Inductive and Functional Types in Ludics

Alice Pavaux

Université Paris 13, Sorbonne Paris Cité, LIPN, CNRS, UMR 7030, Paris, France

Abstract

Ludics is a logical framework in which types/formulas are modelled by sets of terms with the same computational behaviour. This paper investigates the representation of inductive data types and functional types in ludics. We study their structure following a game semantics approach. Inductive types are interpreted as least fixed points, and we prove an internal completeness result giving an explicit construction for such fixed points. The interactive properties of the ludics interpretation of inductive and functional types are then studied. In particular, we identify which higher-order functions types fail to satisfy type safety, and we give a computational explanation.

1998 ACM Subject Classification F.1.1 Models of Computation, F.4.1 Mathematical Logic

Keywords and phrases Ludics, Inductive types, Fixed point, Linear logic, Game semantics

Digital Object Identifier 10.4230/LIPIcs.CSL.2017.34

1 Introduction

1.1 Context and Contributions

Context. *Ludics* was introduced by Girard [10] as a variant of *game semantics* with interactive types. Game Semantics has successfully provided fully abstract models for various logical systems and programming languages, among which PCF [11]. Although very close to Hyland–Ong (HO) games, ludics reverses the approach: in HO games one defines first the interpretation of a type (an arena) before giving the interpretation for the terms of that type (the strategies), while in ludics the interpretation of terms (the *designs*) is primitive and the types (the *behaviours*) are recovered dynamically as well-behaved sets of terms. This approach to types is similar to what exists in realisability [12] or geometry of interaction [9].

The motivation for such a framework was to reconstruct logic around the dynamics of proofs. Girard provides a ludics model for (a polarised version of) multiplicative-additive linear logic (MALL); a key role in his interpretation of logical connectives is played by the *internal completeness* results, which allow for a direct description of the behaviours' content. As most behaviours are not the interpretation of MALL formulas, an interesting question, raised from the beginning of ludics, is whether these remaining behaviours can give a logical counterpart to computational phenomena. In particular, data and functions [16, 15], and also fixed points [2] have been studied in the setting of ludics. The present work follows this line of research.

Real life (functional) programs usually deal with data, functions over it, functions over functions, etc. *Data types* allow one to present information in a structured way. Some data types are defined *inductively*, for example:

Listing 1 Example of inductive types in OCaml

```
> type nat = Zero | Succ of nat ;;
> type 'a list = Nil | Cons of 'a * 'a list ;;
> type 'a tree = Empty | Node of 'a * ('a tree) list ;;
```

Upon this basis we can consider *functional types*, which are either first-order – from data to data – or higher-order – i.e., taking functions as arguments or returning functions as a result. This article aims at interpreting constructively the (potentially inductive) data types and the (potentially higher-order) functional types as behaviours of ludics, so as to study their structural properties. Inductive types are defined as (least) *fixed points*. As pointed out by Baelde, Doumane and Saurin [2], the fact that ludics puts the most constraints on the formation of terms instead of types, conversely to game semantics, makes it a more natural setting for the interpretation of fixed points than HO games [4].

Contributions. The main contributions of this article are the following:

- We prove that internal completeness holds for infinite unions of behaviours satisfying particular conditions (Theorem 30), leading to an explicit construction of the least fixed points in ludics (Proposition 34).
- Inductive and functional types are interpreted as behaviours, and we prove that such behaviours are *regular* (Corollary 35 and Proposition 42). Regularity (that we discuss more in § 1.2) is a property that could be used to characterise the behaviours corresponding to μMALL formulas [1, 2] – i.e., MALL with fixed points.
- We show that a functional behaviour fails to satisfy *purity*, a property ensuring the safety of all possible executions (further explained in § 1.2), if and only if it is higher order and takes functions as argument (Proposition 43); this is typically the case of $(\mathbf{A} \multimap \mathbf{B}) \multimap \mathbf{C}$. In § 5.2 we discuss the computational meaning of this result.

The present work is conducted in the term-calculus reformulation of ludics by Terui [16] restricted to the linear part – the idea is that programs call each argument at most once.

Related Work. The starting point for our study of inductive types as fixed points in ludics is the work by Baelde, Doumane and Saurin [2]. In their article, they provide a ludics model for μMALL, a variant of multiplicative-additive linear logic with least and greatest fixed points. The existence of fixed points in ludics is ensured by Knaster-Tarski theorem, but this approach does not provide an explicit way to construct the fixed points; we will consider Kleene fixed point theorem instead. Let us also mention the work of Melliès and Vouillon [13] which introduces a realisability model for recursive (i.e., inductive and coinductive) polymorphic types.

The representation of both data and functions in ludics has been studied previously. Terui [16] proposes to encode them as designs in order to express computability properties in ludics, but data and functions are not considered at the level of behaviours. Sironi [15] describes the behaviours corresponding to some data types: integers, lists, records, etc. as well as first-order function types; our approach generalises hers by considering generic data types and also higher order functions types.

1.2 Background

Behaviours and Internal Completeness. A behaviour $\mathbf{B}$ is a set of designs which pass the same set of tests $\mathbf{B}^{\perp}$, where tests are also designs. $\mathbf{B}^{\perp}$ is called the *orthogonal* of $\mathbf{B}$, and behaviours are closed under bi-orthogonal: $\mathbf{B}^{\perp\perp} = \mathbf{B}$. New behaviours can be formed upon others using various constructors. In this process, internal completeness, which can be seen as a built-in notion of observational equivalence, ensures that two agents reacting the same way to any test are actually equal. From a technical point of view, this means that it is not necessary to apply a $\perp\perp$-closure for the sets constructed to be behaviours.

Paths: Ludics as Game Semantics. This paper makes the most of the resemblance between ludics and HO game semantics. The connections between them have been investigated in many pieces of work [3, 6, 7] where designs are described as (innocent) strategies, i.e., in terms of the traces of their possible interactions. Following this idea, Fouqueré and Quatrini define *paths* [7], corresponding to legal plays in HO games, and they characterise a behaviour by its set of *visitable paths*. This is the approach we follow. The definitions of regularity and purity rely on paths, since they are properties of the possible interactions of a behaviour.

Regularity: Towards a Characterisation of μMALL? Our proof that internal completeness holds for an infinite union of increasingly large behaviours (Theorem 30) relies in particular on the additional hypothesis of regularity for these behaviours. Intuitively, a behaviour $\mathbf{B}$ is regular if every path in a design of $\mathbf{B}$ is realised by interacting with a design of $\mathbf{B}^{\perp}$, and vice versa. This property is not actually ad hoc: it was introduced by Fouqueré and Quatrini [8] to characterise the denotations of MALL formulas as being precisely the regular behaviours satisfying an additional finiteness condition. In this direction, our intuition is that – forgetting about finiteness – regularity captures the behaviours corresponding to formulas of μMALL. Although such a characterisation is not yet achieved, we provide a first step by showing that the *data patterns*, a subset of positive μMALL formulas, yield only regular behaviours (Proposition 33).

Purity: Type Safety. Ludics has a special feature for termination which is not present in game semantics: the *daimon* ✠. On a computational point of view, the daimon is commonly interpreted as an error, an exception raised at run-time causing the program to stop (see for example the notes of Curien [5]). Thinking of Ludics as a programming language, we would like to guarantee *type safety*, that is, ensure that “well typed programs cannot go wrong” [14]. This is the purpose of purity, a property of behaviours: in a pure behaviour, maximal interaction traces are ✠-free, in other words whenever the interaction stops with ✠ it is actually possible to “ask for more” and continue the computation. Introduced by Sironi [15] (and called *principality* in her work), this property is related to the notions of *winning* designs [10] and *pure* designs [16], but at the level of a behaviour. As expected, data types are pure (Corollary 40), but not always functional types are; we identify the precise cases where impurity arises (Proposition 43), and explain why some types are not safe.

1.3 Outline

In Section 2 we present ludics and we state internal completeness for the logical connectives constructions. In Section 3 we recall the notion of path, so as to define formally regularity and purity and prove their stability under the connectives. Section 4 studies inductive data types, which we interpret as behaviours; Kleene theorem and internal completeness for infinite union allows us to give an explicit and direct construction for the least fixed point, with no need for bi-orthogonal closure; we deduce that data types are regular and pure. Finally, in Section 5, we study functional types, showing in what case purity fails.

2 Computational Ludics

This section introduces the ludics background necessary for the rest of the paper, in the formalism of Terui [16]. The *designs* are the primary objects of ludics, corresponding to (polarised) proofs or programs in a Curry-Howard perspective. Cuts between designs can occur, and their reduction is called *interaction*. The *behaviours*, corresponding to the types or formulas of ludics, are then defined thanks to interaction. Compound behaviours can be formed with *logical connectives* constructions which satisfy *internal completeness.*

2.1 Designs and Interaction

Suppose given a set of variables $\mathcal{V}_0$ and a set $\mathcal{S}$, called **signature**, equipped with an arity function $ar : \mathcal{S} \to \mathbb{N}$. Elements $a, b, \dots \in \mathcal{S}$ are called **names**. A **positive action** is either ✠ (daimon), Ω (divergence), or $\overline{a}$ with $a \in \mathcal{S}$; a **negative action** is $a(x_1, \dots, x_n)$ where $a \in \mathcal{S}$, $\mathrm{ar}(a) = n$ and $x_1, \dots, x_n \in \mathcal{V}_0$ distinct. An action is **proper** if it is neither ✠ nor Ω.

▶ **Definition 1.** Positive and negative **designs**[1] are coinductively defined by:

$$\begin{aligned}\mathfrak{p} &::= \maltese \quad | \quad \Omega \quad | \quad x|\overline{a}\langle \mathfrak{n}_1, \dots, \mathfrak{n}_{\mathrm{ar}(a)}\rangle \quad | \quad \mathfrak{n}_0|\overline{a}\langle \mathfrak{n}_1, \dots, \mathfrak{n}_{\mathrm{ar}(a)}\rangle \\ \mathfrak{n} &::= \textstyle\sum_{a\in\mathcal{S}} a(x^a_1, \dots, x^a_{\mathrm{ar}(a)}).\mathfrak{p}_a\end{aligned}$$

Positive designs play the same role as *applications* in λ-calculus, and negative designs the role of *abstractions*, where each name $a \in \mathcal{S}$ binds $\mathrm{ar}(a)$ variables.

Designs are considered up to α-equivalence. We will often write $a(\overrightarrow{x})$ (resp. $\overline{a}\langle\overrightarrow{\mathfrak{n}}\rangle$) instead of $a(x_1, \dots, x_n)$ (resp. $\overline{a}\langle \mathfrak{n}_1 \dots \mathfrak{n}_n\rangle$). Negative designs can be written as partial sums, for example $a(x, y).\mathfrak{p} + b().\mathfrak{q}$ instead of $a(x, y).\mathfrak{p} + b().\mathfrak{q} + \sum_{c\neq a, c\neq b} c(\overrightarrow{z^c}).\Omega$.

Given a design $\mathfrak{d}$, the definitions of the **free variables** of $\mathfrak{d}$, written $\mathrm{fv}(\mathfrak{d})$, and the (capture-free) **substitution** of x by a negative design $\mathfrak{n}$ in $\mathfrak{d}$, written $\mathfrak{d}[\mathfrak{n}/x]$, can easily be inferred. The design $\mathfrak{d}$ is **closed** if it is positive and it has no free variable. A **subdesign** of $\mathfrak{d}$ is a subterm of $\mathfrak{d}$. A **cut** in $\mathfrak{d}$ is a subdesign of $\mathfrak{d}$ of the form $\mathfrak{n}_0|\overline{a}\langle\overrightarrow{\mathfrak{n}}\rangle$, and a design is **cut-free** if it has no cut.

In the following, we distinguish a particular variable x_0, that cannot be bound. A positive design $\mathfrak{p}$ is **atomic** if $\mathrm{fv}(\mathfrak{p}) \subseteq \{x_0\}$; a negative design $\mathfrak{n}$ is **atomic** if $\mathrm{fv}(\mathfrak{n}) = \emptyset$.

