Inductive, Functional and Non-Linear Types in Ludics

Alice Pavaux

UNIVERSITÉ PARIS 13

Université Paris 13
Laboratoire d'Informatique de Paris Nord

THÈSE

présentée pour l'obtention du grade de

DOCTEUR DE L'UNIVERSITÉ PARIS 13

Discipline : Informatique

Inductive, Functional and Non-Linear Types in Ludics

Alice Pavaux

Soutenue publiquement le 1er décembre 2017

Jury :

Claudia Faggian	Université Paris 7	Examinatrice
Christophe Fouqueré	Université Paris 13	Directeur de thèse
Tom Hirschowitz	Université Savoie Mont Blanc	Rapporteur
Martin Hyland	University of Cambridge	Rapporteur
Damiano Mazza	Université Paris 13	Examinateur
Laurent Regnier	Université d'Aix-Marseille	Examinateur

Remerciements

Le moment d'écrire les remerciements est pour moi le plus agréable – joie et soulagement d'avoir terminé – mais également le plus difficile – comment faire tenir autant de noms en si peu de lignes ? Cette thèse n'aurait jamais pu aboutir sans le soutien de nombreuses personnes, c'est peu dire que je leur dois énormément.

En premier lieu, je tiens à remercier mon directeur de thèse, Christophe Fouqueré, pour son implication constante et son aide précieuse durant mes trois années de doctorat. Sa conviction que j'arriverai au bout a fini par me convaincre aussi.

Je suis très reconnaissante à Tom Hirschowitz et Martin Hyland d'avoir accepté d'être mes rapporteurs ; leurs commentaires m'ont permis de poser un regard nouveau sur certains aspects de mon propre travail et ont conduit à des améliorations de ce manuscrit. Merci également à Claudia Faggian, Damiano Mazza et Laurent Regnier pour leur participation à mon jury.

Malgré ma peur à l'idée de confronter mes idées à celles des autres, il m'est arrivé à de nombreuses reprises d'interagir avec des chercheuses et chercheurs et cela s'est toujours révélé d'une grande utilité. Dans le désordre, pour une discussion fructueuse ou des remarques pertinentes, je tiens à remercier Thomas Seiller, Claudia Faggian, Alexis Saurin, Myriam Quatrini, Patrick Baillot, Paul-André Melliès, Kazushige Terui, et j'en oublie.

La bonne ambiance qui règne au LIPN a rendu mon passage ici très agréable. Pour cela, je dois remercier tout d'abord les membres du groupe de logique, et en particulier Pierre Boudes et Damiano Mazza avec qui j'ai eu le plaisir de partager le meilleur bureau du labo. Merci à Camille Coti qui, quant à elle, a partagé avec moi sa science des bruits de couloir, ainsi que son chat. La bienveillance et l'énergie de Brigitte Guéveneux vont me manquer. Et évidemment, j'adresse un énorme *big up* à tous mes amis camarades de thèse. Je voudrais remercier plus particulièrement Luc Pellissier, grâce à qui j'ai découvert la vie sur Youtube, Thomas Rubiano, qui n'a toujours pas terminé ses formations doctorales, et Antoine Kaszczyc, dont j'ai mangé tous les gâteaux ; sérieusement, je ne sais pas ce qu'il serait advenu de moi sans vous.

Outre la recherche, mon doctorat à l'université Paris 13 a aussi été l'occasion d'enseigner quelques notions d'informatique, ce qui a constitué pour moi un aspect très plaisant – oserai-je dire, le meilleur ? – de mon travail. Je remercie donc d'une part les responsables des cours auxquels j'ai participé, notamment Christophe Fouqueré, Sébastien Guérif, Julien David, Yann Chevaleyre, Pierre Boudes, Nadi Tomeh, et d'autre part les étudiants qui, même s'ils ne s'en sont pas rendu compte, m'ont beaucoup appris.

Sur une note plus légère, merci aux moutons de Paris 13 (et non, je ne parle pas des étudiants cette fois) d'avoir su réconcilier, pour un temps au moins, la part de moi qui voulait faire de la recherche et celle qui se voyait bergère dans les Pyrénées.

Comment en suis-je arrivée là ? Certains professeurs côtoyés au cours de mon cursus scolaire y sont sans doute pour quelque chose : Mme Rigal, ma maîtresse de CM1 et CM2, qui m'a appris qu'une prof pouvait se tromper tandis que ses élèves avaient raison (encore une histoire de moutons…) ; M. Dey, mon extraordinaire prof de latin de quatrième, avec son *age quod agis* et bien d'autres gimmicks ; M. Kotecki, mon prof de math de première, qui m'a réconcilié avec la discipline en me posant enfin des difficultés. Arrivée en licence puis en master à l'université Paris 7, j'ai suivi de nombreux cours de mathématiques et d'informatique formateurs et de qualité. Je tiens à exprimer ma reconnaissance particulière à Thierry Joly, dont les cours géniaux m'ont réellement fait kiffer la logique.

Soit dit en passant, je voudrais remercier le Science et Vie Junior n°185 de février 2005, pour m'avoir ouvert les portes de l'infini.

À ceux – amis et famille – qui m'ont encouragé sans relâche, qui m'ont écouté me plaindre sans perdre leur calme, qui ont su me remonter le moral quand ça n'allait pas, je suis infiniment redevable. J'aimerais remercier Raphaël, qui m'a sauvé la vie à maintes reprises, n'hésitant pas à se jeter dans la gueule de la Louve à ma place. Merci également à Elodie, avec son enthousiasme débordant, d'être toujours prête à m'écouter raconter des histoires sans intérêt. J'ai une pensée particulière pour Saskia qui est non seulement une amie sincère mais également une mathématicienne *badass* ! Je n'oublie pas tous les autres anciens de P7. Merci aux théâtreux des frigos et aux abeilles de Zé (Junior inclus) pour les shoots de bonne humeur. Merci beaucoup à Béa pour la relecture *in English, please !* Je remercie sincèrement mes parents, Brigitte et Vincent, qui m'ont soutenu tout du long alors qu'ils ne comprenaient rien à ce que je faisais, ce qui est tout à fait admirable. Angèle mérite aussi toute ma gratitude, elle qui m'a laissé jouer la petite sœur de temps en temps, pour me ménager. Un immense merci à Julien qui, non content d'être un supporter de la première heure, est carrément devenu mon coach dans la dernière ligne droite. Je pense également au reste de la bande jovienne et parisienne, aux week-ends où l'on se roule tous ensemble dans la flemmardise, et ça fait du bien.

Enfin, merci à Marilène et Nandy pour leur soutien moral, de nature très différente, mais ô combien nécessaire.

Contents

Introduction

This thesis is a contribution to the field of logic in computer science. In the setting of ludics, a system that fits into the Curry–Howard correspondence, we study the formulas/types from both a logical and a computational point of view. The focus is set in particular on inductive data types, functional types and non-linearity. We analyse the structure and the interactive properties of the objects considered, two aspects closely related in ludics where interaction is the central notion.

Context

Proofs as Programs

The *Curry–Howard correspondence* is a striking connection between proof theory and programming language theory. It states that proofs and programs are the same objects, in a sense that can be made precise. This observation has shed a new light on two fields of research, that were previously independent, and has led to fruitful interactions.

Proof theory is the branch of mathematical logic studying proofs as formal objects. It grew in the beginning of the 20^{th} century. At that time, mathematics were going through a foundational crisis, leading to an increasing demand for formalism, in particular under the impulsion of Hilbert. In 1934, Gentzen introduced *natural deduction* [Gen34], and *sequent calculus* the following year [Gen35], as formal syntaxes for proofs, so as to demonstrate the coherence of arithmetic. The main theorem (*Hauptsatz*) of these systems is the *cut-elimination theorem*: any sequent calculus proof with *cuts*, corresponding to the use of a lemma, can be rewritten in a cut-free proof of the same statement.

Around that period also, several models of computation were developed, in particular the famous Turing machines [Tur36]. Another one was the λ*-calculus*, imagined by Church [Chu41] as a theory of functions for the foundations of mathematics, but then recast into a successful computational tool. It has since been particularly well-suited for the theoretical analysis of functional programming.

First observed by Curry [Cur34] in the 1930s for combinatory logic, the proof–program correspondence was later made precise by Howard [How80] who describes how simply-typed λ-calculus and intuitionistic natural deduction coincide. What is now known as the Curry–Howard correspondence can be presented as the identification of three main layers: formulas correspond to types, proofs to programs, and cut-elimination to program

Logic	Programming
formulas	**types**
atomic formula	base type
logical connective	type constructor
proofs	**programs (λ-terms)**
introduction of $\Rightarrow$	abstraction
elimination of $\Rightarrow$	application
cut-elimination	**evaluation (β-reduction)**
cut	redex

Figure 1: The Curry–Howard correspondence

evaluation (see Figure 1). The third (dynamic) part of this correspondence is particularly interesting, since it means that the cut-elimination procedure of proofs actually computes. Curry–Howard has since been extended to various logics or computational features, bringing a better understanding to both logical systems and programming languages. In particular, it was at the origin of the emergence, 30 years ago, of linear logic.

From Linear Logic to Ludics

Linear logic and connectives. *Linear logic* [Gir87, Gir06] arose from Girard's study of coherent spaces as a denotational model for system F, a second-order λ-calculus. It is a logic concerned with resource consumption. Typically, the intuitionistic implication $\Rightarrow$ is decomposed into two operations:

$$A \Rightarrow B \qquad \text{becomes} \qquad !A \multimap B$$

where ! (*of course*) allows multiple uses of A, while $\multimap$ (*lollipop*) is an implication that *linearly* consumes its source to turn it into its target. Hypotheses are now resources: the number of times a hypothesis is used in a proof does matter. As a consequence, we distinguish between the connectives of linear logic that are strictly linear, the *multiplicative–additive* connectives ($\otimes, \oplus, \parr, \&, \multimap$), and those which deal with duplication and erasure, the *exponentials* ($!, ?$). The structural rules are restricted to exponential formulas, so as to precisely control the use of resources. This restriction leads to several versions of the usual connectives. In particular, we have:

- a multiplicative ($\otimes$ *tensor*) and an additive ($\&$ *with*) conjunction,
- a multiplicative ($\parr$ *par*) and an additive ($\oplus$ *plus*) disjunction.

These connectives are multiplicative or additive depending on how the logical rules associated deal with context. They also come with polarities: $\otimes$ and $\oplus$ are said *positive* while $\&$ and $\parr$ are *negative*.

Polarisation and focalisation. The distinction between positive and negative connectives comes from the observation that the inference rules associated to negative ones

are *reversible*, contrarily to the rules for positive ones. Reversibility means that the conclusion of a rule is provable if and only if its premises are. As a consequence, during proof-search, one can safely start by decomposing negative formulas without losing anything in terms of provability. Starting from this idea of reversibility, Andreoli [And92] goes further with *focalisation*, a remarkable property satisfied by linear logic proofs. It relies on the observation that the positive rules also have interesting characteristics. Focalisation states that any provable formula admits a focalised proof, that is, a proof constructed according to the following proof-search strategy:

- if the sequent contains a negative formula then decompose this formula,
- otherwise, try and choose a positive formula, and decompose it and its subformulas until obtaining a negative formula.

Focalised proofs are thus proofs in the branches of which negative and positive layers of rules alternate. Therefore, it is possible to design a proof system with synthetic connectives (gathering layers of connectives of same polarity), where there are a generalised positive rule and a generalised negative one. A proof then alternates such rules. The focalisation discipline has played a major role for the design of *polarised linear logic* [Lau02] and *ludics* [Gir01].

From proofs to designs. In Girard's presentation of ludics [Gir01], the basic objects, called *designs*, look like proofs of multiplicative–additive linear logic (MALL) in which we would have erased the formulas, keeping only the structure. More precisely, designs are derived from proofs that are focalised and with synthetic connectives. We give an idea of how this is done. Below is a very simple example of (part of) a focalised MALL proof with synthetic connectives, where A, B, C, D are positive formulas.

$$\dfrac{\dfrac{\vdots}{\vdash A, B}}{\vdash A \parr B} \quad \dfrac{\dfrac{\vdots}{\vdash C} \quad \dfrac{\vdots}{\vdash D}}{\vdash C \mathbin{\&} D}}{\vdash ((A \parr B) \otimes (C \mathbin{\&} D)) \oplus E}$$

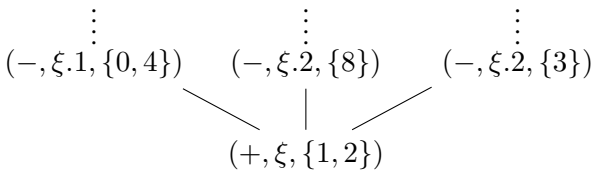
Notice that the first rule from the bottom decomposes the synthetic positive connective $(\_ \otimes \_) \oplus \_$. If we forget about the formulas, and we only keep some information about their *location*, the previous proof becomes:

$$\dfrac{\dfrac{\dfrac{\vdots}{\vdash \xi.1.0, \xi.1.4}}{\xi.1 \vdash} \quad \dfrac{\dfrac{\vdots}{\vdash \xi.2.8} \quad \dfrac{\vdots}{\vdash \xi.2.3}}{\xi.2 \vdash}}{\vdash \xi}$$

After pushing negative formulas on the left side of the symbol $\vdash$, all formulas have been replaced by finite sequences of numbers corresponding to *addresses*, where:

- the conclusion formula is given any address ξ,
- for every formula of address σ, its immediate subformulas (with respect to synthetic connectives) are given an address of the form $\sigma.n$ for $n \in \mathbb{N}$,

For example here, $\xi.1$ is the address of the subformula $A \parr B$. Now, if we keep only the tree structure of the proof and we label each node by an *action* corresponding to the logical rule used, we get the following:



An action (ϵ, σ, R) is composed of:

- a *polarity* $\epsilon \in \{-, +\}$, corresponding to the polarity of the logical rule applied,
- an *address* σ, corresponding to the location of the active formula of the rule,
- a *ramification* $R \subseteq \mathbb{N}$, indicating the subformulas freed by the rule, and that can possibly be used as active formulas above this node.

Such a tree is a design. However, designs can be infinite both in height and in width in general; in particular, there is no equivalent of the axiom rule, replaced instead by an infinitary η-expansion. Moreover, there exists a special action *daimon* in ludics that has no proof-theoretic equivalent. A detailed exposition of the way designs derive from focalised proofs is found in Curien's notes [Cur05].

Computational designs. Terui [Ter11] introduces another syntax for designs: instead of proof skeletons, designs are terms in the style of π-calculus. In this new syntax, the design previously given as example becomes:

$$x|\overline{a}\langle b(y,z).\mathfrak{p}_1, c(s).\mathfrak{p}_2 + d(t).\mathfrak{p}_3\rangle$$

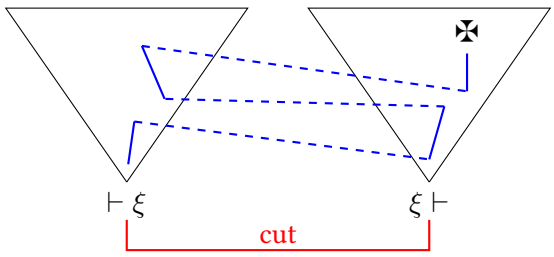
where $\mathfrak{p}_1, \mathfrak{p}_2, \mathfrak{p}_3$ are positive designs. The symbols a, b, c, d are called *names* and generalise the notion of ramification. $x|\overline{a}$ is a positive action of address x, while $b(y,z)$, $c(s)$ and $d(t)$ are negative actions, and do not have explicit addresses. The relation between an address and its sub-addresses in Girard's syntax is partially recovered here as variable binding. The original formulation of ludics by Girard emphasises the geometry of proofs, while Terui's syntax insists on their computational aspect; both have their advantages, but we shall prefer the latter for this thesis.

Ludics, an Interactive Semantics

Ludics. When introduced by Girard [Gir01], ludics was aimed at providing a framework to reconstruct logic *from scratch*. In particular types/formulas are not given but are recovered from interaction. Girard proves that ludics gives a fully complete model for a polarised variant of MALL with second-order quantification, and since then other polarised variants of linear logic have been considered in ludics, with exponentials [BF11, BT10b] or fixed points [BDS15]. The computational meaning of ludics has also been explored, in particular by viewing designs as data or functions [Sir15, Ter11] and by considering ludics as

a foundation for logic programming [Sau08]. The similarities between ludics and game semantics have been investigated [FH02, BF11, FQ13, FQ16]. All these works have proved ludics to be a fruitful setting for both logical and computational purpose.

INTERACTION AND BEHAVIOURS. As we have seen, designs are proof structures without formulas, we could say *untyped* proofs. Designs only retain from proofs the relevant information with respect to cut-elimination, that is the tree structure and the formulas' location. And this is precisely the dynamics of cut-elimination, called *interaction* in ludics, that allows to recover the typing for designs. A cut occurs between two designs when they share a common location with opposite polarities, and interaction eliminates such cuts. A particularly interesting case is *closed* interaction, corresponding intuitively to eliminating a cut between a proof of A and a proof of $A^\perp$, which may seem surprising. This is made possible by the special rule ✠ (*daimon*) of ludics, allowing to "prove" anything. As a consequence, designs are more general than proofs, since some are cheating by using the daimon. Interaction can be seen as a confrontation between two proof sketches, where each one tries to refute what the other asserts, and if one eventually plays ✠ it means "I give up" and the process stops. From a programming perspective, ✠ corresponds to an error/exception at runtime, causing an interruption of the execution.



The reason why closed interaction and the daimon are needed is to be able to recover the types of ludics, the *behaviours*. A closed interaction leads either to ✠ – if the two designs discuss gently until one gives up – or to Ω – if one is unable to answer an unexpected question of the other or if the dialogue lasts forever – which are interpreted respectively as convergence and divergence. Two designs are *orthogonal* if their interaction converges, and a behaviour $\mathbf{B}$ is a set of designs closed under bi-orthogonal.

$$\mathbf{B}^{\perp\perp} = \mathbf{B}$$

Equivalently, it is the set of all the designs which pass the same set of tests $\mathbf{B}^\perp$, where tests are also designs. The idea is that of assimilating a formula to the set of its proofs. Ludics thus fits in Curry–Howard as follows: designs correspond to proofs/programs, interaction is cut-elimination/evaluation, and behaviours are equivalent to formulas/types.

Logical connectives and completeness. As soon as introducing ludics, Girard provides a full completeness result for polarised MALL. The interpretation of MALL formulas as behaviours relies on constructors that combine behaviours in order to form new ones; such constructors are called *logical connectives*, since they correspond to actual connectives of linear logic. The typical way to define a connective in ludics is by taking the bi-orthogonal closure of a set of designs built from the designs of other behaviours.

$$\mathbf{A} \otimes \mathbf{B} = \left\{ \begin{array}{c} \mathfrak{d} \quad \mathfrak{e} \\ \otimes \end{array} \;\middle|\; \mathfrak{d} \in \mathbf{A}, \mathfrak{e} \in \mathbf{B} \right\}^{\perp\perp}$$

Closing by bi-orthogonal ensures that a set constructed this way is a behaviour. However, there exists a fundamental result called *internal completeness* which states that the bi-orthogonal closure is actually not necessary for such a set to be a behaviour. This property thus gives a direct and compositional description of the contents of behaviours constructed by logical connectives. Internal completeness is a key prerequisite for the full completeness result.

Ludics and other semantics. Ludics can be seen as a variant of *game semantics*, and more particularly of the one proposed by Hyland and Ong [HO00] (HO games). Game semantics has successfully provided fully abstract models for various logical systems and programming languages, in particular PCF [HO00, Nic94, AJM00]. A computation is modeled as a *play* between a player and an opponent who move alternatively, following the game rules given by an *arena*. A set of plays on an arena can constitute a *strategy* for the player. In ludics, the interaction between two designs can be described as a play, and designs resemble strategies. Faggian and Hyland [FH02] actually describe ludics as a game semantics on a universal arena. Other pieces of work cultivate the analogy [BF11, FQ13, FQ16], the common idea being to describe designs in terms of the traces of their possible interactions. However, ludics is more symmetrical than game semantics, designs being strategies for either the player or the opponent. This symmetry comes from the daimon, and enables closed interaction, which has no equivalent in game semantics. The possibility to differentiate between divergence and termination allows to recover the types interactively, instead of fixing an arena *a priori* by restricting the possible moves. In other words, no game rules are given before we start playing. As a consequence, studying types in ludics is more about exploring than constructing. The semantic types of ludics, obtained by a bi-orthogonal closure, are reminiscent of the way types are defined in *classical realisability* [Kri09] and in *geometry of interaction* [Gir89]. On a higher level of abstraction, this is also similar to the *double-glueing* categorical construction [HS03]. Let us finally mention that designs and their interaction fit into the framework of *abstract Böhm trees* [CH07], which are kind of strategies together with an abstract machine, general enough to embed Böhm trees, λ-calculus, PCF and, as a matter of fact, ludics.

Inductive, Functional and Non-Linear Types in Ludics

A Structural Approach

Exploring the types of ludics. Original (linear) ludics contains many other behaviours than those interpreting MALL formulas, and even more if we consider non-linear extensions of ludics. The starting point of our research is the following question: since most behaviours are not the interpretation of linear logic formulas, what are they? This question – raised from the beginning of ludics – brings the idea that the remaining behaviours could have an interesting logical or computational counterpart. Exploring the behaviours demands that we study the way they interact, and interaction in ludics is intimately linked to the structure of the objects. We have several tools available to analyse the structure of behaviours, the first and maybe most important one being internal completeness. Other tools are *visitable paths*, *regularity* and *purity*.

Paths. The result of a closed interaction gives no information but convergence or divergence. On the other hand, the trace of the interaction process is much more informative. Exploiting this idea, Fouqueré and Quatrini [FQ13] define *paths* as a ludics equivalent of plays in HO games. A path is a sequence of actions describing an interaction. What is interesting is to consider the set of visitable paths at the level of a behaviour; this set characterises the behaviour in the sense that all the possible interactions are captured. Regularity and purity are two interactive properties of behaviours that rely on the notion of visitable path.

Regularity. Informally, 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. In this thesis, we prove an internal completeness result for infinite unions of behaviours – discussed later – which relies on the hypothesis of regularity for these behaviours. This property is not actually ad hoc: it was introduced by Fouqueré and Quatrini [FQ16] 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 – letting finiteness aside – regularity captures exactly the behaviours corresponding to μMALL formulas, a logic with fixed points [Bae12].

Purity. Thinking of Ludics as a programming language, we would like to guarantee *type safety*, that is, ensure that "well typed programs cannot go wrong" [Mil78]. 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 [Sir15] (and called *principality* in her work), this property is related to the notions of *winning* designs [Gir01] and *pure* designs [Ter11], but at the level of a behaviour.

Our contributions. This thesis begins by providing the tools we will need to study the behaviours in the following. We employ the elegant term-calculus formulation of ludics proposed by Terui [Ter11] (restricted to linear designs). In this syntax, it is not entirely possible to consider actions individually since designs are *compiled* in a sense, but adapting the notion of path requires that we do so. More precisely, we need to recover addresses for all actions, and this is done with a notion of located actions.

A path corresponds to the trace of an interaction between two designs; but designs may split during the process, leading to an interaction between more than two designs after some reduction steps. As a consequence, if we want to define the interaction path inductively, we need to consider interaction between two sets of designs, which we call *multi-designs*. And we do need to give an inductive definition of interaction path, so that some results can be proved by induction on the length of a path. In particular, we prove that two (multi-)designs are orthogonal if and only if there exists a path appearing on both sides.

Following Fouqueré and Quatrini [FQ16], we study the logical connectives $\downarrow$, $\uparrow$, $\oplus$, $\otimes$ and $\multimap$ of ludics (where $\downarrow$ and $\uparrow$ are shifts of polarities) in terms of their visitable paths. For example, the case of $\oplus$ is easy: the set of visitable paths of $\mathbf{M} \oplus \mathbf{N}$, noted $V_{\mathbf{M}\oplus\mathbf{N}}$, is essentially the disjoint union of $V_{\mathbf{M}}$ and $V_{\mathbf{N}}$. Difficulties come with the tensor, because in the general case the set $V_{\mathbf{M}\otimes\mathbf{N}}$ is described using a tricky condition; moreover, the proof relies on results proved with multi-designs. Using the form of the visitable paths of the connectives, we prove that regularity is stable under all such behaviour constructions, and purity is stable under all except $\multimap$.

Fixed points, inductive and functional types

Fixed points theory. Induction and coinduction, especially the first one, are common methods in mathematics for definitions and proofs. They deal with fixed points: a definition by induction corresponds to describing the least fixed point of some operator, while coinduction is a greatest fixed point. The interest that computer science shows in fixed points probably finds its origins in theorems such as Knaster–Tarski's [Tar55] or Kleene's [Kle52]. Given a complete lattice and a monotone operator over it, the Knaster–Tarski fixed point theorem ensures the existence of a complete lattice of fixed points, entailing in particular the existence of a least and a greatest fixed point. Kleene's theorem has a more constructive flavour: it constructs the least fixed point of a continuous function over a complete partial order (CPO) as the supremum of an ascending chain. The use of fixed points has then spread in areas such as recursive functions [Rog67], formal languages [HMU07] and, more interestingly for us, in the study of the semantics of programs [SS71]. One may refer to [San09] for a good historical background of fixed points in computer science.

Fixed points in linear logic. Reasoning about induction and coinduction in logic can be done by adding fixed points. In particular, it is possible to extend MALL with least and greatest fixed points, leading to μMALL [Bae12]. In this logic, the formulas are those of

MALL to which are added some of the form

$$\mu X.A \qquad \text{and} \qquad \nu X.A$$

where second-order variables can appear in A. There are logical rules associated to the (dual) connectives μ and ν, and the whole system enjoys cut-elimination and focalisation. The interest of such a logic is to gain some expressivity by allowing an infinite use of resources, which is usually handled by the exponentials in linear logic. Fixed points can thus be seen as an alternative to exponentials.

Fixed points in ludics. Baelde, Doumane and Saurin [BDS15] provide a ludics model for μMALL; their work is the starting point for our study of inductive types in ludics. They interpret formulas of the form $\mu X.A$ and $\nu X.A$ as respectively the least and the greatest fixed point of an operator over the positive behaviours. The existence of such fixed points behaviours is ensured by the Knaster-Tarski theorem, but this approach is non-constructive in the sense that it does not provide an explicit way to build the fixed points. We shall overcome this by using Kleene's fixed point theorem instead.

Fixed points have also been studied in other settings close to ludics. Both McCusker [McC98] and Clairambault [Cla09] have considered fixed points in game semantics. Melliès and Vouillon [MV05] have introduced a realisability model for recursive (i.e., inductive and coinductive) types.

Data and functions. Functional programming usually deals with data, functions over data and functions over functions. Data types allow one to present information in a structured way. Many data types have a natural inductive (e.g., natural numbers, lists) or coinductive (e.g., streams, infinite trees) structure, thus fixed points are particularly well-suited to present them from a theoretical perspective. The representation of both data and functions in ludics has been studied previously. Terui [Ter11] proposes to encode them as designs in order to express computability properties in ludics; in his work, data and functions are not considered at the level of behaviours, i.e., of types. Sironi [Sir15] describes the behaviours corresponding to some data types: integers, lists, records, etc. as well as first-order function types.

Our contributions. Our aim is to interpret constructively the (potentially inductive) data types and the (potentially higher-order) functional types as behaviours of ludics, so as to study their interactive properties. We describe the language of *data patterns* – much inspired by the syntax of μMALL formulas – in which all the usual (inductive) data types can be written, for example

$$\mathbb{N}\mathrm{at} = \mu X.(\mathtt{zero} \oplus X) \ .$$

Following [BDS15], we interpret these patterns compositionally as behaviours constructed by logical connectives and by fixed points. The novelty is that we provide an internal completeness theorem for fixed points, so that such *data behaviours* can be described in a constructive way. This result states that, given an infinite sequence $(\mathbf{A}_n)_{n\in\mathbb{N}}$ of increasingly

large behaviours satisfying particular conditions (among which regularity), we have

$$(\bigcup_{n\in\mathbb{N}} \mathbf{A}_n)^{\perp\perp} = \bigcup_{n\in\mathbb{N}} \mathbf{A}_n \ .$$

By Kleene's fixed point theorem, the behaviour interpreting an inductive type $\mu X.A$ is of the form $(\bigcup_{n\in\mathbb{N}} \mathbf{A}_n)^{\perp\perp}$; our internal completeness result applies here, and it indicates that the bi-orthogonal closure is unnecessary, thus a fixed point behaviour is obtained as a simple union. Using this, we prove that all data behaviours are regular.

We also combine data behaviours to construct the *functional behaviours*: a behaviour of the form $\mathbf{A} \multimap \mathbf{B}$ is the set of designs that, when interacting with a design of type $\mathbf{A}$, outputs a design of type $\mathbf{B}$. In particular, our framework includes higher-order functions. Functional behaviours are proved to be regular too, but on the other hand not always pure. More precisely, we give a characterisation of impure function types: we show that a functional behaviour fails to satisfy purity if and only if it is higher-order and takes functions as argument; this is typically the case of

$$(\mathbf{A} \multimap \mathbf{B}) \multimap \mathbf{C} \ .$$

We prove however that under a *well-bracketedness* condition we can avoid such errors in the execution of ludics programs. We will discuss the computational meaning of this result, making a parallel with HO games.

Our results concerning inductive and functional types in ludics have been presented in a CSL paper [Pav17].

Exponentials and Non-Linearity

Exponentials. Although nice and well-behaved, MALL is a weak logic. Recovering more expressive power requires that we go beyond this strictly linear fragment of linear logic; one way to do this is with fixed points, that we just mentioned, and another is by considering *exponentials*. The exponentials come as two unary connectives, dual to each other, ! (*of course*) and ? (*why not*). Among the cut-elimination steps dealing with exponentials, the interesting one is between the rules of *contraction* and *promotion*:

$$\dfrac{\dfrac{\dfrac{\vdots\,\pi_1}{\vdash ?\Gamma, A}}{\vdash ?\Gamma, !A}\,!\quad \dfrac{\dfrac{\vdots\,\pi_2}{\vdash \Delta, ?A^{\perp}, ?A^{\perp}}}{\vdash \Delta, ?A^{\perp}}\,\mathrm{c}}{\vdash ?\Gamma, \Delta}\,\mathrm{cut} \quad \rightsquigarrow \quad \dfrac{\dfrac{\dfrac{\vdots\,\pi_1}{\vdash ?\Gamma, A}}{\vdash ?\Gamma, !A}\,!\quad \dfrac{\dfrac{\dfrac{\vdots\,\pi_1}{\vdash ?\Gamma, A}}{\vdash ?\Gamma, !A}\,!\quad \dfrac{\vdots\,\pi_2}{\vdash \Delta, ?A^{\perp}, ?A^{\perp}}}{\vdash ?\Gamma, \Delta, ?A^{\perp}}\,\mathrm{cut}}{\dfrac{\vdash ?\Gamma, ?\Gamma, \Delta}{\vdash ?\Gamma, \Delta}\,\mathrm{c}}\,\mathrm{cut}$$

Notice that proof π_1 is duplicated in this process. Such duplications are precisely what the concept of *non-linearity* wants to capture in logic. On the other hand, from the programming point of view, non-linearity expresses the ability for a program to use its arguments

multiple times, or to be called multiple times. The relationship between exponentials and fixed points, as two alternatives to upgrade MALL, has been studied, in particular by Baelde and Miller [BM07, Bae12]. The two features seem to have different expressive power, but from our understanding it is not so clear what one can do that the other cannot. Baelde and Miller have encoded μMALL in LL2 (linear logic with second order quantification), however they doubt that the same is possible without second order; the other way round, they remark that fixed points satisfy structural rules, but we do not fully apprehend to what extent this makes it possible to get by without exponentials.

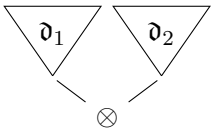
Non-linearity in ludics. Original ludics [Gir01] is linear, which is enough to capture the multiplicative–additive fragment of linear logic but not the exponentials. Moreover, linear designs are unable to model programs that can be called arbitrarily many times during an execution. Hence the need for non-linear ludics. Non-linearity manifests itself in the possibility for an address to be repeated in a design. This implies that, during an interaction, designs may be duplicated. There have been various works to extend ludics so as to handle duplications with non-linearity. Maurel [Mau04] was the first to propose such a framework: he allows repetitions of addresses in Girard's designs, adding justification pointers to discard possible ambiguities. The syntax of Terui [Ter11] is natively able to describe non-linear designs. In a game semantics fashion, Basaldella and Faggian [BF11] introduce neutral actions to get *non-deterministic* counter-designs interacting against non-linear designs, and they show a full completeness result for a polarised version of multiplicative–exponential linear logic (MELLS). The same idea of non-linearity against non-determinism is employed by Basaldella and Terui [BT10b] who simply extend the designs–as–terms syntax of [Ter11] with non-deterministic superpositions of positive designs; this allows them to recover exponentials by capturing full polarised linear logic (LLP). With non-linearity, some interesting properties of ludics are lost. Designs are no longer *separable*, thus two different designs may have exactly the same orthogonal designs. Because of that, internal completeness does not hold for positive connectives in non-linear ludics.

Another approach for modelling the exponentials in ludics relies on linear approximations, in the way of the Taylor expansion of λ-terms [ER08], as done by Mazza [Maz17]. This approach is closer to AJM games [AJM00] than HO.

Non-determinism as the dual of non-linearity. If non-linearity is considered without non-determinism, some designs have parts that cannot be visited by interaction. In order for interaction to take fully advantage of non-linear designs, one can introduce non-deterministic counter-designs. This is the same idea underlying the necessity of ✠ to have enough counter-designs against linear designs. Non-determinism corresponds to a non-uniform semantics for programs: when called twice, a program is allowed to change its mind and to give a different result. The non-determinism considered in ludics [BF11, BT10b] is universal, which means that an interaction converges only if every single choice leads to the daimon.

Our contributions. We settle the bases for an exploration of non-linear ludics, so as to better understand the structure of exponential behaviours. We employ the formalism of Basaldella and Terui [BT10b]. In this enriched framework, we adapt several notions introduced in the linear setting. What we are particularly interested in is internal completeness and a notion of path that can capture interaction traces.

As already mentioned, general internal completeness does not hold anymore for positive connectives. We prove an alternative result for the connectives $\downarrow, \otimes, \oplus$ that can be seen as a weaker form of internal completeness. In the case of the tensor, the result states the following: a design $\mathfrak{d} \in \mathbf{N}_1 \otimes \mathbf{N}_2$ is the gathering of two designs $\mathfrak{d}_1$ and $\mathfrak{d}_2$ linked by a supplementary action corresponding to $\otimes$ at the base:

$\mathfrak{d} =$ 

but instead of having $\mathfrak{d}_i \in \mathbf{N}_i$ (as internal completeness in the linear case), our result states that the result of interaction between $\mathfrak{d}_i$ and any design in $(\mathbf{N}_1 \otimes \mathbf{N}_2)^\perp$ is in $\mathbf{N}_i$. Such a result is similar to what has been obtained by Basaldella and Faggian [BF11]; however, their setting is different, and they do not consider additives.

In a non-deterministic setting, an interaction is not described by a single path anymore. To overcome this, we introduce *n-paths* which are sets of paths that are coherent in some sense. We give arguments, but not a formal proof, of why n-paths must be the good notion to capture interaction between two orthogonal non-linear and non-deterministic designs. We also propose other conjectures on visitable paths in this setting.

A lot of work is still to be undertaken in non-linear ludics. In particular, we aim to generalise our study of data and function types to this non-linear setting. Another goal we would like to achieve is characterising the behaviours corresponding to LLP formulas by a regularity argument, the same way Fouqueré and Quatrini [FQ16] characterised MALL.

Outline

The thesis is organised as follows:

- In **Chapter 1**, after reviewing the basic notions and some important theorems of ludics, we adapt the notion of path to Terui's syntax for designs and we present regularity and purity.
- **Chapter 2** is a technical one, in which we introduce multi-designs so as to prove the existence and uniqueness of the interaction path for two orthogonal (multi-)designs, as well as other useful results.
- **Chapter 3** makes the connection between logical connectives and paths, by providing the form of the visitable paths of behaviours constructed by connectives, and showing that regularity and purity are stable under the connective constructions.

- In **Chapter 4**, we interpret the inductive data types as behaviours using logical connectives and least fixed points. We show that such behaviours are regular and pure, and we provide a new internal completeness result for infinite unions of behaviours, unveiling the structure of fixed points behaviours.
- In **Chapter 5**, functional types are interpreted as behaviours as well, and we show that a behaviour of this kind is impure if and only if it corresponds to a type of functions that take functions as arguments.
- Finally, **Chapter 6** is focused on non-linear ludics. In this setting, we provide alternative results to internal completeness for the positive connectives, and we describe an interaction as an n-path. We end by suggesting some future work concerning paths in non-linear ludics.

1 | Ludics and Paths

This first chapter introduces ludics and settles precisely the tools we will need in the following. We describe *linear* designs and paths, which are our main focus in this thesis. The framework will occasionally be extended, though: in Chapters 2 and 6 we generalise our approach by considering respectively multi-designs and non-linearity. In the other chapters, the definitions given here fully apply.

In his seminal paper [Gir01], Girard gives an extensive presentation of ludics. The original syntax he introduces, driven by a sound conceptual point of view, gives clear intuitions on how designs are derived from sequent calculus proofs. However, it lacks practicality. Instead, we choose to adopt the formalism of Terui's *computational ludics* [Ter11], which is presented as a term calculus (akin to λ-calculus or π-calculus). We find this syntax easier to get accustomed to, moreover it can easily embed non-linearity, thus it is best suited for our purpose.

In Section 1.1, we recall the classical notions of ludics: designs, interaction, behaviours, incarnation and logical connectives. We also state three fundamental theorems that will be needed later; the first two, associativity and monotonicity, are known as being part of the *analytical theorems*, some remarkable properties of designs and normalisation; the third one, internal completeness, gives a direct description of behaviours constructed with logical connectives. Then in Section 1.2 we adapt to Terui's setting the work of Fouqueré and Quatrini [FQ13] about paths, which was originally conducted in Girard's syntax for ludics; this approach, which brings closer ludics and game semantics, will be one of our guidelines all along this thesis. It allows us, in Section 1.3, to define regularity and purity, two interactive properties of behaviours that we will get back to for the study of data and function types.

1.1 Computational Ludics

Let us introduce the necessary ludics background for the rest of the thesis. The *designs* are the primary objects of ludics, corresponding to proofs or programs. Cuts between designs can occur, and their reduction is called *interaction*. The *behaviours*, i.e., the types or formulas of ludics, are particular sets of designs, and are defined thanks to interaction. Compound behaviours can be formed with *logical connectives* constructions which satisfy *internal completeness*, a remarkable property giving a direct description of the behaviours.

1.1.a Designs, Interaction and Associativity

The atomic blocks of designs are *actions*. Suppose given :

- an infinite set $\mathcal{V}_0$ of variables and
- a countably infinite set $\mathcal{S}$, called **signature**, and an arity function $\mathrm{ar} : \mathcal{S} \to \mathbb{N}$. The elements $a, b, \dots \in \mathcal{S}$ are called **names**, and we will sometimes write their arities as superscripts: $a^i, b^j, \dots$ where $i, j \in \mathbb{N}$.

Definition 1.1.1 (*Action*)

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 1.1.2 (*Design*)

Positive and negative **designs** are coinductively defined by:

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

The daimon ✠ is for convergence, Ω for divergence. Proper positive designs (i.e., different from ✠ and Ω) play the same role as *applications* in λ-calculus: either a variable x or a negative design $\mathfrak{n}_0$ is applied, via a name a, to as many negative designs as the arity of a. Negative designs are a superposition of *abstractions*, where each name $a \in \mathcal{S}$ binds $\mathrm{ar}(a)$ variables and is followed by a positive design. Compared to the designs of [Ter11], our definition introduces only a minor change: we do not consider *identities*, a way to consider the axiom rule in ludics, which can instead be encoded as an infinitary η-expansion.

Notation

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. Moreover, we will often use the following notations:

- $a(\overrightarrow{x})$ for $a(x_1, \dots, x_n)$ and $\overline{a}\langle \overrightarrow{\mathfrak{n}} \rangle$ for $\overline{a}\langle \mathfrak{n}_1 \dots \mathfrak{n}_n \rangle$,
- Ω^- for $\sum_{a \in \mathcal{S}} a(\overrightarrow{x^a}).\Omega$ and $x|\overline{a}\langle \overrightarrow{\Omega^-} \rangle$ for $x|\overline{a}\langle \Omega^-, \dots, \Omega^- \rangle$,
- we will write partial sums for negative designs, 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, as well as the α-equivalence between designs. For the formal definitions, refer to [Ter11]. We will consider designs up to α-equivalence.

Let us give some more definitions:

- A design is **total** if it is $\neq \Omega$, it is **proper** if it is $\neq \Omega$ and $\neq ✠$.
- A **subdesign** of a design $\mathfrak{d}$ is a sub-term of $\mathfrak{d}$.

- A **cut** is a positive design of the form $\mathfrak{n}_0|\overline{a}\langle\overrightarrow{\mathfrak{n}}\rangle$, and a **cut in** a design $\mathfrak{d}$ is a subdesign of $\mathfrak{d}$ which is a cut. We call **cut-free** a design that contains no cut.

In this thesis – except in Chapter 6 – we will focus on *linearity*. Thus in the following when writing "design" we mean "linear design", as defined below. Linearity forbids the repetition of a variable in two different parts of a design that may be used during the same interaction.

Definition 1.1.3 (*Linear design*)

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.

Interaction provides a cut-elimination procedure, which corresponds to β-reduction if we consider a cut as a *redex* of λ-calculus.

