant for our work. The following two are canonical.

Termination: the *least* orthogonality is

$$\perp_L = \big\{P \in \mathcal{D}_0 : \text{every reduction sequence starting from } P \text{ terminates with } \maltese\big\}.$$

In fact, it is the least among those subsets of $\mathcal{D}_0$ satisfying ($\mathcal{O}_2$). It corresponds to the original orthogonality of ludics in the sense that "$P \in \perp_L$" corresponds, in the terminology of [12], to "$[\![\mathfrak{R}]\!] = \mathfrak{Dai}$" for a closed cut-net $\mathfrak{R}$.

Safety: the *greatest* orthogonality is

$$\perp_G = \big\{P \in \mathcal{D}_0 : \text{no reduction sequence starting from } P \text{ terminates with } \Omega\big\}.$$

In fact, it is the greatest among those subsets of $\mathcal{D}_0$ satisfying ($\mathcal{O}_1$). It correspond to *safety* (see e.g., [17], [23]).

Since $\perp_G$ is defined to be the greatest, it supports *proof by coinduction*: to prove that a design P belongs to $\perp_G$, it is sufficient to find a subset of $\mathcal{D}_0$ which contains P and satisfies ($\mathcal{O}_1$).

Example II.6. *Let P and Q be the positive and closed designs mutually defined by corecursion as follows:*

$$P := (a(x).Q)\,|\overline{a}\langle 0\rangle, \qquad Q := (b(y).P)\,|\overline{b}\langle 0\rangle,$$

where $0 := \sum a(\vec{x}).\Omega$. *To prove that* P *and* Q *are safe, consider the set* $\mathcal{X} = \{P, Q\}$. *Since* $P \longrightarrow Q$ *and* $Q \longrightarrow P$ *(and these are the only possible reductions), we have* $r(P) = \{Q\} \subseteq \mathcal{X}$ *and* $r(Q) = \{P\} \subseteq \mathcal{X}$. *Hence,* $\mathcal{X}$ *satisfies* $(\mathcal{O}_1)$ *and by coinduction, we conclude* $P, Q \in \bot_G$.

There is another family of orthogonalities that are interesting in view of the automata theoretic aspect of ludics.

Büchi orthogonality: given $\mathcal{B} \subseteq \mathcal{D}_0$, we define $\bot_{\mathcal{B}}$ as the subset of $\mathcal{D}_0$ such that any sequence starting from $\bot_{\mathcal{B}}$ either terminates with ✠ or visits a design $Q \in \mathcal{B}$ infinitely many times i.e.,

$$P = P_0 \longrightarrow P_1 \longrightarrow P_2 \longrightarrow \cdots$$

is such that $Q = P_n$ for infinitely many n. Notice that $\bot_\emptyset = \bot_L$.

Example II.7. *Let* P, Q *be as in Example II.6. If we take* $\mathcal{B} = \{P\}$, *we have that* $P, Q \in \bot_{\mathcal{B}}$. *Notice that* $P, Q \notin \bot_L$.

In [21] it is illustrated how words and deterministic finite automata (DFA) are represented by *deterministic* designs in ludics. Roughly, a word w is represented by a negative design $w^\star$ and a DFA A by a positive design $A^\star$. It is shown that A accepts a finite word w if and only if $A^\star[w^\star/x_0] \in \bot_L$. Thus the orthogonality $\bot_L$ expresses the usual acceptance condition of DFA: A accepts w if the run from the initial state *terminates* in an accepting state. The conjunction operator further allows us to express deterministic *tree* automata, as illustrated in [3]. With this parallelism in mind, Büchi orthogonality $\bot_{\mathcal{B}}$ is related to the Büchi acceptance condition for automata (either on words or on trees): any run starting from the initial state visits an accepting state infinitely many times. We postpone a detailed study of automata theoretic aspects of ludics to a subsequent work.

III. Logical Behaviours

In this section we briefly recall from [3] the notions of behaviour and logical connective. After that, we introduce the concept of (coinductively defined) *logical behaviour*, which is a novelty of this paper.

Definition III.1 (Behaviour)**.** *Let* $\bot$ *be an orthogonality.*

A positive design P *is* ***atomic*** *if it is* cut-free, $P \neq \Omega$ *and* $\mathsf{fv}(P) \subseteq \{x_0\}$ *for a certain* fixed *variable* x_0. *A negative design* N *is* ***atomic*** *if it is* cut-free *and* $\mathsf{fv}(N) = \emptyset$.

Two atomic designs P, N *of opposite polarities yield a design* $P[N/x_0] \in \mathcal{D}_0$. P *and* N *are said* ***orthogonal*** *w.r.t.* $\bot$, *(written* $P \bot N$*) if* $P[N/x_0] \in \bot$.

Given a set $\mathbf{X}$ *of* atomic *designs of the same polarity, we define define* $\mathbf{X}^\bot := \{E : \forall D \in \mathbf{X},\ D \bot E\}$.

A ***behaviour*** *is a set* $\mathbf{X}$ *of atomic designs of the same polarity such that* $\mathbf{X} = \mathbf{X}^{\bot\bot}$.

A behaviour is positive or negative according to the polarity of its designs. We use letters $\mathbf{P}, \mathbf{Q}, \ldots$ for positive behaviours, $\mathbf{N}, \mathbf{M}, \ldots$ for negative behaviours and $\mathbf{D}, \mathbf{E}, \ldots$ for arbitrary ones respectively.

We now describe how behaviours can be built by means of logical connectives in ludics. Let us assume that the set of variables $\mathcal{V}$ is equipped with a linear order $z_1, z_2, \ldots$.

Definition III.2 (Logical connectives)**.** *An* n*-ary* ***logical connective*** α *is a set* $\alpha = \{a_1(\vec{x}_1), \ldots, a_k(\vec{x}_k)\}$ *of negative actions such that the names* $a_1, \ldots, a_k$ *are pairwise distinct and* $\{\vec{x}_i\} \subseteq \{z_1, \ldots, z_n\}$ *for every* $1 \leq i \leq k$.

The intuition is as follows. The variables $z_1, \ldots, z_n$ stand for *placeholders* for subformulas $\mathbf{D}_1, \ldots, \mathbf{D}_n$ and the names $a_1, \ldots, a_k$ for the *logical inference rules* associated to α. For any $a(\vec{x}) \in \alpha$, the indices $i_1, \ldots, i_m$ given by $\vec{x} = z_{i_1}, \ldots, z_{i_m}$ indicate which subformulas $\mathbf{D}_{i_1}, \ldots, \mathbf{D}_{i_m}$ among $\mathbf{D}_1, \ldots, \mathbf{D}_n$ the rule a manipulates.

Formally, given an m-ary name a and negative behaviours $\mathbf{N}_1, \ldots, \mathbf{N}_m$, we define:

$$\overline{a}\langle \mathbf{N}_1, \ldots, \mathbf{N}_m\rangle := \big\{x_0 | \overline{a}\langle N_1, \ldots, N_m\rangle : N_1 \in \mathbf{N}_1, \ldots, N_m \in \mathbf{N}_m\big\}.$$

Given an n-ary logical connective α and behaviours $\mathbf{N}_1, \ldots, \mathbf{N}_n$, $\mathbf{P}_1, \ldots, \mathbf{P}_n$, we define:

$$\overline{\alpha}(\mathbf{N}_1, \ldots, \mathbf{N}_n) := \Big(\bigcup_{a(\vec{x}) \in \alpha} \overline{a}\langle \mathbf{N}_{i_1}, \ldots, \mathbf{N}_{i_m}\rangle\Big)^{\bot\bot},$$
$$\underline{\alpha}(\mathbf{P}_1, \ldots, \mathbf{P}_n) := \big(\overline{\alpha}(\mathbf{P}_1{}^\bot, \ldots, \mathbf{P}_n{}^\bot)\big)^\bot,$$

where the indices $i_1, \ldots, i_m \in \{1, \ldots, n\}$ are given by $\vec{x} = z_{i_1}, \ldots, z_{i_m}$, for each $a(\vec{x}) \in \alpha$.