A design is **linear** if for every subdesign of the form $x|\overline{a}\langle\overrightarrow{\mathfrak{n}}\rangle$ (resp. $\mathfrak{n}_0|\overline{a}\langle\overrightarrow{\mathfrak{n}}\rangle$), the sets $\{x\}$, $\mathrm{fv}(\mathfrak{n}_1)$, ..., $\mathrm{fv}(\mathfrak{n}_{\mathrm{ar}(a)})$ (resp. the sets $\mathrm{fv}(\mathfrak{n}_0)$, $\mathrm{fv}(\mathfrak{n}_1)$, ..., $\mathrm{fv}(\mathfrak{n}_{\mathrm{ar}(a)})$) are pairwise disjoint. This article focuses on linearity, so in the following when writing “design” we mean “linear design”.

▶ **Definition 2.** The **interaction** corresponds to reduction steps applied on cuts:

$$\textstyle\sum_{a\in\mathcal{S}} a(x^a_1, \dots, x^a_{\mathrm{ar}(a)}).\mathfrak{p}_a \mid \overline{b}\langle \mathfrak{n}_1, \dots, \mathfrak{n}_k\rangle \qquad \rightsquigarrow \qquad \mathfrak{p}_b[\mathfrak{n}_1/x^b_1, \dots, \mathfrak{n}_k/x^b_k]$$

We will later describe an interaction as a sequence of actions, a path (Definition 13).

Let $\mathfrak{p}$ be a design, and let $\rightsquigarrow^*$ denote the reflexive transitive closure of $\rightsquigarrow$; if there exists a design $\mathfrak{q}$ which is neither a cut nor Ω and such that $\mathfrak{p} \rightsquigarrow^* \mathfrak{q}$, we write $\mathfrak{p} \Downarrow \mathfrak{q}$; otherwise we write $\mathfrak{p} \Uparrow$. The normal form of a design, defined below, exists and is unique [16].

▶ **Definition 3.** The **normal form** of a design $\mathfrak{d}$, noted $[\![\mathfrak{d}]\!]$, is defined by:

$$\begin{aligned}&[\![\mathfrak{p}]\!] = \maltese \quad \text{if } \mathfrak{p} \Downarrow \maltese && [\![\mathfrak{p}]\!] = x|\overline{a}\langle [\![\mathfrak{n}_1]\!], \dots, [\![\mathfrak{n}_n]\!]\rangle \quad \text{if } \mathfrak{p} \Downarrow x|\overline{a}\langle \mathfrak{n}_1, \dots, \mathfrak{n}_n\rangle \\ &[\![\mathfrak{p}]\!] = \Omega \quad \text{if } \mathfrak{p} \Uparrow && [\![\textstyle\sum_{a\in\mathcal{S}} a(\overrightarrow{x^a}).\mathfrak{p}_a]\!] = \textstyle\sum_{a\in\mathcal{S}} a(\overrightarrow{x^a}).[\![\mathfrak{p}_a]\!]\end{aligned}$$

Note that the normal form of a closed design is either ✠ (convergence) or Ω (divergence). Orthogonality expresses the convergence of the interaction between two atomic designs, and behaviours are sets of designs closed by bi-orthogonal.

[1] In the following, the symbols $\mathfrak{d}, \mathfrak{e}, \dots$ refer to designs of any polarity, while $\mathfrak{p}, \mathfrak{q}, \dots$ and $\mathfrak{m}, \mathfrak{n}, \dots$ are specifically for positive and negative designs respectively.

▶ **Definition 4.** Two atomic designs $\mathfrak{p}$ and $\mathfrak{n}$ are **orthogonal**, noted $\mathfrak{p} \perp \mathfrak{n}$, if $\llbracket \mathfrak{p}[\mathfrak{n}/x_0] \rrbracket = \maltese$.

Given an atomic design $\mathfrak{d}$, define $\mathfrak{d}^\perp = \{\mathfrak{e} \mid \mathfrak{d} \perp \mathfrak{e}\}$; if E is a set of atomic designs of same polarity, define $E^\perp = \{\mathfrak{d} \mid \forall \mathfrak{e} \in E, \mathfrak{d} \perp \mathfrak{e}\}$.

▶ **Definition 5.** A set $\mathbf{B}$ of atomic designs of same polarity is a **behaviour**[2] if $\mathbf{B}^{\perp\perp} = \mathbf{B}$. A behaviour is either positive or negative depending on the polarity of its designs.

Behaviours could alternatively be defined as the orthogonal of a set E of atomic designs of same polarity – E corresponds to a set of *tests* or *trials*. Indeed, $E^\perp$ is always a behaviour, and every behaviour $\mathbf{B}$ is of this form by taking $E = \mathbf{B}^\perp$.

The *incarnation* of a behaviour $\mathbf{B}$ contains the cut-free designs of $\mathbf{B}$ whose actions are all visited during an interaction with a design in $\mathbf{B}^\perp$. Those correspond to the cut-free designs that are minimal for the **stable ordering** $\sqsubseteq$, where $\mathfrak{d}' \sqsubseteq \mathfrak{d}$ if $\mathfrak{d}$ can be obtained from $\mathfrak{d}'$ by substituting positive subdesigns for some occurrences of Ω.

▶ **Definition 6.** Let $\mathbf{B}$ be a behaviour and $\mathfrak{d} \in \mathbf{B}$ cut-free.

- The **incarnation** of $\mathfrak{d}$ in $\mathbf{B}$, written $|\mathfrak{d}|_{\mathbf{B}}$, is the smallest (for $\sqsubseteq$) cut-free design $\mathfrak{d}'$ such that $\mathfrak{d}' \sqsubseteq \mathfrak{d}$ and $\mathfrak{d}' \in \mathbf{B}$. If $|\mathfrak{d}|_{\mathbf{B}} = \mathfrak{d}$ we say that $\mathfrak{d}$ is **incarnated** in $\mathbf{B}$.
- The **incarnation** $|\mathbf{B}|$ of $\mathbf{B}$ is the set of the (cut-free) incarnated designs of $\mathbf{B}$.

2.2 Logical Connectives

Behaviour constructors – the *logical connectives* – can be applied so as to form compound behaviours. These connectives, coming from (polarised) linear logic, are used for interpreting formulas as behaviours, and will also indeed play the role of type constructors for the types of data and functions. In this subsection, after defining the connectives we consider, we state the *internal completeness* theorem for these connectives.

Let us introduce some notations. In the rest of this article, suppose the signature $\mathcal{S}$ contains distinct unary names $\blacktriangle, \pi_1, \pi_2$ and a binary name $\wp$, and write $\blacktriangledown = \overline{\blacktriangle}, \iota_1 = \overline{\pi_1}, \iota_2 = \overline{\pi_2}$ and $\bullet = \overline{\wp}$. Given a behaviour $\mathbf{B}$ and x fresh, define $\mathbf{B}^x = \{\mathfrak{d}[x/x_0] \mid \mathfrak{d} \in \mathbf{B}\}$; such a substitution operates a "delocation" with no repercussion on the behaviour's inherent properties. Given a k-ary name $a \in \mathcal{S}$, we write $\overline{a}\langle \mathbf{N}_1, \dots, \mathbf{N}_k \rangle$ or even $\overline{a}\langle \overrightarrow{\mathbf{N}} \rangle$ for $\{x_0 | \overline{a}\langle \overrightarrow{\mathfrak{n}} \rangle \mid \mathfrak{n}_i \in \mathbf{N}_i\}$, and write $a(\overrightarrow{x}).\mathbf{P}$ for $\{a(\overrightarrow{x}).\mathfrak{p} \mid \mathfrak{p} \in \mathbf{P}\}$. For a negative design $\mathfrak{n} = \sum_{a \in \mathcal{S}} a(\overrightarrow{x^a}).\mathfrak{p}_a$ and a name $a \in \mathcal{S}$, we denote by $\mathfrak{n} \restriction a$ the design $a(\overrightarrow{x^a}).\mathfrak{p}_a$ (that is $a(\overrightarrow{x^a}).\mathfrak{p}_a + \sum_{b \neq a} b(\overrightarrow{x^b}).\Omega$).

▶ **Definition 7** (Logical connectives).

$\downarrow\mathbf{N} = \blacktriangledown\langle \mathbf{N} \rangle^{\perp\perp}$	**(positive shift)**
$\uparrow\mathbf{P} = (\blacktriangle(x).\mathbf{P}^x)^{\perp\perp}$, with x fresh	**(negative shift)**
$\mathbf{M} \oplus \mathbf{N} = (\iota_1\langle \mathbf{M} \rangle \cup \iota_2\langle \mathbf{N} \rangle)^{\perp\perp}$	**(plus)**
$\mathbf{M} \otimes \mathbf{N} = \bullet\langle \mathbf{M}, \mathbf{N} \rangle^{\perp\perp}$	**(tensor)**
$\mathbf{N} \multimap \mathbf{P} = (\mathbf{N} \otimes \mathbf{P}^\perp)^\perp$	**(linear map)**

Our connectives $\downarrow$, $\uparrow$, $\oplus$ and $\otimes$ match exactly those defined by Terui [16], who also proves the following internal completeness theorem stating that connectives apply on behaviours in a constructive way – there is no need to close by bi-orthogonal. For each connective, we

[2] Symbols $\mathbf{A}, \mathbf{B}, \dots$ will designate behaviours of any polarity, while $\mathbf{M}, \mathbf{N} \dots$ and $\mathbf{P}, \mathbf{Q}, \dots$ will be for negative and positive behaviours respectively.

present two versions of internal completeness: one concerned with the full behaviour, the other with the behaviour's incarnation.

▶ **Theorem 8** (Internal completeness for connectives).

$$\begin{array}{ll}
{\downarrow}\mathbf{N} = \blacktriangledown\langle\mathbf{N}\rangle \cup \{\maltese\} & |{\downarrow}\mathbf{N}| = \blacktriangledown\langle|\mathbf{N}|\rangle \cup \{\maltese\} \\
{\uparrow}\mathbf{P} = \{\mathfrak{n} \mid \mathfrak{n}{\restriction}\blacktriangle \in \blacktriangle(x).\mathbf{P}^x\} & |{\uparrow}\mathbf{P}| = \blacktriangle(x).|\mathbf{P}^x| \\
\mathbf{M} \oplus \mathbf{N} = \iota_1\langle\mathbf{M}\rangle \cup \iota_2\langle\mathbf{N}\rangle \cup \{\maltese\} & |\mathbf{M} \oplus \mathbf{N}| = \iota_1\langle|\mathbf{M}|\rangle \cup \iota_2\langle|\mathbf{N}|\rangle \cup \{\maltese\} \\
\mathbf{M} \otimes \mathbf{N} = \bullet\langle\mathbf{M}, \mathbf{N}\rangle \cup \{\maltese\} & |\mathbf{M} \otimes \mathbf{N}| = \bullet\langle|\mathbf{M}|, |\mathbf{N}|\rangle \cup \{\maltese\}
\end{array}$$

3 Paths and Interactive Properties of Behaviours

Paths are sequences of actions recording the trace of a possible interaction. For a behaviour $\mathbf{B}$, we can consider the set of its *visitable paths* by gathering all the paths corresponding to an interaction between a design of $\mathbf{B}$ and a design of $\mathbf{B}^\perp$. This notion is needed for defining *regularity* and *purity* and proving that those two properties of behaviours are stable under (some) connectives constructions.

3.1 Paths

This subsection adapts the definitions of path and visitable path from [7] to the setting of computational ludics. In order to do so, we need first to recover *location* in actions so as to consider sequences of actions.

Location is a primitive idea in Girard's ludics [10] in which the places of a design are identified with *loci* or *addresses*, but this concept is not visible in Terui's presentation of designs-as-terms. We overcome this by introducing actions with more information on location, which we call *located actions*, and which are necessary to:

- represent cut-free designs as trees – actually, forests – in a satisfactory way,
- define views and paths.

▶ **Definition 9.** A **located action**[3] κ is one of: $\maltese \;\mid\; x|\overline{a}\langle x_1, \dots, x_{\mathrm{ar}(a)}\rangle \;\mid\; a_x(x_1, \dots, x_{\mathrm{ar}(a)})$ where in the last two cases (**positive proper** and **negative proper** respectively), $a \in \mathcal{S}$ is the **name** of κ, the variables $x, x_1, \dots, x_{\mathrm{ar}(a)}$ are distinct, x is the **address** of κ and $x_1, \dots, x_{\mathrm{ar}(a)}$ are the **variables bound by** κ.