Definition 1.1.4 (*Normalisation / interaction*)

The **normalisation** of designs – also called **interaction** between designs, when two sides are clearly identified – is obtained by means of a reduction step applied on cuts:

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

Let $\mathfrak{p}$ be a design, and let $\rightsquigarrow^*$ denote the reflexive transitive closure of $\rightsquigarrow$; if there exists a design $\mathfrak{q}$ that 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, which is a particular cut-free design, exists and is unique [Ter11]:

Definition 1.1.5 (*Normal form*)

The **normal form** of a design $\mathfrak{d}$, noted $(\!|\mathfrak{d}|\!)$, is defined by

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

Example 1.1.6

Let $x_0,x_1,x_2,x_3\in\mathcal{V}_0$ be distinct variables and $a^1,b^1,c^0\in\mathcal{S}$ be names. Consider the following designs:

$$\mathfrak{p} = x_0|\overline{a}\langle b(x_1).(x_1|\overline{c}\langle\rangle)\rangle \quad\text{and}\quad \mathfrak{n} = a(x_2).(x_3|\overline{b}\langle c().(x_2|\overline{b}\langle c().\maltese\rangle)\rangle) \ .$$

The interaction between $\mathfrak{n}$ and $\mathfrak{p}$, when $\mathfrak{n}$ is substituted in $\mathfrak{p}$ on variable x_0, corresponds to the following reduction steps (where bold vertical bars correspond to cuts, for the

sake of readability):

$$\begin{aligned}\mathfrak{p}[\mathfrak{n}/x_0] &= a(x_2).(x_3|\overline{b}\langle c().(x_2|\overline{b}\langle c().\Fourth\rangle)\rangle) \mid \overline{a}\langle b(x_1).(x_1|\overline{c}\langle\rangle)\rangle \\ &\rightsquigarrow x_3|\overline{b}\langle c().((b(x_1).(x_1|\overline{c}\langle\rangle)) \mid \overline{b}\langle c().\Fourth\rangle)\rangle \\ &\rightsquigarrow x_3|\overline{b}\langle c().((c().\Fourth) \mid \overline{c}\langle\rangle)\rangle \\ &\rightsquigarrow x_3|\overline{b}\langle c().\Fourth\rangle \quad .\end{aligned}$$

Thus the normal form is $(\!|\mathfrak{p}[\mathfrak{n}/x_0]|\!) = x_3|\overline{b}\langle c().\Fourth\rangle$

Finally, we state the *associativity theorem*, one of the important results in ludics which corresponds to a weak form of the Church-Rosser property. This theorem justifies reference to "the" normal form of a design.

Theorem 1.1.7 (*Associativity*)

Let $\mathfrak{d}$ be a design and $\mathfrak{n}_1, \dots, \mathfrak{n}_k$ be negative designs.

$$(\!|\mathfrak{d}[\mathfrak{n}_1/y_1, \dots, \mathfrak{n}_k/y_k]|\!) = (\!|(\!|\mathfrak{d}|\!)[(\!|\mathfrak{n}_1|\!)/y_1, \dots, (\!|\mathfrak{n}_k|\!)/y_k]|\!) \quad .$$

This result has first been established by Girard [Gir01]. The theorem, in the form given above, has been by Basaldella and Terui [BT10b].

1.1.b Behaviours

In the next section, we will describe a particular interaction as a sequence of actions, or path (Definition 1.2.10). By "particular" we mean that the interaction should be:

- *two-sided*, i.e., we can clearly identify two designs (or two sets of designs, see Chapter 2) corresponding to the duality program/environment or player/opponent, so that the trace of the interaction can be recorded on either side.
- *closed*, i.e., it does not produce a concrete result but outputs only an indication on termination of the computation: either ✠ (convergence) or Ω (divergence).

To this end, we consider *atomic designs* and we study the interaction between two atomic designs, which is the simplest interaction of this form. In the rest of this thesis, we distinguish a particular variable $x_0 \in \mathcal{V}_0$, that cannot be bound: it is reserved for positive atomic designs, and plays the role of an initial location.

Definition 1.1.8 (*Atomic, closed*)

Let $\mathfrak{d}$ be a design.

- $\mathfrak{d}$ is **atomic** if $\mathfrak{d}$ is positive and $\mathrm{fv}(\mathfrak{d}) \subseteq \{x_0\}$, or if $\mathfrak{d}$ is negative and $\mathrm{fv}(\mathfrak{d}) = \emptyset$.
- $\mathfrak{d}$ is **closed** if it is positive and it has no free variable.

Note that the normal form of a closed design is either ✠ or Ω, in other words either *convergence* or *divergence*. This binary possibility of output leads to distinguish between two forms of closed interaction:

- the well-typed one, which guarantees termination (✠),

- the bad one, in which infinite chattering occurs (Ω).

In particular, if $\mathfrak{p}$ and $\mathfrak{n}$ are atomic then $\mathfrak{p}[\mathfrak{n}/x_0]$ is closed, and the *orthogonality* relation between two atomic designs of opposite polarities indicates the convergence of their interaction. A ludics type, called *behaviour*, is then a set of atomic designs interacting the same way with their environment, i.e., closed by bi-orthogonal. These notions will be generalised in Chapter 2 with multi-designs.

Definition 1.1.9 (*Orthogonality*)

Two atomic designs $\mathfrak{p}$ and $\mathfrak{n}$ are **orthogonal**, noted $\mathfrak{p} \perp \mathfrak{n}$ or $\mathfrak{n} \perp \mathfrak{p}$, if $(\!|\mathfrak{p}[\mathfrak{n}/x_0]|\!) = \maltese$.

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

Definition 1.1.10 (*Behaviour*)

A set $\mathbf{B}$ of cut-free atomic designs of same polarity is a **behaviour** if $\mathbf{B}^{\perp\perp} = \mathbf{B}$.

A behaviour is either **positive** or **negative** depending on the polarity of its designs.

Notation

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

With our definition, a behaviour $\mathbf{B}$ only contains cut-free designs. We could have considered the set $\{\mathfrak{d} \text{ design} \mid (\!|\mathfrak{d}|\!) \in \mathbf{B}\}$ instead, which is also closed by bi-orthogonal if we take into account designs with cuts. The reason why we require cut-freeness is technical, in particular it gives a simple formulation of internal completeness (Theorem 1.1.21).

A behaviour really corresponds to a set of designs with the same "behaviour" since it can alternatively be defined as the orthogonal of a set E of cut-free 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 $E = \mathbf{B}^\perp$. All the designs passing the set of tests are equally able to interact with their common opponents, and are therefore part of the same type.

Example 1.1.11

Let $a^2, b^1, c^0 \in \mathcal{S}$. Consider the designs

$$\mathfrak{m}_1 = a(x_1, x_2).(x_1|\overline{b}\langle c().(x_2|\overline{b}\langle c().\maltese\rangle)\rangle)$$
$$\text{and} \quad \mathfrak{m}_2 = a(x_1, x_2).(x_2|\overline{b}\langle c().(x_1|\overline{b}\langle c().\maltese\rangle)\rangle)$$

(note that we will soon introduce a tree syntax for designs, which will make $\mathfrak{m}_1$ and $\mathfrak{m}_2$ more readable, see Figure 4 on page 38). The negative behaviour $\mathbf{A} = \{\mathfrak{m}_1, \mathfrak{m}_2\}^{\perp\perp}$ contains for example the designs

$$\mathfrak{m}'_1 = a(x_1, x_2).(x_1|\overline{b}\langle c().(x_2|\overline{b}\langle c().\maltese\rangle)\rangle) + b(y).\maltese$$
$$\text{and} \quad \mathfrak{m}'_2 = a(x_1, x_2).(x_2|\overline{b}\langle c().\maltese\rangle) \ .$$

Indeed, $\mathbf{A} = \{\mathfrak{p}\}^\perp$ where

$$\mathfrak{p} = x_0|\overline{a}\langle b(y_1).y_1|\overline{c}\langle\rangle, b(y_2).y_2|\overline{c}\langle\rangle\rangle$$

and we have $\mathfrak{m}'_1 \perp \mathfrak{p}$ and $\mathfrak{m}'_2 \perp \mathfrak{p}$.

1.1.c Incarnation

Not everything in a behaviour is useful, in the sense that some of its designs may have parts that can never be visited during interaction. If we go back to Example 1.1.11, notice that, in design $\mathfrak{m}'_1$, the part "$+b(y).\maltese$" is of no use for interacting with designs of $\mathbf{A}^\perp$; if we get rid of this, we obtain the design $\mathfrak{m}_1$ that is still in $\mathbf{A}$. Hence the notion of *incarnation*. The incarnation of a behaviour $\mathbf{B}$ is the subset of $\mathbf{B}$ containing only the designs whose actions are all visited during an interaction with a design in $\mathbf{B}^\perp$. Those correspond to the 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. Studying the incarnation is enough to prove the interactive properties of a behaviour.

Defining formally the incarnation requires to define first both the stable ordering ($\sqsubseteq$) and the intersection ($\cap$). The reader familiar with the original presentation of ludics will note that these correspond respectively to the inclusion and the real intersection of designs as sets of chronicles [Gir01].

Definition 1.1.12 (*Stable ordering*)

$\sqsubseteq$ is the largest binary relation $\mathcal{R}$ over designs such that:

1. if $\maltese\ \mathcal{R}\ \mathfrak{d}$ then $\mathfrak{d} = \maltese$,
2. if $\Omega\ \mathcal{R}\ \mathfrak{d}$ then $\mathfrak{d}$ is positive,
3. if $x|\overline{a}\langle\overrightarrow{\mathfrak{n}}\rangle\ \mathcal{R}\ \mathfrak{d}$ then $\mathfrak{d} = x|\overline{a}\langle\overrightarrow{\mathfrak{m}}\rangle$ and $\mathfrak{n}_i\ \mathcal{R}\ \mathfrak{m}_i$ for $1 \leq i \leq \mathrm{ar}(a)$,
4. if $\mathfrak{n}_0|\overline{a}\langle\overrightarrow{\mathfrak{n}}\rangle\ \mathcal{R}\ \mathfrak{d}$ then $\mathfrak{d} = \mathfrak{m}_0|\overline{a}\langle\overrightarrow{\mathfrak{m}}\rangle$ and $\mathfrak{n}_i\ \mathcal{R}\ \mathfrak{m}_i$ for $0 \leq i \leq \mathrm{ar}(a)$,
5. if $\sum_{a\in\mathcal{S}} a(\overrightarrow{x^a}).\mathfrak{p}_a\ \mathcal{R}\ \mathfrak{d}$ then $\mathfrak{d} = \sum_{a\in\mathcal{S}} a(\overrightarrow{x^a}).\mathfrak{q}_a$ and $\mathfrak{p}_a\ \mathcal{R}\ \mathfrak{q}_a$ for all $a \in \mathcal{S}$.

Definition 1.1.13 (*Intersection*)

$\cap$ is a partial operation over cut-free designs defined by:

- $\maltese \cap \maltese = \maltese$,
- $\mathfrak{p} \cap \Omega = \Omega \cap \mathfrak{p} = \Omega$,
- $x|\overline{a}\langle\overrightarrow{\mathfrak{n}}\rangle \cap x|\overline{a}\langle\overrightarrow{\mathfrak{m}}\rangle = x|\overline{a}\langle\mathfrak{n}_1 \cap \mathfrak{m}_1, \ldots, \mathfrak{n}_{\mathrm{ar}(a)} \cap \mathfrak{m}_{\mathrm{ar}(a)}\rangle$ if all $\mathfrak{n}_i \cap \mathfrak{m}_i$ defined,
- $\sum_{a\in\mathcal{S}} a(\overrightarrow{x^a}).\mathfrak{p}_a \cap \sum_{a\in\mathcal{S}} a(\overrightarrow{x^a}).\mathfrak{q}_a = \sum_{a\in\mathcal{S}} a(\overrightarrow{x^a}).(\mathfrak{p}_a \cap \mathfrak{q}_a)$ if all $\mathfrak{p}_a \cap \mathfrak{q}_a$ defined,
- $\mathfrak{d} \cap \mathfrak{e}$ is not defined otherwise.

The stable ordering means that a design is "less defined" than another, while the intersection takes the maximally defined cut-free design which is less defined than two others. Note that the intersection is defined only between two designs agreeing on the names of proper positive actions, it is not defined otherwise. This allows to describe the incarnation of a behaviour $\mathbf{B}$ as the set of (cut-free atomic) designs that are minimal for $\sqsubseteq$, but still defined enough to interact with designs of $\mathbf{B}^\perp$.

Definition 1.1.14 (*Incarnation*)

Let $\mathbf{B}$ be a behaviour and let $\mathfrak{d} \in \mathbf{B}$.

- The **incarnation** of $\mathfrak{d}$ in $\mathbf{B}$ is $|\mathfrak{d}|_{\mathbf{B}} = \bigcap\{\mathfrak{d}' \in \mathbf{B} \mid \mathfrak{d}' \sqsubseteq \mathfrak{d}\}$. If $|\mathfrak{d}|_{\mathbf{B}} = \mathfrak{d}$ we say that $\mathfrak{d}$ is **incarnated** (or **material**) in $\mathbf{B}$.
- The **incarnation** $|\mathbf{B}|$ of $\mathbf{B}$ is the set of the incarnated designs of $\mathbf{B}$.

Remark 1.1.15

The incarnation $|\mathfrak{d}|_{\mathbf{B}}$ of $\mathfrak{d}$ is the smallest (for $\sqsubseteq$) design $\mathfrak{d}'$ such that $\mathfrak{d}' \sqsubseteq \mathfrak{d}$ and $\mathfrak{d}' \in \mathbf{B}$.

Proof. We easily prove the (contrapositive of the) following assertion (α): if $\mathfrak{d}_1$, $\mathfrak{d}_2$ and $\mathfrak{e}$ are cut-free designs such that $\mathfrak{d}_1 \cap \mathfrak{d}_2$ is defined, if $\mathfrak{d}_1 \sqsubseteq \mathfrak{e}$ or $\mathfrak{d}_2 \sqsubseteq \mathfrak{e}$ then $\mathfrak{d}_1 \cap \mathfrak{d}_2 \sqsubseteq \mathfrak{e}$. Hence, by definition of the incarnation, $|\mathfrak{d}|_{\mathbf{B}} \sqsubseteq \mathfrak{d}$. Moreover, as remarked by Terui [Ter11], the fact that $|\mathfrak{d}|_{\mathbf{B}} \in \mathbf{B}$ is due to the *stability theorem*, one of the analytical theorems of ludics. Finally, for any other design $\mathfrak{d}'$ such that $\mathfrak{d}' \sqsubseteq \mathfrak{d}$ and $\mathfrak{d}' \in \mathbf{B}$, we have $|\mathfrak{d}|_{\mathbf{B}} \sqsubseteq \mathfrak{d}'$, indeed: since $\mathfrak{d}' \sqsubseteq \mathfrak{d}'$, by definition of the incarnation and by (α) we deduce $|\mathfrak{d}|_{\mathbf{B}} = |\mathfrak{d}|_{\mathbf{B}} \cap \mathfrak{d}' \sqsubseteq \mathfrak{d}'$. □

Notation

When there is no ambiguity on the behaviour considered, we simply write $|\mathfrak{d}|$ for $|\mathfrak{d}|_{\mathbf{B}}$.

By restricting to incarnated designs in a behaviour, we ensure that for any action in a design there exists an interaction using it, in the sense that this action will be part of a cut that is reduced at some point of the interaction process. Going further in this idea (Section 1.3), the regularity property will certify that every valid sequence of actions – every path – in designs of the incarnation is the trace of an interaction.

1.1.d Monotonicity

As associativity (Theorem 1.1.7), *monotonicity* is an analytical theorem that we will need in this thesis. In order to state it, we consider the *observational ordering* $\preceq$ over designs. Informally, we have $\mathfrak{d}' \preceq \mathfrak{d}$ if $\mathfrak{d}$ can be obtained from $\mathfrak{d}'$ by substituting:

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

The formal definition is given below.

Definition 1.1.16 (*Observational ordering*)

$\preceq$ is the largest binary relation $\mathcal{R}$ over designs defined as $\sqsubseteq$ (Definition 1.1.12) but replacing 3rd and 4th items by:

3. if $x|\overline{a}\langle\overrightarrow{\mathfrak{n}}\rangle\ \mathcal{R}\ \mathfrak{d}$ then $\mathfrak{d} = ✠$ or $\mathfrak{d} = x|\overline{a}\langle\overrightarrow{\mathfrak{m}}\rangle$ and $\mathfrak{n}_i\ \mathcal{R}\ \mathfrak{m}_i$ for $1 \leq i \leq \mathrm{ar}(a)$,
4. if $\mathfrak{n}_0|\overline{a}\langle\overrightarrow{\mathfrak{n}}\rangle\ \mathcal{R}\ \mathfrak{d}$ then $\mathfrak{d} = ✠$ or $\mathfrak{d} = \mathfrak{m}_0|\overline{a}\langle\overrightarrow{\mathfrak{m}}\rangle$ and $\mathfrak{n}_i\ \mathcal{R}\ \mathfrak{m}_i$ for $0 \leq i \leq \mathrm{ar}(a)$,

Remark 1.1.17

For all positive designs $\mathfrak{p}$ and $\mathfrak{p}'$, we have:
- $\Omega \preceq \mathfrak{p} \preceq \maltese$,
- if $\mathfrak{p} \sqsubseteq \mathfrak{p}'$ then $\mathfrak{p} \preceq \mathfrak{p}'$.

We can now state the monotonicity theorem; a proof of the theorem formulated in this form is found in [Ter11].

Theorem 1.1.18 (*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* $(\!|\mathfrak{d}|\!) \preceq (\!|\mathfrak{e}|\!)$.

Remark 1.1.19

Monotonicity makes explicit the fact 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, given a behaviour $\mathbf{B}$, if $\mathfrak{d} \in \mathbf{B}$ and $\mathfrak{d} \preceq \mathfrak{d}'$ then $\mathfrak{d}' \in \mathbf{B}$.

1.1.e Logical Connectives and Internal Completeness

Behaviour constructors – the *logical connectives* – can be applied so as to form compound behaviours. These connectives, coming from (a polarised variant of) MALL, are used for interpreting formulas as behaviours, and will also indeed play the role of type constructors for the types of data and functions. For example, one can already guess that the tensor $\otimes$ is used to form product types, while the linear map $\multimap$ corresponds to functional types constructions. In addition to the usual multiplicative–additive connectives, we also consider a positive shift $\downarrow$ and a negative shift $\uparrow$ to handle polarities.

In this subsection, after defining the connectives we consider, we state the internal completeness theorem for these connectives, which makes explicit the structure of a behaviour constructed via logical connectives. This theorem constitutes a first step – and an important one – towards the precise analysis of the structure of behaviours that we conduct in this thesis, in particular with the study of data types (Chapter 4) and functional types (Chapter 5).

In the rest of this thesis, suppose the signature $\mathcal{S}$ contains the distinct unary names $\blacktriangle, \pi_1, \pi_2$ and the binary name $\wp$. Remember that x_0 is the distinguished variable introduced at the beginning of § 1.1.b.

Notation

- We 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 the set $\{x_0 | \overline{a}\langle \overrightarrow{\mathfrak{n}} \rangle \mid \mathfrak{n}_i \in \mathbf{N}_i\}$, and we write $a(\overrightarrow{x}).\mathbf{P}$ for $\{a(\overrightarrow{x}).\mathfrak{p} \mid \mathfrak{p} \in \mathbf{P}\}$.

- Given 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 1.1.20 (*Logical connectives*)

Given negative behaviours $\mathbf{M}, \mathbf{N}$ and a positive behaviour $\mathbf{P}$, new behaviours can be constructed by applying the **logical connectives** defined by:

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

The connectives $\downarrow$, $\uparrow$, $\oplus$ and $\otimes$ match those defined by Terui [Ter11]. Except $\multimap$, which is defined dualy to $\otimes$, all these connective constructions require that we close a set of designs by bi-orthogonal, to ensure it is a behaviour. But the internal completeness theorem states that this closure is actually unnecessary, i.e., connectives apply on behaviours in a constructive way. For each connective, we present two versions of internal completeness: one concerned with the full behaviour, the other with the behaviour's incarnation.

Theorem 1.1.21 (*Internal completeness*)

Given negative behaviours $\mathbf{M}, \mathbf{N}$ *and a positive behaviour* $\mathbf{P}$*, we have:*

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

A proof of this theorem, in a more general form, can be found in [Ter11].

1.2 Paths

The correspondence between ludics and game semantics has been studied [BF11, FH02, FQ13, FQ16]. It relies on the identification of designs with strategies, by describing a design as a set of interaction traces. Girard represents such interaction traces as *disputes* [Gir01]; technically, a dispute is a finite sequence of actions that records the history of a possible interaction between two orthogonal designs. We choose to adopt the approach of Fouqueré and Quatrini [FQ13] who give a notion of *path* following the same idea, except that paths are defined independently of any design.

The main interest of paths can be observed at the level of behaviours. Given a behaviour $\mathbf{B}$, we can consider all the possible interactions between a design of $\mathbf{B}$ and a design of $\mathbf{B}^{\perp}$, leading to the set of the *visitable paths* of $\mathbf{B}$. This set characterises the behaviour's structure – being however unable to distinguish between the interaction of an infinite design and the one of arbitrarily large finite designs in a behaviour, see the discussion on coinduction in Section 4.4 – therefore it will be subject to much attention in this thesis. In particular, it is needed for defining and proving regularity and purity of behaviours (see Section 1.3).

The point of this section is to adapt the definitions and first results of [FQ13] to the setting of computational ludics (§ 1.2.b). This requires first the recovery of notions from Girard's ludics: *location* and *chronicles* (§ 1.2.a) – the latter being called *views* here, to fit the game semantics approach. Similarities between ludics and HO games are then discussed further in § 1.2.c.

From now on, anytime we deal with paths, should it even be in the context of multi-designs (Chapter 2) or non-linearity (Chapter 6), we adopt Barendregt's variable convention: for designs in a given context, we always assume that:

1. no variable appears both free and bound, and
2. bound variables all have distinct names.

1.2.a Location and Designs as Trees

Location is a primitive idea in Girard's ludics [Gir01] 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, that 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 (§ 1.2.b).

Definition 1.2.1 (*Located action*)

A **located action** κ is one of

$\maltese$	(**daimon**) ,
$x\|\overline{a}\langle x_1, \ldots, x_{\mathrm{ar}(a)}\rangle$	(**proper positive action**) ,
$a_x(x_1, \ldots, x_{\mathrm{ar}(a)})$	(**proper negative action**) ,

where in the last two cases, $a \in \mathcal{S}$ is the **name** of κ, the variables $x, x_1, \ldots, x_{\mathrm{ar}(a)}$ are distinct, x is the **address** of κ and $x_1, \ldots, x_{\mathrm{ar}(a)}$ are the **variables bound by** κ. A **positive action** is either $\maltese$ or a proper positive action; a **negative action** is a proper negative action.

In the following, "action" will always refer to a located action.

Notation

- We will often denote a located action by symbol κ, sometimes indicating polarity

with an exponent: κ^+ or κ^-.
- Like 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 $a_x(x_1,\dots,x_n)$.

Thanks to located actions, we are able to give a true representation of total cut-free designs as trees (forests in general); compared to what is done in [Ter11], we do not need to label the arrows, all the information is contained inside the nodes. Such a representation enhances the readability of designs, thus we shall use it in examples rather than terms; it moreover reminds of Girard's designs [Gir01] which are related to MALL sequent calculus derivations. The intuitions for constructing the tree corresponding to a design, given below, should be read together with Example 1.2.2 as an illustration.

By construction, every total design is a forest of actions, possibly infinite both in height and in width; it is not always a single tree since 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 $\mathfrak{p}_a \neq \Omega$, but by abuse we might write "tree" in all cases. To turn the nodes into located actions, 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, i.e., part of the same sum, are given the same address. A tree is represented bottom-up: the root is at the bottom, and children are above their parent. There is a one-to-one correspondence between well-formed tree representations and designs, up to α-equivalence for negative addresses bound by positive actions.

A more precise and concise way to describe the tree representation of a design uses the views, which we are about to define (§ 1.2.b): simply write all the views of the design bottom-up, merging the common prefixes.

Example 1.2.2

Let $a^2, b^2, c^1, d^0 \in \mathcal{S}$. The following design is represented by the tree of Figure 2:

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

Fresh variables z_1 and z_2 have been picked, both bound by the second node from the bottom, and used as addresses of negative actions just above it. Two of these three negative actions have address z_1, the other has z_2: this depends on the position they occupy in $x_2|\overline{b}\langle\,\cdot\,,\,\cdot\,\rangle$, first or second.

Definition 1.2.3 (*Justification*)

Suppose a design $\mathfrak{d}$ is represented as a tree, and let κ be a proper action of this tree.
- κ is **justified** if its address is bound by an action κ' of opposite polarity appearing below κ in the tree, we then say that κ' is the **justification** of κ; if κ is not justified, it is called **initial**.
- κ is **hereditarily justified** by an action κ' of the tree if there exist actions $\kappa_1,\dots,\kappa_n$ such that $\kappa = \kappa_1$, $\kappa' = \kappa_n$ and for all i such that $1 \leq i < n$, the action κ_i is justified by κ_{i+1}.

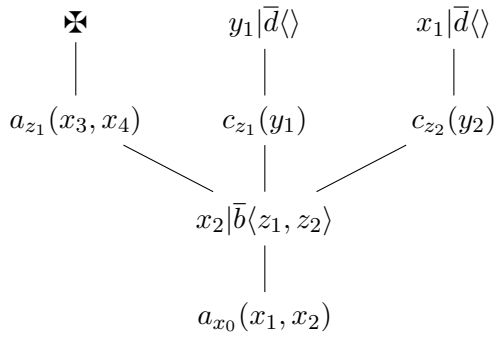


Figure 2: Representation of design $\mathfrak{d}$ from Example 1.2.2.

Note that except the root of a tree, which is always initial, every negative action is justified by the only positive action immediately below it.

1.2.b Views and Paths

Characterising an interaction between two atomic designs of opposite polarities is useful since a behaviour is defined with respect to the possible interactions of its designs, hence the concept of path. A path is a sequence of actions followed in a design during interaction. Here we adapt the notions of *path*, *interaction path* and *visitable path* [FQ13] to ludics *à la Terui*.

Let us start by giving an intuition of what are the views and the paths of a design. On Figure 3 (page 34) are represented a view and a path of design $\mathfrak{d}$ from Example 1.2.2. 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 (see Example 1.2.9).

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 sequence of proper actions $s = \kappa_1 \dots \kappa_n$, write $\overline{s}$ for the sequence $\overline{\kappa_1} \dots \overline{\kappa_n}$.

Definition 1.2.4 (*Dual of a sequence*)

Let s be a finite sequence of proper actions such that if s contains an occurrence of ✠, it is necessarily in last position. The **dual** of s, written $\tilde{s}$, is the sequence defined by:

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

Note that $\overset{\sim}{\tilde{s}} = s$. The notions of **justified**, **hereditarily justified** and **initial** actions (§ 1.2.a) also apply in sequences of actions, where indeed the justification of an action has to be placed *before* it in a sequence (instead of *below* in a tree).

Let us define alternated justified sequences, a basis for both views and paths.

Definition 1.2.5 (*Alternated justified sequence*)

An **alternated justified sequence** (or **aj-sequence**) s is a finite sequence of actions

such that:
- (Alternation) polarities of actions alternate,
- (Daimon) if ✠ appears, it is the last action of s,
- (Linearity) each variable is the address of at most one action in 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.

Notation

- Given an action κ and a set of sequences V, we write κV for $\{\kappa s \mid s \in V\}$.
- We write ϵ for the empty sequence.

In the next definition, remember that the variable x_0 is distinguished.

Definition 1.2.6 (*View, view of a design*)

- A **view** $\mathbb{v}$ is an aj-sequence such that each negative action which is not the first action of $\mathbb{v}$ is justified by the immediate previous action.
- Given a cut-free design $\mathfrak{d}$, the **views of** $\mathfrak{d}$, written $\mathbb{V}[\mathfrak{d}]$, are defined recursively – together with $\mathbb{V}[\mathfrak{n}]_x$ where $\mathfrak{n}$ is a cut-free negative design and x does not appear in $\mathfrak{n}$ – as follows:
  - $\mathbb{V}[\Omega] = \emptyset$,
  - $\mathbb{V}[✠] = \{✠\}$,
  - $\mathbb{V}[x|\overline{a}\langle\overrightarrow{\mathfrak{n}}\rangle] = \bigcup_{i\leq \mathrm{ar}(a)} \kappa_a^+ \mathbb{V}[\mathfrak{n}_i]_{y_i}$ where $\kappa_a^+ = x|\overline{a}\langle\overrightarrow{y}\rangle$ with $\overrightarrow{y}$ fresh,
  - $\mathbb{V}[\mathfrak{n}] = \mathbb{V}[\mathfrak{n}]_{x_0}$,
  - $\mathbb{V}[\sum_{a\in\mathcal{S}} a(\overrightarrow{x^a}).\mathfrak{p}_a]_x = \{\epsilon\} \cup \{\kappa_a^- \mid a \in \mathcal{S} \text{ and } \mathfrak{p}_a \neq \Omega\} \cup \bigcup_{a\in\mathcal{S}} \kappa_a^- \mathbb{V}[\mathfrak{p}_a]$ where $\kappa_a^- = a_x(\overrightarrow{x^a})$.

Note that the views of a design indeed satisfy the definition of view. Views are considered up to α-equivalence, therefore the choice of fresh variables does not matter. This notion corresponds to the one of chronicle in original ludics [Gir01]. We now show how to extract a view from an aj-sequence.

Definition 1.2.7 (*View of a sequence, anti-view of a sequence*)

- The **view of** an aj-sequence is defined inductively by:
  - $\ulcorner\epsilon\urcorner = \epsilon$,
  - $\ulcorner s\kappa^+\urcorner = \ulcorner s\urcorner\kappa^+$,
  - $\ulcorner s\kappa^-\urcorner = \ulcorner s_0\urcorner\kappa^-$ where s_0 is the prefix of s ending on $\mathrm{just}(\kappa^-)$, or $s_0 = \epsilon$ if κ^- initial.
- The **anti-view** of an aj-sequence, noted $\llcorner s \lrcorner$, is defined symmetrically by reversing the role played by polarities; equivalently $\llcorner s \lrcorner = \overbrace{\ulcorner \sim \urcorner}^{s}$.

Informally, taking the view of an aj-sequence consists in the following process, starting from the last action of the sequence: for each action considered, if it is positive go to previous action, if it is negative jump to its justification and erase all the actions in between; stop

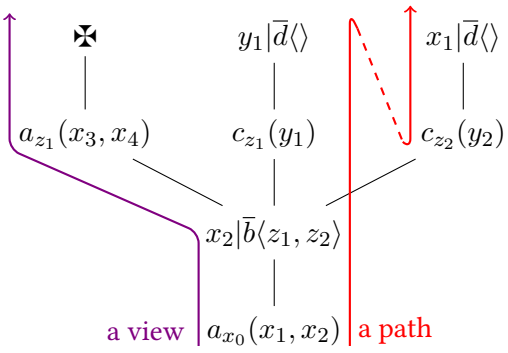


Figure 3: A view and a path of a design.

when reaching either the first action of the sequence or a negative initial action. A problem is that this process might erase the negative justification of a positive action, transforming a justified action into an initial one. The *P-visibility* condition in the definition of path (Definition 1.2.8) ensures we avoid such a situation; symmetrically, *O-visibility* prevents from erasing the positive justification of a negative action while taking the anti-view.

Our definition of path below actually corresponds to a *reversible path* for Fouqueré and Quatrini, since we will not need to consider non-reversible paths. This implies in particular that the dual of a path is a path.

Definition 1.2.8 (*Path, path of a design*)

A **path** s is a positive-ended (or empty) aj-sequence satisfying the following:
(P-visibility) For every prefix $s'\kappa^+$ of s with κ^+ justified, $\mathrm{just}(\kappa^+) \in \ulcorner s' \urcorner$;
(O-visibility) For every prefix $s'\kappa^-$ of s with κ^- justified, $\mathrm{just}(\kappa^-) \in \llcorner s' \lrcorner$.
Given a cut-free design $\mathfrak{d}$, a path s is a **path of** $\mathfrak{d}$ if for every prefix s' of s, $\ulcorner s' \urcorner$ is a view of $\mathfrak{d}$. A non-empty path is **positive** or **negative** depending on the polarity of its first action; the empty path is negative.

Example 1.2.9

Let $\mathfrak{d}$ be the design from Example 1.2.2. The path of $\mathfrak{d}$ indicated on Figure 3 is:

$$s_1 \quad = \quad a_{x_0}(x_1, x_2) \quad x_2|\overline{b}\langle z_1, z_2\rangle \quad c_{z_1}(y_1) \quad y_1|\overline{d}\langle\rangle \quad c_{z_2}(y_2) \quad x_1|\overline{d}\langle\rangle \ .$$

The following sequence is also a promenade in the tree of $\mathfrak{d}$, but not a path of it because O-visibility is not satisfied for the negative action $c_{z_1}(y_1)$:

$$s_2 \quad = \quad a_{x_0}(x_1, x_2) \quad x_2|\overline{b}\langle z_1, z_2\rangle \quad c_{z_2}(y_2) \quad x_1|\overline{d}\langle\rangle \quad c_{z_1}(y_1) \quad y_1|\overline{d}\langle\rangle \ .$$

The definition of path does not refer to interaction, it relies only on conditions external to any design. Let us now establish the link between paths and interaction, keeping in mind that paths are aimed at characterising an interaction between designs. Given

two orthogonal designs $\mathfrak{d}$ and $\mathfrak{e}$, the trace of their interaction can be described as a sequence of actions – the interaction path – corresponding to the succession of reductions performed. The following definition thus assumes that such a sequence is effectively a path, and moreover that it always exists and is unique; these statements will be proved in Chapter 2 (Proposition 2.2.9), since they require more material built on the notion of multi-designs, a generalisation of designs.

Definition 1.2.10 (*Interaction path*)

Let $\mathfrak{d}$ and $\mathfrak{e}$ be cut-free atomic designs such that $\mathfrak{d} \perp \mathfrak{e}$. The **interaction path** of $\mathfrak{d}$ with $\mathfrak{e}$ is the unique path s of $\mathfrak{d}$ such that $\widetilde{s}$ is a path of $\mathfrak{e}$. We write this path $\langle \mathfrak{d} \leftarrow \mathfrak{e} \rangle$.

A good intuition is that $\langle \mathfrak{d} \leftarrow \mathfrak{e} \rangle$ corresponds to the sequence of actions that the interaction between $\mathfrak{d}$ and $\mathfrak{e}$ follows on the side of $\mathfrak{d}$.

Multi-designs will also make it possible to prove (as Proposition 2.2.11) that the convergence of a closed interaction is equivalent to the existence of such a path, which constitutes an alternative – *static* – way of defining orthogonality:

Proposition 1.2.11

Let $\mathfrak{d}$ and $\mathfrak{e}$ be cut-free atomic designs. $\mathfrak{d} \perp \mathfrak{e}$ if and only if there exists a path s of $\mathfrak{d}$ such that $\widetilde{s}$ is a path of $\mathfrak{e}$.

Finally, we consider interaction paths at the level of a behaviour $\mathbf{B}$, leading to a set of visitable paths that describes the possible interactions between a design of $\mathbf{B}$ and a design of $\mathbf{B}^{\perp}$.

Definition 1.2.12 (*Visitable path of a behaviour*)

A path s is **visitable** in a behaviour $\mathbf{B}$ if there exist 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 $V_{\mathbf{B}}$.

Remark 1.2.13

For every behaviour $\mathbf{B}$, $\widetilde{V_{\mathbf{B}}} = V_{\mathbf{B}^{\perp}}$.

1.2.c Ludics vs. Hyland–Ong Games

Game semantics is an interactive semantics for programs and logic, based on the idea of a game between two players. It provided a satisfactory answer to the long open problem of full abstraction for PCF [AJM00, HO00, Nic94]. Programs or proofs are modeled as strategies, while types or formulas correspond to arenas.

There are many similarities between ludics and Hyland–Ong game semantics, indeed: in HO games, strategies are sets of plays that are very much like the paths. More generally, we have the following correspondence:

Action	—	Move
View / Chronicle	—	View
Path / Dispute	—	Play
Design	—	Innocent strategy
Behaviour	—	Arena

This correspondence can be made precise [BF11, FH02]. Let us simply stress the following: the reason why designs correspond to *innocent* strategies is that they can be defined as sets of views; this implies that given a design / strategy, the next positive action / move of a path / play is entirely determined by partial information on the computation history – the view. Note that this will no longer be true when considering designs that are non-deterministic (in Chapter 6).

Although very close to each other, ludics and HO games have an important difference. In HO games, one starts by describing an arena as a set of moves together with an *enabling* relation which indicates what plays are allowed in this game; strategies are then coherent sets of plays following the rules of the game, that is, the enabling. Ludics, on the other hand, adopts the converse approach: designs and interaction are primitive, while behaviours are recovered as sets of designs sharing common ways of playing; this is made possible by the special action ✠ that allows closed interaction.

As remarked by Baelde, Doumane and Saurin [BDS15], this reversal is the reason why it is easier to study fixed points in ludics than in game semantics [Cla09], while game semantics is best suited to model linear logic exponentials than ludics. In a sense, this thesis studies both fixed points and exponentials in ludics, respectively as inductive types (Chapter 4) and non-linearity (Chapter 6), and it appears that we got more fruitful results in the first direction, for the moment.

1.3 Regularity and Purity

We now define *regularity* and *purity*, two properties of behaviours concerned with the visitable paths, i.e., the possible interactions. These properties will be a tool for understanding better the structure of data types (Chapter 4) and functions types (Chapter 5). Recall that the idea behind regularity is that it corresponds to the multiplicative–additive behaviours, and purity ensures type safety thus it is a desirable property for data types.

We start by defining the operations of *shuffle* and *anti-shuffle* on paths, which interleave actions while respecting the alternation of polarities. The idea of the shuffle comes from [BS98] and will be the main ingredient when describing the visitable paths of a tensor in Chapter 3. Recall that a **subsequence** of a sequence $\kappa_1 \dots \kappa_p$ is a sequence $\kappa_{i_1} \dots \kappa_{i_k}$ such that $1 \leq i_1 < \dots < i_k \leq p$, and let $s \restriction s'$ denote the subsequence of s containing only the actions that occur in s'.

Definition 1.3.1 (*Shuffle, Anti-Shuffle*)

Let s and t be paths of same polarity, let S and T be sets of paths of same polarity. The **shuffle** ($\sqcup\!\sqcup$) is defined by:

- $s \sqcup\!\sqcup t = \{u \text{ path formed with actions from } s \text{ and } t \mid u \restriction s = s \text{ and } u \restriction t = t\}$ if s

and t are negative,
- $s \sqcup t = \{\kappa^+ u \text{ path} \mid u \in s' \sqcup t'\}$ if $s = \kappa^+ s'$ and $t = \kappa^+ t'$ are positive with the same first action,
- $s \sqcup t$ is not defined otherwise;
- $S \sqcup T = \{u \text{ path} \mid \exists s \in S, \exists t \in T$ such that $s \sqcup t$ is defined and $u \in s \sqcup t\}$,

The **anti-shuffle** ($\sqcap$) is defined by $s \sqcap t = \widetilde{\widetilde{s} \sqcup \widetilde{t}}$ and $S \sqcap T = \widetilde{\widetilde{S} \sqcup \widetilde{T}}$.

Alternatively, the anti-shuffle can be defined the same way as the shuffle but reversing the role of the polarities.

A path u is in $s \sqcup t$ if it consists of actions from s and t that have been interleaved in a precise way: it must be possible to write

$$\begin{aligned} s &= u_0 s_1 s_2 \dots s_k \ , \\ t &= u_0 t_1 t_2 \dots t_k \ , \\ u &= u_0 s_1 t_1 s_2 t_2 \dots s_k t_k \ , \end{aligned}$$

where u_0 is the maximal common prefix of s and t, and $s_1, \dots, s_k$, $t_1, \dots, t_k$ are sequences of actions beginning on a negative action and ending on a positive one, or empty. Moreover, u must be a path, thus the interleaving has to comply with the justification pointers so that u satisfies the visibility conditions.

Remark 1.3.2

- The shuffle is defined modulo α-equivalence (see example below).
- If non-empty, the common prefix u_0 ends on a positive action; if u_0 is empty then s and t are negative paths.
- After the common prefix u_0, the rests of the paths s and t are disjoint.
- If s and t are both $\maltese$-ended, then $s \sqcup t$ is empty, unless $s = t$ and in this case $s \sqcup t = \{s\} = \{t\}$.

Example 1.3.3

Consider the following paths:

$$\begin{aligned} s &= a_{x_0}(x_1, x_2) \quad x_2|\overline{b}\langle x_3, x_4\rangle \quad c_{x_3}() \quad x_1|\overline{d}\langle\rangle \ , \\ t &= a_{x_0}(y_1, y_2) \quad y_2|\overline{b}\langle y_3, y_4\rangle \quad e_{y_4}(z) \quad z|\overline{f}\langle\rangle \ . \end{aligned}$$

They have a common prefix $a_{x_0}(x_1, x_2) \ \ x_2|\overline{b}\langle x_3, x_4\rangle$ (modulo α-equivalence). By interleaving their actions, we obtain the two following sequences:

$$\begin{aligned} u &= a_{x_0}(x_1, x_2) \quad x_2|\overline{b}\langle x_3, x_4\rangle \quad e_{x_4}(z) \quad z|\overline{f}\langle\rangle \quad c_{x_3}() \quad x_1|\overline{d}\langle\rangle \ , \\ u' &= a_{x_0}(x_1, x_2) \quad x_2|\overline{b}\langle x_3, x_4\rangle \quad c_{x_3}() \quad x_1|\overline{d}\langle\rangle \quad e_{x_4}(z) \quad z|\overline{f}\langle\rangle \end{aligned}$$

(where the address of the action of name e has been changed according to α-renaming). We have $u \in s \sqcup t$ but $u' \notin s \sqcup t$ since u' is not a path, indeed: O-visibility is not satisfied for action $e_{x_4}(z)$.

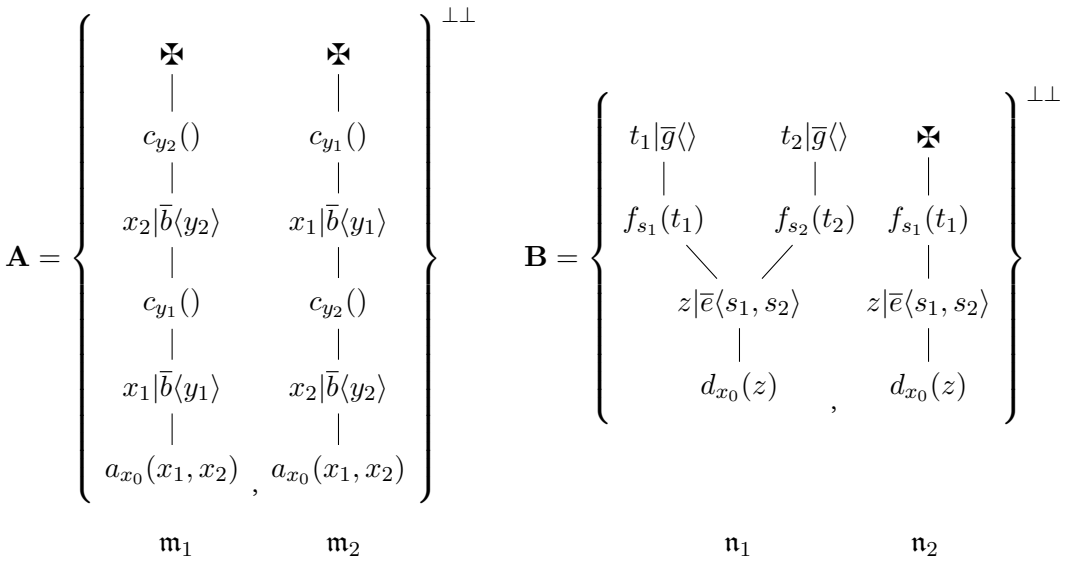


Figure 4: A regular and a non-regular behaviour

Definition 1.3.4 (*Regularity*)

A behaviour $\mathbf{B}$ is **regular** if the following conditions are satisfied:

- for every design $\mathfrak{d} \in |\mathbf{B}|$ and every path s of $\mathfrak{d}$, $s \in V_{\mathbf{B}}$,
- for every design $\mathfrak{d} \in |\mathbf{B}^{\perp}|$ and every 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 (i.e., $V_{\mathbf{B}}$ is stable under $\sqcup\!\sqcup$ and $\sqcap$).