We use the expression $\alpha(\mathbf{D}_1, \ldots, \mathbf{D}_n)$ as a neutral notation for behaviours generated by logical connectives: it stands for either a positive behaviour $\overline{\alpha}(\mathbf{D}_1, \ldots \mathbf{D}_n)$ or a negative behaviour $\underline{\alpha}(\mathbf{D}_1, \ldots \mathbf{D}_n)$. This presupposes that the behaviours $\mathbf{D}_1, \ldots \mathbf{D}_n$ are of the right polarity.

One can easily observe that logical connectives are *monotone* with respect to the inclusion order: if $\mathbf{D}_1 \subseteq \mathbf{E}_1$, …, $\mathbf{D}_n \subseteq \mathbf{E}_n$, then $\alpha(\mathbf{D}_1, \ldots, \mathbf{D}_n) \subseteq \alpha(\mathbf{E}_1, \ldots, \mathbf{E}_n)$.

Example III.3. *Usual connectives of linear logic can be defined if the signature* $\mathcal{A}$ *contains a* 0*-ary name* $*$, *unary names* $\uparrow, \pi_1, \pi_2$ *and a binary name* $\wp$.

$$\begin{array}{llll}
\alpha := \{\wp(z_1, z_2)\}, & \otimes := \overline{\alpha}, & \parr := \underline{\alpha}, & \bullet := \overline{\wp},\\
\beta := \{\pi_1(z_1), \pi_2(z_2)\}, & \oplus := \overline{\beta}, & \& := \underline{\beta}, & \iota_i := \overline{\pi_i},\\
\gamma := \{\uparrow(z_1)\}, & \downarrow := \overline{\gamma}, & \uparrow := \underline{\gamma} & \downarrow := \overline{\uparrow},\\
\delta := \{*\}, & \mathbf{1} := \overline{\delta}, & \bot\!\!\bot := \underline{\delta}, & \\
\emptyset, & \mathbf{0} := \overline{\emptyset}, & \top := \underline{\emptyset}. &
\end{array}$$

Notations $\bullet, \iota_i, \downarrow$ *are employed below. These logical connec-*

tives yield intended behaviours:

$$\begin{array}{rcl} \mathbf{N} \otimes \mathbf{M} & = & \bullet\langle \mathbf{N}, \mathbf{M}\rangle^{\perp\perp}, \\ \mathbf{P} \parr \mathbf{Q} & = & \bullet\langle \mathbf{P}^{\perp}, \mathbf{Q}^{\perp}\rangle^{\perp}, \\ \mathbf{N} \oplus \mathbf{M} & = & (\iota_1\langle \mathbf{N}\rangle \cup \iota_2\langle \mathbf{M}\rangle)^{\perp\perp}, \\ \mathbf{P} \,\&\, \mathbf{Q} & = & \iota_1\langle \mathbf{P}^{\perp}\rangle^{\perp} \cap \iota_2\langle \mathbf{Q}^{\perp}\rangle^{\perp}, \\ \downarrow \mathbf{N} = \downarrow\langle \mathbf{N}\rangle^{\perp\perp}, & & \uparrow \mathbf{P} = \downarrow\langle \mathbf{P}^{\perp}\rangle^{\perp}, \\ \mathbf{1} = \{x_0|\overline{*}\}^{\perp\perp}, & & \perp\!\!\!\perp = \mathbf{1}^{\perp}, \\ \mathbf{0} = \emptyset^{\perp\perp}, & & \top\!\!\!\top = \emptyset^{\perp}. \end{array}$$

Using logical connectives we can finally define logical behaviours:

Definition III.4 (Logical behaviour)**.** *Given an orthogonality $\perp$, we define $\mathbf{LB}(\perp)$ to be the greatest set $\mathcal{X}$ of behaviours with respect to $\perp$ such that:*

($\mathcal{L}$) if $\mathbf{D} \in \mathcal{X}$, then $\mathbf{D} = \alpha(\mathbf{D}_1, \ldots, \mathbf{D}_n)$ for some n-ary connective α and $\mathbf{D}_1, \ldots, \mathbf{D}_n \in \mathcal{X}$.

*We call $\mathbf{D} \in \mathbf{LB}(\perp)$ a **logical behaviour** with respect to $\perp$.*

The class $\mathbf{LB}(\perp)$ is well defined. However, as typical of coinductive definitions, it is not immediate to see what kind of behaviours reside in it. For instance, the definition does not tell us whether there is a logical behaviour such that $\mathbf{D} = \alpha(\mathbf{D}^{\perp})$. Our purpose now is to explore the universe $\mathbf{LB}(\perp)$ for a given $\perp$.

First of all, one can *inductively* build behaviours by:

$$\mathbf{P} ::= \overline{\alpha}(\mathbf{N}_1, \ldots, \mathbf{N}_n), \qquad \mathbf{N} ::= \underline{\alpha}(\mathbf{P}_1, \ldots, \mathbf{P}_n).$$

Proposition III.5. *For any choice of $\perp$:*

(1) *Inductively built logical behaviours belong to $\mathbf{LB}(\perp)$.*

(2) *If $\mathbf{D} \in \mathbf{LB}(\perp)$, then $\mathbf{D}^{\perp} \in \mathbf{LB}(\perp)$.*

Proof: As for (1), let $\mathcal{X}$ be the class of inductively defined logical behaviours. As for (2), let $\mathcal{X} = \{\mathbf{D}^{\perp} : \mathbf{D} \in \mathbf{LB}(\perp)\}$. In both cases, $\mathcal{X}$ satisfies the condition ($\mathcal{L}$). By coinduction, $\mathcal{X} \subseteq \mathbf{LB}(\perp)$. ∎

In particular, $\mathbf{LB}(\perp)$ has the *least positive* behaviour $\mathbf{0} = \emptyset^{\perp\perp} = \{\maltese\}$ and the *greatest negative* one $\top = \mathbf{0}^{\perp} = \{\text{atomic negative designs}\}$, for any signature $\mathcal{A}$ and any $\perp$.

We have all inductively built behaviours whatever the orthogonality is. On the other hand, there is a fundamental limitation on noninductive ones when the orthogonality is $\perp_L$. It is related to *Russell's antinomy*:

Proposition III.6. *There is no behaviour $\mathbf{P}$ with respect to $\perp_L$ such that $\mathbf{P} = \downarrow \mathbf{P}^{\perp}$.*

Proof: We first need to recall the followings facts.

(a) Any behaviour of the form $\uparrow \mathbf{Q}$ consists of negative designs $\sum \uparrow(x_0).Q_{\uparrow}$ where $Q_{\uparrow} \in \mathbf{Q}$ and the other additive components $a(\vec{x}).P_a$ can be arbitrary (see [3]).

(b) Let X be an atomic negative design or a variable. We call "infinitary η-expansion of X", and write $\eta(X)$, the design corecursively defined by $\eta(X) := \sum a(y_1, \ldots, y_n).X|\overline{a}\langle \eta(y_1), \ldots, \eta(y_n)\rangle$. We then have the following property. For P and N atomic: $P \perp_L N \implies P[\eta(N)/x_0] \in \perp_L$ (see [21]).