In the following, "action" will always refer to a located action. Similarly to notations for designs, $x|\overline{a}\langle\overrightarrow{x}\rangle$ stands for $x|\overline{a}\langle x_1, \dots, x_n\rangle$ and $a_x(\overrightarrow{x})$ for $a_x(x_1, \dots, x_n)$.

▶ **Example 10.** We show how cut-free designs can be represented as trees of located actions in this example. Let $a^2, b^2, c^1, d^0 \in \mathcal{S}$, where exponents stand for arities. The following design is represented by the tree of Fig. 1.

$$\mathfrak{d} = a(x_1, x_2).(x_2|\overline{b}\langle a(x_3, x_4).\maltese + c(y_1).(y_1|\overline{d}\langle\rangle), c(y_2).(x_1|\overline{d}\langle\rangle)\rangle)$$

Such a representation is in general a forest: a negative design $\sum_{a \in \mathcal{S}} a(\overrightarrow{x^a}).\mathfrak{p}_a$ gives as many trees as there is $a \in \mathcal{S}$ such that $\mathfrak{p}_a \neq \Omega$. The distinguished variable x_0 is given as address to every negative root of a tree, and fresh variables are picked as addresses for negative actions bound by positive ones. This way, negative actions from the same subdesign,

[3] Located actions will often be denoted by symbol κ, sometimes with its polarity: κ^+ or κ^-.

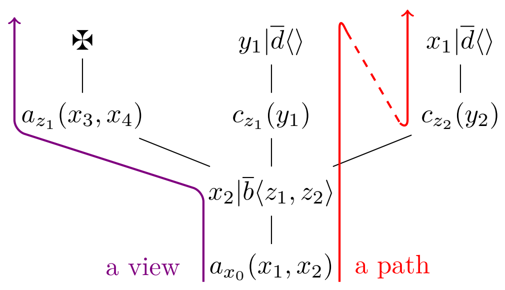
$\maltese$
$y_1|\overline{d}\langle\rangle$
$x_1|\overline{d}\langle\rangle$
$a_{z_1}(x_3, x_4)$
$c_{z_1}(y_1)$
$c_{z_2}(y_2)$
$x_2|\overline{b}\langle z_1, z_2\rangle$
a view
$a_{x_0}(x_1, x_2)$
a path

Figure 1 Representation of design $\mathfrak{d}$ from Example 10, with a path and a view of $\mathfrak{d}$.

i.e., part of the same sum, are given the same address. A tree is indeed to be read bottom-up: a proper action κ is **justified** if its address is bound by an action of opposite polarities appearing below κ in the tree; otherwise κ is called **initial**. Except the root of a tree, which is always initial, every negative action is justified by the only positive action immediately below it. If κ and κ' are proper, κ is **hereditarily justified** by κ' if there exist actions $\kappa_1, \dots, \kappa_n$ such that $\kappa = \kappa_1$, $\kappa' = \kappa_n$ and for all i such that $1 \leq i < n$, κ_i is justified by κ_{i+1}.

Before giving the definitions of *view* and *path*, let us give an intuition. On Fig. 1 are represented a view and a path of design $\mathfrak{d}$. Views are branches in the tree representing a cut-free design (reading bottom-up), while paths are particular "promenades" starting from the root of the tree; not all such promenades are paths, though. Views correspond to *chronicles* in original ludics [10].

For every positive proper action $\kappa^+ = x|\overline{a}\langle\overrightarrow{y}\rangle$ define $\overline{\kappa^+} = a_x(\overrightarrow{y})$, and similarly if $\kappa^- = a_x(\overrightarrow{y})$ define $\overline{\kappa^-} = x|\overline{a}\langle\overrightarrow{y}\rangle$. Given a finite sequence of proper actions $\mathfrak{s} = \kappa_1 \dots \kappa_n$, define $\overline{\mathfrak{s}} = \overline{\kappa_1} \dots \overline{\kappa_n}$. Suppose now that if $\mathfrak{s}$ contains an occurrence of $\maltese$, it is necessarily in last position; the **dual** of $\mathfrak{s}$, written $\tilde{\mathfrak{s}}$, is the sequence defined by:

- $\tilde{\mathfrak{s}} = \overline{\mathfrak{s}}\maltese$ if $\mathfrak{s}$ does not end with $\maltese$,
- $\tilde{\mathfrak{s}} = \overline{\mathfrak{s}'}$ if $\mathfrak{s} = \mathfrak{s}'\maltese$.

Note that $\tilde{\tilde{\mathfrak{s}}} = \mathfrak{s}$. The notions of **justified**, **hereditarily justified** and **initial** actions also apply in sequences of actions.

▶ **Definition 11.** An **alternated justified sequence** (or **aj-sequence**) $\mathfrak{s}$ is a finite sequence of actions such that:

- (Alternation) Polarities of actions alternate.
- (Daimon) If $\maltese$ appears, it is the last action of $\mathfrak{s}$.
- (Linearity) Each variable is the address of at most one action in $\mathfrak{s}$.

The (unique) justification of a justified action κ in an aj-sequence is noted $\mathrm{just}(\kappa)$, when there is no ambiguity on the sequence we consider.

▶ **Definition 12.** A **view** $\mathfrak{v}$ is an aj-sequence such that each negative action which is not the first action of $\mathfrak{v}$ is justified by the immediate previous action. Given a cut-free design $\mathfrak{d}$, $\mathfrak{v}$ is a **view of** $\mathfrak{d}$ if it is a branch in the representation of $\mathfrak{d}$ as a tree (modulo α-equivalence).

The way to extract **the view** of an aj-sequence is given inductively by:

- $\ulcorner\epsilon\urcorner = \epsilon$, where ϵ is the empty sequence,
- $\ulcorner\mathfrak{s}\kappa^+\urcorner = \ulcorner\mathfrak{s}\urcorner\kappa^+$,
- $\ulcorner\mathfrak{s}\kappa^-\urcorner = \ulcorner\mathfrak{s}_0\urcorner\kappa^-$ where $\mathfrak{s}_0$ is the prefix of $\mathfrak{s}$ ending on $\mathrm{just}(\kappa^-)$, or $\mathfrak{s}_0 = \epsilon$ if κ^- initial.

The **anti-view** of an aj-sequence, noted $\llcorner\mathfrak{s}\lrcorner$, is defined symmetrically by reversing the role played by polarities; equivalently $\llcorner\mathfrak{s}\lrcorner = \widetilde{\ulcorner\tilde{\mathfrak{s}}\urcorner}$.

▶ **Definition 13.** A **path** s is a positive-ended aj-sequence satisfying:

- (P-visibility) For all prefix $s'\kappa^+$ of s, just$(\kappa^+) \in \ulcorner s' \urcorner$
- (O-visibility) For all prefix $s'\kappa^-$ of s, just$(\kappa^-) \in \llcorner s' \lrcorner$

Given a cut-free design $\mathfrak{d}$, a path s is a **path of** $\mathfrak{d}$ if for all prefix s' of s, $\ulcorner s' \urcorner$ is a view of $\mathfrak{d}$.

Remark that the dual of a path is a path.

Paths are aimed at describing an interaction between designs. If $\mathfrak{d}$ and $\mathfrak{e}$ are cut-free atomic designs such that $\mathfrak{d} \perp \mathfrak{e}$, there exists a unique path s of $\mathfrak{d}$ such that $\widetilde{s}$ is a path of $\mathfrak{e}$. We write this path $\langle \mathfrak{d} \leftarrow \mathfrak{e} \rangle$, and the good intuition is that it corresponds to the sequence of actions followed by the interaction between $\mathfrak{d}$ and $\mathfrak{e}$ on the side of $\mathfrak{d}$. An alternative way defining orthogonality is then given by the following proposition.

▶ **Proposition 14.** $\mathfrak{d} \perp \mathfrak{e}$ *if and only if there exists a path s of $\mathfrak{d}$ such that $\widetilde{s}$ is a path of $\mathfrak{e}$.*

At the level a behaviour $\mathbf{B}$, the set of visitable paths describes all the possible interactions between a design of $\mathbf{B}$ and a design of $\mathbf{B}^\perp$.

▶ **Definition 15.** A path s is **visitable** in a behaviour $\mathbf{B}$ if there exist cut-free designs $\mathfrak{d} \in \mathbf{B}$ and $\mathfrak{e} \in \mathbf{B}^\perp$ such that $s = \langle \mathfrak{d} \leftarrow \mathfrak{e} \rangle$. The set of visitable paths of $\mathbf{B}$ is written $V_\mathbf{B}$.

Note that for every behaviour $\mathbf{B}$, $\widetilde{V_\mathbf{B}} = V_{\mathbf{B}^\perp}$.

3.2 Regularity, Purity and Connectives

The meaning of regularity and purity has been discussed in the introduction. After giving the formal definitions, we prove that regularity is stable under all the connectives constructions. We also show that purity may fail with $\multimap$, and only a weaker form called *quasi-purity* is always preserved.

▶ **Definition 16.** $\mathbf{B}$ is **regular** if the following conditions are satisfied:

- for all $\mathfrak{d} \in |B|$ and all path s of $\mathfrak{d}$, $s \in V_\mathbf{B}$,
- for all $\mathfrak{d} \in |B^\perp|$ and all path s of $\mathfrak{d}$, $s \in V_{\mathbf{B}^\perp}$,
- The sets $V_\mathbf{B}$ and $V_{\mathbf{B}^\perp}$ are stable under shuffle.

where the operation of **shuffle** ($\sqcup\!\sqcup$) on paths corresponds to an interleaving of actions respecting alternation of polarities, and is defined below.

Let $s\upharpoonright s'$ refer to the subsequence of s containing only the actions that occur in s'. Let s and t be paths of same polarity, let S and T be sets of paths of same polarity. We define:

- $s \sqcup\!\sqcup t = \{u$ path formed with actions from s and $t \mid u\upharpoonright s = s$ and $u\upharpoonright t = t\}$ if s, t negative,
- $s \sqcup\!\sqcup t = \{\kappa^+ u$ path $\mid u \in s' \sqcup\!\sqcup t'\}$ if $s = \kappa^+ s'$ and $t = \kappa^+ t'$ positive with same first action,
- $S \sqcup\!\sqcup T = \{u$ path $\mid \exists s \in S, \exists t \in T$ such that $s \sqcup\!\sqcup t$ is defined and $u \in s \sqcup\!\sqcup t\}$,

In fact, a behaviour $\mathbf{B}$ is regular if every path formed with actions of the incarnation of $\mathbf{B}$, even mixed up, is a visitable path of $\mathbf{B}$, and similarly for $\mathbf{B}^\perp$. Remark that regularity is a property of both a behaviour and its orthogonal since the definition is symmetrical: $\mathbf{B}$ is regular if and only if $\mathbf{B}^\perp$ is regular.

▶ **Definition 17.** A behaviour $\mathbf{B}$ is **pure** if every ✠-ended path $s✠ \in V_\mathbf{B}$ is **extensible**, i.e., there exists a proper positive action κ^+ such that $s\kappa^+ \in V_\mathbf{B}$.

Purity ensures that when an interaction encounters ✠, this does not correspond to a real error but rather to a partial computation, as it is possible to continue this interaction. Note that daimons are necessarily present in all behaviours since the converse property is always true: if $s\kappa^+ \in V_\mathbf{B}$ then $s✠ \in V_\mathbf{B}$.

▶ **Proposition 18.** *Regularity is stable under* $\downarrow$, $\uparrow$, $\oplus$, $\otimes$ *and* $\multimap$.

▶ **Proposition 19.** *Purity is stable under* $\downarrow$, $\uparrow$, $\oplus$ *and* $\otimes$.

Unfortunately, when $\mathbf{N}$ and $\mathbf{P}$ are pure, $\mathbf{N} \multimap \mathbf{P}$ is not necessarily pure, even under regularity assumption. However, a weaker form of purity holds for $\mathbf{N} \multimap \mathbf{P}$.

▶ **Definition 20.** A behaviour $\mathbf{B}$ is **quasi-pure** if all the $\maltese$-ended *well-bracketed* paths in $V_{\mathbf{B}}$ are extensible.

We recall that a path s is **well-bracketed** if, for every justified action κ in s, when we write $s = s_0\kappa' s_1 \kappa s_2$ where κ' justifies κ, all the actions in s_1 are hereditarily justified by κ'.