The intuition is that 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 $\mathbf{B}^{\perp}$. Not all the behaviours are regular, as shown in the following example.

Example 1.3.5

Consider the two behaviours $\mathbf{A}$ and $\mathbf{B}$ generated as indicated on Figure 4 ($\mathbf{A}$ is the same behaviour as in Example 1.1.11). The behaviour $\mathbf{A}$ is regular: we could check that all the conditions are satisfied. On the other hand, $\mathbf{B}$ is not regular: we have $\mathfrak{n}_1 \in |\mathbf{B}|$ but the path corresponding to the right branch of $\mathfrak{n}_1$ is not visitable in $\mathbf{B}$. Indeed, the presence of design $\mathfrak{n}_2$ in $\mathbf{B}$ forces the interaction to visit first the left branch of $\mathfrak{n}_1$; if we wanted to visit the right one first, we would need a counter-design of the form

$$\mathfrak{p} = x_0|\overline{d}\langle e(s_1, s_2).(s_2|\overline{f}\langle \mathfrak{n} \rangle)\rangle$$

but such a design is not orthogonal to $\mathfrak{n}_2$, thus is not in $\mathbf{B}^{\perp}$.

Remark 1.3.6

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.

Now, in order to define purity, consider the following preliminary definition.

Definition 1.3.7 (*Extensible, maximal visitable path*)

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

- A $\maltese$-free path $s \in V_{\mathbf{B}}$ is **extensible** in $V_{\mathbf{B}}$ if there exists a proper negative action κ^- such that $s\kappa^-\maltese \in V_{\mathbf{B}}$; in this case the sequence $s\kappa^-$ is called a **witness** of extensibility.
- A $\maltese$-ended path $t\maltese \in V_{\mathbf{B}}$ is **extensible** in $V_{\mathbf{B}}$ if there exists a proper positive action κ^+ such that $t\kappa^+ \in V_{\mathbf{B}}$; similarly, $t\kappa^+$ is a **witness** of extensibility.
- Given $s, s' \in V_{\mathbf{B}}$, the path s **extends** the path s' in $V_{\mathbf{B}}$ if either $s = s'$ or s' is extensible in $V_{\mathbf{B}}$ with witness a prefix of s.
- A path in $V_{\mathbf{B}}$ is **maximal** in $V_{\mathbf{B}}$ if it is not extensible in $V_{\mathbf{B}}$.

Definition 1.3.8 (*Purity*)

A behaviour $\mathbf{B}$ is **pure** if all the $\maltese$-ended paths in $V_{\mathbf{B}}$ are extensible, in other words if there is no maximal $\maltese$-ended path.

Purity ensures that when an interaction encounters $\maltese$, this does not correspond to a real error but rather to a partial computation, as it is possible to continue this interaction. Note that we cannot require behaviours to be entirely $\maltese$-free, daimons being necessarily present in all behaviours. Indeed, given $\mathfrak{d} \in \mathbf{B}$, for any design $\mathfrak{d}'$ obtained from $\mathfrak{d}$ by cutting off branches and replacing them with $\maltese$, we have $\mathfrak{d}' \in \mathbf{B}$ since $\mathfrak{d} \preceq \mathfrak{d}'$; in other word, the following property always holds: if $s\kappa^+ \in V_{\mathbf{B}}$ then $s\maltese \in V_{\mathbf{B}}$.

We prove in Chapters 3 and 4 that purity and regularity of behaviours are preserved when applying logical connectives $\downarrow, \uparrow, \oplus, \otimes$ or when taking a least fixed point. The connective $\multimap$, however, might break purity: this is detailed in Chapter 5. The computational and logical meaning of these two properties will be made clearer then.

2 | Multi-Designs

Designs are not sufficient in order to prove results by induction on an interaction, that is, by induction on the length of an interaction path. The reason is that, given designs $\mathfrak{d}$ and $\mathfrak{e}$ such that $\mathfrak{d} \perp \mathfrak{e}$, the path $\langle \mathfrak{d} \leftarrow \mathfrak{e} \rangle$ has been defined statically, for the moment. We would rather like to define it as

$$\langle \mathfrak{d} \leftarrow \mathfrak{e} \rangle = \kappa \langle \mathfrak{d}' \leftarrow \mathfrak{e}' \rangle$$

where κ is the only action at the base of $\mathfrak{d}$ such that its dual $\overline{\kappa}$ is at the base of $\mathfrak{e}$, and where $\mathfrak{d}'$ and $\mathfrak{e}'$ are the designs obtained after one reduction step of the interaction between $\mathfrak{d}$ and $\mathfrak{e}$. The problem is that there may not exist such designs $\mathfrak{d}'$ and $\mathfrak{e}'$.

Consider for example the designs $\mathfrak{d}$ and $\mathfrak{e}$ on Figure 5. We have $\mathfrak{d} \perp \mathfrak{e}$ and the first action of $\langle \mathfrak{d} \leftarrow \mathfrak{e} \rangle$ is indeed $\kappa = x_0 | \overline{a} \langle x, y \rangle$. But one reduction step gives

$$\mathfrak{d}[\mathfrak{e}/x_0] \rightsquigarrow \mathfrak{e}'[\mathfrak{d}'_1/x, \mathfrak{d}'_2/y] \ ,$$

in other words the design $\mathfrak{d}$ leads to two designs. In this particular case, we might want to refine our previous idea by setting

$$\langle \mathfrak{d} \leftarrow \mathfrak{e} \rangle = \kappa \langle \{\mathfrak{d}'_1, \mathfrak{d}'_2\} \leftarrow \mathfrak{e}' \rangle \ .$$

This example justifies the necessity of considering not only designs but also *multi-designs* in order to deal with such partial computations. In particular, it will enable us to give a dynamic (i.e., inductive) definition of the interaction path (Definition 2.2.3).

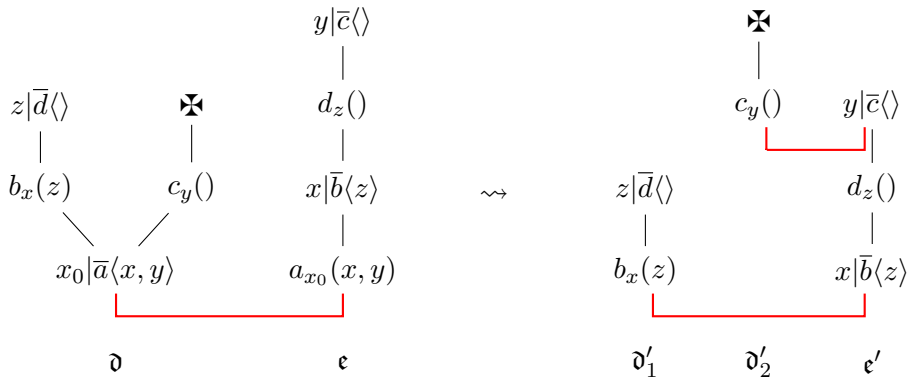


Figure 5: Why we need multi-designs

From the beginnings of ludics [Gir01], the orthogonality between a design and a set of designs has been considered. This allows for an extended notion of behaviour; indeed, note that the behaviours we introduced in the previous chapter (Definition 1.1.10) are very limited because of the restriction to atomic designs (we should actually call them *atomic behaviours*). Though inspired by the *anti-designs* of Terui [Ter11], the multi-designs go further by giving the possibility to describe the interaction between two sets of designs. This generalisation should however be taken as a technical investigation for our purpose, rather than an interesting concept in itself.

The present chapter thus lifts the framework of ludics to multi-designs, so as to prove properties of the interaction path. More precisely, after introducing the necessary notions in Section 2.1, we show the following in Section 2.2:

- the existence and uniqueness of the interaction path between two orthogonal multi-designs (Proposition 2.2.9),
- the equivalence between the existence of such a path and the orthogonality of two multi-designs (Proposition 2.2.11, a generalisation of Proposition 1.2.11),
- the *associativity for paths* (Proposition 2.2.12).

These results are needed for the following, in particular for Chapter 3. Their proofs are rather technical, with many lemmas, so the reader intuitively convinced may not necessarily need to read them throughout.

2.1 A Generalisation of Designs

The aim of this section is to get a notion as general as possible of multi-design. In particular, the *closed-compatible* multi-designs capture exactly what remains of a closed interaction between two designs after some steps of reductions (kind of *residues*). Going back to Figure 5 above, the interaction between $\{\mathfrak{d}'_1/x, \mathfrak{d}'_2/y\}$ and $\mathfrak{e}'$ is an example of closed-compatibility: the design – noted $\mathfrak{Cut}_{\{\mathfrak{d}'_1/x, \mathfrak{d}'_2/y\}|\mathfrak{e}'}$ – obtained after performing the substitutions is closed, thus the interaction leads either to $\maltese$ or Ω. This way, we recover notions of orthogonality and behaviour for multi-designs matching the ones for designs.

2.1.a Multi-Designs

The notion of multi-design introduced below generalises the one of anti-design [Ter11], and in particular it generalises designs. Interaction between two compatible multi-designs $\mathfrak{D}$ and $\mathfrak{E}$ corresponds to the elimination of cuts in the multi-design $\mathfrak{Cut}_{\mathfrak{D}|\mathfrak{E}}$, which is obtained by substituting designs of $\mathfrak{D}$ in designs of $\mathfrak{E}$ and vice versa.

Definition 2.1.1 (*Multi-design*)

- A **negative multi-design** is a set

$$\{(x_1, \mathfrak{n}_1), \dots, (x_k, \mathfrak{n}_k)\}$$

where $x_1, \dots, x_k$ are distinct variables and $\mathfrak{n}_1, \dots, \mathfrak{n}_k$ are negative designs, such that for all i, j with $1 \leq i \neq j \leq k$ we have $\mathrm{fv}(\mathfrak{n}_i) \cap \{x_1, \dots, x_k\} = \emptyset$ and

$\mathrm{fv}(\mathfrak{n}_i) \cap \mathrm{fv}(\mathfrak{n}_j) = \emptyset$.

- A **positive multi-design** is a set

$$\{\mathfrak{p}, (x_1, \mathfrak{n}_1), \dots, (x_k, \mathfrak{n}_k)\}$$

where $\{(x_1, \mathfrak{n}_1), \dots, (x_k, \mathfrak{n}_k)\}$ is a negative multi-design and $\mathfrak{p}$ is a positive design such that $\mathrm{fv}(\mathfrak{p}) \cap \{x_1, \dots, x_k\} = \emptyset$, and for all $1 \leq i \leq k$, $\mathrm{fv}(\mathfrak{p}) \cap \mathrm{fv}(\mathfrak{n}_i) = \emptyset$.

Notation

- We use $\mathfrak{D}, \mathfrak{E}, \dots$ to denote multi-designs of any polarity, $\mathfrak{M}, \mathfrak{N}, \dots$ for negative ones and $\mathfrak{P}, \mathfrak{Q}, \dots$ for positive ones.
- A pair $(x, \mathfrak{n})$ in a multi-design is denoted by $\mathfrak{n}/x$ or $(\mathfrak{n}/x)$; hence a negative multi-design will be written $\{\mathfrak{n}_1/x_1, \dots, \mathfrak{n}_k/x_k\}$ (or even $\{\overrightarrow{\mathfrak{n}/x}\}$), a positive one $\{\mathfrak{p}, \mathfrak{n}_1/x_1, \dots, \mathfrak{n}_k/x_k\}$, and we write $(\mathfrak{n}/x) \in \mathfrak{D}$ instead of $(x, \mathfrak{n}) \in \mathfrak{D}$. This notation makes the parallel with substitution: if $\mathfrak{N} = \{\mathfrak{n}_1/x_1, \dots, \mathfrak{n}_k/x_k\}$ and $\mathfrak{d}$ is a design, then we can write $\mathfrak{d}[\mathfrak{N}]$ for the substitution $\mathfrak{d}[\mathfrak{n}_1/x_1, \dots, \mathfrak{n}_k/x_k]$.
- By abuse, we might even write $\mathfrak{n} \in \mathfrak{D}$ when the variable associated to $\mathfrak{n}$ in the multi-design $\mathfrak{D}$ does not matter; thus when writing "let $\mathfrak{d} \in \mathfrak{D}$", the design $\mathfrak{d}$ can be either positive or negative associated with a variable in $\mathfrak{D}$.
- A design can be viewed as a multi-design: a positive design $\mathfrak{p}$ corresponds to the positive multi-design $\{\mathfrak{p}\}$, and a negative design $\mathfrak{n}$ to the negative multi-design $\{\mathfrak{n}/x_0\}$, where x_0 is the same distinguished variable we introduced for atomic designs. In this case, notations $\mathfrak{p}$ and $\mathfrak{n}$ can replace $\{\mathfrak{p}\}$ and $\{\mathfrak{n}/x_0\}$ respectively.

Note that if $\mathfrak{D}$ and $\mathfrak{E}$ are multi-designs, $\mathfrak{D} \cup \mathfrak{E}$ is not always a multi-design.

Definition 2.1.2 (*Normal form*)

Let $\mathfrak{D}$ be a multi-design. Its **normal form** is the cut-free multi-design defined by

$$(\!|\mathfrak{D}|\!) = \{((\!|\mathfrak{n}|\!)/x) \mid (\mathfrak{n}/x) \in \mathfrak{D}\} \cup \{(\!|\mathfrak{p}|\!) \mid \mathfrak{p} \in \mathfrak{D}\} \ .$$

The associativity theorem naturally extends to multi-designs as follows.

Theorem 2.1.3 (*Multi-associativity*)

Let $\mathfrak{D}$ be a multi-design and $\mathfrak{n}_1, \dots, \mathfrak{n}_k$ be negative designs.

$$(\!|\mathfrak{D}[\mathfrak{n}_1/y_1, \dots, \mathfrak{n}_k/y_k]|\!) = (\!|(\!|\mathfrak{D}|\!)[(\!|\mathfrak{n}_1|\!)/y_1, \dots, (\!|\mathfrak{n}_k|\!)/y_k]|\!) \ .$$

Proof. Immediate from the definition of the normal form of a multi-design (Definition 2.1.2) and simple associativity (Theorem 1.1.7). □

Definition 2.1.4 (*Free variables, negative places*)

Let $\mathfrak{D}$ be a multi-design.

- The **free variables** of $\mathfrak{D}$ are $\mathrm{fv}(\mathfrak{D}) = \bigcup_{\mathfrak{d}\in\mathfrak{D}} \mathrm{fv}(\mathfrak{d})$.
- The **negative places** of $\mathfrak{D}$ are $\mathrm{np}(\mathfrak{D}) = \{x \mid \exists \mathfrak{n}\ (\mathfrak{n}/x) \in \mathfrak{D}\}$.

In Definition 2.1.1, the condition "for all $1 \leq i \leq k$, $\mathrm{fv}(\mathfrak{n}_i) \cap \{x_1, \dots, x_k\} = \emptyset$" (together with the similar condition for $\mathfrak{p}$ in the positive case) can thus be rephrased as "$\mathrm{fv}(\mathfrak{D}) \cap \mathrm{np}(\mathfrak{D}) = \emptyset$". When two multi-designs $\mathfrak{D}$ and $\mathfrak{E}$ interact, this condition ensures that a substitution specified in $\mathfrak{D}$ or in $\mathfrak{E}$ creates a cut between a design from $\mathfrak{D}$ and a design from $\mathfrak{E}$, and never between two designs on the same side. This is exactly the form of interaction we want in the following: an interaction with two distinct sides. But in order to talk about interaction between two multi-designs, we must first determine when two multi-designs are *compatible*, i.e., when we can define substitution between them in a unique way, without ambiguity, which is not the case in general.

2.1.b Compatibility, Orthogonality and Behaviours

Definition 2.1.5 (*Compatible, closed-compatible*)

Let $\mathfrak{D}$ and $\mathfrak{E}$ be multi-designs.

- $\mathfrak{D}$ and $\mathfrak{E}$ are **compatible** if they satisfy the following conditions:
  - $\mathrm{fv}(\mathfrak{D}) \cap \mathrm{fv}(\mathfrak{E}) = \mathrm{np}(\mathfrak{D}) \cap \mathrm{np}(\mathfrak{E}) = \emptyset$,
  - either they are both negative and there exists $x \in \mathrm{np}(\mathfrak{D}) \cup \mathrm{np}(\mathfrak{E})$ such that $x \notin \mathrm{fv}(\mathfrak{D}) \cup \mathrm{fv}(\mathfrak{E})$, or they are of opposite polarities.
- $\mathfrak{D}$ and $\mathfrak{E}$ are **closed-compatible** if they are of opposite polarities, compatible, and satisfying $\mathrm{fv}(\mathfrak{D}) = \mathrm{np}(\mathfrak{E})$ and $\mathrm{fv}(\mathfrak{E}) = \mathrm{np}(\mathfrak{D})$.

Intuitively, compatible means that we are able to define the multi-design $\mathfrak{Cut}_{\mathfrak{D}|\mathfrak{E}}$ corresponding to the interaction between $\mathfrak{D}$ and $\mathfrak{E}$, and closed-compatible means that this multi-design is a closed design: there are no free variables left, nor negative designs that would not have been substituted. $\mathfrak{Cut}_{\mathfrak{D}|\mathfrak{E}}$ is what we obtain after performing all the substitutions possible between designs of $\mathfrak{D}$ and designs of $\mathfrak{E}$. For example:

- if $\mathfrak{p}$ and $\mathfrak{n}$ are atomic then $\mathfrak{Cut}_{\mathfrak{p}|\mathfrak{n}} = \mathfrak{p}[\mathfrak{n}/x_0]$;
- if $\mathfrak{P} = \{\mathfrak{p}, \mathfrak{m}_1/x_1, \dots, \mathfrak{m}_k/x_k, \mathfrak{m}'_1/x'_1, \dots, \mathfrak{m}'_p/x'_p\}$ where $\mathfrak{p}$ is an atomic positive design and $\mathfrak{n}$ is a negative design such that $x_1, \dots x_k \in \mathrm{fv}(\mathfrak{n})$ and $x'_1, \dots x'_p \notin \mathrm{fv}(\mathfrak{n})$, then $\mathfrak{Cut}_{\mathfrak{P}|\mathfrak{n}} = \{\mathfrak{p}[\mathfrak{n}[\overrightarrow{\mathfrak{m}/x}]/x_0], \overrightarrow{\mathfrak{m}'/x'}\}$.

Formally, it is defined as follows.

Definition 2.1.6 ($\mathfrak{Cut}$)

Let $\mathfrak{D}$ and $\mathfrak{E}$ be compatible multi-designs. $\mathfrak{Cut}_{\mathfrak{D}|\mathfrak{E}}$ is a multi-design defined by induction on the number of designs in $\mathfrak{E}$:

$$\mathfrak{Cut}_{\mathfrak{D}|\emptyset} = \mathfrak{D}\ , \tag{1}$$

$$\mathfrak{Cut}_{\mathfrak{D}|\mathfrak{E}} = \mathfrak{Cut}_{(\mathfrak{D}\setminus S)\cup\{\mathfrak{p}[S]\}\ |\ \mathfrak{E}\setminus\{\mathfrak{p}\}} \qquad \text{if } \mathfrak{p} \in \mathfrak{E}\ , \tag{2}$$

$$\mathfrak{Cut}_{\mathfrak{D}|\mathfrak{E}} = \mathfrak{Cut}_{(\mathfrak{D}\setminus S)\cup\{\mathfrak{n}[S]/x\}\ |\ \mathfrak{E}\setminus\{\mathfrak{n}/x\}} \qquad \text{if } (\mathfrak{n}/x)\in\mathfrak{E} \text{ and } x\notin \mathrm{fv}(\mathfrak{D}) \ , \tag{3}$$

$$\mathfrak{Cut}_{\mathfrak{D}|\mathfrak{E}} = \mathfrak{Cut}_{(\mathfrak{D}\setminus S)[\mathfrak{n}[S]/x]\ |\ \mathfrak{E}\setminus\{\mathfrak{n}/x\}} \qquad \text{if } (\mathfrak{n}/x)\in\mathfrak{E} \text{ and } x\in \mathrm{fv}(\mathfrak{D}) \ , \tag{4}$$

where $S \quad = \{(\mathfrak{m}/y)\in\mathfrak{D} \mid y\in\mathrm{fv}(\mathfrak{p})\}$ in (2),
$= \{(\mathfrak{m}/y)\in\mathfrak{D} \mid y\in\mathrm{fv}(\mathfrak{n})\}$ in (3) and (4).

The successive pairs of compatible (resp. closed-compatible) multi-designs stay compatible (resp. closed-compatible) after one step of the definition, thus this is well defined. Moreover, if $\mathfrak{D}$ and $\mathfrak{E}$ are closed-compatible then, according to the base case, $\mathfrak{Cut}_{\mathfrak{D}|\mathfrak{E}}$ is a closed design.

Example 2.1.7

Recall the designs of Figure 5 at the beginning of the chapter. We have:

$$\mathfrak{Cut}_{\{\mathfrak{d}'_1/x,\mathfrak{d}'_2/y\}|\mathfrak{e}'} = \mathfrak{Cut}_{\mathfrak{e}'[\mathfrak{d}'_1/x,\mathfrak{d}'_2/y]|\emptyset} = \mathfrak{e}'[\mathfrak{d}'_1/x,\mathfrak{d}'_2/y]$$

by applying step 2 and then step 1 of Definition 2.1.6; proceeding in a different order:

$$\mathfrak{Cut}_{\mathfrak{e}'|\{\mathfrak{d}'_1/x,\mathfrak{d}'_2/y\}} = \mathfrak{Cut}_{\mathfrak{e}'[\mathfrak{d}'_1/x]|\{\mathfrak{d}'_2/y\}} = \mathfrak{Cut}_{\mathfrak{e}'[\mathfrak{d}'_1/x,\mathfrak{d}'_2/y]|\emptyset} = \mathfrak{e}'[\mathfrak{d}'_1/x,\mathfrak{d}'_2/y]$$

by applying step 4 twice and then step 1. Here we have $S=\emptyset$ every time, this is in fact a very simple example.

We can now extend the notions of orthogonality and behaviours to multi-designs. Note that, for two multi-designs $\mathfrak{D}$ and $\mathfrak{E}$ to be orthogonal, it is necessary that they are closed-compatible: the negative places of $\mathfrak{D}$ (resp. $\mathfrak{E}$) must exactly match the free variables of $\mathfrak{E}$ (resp. $\mathfrak{D}$).

Definition 2.1.8 (*Orthogonality*)

Let $\mathfrak{D}$ and $\mathfrak{E}$ be closed-compatible multi-designs. $\mathfrak{D}$ and $\mathfrak{E}$ are **orthogonal**, noted $\mathfrak{D}\perp\mathfrak{E}$, if $(\!|\mathfrak{Cut}_{\mathfrak{D}|\mathfrak{E}}|\!) = \maltese$.

Definition 2.1.9 (*Behaviour*)

A set $\mathbf{B}$ of cut-free multi-designs of same polarity is a **behaviour** if $\mathbf{B}^{\perp\perp}=\mathbf{B}$.

This definition generalises the behaviours of designs (Definition 1.1.10). The conception of multi-designs was aimed at getting the most general notion of behaviour in ludics, and we claim that we have it.

2.1.c First Properties

We begin with a proposition derived from multi-associativity.

Proposition 2.1.10

Let $\mathfrak{D},\mathfrak{E}$ be compatible multi-designs. We have $(\!|\mathfrak{Cut}_{\mathfrak{D}|\mathfrak{E}}|\!) = (\!|\mathfrak{Cut}_{(\!|\mathfrak{D}|\!)|(\!|\mathfrak{E}|\!)}|\!)$.

Proof. By induction on $\mathfrak{E}$:

- If $\mathfrak{E} = \emptyset$ then $(\!|\mathfrak{Cut}_{\mathfrak{D}|\emptyset}|\!) = (\!|\mathfrak{D}|\!) = (\!|\mathfrak{Cut}_{(\!|\mathfrak{D}|\!)|\emptyset}|\!) = (\!|\mathfrak{Cut}_{(\!|\mathfrak{D}|\!)|(\!|\emptyset|\!)}|\!)$.
- If $\mathfrak{p} \in \mathfrak{E}$, write $\mathfrak{E}' = \mathfrak{E} \setminus \{\mathfrak{p}\}$ and let

$$S = \{\mathfrak{m}_1/y_1, \ldots, \mathfrak{m}_k/y_k\} = \{(\mathfrak{m}/y) \in \mathfrak{D} \mid y \in \mathrm{fv}(\mathfrak{p})\} \ .$$

  By definitions of the normal form of multi-designs (Definition 2.1.2) and of $\mathfrak{Cut}_{|\cdot|}$ (Definition 2.1.6), and using associativity (Theorem 2.1.3), we have:

$$\begin{aligned}
(\!|\mathfrak{Cut}_{\mathfrak{D}|\mathfrak{E}}|\!) &= (\!|\mathfrak{Cut}_{(\mathfrak{D}\setminus S)\cup\{\mathfrak{p}[S]\}|\mathfrak{E}'}|\!) && \text{by Def. 2.1.6}\\
&= (\!|\mathfrak{Cut}_{(\!|(\mathfrak{D}\setminus S)\cup\{\mathfrak{p}[S]\}|\!)|(\!|\mathfrak{E}'|\!)}|\!) && \text{by induction hypothesis}\\
&= (\!|\mathfrak{Cut}_{(\!|\mathfrak{D}\setminus S|\!)\cup\{((\!|\mathfrak{p}|\!))[(\!|\mathfrak{m}_1|\!)/y_1,\ldots,(\!|\mathfrak{m}_k|\!)/y_k]|\!)\}|(\!|\mathfrak{E}'|\!)}|\!) && \text{by Def. 2.1.2 and Thm. 2.1.3}\\
&= (\!|\mathfrak{Cut}_{(\!|(\!|\mathfrak{D}\setminus S|\!)\cup\{(\!|\mathfrak{p}|\!)[(\!|\mathfrak{m}_1|\!)/y_1,\ldots,(\!|\mathfrak{m}_k|\!)/y_k]\}|\!)|(\!|\mathfrak{E}'|\!)}|\!) && \text{by Def. 2.1.2}\\
&= (\!|\mathfrak{Cut}_{(\!|\mathfrak{D}\setminus S|\!)\cup\{(\!|\mathfrak{p}|\!)[(\!|\mathfrak{m}_1|\!)/y_1,\ldots,(\!|\mathfrak{m}_k|\!)/y_k]\}|(\!|\mathfrak{E}'|\!)}|\!) && \text{by induction hypothesis}\\
&= (\!|\mathfrak{Cut}_{(\!|\mathfrak{D}|\!)|(\!|\mathfrak{E}|\!)}|\!) && \text{by Def. 2.1.2 and 2.1.6.}
\end{aligned}$$

- If $(\mathfrak{n}/x) \in \mathfrak{E}$ with $x \notin \mathrm{fv}(\mathfrak{D})$, write $\mathfrak{E}' = \mathfrak{E} \setminus \{\mathfrak{n}/x\}$ and the reasoning is similar as above with $S = \{(\mathfrak{m}/y) \in \mathfrak{D} \mid y \in \mathrm{fv}(\mathfrak{n})\}$.
- If $(\mathfrak{n}/x) \in \mathfrak{E}$ with $x \in \mathrm{fv}(\mathfrak{D})$, write $\mathfrak{E}' = \mathfrak{E} \setminus \{\mathfrak{n}/x\}$ and let

$$S = \{\mathfrak{m}_1/y_1, \ldots, \mathfrak{m}_k/y_k\} = \{(\mathfrak{m}/y) \in \mathfrak{D} \mid y \in \mathrm{fv}(\mathfrak{n})\} \ .$$

  We have:

$$\begin{aligned}
(\!|\mathfrak{Cut}_{\mathfrak{D}|\mathfrak{E}}|\!) &= (\!|\mathfrak{Cut}_{(\mathfrak{D}\setminus S)[\mathfrak{n}[S]/x]|\mathfrak{E}'}|\!) && \text{by Def. 2.1.6}\\
&= (\!|\mathfrak{Cut}_{(\!|(\mathfrak{D}\setminus S)[\mathfrak{n}[S]/x]|\!)|(\!|\mathfrak{E}'|\!)}|\!) && \text{by induction hypothesis}\\
&= (\!|\mathfrak{Cut}_{(\!|(\!|\mathfrak{D}\setminus S|\!)[((\!|\mathfrak{n}|\!))[(\!|\mathfrak{m}_1|\!)/y_1,\ldots,(\!|\mathfrak{m}_k|\!)/y_k]|\!)/x]|\!)|(\!|\mathfrak{E}'|\!)}|\!) && \text{using Thm. 2.1.3 twice}\\
&= (\!|\mathfrak{Cut}_{(\!|(\!|\mathfrak{D}\setminus S|\!)[(\!|\mathfrak{n}|\!)[(\!|\mathfrak{m}_1|\!)/y_1,\ldots,(\!|\mathfrak{m}_k|\!)/y_k]/x]|\!)|(\!|\mathfrak{E}'|\!)}|\!) && \text{by Thm. 2.1.3}\\
&= (\!|\mathfrak{Cut}_{\{(\!|\mathfrak{D}\setminus S|\!)[(\!|\mathfrak{n}|\!)[(\!|\mathfrak{m}_1|\!)/y_1,\ldots,(\!|\mathfrak{m}_k|\!)/y_k]/x]|(\!|\mathfrak{E}'|\!)}|\!) && \text{by induction hypothesis}\\
&= (\!|\mathfrak{Cut}_{(\!|\mathfrak{D}|\!)|(\!|\mathfrak{E}|\!)}|\!) && \text{by Def. 2.1.2 and 2.1.6.}
\end{aligned}$$

□

Now we prove two useful lemmas.

Lemma 2.1.11

Let $\mathfrak{D}, \mathfrak{E}$ be compatible multi-designs. We have $\mathfrak{Cut}_{\mathfrak{D}|\mathfrak{E}} = \mathfrak{Cut}_{\mathfrak{E}|\mathfrak{D}}$.

Proof. By induction on the number n of variables in $(\mathrm{fv}(\mathfrak{D}) \cap \mathrm{np}(\mathfrak{E})) \cup (\mathrm{fv}(\mathfrak{E}) \cap \mathrm{np}(\mathfrak{D}))$.

- If $n = 0$ then $\mathfrak{Cut}_{\mathfrak{D}|\mathfrak{E}} = \mathfrak{Cut}_{\mathfrak{E}|\mathfrak{D}} = \mathfrak{D} \cup \mathfrak{E}$.
- Let $n > 0$ and suppose the property is satisfied for all $k < n$. Without loss of generality suppose there exists $x \in (\mathrm{fv}(\mathfrak{D}) \cap \mathrm{np}(\mathfrak{E}))$. Thus there exists $\{\mathfrak{n}/x\} \in \mathfrak{E}$. Let us write $\mathfrak{E}' = \mathfrak{E} \setminus \{\mathfrak{n}/x\}$. Let $S = \{(\mathfrak{m}/y) \in \mathfrak{D} \mid y \in \mathrm{fv}(\mathfrak{n})\}$.

– If $S = \emptyset$, let $\mathfrak{d} \in \mathfrak{D}$ be the design such that $x \in \mathrm{fv}(\mathfrak{d})$, and let us write $\mathfrak{D}' = \mathfrak{D} \setminus \{\mathfrak{d}\}$. If $\mathfrak{d}$ is positive then:

$$\begin{aligned}
\mathfrak{Cut}_{\mathfrak{D}|\mathfrak{E}} &= \mathfrak{Cut}_{\mathfrak{D}'\cup\{\mathfrak{d}[\mathfrak{n}/x]\}|\mathfrak{E}'} && \text{by one step 4 of Def. 2.1.6}\\
&= \mathfrak{Cut}_{\mathfrak{E}'|\mathfrak{D}'\cup\{\mathfrak{d}[\mathfrak{n}/x]\}} && \text{by induction hypothesis}\\
&= \mathfrak{Cut}_{(\mathfrak{E}'\setminus T')\cup\{\mathfrak{d}[\mathfrak{n}/x,T']\}|\mathfrak{D}'} && \text{by one step 2 of Def. 2.1.6,}
\end{aligned}$$

where $T' = \{(\mathfrak{m}/y) \in \mathfrak{E}' \mid y \in \mathrm{fv}(\mathfrak{d}[\mathfrak{n}/x])\}$. Let $T = \{(\mathfrak{m}/y) \in \mathfrak{E} \mid y \in \mathrm{fv}(\mathfrak{d})\}$, we have $T = T' \cup \{\mathfrak{n}/x\}$, indeed: $\mathrm{fv}(\mathfrak{d}[\mathfrak{n}/x]) = (\mathrm{fv}(\mathfrak{d}) \setminus \{x\}) \cup \mathrm{fv}(\mathfrak{n})$, where $\mathrm{fv}(\mathfrak{n}) \cap \mathrm{np}(\mathfrak{E}) = \emptyset$ by definition of a multi-design, thus also $\mathrm{fv}(\mathfrak{n}) \cap \mathrm{np}(\mathfrak{E}') = \emptyset$. Therefore:

$$\mathfrak{Cut}_{(\mathfrak{E}'\setminus T')\cup\{\mathfrak{d}[\mathfrak{n}/x,T']\}|\mathfrak{D}'} = \mathfrak{Cut}_{(\mathfrak{E}\setminus T)\cup\{\mathfrak{d}[T]\}|\mathfrak{D}'} = \mathfrak{Cut}_{\mathfrak{E}|\mathfrak{D}}$$

by one step 2 of Def. 2.1.6 backwards, hence the result. The reasoning is similar if $\mathfrak{d}$ is negative and $\mathfrak{D} = \mathfrak{D}' \cup \{\mathfrak{d}/y\}$, we just have to distinguish between the cases $y \in \mathrm{fv}(\mathfrak{E}')$ and $y \notin \mathrm{fv}(\mathfrak{E}')$.

– Otherwise, let

$$\begin{aligned}
& S' = \{(\mathfrak{m}/y) \in \mathfrak{E} \mid y \in \mathrm{fv}(S)\}\\
\text{and} \quad & S'' = \{(\mathfrak{m}/y) \in \mathfrak{D} \mid y \in \mathrm{fv}(S')\} \ .
\end{aligned}$$

Note that $S' \subseteq \mathfrak{E}'$ and $S'' \subseteq (\mathfrak{D} \setminus S)$. We have:

$$\begin{aligned}
\mathfrak{Cut}_{\mathfrak{E}|\mathfrak{D}} &= \mathfrak{Cut}_{(\mathfrak{E}'\setminus S')\cup\{\mathfrak{n}[S[S']]\}|\mathfrak{D}\setminus S} && \text{by several steps 4 of Def. 2.1.6}\\
&= \mathfrak{Cut}_{\mathfrak{D}\setminus S|(\mathfrak{E}'\setminus S')\cup\{\mathfrak{n}[S[S']]\}} && \text{by induction hyp., since } S \neq \emptyset\\
&= \mathfrak{Cut}_{(\mathfrak{D}\setminus(S\cup S''))[\mathfrak{n}[S[S'[S'']]]/x]|\mathfrak{E}'\setminus S'} && \text{by one step 4 of Def. 2.1.6}\\
&= \mathfrak{Cut}_{\mathfrak{D}|\mathfrak{E}} && \text{by steps 4 of Def. 2.1.6 backwards.}
\end{aligned}$$

The last equality is obtained by moving successively, from left to right, all the designs from S', and finally the design $\mathfrak{n}$.

□

Lemma 2.1.12

Let $\mathfrak{D}_1$, $\mathfrak{D}_2$ and $\mathfrak{E}$ be multi-designs such that $\mathfrak{D}_1 \cup \mathfrak{D}_2$ is a multi-design with $\mathfrak{D}_1$ and $\mathfrak{D}_2$ disjoint, and $\mathfrak{E}$ is compatible with $\mathfrak{D}_1 \cup \mathfrak{D}_2$. We have:

$$\mathfrak{Cut}_{\mathfrak{D}_1\cup\mathfrak{D}_2|\mathfrak{E}} = \mathfrak{Cut}_{\mathfrak{D}_1|\mathfrak{Cut}_{\mathfrak{E}|\mathfrak{D}_2}} \ .$$

Proof. By induction on $\mathfrak{D}_2$:

- If $\mathfrak{D}_2 = \emptyset$ then $\mathfrak{Cut}_{\mathfrak{E}|\mathfrak{D}_2} = \mathfrak{E}$ hence the result.

- If $\mathfrak{p} \in \mathfrak{D}_2$ then $\mathfrak{Cut}_{\mathfrak{E}|\mathfrak{D}_2} = \mathfrak{Cut}_{(\mathfrak{E}\setminus S)\cup\{\mathfrak{p}[S]\}|\mathfrak{D}'_2}$ where $\mathfrak{D}'_2 = \mathfrak{D}_2 \setminus \{\mathfrak{p}\}$ and $S = \{(\mathfrak{m}/y) \in \mathfrak{E} \mid y \in \mathrm{fv}(\mathfrak{p})\}$. Thus by induction hypothesis:

$$\begin{aligned}\mathfrak{Cut}_{\mathfrak{D}_1|\mathfrak{Cut}_{\mathfrak{E}|\mathfrak{D}_2}} &= \mathfrak{Cut}_{\mathfrak{D}_1\cup\mathfrak{D}'_2|(\mathfrak{E}\setminus S)\cup\{\mathfrak{p}[S]\}} \\ &= \mathfrak{Cut}_{((\mathfrak{D}_1\cup\mathfrak{D}'_2)\setminus S')\cup\{\mathfrak{p}[S[S']]\}|\mathfrak{E}\setminus S} && \text{by one step 2 of Def. 2.1.6} \\ &= \mathfrak{Cut}_{((\mathfrak{D}_1\cup\mathfrak{D}_2)\setminus S')[S[S']]|\mathfrak{E}\setminus S}\ ,\end{aligned}$$

  where $S' = \{(\mathfrak{m}/y) \in (\mathfrak{D}_1 \cup \mathfrak{D}_2) \mid y \in \mathrm{fv}(S)\}$. Finally, by several steps 4 of Definition 2.1.6 backwards, this is equal to $\mathfrak{Cut}_{\mathfrak{D}_1\cup\mathfrak{D}_2|\mathfrak{E}}$.
- If $(\mathfrak{n}/x) \in \mathfrak{D}_2$ and $x \notin \mathrm{fv}(\mathfrak{E})$, then similar to the previous case.
- If $(\mathfrak{n}/x) \in \mathfrak{D}_2$ and $x \in \mathrm{fv}(\mathfrak{E})$, then $\mathfrak{Cut}_{\mathfrak{E}|\mathfrak{D}_2} = \mathfrak{Cut}_{(\mathfrak{E}\setminus S)[\mathfrak{n}[S]/x]|\mathfrak{D}'_2}$ where $\mathfrak{D}'_2 = \mathfrak{D}_2 \setminus \{\mathfrak{n}/x\}$ and $S = \{(\mathfrak{m}/y) \in \mathfrak{E} \mid y \in \mathrm{fv}(\mathfrak{n})\}$. Thus by induction hypothesis:

$$\begin{aligned}\mathfrak{Cut}_{\mathfrak{D}_1|\mathfrak{Cut}_{\mathfrak{E}|\mathfrak{D}_2}} &= \mathfrak{Cut}_{\mathfrak{D}_1\cup\mathfrak{D}'_2|(\mathfrak{E}\setminus S)[\mathfrak{n}[S]/x]} \\ &= \mathfrak{Cut}_{(\mathfrak{E}\setminus S)[\mathfrak{n}[S]/x]|\mathfrak{D}_1\cup\mathfrak{D}'_2} && \text{by Lemma 2.1.11} \\ &= \mathfrak{Cut}_{\mathfrak{E}|\mathfrak{D}_1\cup\mathfrak{D}_2} && \text{by one step 4 backwards of Def. 2.1.6} \\ &= \mathfrak{Cut}_{\mathfrak{D}_1\cup\mathfrak{D}_2|\mathfrak{E}} && \text{by Lemma 2.1.11.}\end{aligned}$$

□

2.2 Paths and Multi-Designs

In this section, we generalise the interaction path to multi-designs. In particular, we give an inductive definition of the *interaction sequence* (Definition 2.2.3) and we show (Proposition 2.2.9) that it corresponds to the same notion as the interaction path. Then we prove (Proposition 2.2.11) that two multi-designs $\mathfrak{D}$ and $\mathfrak{E}$ are orthogonal if and only if there exists a path of $\mathfrak{D}$ such that its dual is a path of $\mathfrak{E}$. Finally, we give a result of associativity for interaction paths (Proposition 2.2.12).

2.2.a Interaction Path

Recall that we write ϵ for the empty sequence.

Definition 2.2.1 (*Path, view*)

Let $\mathfrak{D}$ be a cut-free multi-design.

- A **view of** $\mathfrak{D}$ is a view of a design in $\mathfrak{D}$.
- A **path of** $\mathfrak{D}$ is a path s of same polarity as $\mathfrak{D}$ such that for all prefix s' of s, $\ulcorner s' \urcorner$ is a view of $\mathfrak{D}$.

We are now interested in a particular form of closed interaction, where we can identify two sides of the multi-design: designs are divided in two groups such that there are no cuts between designs of the same group. This corresponds exactly to the interaction between two closed-compatible multi-designs. The notion of interaction path (Definition 1.2.10) is extended to multi-designs.

Definition 2.2.2 (*Interaction path*)

Let $\mathfrak{D}$ and $\mathfrak{E}$ be cut-free closed-compatible multi-designs such that $\mathfrak{D} \perp \mathfrak{E}$. The **interaction path** of $\mathfrak{D}$ with $\mathfrak{E}$ is the unique path s of $\mathfrak{D}$ such that $\check{s}$ is a path of $\mathfrak{E}$.