Assume now that there is $\mathbf{P}$ such that $\mathbf{P} = \downarrow \mathbf{P}^{\perp}$. Let $P := x_0|\downarrow\langle \eta(x_0)\rangle$. For any $N = \sum \uparrow(x_0).Q_{\uparrow} \in \mathbf{P}^{\perp}$, we have:

$$P[N/x_0] \;=\; N \mid \downarrow\langle \eta(N)\rangle \;\longrightarrow\; Q_{\uparrow}[\eta(N)/x_0].$$

Since $\mathbf{P}^{\perp} = \uparrow \mathbf{P}$, by (a) we have $Q_{\uparrow} \in \mathbf{P}$. In particular, $Q_{\uparrow} \perp_L N$. Using (b), we get $Q_{\uparrow}[\eta(N)/x_0] \in \perp_L$. From the fact that P is atomic and by ($\mathcal{O}_2$) we have $P \perp_L N$. In particular, $P \in (\uparrow \mathbf{P})^{\perp} = \mathbf{P}$. By using (a) again, we also have $M := \uparrow(x_0).P \in \uparrow \mathbf{P} = \mathbf{P}^{\perp}$. However:

$$\begin{array}{rcl} P[M/x_0] & = & M \mid \downarrow\langle \eta(M)\rangle \\ & \longrightarrow & \eta(M) \mid \downarrow\langle \eta(\eta(M))\rangle \\ & \longrightarrow & M \mid \downarrow\langle \eta(\eta(\eta(M)))\rangle \longrightarrow \cdots \end{array}$$

which clearly diverges. This shows a contradiction. ∎

Notice that there is no contradiction in the above argument when the orthogonality is $\perp_G$.

IV. Cauchy Sequences and Limits

Proposition III.5 ensures that in our setting we have all standard "types/formalas" of logical interest which are inductively defined, for any choice of orthogonality. We now show that if we choose the greatest one $\perp_G$, we obtain a richer class of types which include all the limits of Cauchy sequences. Thus we are given a powerful way to obtain a type by exhibiting a Cauchy sequence converging to it. In particular, all recursive equations have solutions in the universe of logical behaviours with respect to $\perp_G$.

In this section we deal with *sequences* $(\mathbf{D}_n)_{n\in\mathbb{N}} = \mathbf{D}_0, \mathbf{D}_1, \cdots$ *of logical behaviours* of the same polarity. We use letters $\mathsf{D}, \mathsf{E}, \ldots \mathsf{P}, \mathsf{Q}, \ldots \mathsf{N}, \mathsf{M} \ldots$ for sequences of logical behaviours.

Definition IV.1. *Let $\mathsf{D} = (\mathbf{D}_n)_{n\in\mathbb{N}}$, $\mathsf{E}_1 = (\mathbf{E}_{1,n})_{n\in\mathbb{N}}, \ldots$, $\mathsf{E}_m = (\mathbf{E}_{m,n})_{n\in\mathbb{N}}$ be sequences of logical behaviours and let α be a m-ary logical connective. We write $\mathsf{D} \rhd \alpha(\mathsf{E}_1, \ldots, \mathsf{E}_m)$ if:*

($\rhd$) there exists some $n_0 \in \mathbb{N}$ such that for every $n \in \mathbb{N}$, $\mathbf{D}_{n_0+n}$ is of the form $\alpha(\mathbf{E}_{1,n}, \ldots, \mathbf{E}_{m,n})$.

In words, $\mathsf{D} \rhd \alpha(\mathsf{E}_1, \ldots, \mathsf{E}_m)$ holds if $\mathbf{D}_n$ begins with α for all sufficiently large n ($n \geq n_0$) and $\mathsf{E}_1, \ldots, \mathsf{E}_m$ are sequences of "sub-behaviours" extracted from $(\mathbf{D}_{n_0+n})_{n\in\mathbb{N}}$ by removing the topmost connective α.

Definition IV.2 (Cauchy sequences)**.** *We define the set Cauchy of **Cauchy sequences** as the greatest set $\mathcal{X}$ of sequences of logical behaviours such that:*

($\mathcal{C}$) if $\mathsf{D} \in \mathcal{X}$ then $\mathsf{D} \rhd \alpha(\mathsf{E}_1, \ldots, \mathsf{E}_m)$ for some m-ary connective α and $\mathsf{E}_1, \ldots, \mathsf{E}_m \in \mathcal{X}$.

We also define $\mathsf{Cauchy}_{\Uparrow}$ (resp. $\mathsf{Cauchy}_{\Downarrow}$) to be the set of Cauchy sequences that are monotone increasing *(resp.* decreasing*) with respect to the inclusion order.*

Finally, let $\approx$ be the greatest binary relation $\mathcal{R}$ on Cauchy sequences such that:

($\approx$) $\mathsf{D}\ \mathcal{R}\ \mathsf{D}'$ *implies* $\mathsf{D} \rhd \alpha(\mathsf{E}_1, \ldots, \mathsf{E}_m)$, $\mathsf{D}' \rhd \alpha(\mathsf{E}'_1, \ldots, \mathsf{E}'_m)$ *and* $\mathsf{E}_1\ \mathcal{R}\ \mathsf{E}'_1$, …, $\mathsf{E}_m\ \mathcal{R}\ \mathsf{E}'_m$.

Given a sequence $\mathsf{D} = (\mathbf{D}_n)_{n\in\mathbb{N}}$ we denote by $\mathsf{D}^\perp$ its *dual sequence* $(\mathbf{D}_n^\perp)_{n\in\mathbb{N}}$. It is not hard to see that:

- if $\mathsf{D} \in \mathsf{Cauchy}$ (resp. $\mathsf{Cauchy}_\Downarrow$, resp. $\mathsf{Cauchy}_\Uparrow$) then $\mathsf{D}^\perp \in \mathsf{Cauchy}$ (resp. $\mathsf{Cauchy}_\Uparrow$, resp. $\mathsf{Cauchy}_\Downarrow$);
- if $\mathsf{D} \approx \mathsf{E}$, then $\mathsf{D}^\perp \approx \mathsf{E}^\perp$.

Example IV.3. *Let $\mathsf{P} = (\mathbf{P}_n)_{n\in\mathbb{N}}$ be the positive sequence given by $\mathbf{P}_0 := \mathbf{0}$ and $\mathbf{P}_{n+1} := \downarrow {\mathbf{P}_n}^\perp$. Let us also consider the dual sequence $\mathsf{P}^\perp$. They have the following shape:*

$$\begin{array}{rcccccc} \mathsf{P} & = & \mathbf{0}, & \downarrow\top, & \downarrow\uparrow\mathbf{0}, & \downarrow\uparrow\downarrow\top, & \downarrow\uparrow\downarrow\uparrow\mathbf{0}, \quad \cdots \\ \mathsf{P}^\perp & = & \top, & \uparrow\mathbf{0}, & \uparrow\downarrow\top, & \uparrow\downarrow\uparrow\mathbf{0}, & \uparrow\downarrow\uparrow\downarrow\top, \quad \cdots \end{array}$$

Since $\mathsf{P} \rhd \downarrow\mathsf{P}^\perp$ and $\mathsf{P}^\perp \rhd \uparrow\mathsf{P}$, the set $\{\mathsf{P}, \mathsf{P}^\perp\}$ satisfies the condition $(\mathcal{C})$ above. By coinduction, $\mathsf{P} \in \mathsf{Cauchy}$.

Remark IV.4. *As in* [16]*, it is possible to define a metric d on the set of logical behaviours. Define a binary relation $\sim_n$ by induction on n as follows:*

- $\mathbf{D} \sim_0 \mathbf{D}'$ *always holds;*
- $\mathbf{D} \sim_{n+1} \mathbf{D}'$ *if* $\mathbf{D} = \alpha(\mathbf{E}_1, \ldots, \mathbf{E}_m)$, $\mathbf{D}' = \alpha(\mathbf{E}'_1, \ldots, \mathbf{E}'_m)$ *and* $\mathbf{E}_1 \sim_n \mathbf{E}'_1$,… $\mathbf{E}_m \sim_n \mathbf{E}'_m$.

Given two logical behaviours $\mathbf{D}, \mathbf{E}$, let $d(\mathbf{D}, \mathbf{E}) = 2^{-k}$ where $k = sup\{n : \mathbf{D} \sim_n \mathbf{E}\}$ (with $2^{-\infty} = 0$ by convention). The above definition of Cauchy sequences precisely corresponds to the standard one in this metric space. Nevertheless, we prefer our coinductive approach since it provides a more direct way to reason about infinite sequences.