▶ **Proposition 21.** *If* $\mathbf{N}$ *and* $\mathbf{P}$ *are quasi-pure and regular then* $\mathbf{N} \multimap \mathbf{P}$ *is quasi-pure.*

4 Inductive Data Types

Some important contributions are presented in this section. We interpret inductive data types as positive behaviours, and we prove an internal completeness result allowing us to make explicit the structure of fixed points. Regularity and purity of data follows.

Abusively, we denote the positive behaviour $\{\maltese\}$ by $\maltese$ all along this section.

4.1 Inductive Data Types as Kleene Fixed Points

We define the *data patterns* via a type language and interpret them as behaviours, in particular μ is interpreted as a least fixed point. *Data behaviours* are the interpretation of *steady* data patterns.

Suppose given a countably infinite set $\mathcal{V}$ of second-order variables: $X, Y, \dots \in \mathcal{V}$. Let $\mathcal{S}' = \mathcal{S} \setminus \{\blacktriangle, \pi_1, \pi_2, \wp\}$ and define the set of **constants** $\mathrm{Const} = \{\mathbf{C}_a \mid a \in \mathcal{S}'\}$ which contains a behaviour $\mathbf{C}_a = \{x_0|\overline{a}\langle\overrightarrow{\Omega^-}\rangle\}^{\perp\perp}$ (where $\Omega^- := \sum_{a\in\mathcal{S}} a(\overrightarrow{x^a}).\Omega$) for each $a \in \mathcal{S}'$, i.e., such that a is not the name of a connective. Remark that $V_{\mathbf{C}_a} = \{\maltese\ ,\ x_0|\overline{a}\langle\overrightarrow{x}\rangle\}$, thus $\mathbf{C}_a$ is regular and pure.

▶ **Definition 22.** The set $\mathcal{P}$ of **data patterns** is generated by the inductive grammar:

$$A, B ::= X \in \mathcal{V} \quad | \quad a \in \mathcal{S}' \quad | \quad A \oplus^+ B \quad | \quad A \otimes^+ B \quad | \quad \mu X.A$$

The set of free variables of a data pattern $A \in \mathcal{P}$ is denoted by $\mathrm{FV}(A)$.

▶ **Example 23.** Let $b, n, l, t \in \mathcal{S}'$ and $X \in \mathcal{V}$. The data types given as example in the introduction can be written in the language of data patterns as follows:

$$\mathbb{B}\mathrm{ool} = b \oplus^+ b \qquad \mathbb{N}\mathrm{at} = \mu X.(n \oplus^+ X) \qquad \mathbb{L}\mathrm{ist}_A = \mu X.(l \oplus^+ (A \otimes^+ X))$$
$$\mathbb{T}\mathrm{ree}_A = \mu X.(t \oplus^+ (A \otimes^+ \mathbb{L}\mathrm{ist}_X)) = \mu X.(t \oplus^+ (A \otimes^+ \mu Y.(l \oplus^+ (X \otimes^+ Y))))$$

Let $\mathcal{B}^+$ be the set of positive behaviours. Given a data pattern $A \in \mathcal{P}$ and an environment σ, i.e., a function that maps free variables to positive behaviours, the interpretation of A in the environment σ, written $[\![A]\!]^\sigma$, is the positive behaviour defined by:

$$[\![X]\!]^\sigma = \sigma(X) \qquad\qquad [\![A \oplus^+ B]\!]^\sigma = (\uparrow[\![A]\!]^\sigma) \oplus (\uparrow[\![B]\!]^\sigma)$$
$$[\![a]\!]^\sigma = \mathbf{C}_a \qquad\qquad [\![A \otimes^+ B]\!]^\sigma = (\uparrow[\![A]\!]^\sigma) \otimes (\uparrow[\![B]\!]^\sigma)$$
$$[\![\mu X.A]\!]^\sigma = \mathrm{lfp}(\phi^A_\sigma)$$

where lfp stands for the least fixed point, and the function $\phi_\sigma^A : \mathcal{B}^+ \to \mathcal{B}^+, \mathbf{P} \mapsto [\![A]\!]^{\sigma, X \mapsto \mathbf{P}}$ is well defined and has a least fixed point by Knaster-Tarski fixed point theorem, as shown by Baelde, Doumane and Saurin [2]. Abusively we may write $\oplus^+$ and $\otimes^+$, instead of $(\uparrow\cdot) \oplus (\uparrow\cdot)$ and $(\uparrow\cdot) \otimes (\uparrow\cdot)$ respectively, for behaviours. We call an environment σ regular (resp. pure) if its image contains only regular (resp. pure) behaviours. The notation $\sigma, X \mapsto \mathbf{P}$ stands for the environment σ where the image of X has been changed to $\mathbf{P}$.

In order to understand the structure of fixed point behaviours that interpret the data patterns of the form $\mu X.A$, we need a constructive approach, thus Kleene fixed point theorem is best suited than Knaster-Tarski. We now prove that we can apply this theorem.

Recall the following definitions and theorem. A partial order is a **complete partial order** (CPO) if each directed subset has a supremum, and there exists a smallest element, written $\perp$. A function $f : E \to F$ between two CPOs is **Scott-continuous** (or simply continuous) if for every directed subset $D \subseteq E$ we have $\bigvee_{x \in D} f(x) = f(\bigvee_{x \in D} x)$.

▶ **Theorem 24** (Kleene fixed point theorem). *Let L be a CPO and let $f : L \to L$ be Scott-continuous. The function f has a least fixed point, defined by*

$$\mathrm{lfp}(f) = \bigvee_{n \in \mathbb{N}} f^n(\perp)$$

The set $\mathcal{B}^+$ ordered by $\subseteq$ is a CPO, with least element $\maltese$; indeed, given a subset $\mathbb{P} \subseteq \mathcal{B}^+$, it is directed and we have $\bigvee \mathbb{P} = (\bigcup \mathbb{P})^{\perp\perp}$. Hence next proposition proves that we can apply the theorem.

▶ **Proposition 25.** *Given a data pattern $A \in \mathcal{P}$, a variable $X \in \mathcal{V}$ and an environment $\sigma : \mathrm{FV}(A) \setminus \{X\} \to \mathcal{B}^+$, the function ϕ_σ^A is Scott-continuous.*

▶ **Corollary 26.** *For every $A \in \mathcal{P}$, $X \in \mathcal{V}$ and $\sigma : \mathrm{FV}(A) \setminus \{X\} \to \mathcal{B}^+$,*

$$[\![\mu X.A]\!]^\sigma = \bigvee_{n \in \mathbb{N}} (\phi_\sigma^A)^n(\maltese) = (\bigcup_{n \in \mathbb{N}} (\phi_\sigma^A)^n(\maltese))^{\perp\perp}.$$

This result gives an explicit formulation for least fixed points. However, the $\perp\perp$-closure might add new designs which were not in the union, making it difficult to know the exact content of such a behaviour. The point of next subsection will be to give an internal completeness result proving that the closure is actually not necessary.

Let us finish this subsection by defining a restricted set of data patterns so as to exclude the degenerate ones. Consider for example $\mathbb{L}ist_A{}' = \mu X.(A \otimes^+ X)$, a variant of $\mathbb{L}ist_A$ (see Example 23) which misses the base case. It is degenerate in the sense that the base element, here the empty list, is interpreted as the design $\maltese$. This is problematic: an interaction going through a whole list will end with an error, making it impossible to explore a pair of lists for example. The pattern $\mathbb{N}at' = \mu X.X$ is even worse since $[\![\mathbb{N}at']\!] = \maltese$. The point of steady data patterns is to ensure the existence of a basis; this will be formalised in Lemma 37.

▶ **Definition 27.** The set of **steady** data patterns is the smallest subset $\mathcal{P}^s \subseteq \mathcal{P}$ such that:

- $\mathcal{S}' \subseteq \mathcal{P}^s$
- If $A \in \mathcal{P}^s$ and B is such that $[\![B]\!]^\sigma$ is pure if σ is pure, then $A \oplus^+ B \in \mathcal{P}^s$ and $B \oplus^+ A \in \mathcal{P}^s$
- If $A \in \mathcal{P}^s$ and $B \in \mathcal{P}^s$ then $A \otimes^+ B \in \mathcal{P}^s$
- If $A \in \mathcal{P}^s$ then $\mu X.A \in \mathcal{P}^s$

The condition on B in the case of $\oplus^+$ admits data patterns which are not steady, possibly with free variables, but ensuring the preservation of purity, i.e., type safety; the basis will come from side A. We will prove (§ 4.3) that behaviours interpreting steady data patterns are pure, thus in particular a data pattern of the form $\mu X.A$ is steady if the free variables of A all appear on the same side of a $\oplus^+$ and under the scope of no other μ (since purity is stable under $\downarrow, \uparrow, \oplus, \otimes$). We claim that steady data patterns can represent every type of finite data.

▶ **Definition 28.** A **data behaviour** is the interpretation of a closed steady data pattern.

4.2 Internal Completeness for Infinite Union

Our main result is an internal completeness theorem, stating that an infinite union of *simple* regular behaviours with increasingly large incarnations is a behaviour: $\perp\perp$-closure is useless.

▶ **Definition 29.**

- A **slice** is a design in which all negative subdesigns are either Ω^- or of the form $a(\overrightarrow{x}).\mathfrak{p}_a$, i.e., at most unary branching. $\mathfrak{c}$ is a **slice of** $\mathfrak{d}$ if $\mathfrak{c}$ is a slice and $\mathfrak{c} \sqsubseteq \mathfrak{d}$. A slice $\mathfrak{c}$ of $\mathfrak{d}$ is **maximal** if for any slice $\mathfrak{c}'$ of $\mathfrak{d}$ such that $\mathfrak{c} \sqsubseteq \mathfrak{c}'$, we have $\mathfrak{c} = \mathfrak{c}'$.
- A behaviour $\mathbf{B}$ is **simple** if for every design $\mathfrak{d} \in |\mathbf{B}|$:
  1. $\mathfrak{d}$ has a finite number of maximal slices, and
  2. every positive action of $\mathfrak{d}$ is justified by the immediate previous negative action.

Condition (2) of simplicity ensures that, given $\mathfrak{d} \in |\mathbf{B}|$ and a slice $\mathfrak{c} \sqsubseteq \mathfrak{d}$, one can find a path of $\mathfrak{c}$ containing all the positive proper actions of $\mathfrak{c}$ until a given depth; thus by condition (1), there exists $k \in \mathbb{N}$ depending only on $\mathfrak{d}$ such that k paths can do the same in $\mathfrak{d}$.

Now suppose $(\mathbf{A}_n)_{n\in\mathbb{N}}$ is an infinite sequence of simple regular behaviours such that for all $n \in \mathbb{N}$, $|\mathbf{A}_n| \subseteq |\mathbf{A}_{n+1}|$ (in particular we have $\mathbf{A}_n \subseteq \mathbf{A}_{n+1}$).

▶ **Theorem 30.** *The set $\bigcup_{n\in\mathbb{N}} \mathbf{A}_n$ is a behaviour.*

A union of behaviours is not a behaviour in general. In particular, counterexamples are easily found if releasing either the inclusion of incarnations or the simplicity condition. Moreover, our proof for this theorem relies strongly on regularity. Under the same hypotheses we can prove $V_{\bigcup_{n\in\mathbb{N}} \mathbf{A}_n} = \bigcup_{n\in\mathbb{N}} V_{\mathbf{A}_n}$ and $|\bigcup_{n\in\mathbb{N}} \mathbf{A}_n| = \bigcup_{n\in\mathbb{N}} |\mathbf{A}_n|$, hence the following corollary.

▶ **Corollary 31.**

- *$\bigcup_{n\in\mathbb{N}} \mathbf{A}_n$ is simple and regular;*
- *if moreover all the $\mathbf{A}_n$ are pure then $\bigcup_{n\in\mathbb{N}} \mathbf{A}_n$ is pure.*

4.3 Regularity and Purity of Data

The goal of this subsection is to show that the interpretation of data patterns of the form $\mu X.A$ can be expressed as an infinite union of behaviours $(\mathbf{A}_n)_{n\in\mathbb{N}}$ satisfying the hypotheses of Theorem 30, in order to deduce regularity and purity. We will call an environment σ simple if its image contains only simple behaviours.