But nothing ensures the existence and uniqueness of such a path: this will be proved in the rest of this subsection. We will moreover show that, if $\mathfrak{D} \perp \mathfrak{E}$, this path corresponds to the interaction sequence defined below. For the purpose of giving an inductive definition of the interaction sequence, we define it not only for a pair of closed-compatible multi-designs but for a larger class of pairs of multi-designs. Thus, in the rest of this subsection, we suppose that we have two multi-designs $\mathfrak{D}$ and $\mathfrak{E}$ that are

- cut-free,
- of opposite polarities,
- compatible,
- satisfying $\mathrm{fv}(\mathfrak{D}) \subseteq \mathrm{np}(\mathfrak{E})$ and $\mathrm{fv}(\mathfrak{E}) \subseteq \mathrm{np}(\mathfrak{D})$.

Definition 2.2.3 (*Interaction sequence*)

The **interaction sequence** of $\mathfrak{D}$ with $\mathfrak{E}$, written $\langle \mathfrak{D} \leftarrow \mathfrak{E} \rangle$, is the sequence of actions followed by interaction on the side of $\mathfrak{D}$. More precisely, if we write $\mathfrak{p}$ for the only positive design of $\mathfrak{D} \cup \mathfrak{E}$, the interaction sequence is defined recursively as follows.

- If $\mathfrak{p} = \maltese$ then:

$$\langle \mathfrak{D} \leftarrow \mathfrak{E} \rangle = \maltese \qquad \text{if } \maltese \in \mathfrak{D} \ ,$$
$$\langle \mathfrak{D} \leftarrow \mathfrak{E} \rangle = \epsilon \qquad \text{if } \maltese \in \mathfrak{E} \ .$$

- If $\mathfrak{p} = \Omega$ then $\langle \mathfrak{D} \leftarrow \mathfrak{E} \rangle = \epsilon$.
- If $\mathfrak{p} = x|\overline{a}\langle \overrightarrow{\mathfrak{m}} \rangle$ then there exists $\mathfrak{n}$ such that $(\mathfrak{n}/x) \in \mathfrak{E}$ if $\mathfrak{p} \in \mathfrak{D}$, $(\mathfrak{n}/x) \in \mathfrak{D}$ otherwise. Let us write $\mathfrak{n} = \sum_{b \in \mathcal{S}} b(\overrightarrow{y^b}).\mathfrak{p}_b$. We have

$$\langle \mathfrak{D} \leftarrow \mathfrak{E} \rangle = \kappa \langle \mathfrak{D}' \leftarrow \mathfrak{E}' \rangle$$

  where
  - $\kappa = x|\overline{a}\langle \overrightarrow{y^a} \rangle$, $\mathfrak{D}' = (\mathfrak{D} \setminus \{\mathfrak{p}\}) \cup \{\overrightarrow{\mathfrak{m}/y^a}\}$ and $\mathfrak{E}' = (\mathfrak{E} \setminus \{\mathfrak{n}/x\}) \cup \{\mathfrak{p}_a\}$ if $\mathfrak{p} \in \mathfrak{D}$,
  - $\kappa = a_x(\overrightarrow{y^a})$, $\mathfrak{D}' = (\mathfrak{D} \setminus \{\mathfrak{n}/x\}) \cup \{\mathfrak{p}_a\}$ and $\mathfrak{E}' = (\mathfrak{E} \setminus \{\mathfrak{p}\}) \cup \{\overrightarrow{\mathfrak{m}/y^a}\}$ otherwise.

Note that this applies in particular to two closed-compatible multi-designs. Remark also that this definition follows exactly the interaction between $\mathfrak{D}$ and $\mathfrak{E}$: indeed, in the inductive case of the definition, the multi-designs $\mathfrak{D}'$ and $\mathfrak{E}'$ are obtained from $\mathfrak{D}$ and $\mathfrak{E}$ similarly to the following lemma. In particular the interaction sequence is finite whenever the interaction between $\mathfrak{D}$ and $\mathfrak{E}$ is finite.

In the following, let $\rightsquigarrow$ denote a step of reduction of one design in a multi-design:

$$\mathfrak{D}_1 \rightsquigarrow \mathfrak{D}_2 \qquad \text{if} \qquad \mathfrak{d}_1 \rightsquigarrow \mathfrak{d}_2 \text{ with } \mathfrak{d}_1 \in \mathfrak{D}_1 \text{ and } \mathfrak{D}_2 = \mathfrak{D}_1 \setminus \{\mathfrak{d}_1\} \cup \{\mathfrak{d}_2\} \ .$$

In particular there are several possible reductions from a multi-design, depending on which design we choose to reduce. Note that if $\mathfrak{D}_1 \rightsquigarrow \mathfrak{D}_2$ then $(\!|\mathfrak{D}_1|\!) = (\!|\mathfrak{D}_2|\!)$.

Lemma 2.2.4

Suppose $\mathfrak{D}$ positive and $\mathfrak{E}$ negative. Let $\mathfrak{p} = x|\overline{a}\langle\overrightarrow{\mathfrak{n}}\rangle$ be the only positive design of $\mathfrak{D}$, and suppose there exists $\mathfrak{n}_0$ such that $(\mathfrak{n}_0/x) \in \mathfrak{E}$, say $\mathfrak{n}_0 = \sum_{b\in\mathcal{S}} b(\overrightarrow{b}).\mathfrak{p}_b$. Then:

$$\mathfrak{Cut}_{\mathfrak{D}|\mathfrak{E}} \rightsquigarrow \mathfrak{Cut}_{\mathfrak{D}'|\mathfrak{E}'} \setminus \{(\mathfrak{m}/x_i^a) \mid x_i^a \notin \mathrm{fv}(\mathfrak{p}_a)\}$$

where $\mathfrak{D}' = (\mathfrak{D} \setminus \{\mathfrak{p}\}) \cup \{\overrightarrow{\mathfrak{n}/x^a}\}$ and $\mathfrak{E}' = (\mathfrak{E} \setminus \{\mathfrak{n}_0/x\}) \cup \{\mathfrak{p}_a\}$.

Proof. Let us prove this result in the case $\mathfrak{D}$ and $\mathfrak{E}$ are closed-compatible; in the general case, the reduction step from multi-design $\mathfrak{Cut}_{\mathfrak{D}|\mathfrak{E}}$ in the lemma corresponds to reducing the positive design $\mathfrak{q}$ of $\mathfrak{Cut}_{\mathfrak{D}|\mathfrak{E}}$, thus the proof is similar by simply ignoring the negative designs in $\mathfrak{D} \cup \mathfrak{E}$ that are not substituted in $\mathfrak{q}$.

If $\mathfrak{D}$ and $\mathfrak{E}$ are closed-compatible, $\mathfrak{Cut}_{\mathfrak{D}|\mathfrak{E}}$ is a closed design, and since this design has cuts we can apply one (unique) step of reduction to it. Let $S' = \{(\mathfrak{m}/x_i^a) \mid x_i^a \notin \mathrm{fv}(\mathfrak{p}_a)\}$. We have to prove $\mathfrak{Cut}_{\mathfrak{D}|\mathfrak{E}} \rightsquigarrow \mathfrak{Cut}_{\mathfrak{D}'|\mathfrak{E}'} \setminus S'$. The proof is done by induction on the number of designs in $\mathfrak{E}$.

- If $\mathfrak{E} = \{\mathfrak{n}_0/x\}$, then $\mathfrak{E}' = \{\mathfrak{p}_a\}$. In this case let $S = \{(\mathfrak{m}/y) \in \mathfrak{D} \mid y \in \mathrm{fv}(\mathfrak{n}_0)\}$, and remark that, as $\mathfrak{E}$ and $\mathfrak{D}$ are closed-compatible, $S = \mathfrak{D} \setminus \{\mathfrak{p}\}$. Thus:

$$\begin{aligned}
\mathfrak{Cut}_{\mathfrak{D}|\mathfrak{E}} &= \mathfrak{Cut}_{(\mathfrak{D}\setminus S)[\mathfrak{n}_0[S]/x]|\emptyset} && \text{by one step 4 of Def. 2.1.6}\\
&= \mathfrak{p}[\mathfrak{n}_0[S]/x]\\
&\rightsquigarrow \mathfrak{p}_a[S][\overrightarrow{\mathfrak{n}/x^a}]\\
&= \mathfrak{p}_a[\mathfrak{D}']\\
&= \{\mathfrak{p}_a[\mathfrak{D}' \setminus S'_0]\} \cup S'_0 \setminus S' && \text{where } S'_0 = S'|\mathfrak{D}'\\
&= \mathfrak{Cut}_{S'_0\cup\{\mathfrak{p}_a[\mathfrak{D}'\setminus S'_0]\}|\emptyset} \setminus S'\\
&= \mathfrak{Cut}_{\mathfrak{D}'|\mathfrak{p}_a} \setminus S' && \text{by one step 2 of Def. 2.1.6 backwards}\\
&= \mathfrak{Cut}_{\mathfrak{D}'|\mathfrak{E}'} \setminus S' \ .
\end{aligned}$$

- Otherwise there exists $(\mathfrak{n}_1/z) \in \mathfrak{E}$ such that $x \neq z$. Suppose $z \notin \mathrm{fv}(\mathfrak{D})$ (resp. $z \in \mathrm{fv}(\mathfrak{D})$). Define:
  - $S = \{(\mathfrak{m}/y) \in \mathfrak{D} \mid y \in \mathrm{fv}(\mathfrak{n}_1)\}$, and remark $S = \{(\mathfrak{m}/y) \in \mathfrak{D}' \mid y \in \mathrm{fv}(\mathfrak{n}_1)\}$,
  - $\mathfrak{D}'' = (\mathfrak{D}' \setminus S) \cup \{(\mathfrak{n}_1[S]/z)\}$ (resp. $\mathfrak{D}'' = (\mathfrak{D}' \setminus S)[\mathfrak{n}_1[S]/z]$),
  - $\mathfrak{E}'' = \mathfrak{E}' \setminus \{(\mathfrak{n}_1/z)\}$.

We have:

$$\begin{array}{rll}
\mathfrak{Cut}_{\mathfrak{D}|\mathfrak{E}} = \mathfrak{Cut}_{(\mathfrak{D}\setminus S)\cup\{(\mathfrak{n}_1[S]/z)\}|\mathfrak{E}\setminus\{\mathfrak{n}_1/z\}} & & \text{by one step 3 of Def. 2.1.6} \\
(\quad \text{resp. } = \mathfrak{Cut}_{(\mathfrak{D}\setminus S)[(\mathfrak{n}_1[S]/z)]|\mathfrak{E}\setminus\{\mathfrak{n}_1/z\}} & & \text{by one step 4 of Def. 2.1.6 }) \\
\rightsquigarrow \mathfrak{Cut}_{\mathfrak{D}''|\mathfrak{E}''} \setminus S' & & \text{by induction hypothesis} \\
= \mathfrak{Cut}_{\mathfrak{D}'|\mathfrak{E}'} \setminus S' & & \text{by step 3 (resp. 4) of Def. 2.1.6 backwards.}
\end{array}$$

□

Lemma 2.2.5

If $\maltese \in (\!|\mathfrak{Cut}_{\mathfrak{D}|\mathfrak{E}}|\!)$ *(in particular if* $\mathfrak{D} \perp \mathfrak{E}$*) then* $\langle \mathfrak{D} \leftarrow \mathfrak{E}\rangle = \widetilde{\langle \mathfrak{E} \leftarrow \mathfrak{D}\rangle}$*. Otherwise* $\langle \mathfrak{D} \leftarrow \mathfrak{E}\rangle = \overline{\langle \mathfrak{E} \leftarrow \mathfrak{D}\rangle}$*.*

Proof. It is clear from the definition of the interaction sequence that the proper actions in $\langle \mathfrak{D} \leftarrow \mathfrak{E}\rangle$ are the opposite of those in $\langle \mathfrak{E} \leftarrow \mathfrak{D}\rangle$ (even if $\langle \mathfrak{D} \leftarrow \mathfrak{E}\rangle$ is infinite). Concerning the daimon: since the interaction sequence follows the interaction between $\mathfrak{D}$ and $\mathfrak{E}$, $\maltese$ appears at the end of one of the sequences $\langle \mathfrak{D} \leftarrow \mathfrak{E}\rangle$ or $\langle \mathfrak{E} \leftarrow \mathfrak{D}\rangle$ if and only if $\maltese \in (\!|\mathfrak{Cut}_{\mathfrak{D}|\mathfrak{E}}|\!)$, and in this case $\langle \mathfrak{D} \leftarrow \mathfrak{E}\rangle = \widetilde{\langle \mathfrak{E} \leftarrow \mathfrak{D}\rangle}$. □

Lemma 2.2.6

Every positive-ended prefix of $\langle \mathfrak{D} \leftarrow \mathfrak{E}\rangle$ *is a path of* $\mathfrak{D}$*. In particular, if* $\langle \mathfrak{D} \leftarrow \mathfrak{E}\rangle$ *is finite and positive-ended then it is a path of* $\mathfrak{D}$*.*

Proof. First remark that every (finite) prefix of $\langle \mathfrak{D} \leftarrow \mathfrak{E}\rangle$ is an aj-sequence. Indeed, since $\mathfrak{D}$ and $\mathfrak{E}$ are well shaped multi-designs the definition of interaction sequence ensures that an action cannot appear before its justification, and all the conditions of the definition of an aj-sequence are satisfied: *Alternation* and *Daimon* are immediate from the definition of interaction sequence, while *Linearity* is indeed satisfied as variables are disjoint in $\mathfrak{D}$ and $\mathfrak{E}$ (Barendregt's convention).

By definition, for every prefix s of $\langle \mathfrak{D} \leftarrow \mathfrak{E}\rangle$, $\ulcorner s \urcorner$ is a view. We show that it is a view of $\mathfrak{D}$ by induction on the length of s:

- If $s = \epsilon$ then $\ulcorner \epsilon \urcorner = \epsilon$ is indeed a view of $\mathfrak{D}$.
- If $s = \maltese$ then $\langle \mathfrak{D} \leftarrow \mathfrak{E}\rangle = \maltese$. From the definition of the interaction sequence, we know that in this case $\maltese \in \mathfrak{D}$, hence $\ulcorner \maltese \urcorner = \maltese$ is a view of $\mathfrak{D}$.
- If $s = \kappa s'$ where κ is proper, then $\langle \mathfrak{D} \leftarrow \mathfrak{E}\rangle = \kappa\langle \mathfrak{D}' \leftarrow \mathfrak{E}'\rangle$ where $\mathfrak{D}'$ and $\mathfrak{E}'$ are as in Definition 2.2.3, and s' is a prefix of $\langle \mathfrak{D}' \leftarrow \mathfrak{E}'\rangle$. By induction hypothesis, $\ulcorner s' \urcorner$ is a view of $\mathfrak{D}'$. Two possibilities:
  - Either $\kappa = \kappa^+$ is positive. From the definition of the interaction sequence, it means that $\mathfrak{p} := x|\overline{a}\langle \overrightarrow{\mathfrak{m}}\rangle \in \mathfrak{D}$, $\kappa^+ = x|\overline{a}\langle \overrightarrow{y^a}\rangle$ and $\mathfrak{D}' = (\mathfrak{D}\setminus\{\mathfrak{p}\})\cup\{\overrightarrow{\mathfrak{m}/y^a}\}$. We have $\ulcorner s \urcorner = \ulcorner \kappa^+ s' \urcorner$ and either $\ulcorner \kappa^+ s' \urcorner = \kappa^+ \ulcorner s' \urcorner$ if the first negative action of

$\ulcorner s' \urcorner$ is justified by κ^+ (i.e., $\exists i$ such that $\ulcorner s' \urcorner$ is a view of $\mathfrak{m}_i/y_i^a$), or $\ulcorner \kappa^+ s' \urcorner = \ulcorner s' \urcorner$ otherwise (i.e., $\ulcorner s' \urcorner$ is a view of $\mathfrak{D} \setminus \{\mathfrak{p}\}$). In the second case, there is nothing more to show; in the first one, by definition of the views of a design, $\kappa^+ \ulcorner s' \urcorner$ is a view of $\mathfrak{p} = x|\overline{a}\langle\overrightarrow{\mathfrak{m}}\rangle$.

– Or $\kappa = \kappa^-$ is negative. Hence there exists a design $\mathfrak{n} = \sum_{b \in \mathcal{S}} b(\overrightarrow{y^b}).\mathfrak{p}_b$ such that $(\mathfrak{n}/x) \in \mathfrak{D}$, $\kappa^- = a_x(\overrightarrow{y^a})$, and $\mathfrak{D}' = (\mathfrak{D} \setminus \{\mathfrak{n}/x\}) \cup \{\mathfrak{p}_a\}$. We have $\ulcorner s \urcorner = \ulcorner \kappa^- s' \urcorner$ and either $\ulcorner \kappa^- s' \urcorner = \kappa^- \ulcorner s' \urcorner$ if the first action of $\ulcorner s' \urcorner$ is positive (i.e., $\ulcorner s' \urcorner$ is a view of $\mathfrak{p}_a$), or $\ulcorner \kappa^- s' \urcorner = \ulcorner s' \urcorner$ otherwise (i.e., $\ulcorner s' \urcorner$ is a view of $\mathfrak{D}' \setminus \{\mathfrak{p}_a\} \subseteq \mathfrak{D}$). In the second case, there is nothing to do; in the first one, note that $\kappa^- \ulcorner s' \urcorner$ is a view of $(\mathfrak{n}/x)$, hence the result.

We have proved that $\ulcorner s \urcorner$ is a view of $\mathfrak{D}$. This implies in particular that $\langle \mathfrak{D} \leftarrow \mathfrak{E} \rangle$ satisfies P-visibility, indeed: given a prefix $s\kappa^+$ of $\langle \mathfrak{D} \leftarrow \mathfrak{E} \rangle$, the action κ^+ is either initial or it is justified in s by the same action that justifies it in $\mathfrak{D}$; since $\ulcorner s \urcorner$ is a view of $\mathfrak{D}$, the justification of κ^+ is in it, thus P-visibility is satisfied. Similarly, we can prove that $\ulcorner t \urcorner$ is a view of $\mathfrak{E}$ whenever t is a prefix of $\langle \mathfrak{E} \leftarrow \mathfrak{D} \rangle$, therefore $\langle \mathfrak{E} \leftarrow \mathfrak{D} \rangle$ also satisfies P-visibility; by Lemma 2.2.5 either $\langle \mathfrak{E} \leftarrow \mathfrak{D} \rangle = \widetilde{\langle \mathfrak{D} \leftarrow \mathfrak{E} \rangle}$ or $\langle \mathfrak{E} \leftarrow \mathfrak{D} \rangle = \overline{\langle \mathfrak{D} \leftarrow \mathfrak{E} \rangle}$, thus this implies that $\langle \mathfrak{D} \leftarrow \mathfrak{E} \rangle$ satisfies O-visibility. Hence every positive-ended prefix of $\langle \mathfrak{D} \leftarrow \mathfrak{E} \rangle$ is a path, and since the views of its prefixes are views of $\mathfrak{D}$, it is a path of $\mathfrak{D}$. □

Remark 2.2.7

If $s\kappa_1^+$ and $s\kappa_2^+$ are views (resp. paths) of a multi-design $\mathfrak{D}$ then $\kappa_1^+ = \kappa_2^+$. Indeed, if $s\kappa_1^+$ and $s\kappa_2^+$ are views of $\mathfrak{D}$, the result is immediate by definition of the views of a design; if they are paths of $\mathfrak{D}$, just remark that $\ulcorner s\kappa_1^+ \urcorner = \ulcorner s \urcorner \kappa_1^+$ and $\ulcorner s\kappa_2^+ \urcorner = \ulcorner s \urcorner \kappa_2^+$ are views of $\mathfrak{D}$, hence the conclusion.

Lemma 2.2.8

Suppose $\mathfrak{D} \perp \mathfrak{E}$, s is a path of $\mathfrak{D}$ and $\overline{s}$ is a path of $\mathfrak{E}$. The path s is a prefix of $\langle \mathfrak{D} \leftarrow \mathfrak{E} \rangle$.

Proof. Suppose s is not a prefix of $\langle \mathfrak{D} \leftarrow \mathfrak{E} \rangle$. Let t be the longest common prefix of s and $\langle \mathfrak{D} \leftarrow \mathfrak{E} \rangle$ (possibly ϵ). Without loss of generality, we can assume there exist actions of same polarity κ_1 and κ_2 such that $\kappa_1 \neq \kappa_2$, $t\kappa_1$ is a prefix of s and $t\kappa_2$ is a prefix of $\langle \mathfrak{D} \leftarrow \mathfrak{E} \rangle$: indeed, if there are no such actions, it is because $\langle \mathfrak{D} \leftarrow \mathfrak{E} \rangle$ is a strict prefix of s; in this case, it suffices to consider $\langle \mathfrak{E} \leftarrow \mathfrak{D} \rangle$ and $\overline{s}$ instead.

- If κ_1 and κ_2 are positive, then $t\kappa_1$ and $t\kappa_2$ are paths of $\mathfrak{D}$, and by Remark 2.2.7 we have $\kappa_1 = \kappa_2$, contradiction.
- If κ_1 and κ_2 are negative, a contradiction arises similarly from the fact that $\overline{t\kappa_1}$ and $\overline{t\kappa_2}$ are paths of $\mathfrak{E}$ where $\overline{\kappa_1}$ and $\overline{\kappa_2}$ are positive.

Hence the result. □

The following result ensures that the interaction path is well defined, i.e., that such a path exists and is unique.

Proposition 2.2.9

If $\mathfrak{D} \perp \mathfrak{E}$, there exists a unique path s of $\mathfrak{D}$ such that $\tilde{s}$ is a path of $\mathfrak{E}$, and $s = \langle \mathfrak{D} \leftarrow \mathfrak{E} \rangle$.

***Proof*.** Lemmas 2.2.5 and 2.2.6 show that $\langle \mathfrak{D} \leftarrow \mathfrak{E} \rangle$ is a path of $\mathfrak{D}$, and its dual is a path of $\mathfrak{E}$. Uniqueness follows from Lemma 2.2.8. □

Conversely, we prove that the existence of such a path implies the orthogonality of multi-designs (Proposition 2.2.11). First a lemma.

Lemma 2.2.10

Suppose $\mathfrak{D}$ and $\mathfrak{E}$ are closed-compatible and have a finite interaction, with $\mathfrak{D}$ positive and $\Omega \notin \mathfrak{D}$. Suppose that for every path $s\kappa^+$ of $\mathfrak{D}$ such that κ^+ is proper and $\overline{s}$ is a path of $\mathfrak{E}$, $\overline{s\kappa^+}$ is a path of $\mathfrak{E}$, and suppose also that the same condition is satisfied when reversing $\mathfrak{D}$ and $\mathfrak{E}$. Then $\mathfrak{D} \perp \mathfrak{E}$.

***Proof*.** By induction on the number n of steps of the interaction before divergence/convergence:

- If $n = 0$, then we must have $\mathfrak{D} = \maltese$, since $\Omega \notin \mathfrak{D}$. Hence the result.
- If $n > 0$ then $\mathfrak{p} \in \mathfrak{D}$ is of the form $\mathfrak{p} = x|\overline{a}\langle \overrightarrow{\mathfrak{n}} \rangle$ and there exists $\mathfrak{n}_0 = \sum_{b \in \mathcal{S}} b(\overrightarrow{x^b}).\mathfrak{p}_b$ such that $(\mathfrak{n}_0/x) \in \mathfrak{E}$. Let $\kappa^+ = x|\overline{a}\langle \overrightarrow{x^a} \rangle$ and remark that κ^+ is a path of $\mathfrak{p}$. By hypothesis, $\overline{\kappa^+} = a_x(\overrightarrow{x^a})$ is a path of $\mathfrak{E}$, thus a path of $\mathfrak{n}_0$, and this implies $\mathfrak{p}_a \neq \Omega$. By Lemma 2.2.4, we have

  $$\mathfrak{Cut}_{\mathfrak{D}|\mathfrak{E}} \rightsquigarrow \mathfrak{Cut}_{\mathfrak{D}'|\mathfrak{E}'} \setminus \{(\mathfrak{m}/x_i^a) \mid x_i \notin \mathrm{fv}(\mathfrak{p}_a)\}$$

  where $\mathfrak{D}' = (\mathfrak{D} \setminus \{\mathfrak{p}\}) \cup \{\overrightarrow{\mathfrak{n}/x^a}\}$ and $\mathfrak{E}' = (\mathfrak{E} \setminus \{\mathfrak{n}_0/x\}) \cup \{\mathfrak{p}_a\}$. This corresponds to the $\mathfrak{Cut}$ between two closed-compatible multi-designs $\mathfrak{D}'' \subseteq \mathfrak{D}'$ (negative) and $\mathfrak{E}'' \subseteq \mathfrak{E}'$ (positive), where:
  - $\Omega \notin \mathfrak{E}''$ because $\mathfrak{p}_a \neq \Omega$;
  - their interaction is finite and takes $n - 1$ steps;
  - the condition on paths stated in the proposition is satisfied for $\mathfrak{D}''$ and $\mathfrak{E}''$, because it is for $\mathfrak{D}$ and $\mathfrak{E}$: indeed, the paths of $\mathfrak{D}''$ (resp. $\mathfrak{E}''$) are the paths t such that $\kappa^+ t$ is a path of $\mathfrak{D}$ (resp. $\overline{\kappa^+} t$ is a path of $\mathfrak{E}$), unless such a path t contains a negative initial action whose address is not the address of a positive action on the other side, but this restriction is harmless with respect to our condition.

  We apply the induction hypothesis to get $\mathfrak{D}'' \perp \mathfrak{E}''$. Finally $\mathfrak{D} \perp \mathfrak{E}$.

□

The following proposition reduces orthogonality between multi-design (thus also between designs) to the existence of an interaction path.

Proposition 2.2.11

Suppose $\mathfrak{D}$ and $\mathfrak{E}$ are closed-compatible. $\mathfrak{D} \perp \mathfrak{E}$ if and only if there exists a path s of $\mathfrak{D}$ such that $\tilde{s}$ is a path of $\mathfrak{E}$.

Proof. ($\Rightarrow$) If $\mathfrak{D} \perp \mathfrak{E}$ then the result follows from Proposition 2.2.9.
($\Leftarrow$) We prove that the hypothesis of Lemma 2.2.10 is satisfied. Let us show that every path of $\mathfrak{D}$ (resp. of $\mathfrak{E}$) of the form $t\kappa^+$ where κ^+ is proper and $\overline{t}$ is a path of $\mathfrak{E}$ (resp. of $\mathfrak{D}$) is a prefix of s (resp. of $\overline{s}$). By induction on the length of t, knowing that it is either empty or negative-ended:

- If t is empty, κ^+ is necessarily the first action of the positive design in $\mathfrak{D}$ (resp. in $\mathfrak{E}$), hence the first action of s (resp. of $\overline{s}$).
- If $t = t_0\kappa^-$, then $\overline{t_0\kappa^-}$ is a path of $\mathfrak{E}$ (resp. of $\mathfrak{D}$) and t_0 is a path of $\mathfrak{D}$ (resp. of $\mathfrak{E}$). By induction hypothesis, $\overline{t} = \overline{t_0\kappa^-}$ is a prefix of $\overline{s}$ (resp. of s), thus t is a prefix of s (resp. of $\overline{s}$). The path s is of the form $s = t\kappa'^+s'$. But since s and $t\kappa^+$ are both paths of $\mathfrak{D}$ (resp. $\mathfrak{E}$), they cannot differ on a positive action, hence $\kappa^+ = \kappa'^+$. Thus $t\kappa^+$ is a prefix of s.

□

2.2.b Associativity for Interaction Paths

Let us conclude this chapter with an important proposition.

Notation

If s is a path of a multi-design $\mathfrak{D}$, and $\mathfrak{E} \subseteq \mathfrak{D}$, then we write $s\restriction\mathfrak{E}$ for the longest subsequence of s that is a path of $\mathfrak{E}$. Notice that this is well defined.

Proposition 2.2.12 (*Associativity for paths*)

Let $\mathfrak{D}$, $\mathfrak{E}$ and $\mathfrak{F}$ be cut-free multi-designs such that $\mathfrak{E} \cup \mathfrak{F}$ is a multi-design with $\mathfrak{E}$ and $\mathfrak{F}$ disjoint, and suppose $\mathfrak{D} \perp (\mathfrak{E} \cup \mathfrak{F})$. We have:

$$\langle \mathfrak{E} \leftarrow (\!|\mathfrak{Cut}_{\mathfrak{F}|\mathfrak{D}}|\!) \rangle = \langle \mathfrak{E} \cup \mathfrak{F} \leftarrow \mathfrak{D} \rangle \restriction \mathfrak{E} \quad .$$

This proposition states that we can view the restriction of the interaction path $\langle \mathfrak{E} \cup \mathfrak{F} \leftarrow \mathfrak{D} \rangle$ to its actions coming from $\mathfrak{E}$ as another interaction path itself, namely the interaction path between $\mathfrak{E}$ and the result of the interaction of $\mathfrak{F}$ with $\mathfrak{D}$. It looks like associativity in the sense that $\mathfrak{F}$ can switch to either side of the interaction.

Proof. We prove the result for a larger class of multi-designs. Instead of the assumption $\mathfrak{D} \perp (\mathfrak{E} \cup \mathfrak{F})$, suppose that $\mathfrak{D}$ and $\mathfrak{E} \cup \mathfrak{F}$ are:

- of opposite polarities,
- compatible,
- satisfying $\mathrm{fv}(\mathfrak{D}) \subseteq \mathrm{np}(\mathfrak{E} \cup \mathfrak{F})$ and $\mathrm{fv}(\mathfrak{E} \cup \mathfrak{F}) \subseteq \mathrm{np}(\mathfrak{D})$

- and such that $\maltese \in (\!|\mathfrak{Cut}_{\mathfrak{E}\cup\mathfrak{F}|\mathfrak{D}}|\!)$ (in particular their interaction is finite).

First remark that $\mathfrak{F}$ and $\mathfrak{D}$ are compatible, hence it is possible to define $\mathfrak{Cut}_{\mathfrak{F}|\mathfrak{D}}$. Then since $\maltese \in (\!|\mathfrak{Cut}_{\mathfrak{E}\cup\mathfrak{F}|\mathfrak{D}}|\!)$, we have $\maltese \in (\!|\mathfrak{Cut}_{\mathfrak{E}|(\!|\mathfrak{Cut}_{\mathfrak{F}|\mathfrak{D}}|\!)}|\!)$, indeed:

$$\begin{aligned}(\!|\mathfrak{Cut}_{\mathfrak{E}\cup\mathfrak{F}|\mathfrak{D}}|\!) &= (\!|\mathfrak{Cut}_{\mathfrak{E}|\mathfrak{Cut}_{\mathfrak{F}|\mathfrak{D}}}|\!) && \text{by Lemmas 2.1.12 and 2.1.11}\\ &= (\!|\mathfrak{Cut}_{\mathfrak{E}|(\!|\mathfrak{Cut}_{\mathfrak{F}|\mathfrak{D}}|\!)}|\!) && \text{by Proposition 2.1.10.}\end{aligned}$$

This also shows that $\mathfrak{E}$ and $(\!|\mathfrak{Cut}_{\mathfrak{F}|\mathfrak{D}}|\!)$ are compatible. As they are of opposite polarities and they satisfy the condition on variables, $\langle \mathfrak{E} \leftarrow (\!|\mathfrak{Cut}_{\mathfrak{F}|\mathfrak{D}}|\!)\rangle$ is defined.

Let $s = \langle \mathfrak{E}\cup\mathfrak{F} \leftarrow \mathfrak{D}\rangle$, and let us show the result (i.e., $s\restriction\mathfrak{E} = \langle \mathfrak{E} \leftarrow (\!|\mathfrak{Cut}_{\mathfrak{F}|\mathfrak{D}}|\!)\rangle$) by induction on the length of s, which is finite because the interaction between $\mathfrak{D}$ and $\mathfrak{E}\cup\mathfrak{F}$ is finite.

- If $s = \epsilon$ then necessarily $\maltese \in \mathfrak{D}$ thus also $\maltese \in (\!|\mathfrak{Cut}_{\mathfrak{F}|\mathfrak{D}}|\!)$. Hence
$$s\restriction\mathfrak{E} = \epsilon = \langle \mathfrak{E} \leftarrow (\!|\mathfrak{Cut}_{\mathfrak{F}|\mathfrak{D}}|\!)\rangle \ .$$
- If $s = \maltese$ then $\maltese \in \mathfrak{E}\cup\mathfrak{F}$. In this case, either $\maltese \in \mathfrak{E}$ and then
$$\langle \mathfrak{E} \leftarrow (\!|\mathfrak{Cut}_{\mathfrak{F}|\mathfrak{D}}|\!)\rangle = \maltese = s\restriction\mathfrak{E}$$
or $\maltese \in \mathfrak{F}$ thus $\maltese \in (\!|\mathfrak{Cut}_{\mathfrak{F}|\mathfrak{D}}|\!)$ and
$$\langle \mathfrak{E} \leftarrow (\!|\mathfrak{Cut}_{\mathfrak{F}|\mathfrak{D}}|\!)\rangle = \epsilon = s\restriction\mathfrak{E} \ .$$
- If $s = \kappa^+ s'$ where $\kappa^+ = x|\overline{a}\langle\overrightarrow{x^a}\rangle$ is a proper positive action, then $\mathfrak{E}\cup\mathfrak{F}$ is a positive multi-design such that its only positive design is of the form $\mathfrak{p} = x|\overline{a}\langle\overrightarrow{\mathfrak{m}}\rangle$. Thus $\mathfrak{D}$ is negative, and there exists $\mathfrak{n}$ such that $(\mathfrak{n}/x) \in \mathfrak{D}$ of the form $\mathfrak{n} = \sum_{b\in\mathcal{S}} b(\overrightarrow{x^b}).\mathfrak{p}_b$, where $\mathfrak{p}_a \neq \Omega$ because the interaction converges. Let $\mathfrak{D}' = (\mathfrak{D}\setminus\{\mathfrak{n}/x\})\cup\{\mathfrak{p}_a\}$.
  - Either $\mathfrak{p} \in \mathfrak{F}$ [**reduction step**]. In this case, we have
  $$s\restriction\mathfrak{E} = s'\restriction\mathfrak{E}$$
  so let us show that $s'\restriction\mathfrak{E} = \langle \mathfrak{E} \leftarrow (\!|\mathfrak{Cut}_{\mathfrak{F}|\mathfrak{D}}|\!)\rangle$. By definition of the interaction sequence, we have $s' = \langle \mathfrak{E}\cup\mathfrak{F}' \leftarrow \mathfrak{D}'\rangle$ where $\mathfrak{F}' = (\mathfrak{F}\setminus\{\mathfrak{p}\})\cup\{\overrightarrow{(\mathfrak{m}/x^a)}\}$. Thus by induction hypothesis
  $$s'\restriction\mathfrak{E} = \langle \mathfrak{E} \leftarrow (\!|\mathfrak{Cut}_{\mathfrak{F}'|\mathfrak{D}'}|\!)\rangle \ .$$
  But by Lemma 2.2.4,
  $$\langle \mathfrak{E} \leftarrow (\!|\mathfrak{Cut}_{\mathfrak{F}'|\mathfrak{D}'}|\!)\rangle = \langle \mathfrak{E} \leftarrow (\!|\mathfrak{Cut}_{\mathfrak{F}|\mathfrak{D}}|\!)\rangle$$
  because the negatives among $\overrightarrow{(\mathfrak{m}/x^a)}$ in $(\!|\mathfrak{Cut}_{\mathfrak{F}'|\mathfrak{D}'}|\!)$ will not interfere in the interaction with $\mathfrak{E}$, since the variables $\overrightarrow{x^a}$ do not appear in $\mathfrak{E}$. Hence the result.

– Or $\mathfrak{p} \in \mathfrak{E}$ [**commutation step**]. In this case, we have

$$s\restriction\mathfrak{E} = \kappa^+(s'\restriction\mathfrak{E})$$

and by definition of the interaction sequence $s' = \langle \mathfrak{E}' \cup \mathfrak{F} \leftarrow \mathfrak{D}' \rangle$ where $\mathfrak{E}' = (\mathfrak{E} \setminus \{\mathfrak{p}\}) \cup \{\overrightarrow{(\mathfrak{m}/x^a)}\}$. Thus by induction hypothesis

$$s'\restriction\mathfrak{E} = s'\restriction\mathfrak{E}' = \langle \mathfrak{E}' \leftarrow (\!|\mathfrak{Cut}_{\mathfrak{F}|\mathfrak{D}'}|\!) \rangle \ .$$

But we have

$$\begin{aligned}\langle \mathfrak{E} \leftarrow (\!|\mathfrak{Cut}_{\mathfrak{F}|\mathfrak{D}}|\!) \rangle &= \langle \mathfrak{E} \leftarrow (\!|\mathfrak{Cut}_{\mathfrak{F}|\mathfrak{D}' \cup \{(\mathfrak{n}/x)\} \setminus \{\mathfrak{p}_a\}}|\!) \rangle \\ &= \langle \mathfrak{E} \leftarrow (\!|\mathfrak{Cut}_{\mathfrak{F}|\mathfrak{D}'}|\!) \cup \{(\mathfrak{n}'/x)\} \setminus \{\mathfrak{p}'_a\} \rangle \\ &= \kappa^+ \langle \mathfrak{E}' \leftarrow (\!|\mathfrak{Cut}_{\mathfrak{F}|\mathfrak{D}'}|\!) \rangle\end{aligned}$$

where $\mathfrak{n}'$ is the only negative design of $(\!|\mathfrak{Cut}_{\mathfrak{F}|\mathfrak{D}}|\!)$ on variable x, and $\mathfrak{p}'_a$ the only positive design of $(\!|\mathfrak{Cut}_{\mathfrak{F}|\mathfrak{D}'}|\!)$. Hence

$$\langle \mathfrak{E} \leftarrow (\!|\mathfrak{Cut}_{\mathfrak{F}|\mathfrak{D}}|\!) \rangle = \kappa^+(s'\restriction\mathfrak{E}) = s\restriction\mathfrak{E} \ .$$

- If $s = \kappa^- s'$ where $\kappa^- = a_x(\overrightarrow{a})$, then $\mathfrak{D}$ is positive with only positive design of the form $\mathfrak{p} = x|\overline{a}\langle\overrightarrow{\mathfrak{m}}\rangle$, and there exists a negative design $\mathfrak{n}$ such that $(\mathfrak{n}/x) \in \mathfrak{E} \cup \mathfrak{F}$, with $\mathfrak{n}$ of the form $\mathfrak{n} = \sum_{b \in \mathcal{S}} b(\overrightarrow{x^b}).\mathfrak{p}_b$ where $\mathfrak{p}_a \neq \Omega$. By definition of the interaction sequence, we have $s' = \langle ((\mathfrak{E} \cup \mathfrak{F}) \setminus \{\mathfrak{n}/x\}) \cup \{\mathfrak{p}_a\} \leftarrow \mathfrak{D}' \rangle$ where $\mathfrak{D}' = (\mathfrak{D} \setminus \{\mathfrak{p}\}) \cup \{\overrightarrow{(\mathfrak{m}/x^a)}\}$.
  – Either $\mathfrak{n} \in \mathfrak{F}$ [**reduction step**]. In this case, we have

$$s\restriction\mathfrak{E} = s'\restriction\mathfrak{E}$$

so let us show that $s'\restriction\mathfrak{E} = \langle \mathfrak{E} \leftarrow (\!|\mathfrak{Cut}_{\mathfrak{F}|\mathfrak{D}}|\!) \rangle$. By induction hypothesis

$$s'\restriction\mathfrak{E} = \langle \mathfrak{E} \leftarrow (\!|\mathfrak{Cut}_{\mathfrak{F}'|\mathfrak{D}'}|\!) \rangle$$

where $\mathfrak{F}' = (\mathfrak{F} \setminus \{\mathfrak{n}/x\}) \cup \{\mathfrak{p}_a\}$, and by Lemma 2.2.4 we deduce

$$s'\restriction\mathfrak{E} = \langle \mathfrak{E} \leftarrow (\!|\mathfrak{Cut}_{\mathfrak{F}|\mathfrak{D}}|\!) \rangle \ ,$$

hence the result.
  – Or $\mathfrak{n} \in \mathfrak{E}$ [**commutation step**]. In this case, we have

$$s\restriction\mathfrak{E} = \kappa^-(s'\restriction\mathfrak{E}) \ .$$

By induction hypothesis

$$s'\restriction\mathfrak{E} = s'\restriction\mathfrak{E}' = \langle \mathfrak{E}' \leftarrow (\!|\mathfrak{Cut}_{\mathfrak{F}|\mathfrak{D}'}|\!) \rangle$$

where $\mathfrak{E}' = (\mathfrak{E} \setminus \{\mathfrak{n}/x\}) \cup \{\mathfrak{p}_a\}$. But we have

$$\begin{aligned}\langle \mathfrak{E} \leftarrow (\!|\mathfrak{Cut}_{\mathfrak{F}|\mathfrak{D}}|\!)\rangle &= \langle \mathfrak{E} \leftarrow (\!|\mathfrak{Cut}_{\mathfrak{F}|\mathfrak{D}'\cup\{\mathfrak{p}\}\setminus\{\overrightarrow{(\mathfrak{m}/x^a)}\}}|\!)\rangle \\ &= \langle \mathfrak{E} \leftarrow (\!|\mathfrak{Cut}_{\mathfrak{F}|\mathfrak{D}'}|\!) \cup \{\mathfrak{p}'\} \setminus \{\overrightarrow{(\mathfrak{m}'/x^a)}\}\rangle \\ &= \kappa^-\langle \mathfrak{E}' \leftarrow (\!|\mathfrak{Cut}_{\mathfrak{F}|\mathfrak{D}'}|\!)\rangle\end{aligned}$$

where $\mathfrak{p}'$ is the only positive design of $(\!|\mathfrak{Cut}_{\mathfrak{F}|\mathfrak{D}}|\!)$, and for each $i \leq \mathrm{ar}(a)$, $\mathfrak{m}'_i$ is the only negative design of $(\!|\mathfrak{Cut}_{\mathfrak{F}|\mathfrak{D}'}|\!)$ on variable x^a_i. Therefore

$$\langle \mathfrak{E} \leftarrow (\!|\mathfrak{Cut}_{\mathfrak{F}|\mathfrak{D}}|\!)\rangle = \kappa^-(s'\restriction\mathfrak{E}) = s\restriction\mathfrak{E} \quad ,$$

which ends the proof.

□

To conclude, let us stress that associativity for paths is necessary for describing, in the next chapter, the visitable paths of a behaviour constructed with a tensor $\otimes$ (Proposition 3.2.6).

3 | Connectives and Interaction

From now on, we go back to designs and behaviours of designs as introduced in Chapter 1. The present chapter is devoted to a precise analysis of behaviours constructed by logical connectives, in preparation for the subsequent study of inductive data types (Chapter 4) and functional types (Chapter 5). Our focus is two-fold: visitable paths on one side, regularity and purity on the other.