Any Cauchy sequence can be "approximated" by monotone increasing and decreasing ones as follows:

Lemma IV.5. *To every $\mathsf{D} \in \mathsf{Cauchy}$ we can associate $\mathsf{D}_\Uparrow \in \mathsf{Cauchy}_\Uparrow$ and $\mathsf{D}_\Downarrow \in \mathsf{Cauchy}_\Downarrow$ with the following properties:*

(1) $\mathsf{D}_\Uparrow \approx \mathsf{D} \approx \mathsf{D}_\Downarrow$.

(2) *For every $\mathsf{E} \in \mathsf{Cauchy}_\Uparrow$ such that $\mathsf{E} \approx \mathsf{D}$,* $(\bigcup \mathsf{D}_\Uparrow)^{\perp\perp} \subseteq (\bigcup \mathsf{E})^{\perp\perp} \subseteq \bigcap \mathsf{D}_\Downarrow$.

(3) *For every $\mathsf{F} \in \mathsf{Cauchy}_\Downarrow$ such that $\mathsf{F} \approx \mathsf{D}$,* $(\bigcup \mathsf{D}_\Uparrow)^{\perp\perp} \subseteq \bigcap \mathsf{F} \subseteq \bigcap \mathsf{D}_\Downarrow$.

Here the union and the intersection are taken over members of the sequences: given $\mathsf{D} = (\mathbf{D}_n)_{n\in\mathbb{N}}$, we write $\bigcup \mathsf{D}$ for $\bigcup_{n\in\mathbb{N}} \mathbf{D}_n$ and $\bigcap \mathsf{D}$ for $\bigcap_{n\in\mathbb{N}} \mathbf{D}_n$ respectively.

Proof: Given $\mathsf{D} = (\mathbf{D}_n)_{n\in\mathbb{N}}$ with $\mathsf{D} \rhd \alpha(\mathsf{E}_1, \ldots, \mathsf{E}_m)$, let $n_0 \in \mathbb{N}$ be the natural number given by the definition of $\rhd$. Consider then the sequence $\mathsf{D}_\Uparrow = (\mathbf{D}_n^\Uparrow)_{n\in\mathbb{N}}$ corecursively given by:

$$\begin{array}{rcl} \mathbf{D}_0^\Uparrow = \cdots = \mathbf{D}_{n_0-1}^\Uparrow & := & \mathbf{D}_\alpha, \\ \mathbf{D}_{n_0+n}^\Uparrow & := & \alpha(\mathbf{E}_{1,n}^\Uparrow, \ldots \mathbf{E}_{m,n}^\Uparrow), \end{array}$$

where $\mathbf{D}_\alpha = \mathbf{0}$ (resp. $\mathbf{D}_\alpha = \underline{\alpha}(\mathbf{0}, \ldots, \mathbf{0})$) if D is a positive (resp. negative) sequence.

It is then easy to check that $\mathsf{D}_\Uparrow \in \mathsf{Cauchy}_\Uparrow$. To show that $\mathsf{D} \approx \mathsf{D}_\Uparrow$, consider the binary relation on Cauchy sequences $\mathcal{R} = \{(\mathsf{D}, \mathsf{D}_\Uparrow) : \mathsf{D} \in \mathsf{Cauchy}\}$ and verify that $\mathcal{R}$ satisfies condition ($\approx$) above. We define $\mathsf{D}_\Downarrow = ((\mathsf{D}^\perp)_\Uparrow)^\perp$.

For the first inclusion of (2), observe that for every member $\mathbf{D}$ of $\mathsf{D}_\Uparrow$ there is a member $\mathbf{E}$ of E such that $\mathbf{D} \subseteq \mathbf{E}$. For the second inclusion, assume for simplicity that logical connectives used to build D are all unary and $\mathsf{E} = (\mathbf{E}_n)_{n\in\mathbb{N}}$. Then by definition every member $\mathbf{D}$ of $\mathsf{D}_\Downarrow$ is of the form $\alpha_1(\cdots\alpha_n(\top)\cdots)$ for some $n \in \mathbb{N}$. On the other hand, since $\mathsf{E} \approx \mathsf{D} \approx \mathsf{D}_\Downarrow$, one can find k_0 such that $\mathbf{E}_k$ is of the form $\alpha_1(\cdots\alpha_n(\mathbf{G}_k)\cdots)$ for every $k \geq k_0$. We therefore have $\mathbf{E}_k \subseteq \mathbf{D}$ for $k \geq k_0$ because $\mathbf{G}_k \subseteq \top$ and logical connectives are monotone. We also have $\mathbf{E}_k \subseteq \mathbf{D}$ for $k < k_0$ because $\mathbf{E}_k \subseteq \mathbf{E}_{k_0}$. This proves $\bigcup \mathsf{E} \subseteq \bigcap \mathsf{D}_\Downarrow$, so $(\bigcup \mathsf{E})^{\perp\perp} \subseteq \bigcap \mathsf{D}_\Downarrow$.

Claim (3) is dual to (2). ∎

Intuitively, the behaviour $(\bigcup \mathsf{D}_\Uparrow)^{\perp\perp}$ may be considered the "limit inferior" $\liminf \mathsf{D}$ of the sequence D, while $\bigcap \mathsf{D}_\Downarrow$ gives the "limit superior" $\limsup \mathsf{D}$. While we always have $\liminf \mathsf{D} \subseteq \limsup \mathsf{D}$, it is not ensured in general that they coincide. The following theorem ensures coincidence when the orthogonality is $\perp_G$.

Theorem IV.6 (Coincidence)**.** *Let the orthogonality be $\perp_G$ and $\mathsf{E} \in \mathsf{Cauchy}_\Uparrow$, $\mathsf{F} \in \mathsf{Cauchy}_\Downarrow$ such that $\mathsf{E} \approx \mathsf{F}$. Then:*

$$(\bigcup \mathsf{E})^{\perp\perp} = \bigcap \mathsf{F}.$$

This property is indeed the essence of various works on recursive types [19], [22], [17]. We will prove it as a consequence of the completeness theorem in Section VI. Hereafter, in the rest of the paper we assume that the orthogonality is $\perp_G$.

Definition IV.7 (Limit)**.** *Given $\mathsf{D} \in \mathsf{Cauchy}$, we define*

$$\lim \mathsf{D} = (\bigcup \mathsf{D}_\Uparrow)^{\perp\perp} = \bigcap \mathsf{D}_\Downarrow,$$

where $\mathsf{D}_\Uparrow$ and $\mathsf{D}_\Downarrow$ are given by Lemma IV.5.

By Lemma IV.5 (2), (3), we have $\lim \mathsf{D} = (\bigcup \mathsf{E})^{\perp\perp} = \bigcap \mathsf{F}$ for every $\mathsf{E} \in \mathsf{Cauchy}_\Uparrow$ and $\mathsf{F} \in \mathsf{Cauchy}_\Downarrow$.

The following proposition shows that negation and logical connectives are "continuous" operations.

Proposition IV.8. *Let D be a Cauchy sequence such that $\mathsf{D} \rhd \alpha(\mathsf{E}_1, \ldots, \mathsf{E}_m)$. We have:*

(1) $(\lim \mathsf{D})^\perp = \lim(\mathsf{D}^\perp)$.

(2) $\lim \mathsf{D} = \alpha(\lim \mathsf{E}_1, \ldots, \lim \mathsf{E}_m)$.

Proof: (1) $(\lim \mathsf{D})^\perp = (\bigcup \mathsf{D}_{\Uparrow})^\perp = \bigcap (\mathsf{D}_{\Uparrow})^\perp$. Since $(\mathsf{D}_{\Uparrow})^\perp \in \mathsf{Cauchy}_{\Downarrow}$ and $(\mathsf{D}_{\Uparrow})^\perp \approx \mathsf{D}^\perp$, we have $\lim \mathsf{D}^\perp = \bigcap (\mathsf{D}_{\Uparrow})^\perp = (\lim \mathsf{D})^\perp$.