▶ **Lemma 32.** *For all $A \in \mathcal{P}$, $X \in \mathcal{V}$, $\sigma : \mathrm{FV}(A) \setminus \{X\} \to \mathcal{B}^+$ and $n \in \mathbb{N}$ we have*

$$|(\phi^A_\sigma)^n(\maltese)| \subseteq |(\phi^A_\sigma)^{n+1}(\maltese)| \,.$$

▶ **Proposition 33.** *For all $A \in \mathcal{P}$ and simple regular environment σ, $[\![A]\!]^\sigma$ is simple regular.*

Proof. By induction on data patterns. If $A = X$ or $A = a$ the conclusion is immediate. If $A = A_1 \oplus^+ A_2$ or $A = A_1 \otimes^+ A_2$ then regularity comes from Proposition 18, and simplicity is easy since the structure of the designs in $[\![A]\!]^\sigma$ is given by internal completeness for the logical connectives (Theorem 8). So suppose $A = \mu X.A_0$. By induction hypothesis, for every simple regular behaviour $\mathbf{P} \in \mathcal{B}^+$ we have $\phi_\sigma^{A_0}(\mathbf{P}) = [\![A_0]\!]^{\sigma, X \mapsto \mathbf{P}}$ simple regular. From this, it is straightforward to show by induction that for every $n \in \mathbb{N}$, $(\phi_\sigma^{A_0})^n(\maltese)$ is simple regular. Moreover, for every $n \in \mathbb{N}$ we have $|(\phi_\sigma^{A_0})^n(\maltese)| \subseteq |(\phi_\sigma^{A_0})^{n+1}(\maltese)|$ by Lemma 32, thus by Corollary 26 and Theorem 30, $[\![\mu X.A_0]\!]^\sigma = \bigvee_{n\in\mathbb{N}}(\phi_\sigma^{A})^n(\maltese) = (\bigcup_{n\in\mathbb{N}}(\phi_\sigma^{A_0})^n(\maltese))^{\perp\perp} = \bigcup_{n\in\mathbb{N}}(\phi_\sigma^{A_0})^n(\maltese)$. Consequently, by Corollary 31, $[\![\mu X.A_0]\!]^\sigma$ is simple regular. ◀

Remark that we have proved at the same time, using Theorem 30, that behaviours interpreting data patterns $\mu X.A$ admit an explicit construction:

▶ **Proposition 34.** *If $A \in \mathcal{P}$, $X \in \mathcal{V}$, and $\sigma : \mathrm{FV}(A) \setminus X \to \mathcal{B}^+$ is simple regular,*

$$[\![\mu X.A]\!]^\sigma = \bigcup_{n\in\mathbb{N}} (\phi_\sigma^A)^n(\maltese)$$

▶ **Corollary 35.** *Data behaviours are regular.*

We now move on to proving purity. The proof that the interpretation of a steady data pattern A is pure relies on the existence of a basis for A (Lemma 37). Let us first widen (to $\maltese$-free paths) and express in a different way (for $\maltese$-ended paths) the notion of extensible visitable path.

▶ **Definition 36.** Let $\mathbf{B}$ be a behaviour.

- A $\maltese$-free path $\mathsf{s} \in V_\mathbf{B}$ is **extensible** if there exists $\mathsf{t} \in V_\mathbf{B}$ of which s is a strict prefix.
- A $\maltese$-ended path $\mathsf{s}\maltese \in V_\mathbf{B}$ is **extensible** if there exists a positive action κ^+ and $\mathsf{t} \in V_\mathbf{B}$ of which $\mathsf{s}\kappa^+$ is a prefix.

Write $V_\mathbf{B}^{max}$ for the set of maximal, i.e., non extensible, visitable paths of $\mathbf{B}$.

▶ **Lemma 37.** *Every steady data pattern $A \in \mathcal{P}^s$ has a basis, i.e., a simple regular behaviour $\mathbf{B}$ such that for all simple regular environment σ we have*

- $\mathbf{B} \subseteq [\![A]\!]^\sigma$,
- *for every path $\mathsf{s} \in V_\mathbf{B}$, there exists $\mathsf{t} \in V_\mathbf{B}^{max}$ $\maltese$-free extending s (in particular $\mathbf{B}$ pure),*
- $V_\mathbf{B}^{max} \subseteq V_{[\![A]\!]^\sigma}^{max}$.

Proof (Idea). If $A = a$, a basis is $\mathbf{C}_a$. If $A = A_1 \oplus^+ A_2$, and A_i is steady with basis $\mathbf{B}_i$, then $\otimes_i{\uparrow}\mathbf{B}_i := \iota_i\langle{\uparrow}\mathbf{B}_i\rangle$ is a basis for A. If $A = A_1 \otimes^+ A_2$, a basis is $\mathbf{B}_1 \otimes^+ \mathbf{B}_2$ where $\mathbf{B}_1$ and $\mathbf{B}_2$ are basis of A_1 and A_2 respectively. If $A = \mu X.A_0$, its basis is the same as A_0. ◀

▶ **Proposition 38.** *If $A \in \mathcal{P}^s$ of basis $\mathbf{B}$, $X \in \mathcal{V}$, and $\sigma : \mathrm{FV}(A) \setminus X \to \mathcal{B}^+$ simple regular,*

$$[\![\mu X.A]\!]^\sigma = \bigcup_{n\in\mathbb{N}} (\phi_\sigma^A)^n(\mathbf{B})$$

Proof. Since $\mathbf{B}$ is a basis for A we have $\maltese \subseteq \mathbf{B} \subseteq [\![A]\!]^{\sigma, X \to \maltese} = \phi_\sigma^A(\maltese)$. The Scott-continuity of the function ϕ_σ^A implies that it is increasing, thus $(\phi_\sigma^A)^n(\maltese) \subseteq (\phi_\sigma^A)^n(\mathbf{B}) \subseteq (\phi_\sigma^A)^{n+1}(\maltese)$ for all $n \in \mathbb{N}$. Hence $[\![A]\!]^\sigma = \bigcup_{n\in\mathbb{N}}(\phi_\sigma^A)^n(\maltese) = \bigcup_{n\in\mathbb{N}}(\phi_\sigma^A)^n(\mathbf{B})$. ◀

▶ **Proposition 39.** *For all $A \in \mathcal{P}^s$ and simple regular pure environment σ, $[\![A]\!]^\sigma$ is pure.*

Proof. By induction on A. The base cases are immediate and the connective cases are solved using Proposition 19. Suppose now $A = \mu X.A_0$, where A_0 is steady with basis $\mathbf{B}_0$. We have $[\![A]\!]^\sigma = \bigcup_{n\in\mathbb{N}}(\phi_\sigma^{A_0})^n(\mathbf{B}_0)$ by Proposition 38, let us prove it satisfies the hypotheses needed to apply Corollary 31(2). By induction hypothesis and Proposition 33, for every simple, regular and pure behaviour $\mathbf{P} \in \mathcal{B}^+$ we have $\phi_\sigma^{A_0}(\mathbf{P}) = [\![A_0]\!]^{\sigma, X\mapsto \mathbf{P}}$ simple, regular and pure, hence it is easy to show by induction that for every $n \in \mathbb{N}$, $(\phi_\sigma^{A_0})^n(\mathbf{B}_0)$ is as well. Moreover, for every $n \in \mathbb{N}$ we prove that $|(\phi_\sigma^{A_0})^n(\mathbf{B}_0)| \subseteq |(\phi_\sigma^{A_0})^{n+1}(\mathbf{B}_0)|$ similarly to Lemma 32, replacing ✠ by the basis $\mathbf{B}_0$. Finally, by Corollary 31, $[\![A]\!]^\sigma$ is pure. ◀

▶ **Corollary 40.** *Data behaviours are pure.*

▶ Remark. Although here the focus is on the interpretation of data patterns, we should say a word about the interpretation of (polarised) μMALL formulas, which are a bit more general. These formulas are generated by:

$$\begin{array}{lll} P, Q & ::= & X_P \;\mid\; X_N^\perp \;\mid\; 1 \;\mid\; 0 \;\mid\; M \oplus N \;\mid\; M \otimes N \;\mid\; \downarrow N \;\mid\; \mu X.P \\ M, N & ::= & P^\perp \end{array}$$

where the usual involutive negation hides the negative connectives and constants, through the dualities $1/\perp$, $0/\top$, $\oplus/\&$, $\otimes/\parr$, $\downarrow/\uparrow$, μ/ν . The interpretation as ludics behaviours, given in [2], is as follows: 1 is interpreted as a constant behaviour $\mathbf{C}_a$, 0 is the daimon ✠, the positive connectives match their ludics counterparts, μ is interpreted as the least fixed point of a function ϕ_σ^A similarly to data patterns, and the negation corresponds to the orthogonal. Since in ludics constants and ✠ are regular, and since regularity is preserved by the connectives (Proposition 18) and by orthogonality, the only thing we need in order to prove that all the behaviours interpreting μMALL formulas are regular is a generalisation of regularity stability under fixed points (for now we only have it in our particular case: Corollary 31 together with Proposition 34).

Note however that interpretations of μMALL formulas are not all pure. Indeed, as we will see in next section, orthogonality (introduced through the connective $\multimap$) does not preserve purity in general.

5 Functional Types

In this section we define *functional behaviours* which combine data behaviours with the connective $\multimap$. A behaviour of the form $\mathbf{N} \multimap \mathbf{P}$ is the set of designs such that, when interacting with a design of type $\mathbf{N}$, outputs a design of type $\mathbf{P}$; this is exactly the meaning of its definition $\mathbf{N} \multimap \mathbf{P} := (\mathbf{N} \otimes \mathbf{P}^\perp)^\perp$. We prove that some particular higher-order functional types – where functions are taken as arguments, typically $(A \multimap B) \multimap C$ – are exactly those who fail at being pure, and we interpret this result from a computational point of view.

5.1 Where Impurity Arises

We have proved that data behaviours are regular and pure. However, if we introduce functional behaviours with the connective $\multimap$, purity does not hold in general. Proposition 42 indicates that a weaker property, quasi-purity, holds for functional types, and Proposition 43 identifies exactly the cases where purity fails.

Let us write $\mathcal{D}$ for the set of data behaviours.

▶ **Definition 41.** A **functional behaviour** is a behaviour inductively generated by the grammar below, where $\mathbf{P} \multimap^+ \mathbf{Q}$ stands for $\downarrow((\uparrow\mathbf{P}) \multimap \mathbf{Q})$.

$$\mathbf{P}, \mathbf{Q} ::= \mathbf{P}_0 \in \mathcal{D} \quad | \quad \mathbf{P} \oplus^+ \mathbf{Q} \quad | \quad \mathbf{P} \otimes^+ \mathbf{Q} \quad | \quad \mathbf{P} \multimap^+ \mathbf{Q}\,.$$

From Propositions 18, 19 and 21 we easily deduce the following result.

▶ **Proposition 42.** *Functional behaviours are regular and quasi-pure.*

For next proposition, consider **contexts** defined inductively as follows (where $\mathbf{P}$ is a functional behaviour):

$$\mathcal{C} ::= [\,] \quad | \quad \mathcal{C} \oplus^+ \mathbf{P} \quad | \quad \mathbf{P} \oplus^+ \mathcal{C} \quad | \quad \mathcal{C} \otimes^+ \mathbf{P} \quad | \quad \mathbf{P} \otimes^+ \mathcal{C} \quad | \quad \mathbf{P} \multimap^+ \mathcal{C}\,.$$

▶ **Proposition 43.** *A functional behaviour* $\mathbf{P}$ *is impure if and only if there exist contexts* $\mathcal{C}_1, \mathcal{C}_2$ *and functional behaviours* $\mathbf{Q}_1, \mathbf{Q}_2, \mathbf{R}$ *with* $\mathbf{R} \notin \mathrm{Const}$ *such that*

$$\mathbf{P} = \mathcal{C}_1[\ \mathcal{C}_2[\mathbf{Q}_1 \multimap^+ \mathbf{Q}_2] \multimap^+ \mathbf{R}\ ]\,.$$

5.2 Example and Discussion

Proposition 43 states that a functional behaviour which takes functions as argument is not pure: some of its visitable paths end with a daimon ✠, and there is no possibility to extend them. In terms of proof-search, playing the daimon is like giving up; on a computational point of view, the daimon appearing at the end of an interaction expresses the sudden interruption of the computation. In order to understand why such an interruption can occur in the specific case of higher-order functions, consider the following example which illustrates the proposition.