Thanks to internal completeness, we know what kind of designs a behaviour constructed by connectives consists of. It is then natural to wonder what kind of interactions can such a behaviour perform. The trace of an interaction being recorded in a path, asking this question amounts to wondering what the visitable paths of this behaviour are. After some preliminaries in Section 3.1, we describe in Section 3.2 the form of the visitable paths that each connective leads to. Following internal completeness, on which they rely, these results push further the study of the behaviours' structure.

Using these results, we can then study the interactive properties of behaviours constructed by connectives: regularity and purity. Indeed, these properties are both concerned with the visitable paths of a behaviour. We prove that all our connectives preserve regularity (Section 3.3), and that all of them except $\multimap$ preserve purity (Section 3.4). This study will be completed in the next chapter by showing that the two properties are preserved by least fixed points, thus that they hold for all (finite) data types.

Fouqueré and Quatrini [FQ16] proved similar results about visitable paths and the stability of regularity in the original framework of ludics, where the definition of connectives is slightly different. Sironi [Sir15] studied purity in original ludics as well. Our proofs are mostly different, though.

3.1 Preliminaries

First, we establish several results useful for the rest of the chapter.

3.1.a Paths and Observational Ordering

We show how to construct, from a visitable path, a design which is maximal for $\preceq$.

Lemma 3.1.1

Let s be a path of a design. There is a unique design maximal for $\preceq$ such that s is a path

of it. This design is noted $\ulcorner s \urcorner^c$.

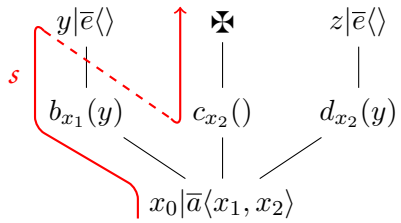
Proof. If s is a path of $\mathfrak{d}$, $\ulcorner s \urcorner^c$ is obtained from $\mathfrak{d}$ by replacing all positive subdesigns (possibly Ω) whose first positive action is not in s by $\maltese$. □

Notice that, actually, the design $\ulcorner s \urcorner^c$ does not depend on $\mathfrak{d}$ but only on path s.

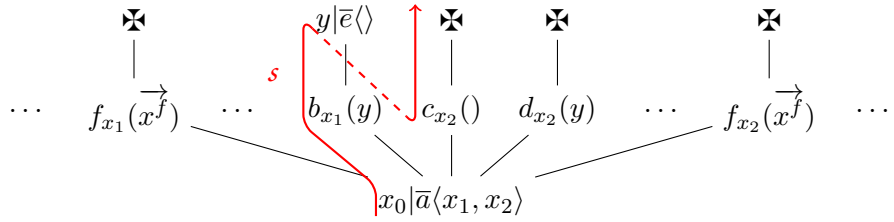
Example 3.1.2

Consider the design $\mathfrak{d}$ and the path s below:

$$\begin{array}{lcl} \mathfrak{d} & = & x|\overline{a}\langle b(y).(y|\overline{e}\langle\rangle), c().\maltese + d(z).(z|\overline{e}\langle\rangle)\rangle \ , \\ s & = & x|\overline{a}\langle x_1, x_2\rangle \qquad b_{x_1}(y) \qquad y|\overline{e}\langle\rangle \qquad c_{x_2}() \qquad \maltese \ . \end{array}$$



We have $\ulcorner s \urcorner^c = x|\overline{a}\langle b(y).(y|\overline{e}\langle\rangle) + \sum_{f \neq b} f(\overrightarrow{x^f}).\maltese, \sum_{f \in \mathcal{S}} f(\overrightarrow{x^f}).\maltese\rangle$.



Lemma 3.1.3

If $s \in V_{\mathbf{B}}$ *then* $\ulcorner s \urcorner^c \in \mathbf{B}$.

Proof. If $s \in V_{\mathbf{B}}$, there exists $\mathfrak{d} \in \mathbf{B}$ such that s is a path of $\mathfrak{d}$, thus $\mathfrak{d} \preceq \ulcorner s \urcorner^c$. The result then comes from monotonicity (Theorem 1.1.18). □

3.1.b More on Paths

Let $\mathbf{B}$ be a behaviour. We start by proving that the visitable paths of $\mathbf{B}$ are paths of designs in the incarnation of $\mathbf{B}$.

Lemma 3.1.4

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

Proof. Let $\mathfrak{e} \in \mathbf{B}^{\perp}$ such that $s = \langle \mathfrak{d} \leftarrow \mathfrak{e} \rangle$, and let $t = \langle |\mathfrak{d}| \leftarrow \mathfrak{e} \rangle$.

- Since $|\mathfrak{d}| \sqsubseteq \mathfrak{d}$, the path s cannot be a strict prefix of t, and s and t cannot differ on a positive action by Remark 2.2.7.
- If t is a strict prefix of s then it is positive-ended. So $\widetilde{s}$ and $\widetilde{t}$ are paths of $\mathfrak{e}$ differing on a positive action, which is impossible.
- If s and t differ on a negative action, say $u\kappa_1^-$ and $u\kappa_2^-$ are respective prefixes of s and t with $\kappa_1^- \neq \kappa_2^-$, then $\overline{u\kappa_1^-}$ and $\overline{u\kappa_2^-}$ are paths of $\mathfrak{e}$ differing on a positive action, which is impossible.

Thus we must have $s = t$, hence the result. □

The next lemma essentially asserts that the prefixes of visitable paths of $\mathbf{B}$ are also visitable in $\mathbf{B}$.

Lemma 3.1.5

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. $s'\maltese \in V_{\mathbf{B}}$).

Proof. Let $s = \langle \mathfrak{d} \leftarrow \mathfrak{e} \rangle$ where $\mathfrak{d} \in \mathbf{B}$ and $\mathfrak{e} \in \mathbf{B}^{\perp}$, and let s' be a prefix of s.

- If s' is negative-ended, let κ^+ be such that $s'\kappa^+$ is a prefix of s. The action κ^+ comes from $\mathfrak{d}$. Consider the design $\mathfrak{d}'$ obtained from $\mathfrak{d}$ by replacing the positive subdesign of $\mathfrak{d}$ starting on action κ^+ with $\maltese$. Since $\mathfrak{d} \preceq \mathfrak{d}'$, by monotonicity $\mathfrak{d}' \in \mathbf{B}$. Moreover $s'\maltese = \langle \mathfrak{d}' \leftarrow \mathfrak{e} \rangle$, hence the result.
- If s' is positive-ended then either $s' = s$ and there is nothing to prove or s' is a strict prefix of s, so assume we are in the second case. s' is $\maltese$-free, hence $\overline{s'}$ is a negative-ended prefix of $\widetilde{s} \in V_{\mathbf{B}^{\perp}}$. Using the argument above, it comes $\widetilde{s'} = \overline{s'}\maltese \in V_{\mathbf{B}^{\perp}}$, thus $s' \in V_{\mathbf{B}}$.

□

Now we show that, after following any positive-ended prefix s' of a visitable path $s \in V_{\mathbf{B}}$, a design $\mathfrak{d}$ of $\mathbf{B}$ is always receptive to the next action κ^- of the path s.

Lemma 3.1.6

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 $\mathfrak{d}$.

Proof. There exist $\mathfrak{d}_0 \in \mathbf{B}$ and $\mathfrak{e}_0 \in \mathbf{B}^{\perp}$ such that $s = \langle \mathfrak{d}_0 \leftarrow \mathfrak{e}_0 \rangle$. Let $s'\kappa^-$ be a prefix of s, and let $\mathfrak{d} \in \mathbf{B}$ such that s' is a path of $\mathfrak{d}$. Since $\overline{s'}$ is a prefix of a path of $\mathfrak{e}_0$, s' is a prefix of $\langle \mathfrak{d} \leftarrow \mathfrak{e}_0 \rangle$. We cannot have $s' = \langle \mathfrak{d} \leftarrow \mathfrak{e}_0 \rangle$, otherwise $\widetilde{s'} = \overline{s'}\maltese$ and $\overline{s'\kappa^-}$ would be paths

of $\mathfrak{e}_0$ differing on a positive action, which is impossible by Remark 2.2.7. Thus there exists κ'^- such that $s'\kappa'^-$ is a prefix of $\langle \mathfrak{d} \leftarrow \mathfrak{e}_0 \rangle$, which is a path of $\mathfrak{d}$, and necessarily $\kappa^- = \kappa'^-$. Finally $s'\kappa^-$ is a prefix of a path of $\mathfrak{d}$. □

3.1.c An Alternative Definition of Regularity

We give, in Proposition 3.1.10, another definition of regularity which is equivalent to Definition 1.3.4 and best suited to conduct the proofs. Indeed, given a behaviour $\mathbf{B}$ such that $V_{\mathbf{B}}$ and $V_{\mathbf{B}^\perp}$ are stable by shuffle, Proposition 3.1.10 states that we only need to check that some specific sequences, the *bi-views*, are visitable in $\mathbf{B}$ to ensure that all paths of $|\mathbf{B}|$ and $|\mathbf{B}^\perp|$ are visitable, in other words to ensure that $\mathbf{B}$ is regular. Note that the bi-views (terminology coming from [Lau04]) correspond to *trivial chronicles* in [FQ16].

Definition 3.1.7 (*Bi-view*)

- A **bi-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 **bi-view of** an aj-sequence is defined inductively by:

$$\begin{aligned}
\langle \epsilon \rangle &= \epsilon && \text{empty sequence} \ , \\
\langle s\maltese \rangle &= \langle s \rangle \maltese \ , && \\
\langle s\kappa \rangle &= \kappa && \text{if } \kappa \neq \maltese \text{ initial} \ , \\
\langle s\kappa \rangle &= \langle s_0 \rangle \kappa && \text{if } \kappa \neq \maltese \text{ justified, where } s_0 \text{ prefix of } s \text{ ending on } \mathrm{just}(\kappa) \ .
\end{aligned}$$

  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.
- The **bi-views of a design** $\mathfrak{d}$ are the bi-views of its paths (or of its views). In particular, ϵ is a bi-view of negative designs only.
- The bi-views of designs in $|\mathbf{B}|$ are called the **bi-views of** $\mathbf{B}$.

Note that $\langle s \rangle = \ulcorner \llcorner s \lrcorner \urcorner = \llcorner \ulcorner s \urcorner \lrcorner$.

Lemma 3.1.8

1. *Every view is in the anti-shuffle of bi-views.*
2. *Every path is in the shuffle of views.*

Proof.

1. Let $\mathbb{v}$ be a view, the result is shown by induction on $\mathbb{v}$:
   - If $\mathbb{v} = \epsilon$ or $\mathbb{v} = \kappa$, it is itself a bi-view, hence the result.
   - Suppose $\mathbb{v} = \mathbb{v}'\kappa$ with $\mathbb{v}' \neq \epsilon$ and $\mathbb{v}' \in \mathbb{t}_1 \between \ldots \between \mathbb{t}_n$ where the $\mathbb{t}_i$ are bi-views.
     - If κ is negative, as $\mathbb{v}$ is a view, the action κ is justified by the last action of $\mathbb{v}'$, say κ^+. Hence κ^+ is the last action of some bi-view $\mathbb{t}_{i_0}$. Hence $\mathbb{v} \in \mathbb{t}_1 \between \ldots \between \mathbb{t}_{i_0-1} \between (\mathbb{t}_{i_0}\kappa) \between \mathbb{t}_{i_0+1} \between \ldots \between \mathbb{t}_n$.

- If κ is positive, either it is initial and $\mathbb{v} \in \mathbb{t}_1 \pitchfork \ldots \pitchfork \mathbb{t}_n \pitchfork \kappa$ with κ a bi-view, or it is justified by κ^- in $\mathbb{v}'$. In the last case, there exists a unique i_0 such that κ^- appears in $\mathbb{t}_{i_0}$, so let $\mathbb{t}\kappa^-$ be the prefix of $\mathbb{t}_{i_0}$ ending with κ^-. We have that $\mathbb{v} \in \mathbb{t}_1 \pitchfork \ldots \pitchfork \mathbb{t}_n \pitchfork (\mathbb{t}\kappa^- \kappa)$ where $\mathbb{t}\kappa^-\kappa$ is a bi-view.

2. Similar reasoning as above, but replacing $\pitchfork$ by $\sqcup\!\sqcup$, "bi-view" by "view", "view" by "path", and exchanging the role of the polarities of actions.

□

Remark 3.1.9

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

Proposition 3.1.10

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

Remember that $V_{\mathbf{B}^\perp} = \widetilde{V_{\mathbf{B}}}$ (Remark 1.2.13), hence the equivalence between $V_{\mathbf{B}^\perp}$ being stable under $\sqcup\!\sqcup$ and $V_{\mathbf{B}}$ under $\pitchfork$.

Proof. Let $\mathbf{B}$ be a behaviour.
($\Rightarrow$) Suppose $\mathbf{B}$ is regular, and let $\mathbb{t}$ be a positive-ended bi-view of $\mathbf{B}$. There exists a view $\mathbb{v}$ of a design $\mathfrak{d} \in |\mathbf{B}|$ such that $\mathbb{t}$ is a subsequence of $\mathbb{v}$, and such that $\mathbb{v}$ ends with the same action as $\mathbb{t}$. Since $\mathbb{v}$ is a view of $\mathfrak{d}$, $\mathbb{v}$ is in particular a path of $\mathfrak{d}$, hence by regularity $\mathbb{v} \in V_{\mathbf{B}}$. There exists $\mathfrak{e} \in \mathbf{B}^\perp$ such that $\mathbb{v} = \langle \mathfrak{d} \leftarrow \mathfrak{e} \rangle$, and by Lemma 3.1.4 we can even take $\mathfrak{e} \in |\mathbf{B}^\perp|$. Since $\tilde{\mathbb{v}}$ is a path of $\mathfrak{e}$, $\ulcorner \tilde{\mathbb{v}} \urcorner$ is a view of $\mathfrak{e}$. But notice that $\ulcorner \tilde{\mathbb{v}} \urcorner = \ulcorner \tilde{\mathbb{t}} \urcorner = \tilde{\mathbb{t}}$ by definition of a view and of a bi-view. We deduce that $\tilde{\mathbb{t}}$ is a view (and in particular a path) of $\mathfrak{e}$, hence $\tilde{\mathbb{t}} \in V_{\mathbf{B}^\perp}$ by regularity. Finally, $\mathbb{t} \in V_{\mathbf{B}}$.
($\Leftarrow$) Assume the two conditions of the statement. Let s be a path of some design of $|\mathbf{B}|$. By Lemma 3.1.8, we know that there exist views $\mathbb{v}_1, \ldots, \mathbb{v}_n$ such that $s \in \mathbb{v}_1 \sqcup\!\sqcup \cdots \sqcup\!\sqcup \mathbb{v}_n$, and for each $\mathbb{v}_i$ there exist bi-views $\mathbb{t}_{i,1}, \ldots, \mathbb{t}_{i,m_i}$ such that $\mathbb{v}_i \in \mathbb{t}_{i,1} \pitchfork \ldots \pitchfork \mathbb{t}_{i,m_i}$. By hypothesis each $\mathbb{t}_{i,j}$ is visitable in $\mathbf{B}$, hence as $V_{\mathbf{B}}$ is stable by anti-shuffle, $\mathbb{v}_i \in V_{\mathbf{B}}$. Thus as $V_{\mathbf{B}}$ is stable by shuffle, $s \in V_{\mathbf{B}}$. Similarly the paths of designs of $|\mathbf{B}^\perp|$ are visitable in $\mathbf{B}^\perp$. Hence the result. □

3.2 Visitable Paths and Connectives

The goal of this section is to make explicit the form of the visitable paths for behaviours constructed by logical connectives. First recall the notations given at the beginning of § 1.1.e (page 28), and we also introduce new ones.

Notation

- Let us note $\kappa_{\blacktriangledown} = x_0|\blacktriangledown\langle x\rangle$, $\kappa_{\blacktriangle} = \blacktriangle_{x_0}(x)$, $\kappa_{\bullet} = x_0|\bullet\langle x, y\rangle$ and $\kappa_{\iota_i} = x_0|\iota_i\langle x_i\rangle$ for $i \in \{1, 2\}$.
- Given a path s and a fresh variable x, define $s^x = s[x/x_0]$ where the substitution affects both free variables and variables that are addresses of negative initial actions; for example:

$$\text{if} \quad s = x_0|\overline{a}\langle y\rangle \quad b_y() \quad \maltese \qquad \text{then} \quad s^x = x|\overline{a}\langle y\rangle \quad b_y() \quad \maltese$$

  replacing a free variable – that is, the address of a positive initial action;

$$\text{if} \quad t = a_{x_0}(y) \quad y|\overline{b}\langle\rangle \qquad \text{then} \quad t^x = a_x(y) \quad y|\overline{b}\langle\rangle$$

  replacing a "free" negative place – that is, the address of a negative initial action.
- Given a design $\mathfrak{d}$ and a fresh variable x, we write $\mathfrak{d}^x$ for $\mathfrak{d}[x/x_0]$; if $\mathfrak{d} = \mathfrak{n}$ is negative then we consider that the substitution also affects views and paths: $\mathbb{V}[\mathfrak{n}^x] = \mathbb{V}[\mathfrak{n}]_x$, thus the paths of $\mathfrak{n}^x$ are $\{s^x \mid s \text{ path of } \mathfrak{n}\}$.
- Write $V^x_{\mathbf{B}}$ for $\{s^x \mid s \in V_{\mathbf{B}}\}$, and remark that $V^x_{\mathbf{B}} = V_{\mathbf{B}^x}$.
- Recall that, for an action κ and a set of paths V, we write $\kappa V = \{\kappa s \mid s \in V\}$.

We show how we can, from the visitable paths of behaviours $\mathbf{M}, \mathbf{N}, \mathbf{P}$, deduce the ones of $\downarrow\mathbf{N}, \uparrow\mathbf{P}, \mathbf{M} \oplus \mathbf{N}, \mathbf{M} \otimes \mathbf{N}$ and $\mathbf{N} \multimap \mathbf{P}$, in other words we give a compositional description of the visitable paths. More precisely, we prove the following:

$$V_{\downarrow\mathbf{N}} = \kappa_{\blacktriangledown} V^x_{\mathbf{N}} \cup \{\maltese\} \qquad \text{(Proposition 3.2.1)},$$
$$V_{\uparrow\mathbf{P}} = \kappa_{\blacktriangle} V^x_{\mathbf{P}} \cup \{\epsilon\} \qquad \text{(Proposition 3.2.1)},$$
$$V_{\mathbf{M}\oplus\mathbf{N}} = \kappa_{\iota_1} V^{x_1}_{\mathbf{M}} \cup \kappa_{\iota_2} V^{x_2}_{\mathbf{N}} \cup \{\maltese\} \qquad \text{(Proposition 3.2.4)},$$
$$V_{\mathbf{M}\otimes\mathbf{N}} = \kappa_{\bullet}(V^x_{\mathbf{M}} \sqcup V^y_{\mathbf{N}}) \cup \{\maltese\} \quad \text{if } \mathbf{M} \text{ and } \mathbf{N} \text{ regular} \qquad \text{(Proposition 3.2.8)}.$$

We also prove the general form of the visitable paths of $\mathbf{M} \otimes \mathbf{N}$, which is not as simple as in the regular case (Proposition 3.2.6). Finally, the visitable paths of $\mathbf{N} \multimap \mathbf{P}$ are easily deduced from those of the tensor in both the general and the regular case (Corollaries 3.2.7 and 3.2.9).

These results are necessary for proving the stability of regularity and purity in the next sections. Internal completeness plays a central role in the proofs.

3.2.a Shifts

The visitable paths of the shifts have a really simple form: we essentially just need to add an action at the beginning of the paths of $\mathbf{N}$ or $\mathbf{P}$.

Proposition 3.2.1 (*Visitable paths of the shifts*)

1. $V_{\downarrow\mathbf{N}} = \kappa_{\blacktriangledown} V^x_{\mathbf{N}} \cup \{\maltese\}$.
2. $V_{\uparrow\mathbf{P}} = \kappa_{\blacktriangle} V^x_{\mathbf{P}} \cup \{\epsilon\}$.

We need two lemmas before proving this proposition.

Lemma 3.2.2

1. $(\blacktriangle(x).(\mathbf{N}^\perp)^x)^\perp \subseteq \blacktriangledown\langle\mathbf{N}\rangle \cup \{\maltese\}$.
2. $\blacktriangle(x).(\mathbf{N}^\perp)^x \subseteq \blacktriangledown\langle\mathbf{N}\rangle^\perp$.

Proof. Let $E = \blacktriangledown\langle\mathbf{N}\rangle$ and $F = \blacktriangle(x).(\mathbf{N}^\perp)^x$. To show the lemma, we must show $F^\perp \subseteq E \cup \{\maltese\}$ and $F \subseteq E^\perp$.

1. Let $\mathfrak{q} \in F^\perp$. If $\mathfrak{q} \neq \maltese$, $\mathfrak{q}$ is necessarily of the form $\blacktriangledown\langle\mathfrak{n}\rangle$ where $\mathfrak{n}$ is a negative atomic design. For every design $\mathfrak{p} \in \mathbf{N}^\perp$, we have $\blacktriangle(x).\mathfrak{p}^x \in F$ and $\mathfrak{q}[\blacktriangle(x).\mathfrak{p}^x/x_0] \rightsquigarrow \mathfrak{p}[\mathfrak{n}/x_0]$, thus $(\!|\mathfrak{q}[\blacktriangle(x).\mathfrak{p}^x/x_0]|\!) = (\!|\mathfrak{p}[\mathfrak{n}/x_0]|\!) = \maltese$ since $\mathfrak{q} \perp \blacktriangle(x).\mathfrak{p}^x$. We deduce $\mathfrak{n} \in \mathbf{N}$, hence $\mathfrak{q} \in E$.
2. Let $\mathfrak{m} = \blacktriangle(x).\mathfrak{p}^x \in F$. For every design $\mathfrak{n} \in \mathbf{N}$, we have $\blacktriangledown\langle\mathfrak{n}\rangle[\mathfrak{m}/x_0] \rightsquigarrow \mathfrak{p}[\mathfrak{n}/x_0]$, thus $(\!|\blacktriangledown\langle\mathfrak{n}\rangle[\mathfrak{m}/x_0]|\!) = (\!|\mathfrak{p}[\mathfrak{n}/x_0]|\!) = \maltese$ since $\mathfrak{p} \in \mathbf{N}^\perp$ and $\mathfrak{n} \in \mathbf{N}$. Hence $\mathfrak{m} \in E^\perp$.

□

The following lemma states that, indeed, $\downarrow$ and $\uparrow$ are dual connectives.

Lemma 3.2.3

$\uparrow\mathbf{P} = (\downarrow\mathbf{P}^\perp)^\perp$.

Proof. If we take $\mathbf{N} = \mathbf{P}^\perp$, Lemma 3.2.2 gives us:

1. $(\blacktriangle(x).\mathbf{P}^x)^\perp \subseteq \blacktriangledown\langle\mathbf{P}^\perp\rangle \cup \{\maltese\}$ and
2. $\blacktriangle(x).\mathbf{P}^x \subseteq \blacktriangledown\langle\mathbf{P}^\perp\rangle^\perp$.

Let $E = \blacktriangledown\langle\mathbf{P}^\perp\rangle$, and let $F = \blacktriangle(x).\mathbf{P}^x$. By definition $\uparrow\mathbf{P} = F^{\perp\perp}$. From (2) we deduce $F^{\perp\perp} \subseteq E^\perp$, and from (1) $E^\perp = (E \cup \{\maltese\})^\perp \subseteq F^{\perp\perp}$. Hence $\uparrow\mathbf{P} = F^{\perp\perp} = E^\perp = (\downarrow\mathbf{P}^\perp)^\perp$. □

Proof (Proposition 3.2.1).

1. ($\subseteq$) Let $\mathfrak{q} \in \downarrow\mathbf{N}$ and $\mathfrak{m} \in (\downarrow\mathbf{N})^\perp$, let us show that $\langle\mathfrak{q} \leftarrow \mathfrak{m}\rangle \in \kappa_{\blacktriangledown} V^x_{\mathbf{N}} \cup \{\maltese\}$. By Lemma 3.2.3, $\mathfrak{m} \in \uparrow\mathbf{N}^\perp$. If $\mathfrak{q} = \maltese$ then $\langle\mathfrak{q} \leftarrow \mathfrak{m}\rangle = \maltese$. Otherwise, by Theorem 1.1.21, $\mathfrak{q} = \blacktriangledown\langle\mathfrak{n}\rangle$ with $\mathfrak{n} \in \mathbf{N}$. We have $\langle\mathfrak{q} \leftarrow \mathfrak{m}\rangle = \langle\mathfrak{q} \leftarrow |\mathfrak{m}|\rangle$ by Lemma 3.1.4 , where $|\mathfrak{m}| \in \blacktriangle(x).|(\mathbf{N}^\perp)^x|$ by Theorem 1.1.21, hence $|\mathfrak{m}|$ is of the form $|\mathfrak{m}| = \blacktriangle(x).\mathfrak{p}^x$ with $\mathfrak{p} \in \mathbf{N}^\perp$. By definition $\langle\mathfrak{q} \leftarrow |\mathfrak{m}|\rangle = \kappa_{\blacktriangledown}\langle\mathfrak{n}^x \leftarrow \mathfrak{p}^x\rangle$, where $\langle\mathfrak{n}^x \leftarrow \mathfrak{p}^x\rangle \in V^x_{\mathbf{N}}$.
   ($\supseteq$) Indeed $\maltese \in V_{\downarrow\mathbf{N}}$. Now let $s \in \kappa_{\blacktriangledown} V^x_{\mathbf{N}}$. There exist $\mathfrak{n} \in \mathbf{N}$ and $\mathfrak{p} \in \mathbf{N}^\perp$ such that $s = \kappa_{\blacktriangledown}\langle\mathfrak{n}^x \leftarrow \mathfrak{p}^x\rangle$. Note that $\blacktriangledown\langle\mathfrak{n}\rangle \in \blacktriangledown\langle\mathbf{N}\rangle$ and $\blacktriangle(x).\mathfrak{p}^x \in \blacktriangle(x).(\mathbf{N}^\perp)^x$. By Lemma 3.2.3, $\uparrow\mathbf{N}^\perp = (\downarrow\mathbf{N})^\perp$, hence $\blacktriangledown\langle\mathfrak{n}\rangle \perp \blacktriangle(x).\mathfrak{p}^x$. Moreover $\langle\blacktriangledown\langle\mathfrak{n}\rangle \leftarrow \blacktriangle(x).\mathfrak{p}^x\rangle = \kappa_{\blacktriangledown}\langle\mathfrak{n}^x \leftarrow \mathfrak{p}^x\rangle = s$, therefore $s \in V_{\downarrow\mathbf{N}}$.
2. By Lemma 3.2.3, the previous item, and the fact that $V_{\mathbf{B}} = \widetilde{V_{\mathbf{B}^\perp}}$ for any behaviour $\mathbf{B}$, we have: $V_{\uparrow\mathbf{P}} = \widetilde{V_{(\uparrow\mathbf{P})^\perp}} = \widetilde{V_{\downarrow\mathbf{P}^\perp}} = (\widetilde{\kappa_{\blacktriangledown} V^x_{\mathbf{P}^\perp} \cup \{\maltese\}}) = \kappa_{\blacktriangle}\widetilde{V^x_{\mathbf{P}^\perp}} \cup \{\epsilon\} = \kappa_{\blacktriangle} V^x_{\mathbf{P}} \cup \{\epsilon\}$.

□

3.2.b Plus

The visitable paths of the plus are simple too: we add a different action at the beginning of a path depending on if it comes from $V_{\mathbf{M}}$ or $V_{\mathbf{N}}$.

Proposition 3.2.4 (*Visitable paths of the plus*)
$V_{\mathbf{M}\oplus\mathbf{N}} = \kappa_{\iota_1} V_{\mathbf{M}}^{x_1} \cup \kappa_{\iota_2} V_{\mathbf{N}}^{x_2} \cup \{\maltese\}$.

***Proof*.** Remark that

$$\mathbf{M} \oplus \mathbf{N} = (\iota_1\langle \mathbf{M}\rangle \cup \{\maltese\}) \cup (\iota_2\langle \mathbf{N}\rangle \cup \{\maltese\})$$

is the union of behaviours $\oplus_1\mathbf{M}$ and $\oplus_2\mathbf{N}$, which correspond respectively to $\downarrow\mathbf{M}$ and $\downarrow\mathbf{N}$ with a different name for the first action. Moreover,

$$\begin{aligned}(\mathbf{M} \oplus \mathbf{N})^\perp &= \{\mathfrak{n} \mid \mathfrak{n}\restriction\pi_1 \in \pi_1(x).(\mathbf{M}^\perp)^x \text{ and } \mathfrak{n}\restriction\pi_2 \in \pi_2(x).(\mathbf{N}^\perp)^x\} \\ &= (\&_1\mathbf{M}^\perp) \cap (\&_2\mathbf{N}^\perp)\end{aligned}$$

where the behaviours $\&_1\mathbf{M}^\perp$ and $\&_2\mathbf{N}^\perp$ correspond to $\uparrow\mathbf{M}^\perp$ and $\uparrow\mathbf{N}^\perp$ with different names; note also that for every $\mathfrak{d} \in |\,\&_1\,\mathbf{M}^\perp|$ (resp. $|\,\&_2\,\mathbf{N}^\perp|$) there exists $\mathfrak{d}' \in (\mathbf{M}\oplus\mathbf{N})^\perp$ such that $\mathfrak{d} \sqsubseteq \mathfrak{d}'$, in other words such that $\mathfrak{d} = |\mathfrak{d}'|_{\&_1\mathbf{M}^\perp}$ (resp. $|\mathfrak{d}'|_{\&_2\mathbf{N}^\perp}$). Therefore the proof can be conducted similarly to the one of Proposition 3.2.1(1). □

3.2.c Tensor and Linear Map

The case of the tensor is much trickier than the shifts or the plus. Though all the visitable paths of $\mathbf{M}\otimes\mathbf{N}$ (except $\maltese$) are obtained by shuffling a path of $V_{\mathbf{M}}$ and a path of $V_{\mathbf{N}}$, tensor and shuffle do not match exactly. More precisely, we have

$$V_{\mathbf{M}\otimes\mathbf{N}} \subseteq \kappa_\bullet(V_{\mathbf{M}}^x \sqcup\!\sqcup V_{\mathbf{N}}^y) \cup \{\maltese\} \ ,$$

but the converse inclusion does not hold in general, as shown in the following example; we will however prove in the next subsection that it holds if both behaviours are regular (Proposition 3.2.8).

Example 3.2.5

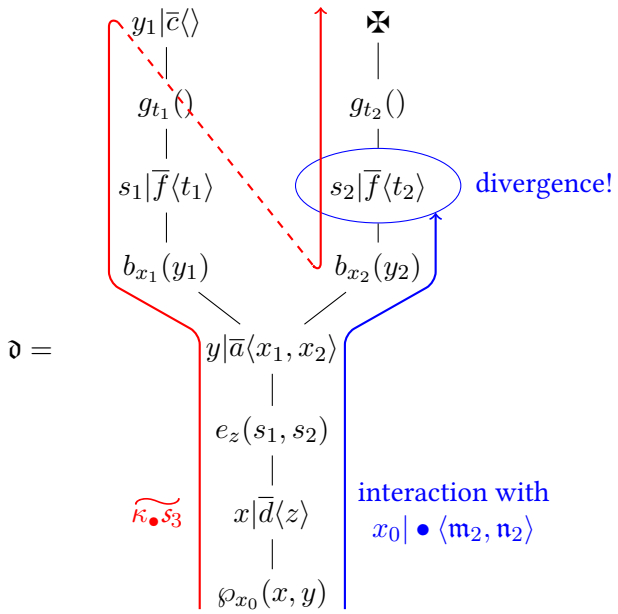
Go back to the behaviours $\mathbf{A}$ and $\mathbf{B}$ of Figure 4 (page 38), and recall that $\mathbf{A}$ is regular but $\mathbf{B}$ is not. We have

$$\begin{aligned} s_1 &= a_{x_0}(x_1,x_2)\quad x_1|\overline{b}\langle y_1\rangle\quad c_{y_1}()\quad x_2|\overline{b}\langle y_2\rangle && \in V_{\mathbf{A}} \ , \\ s_2 &= d_{x_0}(z)\quad z|\overline{e}\langle s_1,s_2\rangle\quad f_{s_1}(t_1)\quad t_1|\overline{g}\langle\rangle\quad f_{s_2}(t_2)\quad t_2|\overline{g}\langle\rangle && \in V_{\mathbf{B}} \ , \end{aligned}$$

and we can define the path

$$\begin{aligned} s_3 = \ & d_x(z)\quad z|\overline{e}\langle s_1,s_2\rangle\quad a_y(x_1,x_2)\quad x_1|\overline{b}\langle y_1\rangle\quad f_{s_1}(t_1)\quad t_1|\overline{g}\langle\rangle \\ & c_{y_1}()\quad x_2|\overline{b}\langle y_2\rangle\quad f_{s_2}(t_2)\quad t_2|\overline{g}\langle\rangle \end{aligned}$$

which satisfies $s_3 \in {s_1}^x \sqcup {s_2}^y$, but $\kappa_\bullet s_3 \notin V_{\mathbf{A}\otimes\mathbf{B}}$. Indeed, we can check that for any design $\mathfrak{d}$ such that $\widetilde{\kappa_\bullet s_3}$ is a path of $\mathfrak{d}$, we have $\mathfrak{d} \not\perp x_0| \bullet \langle \mathfrak{m}_2, \mathfrak{n}_2\rangle$ (where $\mathfrak{m}_2 \in \mathbf{A}$ and $\mathfrak{n}_2 \in \mathbf{B}$ are the designs from Figure 4), hence $\mathfrak{d} \notin (\mathbf{A}\otimes\mathbf{B})^\perp$. The minimal (in the sense of the stable ordering $\sqsubseteq$) such design $\mathfrak{d}$ is the following:



The idea highlighted in the previous example is that a non-regular behaviour can induce an orientation (here $\mathbf{B}$ forces to visit $f_{s_1}(t_1)$ before $f_{s_2}(t_2)$), but this orientation might not be respected by all of the paths in the shuffle of the two behaviours of which we take the tensor. Thus we need a supplementary condition in order to capture the visitable paths of the tensor in general.

Proposition 3.2.6 (*Visitable paths of the tensor*)

$s \in V_{\mathbf{M}\otimes\mathbf{N}}$ if and only if the two conditions below are satisfied:

1. *$s \in \kappa_\bullet(V^x_{\mathbf{M}} \sqcup V^y_{\mathbf{N}}) \cup \{\maltese\}$,*
2. *for all $t \in V^x_{\mathbf{M}} \sqcup V^y_{\mathbf{N}}$, for all κ^- such that $\overline{\kappa_\bullet t \kappa^-}$ is a path of $\widetilde{\overline{s}}^{\overline{\Pi}\overline{\Gamma}c}$, we have*

$$t\kappa^-\maltese \in V^x_{\mathbf{M}} \sqcup V^y_{\mathbf{N}} \ .$$

This proposition is a joint work with Fouqueré and Quatrini; in [FQ16], they prove a similar result in the framework of original ludics. In our setting, the proof uses some material on multi-designs introduced in Chapter 2. In this proof, for all negative designs $\mathfrak{m}$ and $\mathfrak{n}$, let us write $\mathfrak{m}\otimes\mathfrak{n}$ instead of $x_0| \bullet \langle \mathfrak{m}, \mathfrak{n}\rangle$. The associativity for paths (Proposition 2.2.12), in particular, plays an important role in this proof, by relating the interaction path of $\mathfrak{m}\otimes\mathfrak{n}$ with a negative design $\mathfrak{n}_0$ to the one of $\mathfrak{m}$ (resp. $\mathfrak{n}$) with the normalisation of $\mathfrak{p}[\mathfrak{n}/y]$ (resp. $\mathfrak{p}[\mathfrak{m}/y]$), where $\mathfrak{p}$ is a subdesign of $\mathfrak{n}_0$; this is what allows us to decompose a path of $V_{\mathbf{M}\otimes\mathbf{N}}$

in two paths, one from $V_{\mathbf{M}}$ and one from $V_{\mathbf{N}}$.

Proof. ($\Rightarrow$) Let $s \in V_{\mathbf{M}\otimes\mathbf{N}}$. If $s = \maltese$ then both conditions are trivial, so suppose $s \neq \maltese$. By internal completeness (Theorem 1.1.21), there exist $\mathfrak{m} \in \mathbf{M}$, $\mathfrak{n} \in \mathbf{N}$ and $\mathfrak{n}_0 \in (\mathbf{M}\otimes\mathbf{N})^{\perp}$ such that

$$s = \langle \mathfrak{m} \otimes \mathfrak{n} \leftarrow \mathfrak{n}_0 \rangle \ .$$

Thus $\mathfrak{n}_0$ must be of the form $\mathfrak{n}_0 = \sum_{a\in\mathcal{S}} a(\overrightarrow{z^a}).\mathfrak{p}_a$ with $\mathfrak{p}_\wp \neq \Omega$ (remember that $\bullet = \overline{\wp}$), and we have

$$s = \kappa_\bullet s' \qquad \text{where} \qquad s' = \langle \{\mathfrak{m}^x, \mathfrak{n}^y\} \leftarrow \mathfrak{p}_\wp \rangle \ .$$

Let us prove both properties:

1. By Proposition 2.2.12,
$$s'\restriction \mathfrak{m}^x = \langle \mathfrak{m}^x \leftarrow (\!|\mathfrak{p}_\wp[\mathfrak{n}/y]|\!) \rangle \ ,$$
where $\mathfrak{m}^x \in \mathbf{M}^x$. Moreover
$$(\!|\mathfrak{p}_\wp[\mathfrak{n}/y]|\!) \in \mathbf{M}^{x\perp} \ ,$$
indeed: for any $\mathfrak{m}' \in \mathbf{M}$, we have
$$(\!|(\!|\mathfrak{p}_\wp[\mathfrak{n}/y]|\!)[\mathfrak{m}'/x]|\!) = (\!|\mathfrak{p}_\wp[\mathfrak{n}/y, \mathfrak{m}'/x]|\!) = (\!|(\mathfrak{m}' \otimes \mathfrak{n})[\mathfrak{n}_0/x_0]|\!) = \maltese$$
using associativity and one reduction step backwards. Thus
$$s'\restriction \mathfrak{m}^x \in V^x_{\mathbf{M}} \ .$$
Likewise,
$$s'\restriction \mathfrak{n}^y = \langle \mathfrak{n}^y \leftarrow (\!|\mathfrak{p}_\wp[\mathfrak{m}/x]|\!) \rangle \qquad \text{so} \qquad s'\restriction \mathfrak{n}^y \in V^y_{\mathbf{N}} \ .$$
Therefore
$$s' \in (V^x_{\mathbf{M}} \sqcup V^y_{\mathbf{N}}) \ .$$
2. Now let $t_1 \in V^x_{\mathbf{M}}$, $t_2 \in V^y_{\mathbf{N}}$. Suppose $t \in (t_1 \sqcup t_2)$ and κ^- is a negative action such that $\overline{\kappa_\bullet t \kappa^-}$ is a path of $\overset{\Vdash\sim\dashv^c}{s}$. Without loss of generality, suppose moreover that the action κ^- comes from $\mathfrak{m}^x$, and let us show that $t_1\kappa^-\maltese \in V^x_{\mathbf{M}}$. Let
$$t' = \langle \{ {}^{\Vdash}t_1{}^{\dashv^c}/x, {}^{\Vdash}t_2{}^{\dashv^c}/y\} \leftarrow \overset{\Vdash\sim\dashv^c}{s'} \rangle \ .$$
We show that $t_1\kappa^-$ is a prefix of $t'\restriction {}^{\Vdash}t_1{}^{\dashv^c}$ and that $t'\restriction {}^{\Vdash}t_1{}^{\dashv^c} \in V^x_{\mathbf{M}}$, leading to the conclusion by Lemma 3.1.5. Note the following facts:
   (a) $\overset{\Vdash\sim\dashv^c}{s} = \wp(x,y).\overset{\Vdash\sim\dashv^c}{s'} + \sum_{a\neq\wp} a(\overrightarrow{z^a}).\maltese$, and thus $\overset{\Vdash\sim\dashv^c}{s'} \neq \maltese$ (otherwise a path of the form $\overline{\kappa_\bullet t\kappa^-}$ cannot be path of $\overset{\Vdash\sim\dashv^c}{s}$).
   (b) t is a path of the multi-design $\{{}^{\Vdash}t_1{}^{\dashv^c}/x, {}^{\Vdash}t_2{}^{\dashv^c}/y\}$, and $\overline{t}$ is a prefix of a path of $\overset{\Vdash\sim\dashv^c}{s'}$ since $\overline{\kappa_\bullet t\kappa^-}$ is a path of $\overset{\Vdash\sim\dashv^c}{s}$, thus t is a prefix of t' by Lemma 2.2.8.

(c) Since t is a $\maltese$-free positive-ended prefix of t', we have that $\overline{\kappa_\bullet t}$ is a strict prefix of $\widetilde{\kappa_\bullet t'}$. Thus there exists a positive action κ_0^+ such that $\overline{\kappa_\bullet t}\kappa_0^+$ is a prefix of $\widetilde{\kappa_\bullet t'}$. The paths $\overline{\kappa_\bullet t\kappa^-}$ and $\overline{\kappa_\bullet t}\kappa_0^+$ are both paths of $\overset{\Gamma\widetilde{\ }\neg^c}{s}$, hence necessarily $\kappa_0^+ = \overline{\kappa^-}$. We deduce that $t\kappa^-$ is a prefix of t'.

(d) The sequence $t'\restriction^{\Gamma}t_1{}^{\neg^c}$ therefore starts with $(t\kappa^-)\restriction^{\Gamma}t_1{}^{\neg^c}$.

(e) We have $(t\kappa^-)\restriction^{\Gamma}t_1{}^{\neg^c} = (t\restriction^{\Gamma}t_1{}^{\neg^c})\kappa^-$ because, since κ^- comes from $\mathfrak{m}^x$, it is hereditarily justified by an initial negative action of address x, and thus κ^- appears in design ${}^{\Gamma}t_1{}^{\neg^c}$. We deduce $(t\kappa^-)\restriction^{\Gamma}t_1{}^{\neg^c} = (t\restriction^{\Gamma}t_1{}^{\neg^c})\kappa^- = t_1\kappa^-$.