(2) Assume for simplicity that α is unary. We then have $\mathsf{D}_{\Uparrow} \rhd \alpha(\mathsf{E}_{\Uparrow})$, $\mathsf{D}_{\Downarrow} \rhd \alpha(\mathsf{E}_{\Downarrow})$. Since α is monotone, we obtain

$$\lim \mathsf{D} = (\bigcup \mathsf{D}_{\Uparrow})^{\perp\perp} \subseteq \alpha\left((\bigcup \mathsf{E}_{\Uparrow})^{\perp\perp}\right) = \alpha(\lim \mathsf{E}),$$

$$\alpha(\lim \mathsf{E}) = \alpha\left(\bigcap \mathsf{E}_{\Downarrow}\right) \subseteq \bigcap \mathsf{D}_{\Downarrow} = \lim \mathsf{D}.$$

Hence, $\lim \mathsf{D} = \alpha(\lim \mathsf{E})$. ∎

Proposition IV.8 (2) immediately implies the Cauchy completeness of $\mathbf{LB}(\bot_G)$.

Theorem IV.9 (Cauchy completeness)**.** *For every Cauchy sequence* D*,* $\lim \mathsf{D}$ *is a logical behaviour with respect to* $\bot_G$*. Moreover, if* $\mathsf{D} \approx \mathsf{E}$*, then* $\lim \mathsf{D} = \lim \mathsf{E}$*.*

Proof: Let $\mathcal{X}$ be the set of all limits of Cauchy sequences. Then it satisfies $(\mathcal{L})$ of Definition III.4 thanks to Proposition IV.8 (2), so that $\mathcal{X} \subseteq \mathbf{LB}(\bot_G)$. ∎

As a consequence, all nontrivial recursive equations, including mixed variance ones, have unique solutions in $\mathbf{LB}(\bot_G)$.

Theorem IV.10 (Unique solution)**.** *Let* $\Phi(X, X^\perp)$ *be an expression which is inductively built from formal variables* $X, X^\perp$ *by means of logical connectives. We assume that* $\Phi(X, X^\perp)$ *has the same polarity as* X *and is not identical with* X *nor* $X^\perp$ *(i.e., it contains at least one logical connective). Then there exists a* unique *behaviour* $\mathbf{D}$ *in* $\mathbf{LB}(\bot_G)$ *such that* $\mathbf{D} = \Phi(\mathbf{D}, \mathbf{D}^\perp)$*.*

Proof: Define a Cauchy sequence $\mathsf{D} = (\mathbf{D}_n)_{n\in\mathbb{N}}$ by setting $\mathbf{D}_0 = \mathbf{0}$ (resp. $\mathbf{D}_0 = \top$) if $\Phi(X, X^\perp)$ is a positive (resp. negative) expression and $\mathbf{D}_{n+1} = \Phi(\mathbf{D}_n, \mathbf{D}_n^\perp)$. Then we have $\mathsf{D} \rhd^* \Phi(\mathsf{D}, \mathsf{D}^\perp)$, where $\rhd^*$ is an obvious extension of $\rhd$ to clusters Φ of logical connectives. By applying Proposition IV.8 repeatedly, we eventually obtain $\lim \mathsf{D} = \Phi(\lim \mathsf{D}, \lim \mathsf{D}^\perp) = \Phi(\lim \mathsf{D}, (\lim \mathsf{D})^\perp)$.

To show uniqueness, suppose that a logical behaviour $\mathbf{E}$ satisfies $\mathbf{E} = \Phi(\mathbf{E}, \mathbf{E}^\perp)$. Then we can show $\mathsf{D} \approx \mathsf{E}$, where E is the constant sequence $\mathbf{E}, \mathbf{E}, \cdots$. Hence by Theorem IV.9 we conclude $\lim \mathsf{D} = \lim \mathsf{E} = \mathbf{E}$. ∎

V. Proof System and Full Completeness

In this section we introduce an infinitary (coinductive) proof system and show interactive completeness.

A. Proof System

The logical system we introduce is an infinitary extension of the one given in [3]. The main novelty, w.r.t. the finitary version, is the notion of *derivability of sequents* which is here taken in its *coinductive* interpretation.

A *positive context* $\mathbf{\Gamma}$ is of the form $x_1 : \mathbf{P}_1, \ldots, x_n : \mathbf{P}_n$, where $x_1, \ldots, x_n$ are distinct variables and $\mathbf{P}_1, \ldots, \mathbf{P}_n$ are positive logical behaviours. We denote by $\mathsf{fv}(\mathbf{\Gamma})$ the set $\{x_1, \ldots, x_n\}$. A positive *sequent* is a pair of the form $P \vdash \mathbf{\Gamma}$ where P is a positive design with $\mathsf{fv}(P) \subseteq \mathsf{fv}(\mathbf{\Gamma})$.

A *negative context* $\mathbf{\Gamma}, \mathbf{N}$ is a positive context $\mathbf{\Gamma}$ enriched with a negative logical behaviour $\mathbf{N}$, to which no variable is associated. A negative *sequent* is a pair of the form $N \vdash \mathbf{\Gamma}, \mathbf{N}$ where N is a negative design with $\mathsf{fv}(N) \subseteq \mathsf{fv}(\mathbf{\Gamma})$.

We write $D \vdash \mathbf{\Lambda}$ for a generic sequent.

We let $\mathcal{S}, \mathcal{T}, \ldots$ range over sequents and $\mathbb{X}, \mathbb{Y}, \ldots$ over sets of sequents. We denote by $\mathbb{S}$ the set of all sequents.

Our proof system has three sorts of inference rules:

- A *positive* rule:

$$\frac{N_1 \vdash \mathbf{\Gamma}, z : \mathbf{P}, \mathbf{N}_{i_1} \quad \ldots \quad N_m \vdash \mathbf{\Gamma}, z : \mathbf{P}, \mathbf{N}_{i_m}}{z|\overline{a}\langle N_1, \ldots, N_m\rangle \vdash \mathbf{\Gamma}, z : \mathbf{P}} \ (\overline{\alpha}, \overline{a})$$

  where $\mathbf{P} = \overline{\alpha}(\mathbf{N}_1, \ldots, \mathbf{N}_n)$, $a(\vec{x}) \in \alpha$ and the behaviours $\mathbf{N}_{i_1}, \ldots, \mathbf{N}_{i_m} \in \{\mathbf{N}_1, \ldots, \mathbf{N}_n\}$ are determined by $\vec{x} = z_{i_1}, \ldots, z_{i_m}$ (cf. Definition III.2).

- A *negative* rule:

$$\frac{\{P_a \vdash \mathbf{\Gamma}, \vec{x} : \vec{\mathbf{P}}_a\}_{a(\vec{x})\in\alpha}}{\sum a(\vec{x}).P_a \vdash \mathbf{\Gamma}, \mathbf{N}} \ (\underline{\alpha})$$

  where $\mathbf{N} = \underline{\alpha}(\mathbf{P}_1, \ldots, \mathbf{P}_n)$ and $\vec{x} : \vec{\mathbf{P}}_a$ stands for $z_{i_1} : \mathbf{P}_{i_1}, \ldots, z_{i_m} : \mathbf{P}_{i_m}$. The behaviours $\mathbf{P}_{i_1}, \ldots, \mathbf{P}_{i_m} \in \{\mathbf{P}_1, \ldots, \mathbf{P}_n\}$ are determined by $\vec{x} = z_{i_1}, \ldots, z_{i_m}$. We assume that $\mathsf{fv}(\mathbf{\Gamma})$ and $\vec{x}$ consist of distinct variables. If $b(\vec{y}) \notin \alpha$, the subdesign P_b of $\sum a(\vec{x}).P_a$ can be arbitrary.