▶ **Example 44.** Let $\mathbf{Q}_1, \mathbf{Q}_2, \mathbf{1}$ be functional behaviours, with $\mathbf{1} \in \mathrm{Const}$. Define $\mathbf{Bool} = \mathbf{1} \oplus^+ \mathbf{1}$ and consider the behaviour $\mathbf{P} = (\mathbf{Q}_1 \multimap^+ \mathbf{Q}_2) \multimap^+ \mathbf{Bool}$: this is a type of functions which take a function as argument and output a boolean. Let $\alpha_1, \alpha_2, \beta$ be respectively the first positive action of the designs of $\mathbf{Q}_1, \mathbf{Q}_2, \mathbf{1}$. It is possible to exhibit a design $\mathfrak{p} \in \mathbf{P}$ and a design $\mathfrak{n} \in \mathbf{P}^\perp$ such that the visitable path $s = \langle \mathfrak{p} \leftarrow \mathfrak{n} \rangle$ is ✠-ended and maximal in $V_\mathbf{P}$, in other words s is a witness of the impurity of $\mathbf{P}$. The path s contains the actions α_1 and $\overline{\alpha_2}$ in such a way that it cannot be extended with β without breaking the P-visibility condition, and there is no other available action in designs of $\mathbf{P}$ to extend it. Reproducing the designs $\mathfrak{p}$ and $\mathfrak{n}$ and the path s here would be of little interest since those objects are too large to be easily readable (s visits the entire design $\mathfrak{p}$, which contains 11 actions). We however give an intuition in the style of game semantics: Fig. 2 represents s as a legal play in a strategy of type $\mathbf{P} = (\mathbf{Q}_1 \multimap^+ \mathbf{Q}_2) \multimap^+ \mathbf{Bool}$ (note that only one "side" $\oplus_1 \uparrow \mathbf{1}$ of $\mathbf{Bool}$ is represented, corresponding for example to `True`, because we cannot play in both sides). This analogy is informal, it should stand as an intuition rather than as a precise correspondence with ludics; for instance, and contrary to the way it is presented in game semantics, the questions are asked on the connectives, while the answers are given in the sub-types of $\mathbf{P}$. On the right are given the actions in s corresponding to the moves played. The important thing to remark is the following: if a move b corresponding to action β were played instead of ✠ at the end of this play, it would break the P-visibility of the strategy, since this move would be justified by move $q_\uparrow$.

The computational interpretation of the ✠-ended interaction between $\mathfrak{p}$ and $\mathfrak{n}$ is the following: a program p of type $\mathbf{P}$ launches a child process p' to compute the argument of type $\mathbf{Q}_1 \to \mathbf{Q}_2$, but p starts to give a result in $\mathbf{Bool}$ before the execution of p' terminates,

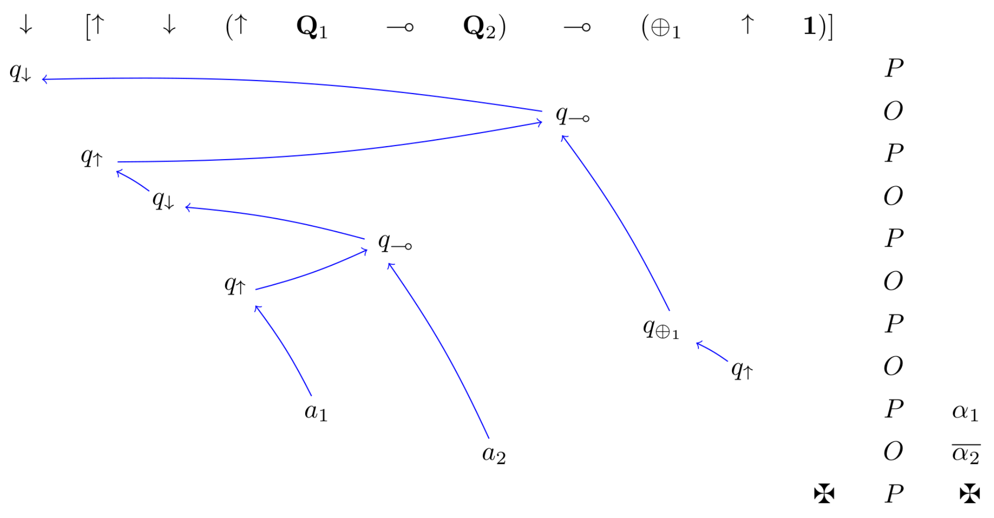

Figure 2 Representation of path $\mathfrak{s}$ from Example 44 in the style of a legal play.

leading to a situation where p cannot compute the whole data in **Bool**. The interaction outputs ✠, i.e., the answer given in **Bool** by p is incomplete.

Moreover by Proposition 42 functional behaviours are quasi-pure, therefore the maximal ✠-ended visitable paths are necessarily not well-bracketed. This is indeed the case of $\mathfrak{s}$: remark for example that the move $q_{\oplus_1}$ appears between a_1 and its justification $q_\uparrow$ in the sequence, but $q_{\oplus_1}$ is not hereditarily justified by $q_\uparrow$. In HO games, well-bracketedness is a well studied notion, and relaxing it introduces control operators in program. If we extend such an argument to ludics, this would mean that the appearance of ✠ in the execution of higher-order functions can only happen in the case of programs with control operators such as *jumps*, i.e. programs which are not purely functional.

6 Conclusion

This article is a contribution to the exploration of the behaviours of linear ludics in a computational perspective. Our focus is on the behaviours representing data types and functional types. Inductive data types are interpreted using the logical connectives constructions and a least fixed point operation. Adopting a constructive approach, we provide an internal completeness result for fixed points, which unveils the structure of data behaviours. This leads us to proving that such behaviours are regular – the key notion for the characterisation of MALL in ludics – and pure – that is, type safe. But behaviours interpreting types of functions taking functions as argument are impure; for well-bracketed interactions, corresponding to the evaluation of purely functional programs, safety is however guaranteed.

Further Work. Two directions for future research arise naturally:

- Extending our study to greatest fixed points $\nu X.A$, i.e., coinduction, is the next objective. Knaster–Tarski ensures that such greatest fixed point behaviours exist [2], but Kleene fixed point theorem does not apply here, hence we cannot find an explicit form for coinductive behaviours the same way we did for the inductive ones. However it is intuitively clear that, compared to least fixed points, greatest ones add the infinite “limit” designs in (the incarnation of) behaviours. For example, if $\mathbb{N}at_\omega = \nu X.(1 \oplus X)$ then we should have $|[\![\mathbb{N}at_\omega]\!]| = |[\![\mathbb{N}at]\!]| \cup \{\mathfrak{d}_\omega\}$ where $\mathfrak{d}_\omega = \mathrm{succ}(\mathfrak{d}_\omega) = x_0|\iota_2\langle\uparrow(x).{\mathfrak{d}_\omega}^x\rangle$.

- Another direction would be to get a complete characterisation of μMALL in ludics, by proving that a behaviour is regular – and possibly satisfying a supplementary condition – if and only if it is the denotation of a μMALL formula.

Acknowledgements. I thank Claudia Faggian, Christophe Fouqueré, Thomas Seiller and the anonymous referees for their wise and helpful comments.

References

1 David Baelde. Least and greatest fixed points in linear logic. *ACM Trans. Comput. Logic*, 13(1):2:1–2:44, January 2012. doi:10.1145/2071368.2071370.

2 David Baelde, Amina Doumane, and Alexis Saurin. Least and greatest fixed points in ludics. In Stephan Kreuzer, editor, *Proceedings of the 24th Annual EACSL Conference on Computer Science Logic (CSL'15)*, pages 549–566, Berlin, Germany, September 2015. doi:10.4230/LIPIcs.CSL.2015.549.

3 Michele Basaldella and Claudia Faggian. Ludics with repetitions (exponentials, interactive types and completeness). *Logical Methods in Computer Science*, 7(2), 2011.

4 Pierre Clairambault. Least and greatest fixpoints in game semantics. In *Foundations of Software Science and Computational Structures, 12th International Conference (FOSSACS 2009). Proceedings*, pages 16–31, 2009.

5 Pierre-Louis Curien. Introduction to linear logic and ludics, part II. *CoRR*, abs/cs/0501039, 2005.

6 Claudia Faggian and Martin Hyland. Designs, disputes and strategies. In *CSL*, pages 442–457, 2002.

7 Christophe Fouqueré and Myriam Quatrini. Incarnation in ludics and maximal cliques of paths. *Logical Methods in Computer Sciences*, 9(4), 2013.

8 Christophe Fouqueré and Myriam Quatrini. Study of behaviours via visitable paths. *CoRR*, abs/1403.3772v3, 2016. URL: https://arxiv.org/abs/1403.3772v3.

9 Jean-Yves Girard. Geometry of interaction. I. Interpretation of system f. In *Proc. Logic Colloquium 1988*, pages 221–260, North-Holland, Amsterdam, 1989.

10 Jean-Yves Girard. Locus solum: From the rules of logic to the logic of rules. *Mathematical Structures in Computer Science*, 11(3):301–506, 2001.

11 Martin Hyland and Luke Ong. On full abstraction for PCF: I, II, and III. *Information and Computation*, 163(2):285–408, 2000.

12 Jean-Louis Krivine. Realizability in classical logic. *Panoramas et synthèses*, 27:197–229, 2009. URL: https://hal.archives-ouvertes.fr/hal-00154500.

13 Paul-André Melliès and Jerome Vouillon. Recursive polymorphic types and parametricity in an operational framework. In *20th IEEE Symposium on Logic in Computer Science (LICS 2005), 26-29 June 2005, Chicago, IL, USA, Proceedings*, pages 82–91, 2005. doi:10.1109/LICS.2005.42.

14 Robin Milner. A theory of type polymorphism in programming. *Journal of Computer and System Sciences*, 17:348–375, 1978.

15 Eugenia Sironi. *Types in Ludics.* PhD thesis, Aix-Marseille Université, January 2015.

16 Kazushige Terui. Computational ludics. *Theor. Comput. Sci.*, 412(20):2048–2071, 2011.

A Technical Appendix

This appendix presents the proof of Theorem 30, which requires first some preliminaries.

A.1 Observational Ordering and Monotonicity

We consider the **observational ordering** $\preceq$ over designs: $\mathfrak{d}' \preceq \mathfrak{d}$ if $\mathfrak{d}$ can be obtained from $\mathfrak{d}'$ by substituting:

- positive subdesigns for some occurrences of Ω.
- ✠ for some positive subdesigns.

Remark in particular that for all positive designs $\mathfrak{p}$ and $\mathfrak{p}'$, we have $\Omega \preceq \mathfrak{p} \preceq$ ✠, and if $\mathfrak{p} \sqsubseteq \mathfrak{p}'$ then $\mathfrak{p} \preceq \mathfrak{p}'$. We can now state the *monotonicity* theorem, an important result of ludics. A proof of the theorem formulated in this form is found in [16].

▶ **Theorem 45** (Monotonicity).

- *If $\mathfrak{d} \preceq \mathfrak{e}$ and $\mathfrak{m} \preceq \mathfrak{n}$, then $\mathfrak{d}[\mathfrak{m}/x] \preceq \mathfrak{e}[\mathfrak{n}/x]$.*
- *If $\mathfrak{d} \preceq \mathfrak{e}$ then $(\!|\mathfrak{d}|\!) \preceq (\!|\mathfrak{e}|\!)$.*

This means that the relation $\preceq$ compares the likelihood of convergence: if $\mathfrak{d} \perp \mathfrak{e}$ and $\mathfrak{d} \preceq \mathfrak{d}'$ then $\mathfrak{d}' \perp \mathfrak{e}$. In particular, if $\mathbf{B}$ is a behaviour, if $\mathfrak{d} \in \mathbf{B}$ and $\mathfrak{d} \preceq \mathfrak{d}'$ then $\mathfrak{d}' \in \mathbf{B}$.