(f) Moreover, by Proposition 2.2.12 $t'\restriction^{\Gamma}t_1{}^{\neg^c} = \langle {}^{\Gamma}t_1{}^{\neg^c} \leftarrow (\!| \overset{\Gamma\widetilde{\ }\neg^c}{s'} [{}^{\Gamma}t_2{}^{\neg^c}/y] |\!)\rangle$.

Hence (by d, e, f)

$$t_1\kappa^- \text{ is a prefix of } t'\restriction^{\Gamma}t_1{}^{\neg^c} = \langle {}^{\Gamma}t_1{}^{\neg^c} \leftarrow (\!| \overset{\Gamma\widetilde{\ }\neg^c}{s'} [{}^{\Gamma}t_2{}^{\neg^c}/y] |\!)\rangle \ .$$

Since ${}^{\Gamma}t_1{}^{\neg^c} \in \mathbf{M}^x$ (by Lemma 3.1.3) and $(\!| \overset{\Gamma\widetilde{\ }\neg^c}{s'} [{}^{\Gamma}t_2{}^{\neg^c}/y] |\!) \in \mathbf{M}^{x\perp}$ (by associativity, similar reasoning as item 1), we deduce

$$t'\restriction^{\Gamma}t_1{}^{\neg^c} \in V^x_{\mathbf{M}} \ .$$

Finally, by Lemma 3.1.5,

$$t_1\kappa^-\maltese \in V^x_{\mathbf{M}} \ .$$

($\Leftarrow$) Let $s \in \kappa_\bullet(V^x_{\mathbf{M}} \sqcup V^y_{\mathbf{N}}) \cup \{\maltese\}$ such that the second constraint is also satisfied. If $s = \maltese$ then $s \in V_{\mathbf{M}\otimes\mathbf{N}}$ is immediate, so suppose $s = \kappa_\bullet s'$ where $s' \in (V^x_{\mathbf{M}} \sqcup V^y_{\mathbf{N}})$. Consider the design $\overset{\Gamma\widetilde{\ }\neg^c}{s}$, and note that $\overset{\Gamma\widetilde{\ }\neg^c}{s} = \wp(x,y).\overset{\Gamma\widetilde{\ }\neg^c}{s'} + \sum_{a\neq\wp} a(\overrightarrow{z^a}).\maltese$. We show by contradiction that $\overset{\Gamma\widetilde{\ }\neg^c}{s} \in (\mathbf{M}\otimes\mathbf{N})^\perp$, leading to the conclusion.

Let $\mathfrak{m} \in \mathbf{M}$ and $\mathfrak{n} \in \mathbf{N}$ and suppose that $\mathfrak{m}\otimes\mathfrak{n} \not\perp \overset{\Gamma\widetilde{\ }\neg^c}{s}$. By Lemma 2.2.10 and given the form of design $\overset{\Gamma\widetilde{\ }\neg^c}{s}$, the interaction with $\mathfrak{m}\otimes\mathfrak{n}$ is finite and the cause of divergence is necessarily the existence of a path t and an action κ^- such that:

1. t is a path of $\mathfrak{m}\otimes\mathfrak{n}$,
2. $\overline{t\kappa^-}$ is a path of $\overset{\Gamma\widetilde{\ }\neg^c}{s}$
3. $t\kappa^-$ is not a path of $\mathfrak{m}\otimes\mathfrak{n}$.

Hence t is of the form $t = \kappa_\bullet t'$. Choose $\mathfrak{m}$ and $\mathfrak{n}$ such that t is of minimal length with respect to all such pairs of designs non orthogonal to $\overset{\Gamma\widetilde{\ }\neg^c}{s}$. Let $t_1 = t'\restriction\mathfrak{m}^x$ and $t_2 = t'\restriction\mathfrak{n}^y$, we have $t \in \kappa_\bullet(t_1 \sqcup t_2)$. Consider the design $\overset{\Gamma\widetilde{\ }\neg^c}{t}$, and note that $\overset{\Gamma\widetilde{\ }\neg^c}{t} = \wp(x,y).\overset{\Gamma\widetilde{\ }\neg^c}{t'} + \sum_{a\neq\wp} a(\overrightarrow{z^a}).\maltese$. We prove the following:

- $\overline{\overset{\Gamma\widetilde{\ }\neg^c}{t}} \in (\mathbf{M}\otimes\mathbf{N})^\perp$: By contradiction. Let $\mathfrak{m}' \in \mathbf{M}$ and $\mathfrak{n}' \in \mathbf{N}$ such that $\mathfrak{m}'\otimes\mathfrak{n}' \not\perp \overline{\overset{\Gamma\widetilde{\ }\neg^c}{t}}$. Again using Lemma 2.2.10, divergence occurs necessarily because there exists a path v and a negative action κ'^- such that:
  1. v is a path of $\mathfrak{m}'\otimes\mathfrak{n}'$,
  2. $\overline{v\kappa'^-}$ is a path of $\overset{\Gamma\widetilde{\ }\neg^c}{t}$,
  3. $v\kappa'^-$ is not a path of $\mathfrak{m}'\otimes\mathfrak{n}'$.

Since the views of $\overline{v\kappa'^-}$ are views of $\overline{t}$, $\overline{v\kappa'^-}$ is a path of $\ulcorner \tilde{s} \urcorner^c$. Thus $\mathfrak{m}' \otimes \mathfrak{n}' \not\perp \ulcorner \tilde{s} \urcorner^c$. Moreover v is strictly shorter than t, indeed: v and t are $\maltese$-free, and since $\overline{v\kappa'^-}$ is a path of $\ulcorner \tilde{t} \urcorner^c$ any action of $v\kappa'^-$ is an action of t. This contradicts the fact that t is of minimum length. We deduce $\ulcorner \tilde{t} \urcorner^c \in (\mathbf{M} \otimes \mathbf{N})^\perp$.

- $\underline{t \in \kappa_\bullet(V^x_{\mathbf{M}} \sqcup V^y_{\mathbf{N}})}$: We show $t_1 \in V^x_{\mathbf{M}}$, the proof of $t_2 \in V^y_{\mathbf{N}}$ being similar. Since t is a path of $\mathfrak{m} \otimes \mathfrak{n}$ and $\tilde{t}$ a path of $\ulcorner \tilde{t} \urcorner^c$, we have

$$t = \langle \mathfrak{m} \otimes \mathfrak{n} \leftarrow \ulcorner \tilde{t} \urcorner^c \rangle = \kappa_\bullet \langle \{\mathfrak{m}^x, \mathfrak{n}^y\} \leftarrow \ulcorner \tilde{t'} \urcorner^c \rangle \ ,$$

hence

$$t' = \langle \{\mathfrak{m}^x, \mathfrak{n}^y\} \leftarrow \ulcorner \tilde{t'} \urcorner^c \rangle \ .$$

Thus by Proposition 2.2.12

$$t_1 = t' \restriction \mathfrak{m}^x = \langle \mathfrak{m}^x \leftarrow (\![\ulcorner \tilde{t'} \urcorner^c [\mathfrak{n}/y]]\!) \rangle \ .$$

Moreover

$$(\![\ulcorner \tilde{t'} \urcorner^c [\mathfrak{n}/y]]\!) \in \mathbf{M}^{x\perp} \ ,$$

since for every design $\mathfrak{m}' \in \mathbf{M}$ we have

$$(\![(\![\ulcorner \tilde{t'} \urcorner^c [\mathfrak{n}/y]]\!) [\mathfrak{m}'/x]]\!) = (\![\ulcorner \tilde{t'} \urcorner^c [\mathfrak{n}/y, \mathfrak{m}'/x]]\!) = (\![(\mathfrak{m}' \otimes \mathfrak{n}) [\ulcorner \tilde{t} \urcorner^c / x_0]]\!) = \maltese \ ,$$

by associativity, one reduction step backwards, and the fact that $\ulcorner \tilde{t} \urcorner^c \in (\mathbf{M} \otimes \mathbf{N})^\perp$. It follows that $t_1 \in V^x_{\mathbf{M}}$.

- $\underline{t\kappa^-}$ is a path of $\mathfrak{m} \otimes \mathfrak{n}$: Remember that $\overline{t\kappa^-}$ is a path of $\ulcorner \tilde{s} \urcorner^c$, and we have just seen that $t \in \kappa_\bullet(V^x_{\mathbf{M}} \sqcup V^y_{\mathbf{N}})$. Using the second constraint of the proposition, we should have $t_1\kappa^-\maltese \in V^x_{\mathbf{M}}$ or $t_2\kappa^-\maltese \in V^y_{\mathbf{N}}$. Without loss of generality suppose $t_1\kappa^-\maltese \in V^x_{\mathbf{M}}$. Since $\mathfrak{m}^x \in \mathbf{M}^x$ and t_1 is a path of $\mathfrak{m}^x$, we should also have that $t_1\kappa^-$ is a prefix of a path of $\mathfrak{m}^x$ by Lemma 3.1.6, hence $\ulcorner t'\kappa^- \urcorner = \ulcorner t_1\kappa^- \urcorner$ is a view of $\mathfrak{m}^x$. But in this case, knowing that t is a path of $\mathfrak{m} \otimes \mathfrak{n}$ and that $\ulcorner t\kappa^- \urcorner = \kappa_\bullet \ulcorner t'\kappa^- \urcorner$ is a view of $\mathfrak{m} \otimes \mathfrak{n}$, we deduce that $t\kappa^-$ is a path of $\mathfrak{m} \otimes \mathfrak{n}$.

Last point contradicts the cause of divergence between $\mathfrak{m} \otimes \mathfrak{n}$ and $\ulcorner \tilde{s} \urcorner^c$. Hence $\ulcorner \tilde{s} \urcorner^c \in (\mathbf{M} \otimes \mathbf{N})^\perp$. Moreover, $\tilde{s}$ is a path of $\ulcorner \tilde{s} \urcorner^c$, and since $s \in \kappa_\bullet(V^x_{\mathbf{M}} \sqcup V^y_{\mathbf{N}})$ there exist $\mathfrak{m}_0 \in \mathbf{M}$ and $\mathfrak{n}_0 \in \mathbf{N}$ such that s is a path of $\mathfrak{m}_0 \otimes \mathfrak{n}_0$ (and $\mathfrak{m}_0 \otimes \mathfrak{n}_0 \in \mathbf{M} \otimes \mathbf{N}$). We deduce $s = \langle \mathfrak{m}_0 \otimes \mathfrak{n}_0 \leftarrow \ulcorner \tilde{s} \urcorner^c \rangle$, hence $s \in V_{\mathbf{M}\otimes\mathbf{N}}$. □

Corollary 3.2.7 (*Visitable paths of linear map*)

$s \in V_{\mathbf{N}\multimap\mathbf{P}}$ if and only if the two conditions below are satisfied:

1. *$\tilde{s} \in \kappa_\bullet(V^x_{\mathbf{N}} \sqcup \widetilde{V^y_{\mathbf{P}}}) \cup \{\maltese\}$,*
2. *for all $t \in V^x_{\mathbf{N}} \sqcup \widetilde{V^y_{\mathbf{P}}}$, for all κ^- such that $\overline{\kappa_\bullet t\kappa^-}$ is a path of $\ulcorner s \urcorner^c$, we have*

$$t\kappa^-\maltese \in V^x_{\mathbf{N}} \sqcup \widetilde{V^y_{\mathbf{P}}} \ .$$

Proof. Immediate from the definition of $\multimap$, Proposition 3.2.6 and Remark 1.2.13. □

3.2.d Tensor and Linear Map, Regular Case

In the regular case, Proposition 3.2.8 shows that the visitable paths of a behaviour $V_{\mathbf{M}\otimes\mathbf{N}}$ can be constructed as a simple shuffle of paths.

Proposition 3.2.8

If $\mathbf{M}$ *and* $\mathbf{N}$ *are regular then*

$$V_{\mathbf{M}\otimes\mathbf{N}} = \kappa_\bullet(V^x_{\mathbf{M}} \sqcup\!\sqcup V^y_{\mathbf{N}}) \cup \{\maltese\} \ .$$

Proof. Suppose $\mathbf{M}$ and $\mathbf{N}$ regular. Following Proposition 3.2.6, it suffices to show that any path $s \in \kappa_\bullet(V^x_{\mathbf{M}} \sqcup\!\sqcup V^y_{\mathbf{N}}) \cup \{\maltese\}$ satisfies the following condition: for all $t \in V^x_{\mathbf{M}} \sqcup\!\sqcup V^y_{\mathbf{N}}$, for all negative action κ^- such that $\overline{\kappa_\bullet t \kappa^-}$ is a path of $\overset{\Gamma\!\sim\!\neg^C}{s}$, $t\kappa^-\maltese \in V^x_{\mathbf{M}} \sqcup\!\sqcup V^y_{\mathbf{N}}$.

If $s = \maltese$, there is nothing to prove, so suppose $s = \kappa_\bullet s'$ where $s' \in V^x_{\mathbf{M}} \sqcup\!\sqcup V^y_{\mathbf{N}}$. Let $t \in V^x_{\mathbf{M}} \sqcup\!\sqcup V^y_{\mathbf{N}}$ and κ^- be such that $\overline{\kappa_\bullet t\kappa^-}$ is a path of $\overset{\Gamma\!\sim\!\neg^C}{s}$, that is $\overline{t\kappa^-}$ is a path of $\overset{\Gamma\!\sim\!\neg^C}{s'}$. Let $s_1, t_1 \in V^x_{\mathbf{M}}$ and $s_2, t_2 \in V^y_{\mathbf{N}}$ such that $s' \in s_1 \sqcup\!\sqcup s_2$ and $t \in t_1 \sqcup\!\sqcup t_2$. Without loss of generality, suppose κ^- is an action in s_1, thus we must show $t_1\kappa^-\maltese \in V^x_{\mathbf{M}}$. Notice that $\langle t_1\kappa^-\rangle = \langle \overset{\Gamma\!\sim\!\neg^C}{t\kappa^-}\rangle = \langle\kappa^-\rangle_{s'} = \langle\kappa^-\rangle_{s_1}$ (the second equality follows from the fact that $\overline{t\kappa^-}$ is a path of $\overset{\Gamma\!\sim\!\neg^C}{s'}$). Since $s_1 \in V^x_{\mathbf{M}}$, the sequence $\langle\kappa^-\rangle_{s_1} = \langle t_1\kappa^-\rangle$ is a bi-view of $\mathbf{M}^x$. Let $s_1'\kappa^-$ be the prefix of s_1 ending with κ^-. By Lemma 3.1.5 $s_1'\kappa^-\maltese \in V^x_{\mathbf{M}}$, so $\langle t_1\kappa^-\maltese\rangle = \langle s_1'\kappa^-\maltese\rangle$ is also a bi-view of $\mathbf{M}^x$; by regularity of $\mathbf{M}$, we deduce $\langle t_1\kappa^-\maltese\rangle \in V^x_{\mathbf{M}}$. We have $t_1\kappa^-\maltese \in t_1 \sqcup\!\sqcup \langle t_1\kappa^-\maltese\rangle$, where both t_1 and $\langle t_1\kappa^-\maltese\rangle$ are in $V^x_{\mathbf{M}}$, hence $t_1\kappa^-\maltese \in V^x_{\mathbf{M}}$ by regularity of $\mathbf{M}$. □

Corollary 3.2.9

If $\mathbf{N}$ *and* $\mathbf{P}$ *are regular then*

$$V_{\mathbf{N}\multimap\mathbf{P}} = \overbrace{\kappa_\bullet(V_{\mathbf{N}} \sqcup\!\sqcup \widetilde{V_{\mathbf{P}}})}^{\sim} \cup \{\epsilon\} \ .$$

3.3 Regularity and Connectives

Proposition 3.3.1

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

The proof of this result relies on internal completeness and on the form of visitable paths. We break it into several proofs, one for each connective. Like for visitable paths, the shifts

(Proposition 3.3.2) and the plus (Proposition 3.3.3) are rather simple. The tensor (Proposition 3.3.4) occasions the most difficulties; in particular, it requires the alternative formulation of regularity we gave at the beginning of this chapter (Proposition 3.1.10). As usual, the case of the linear map (Corollary 3.3.6) is easily deduced from the tensor.

Proposition 3.3.2

1. If $\mathbf{N}$ is regular then $\downarrow\mathbf{N}$ is regular.
2. If $\mathbf{P}$ is regular then $\uparrow\mathbf{P}$ is regular.

Proof.

1. Following Proposition 3.1.10:
   - By internal completeness, the bi-views of $\downarrow\mathbf{N}$ are of the form $\kappa_{\blacktriangledown}\mathbb{t}$ where $\mathbb{t}$ is a bi-view of $\mathbf{N}$. Since $\mathbf{N}$ is regular $\mathbb{t} \in V_{\mathbf{N}}$. Hence by Proposition 3.2.1, $\kappa_{\blacktriangledown}\mathbb{t} \in V_{\downarrow\mathbf{N}}$.
   - Since $V_{\mathbf{N}}$ is stable by shuffle, so is $V_{\downarrow\mathbf{N}} = \kappa_{\blacktriangledown}V^x_{\mathbf{N}}$ where $\kappa_{\blacktriangledown}$ is a positive action.
   - For all paths $\kappa_{\blacktriangle}s, \kappa_{\blacktriangle}t \in V_{(\downarrow\mathbf{N})^\perp} = \kappa_{\blacktriangle}V^x_{\mathbf{N}^\perp}$ such that $\kappa_{\blacktriangle}s \sqcup\!\sqcup \kappa_{\blacktriangle}t$ is defined, s and t necessarily start by the same positive action and $s \sqcup\!\sqcup t \subseteq V^x_{\mathbf{N}^\perp}$ because $V_{\mathbf{N}^\perp}$ (thus also $V^x_{\mathbf{N}^\perp}$) is stable by $\sqcup\!\sqcup$, hence $\kappa_{\blacktriangle}s \sqcup\!\sqcup \kappa_{\blacktriangle}t = \kappa_{\blacktriangle}(s \sqcup\!\sqcup t) \subseteq V_{(\downarrow\mathbf{N})^\perp}$.
2. If $\mathbf{P}$ is regular then $\mathbf{P}^\perp$ is too. Then, by the previous point, $\downarrow\mathbf{P}^\perp$ is regular, therefore so is $(\downarrow\mathbf{P}^\perp)^\perp$. By Lemma 3.2.3, this means that $\uparrow\mathbf{P}$ is regular.

□

Proposition 3.3.3

If $\mathbf{M}$ and $\mathbf{N}$ are regular then $\mathbf{M} \oplus \mathbf{N}$ is regular.

Proof. Similar to Proposition 3.3.2 (1), by the same remark we made in proof of Proposition 3.2.4. □

Proposition 3.3.4

If $\mathbf{M}$ and $\mathbf{N}$ are regular, then $\mathbf{M} \otimes \mathbf{N}$ is regular.

In order to prove this proposition, we need to be able to take the shuffle of two sequences for which we do not know if they satisfy O-visibility. We call **pre-path** a positive-ended P-visible aj-sequence. The **shuffle** $s \sqcup\!\sqcup t$ of two negative pre-paths s and t is the set of *paths* u formed with actions from s and t such that $u\restriction s = s$ and $u\restriction t = t$. The following lemma states that, in fact, if the shuffle of two pre-paths is non-empty then these sequences satisfy O-visibility. It will be useful for proving that certain sequences are paths in the proof of the proposition.

Lemma 3.3.5

Let s and t be negative pre-paths. If $s \sqcup\!\sqcup t \neq \emptyset$ then s and t are paths.

Proof. We prove the result by contradiction. Let us suppose that there exists a triple (s, t, u) such that s and t are two negative pre-paths, $u \in s \sqcup\!\sqcup t$ is a path, and at least one of s or t does not satisfy O-visibility, say s: there exists a negative action κ^- and a prefix $s_0\kappa^-$ of s such that the action κ^- is justified in s_0 but $\mathrm{just}(\kappa^-)$ does not appear in $\llcorner s_0 \lrcorner$.

We choose the triple (s, t, u) such that the length of u is minimal with respect to all such triples. Without loss of generality, we can assume that u and s are of the form $u = u_0\kappa^-\maltese$ and $s = s_0\kappa^-\maltese$ respectively. Indeed, if this is not true, u has a strict prefix of the form $u_0\kappa^-$; in this case we can replace (s, t, u) by the triple $(s_0\kappa^-\maltese, u_0{\restriction}t\,, u_0\kappa^-\maltese)$ which satisfies all the constraints, and where the length of $u_0\kappa^-\maltese$ is less or equal to the length of u.

Let $\kappa^+ = \mathrm{just}(\kappa^-)$. u is necessarily of the form $u = u_1\alpha^- u_2\alpha^+\kappa^-\maltese$ where α^- justifies α^+ and κ^+ appears in u_1, indeed:

- κ^+ does not appear immediately before κ^- in u, otherwise it would also be the case in s, contradicting the fact that κ^- is not O-visible in s.
- The action α^+ which is immediately before κ^- in u is justified by an action α^-, and κ^+ appears before α^- in u, otherwise κ^+ would not appear in $\llcorner u_0 \lrcorner$ and that would contradict the O-visibility of u.

Let us show by contradiction something that will be useful for the rest of this proof: in the path u, all the actions of u_2 (which cannot be initial) are justified in $\alpha^- u_2$. If it is not the case, let $u_1\alpha^- u_2'\beta$ be longest prefix of u such that β is an action of u_2 justified in u_1, and let β' be the following action (necessarily in $u_2\alpha^+$), thus β' is justified in $\alpha^- u_2$. If β' is positive (resp. negative) then β is negative (resp. positive), thus $\ulcorner u_1\alpha^- u_2'\beta \urcorner = \ulcorner u_1' \urcorner$ (resp. $\llcorner u_1\alpha^- u_2'\beta \lrcorner = \llcorner u_1' \lrcorner$) where u_1' is the prefix of u_1 ending on $\mathrm{just}(\beta)$. But then $\ulcorner u_1\alpha^- u_2'\beta \urcorner$ (resp. $\llcorner u_1\alpha^- u_2'\beta \lrcorner$) does not contain $\mathrm{just}(\beta')$: this contradicts the fact that u is a path, since P-visibility (resp. O-visibility) is not satisfied.

Now define $u' = u_1\kappa^-\maltese$, $s' = u'{\restriction}s$ and $t' = u'{\restriction}t$, and remark that:

- u' **is a path**, indeed, O-visibility for κ^- is still satisfied since $\llcorner u_1\alpha^- u_2\alpha^+\kappa^- \lrcorner = \llcorner u_1 \lrcorner\alpha^-\alpha^+\kappa^-$ and $\llcorner u_1\kappa^- \lrcorner = \llcorner u_1 \lrcorner\kappa^-$ both contain κ^+ in $\llcorner u_1 \lrcorner$.
- s' **and** t' **are pre-paths**, since s' is of the form $s' = s_1\kappa^-\maltese$ where $s_1 = u_1{\restriction}s$ is a prefix of s containing $\kappa^+ = \mathrm{just}(\kappa^-)$, and $t' = u'{\restriction}t = u_1{\restriction}t$ is a prefix of t.
- $u' \in s' \sqcup\!\sqcup t'$.
- s' **is not a path**: Note that s is of the form $s_1 s_2\kappa^-\maltese$ where $s_1 = u_1{\restriction}s$ and $s_2 = \alpha^- u_2\alpha^+{\restriction}s$. By hypothesis, s is not a path because κ^+ does not appear in $\llcorner s_1 s_2 \lrcorner$. But $\llcorner s_1 s_2 \lrcorner$ is of the form $\llcorner s_1 \lrcorner s_2'$, since all the actions of s_2 are hereditarily justified by the first (necessarily negative) action of s_2, indeed: we have proved that, in u, all the actions of u_2 (in particular those of s_2) were justified in $\alpha^- u_2$. Thus κ^+ does not appear in $\llcorner s_1 \lrcorner$, which means that O-visibility is not satisfied for κ^- in $s' = s_1\kappa^-\maltese$.

Hence the triple (s', t', u') satisfies all the conditions. This contradicts the minimality of u. □

Proof (Proposition 3.3.4). Following Proposition 3.1.10, we prove that the positive-ended bi-views of $\mathbf{M} \otimes \mathbf{N}$ are visitable in $\mathbf{M} \otimes \mathbf{N}$, and that $V_{\mathbf{M}\otimes\mathbf{N}}$ and $V_{(\mathbf{M}\otimes\mathbf{N})^\perp}$ are stable by shuffle.

Every bi-view of $\mathbf{M} \otimes \mathbf{N}$ is of the form $\kappa_\bullet \Bbbt$. It follows from internal completeness (incarnated form) that $\kappa_\bullet \Bbbt$ is a bi-view of $\mathbf{M} \otimes \mathbf{N}$ if and only if $\Bbbt$ is a bi-view either of $\mathbf{M}^x$ or of $\mathbf{N}^y$. As $\mathbf{M}$ (resp. $\mathbf{N}$) is regular, positive-ended bi-views of $\mathbf{M}^x$ (resp. $\mathbf{N}^y$) are in $V^x_{\mathbf{M}}$ (resp. $V^y_{\mathbf{N}}$). Thus by Proposition 3.2.8, positive-ended bi-views of $\mathbf{M} \otimes \mathbf{N}$ are in $V_{\mathbf{M}\otimes\mathbf{N}}$.

From Proposition 3.2.8, and from the fact that $\sqcup\!\sqcup$ is associative and commutative, we also have that $V_{\mathbf{M}\otimes\mathbf{N}}$ is stable by shuffle.

Let us prove that $V_{\mathbf{M}\otimes\mathbf{N}}$ is stable by anti-shuffle. Let $t, u \in V_{\mathbf{M}\otimes\mathbf{N}}$ and let $s \in t \pitchfork u$, we show that $s \in V_{\mathbf{M}\otimes\mathbf{N}}$ by induction on the length of s. Notice first that, from Proposition 3.2.8, there exist paths $t_1, u_1 \in V^x_{\mathbf{M}}$ and $t_2, u_2 \in V^y_{\mathbf{N}}$ such that $t \in \kappa_\bullet(t_1 \sqcup\!\sqcup t_2)$ and $u \in \kappa_\bullet(u_1 \sqcup\!\sqcup u_2)$. In the case s of length 1, either $s = \maltese$ or $s = \kappa_\bullet$, thus the result is immediate. So suppose $s = s'\kappa^-\kappa^+$ and by induction hypothesis $s' \in V_{\mathbf{M}\otimes\mathbf{N}}$. Hence, it follows from Proposition 3.2.8 that there exist paths $s_1 \in V^x_{\mathbf{M}}$ and $s_2 \in V^y_{\mathbf{N}}$ such that $s' \in \kappa_\bullet(s_1 \sqcup\!\sqcup s_2)$. Without loss of generality, we can suppose that κ^- is an action of t_1, hence of t. We study the different cases, proving each time either that $s \in V_{\mathbf{M}\otimes\mathbf{N}}$ or that the case is impossible.

- Either $\kappa^+ = \maltese$. In this case, $s_1\kappa^-\maltese$ is a negative pre-path. As s is a path and $s \in \kappa_\bullet(s_1\kappa^-\maltese \sqcup\!\sqcup s_2)$, by Lemma 3.3.5, we have moreover that $s_1\kappa^-\maltese$ is a path. Notice that $\kappa_\bullet\langle s_1\kappa^-\rangle = \langle\kappa^-\rangle_s = \langle\kappa^-\rangle_t = \kappa_\bullet\langle\kappa^-\rangle_{t_1}$. Hence $\langle s_1\kappa^-\rangle = \langle\kappa^-\rangle_{t_1}$ is a bi-view of $\mathbf{M}^x$. Let $\Bbbt\kappa^- = \langle s_1\kappa^-\rangle$. By Lemma 3.1.8, s_1 is a shuffle of anti-shuffles of bi-views of $\mathbf{M}^x$, one of which is the bi-view $\Bbbt$. Then remark that $s_1\kappa^-\maltese$ is also a shuffle of anti-shuffles of bi-views of $\mathbf{M}^x$, replacing $\Bbbt$ by $\Bbbt\kappa^-\maltese$ (note that $\Bbbt\kappa^-\maltese$ is indeed a bi-view of $\mathbf{M}^x$ since $\Bbbt\kappa^-\maltese = \langle t_0\kappa^-\maltese\rangle$ where $t_0\kappa^-$ is the prefix of t_1 ending with κ^-, and $t_0\kappa^-\maltese \in V^x_{\mathbf{M}}$ by Lemma 3.1.5). It follows from Proposition 3.1.10 that $s_1\kappa^-\maltese \in V^x_{\mathbf{M}}$. Finally, as $s \in \kappa_\bullet(s_1\kappa^-\maltese \sqcup\!\sqcup s_2)$ and by Proposition 3.2.8, we have $s \in V_{\mathbf{M}\otimes\mathbf{N}}$.
- Or κ^+ is a proper action of t_1, hence of t. Remark that $\ulcorner s'\kappa^-\urcorner = \ulcorner\kappa_\bullet s_1\kappa^-\urcorner = \kappa_\bullet\ulcorner s_1\kappa^-\urcorner$, thus $\mathrm{just}(\kappa^+)$ appears in $\ulcorner s_1\kappa^-\urcorner$ hence $s_1\kappa^-\kappa^+$ is a (negative) pre-path. As s is a path and as $s \in \kappa_\bullet(s_1\kappa^-\kappa^+ \sqcup\!\sqcup s_2)$, by Lemma 3.3.5 $s_1\kappa^-\kappa^+$ is a path. We already know from the previous item that $s_1\kappa^-\maltese \in V^x_{\mathbf{M}}$. Notice that $\kappa_\bullet\langle s_1\kappa^-\kappa^+\rangle = \langle\kappa^+\rangle_s = \langle\kappa^+\rangle_t = \kappa_\bullet\langle\kappa^+\rangle_{t_1}$. Hence $\langle s_1\kappa^-\kappa^+\rangle = \langle\kappa^+\rangle_{t_1}$ is a bi-view of $\mathbf{M}^x$. Let $\mathbb{u}\kappa^+ = \langle s_1\kappa^-\kappa^+\rangle$. By Lemma 3.1.8, $s_1\kappa^-\maltese$ is a shuffle of anti-shuffles of bi-views of $\mathbf{M}^x$, one of which is the bi-view $\mathbb{u}\maltese$. Remark that $s_1\kappa^-\kappa^+$ is also a shuffle of anti-shuffles of bi-views of $\mathbf{M}^x$, replacing $\mathbb{u}\maltese$ by $\mathbb{u}\kappa^+$. By Proposition 3.1.10, $s_1\kappa^-\kappa^+ \in V^x_{\mathbf{M}}$. Finally, as $s \in \kappa_\bullet(s_1\kappa^-\kappa^+ \sqcup\!\sqcup s_2)$ and by Proposition 3.2.8, we have $s \in V_{\mathbf{M}\otimes\mathbf{N}}$.
- Or κ^+ is a proper action of u_1, hence of u. The reasoning is similar to the previous item, using u and u_1 instead of t and t_1 respectively.
- Or κ^+ is a proper action of t_2, hence of t. This is impossible, being given the structure of s: the action κ_0^+ following the negative action κ^- in t is necessarily in t_1 (due to the structure of a shuffle), hence the action following κ^- in s is necessarily either κ_0^+ (hence in t_1) or in u.
- Or κ^+ is a proper action of u_2, hence of u: this case also leads to a contradiction. We know from the previous item that a positive action of t_2 cannot immediately follow a negative action of t_1 in s. Similarly, a positive action of u_2 (resp. t_1, u_1)

cannot immediately follow a negative action of u_1 (resp. t_2, u_2) in s. Suppose that there exists a positive action κ_0^+ of u_2 (or resp. t_2, u_1, t_1) which follows immediately a negative action κ_0^- of t_1 (or resp. u_1, t_2, u_2). Let $s_0\kappa_0^-\kappa_0^+$ be the shortest prefix of s satisfying such a property, say κ_0^+ is an action of u_2 and κ_0^- is an action of t_1. Then the view $\ulcorner s_0\kappa_0^- \urcorner$ is necessarily only made of $\kappa_\bullet$ and of actions from t_1 or u_1, thus it does not contain $\mathrm{just}(\kappa_0^+)$ (where κ_0^+ cannot be initial because $\mathbf{N}$ is negative), i.e., s does not satisfy P-visibility: contradiction.

□

Corollary 3.3.6

If $\mathbf{N}$ *and* $\mathbf{P}$ *are regular, then* $\mathbf{N} \multimap \mathbf{P}$ *is regular.*

3.4 Purity and Connectives

We end this chapter by studying purity when applying connectives.

Proposition 3.4.1

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

Proof (Proposition 3.4.1). We must prove that:

- if $\mathbf{N}$ is pure then $\downarrow\mathbf{N}$ is pure,
- if $\mathbf{P}$ is pure then $\uparrow\mathbf{P}$ is pure,
- if $\mathbf{M}$ and $\mathbf{N}$ are pure then $\mathbf{M} \oplus \mathbf{N}$ is pure,
- if $\mathbf{M}$ and $\mathbf{N}$ are pure then $\mathbf{M} \otimes \mathbf{N}$ is pure.

For the shifts and the plus, the result is immediate given the form of visitable paths of $\downarrow\mathbf{N}$, $\uparrow\mathbf{P}$ and $\mathbf{M} \oplus \mathbf{N}$ (Propositions 3.2.1 and 3.2.4). Let us prove the result for the tensor.

Let $s = s'\maltese \in V_{\mathbf{M}\otimes\mathbf{N}}$. According to Proposition 3.2.6, either $s = \maltese$ or there exist $s_1 \in V_{\mathbf{M}}^x$ and $s_2 \in V_{\mathbf{N}}^y$ such that $s \in \kappa_\bullet(s_1 \sqcup s_2)$. If $s = \maltese$ then it is extensible with $\kappa_\bullet$, so suppose $s \in \kappa_\bullet(s_1 \sqcup s_2)$. Without loss of generality, suppose $s_1 = s_1'\maltese$. Since $\mathbf{M}$ is pure, s_1 is extensible: there exists a proper positive action κ^+ such that $s_1'\kappa^+ \in V_{\mathbf{M}}^x$. Then, note that $s'\kappa^+$ is a path: indeed, since $s_1'\kappa^+$ is a path, the justification of κ^+ appears in $\ulcorner s_1' \urcorner = \ulcorner s' \urcorner$. Moreover $s'\kappa^+ \in \kappa_\bullet(V_{\mathbf{M}}^x \sqcup V_{\mathbf{N}}^y)$, let us show that $s'\kappa^+ \in V_{\mathbf{M}\otimes\mathbf{N}}$. Let $t \in V_{\mathbf{M}}^x \sqcup V_{\mathbf{N}}^y$ and κ^- a negative action such that $\overline{\kappa_\bullet t\kappa^-}$ is a path of $\overbrace{s'\kappa^+}^{\Pi^c}$, and by Proposition 3.2.6 it suffices to show that $t\kappa^-\maltese \in V_{\mathbf{M}}^x \sqcup V_{\mathbf{N}}^y$. But

$$\overbrace{s'\kappa^+}^{\Pi^c} = \overbrace{\overline{s'\kappa^+}\maltese}^{\Pi^c} = \overbrace{\overline{s'}}^{\Pi^c} = \overbrace{\tilde{s}}^{\Pi^c},$$

therefore $\overline{\kappa_\bullet t\kappa^-}$ is a path of $\overbrace{\tilde{s}}^{\Pi^c}$. Since $s \in V_{\mathbf{M}\otimes\mathbf{N}}$, by Proposition 3.2.6 we get $t\kappa^-\maltese \in V_{\mathbf{M}}^x \sqcup V_{\mathbf{N}}^y$. Finally $s'\kappa^+ \in V_{\mathbf{M}\otimes\mathbf{N}}$, hence s is extensible. □

Unfortunately, when $\mathbf{N}$ and $\mathbf{P}$ are pure, $\mathbf{N} \multimap \mathbf{P}$ is not necessarily pure. However, we prove that a weaker form of purity, called *quasi-purity*, holds for $\mathbf{N} \multimap \mathbf{P}$ under regularity

assumption (Proposition 3.4.3). In Chapter 5, with a notion of functional behaviour, we will identify precisely the case when purity is not preserved.

Definition 3.4.2 (*Well-bracketed path, quasi-purity*)

- We say 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'.
- A behaviour $\mathbf{B}$ is **quasi-pure** if all the $\maltese$-ended well-bracketed paths in $V_{\mathbf{B}}$ are extensible, in other words if there is no maximal $\maltese$-ended well-bracketed path.

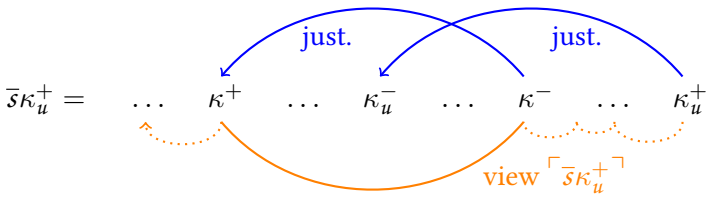
Note that a pure behaviour is indeed quasi-pure.

Proposition 3.4.3

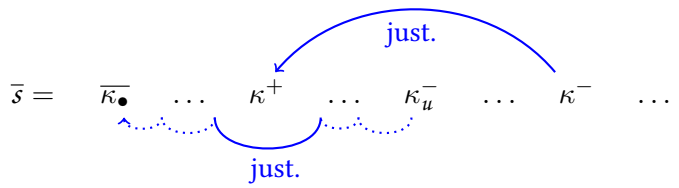
If $\mathbf{N}$ *and* $\mathbf{P}$ *are quasi-pure and regular then* $\mathbf{N} \multimap \mathbf{P}$ *is quasi-pure.*

Proof. Since $\mathbf{N}$ and $\mathbf{P}$ are regular, $V_{(\mathbf{N}\multimap\mathbf{P})^\perp} = \kappa_\bullet(V^x_{\mathbf{N}} \sqcup \widetilde{V^y_{\mathbf{P}}}) \cup \{\maltese\}$ by Corollary 3.2.9. Let $s \in V_{(\mathbf{N}\multimap\mathbf{P})^\perp}$ and suppose $\widetilde{s}$ is $\maltese$-ended, i.e., s is $\maltese$-free. We must show that either $\widetilde{s}$ is extensible or $\widetilde{s}$ is not well-bracketed. The path s is of the form $s = \kappa_\bullet s'$ and there exist $\maltese$-free paths $t \in V^x_{\mathbf{N}}$ and $u \in \widetilde{V^y_{\mathbf{P}}}$ such that $s' \in t \sqcup u$. We are in one of the following situations:

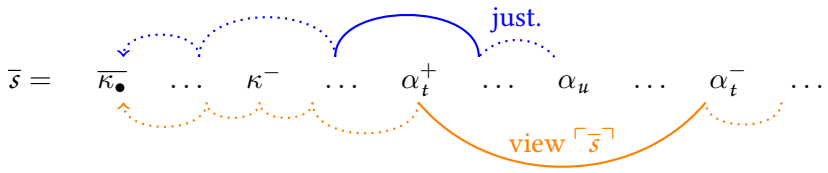
- Either $\widetilde{u} \in V^y_{\mathbf{P}}$ is not well-bracketed, hence neither is $\widetilde{s}$.
- Otherwise, since $\mathbf{P}$ is quasi-pure, $\widetilde{u} = \overline{u}\maltese$ is extensible, i.e., there exists a proper positive action κ^+_u such that $\overline{u}\kappa^+_u \in V^y_{\mathbf{P}}$. If $\overline{s}\kappa^+_u$ is a path, then $\overline{s}\kappa^+_u \in V_{\mathbf{N}\multimap\mathbf{P}}$, hence $\widetilde{s}$ is extensible: indeed, $\widetilde{\overline{s}\kappa^+_u} = s\overline{\kappa^+_u}\maltese \in \kappa_\bullet(t \sqcup u\overline{\kappa^+_u}\maltese)$, thus $s\overline{\kappa^+_u}\maltese \in \kappa_\bullet(V^x_{\mathbf{N}} \sqcup \widetilde{V^y_{\mathbf{P}}})$. In the case $\overline{s}\kappa^+_u$ is not a path, this means that κ^+_u is justified by an action κ^-_u that does not appear in $\ulcorner\overline{s}\urcorner$, thus we have something of the form:



If κ^- comes from $\overline{t}$, and thus also κ^+, then $\overline{s}$ is not well-bracketed, indeed: since κ^-_u is hereditarily justified by $\overline{\kappa_\bullet}$ and by no action from $\overline{t}$, we have:



So suppose now that κ^- comes from $\overline{u}$, thus also κ^+. We know that $\ulcorner \overline{u} \urcorner$ contains $\kappa_u^- = \mathrm{just}(\kappa_u^+)$, thus in particular $\ulcorner \overline{u} \urcorner$ does not contain κ^-; on the contrary, we have seen that $\ulcorner \overline{s} \urcorner$ contains κ^-. By definition of the view of a sequence, this necessarily means that, in $\overline{s}$, between the action κ^- and the end of the sequence, the following happens: $\ulcorner \overline{s} \urcorner$ comes across an action α_t^- from $\overline{t}$, justified by an action α_t^+ also from $\overline{t}$, making the view miss at least one action α_u from $\overline{u}$ appearing in $\ulcorner \overline{u} \urcorner$, as depicted below.



Since α_u is hereditarily justified by $\overline{\kappa_\bullet}$ and by no action from $\overline{t}$, the path $\overline{s}$ is not well-bracketed: the justifications of α_u and of α_t^- intersect.

To sum up, we have proved that in the case when $\widetilde{u} = \overline{u}✠$ is extensible, either $\widetilde{s}$ is extensible too or it is not well-bracketed.

Hence $\mathbf{N} \multimap \mathbf{P}$ is quasi-pure. □

4 | Inductive Types

Though ludics draws inspiration from (linear) logic, it can also be considered as a functional programming language. This is the point of view we adopt in Chapters 4 and 5 by studying respectively data types and functions types in ludics. As already mentioned, there have been previous works on data and functions in ludics [Ter11, Sir15]. Our approach aims at being as generic as possible by studying the behaviours corresponding to all the usual data types (of finite data) and to functional types that might be higher-order.