- A *conjunction rule*:

$$\frac{\{S_i \vdash \mathbf{\Gamma}\}_{i\in I}}{\bigwedge\{S_i : i \in I\} \vdash \mathbf{\Gamma}} \ (conj)$$

  where I is an arbitrary index set and each S_i is a predesign.

Example V.1. *For connectives* $\otimes$ *and* $\parr$ *(as given in Example III.3), our inference rules specialize as follows:*

$$\frac{N \vdash \mathbf{\Gamma}, z : \mathbf{N} \otimes \mathbf{M}, \mathbf{N} \quad M \vdash \mathbf{\Gamma}, z : \mathbf{N} \otimes \mathbf{M}, \mathbf{M}}{z|\overline{a}\langle N, M\rangle \vdash \mathbf{\Gamma}, z : \mathbf{N} \otimes \mathbf{M}} \ (\otimes, \bullet)$$

$$\frac{P \vdash \mathbf{\Gamma}, x_1 : \mathbf{P}, x_2 : \mathbf{Q}}{\wp(x_1, x_2).P + \cdots \vdash \mathbf{\Gamma}, \mathbf{P} \parr \mathbf{Q}} \ (\parr),$$

where the irrelevant components of the sum are suppressed by "$+\cdots$*."*

We now define the notion of derivability as a fixpoint of an operator defined on sets of sequents. Let $Dv : \mathcal{P}(\mathbb{S}) \longrightarrow \mathcal{P}(\mathbb{S})$ (*derivability*) be the operator defined as follows:

$$Dv(\mathbb{X}) := \{\mathcal{S} : \text{the sequent } \mathcal{S} \text{ can be inferred from some sequents in } \mathbb{X} \text{ using an instance of an inference rule } \}.$$

The operator Dv is obviously monotone and hence it admits the least and the greatest fixpoints:

$$\mathbb{S}_L = \bigcap\{\mathbb{X} \subseteq \mathbb{S} : Dv(\mathbb{X}) \subseteq \mathbb{X}\},$$
$$\mathbb{S}_G = \bigcup\{\mathbb{X} \subseteq \mathbb{S} : \mathbb{X} \subseteq Dv(\mathbb{X})\}.$$

In this paper, we adopt the *coinductive interpretation* of derivability:

Definition V.2 (Derivability)**.** *A sequent $D \vdash \mathbf{\Lambda}$ is said **derivable** if $D \vdash \mathbf{\Lambda} \in \mathbb{S}_G$.*

Namely, a sequent is derivable if it admits an infinitary derivation tree. Since $\mathbb{S}_G$ is the greatest fixpoint, it supports *proof by coinduction*: to prove that a sequent $\mathcal{S}$ is derivable, it suffices to find $\mathbb{X} \subseteq \mathbb{S}$ such that $\mathcal{S} \in \mathbb{X}$ and $\mathbb{X} \subseteq Dv(\mathbb{X})$.

We have *structural rules* as derived rules:

Proposition V.3 (Structural rules)**.**

(1) Weakening *: If $D \vdash \mathbf{\Lambda} \in \mathbb{S}_G$, then for any positive context $\mathbf{\Gamma}$ we have $D \vdash \mathbf{\Lambda}, \mathbf{\Gamma} \in \mathbb{S}_G$;*

(2) Contraction *: If $D \vdash \mathbf{\Lambda}, x : \mathbf{P}, y : \mathbf{P} \in \mathbb{S}_G$, then we have $D[z/x, z/y] \vdash \mathbf{\Lambda}, z : \mathbf{P} \in \mathbb{S}_G$*

(provided the variables in $\mathsf{fv}(\mathbf{\Gamma})$ and z are fresh).

Notice that in our proof system sequents like $P \vdash z : \mathbf{P}$ and $N \vdash \mathbf{P}^\perp$ can be both derivable. We would like to relate them, via a suitable *cut rule*.

Definition V.4 (Cut-derivability)**.** *We say that a sequent $\mathcal{S}$ is **cut-derivable** if there is a finite set $\mathbb{X} \subseteq \mathbb{S}_G$ from which we can derive $\mathcal{S}$ using finitely many times the following cut-rule:*

$$\frac{P \vdash \mathbf{\Gamma}, z : \mathbf{P} \quad N \vdash \mathbf{\Gamma}, \mathbf{P}^\perp}{P[N/z] \vdash \mathbf{\Gamma}}\ (cut)$$

By using cut, one can derive a "contradiction" $P \vdash$. A natural question would be whether such a contradictory proof system would give us any useful information. The following theorem gives an answer.

Theorem V.5 (Safety)**.** *If $P \vdash$ is cut-derivable, then $P \in \perp_G$. Hence P never leads to Ω.*

Proof sketch: Define $\mathcal{X} = \{P : P \vdash \text{ is cut-derivable}\}$ and show that $\mathcal{X}$ satisfies $(\mathcal{O}_1)$ i.e., if $P \in \mathcal{X}$ and $P \longrightarrow P'$ then $P' \in \mathcal{X}$. It is just a special case of the *subject reduction property*. ∎

B. Interactive Completeness

Let us now switch to the "semantical" side. Corresponding to derivability of sequents in the "syntactical" side, we have a relation $\models$ of *semantical entailment*. It is a natural extension of the notion of behaviour to contexts of behaviours. In particular, it is defined via *orthogonality*.

Given a positive context $\mathbf{\Gamma} = x_1 : \mathbf{P}_1, \ldots, x_n : \mathbf{P}_n$ and a negative context $\mathbf{\Gamma}, \mathbf{N}$ we define the relation $\models$ as follows:

$$\begin{array}{rrl} P \models \mathbf{\Gamma} & \text{iff} & P \text{ cut-free }, \mathsf{fv}(P) \subseteq \mathsf{fv}(\mathbf{\Gamma}) \\ & \text{and} & P[N_1/x_1, \ldots, N_n/x_n] \in \perp \\ & \text{for any} & N_1 \in \mathbf{P}_1^\perp, \ldots, N_n \in \mathbf{P}_n^\perp; \end{array}$$

$$\begin{array}{rrl} N \models \mathbf{\Gamma}, \mathbf{N} & \text{iff} & N \text{ cut-free }, \mathsf{fv}(N) \subseteq \mathsf{fv}(\mathbf{\Gamma}) \\ & \text{and} & P[N[N_1/x_1, \ldots, N_n/x_n]/x_0] \in \perp \\ & \text{for any} & N_1 \in \mathbf{P}_1^\perp, \ldots, N_n \in \mathbf{P}_n^\perp, P \in \mathbf{N}^\perp. \end{array}$$

By the definition, it immediately follows that $P \models x_0 : \mathbf{P}$ if and only if $P \in \mathbf{P}$ and $N \models \mathbf{N}$ if and only if $N \in \mathbf{N}$.

As for sequents, we use the generic notation $D \models \mathbf{\Lambda}$.

The semantical entailment enjoys the following properties. With the same notation of the inference rules above:

Proposition V.6.

(1) $z|\overline{a}\langle N_1, \ldots, N_m\rangle \models \mathbf{\Gamma}, z : \mathbf{P}$ *if and only if* $a(\vec{x}) \in \alpha$ *and* $N_1 \models \mathbf{\Gamma}, \mathbf{N}_{i_1}, z : \mathbf{P}, \ldots, N_m \models \mathbf{\Gamma}, \mathbf{N}_{i_m}, z : \mathbf{P}$.

(2) $\sum a(\vec{x}).P_a \models \mathbf{\Gamma}, \mathbf{N}$ *if and only if for every* $a(\vec{x}) \in \alpha$, $P_a \models \mathbf{\Gamma}, \vec{x} : \vec{\mathbf{P}}_a$.

(3) *For any index set I, $\bigwedge\{S_i : i \in I\} \models \mathbf{\Gamma}$ if and only if $S_i \models \mathbf{\Gamma}$ for every $i \in I$. In particular, $\maltese \models \mathbf{\Gamma}$.*

Proof: The "left to right" implications of (1) and (2) follow by $(\mathcal{O}_1)$ and further properties of logical connectives studied in [3]. The converse implications follow by $(\mathcal{O}_2)$.