Remark the following important fact: given a path s of some design $\mathfrak{d}$, there is a unique design maximal for $\preceq$ such that s is a path of it. Indeed, this design $\ulcorner\!\ulcorner s \urcorner\!\urcorner^c$ is obtained from $\mathfrak{d}$ by replacing all positive subdesigns (possibly Ω) whose first positive action is not in s by ✠. Note that, actually, the design $\ulcorner\!\ulcorner s \urcorner\!\urcorner^c$ does not depend on $\mathfrak{d}$ but only on the path s.

▶ **Proposition 46.** *For every behaviour $\mathbf{B}$, if $s \in V_{\mathbf{B}}$ then $\ulcorner\!\ulcorner s \urcorner\!\urcorner^c \in \mathbf{B}$.*

A.2 More on Paths

Let $\mathbf{B}$ be a behaviour.

▶ **Lemma 47.** *If $\mathfrak{d} \in \mathbf{B}$ and $s \in V_{\mathbf{B}}$ is a path of $\mathfrak{d}$, then s is a path of $|\mathfrak{d}|$.*

▶ **Lemma 48.** *Let $s \in V_{\mathbf{B}}$. For every positive-ended (resp. negative-ended) prefix s' of s, we have $s' \in V_{\mathbf{B}}$ (resp. s'✠ $\in V_{\mathbf{B}}$).*

▶ **Lemma 49.** *Let $s \in V_{\mathbf{B}}$. For every prefix $s'\kappa^-$ of s and every $\mathfrak{d} \in \mathbf{B}$ such that s' is a path of $\mathfrak{d}$, $s'\kappa^-$ is a prefix of a path of $\mathfrak{d}$.*

A.3 An Alternative Definition of Regularity

Define the **anti-shuffle** ($\sqcap\!\sqcap$) as the dual operation of shuffle, that is:

- $s \sqcap\!\sqcap t = \widetilde{\tilde{s} \sqcup\!\sqcup \tilde{t}}$ if s and t are paths of same polarity;
- $S \sqcap\!\sqcap T = \widetilde{\tilde{S} \sqcup\!\sqcup \tilde{T}}$ if S and T are sets of paths of same polarity.

▶ **Definition 50.**

- A **trivial view** is an aj-sequence such that each proper action except the first one is justified by the immediate previous action. In other words, it is a view such that its dual is a view as well.

- The **trivial view of** an aj-sequence is defined inductively by:

$$\begin{array}{rll} \langle\epsilon\rangle = \epsilon & & \text{empty sequence} \\ \langle s✠\rangle = \langle s\rangle✠ & & \\ \langle s\kappa\rangle = \kappa & & \text{if } \kappa \neq ✠ \text{ initial} \\ \langle s\kappa\rangle = \langle s_0\rangle\kappa & & \text{if } \kappa \neq ✠ \text{ justified, where } s_0 \text{ prefix of } s \text{ ending on } \mathit{just}(\kappa) \end{array}$$

  We also write $\langle\kappa\rangle_s$ (or even $\langle\kappa\rangle$) instead of $\langle s'\kappa\rangle$ when $s'\kappa$ is a prefix of s.
- **Trivial views of a design** $\mathfrak{d}$ are the trivial views of its paths (or of its views). In particular, ϵ is a trivial view of negative designs only.
- Trivial views of designs in $|\mathbf{B}|$ are called **trivial views of B**.

▶ **Lemma 51.**
1. *Every view is in the anti-shuffle of trivial views.*
2. *Every path is in the shuffle of views.*

▶ Remark. Following previous result, note that every view (resp. path) of a design $\mathfrak{d}$ is in the anti-shuffle of trivial views (resp. in the shuffle of views) of $\mathfrak{d}$.

▶ **Proposition 52.** $\mathbf{B}$ *is regular if and only if the following conditions hold:*
- *the positive-ended trivial views of* $\mathbf{B}$ *are visitable in* $\mathbf{B}$*,*
- $V_{\mathbf{B}}$ *and* $V_{\mathbf{B}^\perp}$ *are stable under* $\sqcup$ *(i.e.,* $V_{\mathbf{B}}$ *is stable under* $\sqcup$ *and* $\sqcap$*).*

A.4 Proof of Theorem 30

Before proving Theorem 30 we need some lemmas. Suppose $(\mathbf{A}_n)_{n\in\mathbb{N}}$ is an infinite sequence of regular behaviours such that for all $n \in \mathbb{N}$, $|\mathbf{A}_n| \subseteq |\mathbf{A}_{n+1}|$; the simplicity hypothesis is not needed for now. Let us note $\mathbf{A} = \bigcup_{n\in\mathbb{N}} \mathbf{A}_n$. Notice that the definition of visitable paths can harmlessly be extended to any set E of designs of same polarity, even if it is not a behaviour; the same applies to the definition of incarnation, provided that E satisfies the following: if $\mathfrak{d}, \mathfrak{e}_1, \mathfrak{e}_2 \in E$ are cut-free designs such that $\mathfrak{e}_1 \sqsubseteq \mathfrak{d}$ and $\mathfrak{e}_2 \sqsubseteq \mathfrak{d}$ then there exists $\mathfrak{e} \in E$ cut-free such that $\mathfrak{e} \sqsubseteq \mathfrak{e}_1$ and $\mathfrak{e} \sqsubseteq \mathfrak{e}_2$. In particular, as a union of behaviours, $\mathbf{A}$ satisfies this condition.

▶ **Lemma 53.**
1. $\forall n \in \mathbb{N}$*,* $V_{\mathbf{A}_n} \subseteq V_{\mathbf{A}_{n+1}}$*.*
2. $V_{\bigcup_{n\in\mathbb{N}} \mathbf{A}_n} = \bigcup_{n\in\mathbb{N}} V_{\mathbf{A}_n}$*.*
3. $|\bigcup_{n\in\mathbb{N}} \mathbf{A}_n| = \bigcup_{n\in\mathbb{N}} |\mathbf{A}_n|$*.*

Proof.
1. Fix n and let $s \in V_{A_n}$. There exist $\mathfrak{d} \in |\mathbf{A}_n|$ such that s is a path of $\mathfrak{d}$. Since $|\mathbf{A}_n| \subseteq |\mathbf{A}_{n+1}|$ we have $\mathfrak{d} \in |\mathbf{A}_{n+1}|$, thus by regularity of $\mathbf{A}_{n+1}$, $s \in V_{A_{n+1}}$.
2. ($\subseteq$) Let $s \in V_{\mathbf{A}}$. There exist $n \in \mathbb{N}$ and $\mathfrak{d} \in |\mathbf{A}_n|$ such that s is a path of $\mathfrak{d}$. By regularity of $\mathbf{A}_n$ we have $s \in V_{\mathbf{A}_n}$.
   ($\supseteq$) Let $m \in \mathbb{N}$ and $s \in V_{\mathbf{A}_m}$. For all $n \geq m$, $V_{\mathbf{A}_m} \subseteq V_{\mathbf{A}_n}$ by previous item, thus $s \in V_{\mathbf{A}_n}$. Hence if we take $\mathfrak{e} = \ulcorner\!\ulcorner\widetilde{s}\urcorner\!\urcorner^c$, we have $\mathfrak{e} \in \mathbf{A}_n{}^\perp$ for all $n \geq m$ by monotonicity. We deduce $\mathfrak{e} \in \bigcap_{n\geq m} \mathbf{A}_n{}^\perp = (\bigcup_{n\geq m} \mathbf{A}_n)^\perp = (\bigcup_{n\in\mathbb{N}} \mathbf{A}_n)^\perp = \mathbf{A}^\perp$. Let $\mathfrak{d} \in \mathbf{A}_m$ such that s is a path of $\mathfrak{d}$; we have $\mathfrak{d} \in \mathbf{A}$ and $\mathfrak{e} \in \mathbf{A}^\perp$, thus $\langle\mathfrak{d} \leftarrow \mathfrak{e}\rangle = s \in V_{\mathbf{A}}$.
3. ($\subseteq$) Let $\mathfrak{d}$ be cut-free and minimal for $\sqsubseteq$ in $\mathbf{A}$. There exists $m \in \mathbb{N}$ such that $\mathfrak{d} \in \mathbf{A}_m$. Thus $\mathfrak{d}$ is minimal for $\sqsubseteq$ in $\mathbf{A}_m$ otherwise it would not be minimal in $\mathbf{A}$, hence the result.
   ($\supseteq$) Let $m \in \mathbb{N}$, and let $\mathfrak{d} \in |\mathbf{A}_m|$. By hypothesis, $\mathfrak{d} \in |\mathbf{A}_n|$ for all $n \geq m$. Suppose $\mathfrak{d}$ is not in $|\mathbf{A}|$, so there exists $\mathfrak{d}' \in \mathbf{A}$ such that $\mathfrak{d}' \sqsubseteq \mathfrak{d}$ and $\mathfrak{d}' \neq \mathfrak{d}$. In this case, there exists $n \geq m$ such that $\mathfrak{d}' \in \mathbf{A}_n$, but this contradicts the fact that $\mathfrak{d} \in |\mathbf{A}_n|$. ◀

▶ **Lemma 54.** $V_{\bigcup_{n\in\mathbb{N}}\mathbf{A}_n} = \widetilde{V_{\bigcup_{n\in\mathbb{N}}\mathbf{A}_n{}^\perp}} = V_{(\bigcup_{n\in\mathbb{N}}\mathbf{A}_n)^{\perp\perp}}$.

Proof. In this proof we use the alternative definition of regularity (Proposition 52). We prove $V_{\mathbf{A}} = \widetilde{V_{\mathbf{A}^\perp}}$, and the result will follow from the fact that for any behaviour $\mathbf{B}$ (in particular if $\mathbf{B} = \mathbf{A}^{\perp\perp}$) we have $\widetilde{V_{\mathbf{B}^\perp}} = V_{\mathbf{B}}$. First note that the inclusion $V_{\mathbf{A}} \subseteq \widetilde{V_{\mathbf{A}^\perp}}$ is immediate.

Let $\mathscr{s} \in V_{\mathbf{A}^\perp}$ and let us show that $\widetilde{\mathscr{s}} \in V_{\mathbf{A}}$. Let $\mathfrak{e} \in |\mathbf{A}^\perp|$ such that $\mathscr{s}$ is a path of $\mathfrak{e}$. By Lemma 51 and the remark following it, $\mathscr{s}$ is in the shuffle of anti-shuffles of trivial views $\approx_1, \dots, \approx_k$ of $\mathbf{A}^\perp$. For every $i \leq k$, suppose $\approx_i = \langle \kappa_i \rangle$; necessarily, there exists a design $\mathfrak{d}_i \in \mathbf{A}$ such that κ_i occurs in $\langle \mathfrak{e} \leftarrow \mathfrak{d}_i \rangle$, i.e., such that $\approx_i$ is a subsequence of $\langle \mathfrak{e} \leftarrow \mathfrak{d}_i \rangle$, otherwise $\mathfrak{e}$ would not be in the incarnation of $\mathbf{A}^\perp$ (it would not be minimal). Let n be big enough such that $\mathfrak{d}_1, \dots, \mathfrak{d}_k \in \mathbf{A}_n$, and note that in particular $\mathfrak{e} \in \mathbf{A}_n{}^\perp$. For all i, $\widetilde{\approx_i}$ is a trivial view of $|\mathfrak{d}_i|_{\mathbf{A}_n}$, thus it is a trivial view of $\mathbf{A}_n$. By regularity of $\mathbf{A}_n$ we have $\widetilde{\approx_i} \in V_{\mathbf{A}_n}$. Since $\widetilde{\mathscr{s}}$ is in the anti-shuffle of shuffles of $\widetilde{\approx_1}, \dots, \widetilde{\approx_k}$, we have $\widetilde{\mathscr{s}} \in V_{\mathbf{A}_n}$ using regularity again. Therefore $\widetilde{\mathscr{s}} \in V_{\mathbf{A}}$ by Lemma 53. ◀

▶ **Lemma 55.** *$(\bigcup_{n\in\mathbb{N}}\mathbf{A}_n)^\perp$ and $(\bigcup_{n\in\mathbb{N}}\mathbf{A}_n)^{\perp\perp}$ are regular.*

Proof. Let us show $\mathbf{A}^\perp$ is regular using the equivalent definition (Proposition 52).