In the previous chapters, we have laid the foundations for representing and studying data types in ludics. Indeed, the logical connectives play the role of type constructors, and we have proved some results about their visitable paths, in particular regularity and purity. However, this is yet very limited, since with these connectives we cannot have infinite data types. We need least fixed points to be able to consider inductive data types, such as the following (here 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 ;;
```

This chapter brings important contributions in the study of inductive types and least fixed points in ludics, building on the work of Baelde, Doumane and Saurin [BDS15]. In Section 4.1, we give a grammar for *data patterns* corresponding to data types, where $\oplus$ and $\otimes$ are for sum types and product types respectively, and the symbol μ is an operator for inductive types. Data patterns are interpreted as behaviours; in particular μ is interpreted as a least fixed point, which has an explicit form thanks to Kleene's fixed point theorem. Going further, we prove in Section 4.2 a new internal completeness theorem for infinite unions satisfying some conditions (Theorem 4.2.2), giving a direct way of constructing the least fixed points. Then we conduct in Section 4.3 a structural study of data behaviours: internal completeness gives us information on the visitable paths, thus we can prove that such behaviours are regular and pure. Finally, in Section 4.4, we give some ideas for the future study of coinductive types in ludics.

From a programming perspective, designs play a double role:

- the role of programs (player's strategy),
- the role of the environment (opponent's strategy).

Indeed, the (closed) interaction between two designs corresponds to running a certain program in a certain environment. Typically, if a design in the behaviour $\mathbf{A} \multimap \mathbf{B}$ interacts

with a design in the orthogonal behaviour $(\mathbf{A} \multimap \mathbf{B})^\perp = \mathbf{A} \otimes \mathbf{B}^\perp$, this corresponds to the evaluation of a program of type $A \to B$ in an environment that provides an argument of type A and consumes the result of type B returned by the program. In this context, a daimon corresponds to the interruption of the computation. If ✠ comes from the environment, it corresponds either to an incomplete argument of type A or to the natural end of the computation after the environment received and consumed the result of type B. If ✠ comes from the side of the program, then it corresponds to a bug of this program, which is not able to provide what is required. It is thus natural to expect that the daimon comes from the environment rather than from the program, and the purpose of *purity* is precisely to ensure this is the case for all the programs of a given type. Hence our interest for this property in a computational context, and we show in this chapter that it holds for the ludics interpretation of inductive data types.

Regularity, on the other hand, means *unoriented* in the sense that every possible interaction trace of a type is realised. This is indeed a desirable property for data types, since for example we would like to have the choice between visiting first a or b when we are given a pair (a, b). But this property also has two major benefits for our study:

- it guarantees that we can apply the internal completeness theorem for fixed points,
- it might lead to characterise μMALL in ludics (discussed in § 4.3.c).

This is why we need to ensure it holds for data behaviours.

Notation

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

4.1 Inductive Data Types as Kleene Fixed Points

The point of this section is to interpret data types as ludics behaviours, called the *data behaviours*, and to give an explicit description of the behaviours corresponding to inductive types thanks to Kleene's fixed point theorem.

4.1.a Data Patterns

We define the data patterns via a type language and we interpret them as behaviours. Data patterns correspond to some (positive) formulas of polarised MALL extended with a constructor μ for inductive types; similarly to [BDS15], we will interpret the logical connectives of the logic by the corresponding constructors of ludics, and μ as a least fixed point behaviour.

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

Definition 4.1.1 (*Data pattern*)

The set $\mathcal{P}$ of **data patterns** is generated by the inductive grammar below.

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

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

Example 4.1.2

Let $zero, nil, empty \in \mathcal{S}'$ and $X \in \mathcal{V}$. The data types given as example in the introduction of this chapter can be written in the language of data patterns as follows:

$$\begin{aligned}
\mathbb{N}\mathrm{at} &= \mu X.(zero \oplus^+ X) \ , \\
\mathbb{L}\mathrm{ist}_A &= \mu X.(nil \oplus^+ (A \otimes^+ X)) \ , \\
\mathbb{T}\mathrm{ree}_A &= \mu X.(empty \oplus^+ (A \otimes^+ \mathbb{L}\mathrm{ist}_X)) \\
&= \mu X.(empty \oplus^+ (A \otimes^+ \mu Y.(nil \oplus^+ (X \otimes^+ Y)))) \ .
\end{aligned}$$

Notation

In the following, we will denote by $\mathcal{B}^+$ the set of positive behaviours.

An **environment** σ is a function that maps the free variables of a data pattern to positive behaviours. The notation $\sigma, X \mapsto \mathbf{P}$ stands for the environment σ where the image of X has been set (or changed) to behaviour $\mathbf{P}$. We call an environment **regular** (resp. **pure**) if its image contains only regular (resp. pure) behaviours.

Given a data pattern $A \in \mathcal{P}$ and an environment σ, the **interpretation** of A in the environment σ, written $[\![A]\!]^\sigma$, is the positive behaviour defined by:

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

where $\mathrm{lfp}(\phi^A_\sigma)$ stands for the least fixed point of the function defined by

$$\begin{aligned}
\phi^A_\sigma : \mathcal{B}^+ &\to \mathcal{B}^+ \\
\mathbf{P} &\mapsto [\![A]\!]^{\sigma, X \mapsto \mathbf{P}} \ .
\end{aligned}$$

The fact that this function has a least fixed point is ensured by the Knaster-Tarski fixed point theorem, as pointed out by Baelde, Doumane and Saurin [BDS15].

Notation

Abusively we will write $\oplus^+$ and $\otimes^+$, instead of $(\uparrow\cdot) \oplus (\uparrow\cdot)$ and $(\uparrow\cdot) \otimes (\uparrow\cdot)$ respectively, for behaviours.

4.1.b Kleene Fixed Point Theorem

Although we know that all data patterns have a behaviour counterpart, the Knaster–Tarski fixed point theorem is non-constructive, it gives no concrete information on fixed points. Thus we do not know what all the designs in the interpretation of $\mu X.A$ are. For example, is there really nothing else in $[\![\mathbb{N}\mathrm{at}]\!]$ than the designs corresponding to natural numbers? A more constructive approach is needed so as to understand the structure of fixed points behaviours, and we start in this subsection by proving that we can apply Kleene's fixed point theorem instead of Knaster–Tarski's. In the next section, we will moreover prove an internal completeness result for infinite union, which will give a direct construction for least fixed points behaviours.

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 $\bot$. 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 $\bigvee_{x\in D} f(x) = f(\bigvee_{x\in D} x)$.

Remark 4.1.3

A continuous function $f : E \to F$ is **monotone**, i.e., if $x \leq_E y$ then $f(x) \leq_F f(y)$.

Theorem 4.1.4 (*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

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

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

Proposition 4.1.5

Given a data pattern $A \in \mathcal{P}$, a variable $X \in \mathcal{V}$, an environment $\sigma : \mathrm{FV}(A)\backslash\{X\} \to \mathcal{B}^+$, the function ϕ^A_σ is Scott-continuous.

To prove this proposition, we first need the following lemma.

Lemma 4.1.6

Let E, F be sets of cut-free atomic negative designs and G be a set of cut-free atomic positive designs.

1. $\downarrow(E^{\bot\bot}) = \blacktriangledown\langle E\rangle^{\bot\bot}$.
2. $\uparrow(G^{\bot\bot}) = \{\mathfrak{n} \mid \mathfrak{n}\restriction\blacktriangle \in \blacktriangle(x).G^x\}^{\bot\bot}$.
3. $(E^{\bot\bot}) \oplus (F^{\bot\bot}) = (\iota_1\langle E\rangle \cup \iota_2\langle F\rangle)^{\bot\bot}$.
4. $(E^{\bot\bot}) \otimes (F^{\bot\bot}) = \bullet\langle E, F\rangle^{\bot\bot}$.

Proof. We prove (1) and (2), the other cases are very similar to (1).

1. $\blacktriangledown\langle E\rangle^{\perp\perp} = \{\mathfrak{n} \mid \mathfrak{n}\restriction\blacktriangle \in \blacktriangle(x).(E^\perp)^x\}^\perp = (\uparrow(E^\perp))^\perp = (\downarrow(E^{\perp\perp}))^{\perp\perp} = \downarrow(E^{\perp\perp})$,
2. $\{\mathfrak{n} \mid \mathfrak{n}\restriction\blacktriangle \in \blacktriangle(x).G^x\}^{\perp\perp} = \{\blacktriangledown\langle\mathfrak{m}\rangle \mid \mathfrak{m} \in G^\perp\}^\perp = (\downarrow(G^\perp))^\perp = \uparrow(G^{\perp\perp})$,

using the definition of the orthogonal, internal completeness, and Lemma 3.2.3. □

We can now prove Proposition 4.1.5. Note that, while this proposition justifies that it is possible to apply Kleene's fixed point theorem, its proof uses this theorem itself! But no contradiction here: the proof is done by induction on data patterns, and we apply Kleene's theorem solely on patterns of the previous induction step. This procedure, for proving a result by induction, of using what seems like a corollary of this result (here corresponding to Corollary 4.1.7) is typical in this chapter, because of fixed points. It will sometimes lead to very long induction hypotheses, where many things have to be proved at the same time, for example in the proof of Propositions 4.3.1 and 4.3.2.

Proof (Proposition 4.1.5). By induction on A, we prove that for every X and every σ the function ϕ^A_σ is continuous. Note that ϕ^A_σ is continuous if and only if for every directed subset $\mathbb{P} \subseteq \mathcal{B}^+$ we have $\bigvee_{\mathbf{P}\in\mathbb{P}}(\llbracket A\rrbracket^{\sigma,X\mapsto\mathbf{P}}) = \llbracket A\rrbracket^{\sigma,X\mapsto\bigvee\mathbb{P}}$. The cases $A = Y \in \mathcal{V}$ and $A = a \in \mathcal{S}$ are trivial, and the case $A = A_1 \oplus^+ A_2$ is very similar to the tensor, hence we only treat the two remaining cases. Let $\mathbb{P} \subseteq \mathcal{B}^+$ be directed.

- Suppose $A = A_1 \otimes^+ A_2$, thus $\llbracket A\rrbracket^{\sigma,X\mapsto\mathbf{P}} = \llbracket A_1\rrbracket^{\sigma,X\mapsto\mathbf{P}} \otimes^+ \llbracket A_2\rrbracket^{\sigma,X\mapsto\mathbf{P}}$, with both functions $\phi^{A_i}_\sigma : \mathbf{P} \mapsto \llbracket A_i\rrbracket^{\sigma,X\mapsto\mathbf{P}}$ continuous by induction hypothesis. For any positive behaviour $\mathbf{P}$, let us write $\sigma_\mathbf{P}$ instead of $\sigma, X \mapsto \mathbf{P}$. We have

$$\bigvee_{\mathbf{P}\in\mathbb{P}} \llbracket A\rrbracket^{\sigma_\mathbf{P}} = (\bigcup_{\mathbf{P}\in\mathbb{P}} \llbracket A\rrbracket^{\sigma_\mathbf{P}})^{\perp\perp} = (\bigcup_{\mathbf{P}\in\mathbb{P}} (\llbracket A_1\rrbracket^{\sigma_\mathbf{P}} \otimes^+ \llbracket A_2\rrbracket^{\sigma_\mathbf{P}}))^{\perp\perp} \ .$$

  Let us show that

$$\bigcup_{\mathbf{P}\in\mathbb{P}} (\llbracket A_1\rrbracket^{\sigma_\mathbf{P}} \otimes^+ \llbracket A_2\rrbracket^{\sigma_\mathbf{P}}) = \bullet\langle \bigcup_{\mathbf{P}'\in\mathbb{P}} \uparrow\llbracket A_1\rrbracket^{\sigma_{\mathbf{P}'}}, \bigcup_{\mathbf{P}''\in\mathbb{P}} \uparrow\llbracket A_2\rrbracket^{\sigma_{\mathbf{P}''}}\rangle \cup \{\maltese\} \ . \quad (*)$$

  By internal completeness, for every $\mathbf{P} \in \mathbb{P}$ we have

$$\llbracket A_1\rrbracket^{\sigma_\mathbf{P}} \otimes^+ \llbracket A_2\rrbracket^{\sigma_\mathbf{P}} = \bullet\langle\uparrow\llbracket A_1\rrbracket^{\sigma_\mathbf{P}}, \uparrow\llbracket A_2\rrbracket^{\sigma_\mathbf{P}}\rangle \cup \{\maltese\} \ .$$

  The inclusion $(\subseteq)$ of $(*)$ is then immediate, so let us prove $(\supseteq)$. First, indeed, $\maltese$ belongs to the left side. Let $\mathbf{P}', \mathbf{P}'' \in \mathbb{P}$, let $\mathfrak{m} \in \uparrow\llbracket A_1\rrbracket^{\sigma_{\mathbf{P}'}}$, $\mathfrak{n} \in \uparrow\llbracket A_2\rrbracket^{\sigma_{\mathbf{P}''}}$, and let us show that $\bullet\langle\mathfrak{m},\mathfrak{n}\rangle \in \llbracket A_1\rrbracket^{\sigma_\mathbf{P}} \otimes^+ \llbracket A_2\rrbracket^{\sigma_\mathbf{P}}$ where $\mathbf{P} = \mathbf{P}' \vee \mathbf{P}''$ (note that $\mathbf{P} \in \mathbb{P}$ since $\mathbb{P}$ is directed). By induction hypothesis, $\phi^{A_1}_\sigma$ is continuous, thus in particular monotone; since $\mathbf{P}' \subseteq \mathbf{P}$, it follows that

$$\llbracket A_1\rrbracket^{\sigma_{\mathbf{P}'}} = \phi^{A_1}_\sigma(\mathbf{P}') \subseteq \phi^{A_1}_\sigma(\mathbf{P}) = \llbracket A_1\rrbracket^{\sigma_\mathbf{P}} \ .$$

  Similarly we have

$$\llbracket A_2\rrbracket^{\sigma_{\mathbf{P}''}} \subseteq \llbracket A_2\rrbracket^{\sigma_\mathbf{P}} \ .$$

  Using internal completeness for the negative shift, we get

$$\bullet\langle\mathfrak{m},\mathfrak{n}\rangle \in \bullet\langle\uparrow\llbracket A_1\rrbracket^{\sigma_\mathbf{P}}, \uparrow\llbracket A_2\rrbracket^{\sigma_\mathbf{P}}\rangle \subseteq \llbracket A_1\rrbracket^{\sigma_\mathbf{P}} \otimes^+ \llbracket A_2\rrbracket^{\sigma_\mathbf{P}}$$

which proves $(*)$. By internal completeness, Lemma 4.1.6 and induction hypothesis, we deduce:

$$\begin{aligned}
(\bigcup_{\mathbf{P}\in\mathbb{P}}([\![A_1]\!]^{\sigma_{\mathbf{P}}}\otimes^+[\![A_2]\!]^{\sigma_{\mathbf{P}}}))^{\perp\perp} &= \bullet\langle \bigcup_{\mathbf{P}'\in\mathbb{P}}\uparrow[\![A_1]\!]^{\sigma_{\mathbf{P}'}}, \bigcup_{\mathbf{P}''\in\mathbb{P}}\uparrow[\![A_2]\!]^{\sigma_{\mathbf{P}''}}\rangle^{\perp\perp}\\
&= (\bigcup_{\mathbf{P}'\in\mathbb{P}}\uparrow[\![A_1]\!]^{\sigma_{\mathbf{P}'}})^{\perp\perp}\otimes(\bigcup_{\mathbf{P}''\in\mathbb{P}}\uparrow[\![A_2]\!]^{\sigma_{\mathbf{P}''}})^{\perp\perp}\\
&= (\bigcup_{\mathbf{P}'\in\mathbb{P}}[\![A_1]\!]^{\sigma_{\mathbf{P}'}})^{\perp\perp}\otimes^+(\bigcup_{\mathbf{P}''\in\mathbb{P}}[\![A_2]\!]^{\sigma_{\mathbf{P}''}})^{\perp\perp}\\
&= [\![A_1]\!]^{\sigma,X\mapsto\bigvee\mathbb{P}}\otimes^+[\![A_2]\!]^{\sigma,X\mapsto\bigvee\mathbb{P}}\\
&= [\![A]\!]^{\sigma,X\mapsto\bigvee\mathbb{P}} \ .
\end{aligned}$$

Consequently ϕ_σ^A is continuous.

- If $A=\mu Y.A_0$, define
$$f_0:\mathbf{Q}\mapsto[\![A_0]\!]^{\sigma,X\mapsto\bigvee\mathbb{P},Y\mapsto\mathbf{Q}}$$
and, for every $\mathbf{P}\in\mathcal{B}^+$,
$$f_{\mathbf{P}}:\mathbf{Q}\mapsto[\![A_0]\!]^{\sigma,X\mapsto\mathbf{P},Y\mapsto\mathbf{Q}} \ .$$
Those functions are continuous by induction hypothesis, thus using Kleene's fixed point theorem we have
$$\mathrm{lfp}(f_0)=\bigvee_{n\in\mathbb{N}}{f_0}^n(\maltese) \quad \text{and} \quad \mathrm{lfp}(f_{\mathbf{P}})=\bigvee_{n\in\mathbb{N}}{f_{\mathbf{P}}}^n(\maltese) \ .$$
Therefore
$$\bigvee_{\mathbf{P}\in\mathbb{P}}([\![A]\!]^{\sigma,X\mapsto\mathbf{P}})=\bigvee_{\mathbf{P}\in\mathbb{P}}(\mathrm{lfp}(f_{\mathbf{P}}))=\bigvee_{\mathbf{P}\in\mathbb{P}}(\bigvee_{n\in\mathbb{N}}{f_{\mathbf{P}}}^n(\maltese))=\bigvee_{n\in\mathbb{N}}(\bigvee_{\mathbf{P}\in\mathbb{P}}{f_{\mathbf{P}}}^n(\maltese)) \ .$$
For every $\mathbf{Q}\in\mathcal{B}^+$ the function
$$g_{\mathbf{Q}}:\mathbf{P}\mapsto f_{\mathbf{P}}(\mathbf{Q})$$
is continuous by induction hypothesis, hence
$$f_0(\mathbf{Q})=\bigvee_{\mathbf{P}\in\mathbb{P}}f_{\mathbf{P}}(\mathbf{Q}) \ .$$
From this, we prove easily by induction on m that for every $\mathbf{Q}\in\mathcal{B}^+$ we have
$${f_0}^m(\mathbf{Q})=\bigvee_{\mathbf{P}\in\mathbb{P}}{f_{\mathbf{P}}}^m(\mathbf{Q}) \ .$$
Thus
$$\bigvee_{\mathbf{P}\in\mathbb{P}}([\![A]\!]^{\sigma,X\mapsto\mathbf{P}})=\bigvee_{n\in\mathbb{N}}{f_0}^n(\maltese)=\mathrm{lfp}(f_0)=[\![A]\!]^{\sigma,X\mapsto\bigvee\mathbb{P}} \ .$$
We conclude that the function ϕ_σ^A is continuous.

□

Corollary 4.1.7

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

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

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 Section 4.2 will be to provide an internal completeness result proving that the closure is actually not necessary.

4.1.c Steady Data Patterns and Data Behaviours

Let us finish the section by defining a restricted set of data patterns so as to exclude the degenerate ones. Consider for example $\mathbb{L}\mathrm{ist}_A{}' = \mu X.(A \otimes^+ X)$, a variant of $\mathbb{L}\mathrm{ist}_A$ (see Example 4.1.2) 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 $\mathbb{N}\mathrm{at}' = \mu X.X$ is even worse since $[\![\mathbb{N}\mathrm{at}']\!] = \maltese$. The point of steady data patterns is to ensure the existence of a basis; this statement will be formalised in Lemma 4.3.4.

Definition 4.1.8 (*Steady data pattern*)

The set $\mathcal{P}^s$ of **steady data patterns** is generated by the inductive grammar:

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

where K is a *data pattern* such that $[\![K]\!]^\sigma$ is pure if σ is pure.

In the cases of $\oplus^+$, the condition on K ensures the preservation of purity, i.e., type safety; note that K is not necessarily steady, in particular variables can be introduced on this side, while the basis will come from the side of A. We will prove (in Section 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, hence the following definition.

Definition 4.1.9 (*Data behaviour*)

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

4.2 Internal Completeness for Infinite Union

An important contribution of this thesis is the following internal completeness theorem, stating that an infinite union of *simple* regular behaviours with increasingly large incarnations is a behaviour: the $\bot\bot$-closure is useless.

Definition 4.2.1 (*Slice, simple behaviour*)

- 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 $\mathfrak{d}$.

Theorem 4.2.2

Let $(\mathbf{A}_n)_{n\in\mathbb{N}}$ be 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 $\mathbf{A}_n \subseteq \mathbf{A}_{n+1}$). The set $\bigcup_{n\in\mathbb{N}} \mathbf{A}_n$ is a behaviour.

In this theorem's statement, we demand that the behaviours' incarnations – and not simply the behaviours themselves – are increasing; this is because there are many examples of behaviours $\mathbf{A}$ and $\mathbf{B}$ with $\mathbf{A} \subsetneq \mathbf{B}$ but $|\mathbf{A}| \supsetneq |\mathbf{B}|$. In fact, adding carelessly designs to a behaviour has more chances to globally restrict the possible interactions within the behaviour, that is, to reduce its incarnation. Since incarnation is the part of the behaviour that truly interacts, it is the part that we want increasing; moreover, $|\mathbf{A}| \subseteq |\mathbf{B}|$ always implies $\mathbf{A} \subseteq \mathbf{B}$.

A union of behaviours is not a behaviour in general; Theorem 4.2.2 gives sufficient conditions so that it is. Counterexamples are easily found if releasing either the inclusion of the incarnations or the simplicity condition. It is probably possible to refine the definition of simple with a less strict second condition, so that the theorem still holds. For example, Faggian gives in her thesis [Fag01] a characterisation of designs that can be entirely visited by a single interaction, which might provide us with a better condition, but this is yet to be explored.

Note moreover that the proof of this theorem relies strongly on regularity. We do believe it is possible to find a counterexample of the theorem if we remove the regularity hypothesis, but we have not got one by the time this thesis is completed.

Before proving the theorem, we need several lemmas. Suppose $(\mathbf{A}_n)_{n\in\mathbb{N}}$ is an infinite sequence of regular behaviours, with $|\mathbf{A}_n| \subseteq |\mathbf{A}_{n+1}|$ for all $n \in \mathbb{N}$; the simplicity hypothesis is not needed for now. Notice that the definition of visitable paths can harmlessly be extended to any set E of cut-free atomic 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 such that $\mathfrak{e}_1 \sqsubseteq \mathfrak{d}$ and $\mathfrak{e}_2 \sqsubseteq \mathfrak{d}$ then there exists $\mathfrak{e} \in E$ such that $\mathfrak{e} \sqsubseteq \mathfrak{e}_1$ and $\mathfrak{e} \sqsubseteq \mathfrak{e}_2$. In particular, as a directed union of behaviours, $\bigcup_{n\in\mathbb{N}} \mathbf{A}_n$ satisfies this condition.

Notation

In the proofs of this section, we will denote by $\mathbf{A}$ the set $\bigcup_{n\in\mathbb{N}} \mathbf{A}_n$.

Lemma 4.2.3

Let $(\mathbf{A}_n)_{n\in\mathbb{N}}$ be an infinite sequence of regular behaviours such that for all $n \in \mathbb{N}$, $|\mathbf{A}_n| \subseteq |\mathbf{A}_{n+1}|$. We have:

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 exists $\mathfrak{d} \in |\mathbf{A}_n|$ 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 the previous item, thus $s \in V_{\mathbf{A}_n}$. Hence if we take $\mathfrak{e} = \tilde{s}^{\Pi^C}$, we have $\mathfrak{e} \in {\mathbf{A}_n}^{\perp}$ for all $n \geq m$ by Lemma 3.1.3. 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 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 4.2.4

Let $(\mathbf{A}_n)_{n\in\mathbb{N}}$ be an infinite sequence of regular behaviours such that for all $n \in \mathbb{N}$, $|\mathbf{A}_n| \subseteq |\mathbf{A}_{n+1}|$. We have:

$$V_{\bigcup_{n\in\mathbb{N}} \mathbf{A}_n} = \overbrace{V_{(\bigcup_{n\in\mathbb{N}} \mathbf{A}_n)^{\perp}}} = V_{(\bigcup_{n\in\mathbb{N}} \mathbf{A}_n)^{\perp\perp}} \ .$$

Proof. In this proof we use the alternative definition of regularity (Proposition 3.1.10). 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 $s \in V_{\mathbf{A}^\perp}$ and let us show that $\widetilde{s} \in V_{\mathbf{A}}$. Let $\mathfrak{e} \in |\mathbf{A}^\perp|$ such that s is a path of $\mathfrak{e}$. By Lemma 3.1.8 and the remark following it, s is in the shuffle of anti-shuffles of bi-views $\mathbb{t}_1, \dots, \mathbb{t}_k$ of $\mathbf{A}^\perp$. For every $i \leq k$, suppose $\mathbb{t}_i = \langle \kappa_i \rangle$; 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 $\mathbb{t}_i$ 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, $\widetilde{\mathbb{t}_i}$ is a bi-view of $|\mathfrak{d}_i|_{\mathbf{A}_n}$, thus it is a bi-view of $\mathbf{A}_n$. By regularity of $\mathbf{A}_n$ we have $\widetilde{\mathbb{t}_i} \in V_{\mathbf{A}_n}$. Since $\widetilde{s}$ is in the anti-shuffle of shuffles of $\widetilde{\mathbb{t}_1}, \dots, \widetilde{\mathbb{t}_k}$, we have $\widetilde{s} \in V_{\mathbf{A}_n}$ using regularity again. Therefore $\widetilde{s} \in V_{\mathbf{A}}$ by Lemma 4.2.3(2). □

Lemma 4.2.5

Let $(\mathbf{A}_n)_{n\in\mathbb{N}}$ be an infinite sequence of regular behaviours such that for all $n \in \mathbb{N}$, $|\mathbf{A}_n| \subseteq |\mathbf{A}_{n+1}|$. The behaviours $(\bigcup_{n\in\mathbb{N}} \mathbf{A}_n)^\perp$ 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 3.1.10).

- Let $\mathbb{t}$ be a bi-view of $\mathbf{A}^\perp$. By a similar argument as in the proof above, there exists $n \in \mathbb{N}$ such that $\widetilde{\mathbb{t}}$ is a bi-view of $\mathbf{A}_n$, thus $\widetilde{\mathbb{t}} \in V_{\mathbf{A}_n} \subseteq V_{\mathbf{A}}$. By Lemma 4.2.4, $\mathbb{t} \in V_{\mathbf{A}^\perp}$.
- Let $s, t \in V_{\mathbf{A}^\perp}$. By Lemma 4.2.4, $\widetilde{s}, \widetilde{t} \in V_{\mathbf{A}}$. By Lemma 4.2.3(2), there exists $n \in \mathbb{N}$ such that $\widetilde{s}, \widetilde{t} \in V_{\mathbf{A}_n}$, thus by regularity of $\mathbf{A}_n$ we have

  $$\widetilde{s} \pitchfork \widetilde{t} \subseteq V_{\mathbf{A}_n} \text{ and } \widetilde{s} \sqcup \widetilde{t} \subseteq V_{\mathbf{A}_n}$$

  where $V_{\mathbf{A}_n} \subseteq V_{\mathbf{A}}$, in other words

  $$\widetilde{s \sqcup t} \subseteq V_{\mathbf{A}} \text{ and } \widetilde{s \pitchfork t} \subseteq V_{\mathbf{A}} \ .$$

  By Lemma 4.2.4 we deduce

  $$s \sqcup t \subseteq V_{\mathbf{A}^\perp} \text{ and } s \pitchfork t \subseteq V_{\mathbf{A}^\perp} \ ,$$

  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 (Remark 1.3.6). □

Let us introduce some more notions for the following 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 s of $\mathfrak{p}$ such that $\tilde{s}$ is an ∞-path of $\mathfrak{n}$ (where, when s is infinite, $\tilde{s}$ is obtained from s by simply reversing the polarities of all the actions). Given an infinite ∞-path s, the design $⌜s⌝^c$ is constructed similarly to the case when s is finite (see the proof of Lemma 3.1.1 in Chapter 3).

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 (Theorem 4.2.2).

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 4.2.4 and 4.2.5, every path s of $\mathfrak{d}$ is in $V_{\mathbf{A}^{\perp\perp}} = V_{\mathbf{A}}$, thus there exists $\mathfrak{d}' \in |\mathbf{A}|$ containing 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, using König's lemma (*every infinite tree contains either a vertex of infinite degree or an infinite branch*), we are in one of the following cases:

- 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 $a \in \mathcal{S}$ such that $\mathfrak{p}_a \neq \Omega$. In this case, let $\mathbb{v}$ be the view of $\mathfrak{d}$ such that for every action κ^- initial in $\mathfrak{n}$, $\mathbb{v}\kappa^-$ is the prefix of a view of $\mathfrak{d}$. All such sequences $\mathbb{v}\kappa^-$ being prefixes of paths of $\mathfrak{d}$, we deduce by regularity of $\mathbf{A}^{\perp\perp}$ and using Lemma 3.1.5 that $\mathbb{v}\kappa^-\maltese \in V_{\mathbf{A}^{\perp\perp}}$. Let $\mathfrak{d}' \in |\mathbf{A}|$ be such that $\mathbb{v}$ is a view of $\mathfrak{d}'$. Since $\mathfrak{d}'$ is also in $\mathbf{A}^{\perp\perp}$, we deduce by Lemma 3.1.6 that for every action κ^- initial in $\mathfrak{n}$, $\mathbb{v}\kappa^-$ 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 $\mathbb{v} = (\kappa_0^-)\kappa_1^+\kappa_1^-\kappa_2^+\kappa_1^-\kappa_3^+\kappa_3^- \dots$ 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 $a \in \mathcal{S}$ such that $\mathfrak{p}_a \neq \Omega$. Let $\mathbb{v}_i$ be the prefix of $\mathbb{v}$ ending on κ_i^+. There is no design $\mathfrak{d}' \in |\mathbf{A}|$ containing $\mathbb{v}$, indeed: in this case, for all i and all negative action κ^- such that $\mathbb{v}_i\kappa^-$ is a prefix of a view of $\mathfrak{d}$, $\mathbb{v}_i\kappa^-$ would be a prefix of a view of $\mathfrak{d}'$ by Lemma 3.1.6, thus $\mathfrak{d}'$ would have an infinite number of slices, which is impossible since the $\mathbf{A}_n$ are simple. Thus consider $\mathfrak{e} = ⌜\tilde{\mathbb{v}}⌝^c$: since all the $\mathbb{v}_i$ 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 $\widetilde{\mathbb{v}_i}$ are views, thus $\tilde{\mathbb{v}}$ 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 $\mathbb{v}$ forever, and interaction cannot fail after following a finite portion of $\mathbb{v}$ since

those finite portions $\mathfrak{v}_i$ are in $V_{\mathbf{A}}$. Hence $\mathfrak{e} \in \mathbf{A}^{\perp}$. But $\mathfrak{d} \not\perp \mathfrak{e}$, because of infinite chattering following $\mathfrak{v}$. Contradiction.

If $\mathfrak{d}$ has a finite number of maximal slices $\mathfrak{c}_1, \ldots, \mathfrak{c}_k$ then for every $i \leq k$ there exists an ∞-path s_i that visits 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 pairs 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.

□

Under the same hypotheses as Theorem 4.2.2 and by Lemma 4.2.3, we thus have that the set $\bigcup_{n \in \mathbb{N}} \mathbf{A}_n$ is a behaviour satisfying

$$V_{\bigcup_{n \in \mathbb{N}} \mathbf{A}_n} = \bigcup_{n \in \mathbb{N}} V_{\mathbf{A}_n} \qquad \text{and} \qquad |\bigcup_{n \in \mathbb{N}} \mathbf{A}_n| = \bigcup_{n \in \mathbb{N}} |\mathbf{A}_n| \ ,$$

hence the following corollary.

Corollary 4.2.6

Let $(\mathbf{A}_n)_{n \in \mathbb{N}}$ be an infinite sequence of simple regular behaviours such that for all $n \in \mathbb{N}$, $|\mathbf{A}_n| \subseteq |\mathbf{A}_{n+1}|$.

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

4.3 Regularity and Purity of Data

In this section, we show that the interpretation $\mathbf{A}$ of a data pattern 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 4.2.2. By applying this theorem, the goal is to be able to construct $\mathbf{A}$ without performing the bi-orthogonal, and to deduce its regularity and purity. More precisely, we prove that behaviours interpreting data patterns (resp. steady data patterns) are regular (resp. pure). We end this section by discussing the relation between regularity and μMALL.

4.3.a Regularity of Data

We call an environment σ **simple** if its image contains only simple behaviours. In this subsection, we prove the two following propositions, stating respectively that:

- the interpretation of a data pattern in a simple regular environment is a simple regular behaviour (Proposition 4.3.1), the interesting case being that data patterns without free variable (closed) only generate simple regular behaviours;
- the explicit form for the interpretation of a pattern of the form $\mu X.A$, that has been given in Corollary 4.1.7, need not be closed by bi-orthogonal (Proposition 4.3.2); this indeed relies on our previous internal completeness result (Theorem 4.2.2).

Proposition 4.3.1

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

Proposition 4.3.2

For all $A \in \mathcal{P}$, $X \in \mathcal{V}$, and $\sigma : \mathrm{FV}(A) \setminus \{X\} \to \mathcal{B}^+$ simple regular,

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

It seems that each of these statements is a prerequisite for the other one, indeed: from Proposition 4.3.1 we could deduce Proposition 4.3.2 using Theorem 4.2.2, and the other implication could be obtained from Corollary 4.2.6. For doing this, we would need the following "lemma", which is required as a hypothesis for both Theorem 4.2.2 and Corollary 4.2.6: for all $A \in \mathcal{P}$, $X \in \mathcal{V}$, $\sigma : \mathrm{FV}(A) \setminus \{X\} \to \mathcal{B}^+$ simple regular, and $n \in \mathbb{N}$, we have

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

Actually, these three results (the two propositions and the inclusion above) are proved simultaneously, included in the same induction hypothesis.

Proof (Propositions 4.3.1 and 4.3.2). By induction on A, we prove that for all $X \in \mathcal{V}$ and $\sigma : \mathrm{FV}(A) \setminus \{X\} \to \mathcal{B}^+$ simple and regular, the induction hypothesis consisting in the five following statements holds:

1. for all $\mathbf{P}, \mathbf{P}' \in \mathcal{B}^+$ simple regular, if $|\mathbf{P}| \subseteq |\mathbf{P}'|$ then $|\phi^A_\sigma(\mathbf{P})| \subseteq |\phi^A_\sigma(\mathbf{P}')|$;

2. for every $n \in \mathbb{N}$, $|(\phi_\sigma^A)^n(\maltese)| \subseteq |(\phi_\sigma^A)^{n+1}(\maltese)|$;
3. for every $\mathbf{P} \in \mathcal{B}^+$ simple regular, $\phi_\sigma^A(\mathbf{P})$ is simple and regular;
4. $[\![\mu X.A]\!]^\sigma = \bigcup_{n\in\mathbb{N}} (\phi_\sigma^A)^n(\maltese)$;
5. $|[\![\mu X.A]\!]^\sigma| = \bigcup_{n\in\mathbb{N}} |(\phi_\sigma^A)^n(\maltese)|$.

In this proof, given any positive behaviour $\mathbf{P}$, we write $\sigma_{\mathbf{P}}$ for $\sigma, X \mapsto \mathbf{P}$.

If $A = X \in \mathcal{V}$ or $A = a \in \mathcal{S}'$: immediate.

If $A = A_1 \oplus^+ A_2$ or $A = A_1 \otimes^+ A_2$:

1. Follows from the incarnated form of internal completeness (in Theorem 1.1.21).
2. Easy by induction on n, using the previous item.
3. Regularity of $\phi_\sigma^A(\mathbf{P})$ comes from Proposition 3.3.1, and simplicity is easy since the structure of the designs in $[\![A]\!]^{\sigma_{\mathbf{P}}}$ is given by internal completeness.
4. By Corollary 4.1.7 we have

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

and by Theorem 4.2.2 we have

$$(\bigcup_{n\in\mathbb{N}} (\phi_\sigma^A)^n(\maltese))^{\perp\perp} = \bigcup_{n\in\mathbb{N}} (\phi_\sigma^A)^n(\maltese)$$

since items (2) and (3) guarantee that the hypotheses of the theorem are satisfied.

5. By the previous item and Lemma 4.2.3(3).

If $A = \mu Y.A_0$:

1. Suppose $|\mathbf{P}| \subseteq |\mathbf{P}'|$, where $\mathbf{P}$ and $\mathbf{P}'$ are simple regular. We have

$$|\phi_\sigma^A(\mathbf{P})| = |[\![\mu Y.A_0]\!]^{\sigma_{\mathbf{P}}}| = \bigcup_{n\in\mathbb{N}} |(\phi_{\sigma_{\mathbf{P}}}^{A_0})^n(\maltese)|$$

by induction hypothesis (5), and similarly for $\mathbf{P}'$. By induction on n, we prove that

$$|(\phi_{\sigma_{\mathbf{P}}}^{A_0})^n(\maltese)| \subseteq |(\phi_{\sigma_{\mathbf{P}'}}^{A_0})^n(\maltese)| \qquad (\delta)$$

It is immediate for $n = 0$, and the inductive case is:

$$\begin{aligned}
|(\phi_{\sigma_{\mathbf{P}}}^{A_0})^{n+1}(\maltese)| &= |\phi_{\sigma_{\mathbf{P}}}^{A_0}((\phi_{\sigma_{\mathbf{P}}}^{A_0})^n(\maltese))| \\
&\subseteq |\phi_{\sigma_{\mathbf{P}}}^{A_0}((\phi_{\sigma_{\mathbf{P}'}}^{A_0})^n(\maltese))| && \text{by induction hypotheses (1), (3) and } (\delta) \\
&= |\phi^{A_0}_{\sigma, Y\mapsto(\phi_{\sigma_{\mathbf{P}'}}^{A_0})^n(\maltese)}(\mathbf{P})| \\
&\subseteq |\phi^{A_0}_{\sigma, Y\mapsto(\phi_{\sigma_{\mathbf{P}'}}^{A_0})^n(\maltese)}(\mathbf{P}')| && \text{by induction hypotheses (1) and (3)} \\
&= |(\phi_{\sigma_{\mathbf{P}'}}^{A_0})^{n+1}(\maltese)| \ .
\end{aligned}$$

3. By induction hypotheses (2), (3) and (4) respectively, we have
   - for every $n \in \mathbb{N}$, $|(\phi_\sigma^{A_0})^n(\maltese)| \subseteq |(\phi_\sigma^{A_0})^{n+1}(\maltese)|$,
   - for every $n \in \mathbb{N}$, $(\phi_\sigma^{A_0})^n(\maltese)$ is simple regular,

- $[\![\mu Y.A_0]\!]^\sigma = \bigcup_{n\in\mathbb{N}}(\phi^{A_0}_\sigma)^n(\maltese)$.

Consequently, by Corollary 4.2.6, $[\![\mu Y.A_0]\!]^\sigma$ is simple regular.

2. 4. 5. Similar to the cases $A = A_1 \oplus^+ A_2$ and $A = A_1 \otimes^+ A_2$.

To conclude this proof, remark that (3) proves Propositions 4.3.1, indeed: if $X \notin \mathrm{FV}(A)$, i.e., if $\sigma : \mathrm{FV}(A) \to \mathcal{B}^+$, then for every behaviour $\mathbf{P}$ we have $\phi^A_\sigma(\mathbf{P}) = [\![A]\!]^\sigma$. Moreover, indeed, (4) corresponds to Proposition 4.3.2. □

Corollary 4.3.3

Data behaviours are regular.

4.3.b Purity of Data

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 4.3.4).

Notation

Write $V^{max}_{\mathbf{B}}$ for the set of maximal visitable paths of $\mathbf{B}$.

Lemma 4.3.4

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

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

Proof. By induction on A:

- If $A = a$ then it has basis $[\![a]\!] = \mathbf{C}_a$.
- If $A = A_1 \oplus^+ A_2$, without loss of generality suppose A_1 is steady, with basis $\mathbf{B}_1$. Take $\otimes_1{\uparrow}\mathbf{B}_1$, as a basis for A, where the connective $\otimes_1$ is defined like $\downarrow$ with a different name of action: $\otimes_1\mathbf{N} = \iota_1\langle\mathbf{N}\rangle^{\perp\perp}$ and by internal completeness $\otimes_1\mathbf{N} = \iota_1\langle\mathbf{N}\rangle \cup \{\maltese\}$.