(3) is a consequence of Proposition II.5. ∎

We are now ready to prove the interactive completeness theorem. In [3], a completeness theorem for finite derivability $\mathbb{S}_L$, least orthogonality $\perp_L$ and *inductively* defined logical behaviours has been proved. The statement below is its infinitary counterpart.

Theorem V.7 (Interactive completeness)**.** *Let $\mathbf{\Lambda}$ be a context of logical behaviours with respect to $\perp_G$ and D a cut-free design. We have:*

$$D \models \mathbf{\Lambda} \iff D \vdash \mathbf{\Lambda} \in \mathbb{S}_G.$$

It follows from the two lemmas below.

Lemma V.8 (Completeness)**.** *If $D \models \mathbf{\Lambda}$, then $D \vdash \mathbf{\Lambda} \in \mathbb{S}_G$.*

Proof: Define $\mathbb{X} = \{D \vdash \mathbf{\Lambda} : D \models \mathbf{\Lambda}\}$ and verify $\mathbb{X} \subseteq Dv(\mathbb{X})$ using Proposition V.6 (1) if D is a predesign, (2) if D is negative, (3) if D is a conjunction respectively. ∎

Lemma V.9 (Soundness)**.** *If $D \vdash \mathbf{\Lambda} \in \mathbb{S}_G$, then $D \models \mathbf{\Lambda}$.*

Proof: Consider a positive sequent $P \vdash x : \mathbf{P}, y : \mathbf{Q} \in \mathbb{S}_G$. We have to show that for any $N \in \mathbf{P}^\perp$ and $M \in \mathbf{Q}^\perp$, $P[N/x, M/y] \in \perp_G$. By Lemma V.8, $N \vdash \mathbf{P}^\perp \in \mathbb{S}_G$ and $M \vdash \mathbf{Q}^\perp \in \mathbb{S}_G$. Using these sequents we can easily find (thanks to Proposition V.3 (1)) a cut-derivation of $P[N/x, M/y] \vdash$. By Safety (Theorem V.5) we conclude $P[N/x, M/y] \in \perp_G$. The negative case is similar. ∎

VI. Coincidence

Interestingly, the infinitary completeness theorem can be used to derive Coincidence (Theorem IV.6), which is left in Section IV. Thus denotational completeness helps interpret recursive types. For this purpose, we consider contexts and sequents of *Cauchy sequences* (of logical behaviours).

In analogy to the previous section, a positive contexts Γ is of the form $x_1 : \mathsf{P}_1, \dots, x_m : \mathsf{P}_m$, where each $\mathsf{P}_i = (\mathbf{P}_{i,n})_{n\in\mathbb{N}}$ is a Cauchy sequence of positive logical behaviours. A negative context is of the form Γ, N where Γ is a positive context and $\mathsf{N} = (\mathbf{N}_n)_{n\in\mathbb{N}}$ is a Cauchy sequence of negative logical behaviours. We write Λ for a generic context of Cauchy sequences. A *sequent of Cauchy sequences* is a pair of the form $D \vdash \Lambda$ where D is a design and Λ a context of Cauchy sequences (with the same conditions on D we gave for sequents of logical behaviours).

We say that a sequent $D \vdash \Lambda$ is *derivable* if $D \vdash \mathbf{\Lambda}_n \in \mathbb{S}_G$ for every $n \in \mathbb{N}$, where:

- $\mathbf{\Lambda}_n = x_1 : \mathbf{P}_{1,n}, \dots, x_m : \mathbf{P}_{m,n}$, if $\Lambda = \Gamma$;
- $\mathbf{\Lambda}_n = x_1 : \mathbf{P}_{1,n}, \dots, x_m : \mathbf{P}_{m,n}, \mathbf{N}_n$, if $\Lambda = \Gamma, \mathsf{N}$.

While the above notion of derivability is defined elementwise, we may consider global inference rules that directly manipulate sequents of Cauchy sequences.

$$\frac{N_1 \vdash \Gamma', \mathsf{N}_{i_1}, z : \mathsf{P}' \quad \dots \quad N_m \vdash \Gamma', \mathsf{N}_{i_m}, z : \mathsf{P}'}{z|\overline{a}\langle N_1, \dots, N_m\rangle \vdash \Gamma, z : \mathsf{P}} \ (\overline{\alpha}, \overline{a})^C$$

where $\mathsf{P} \approx \overline{\alpha}(\mathsf{N}_1, \dots, \mathsf{N}_n) \approx \mathsf{P}'$ and $\Gamma \approx \Gamma'$ (the binary relation $\approx$ is naturally extended to contexts);

$$\frac{\{P_a \vdash \Gamma', \vec{x} : \vec{\mathsf{P}}_a\}_{a(\vec{x})\in\alpha}}{\sum a(\vec{x}).P_a \vdash \Gamma, \mathsf{N}} \ (\underline{\alpha})^C$$

where $\mathsf{N} \approx \underline{\alpha}(\mathsf{P}_1, \dots, \mathsf{P}_n)$ and $\Gamma \approx \Gamma'$;

$$\frac{\{S_i \vdash \Gamma'\}_{i\in I}}{\bigwedge\{S_i : i \in I\} \vdash \Gamma} \ (conj)^C$$

where $\Gamma \approx \Gamma'$.

Now we have the following "generation lemma" as the crucial step; although the derivability of $D \vdash \Lambda$ is defined elementwise, i.e., from the derivability of $D \vdash \mathbf{\Lambda}_n$ for each $n \in \mathbb{N}$, a "common last rule" can be found for all sufficiently large n.

Lemma VI.1. *If $D \vdash \Lambda$ is derivable, then it is a conclusion of an instance of the rules $(\overline{\alpha}, \overline{a})^C$, $(\underline{\alpha})^C$, $(conj)^C$ above.*

In analogy with the previous section, we say that a sequent of Cauchy sequences is **cut-derivable** if there is a finite set of derivable sequents (of Cauchy sequences) from which we can derive it using finitely many times the following cut-rule:

$$\frac{P \vdash \Gamma', z : \mathsf{P} \quad N \vdash \Gamma', \mathsf{N}}{P[N/z] \vdash \Gamma} \ (cut)^C$$

where $\mathsf{P}^\perp \approx \mathsf{N}$ and $\Gamma \approx \Gamma'$.

Thanks to the previous "generation lemma," we can naturally extend the safety theorem to Cauchy sequences.

Theorem VI.2 (Cauchy safety)**.** *If $P \vdash$ is cut-derivable in the proof system of Cauchy sequences, then $P \in \bot_G$.*

We can finally give a proof of Theorem IV.6.

Thanks to Lemma IV.5 (2), (3), it is sufficient to show that $\bigcap \mathsf{D}_\Downarrow \subseteq (\bigcup \mathsf{D}_\Uparrow)^{\perp\perp}$ for $\mathsf{D} \in \mathsf{Cauchy}$. Suppose that D is positive, $\mathsf{D}_\Uparrow = (\mathbf{P}_m)_{m\in\mathbb{N}}$ and $\mathsf{D}_\Downarrow = (\mathbf{Q}_n)_{n\in\mathbb{N}}$. Let $Q \in \bigcap \mathsf{D}_\Downarrow$ and $N \in (\bigcup \mathsf{D}_\Uparrow)^\perp = \bigcap (\mathsf{D}_\Uparrow)^\perp$. By interactive completeness (Theorem V.7), we have $Q \vdash x_0 : \mathbf{Q}_n$ and $N \vdash \mathbf{P}_m^\perp$ for every n, m, so $Q \vdash x_0 : \mathsf{D}_\Downarrow$ and $N \vdash (\mathsf{D}_\Uparrow)^\perp$. Since $\mathsf{D}_\Downarrow \approx ((\mathsf{D}_\Uparrow)^\perp)^\perp$, we may apply Theorem VI.2 to obtain $Q[N/x_0] \in \bot_G$. This proves $\bigcap \mathsf{D}_\Downarrow \subseteq (\bigcup \mathsf{D}_\Uparrow)^{\perp\perp}$.