- Let $\approx$ be a trivial view of $\mathbf{A}^\perp$. By a similar argument as in the proof above, there exists $n \in \mathbb{N}$ such that $\widetilde{\approx}$ is a trivial view of $\mathbf{A}_n$, thus $\widetilde{\approx} \in V_{\mathbf{A}_n} \subseteq V_{\mathbf{A}}$. By Lemma 54 $\approx \in V_{\mathbf{A}^\perp}$.
- Let $\mathscr{s}, t \in V_{\mathbf{A}^\perp}$. By Lemma 54, $\widetilde{\mathscr{s}}, \widetilde{t} \in V_{\mathbf{A}}$. By Lemma 53(2), there exists $n \in \mathbb{N}$ such that $\widetilde{\mathscr{s}}, \widetilde{t} \in V_{\mathbf{A}_n}$, thus by regularity of $\mathbf{A}_n$ we have $\widetilde{\mathscr{s}} \sqcap\!\sqcap \widetilde{t}$, $\widetilde{\mathscr{s}} \sqcup\!\sqcup \widetilde{t} \subseteq V_{\mathbf{A}_n} \subseteq V_{\mathbf{A}}$, in other words $\widetilde{\mathscr{s} \sqcup\!\sqcup t}$, $\widetilde{\mathscr{s} \sqcap\!\sqcap t} \subseteq V_{\mathbf{A}}$. By Lemma 54 we deduce $\mathscr{s} \sqcup\!\sqcup t$, $\mathscr{s} \sqcap\!\sqcap t \subseteq V_{\mathbf{A}^\perp}$, hence $V_{\mathbf{A}^\perp}$ is stable under shuffle and anti-shuffle.

Finally $\mathbf{A}^\perp$ is regular. We deduce that $\mathbf{A}^{\perp\perp}$ is regular since regularity is stable under orthogonality. ◀

Let us introduce some more notions for next proof. An **∞-path** (resp. **∞-view**) is a finite or infinite sequence of actions satisfying all the conditions of the definition of path (resp. view) but the requirement of finiteness. In particular, a finite ∞-path (resp. ∞-view) is a path (resp. a view). An **∞-path** (resp. **∞-view**) **of** a design $\mathfrak{d}$ is such that any of its positive-ended prefix is a path (resp. a view) of $\mathfrak{d}$. We call **infinite chattering** a closed interaction which diverges because the computation never ends; note that infinite chattering occurs in the interaction between two atomic designs $\mathfrak{p}$ and $\mathfrak{n}$ if and only if there exists an infinite ∞-path $\mathscr{s}$ of $\mathfrak{p}$ such that $\widetilde{\mathscr{s}}$ is an ∞-path of $\mathfrak{n}$ (where, when $\mathscr{s}$ is infinite, $\widetilde{\mathscr{s}}$ is obtained from $\mathscr{s}$ by simply reversing the polarities of all the actions). Given an infinite ∞-path $\mathscr{s}$, the design $\ulcorner\!\ulcorner \mathscr{s} \urcorner\!\urcorner^c$ is constructed similarly to the case when $\mathscr{s}$ is finite (see §A.1).

For the proof of the theorem, suppose now that the behaviours $(\mathbf{A}_n,)_{n\in\mathbb{N}}$ are simple. Remark that the second condition of simplicity implies in particular that the dual of a path in a design of a simple behaviour is a view.

Proof of Theorem 30. We must show that $\mathbf{A}^{\perp\perp} \subseteq \mathbf{A}$ since the other inclusion is trivial. Remark the following: given designs $\mathfrak{d}$ and $\mathfrak{d}'$, if $\mathfrak{d} \in \mathbf{A}$ and $\mathfrak{d} \sqsubseteq \mathfrak{d}'$ then $\mathfrak{d}' \in \mathbf{A}$. Indeed, if $\mathfrak{d} \in \mathbf{A}$ then there exists $n \in \mathbb{N}$ such that $\mathfrak{d} \in \mathbf{A}_n$; if moreover $\mathfrak{d} \sqsubseteq \mathfrak{d}'$ then in particular $\mathfrak{d} \preceq \mathfrak{d}'$, and by monotonicity $\mathfrak{d}' \in \mathbf{A}_n$, hence $\mathfrak{d}' \in \mathbf{A}$. Thus it is sufficient to show $|\mathbf{A}^{\perp\perp}| \subseteq \mathbf{A}$ since for every $\mathfrak{d}' \in \mathbf{A}^{\perp\perp}$ we have $|\mathfrak{d}'| \in |\mathbf{A}^{\perp\perp}|$ and $|\mathfrak{d}'| \sqsubseteq \mathfrak{d}'$.

So let $\mathfrak{d} \in |\mathbf{A}^{\perp\perp}|$ and suppose $\mathfrak{d} \notin \mathbf{A}$. First note the following: by Lemmas 54 and 55, every path $\mathscr{s}$ of $\mathfrak{d}$ is in $V_{\mathbf{A}^{\perp\perp}} = V_{\mathbf{A}}$, thus there exists $\mathfrak{d}' \in |\mathbf{A}|$ containing $\mathscr{s}$. We explore separately the possible cases, and show how they all lead to a contradiction.

If $\mathfrak{d}$ has an infinite number of maximal slices then:

- Either there exists a negative subdesign $\mathfrak{n} = \sum_{a\in\mathcal{S}} a(\overrightarrow{x^a}).\mathfrak{p}_a$ of $\mathfrak{d}$ for which there is an infinity of names $a \in \mathcal{A}$ such that $\mathfrak{p}_a \neq \Omega$. In this case, let $\gtrapprox$ be the view of $\mathfrak{d}$ such that for every action κ^- among the first ones of $\mathfrak{n}$, $\gtrapprox\kappa^-$ is the prefix of a view of $\mathfrak{d}$. All such sequences $\gtrapprox\kappa^-$ being prefixes of paths of $\mathfrak{d}$, we deduce by regularity of $\mathbf{A}^{\perp\perp}$ and using Lemma 48 that $\gtrapprox\kappa^-\maltese \in V_{\mathbf{A}^{\perp\perp}}$. Let $\mathfrak{d}' \in |\mathbf{A}|$ be such that $\gtrapprox$ is a view of $\mathfrak{d}'$. Since $\mathfrak{d}'$ is also in $\mathbf{A}^{\perp\perp}$, we deduce by Lemma 49 that for every action κ^- among the first ones of $\mathfrak{n}$, $\gtrapprox\kappa^-$ is the prefix of a view of $\mathfrak{d}'$. Thus $\mathfrak{d}'$ has an infinite number of slices: contradiction.
- Or we can find an infinite ∞-view $\gtrapprox = (\kappa_0^-)\kappa_1^+\kappa_1^-\kappa_2^+\kappa_1^-\kappa_3^+\kappa_3^- \dots$ of $\mathfrak{d}$ (the first action κ_0^- being optional depending on the polarity of $\mathfrak{d}$) satisfying the following: there is an infinity of $i \in \mathbb{N}$ such than κ_i^- is one of the first actions of a negative subdesign $\sum_{a\in\mathcal{S}} a(\overrightarrow{x^a}).\mathfrak{p}_a$ of $\mathfrak{d}$ with at least two names $a \in \mathcal{A}$ such that $\mathfrak{p}_a \neq \Omega$. Let $\gtrapprox_i$ be the prefix of $\gtrapprox$ ending on κ_i^+. There is no design $\mathfrak{d}' \in |\mathbf{A}|$ containing $\gtrapprox$, indeed: in this case, for all i and all negative action κ^- such that $\gtrapprox_i\kappa^-$ is a prefix of a view of $\mathfrak{d}$, $\gtrapprox_i\kappa^-$ would be a prefix of a view of $\mathfrak{d}'$ by Lemma 49, thus $\mathfrak{d}'$ would have an infinite number of slices, which is impossible since the $\mathbf{A}_n$ are simple. Thus consider $\mathfrak{e} = \ulcorner\widetilde{\gtrapprox}\urcorner^c$: since all the $\gtrapprox_i$ are views of designs in $|\mathbf{A}| = \bigcup_{n\in\mathbb{N}} |\mathbf{A}_n|$ and since the $\mathbf{A}_n$ are simple, the sequences $\widetilde{\gtrapprox_i}$ are views, thus $\widetilde{\gtrapprox}$ is an ∞-view. Therefore an interaction between a design $\mathfrak{d}' \in \mathbf{A}$ and $\mathfrak{e}$ necessarily eventually converges by reaching a daimon of $\mathfrak{e}$, indeed: infinite chattering is impossible since we cannot follow $\gtrapprox$ forever, and interaction cannot fail after following a finite portion of $\gtrapprox$ since those finite portions $\gtrapprox_i$ are in $V_{\mathbf{A}}$. Hence $\mathfrak{e} \in \mathbf{A}^\perp$. But $\mathfrak{d} \not\perp \mathfrak{e}$, because of infinite chattering following $\gtrapprox$. Contradiction.

If $\mathfrak{d}$ has a finite number of maximal slices $\mathfrak{c}_1, \dots, \mathfrak{c}_k$ then for every $i \leq k$ there exist an ∞-path s_i that visit all the positive proper actions of $\mathfrak{c}_i$. Indeed, any (either infinite or positive-ended) sequence s of proper actions in a slice $\mathfrak{c} \sqsubseteq \mathfrak{d}$, without repetition, such that polarities alternate and the views of prefixes of s are views of $\mathfrak{c}$, is an ∞-path:

- (Linearity) is ensured by the fact that we are in only one slice,
- (O-visibility) is satisfied since positive actions of $\mathfrak{d}$, thus also of $\mathfrak{c}$, are justified by the immediate previous negative action (a condition true in $|\mathbf{A}|$, thus also satisfied in $\mathfrak{d}$ because all its views are views of designs in $|\mathbf{A}|$)
- (P-visibility) is natively satisfied by the fact that s is a promenade in the tree representing a design.

For example, s can travel in the slice $\mathfrak{c}$ as a breadth-first search on couples of nodes (κ^-, κ^+) such that κ^+ is just above κ^- in the tree, and κ^+ is proper. Then 2 cases:

- Either for all i, there exists $n_i \in \mathbb{N}$ and $\mathfrak{d}_i \in \mathbf{A}_{n_i}$ such that s_i is an ∞-path of $\mathfrak{d}_i$. Without loss of generality we can even suppose that $\mathfrak{c}_i \sqsubseteq \mathfrak{d}_i$: if it is not the case, replace some positive subdesigns (possibly Ω) of $\mathfrak{d}_i$ by $\maltese$ until you obtain $\mathfrak{d}'_i$ such that $\mathfrak{c}_i \sqsubseteq \mathfrak{d}'_i$, and note that indeed $\mathfrak{d}'_i \in \mathbf{A}_{n_i}$ since $\mathfrak{d}_i \preceq \mathfrak{d}'_i$. Let $N = \max_{1\leq i\leq k}(n_i)$. Since $\mathfrak{d} \notin \mathbf{A}$, thus in particular $\mathfrak{d} \notin \mathbf{A}_N$, there exists $\mathfrak{e} \in \mathbf{A}_N^\perp$ such that $\mathfrak{d} \not\perp \mathfrak{e}$. The reason of divergence cannot be infinite chattering, otherwise there would exist an infinite ∞-path t in $\mathfrak{d}$ such that $\widetilde{t}$ is in $\mathfrak{e}$, and t is necessarily in a single slice of $\mathfrak{d}$ (say $\mathfrak{c}_i$) to ensure its linearity; but in this case we would also have $\mathfrak{d}_i \not\perp \mathfrak{e}$ where $\mathfrak{d}_i \in \mathbf{A}_N$, impossible. Similarly, for all (finite) path s of $\mathfrak{d}$, there exists i such that s is a path of $\mathfrak{c}_i$ thus of $\mathfrak{d}_i \in \mathbf{A}_N$; this ensures that interaction between $\mathfrak{d}$ and $\mathfrak{e}$ cannot diverge after a finite number of steps either, leading to a contradiction.
- Or there is an i such that the (necessarily infinite) ∞-path s_i is in no design of $\mathbf{A}$. In this case, let $\mathfrak{e} = \ulcorner\widetilde{s_i}\urcorner^c$ (where $\widetilde{s_i}$ is a view since the $\mathbf{A}_n$ are simple), and with a similar argument as previously we have $\mathfrak{e} \in \mathbf{A}^\perp$ but $\mathfrak{d} \not\perp \mathfrak{e}$ by infinite chattering, contradiction. ◀