VII. Related Work and Conclusion

Our work is built upon vast amount of work on recursive types. Due to lack of space, we cannot do an exhaustive survey here. Below, we just mention three influential works, referring to [22], [17] for a more global perspective on the literature.

1. In domain theory, domains for recursive types are obtained by simply solving domain equations. To reason about properties of programs, however, one further wishes to interpret types more strictly as subsets of (or relations on) suitable domains. It is this line of research that our work is related to (since the universal "domain" of designs is already given).

In [16], MacQueen, Plotkin and Sethi consider a universal domain $\mathbf{V}$ that arises as the canonical solution of the equation

$$\mathbf{V} \cong \mathbf{Bool} + \mathbf{Nat} + (\mathbf{V} \to \mathbf{V}) + (\mathbf{V} \times \mathbf{V}) + (\mathbf{V} + \mathbf{V}) + \{\mathsf{error}\}_\perp$$

and then interpret types by ideals of $\mathbf{V}$. The key observation is that $\mathbf{V}$ is the limit of its "approximations"

$$\mathbf{V}_0 \subseteq \mathbf{V}_1 \subseteq \mathbf{V}_2 \subseteq \cdots \subseteq \mathbf{V},$$

so that each (finite) element d of $\mathbf{V}$ is endowed with a *rank*, that is the first number n such that d appears in $\mathbf{V}_n$. This notion of rank "stratifies" $\mathbf{V}$, inducing a complete metric over the set of ideals. Banach's fixpoint theorem then yields a solution to every suitable recursive equation. A similar metric argument is employed in many of subsequent works.

2. Following Freyd [10], Pitts [19] identifies a key property of the domains that arises as the canonical solution of a domain equation, that is the *minimal invariant property*. As above, it yields stratification, i.e., endows each element d of the domain with a uniform expression as the least upper bound of its "approximations" (projections): $d = \bigsqcup_{n\in\mathbb{N}} \pi_n(d)$. This property is then used to give a quite general, flexible method to interpret recursive types as sets (or relations) on suitable domains.

3. Finally, two papers of Melliès and Vouillon [22], [17] are most influential to our work. They interpret types as sets of lambda terms closed under biorthogonality, where orthogonality is defined in terms of safety. Their first paper [22] considers syntactically defined approximation operators $(\ )^n$ such that a term t belongs to a closed set $\mathbf{D}$ if and only if all its approximations $(t)^n$ belongs to it. Again, it is this stratified structure that allows to solve recursive equations. The same idea also underlies [17], though this time approximation operations are "type-driven," i.e., associated to (interval) types.

Now we find the following common pattern:

(1) Interpret types as sets of elements of a domain or terms that satisfy certain closure properties.

(2) Stratify elements or terms (e.g., by the minimal invariant property or the approximation operator).

(3) Using the stratified structure, one proves that a sequence of "semantic types" converges, either by defining a complete metric, or more directly by taking the limits of lower and upper approximations and then showing that they coincide.

Our work also follows this pattern. A conceptual novelty is an observation that derivations in the proof system serve as the required stratification. Indeed, our intuition is that, given $\mathsf{Q} = (\mathbf{Q}_n)_{n\in\mathbb{N}} \in \mathsf{Cauchy}_{\Downarrow}$ and $Q \in \bigcap \mathbf{Q}_n$, derivations for $Q \vdash x_0 : \mathbf{Q}_n$ (that may be considered approximations) should "converge" to a derivation for $Q \vdash x_0 : \lim \mathsf{Q}$. Hence interpreting recursive types should be easy once derivations are assigned to designs by the completeness theorem. Thus, *denotational completeness yields interpretations of recursive types*.

Another feature of our work is an emphasis on the use of coinduction. A traditional approach to infinity is to think of an infinite object as the limit of its finite approximations. Such a view of infinity prevails in the literature of recursive types. On the other hand, we have tried as much as possible to directly deal with infinite objects by coinduction, as most typically seen in the direct, coinductive definition of Cauchy sequences rather than the metric one (see Remark IV.4).

Turning on to the technical side, we have achieved interactive (denotational) completeness and interpretations of recursive types in the polarized setting with tensor product $\otimes$, sum $\oplus$, their duals, and more generally all logical connectives in our sense. This certainly contains new connectives that have not been dealt with in the literature. On the other hand, our fragment does not include union, intersection, nor quantifications. These are left to the future considerations.

An obvious disadvantage of our work is the little flexibility. Indeed, denotational completeness is a hard requirement to achieve. Nevertheless, ludics based on safety is close to game semantics (especially to the polarized games [15]), for which some infinitary completeness results have been established. We plan to apply our insights to game semantics, revisiting some existing works on recursive types [1], [4], [5], to establish a coinductive approach to recursive games.

References

[1] Abramsky, S., McCusker, G.: Games for Recursive Types. Theory and Formal Methods (1994) 1–20.

[2] Basaldella, M., Faggian, C.: Ludics with Repetitions (Exponentials, Interactive types and Completeness). LICS (2009) 375–384.

[3] Basaldella, M., Terui, K.: On the meaning of logical completeness. TLCA (2009) 50–64. Extended version available at the homepage of the authors.

[4] Chroboczek, J.: Subtyping Recursive Games. TLCA (2001) 61–75.

[5] Clairambault, P.: Least and Greatest Fixpoints in Game Semantics. FoSSaCS (2009) 16–31.

[6] Curien, P.-L.: Introduction to linear logic and ludics, part II. Advances in Mathematics (China) **35**(1) (2006) 1–44.

[7] Curien, P.-L., Herbelin, H.: Abstract machines for dialogue games. Panoramas et Synthèses **27** (2008).

[8] Faggian, C.: Interactive observability in ludics: The geometry of tests. Theor. Comput. Sci. **350**(2) (2006) 213–233.

[9] Faggian, C., Piccolo, M.: Ludics is a model for the finitary linear pi-calculus. TLCA (2007) 148–162.

[10] Freyd, P. J.: Algebraically complete categories. Proc. Conf. Category Theory, Como 1990, LNM **1488** (1991) 131–156.

[11] Girard, J.-Y.: Linear Logic. Theor. Comput. Sci. **11** (1987) 1–102.

[12] Girard, J.-Y.: Locus solum: From the rules of logic to the logic of rules. Math. Struct. in Comp. Sci. **11**(3) (2001) 301–506.

[13] Hyland, J.M.E., Ong, C.H.L.: On full abstraction for PCF: I, II, and III. Inf. Comput. **163**(2) (2000) 285–408.

[14] Krivine, J.-L.: Realizability in classical logic. Panoramas et Synthèses **27** (2008).

[15] Laurent, O.: Polarized games. Ann. Pure Appl. Logic **130** (2004) 79–123.

[16] MacQueen, D., Plotkin, G., Sethi. R.: An ideal model for recursive polymorphic types. Information and Control **71**(1-2) (1986) 95–130.

[17] Melliès, P.-A., Vouillon, J.: Recursive Polymorphic Types and Parametricity in an Operational Framework. LICS (2005) 82–91.

[18] Paolini, L.: Parametric λ-Theories. Theor. Comput. Sci. **398**(1-3) (2008) 51–62.

[19] Pitts, A.M.: Relational properties of domains. Inform. and Comput. **127**(2) (1996) 66–90.

[20] Pitts, A.M.: Parametric polymorphism and operational equivalence. Math. Struct. in Comp. Sci. **10**(3) (2000) 321–359.

[21] Terui, K.: Computational ludics (2008) To appear in Theor. Comp. Sci.

[22] Vouillon, J., Melliès, P.-A.: Semantic types: a fresh look at the ideal model for types. POPL (2004) 52–63.

[23] Zeilberger, N.: Refinement types and computational duality. PLPV (2009) 15–26.