- If $A = A_1 \otimes^+ A_2$ then both A_1 and A_2 are steady, of respective base $\mathbf{B}_1$ and $\mathbf{B}_2$. The behaviour $\mathbf{B} = \mathbf{B}_1 \otimes^+ \mathbf{B}_2$ is a basis for A, indeed: since $\mathbf{B}_1$ and $\mathbf{B}_2$ are regular, Proposition 3.2.8 gives

$$V_{\mathbf{B}_1\otimes^+\mathbf{B}_2} = \kappa_\bullet(V^x_{{\uparrow}\mathbf{B}_1} \sqcup\!\sqcup V^y_{{\uparrow}\mathbf{B}_2}) \cup \{\maltese\}$$

where, by Proposition 3.2.1,

$$V_{{\uparrow}\mathbf{B}_i} = \kappa_\blacktriangle V^x_{\mathbf{B}_i} \cup \{\epsilon\} \qquad \text{for } i \in \{1,2\}\ ;$$

from this, and using internal completeness, we deduce that $\mathbf{B}$ satisfies all the conditions.

- Suppose $A = \mu X.A_0$, where A_0 is steady and has a basis $\mathbf{B}_0$, let us show that $\mathbf{B}_0$ is also a basis for A.
  - By (the proof of) Proposition 4.3.2,
    $$|[\![A]\!]^\sigma| = \bigcup_{n\in\mathbb{N}} |(\phi_\sigma^{A_0})^n(\maltese)| \ ,$$
    and since $\mathbf{B}_0$ is a basis for A_0 we have
    $$|\mathbf{B}_0| \subseteq |[\![A_0]\!]^{\sigma,X\to\maltese}| = |(\phi_\sigma^{A_0})(\maltese)| \ ,$$
    so indeed $|\mathbf{B}_0| \subseteq |[\![A]\!]^\sigma|$.
  - By induction hypothesis, we immediately have that for every path $s \in V_{\mathbf{B}_0}$, there exists $t \in V_{\mathbf{B}_0}^{max}$ $\maltese$-free extending s.
  - By Lemma 4.2.3(2)
    $$V_{[\![A]\!]^\sigma} = \{\maltese\} \cup \bigcup_{n\in\mathbb{N}} V_{(\phi_\sigma^{A_0})^{n+1}(\maltese)} = \{\maltese\} \cup \bigcup_{n\in\mathbb{N}} V_{[\![A_0]\!]^{\sigma_n}}$$
    where $\sigma_n = \sigma, X \mapsto (\phi_\sigma^{A_0})^n(\maltese)$ has a simple regular image. By induction hypothesis, for all $n \in \mathbb{N}$, $V_{\mathbf{B}}^{max} \subseteq V_{[\![A_0]\!]^{\sigma_n}}^{max}$, therefore $V_{\mathbf{B}}^{max} \subseteq V_{[\![A]\!]^\sigma}^{max}$.

□

Proposition 4.3.5

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 continuity of the function ϕ_σ^A implies that it is monotone, thus

$$(\phi_\sigma^A)^n(\maltese) \subseteq (\phi_\sigma^A)^n(\mathbf{B}) \subseteq (\phi_\sigma^A)^{n+1}(\maltese)$$

for all $n \in \mathbb{N}$ (straightforward induction). Hence

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

□

Proposition 4.3.6

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 3.4.1. 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 4.3.5, let us prove it satisfies the hypotheses needed to apply the second point of Corollary 4.2.6. By induction hypothesis and Proposition 4.3.1, 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 have

$$|(\phi_\sigma^{A_0})^n(\mathbf{B}_0)| \subseteq |(\phi_\sigma^{A_0})^{n+1}(\mathbf{B}_0)| \ ,$$

indeed: we showed in the proof of Propositions 4.3.1 and 4.3.2 that for $\mathbf{P}, \mathbf{P'} \in \mathcal{B}^+$ simple regular such that $|\mathbf{P}| \subseteq |\mathbf{P'}|$ we had $|\phi_\sigma^{A_0}(\mathbf{P})| \subseteq |\phi_\sigma^{A_0}(\mathbf{P'})|$; thus it is easy to prove the inclusion above by induction on n since the base case (for $n = 0$) corresponds to $|\mathbf{B}_0| \subseteq |[\![A_0]\!]^{\sigma, X\mapsto\mathbf{B}_0}|$, and this holds by definition of a basis. Finally, by Corollary 4.2.6, $[\![A]\!]^\sigma$ is pure. □

Corollary 4.3.7

Data behaviours are pure.

Remark 4.3.8

We have just seen that data behaviours – i.e., behaviours interpreting closed steady data patterns – are regular (Corollary 4.3.3), pure (Corollary 4.3.7), and simple (deduced from Proposition 4.3.1). An interesting question is whether there are other behaviours that are regular, pure and simple. The answer is yes, here are some ideas why:

- In a technical – and not so interesting – way, we could consider generalised data patterns with n-ary connectives, and/or interpret a connective by any name in $\mathcal{S}$ of the same arity instead of fixing one (the way we fixed $\bullet$ for $\otimes$), and we would still get regular pure simple behaviours.
- The interpretations of non-steady data patterns are simple and regular (Proposition 4.3.1), and we are convinced that they are pure as well but we do not know how to prove it. On the other hand, we believe that the following stronger form of purity is satisfied by steady data patterns, but not by the non-steady ones: *any ✠-ended visitable path can be extended by a ✠-free maximal visitable path.*
- Coinductive behaviours, that we will discuss briefly in Section 4.4, are regular. At least some of them are also pure and simple – typically the examples $\mathbb{N}\mathrm{at}^\omega$, $\mathbb{L}\mathrm{ist}_A^\infty$ and $\mathbb{S}\mathrm{tr}_A$ we will give then – and they might all be but this is still to prove.
- Apart from those, we have the feeling that the only other behaviours able to satisfy all the conditions would correspond to *non-recursive* types, i.e., types that cannot be described by a grammar but only by mean of a non-recursive function. This direction has to be explored further so as to determine if such behaviours

could be regular, pure and simple.

4.3.c About Regularity and μMALL

Although in this chapter the focus is on the interpretation of data patterns, we should say a word about the interpretation of (polarised) μMALL [Bae12] in ludics. μMALL corresponds to multiplicative–additive linear logic extended with least and greatest fixed points, denoted $\mu X.A$ and $\nu X.A$ respectively. The formulas of a polarised version of μMALL, which are a bit more general than data patterns, are generated by:

$$\begin{array}{rcl} P, Q & ::= & X_p \mid X_n^\perp \mid 1 \mid 0 \mid M \oplus N \mid M \otimes N \mid \downarrow N \mid \mu X_p.P \\ M, N & ::= & X_n \mid X_p^\perp \mid \bot \mid \top \mid P \,\&\, Q \mid P \parr Q \mid \uparrow P \mid \nu X_n.M \end{array}$$

where the usual involutive negation relies on the dualities $1/\bot$, $0/\top$, $\oplus/\&$, $\otimes/\parr$, $\downarrow/\uparrow$, μ/ν; in particular, we have

$$(\mu X_p.P)^\perp = \nu X_n.(P^\perp[X_n^\perp/X_p])$$

where X_n is a fresh negative variable. The interpretation of these formulas as ludics behaviours, given in [BDS15], 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,
- the negation corresponds to the orthogonal.

We know that constants and ✠ are regular, and that regularity is preserved by the connectives (Proposition 3.3.1) and by orthogonality (Remark 1.3.6). Notice moreover that, by applying Kleene's theorem to fixed points behaviours (Corollary 4.1.7) and by Lemma 4.2.5, we get that *regularity is preserved by least fixed points*. Hence the following.

Proposition 4.3.9

The behaviours interpreting μMALL formulas are regular.

Why is this interesting? Fouqueré and Quatrini [FQ16] have proved that regularity captures exactly MALL if we restrict to *finite behaviours*, where a behaviour is **finite** if its incarnation contains a finite number of designs, each of which has a finite tree representation.

Proposition 4.3.10

A behaviour is the denotation of a polarised MALL formula if and only if it is regular and finite.

Note that Proposition 4.3.10 has been formalised in Girard's framework, thus some technical details in the definition of connectives are different, but it is essentially the same. Now, if we drop finiteness, we expect to get the following result (modulo some uninteresting technical details corresponding to first item of Remark 4.3.8).

Conjecture 4.3.11

A behaviour is the denotation of a polarised μMALL formula if and only if it is regular.

In fact, this conjecture would not hold if there existed regular non-recursive behaviours, as mentioned at the end of Remark 4.3.8. It might therefore be necessary to add a condition to the conjecture: this has to be investigated. Fouqueré and Quatrini do not have this recursivity issue since the behaviours – thus also the designs – they consider are finite.

4.4 Coinductive Types: Ideas

Extending our study to greatest fixed points $\nu X.A$, i.e., coinductive types, is the next objective. Here we give some ideas for future work in this direction.

We can for example define the following coinductive types, where $zero, nil \in \mathcal{S}'$:

$$\begin{aligned} \mathbb{N}\mathrm{at}^{\omega} &= \nu X.(zero \oplus^{+} X) \ , \\ \mathbb{L}\mathrm{ist}_{A}^{\infty} &= \nu X.(nil \oplus^{+} (A \otimes^{+} X)) \ , \\ \mathbb{S}\mathrm{tr}_{A} &= \nu X.(A \otimes^{+} X) \ . \end{aligned}$$

They correspond respectively to:

- the natural numbers extended with the infinite ordinal ω,
- the finite and infinite lists of elements of type A,
- the infinite lists (streams) of elements of type A.

Note that, contrarily to the other two examples, the inductive counterpart $\mu X.(A \otimes^{+} X)$ of type $\mathbb{S}\mathrm{tr}_A$ is not a steady data pattern. The idea is that such coinductive patterns correspond to infinite objects only (and not finite and infinite, like the two others); in particular, a computation in this type can only end by a daimon, i.e., a voluntary interruption of the program, otherwise the computation runs forever.

In ludics, the interpretation of coinductive types in an environment σ is straightforwardly given by

$$[\![\nu X.A]\!]^{\sigma} = \mathrm{gfp}(\phi_{\sigma}^{A})$$

and the Knaster–Tarski fixed point theorem ensures that such greatest fixed points behaviours exist [BDS15]. Although Kleene's theorem does not apply here, there exists a dual of this theorem for greatest fixed points (see e.g. [San09]), this might help us finding the explicit form of coinductive behaviours. Intuitively, it is clear that, compared to least fixed points, greatest ones add the infinite "limit" designs in (the incarnation of) behaviours. For the example of $\mathbb{N}\mathrm{at}^{\omega}$ given above, we should have

$$|[\![\mathbb{N}\mathrm{at}^{\omega}]\!]| = |[\![\mathbb{N}\mathrm{at}]\!]| \cup \{\mathfrak{d}_{\omega}\}$$

where

$$\mathfrak{d}_{\omega} = \mathrm{succ}(\mathfrak{d}_{\omega}) = x_0 | \iota_2 \langle \uparrow(x).{\mathfrak{d}_{\omega}}^{x} \rangle \ .$$

Concerning the visitable paths of such coinductive types, it seems that they are the same as the corresponding inductive ones, for example

$$V_{[\![\mathbb{N}\mathrm{at}^{\omega}]\!]} = V_{[\![\mathbb{N}\mathrm{at}]\!]} \ .$$

Indeed, a computation is always finite, thus it cannot distinguish between infinite objects and finite objects of unbounded size. Unless we consider a notion of ∞-path, as in the proof of Theorem 4.2.2, and we choose to accept infinite chattering as a convergent computation, as it is the case in [BT10a] for example. This choice would make orthogonality characterise safety rather than termination, which is sound when dealing with infinite data.

It would then be interesting to study the purity of these coinductive behaviours; note that regularity is already ensured since it holds for all denotations of μMALL formulas.

5 | Functional Types

In this chapter we combine data behaviours with the connective $\multimap$ to get *functional behaviours*, which are the ludics interpretation of functional types. The idea of functional behaviours is not a novelty. Girard [Gir01] has introduced sequents of behaviours ; for example the – non atomic – behaviour

$$\mathbf{A} \vdash \mathbf{B}$$

is the set of all the designs that, when interacting with a design in $\mathbf{A}$, produce a design in $\mathbf{B}$. Our interpretation of functional types in ludics is essentially the same, with some slight changes to ensure that functional behaviours are:

- atomic, i.e., composed of atomic designs, in the sense of Definition 1.1.10;
- positive (the important being that they are all of same polarity).

This allows us to combine types so as to get higher order function types, where functions can have other functions as arguments or as outputs.

In Section 5.1, we define the functional behaviours thanks to the connective $\multimap$. The goal of Section 5.2 is then to prove that, among those behaviours, the ones that correspond to higher-order functional types taking functions as arguments, which is typically the case of the type

$$(A \multimap B) \multimap C \ ,$$

are exactly the impure ones. This fact is interesting from a computational point of view, if we recall that purity ensures the safety of execution: given such a type, it means that some ludics programs of this type have bugs. This will be discussed in Section 5.3.

5.1 Functional Behaviours

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

Definition 5.1.1 (*Functional behaviour*)

A **functional behaviour** is a behaviour inductively generated by the grammar

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

where $\mathbf{P} \multimap^+ \mathbf{Q}$ stands for $\downarrow((\uparrow\mathbf{P}) \multimap \mathbf{Q})$.

Functional behaviours combine data behaviours with the logical connective $\multimap$, but also $\oplus$ and $\otimes$ so as to get sum and product types on functions. As for data, shifts are added so as to respect the polarities, and all the functional behaviours are positive.

In the previous chapter, we have shown that data behaviours are regular and pure. However, building up the functional behaviours with $\multimap$, we may loose purity. The next proposition ensures at least regularity and quasi-purity, a weaker form of purity that we introduced in Chapter 3 (Definition 3.4.2), for all functional types. It is immediately deduced from Propositions 3.3.1, 3.4.1 and 3.4.3.

Proposition 5.1.2

Functional behaviours are regular and quasi-pure.

Remark 5.1.3

In this thesis, we consider the functional types separately from fixed points, for simplicity. However, this keeps us from taking into account some interesting types, for example lists of functions. Allowing to take fixed points over functional types is a future work, but in order to do this we must carefully determine the restrictions to impose. Typically, we should probably forbid variables to appear in a negative position, for example we want to accept

$$\mu X.(A \multimap^+ X) \oplus^+ B \qquad \text{but not} \qquad \mu X.(X \multimap^+ A) \oplus^+ B \ .$$

Indeed, the first type corresponds to functions taking a finite number of arguments of type A and returning a result of type B, while the meaning of the second one is rather unclear. It would then be interesting to see if it is possible to generalise our internal completeness result for infinite unions to these new fixed points types.

5.2 Where Impurity Arises

The goal of this section is to prove Proposition 5.2.1, which identifies exactly the impure functional behaviours. In order to state it, consider **contexts** defined inductively as follows:

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

where $\mathbf{P}$ is a functional behaviour.

Proposition 5.2.1

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

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

The proof of this proposition first requires several lemmas.

Notation

Let us denote the set of functional behaviours by $\mathcal{F}$, and recall that $\mathcal{D}$ stands for the set of data behaviours.

Lemma 5.2.2

Let $\mathbf{P} \in \mathcal{D}$, and let $\mathbf{Q}$ be a pure regular behaviour. The behaviour $\mathbf{P} \multimap^+ \mathbf{Q}$ is pure.

Proof. By Proposition 3.4.1 it suffices to show that $(\uparrow\mathbf{P}) \multimap \mathbf{Q}$ is pure. Remark first that, by construction of data behaviours, the following assertion is satisfied in views (thus also in paths) of $\uparrow\mathbf{P}$: every proper positive action is justified by the negative action preceding it.

By regularity and Corollary 3.2.9, we have

$$V_{(\uparrow\mathbf{P})\multimap\mathbf{Q}} = \overbrace{\kappa_\bullet(V_{\uparrow\mathbf{P}} \sqcup\!\sqcup \widetilde{V_\mathbf{Q}})} \cup \{\epsilon\} \ .$$

Let $s\maltese \in V_{(\uparrow\mathbf{P})\multimap\mathbf{Q}}$, and we prove that it is extensible. There exist $t_1 \in V_{\uparrow\mathbf{P}}$ and $t_2 \in V_\mathbf{Q}$ such that

$$\widetilde{s\maltese} = \overline{s} \in \kappa_\bullet(t_1 \sqcup\!\sqcup \widetilde{t_2}) \ .$$

In particular t_1 is $\maltese$-free and t_2 is $\maltese$-ended, say $t_2 = t_2'\maltese$. Since $\mathbf{Q}$ is pure, there exists κ^+ such that $t_2'\kappa^+ \in V_\mathbf{Q}$. Let us show that $s\kappa^+$ is a path, i.e., that if κ^+ is justified then $\mathrm{just}(\kappa^+)$ appears in $\ulcorner s \urcorner$, by induction on the length of t_1:

- If $t_1 = \epsilon$ then $s\kappa^+ = t_2'\kappa^+$ hence it is a path.
- Suppose $t_1 = t_1'\kappa_p^-\kappa_p^+$. Since t_1 is $\maltese$-free, κ_p^+ is proper. Thus s is of the form

  $$s = s_1\overline{\kappa_p^-\kappa_p^+}s_2 \ ,$$

  and by induction hypothesis $s_1s_2\kappa^+$ is a path, i.e., $\mathrm{just}(\kappa^+)$ appears in $\ulcorner s_1s_2 \urcorner$.
  - Either $\ulcorner s \urcorner = \ulcorner s_1s_2 \urcorner$ and indeed $\mathrm{just}(\kappa^+)$ also appears in $\ulcorner s \urcorner$.
  - Or $\ulcorner s \urcorner$ is of the form

    $$\ulcorner s \urcorner = \ulcorner s_1 \urcorner\overline{\kappa_p^-\kappa_p^+}s_2'$$

    as, by the remark at the beginning of this proof, κ_p^+ is justified by κ_p^-. This means in particular that s_2' starts with the same positive action as s_2, therefore

    $$\ulcorner s_1s_2 \urcorner = \ulcorner s_1 \urcorner s_2' \ .$$

    Since $\mathrm{just}(\kappa^+)$ appears in $\ulcorner s_1s_2 \urcorner$, it also appears in $\ulcorner s \urcorner$.

Therefore $s\kappa^+$ is a path. Since $s\kappa^+ \in \overbrace{\kappa_\bullet(V_{\uparrow\mathbf{P}} \sqcup\!\sqcup \widetilde{V_\mathbf{Q}})}$ and the behaviours are regular, $s\kappa^+ \in V_{\mathbf{P}\multimap^+\mathbf{Q}}$, thus $s\maltese$ is extensible. As this is true for every $\maltese$-ended path in $V_{(\uparrow\mathbf{P})\multimap\mathbf{Q}}$, the behaviour $(\uparrow\mathbf{P}) \multimap \mathbf{Q}$ is pure, and so is $\mathbf{P} \multimap^+ \mathbf{Q}$. □

Lemma 5.2.3

If $\mathbf{P} \in \mathcal{F}$ *and* $\mathbf{Q} \in \mathrm{Const}$ *then* $\mathbf{P} \multimap^+ \mathbf{Q}$ *is pure.*

Proof. We prove that $(\uparrow\mathbf{P}) \multimap \mathbf{Q}$ is pure, and the conclusion will follow from Proposition 3.4.1. Let $\kappa^+ = x_0|\overline{a}\langle\overrightarrow{y}\rangle$ where $\mathbf{Q} = \mathbf{C}_a$, and let

$$s\maltese \in V_{(\uparrow\mathbf{P})\multimap\mathbf{Q}} \ .$$

As in the proof of Lemma 5.2.2, there exist $t_1 \in V_{\uparrow\mathbf{P}}$ and $t_2 \in V_{\mathbf{Q}}$ such that

$$\widetilde{s\maltese} = \overline{s} \in \kappa_\bullet(t_1 \sqcup\!\sqcup \widetilde{t_2})$$

with t_2 $\maltese$-ended. But $V_{\mathbf{Q}} = \{\maltese, \kappa^+\}$, thus $t_2 = \maltese$ and $\widetilde{t_2} = \epsilon$. Hence

$$s\maltese = \widetilde{\kappa_\bullet t_1} \ ,$$

and this path is extensible with action κ^+, indeed: $s\kappa^+$ is a path because κ^+ is justified by $\kappa_\bullet$, which is the only initial action of $s\kappa^+$ thus appearing in $\ulcorner s \urcorner$; moreover

$$\widetilde{s\kappa^+} \in \kappa_\bullet(t_1 \sqcup\!\sqcup \widetilde{\kappa^+})$$

where $\kappa^+ \in V_{\mathbf{Q}}$, therefore

$$s\kappa^+ \in V_{(\uparrow\mathbf{P})\multimap\mathbf{Q}} \ .$$

□

Lemma 5.2.4

Let $\mathbf{P}, \mathbf{Q} \in \mathcal{F}$*. If there exists a* $\maltese$*-free (resp.* $\maltese$*-ended) maximal path* $s \in V_{\mathbf{Q}}$*, then there exists a* $\maltese$*-free (resp.* $\maltese$*-ended) maximal path* $t \in V_{\mathbf{P}\multimap^+\mathbf{Q}}$*.*

Proof. Suppose there exists a $\maltese$-free (resp. $\maltese$-ended) maximal path $s \in V_{\mathbf{Q}}$. Since $\mathbf{P}$ is positive and different from $\maltese$, there exists $s' \in V_{\uparrow\mathbf{P}}$ non-empty and $\maltese$-free. Let $t' = \widetilde{\kappa_\bullet s' \tilde{s}}$, and remark that $t' = \overline{\kappa_\bullet s'} s$. This is a path (O- and P-visibility are satisfied), it belongs to $V_{(\uparrow\mathbf{P})\multimap\mathbf{Q}}$, it is $\maltese$-free (resp. $\maltese$-ended). Suppose it is extensible, and consider both the "$\maltese$-free" and the "$\maltese$-ended" cases:

- if s and t' are $\maltese$-free, then there exists a negative action κ^- such that

  $$t'\kappa^-\maltese \in V_{(\uparrow\mathbf{P})\multimap\mathbf{Q}} = \widetilde{\kappa_\bullet(V_{\uparrow\mathbf{P}} \sqcup\!\sqcup V_{\mathbf{Q}^\perp})} \cup \{\epsilon\} \ .$$

  Since $t'\kappa^-\maltese = \overline{\kappa_\bullet s'} s\kappa^-\maltese$, we necessarily have $s\kappa^-\maltese \in V_{\mathbf{Q}}$ (indeed: the sequence $\overline{s'}\kappa^-$ has two adjacent negative actions thus cannot be a path). This contradicts the maximality of s in $V_{\mathbf{Q}}$.
- if s and t' are $\maltese$-ended, there exists a positive action κ^+ that extends t' and a contradiction arises with a similar reasoning.

Hence t' is maximal in $V_{(\uparrow \mathbf{P}) \multimap \mathbf{Q}}$. Finally, $t = \kappa_{\blacktriangledown} t'$ fulfills the requirements. □

Lemma 5.2.5

For every behaviour $\mathbf{P} \in \mathcal{F}$, there exists $s \in V_{\mathbf{P}}$ maximal and $\maltese$-free.

***Proof*.** By induction on $\mathbf{P}$. If $\mathbf{P} \in \mathcal{D}$ then take $s \in V_{\mathbf{B}}$ maximal, where $\mathbf{B}$ is a base of $\mathbf{P}$. Use Lemma 5.2.4 in the case of $\multimap^+$, and the result is easy for $\otimes^+$ and $\oplus^+$. □

Lemma 5.2.6

Let $\mathbf{P} \in \mathcal{F}$ and let $\mathcal{C}$ be a context. If $\mathcal{C}[\mathbf{P}]$ pure then $\mathbf{P}$ pure.

***Proof*.** We prove the contrapositive by induction on $\mathcal{C}$. Suppose $\mathbf{P}$ is impure.

- If $\mathcal{C} = [\,]$ then $\mathcal{C}[\mathbf{P}] = \mathbf{P}$, thus $\mathcal{C}[\mathbf{P}]$ is impure.
- If $\mathcal{C} = \mathcal{C}' \oplus^+ \mathbf{Q}$ or $\mathbf{Q} \oplus^+ \mathcal{C}'$ and by induction hypothesis $\mathcal{C}'[\mathbf{P}]$ is impure, i.e., there exists a maximal path $s\maltese \in V_{\mathcal{C}'[\mathbf{P}]}$, then one of $\kappa_{\iota_1}\kappa_{\blacktriangle} s\maltese$ or $\kappa_{\iota_2}\kappa_{\blacktriangle} s\maltese$ is maximal in $V_{\mathcal{C}[\mathbf{P}]}$, hence the result.
- If $\mathcal{C} = \mathcal{C}' \otimes^+ \mathbf{Q}$ or $\mathbf{Q} \otimes^+ \mathcal{C}'$ and by induction hypothesis there exists a maximal path $s\maltese \in V_{\mathcal{C}'[\mathbf{P}]}$, then by Lemma 5.2.5, there exists a $\maltese$-free maximal path $t \in V_{\mathbf{Q}}$. Consider the path $u = \kappa_{\bullet}\kappa^t_{\blacktriangle}\, t \kappa^s_{\blacktriangle}\, s\maltese$, where:
  - $\kappa^t_{\blacktriangle}$ justifies the first action of t,
  - $\kappa^s_{\blacktriangle}$ justifies the first action of s, and
  - $\kappa_{\bullet}$ justifies $\kappa^t_{\blacktriangle}$ and $\kappa^s_{\blacktriangle}$, one on each ($1^{\text{st}}$ or 2^{nd}) position, depending on the form of $\mathcal{C}$.

  We have $u \in V_{\mathcal{C}[\mathbf{P}]}$, and u is $\maltese$-ended and maximal, hence the result.
- If $\mathcal{C} = \mathbf{Q} \multimap^+ \mathcal{C}'$ and by induction hypothesis $\mathcal{C}'[\mathbf{P}]$ is impure, then Lemma 5.2.4 (in its "$\maltese$-ended" version) concludes the proof.

□

We can now prove the proposition.

***Proof (Proposition 5.2.1)*.** ($\Rightarrow$) Suppose $\mathbf{P}$ impure. By induction on behaviour $\mathbf{P}$:

- $\mathbf{P} \in \mathcal{D}$ is impossible by Corollary 4.3.7.
- If $\mathbf{P} = \mathbf{P}_1 \oplus^+ \mathbf{P}_2$ (resp. $\mathbf{P} = \mathbf{P}_1 \otimes^+ \mathbf{P}_2$) then one of $\mathbf{P}_1$ or $\mathbf{P}_2$ is impure by Proposition 3.4.1, say $\mathbf{P}_1$. By induction hypothesis, $\mathbf{P}_1$ is of the form

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

  Let $\mathcal{C}_1 = \mathcal{C}'_1 \oplus^+ \mathbf{P}_2$ (resp. $\mathcal{C}_1 = \mathcal{C}'_1 \otimes^+ \mathbf{P}_2$) and $\mathcal{C}_2 = \mathcal{C}'_2$, and we get the result for $\mathbf{P}$.
- If $\mathbf{P} = \mathbf{P}_1 \multimap^+ \mathbf{P}_2$, then $\mathbf{P}_2 \notin \mathrm{Const}$ by Lemma 5.2.3, and:
  - If $\mathbf{P}_2$ impure, then by induction hypothesis $\mathbf{P}_2$ is of the form

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

and it suffices to take $\mathcal{C}_1 = \mathbf{P}_1 \to \mathcal{C}'_1$ and $\mathcal{C}_2 = \mathcal{C}'_2$ to get the result for $\mathbf{P}$.

– If $\mathbf{P}_2$ is pure, since it is also regular the conclusion follows from Lemma 5.2.2.

($\Leftarrow$) Let $\mathcal{C}_1, \mathcal{C}_2$ be contexts, $\mathbf{Q}_1, \mathbf{Q}_2, \mathbf{R} \in \mathcal{P}$ with $\mathbf{R} \notin \mathrm{Const}$. Let

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

We prove that $\mathbf{P}$ is impure.

First suppose that

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

and in this case we show the result by induction on the depth of context $\mathcal{C}_2$. The exact induction hypothesis will be:

$$\text{there exists a maximal } \maltese\text{-ended path in } V_{\mathbf{P}} \text{ of the form } \kappa_{\blacktriangledown} s \maltese$$
$$\text{where } \overline{s} \in \kappa_\bullet((\kappa_{\blacktriangle} V_{\mathbf{Q}}) \sqcup\!\sqcup \widetilde{V_{\mathbf{R}}}).$$

- If $\mathcal{C}_2 = [\,]$, then

$$\mathbf{Q} = \mathbf{Q}_1 \multimap^+ \mathbf{Q}_2 = \downarrow(\uparrow\mathbf{Q}_1 \multimap \mathbf{Q}_2) \qquad \text{and} \qquad \mathbf{P} = \mathbf{Q} \multimap^+ \mathbf{R} = \downarrow(\uparrow\mathbf{Q} \multimap \mathbf{R}) \ .$$

In order to differentiate actions $\kappa_{\blacktriangledown}, \kappa_{\blacktriangle}, \kappa_\bullet$ used to construct $\mathbf{Q}$ from those to construct $\mathbf{P}$, we will use corresponding superscripts. Let $\kappa^Q_{\blacktriangle} t_1 \in V_{\uparrow\mathbf{Q_1}}$ be $\maltese$-free (and non-empty). Let $t_2 \in V_{\mathbf{Q}_2}$ be a maximal $\maltese$-free path: its existence is ensured by Lemma 5.2.5, and it has one proper positive initial action κ_2^+. Now let:

$$t = \widetilde{\kappa^Q_\bullet \kappa^Q_{\blacktriangle} t_1 \widetilde{t_2}} = \overline{\kappa^Q_\bullet \kappa^Q_{\blacktriangle} t_1}\, t_2 \ .$$

Similarly to the path constructed in proof of Lemma 5.2.4, we have that t is $\maltese$-free, it is in $V_{(\uparrow\mathbf{Q_1})\multimap\mathbf{Q_2}}$, and it is maximal. Thus $\kappa^Q_{\blacktriangledown} t \in V_{\mathbf{Q}}$. Since $\mathbf{R} \notin \mathrm{Const}$, there exists a path of the form $\kappa^+\kappa^-\maltese \in V_{\mathbf{R}}$, and thus necessarily κ^+ justifies κ^-. Define the sequence:

$$s\maltese = \overline{\kappa^P_\bullet \kappa^P_{\blacktriangle} \kappa^Q_{\blacktriangledown}} \kappa^Q_\bullet \kappa^Q_{\blacktriangle} \kappa^+ \kappa^- \, t_1 \overline{t_2} \maltese$$

and notice the following facts:

1. $s\maltese$ is a path: it is a linear aj-sequence. Since κ^- is justified by κ^+, O- and P-visibility are easy to check.
2. $s\maltese \in V_{\uparrow\mathbf{Q}\multimap\mathbf{R}}$: indeed, we have

$$\widetilde{s\maltese} \in \kappa^P_\bullet(\kappa^P_{\blacktriangle} \kappa^Q_{\blacktriangledown} t \sqcup\!\sqcup \widetilde{\kappa^+\kappa^-\maltese})$$

where $\kappa^P_{\blacktriangle} \kappa^Q_{\blacktriangledown} t \in V_{\uparrow\mathbf{Q}}$ and $\kappa^+\kappa^-\maltese \in V_{\mathbf{R}}$.

3. $s\maltese$ is maximal: Let us show that $s\maltese$ is not extensible. First, it is not possible to extend it with an action from $\mathbf{Q}^\perp$, because this would contradict the maximality of t in $V_{\mathbf{Q}}$. Suppose it is extensible with an action $\kappa^{+\prime}$ from $\mathbf{R}$, i.e.,

$$s\kappa^{+\prime} \in V_{\uparrow\mathbf{Q}\multimap\mathbf{R}} \qquad \text{and} \qquad \widetilde{s\kappa^{+\prime}} \in \kappa^P_\bullet(\kappa^P_{\blacktriangle} \kappa^Q_{\blacktriangledown} t \sqcup\!\sqcup \widetilde{\kappa^+\kappa^-\kappa^{+\prime}})$$

where $\kappa^+\kappa^-\kappa^{+\prime} \in V_{\mathbf{R}}$. The action $\kappa^{+\prime}$ (that cannot be initial) is necessarily justified by κ^-. But $\ulcorner s \urcorner$ contains necessarily the first negative action of $\overline{t_2}$, which is the only initial action in $\overline{t_2}$, and this action is justified by $\kappa^Q_\bullet$ in s. Therefore $\ulcorner s \urcorner$ does not contain any action from s between $\kappa^Q_\bullet$ and $\overline{t_2}$, in particular it does not contain $\kappa^- = \mathrm{just}(\kappa^{+\prime})$. Thus $s\kappa^{+\prime}$ is not P-visible: contradiction. Hence $s\maltese$ maximal.

Finally $\kappa^P_\blacktriangledown s\maltese \in V_{\mathbf{P}}$ is not extensible, and of the required form.

- If $\mathcal{C}_2 = \mathbf{Q}_0 \multimap^+ \mathcal{C}$, then $\mathbf{Q}$ is of the form

$$\mathbf{Q} = \mathbf{Q}_0 \multimap^+ \mathbf{Q}' \ ,$$

thus the previous reasoning applies.

- If $\mathcal{C}_2 = \mathcal{C} \otimes^+ \mathbf{Q}_0$ or $\mathbf{Q}_0 \otimes^+ \mathcal{C}$, the induction hypothesis gives us the existence of a maximal path in $V_{\mathcal{C}[\mathbf{Q_1} \multimap^+ \mathbf{Q_2}] \multimap^+ \mathbf{R}}$ of the form $\kappa^P_\blacktriangledown \overline{\kappa^P_\bullet \kappa^P_\blacktriangle} s'\maltese$ where $\kappa^P_\blacktriangle \overline{s'} \in (\kappa^P_\blacktriangle t') \sqcup\!\sqcup \widetilde{u}$ with $t' \in V_{\mathcal{C}[\mathbf{Q_1} \multimap^+ \mathbf{Q_2}]}$ and $u \in V_{\mathbf{R}}$. Let $t_0 \in V_{\mathbf{Q_0}}$ be $\maltese$-free and maximal, using Lemma 5.2.5. Consider the following sequence:

$$s\maltese = \overline{\kappa^P_\bullet \kappa^P_\blacktriangle \kappa^Q_\bullet \kappa^0_\blacktriangle t_0 \kappa^1_\blacktriangle} s'\maltese$$

where:

  - $\kappa^0_\blacktriangle$ justifies the first action of t_0,
  - $\kappa^1_\blacktriangle$ justifies the first action of $\overline{s'}$ thus the first action of t',
  - $\kappa^Q_\bullet$ justifies $\kappa^0_\blacktriangle$ and $\kappa^1_\blacktriangle$,
  - $\kappa^P_\blacktriangle$ now justifies $\kappa^Q_\bullet$,
  - $\kappa^P_\bullet$ justifies the same actions as before.

Notice that:

1. $s\maltese$ is a path: O- and P-visibility are satisfied.
2. $\overline{s\maltese \in V_{\uparrow \mathbf{Q} \multimap \mathbf{R}}}$: We have

$$\kappa^Q_\bullet \kappa^0_\blacktriangle t_0 \kappa^1_\blacktriangle \overline{t'} \quad \in \quad \kappa^Q_\bullet (\kappa^0_\blacktriangle V_{\mathbf{Q_0}} \sqcup\!\sqcup \kappa^1_\blacktriangle V_{\mathcal{C}[\mathbf{Q_1} \multimap^+ \mathbf{Q_2}]}) = V_{\mathbf{Q}} \ ,$$

hence $\widetilde{s\maltese} \in \kappa^P_\bullet (V_{\uparrow \mathbf{Q}} \sqcup\!\sqcup \widetilde{V_{\mathbf{R}}})$.

3. $s\maltese$ is maximal: Indeed, it cannot be extended neither by an action of $\mathbf{Q}_0^\perp$ (contradicts the maximality of t_0) nor by an action of $\mathcal{C}[\mathbf{Q}_1 \multimap^+ \mathbf{Q}_2]^\perp$ or $\mathbf{R}$ (contradicts the maximality of s').

Finally $\kappa^P_\blacktriangledown s\maltese \in V_{\mathbf{P}}$ is a path satisfying the constraints.

- If $\mathcal{C}_2 = \mathcal{C} \oplus^+ \mathbf{Q}_0$ or $\mathbf{Q}_0 \oplus^+ \mathcal{C}$, by induction hypothesis, there exists a path of the form $\kappa^P_\blacktriangledown \overline{\kappa^P_\bullet \kappa^P_\blacktriangle} s'\maltese$ maximal in $V_{\mathcal{C}[\mathbf{Q_1} \multimap^+ \mathbf{Q_2}] \multimap^+ \mathbf{R}}$, where $\kappa^P_\blacktriangle \overline{s'} \in (\kappa^P_\blacktriangle t') \sqcup\!\sqcup \widetilde{u}$ with $t' \in V_{\mathcal{C}[\mathbf{Q_1} \multimap^+ \mathbf{Q_2}]}$ and $u \in V_{\mathbf{R}}$. Reasoning as the previous item, we see that for one of $i \in \{1, 2\}$ (depending on the form of context $\mathcal{C}_2$) the path $\kappa^P_\blacktriangledown \overline{\kappa^P_\bullet \kappa^P_\blacktriangle \kappa^Q_{\iota_i} \kappa_\blacktriangle} s'\maltese$ (where $\kappa^P_\blacktriangle$ now justifies $\kappa^Q_{\iota_i}$) is in $V_{\mathbf{P}}$, maximal, and of the required form.

The result for the general case, where $\mathbf{P} = \mathcal{C}_1[\, \mathcal{C}_2[\mathbf{Q}_1 \multimap^+ \mathbf{Q}_2] \multimap^+ \mathbf{R}\,]$, finally comes from Lemma 5.2.6. □

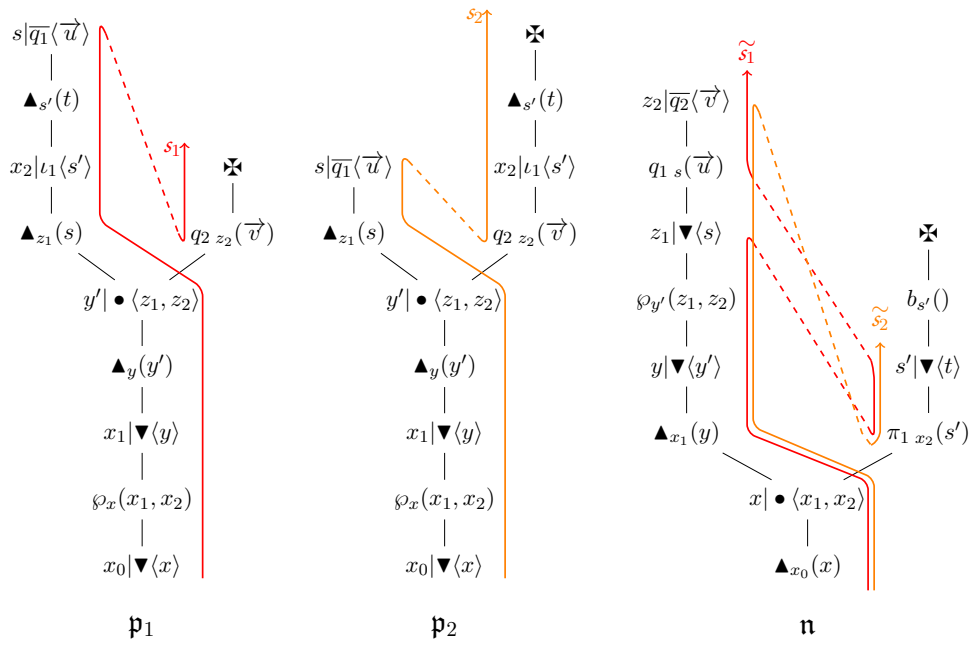


Figure 6: Designs $\mathfrak{p}_1$, $\mathfrak{p}_2$ and $\mathfrak{n}$

5.3 Example and Discussion

Proposition 5.2.1 states that a functional behaviour which takes functions as argument is not pure: some of its visitable paths end with a daimon $\maltese$, and there is no possibility to extend them. In terms of proof-search, playing the daimon is like giving up; from 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.

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. The designs $\mathfrak{p}_1$ and $\mathfrak{p}_2$ represented in Figure 6 are in $\mathbf{P}$, while $\mathfrak{n} \in \mathbf{P}^\perp$. The visitable path $s_1 = \langle \mathfrak{p}_1 \leftarrow \mathfrak{n} \rangle$ is $\maltese$-ended and maximal in $V_{\mathbf{P}}$, in other words this path is an evidence of the impurity of $\mathbf{P}$. Indeed, if we let respectively

$$\begin{aligned} \kappa_1 &= x_0|\overline{q_1}\langle\overrightarrow{u}\rangle && \text{be the first action of designs in} && \mathbf{Q}_1 \ , \\ \kappa_2 &= x_0|\overline{q_2}\langle\overrightarrow{v}\rangle && \text{be the first action of designs in} && \mathbf{Q}_2 \ , \\ \beta &= x_0|\bar{b}\langle\rangle && \text{be the first action of designs in} && \mathbf{1} \ , \end{aligned}$$

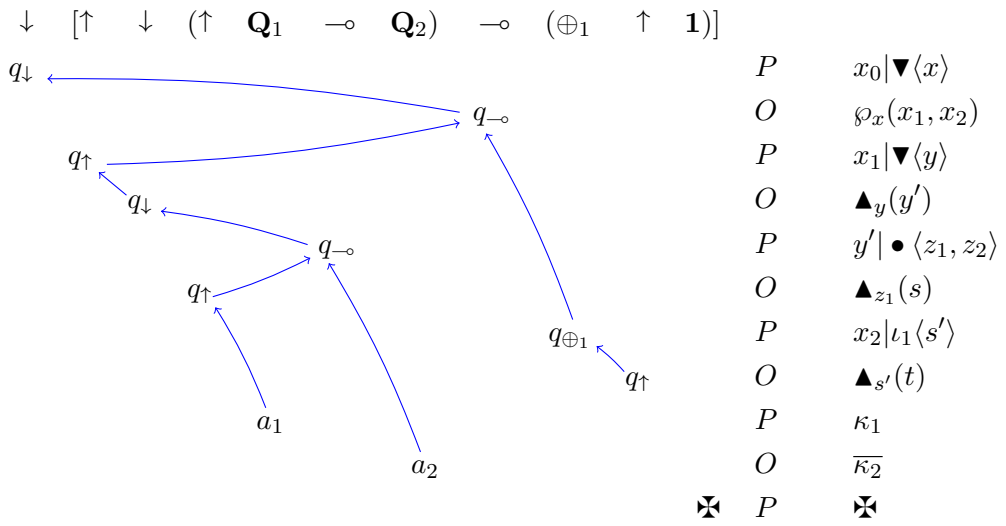


Figure 7: Representation of path $s_1 \in V_{\mathbf{P}}$ in the style of a play

then s_1 contains the actions κ_1 and $\overline{\kappa_2}$ 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. On the contrary, the path $s_2 = \langle \mathfrak{p}_2 \leftarrow \mathfrak{n} \rangle$ is ✠-ended but extensible with the action β.

We also give an intuition in the style of game semantics: Figure 7 represents s_1 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_1 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 $\mathfrak{p}_1$ 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, leading to a situation where p cannot compute the whole data in $\mathbf{Bool}$. The interaction outputs ✠, i.e., the answer given in $\mathbf{Bool}$ by p is incomplete.

Moreover, by Proposition 5.1.2, functional behaviours are quasi-pure, therefore the maximal ✠-ended visitable paths are necessarily not well-bracketed. This is indeed the case of s_1: 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 programs (see e.g [AM99]). If we extend such an argument to ludics, this would mean that the appearance of ✠ in the evaluation 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 | Non-Linear Ludics

Original ludics [Gir01] is strictly linear – one should actually say *affine* – hence it lacks the possibility of expressing the exponentials of linear logic. There has been various propositions to extend ludics with non-linearity [BF11, BT10b, Mau04, Ter11]. Here we follow the one of Basaldella and Terui [BT10b], in which designs are non-linear and non-deterministic, extending the syntax used in the previous chapters. They prove that their version of ludics is expressive enough to capture LLP. By abuse, we call this setting *non-linear ludics*.

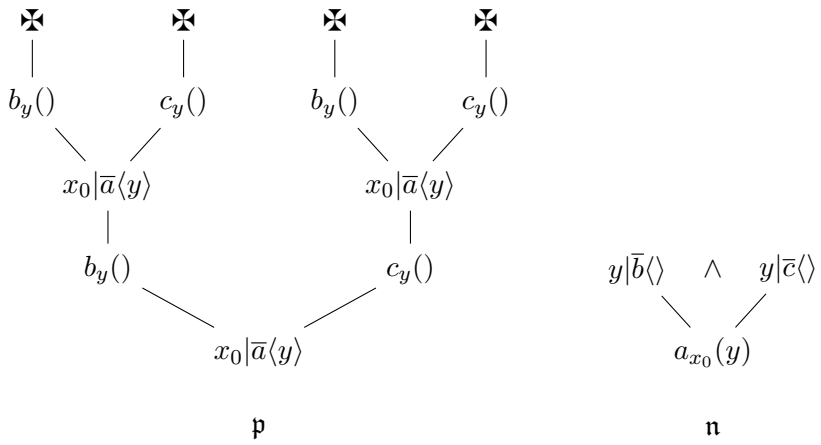


Figure 8: Non-linearity vs. non-determinism

Non-linearity in ludics corresponds to the possibility of repeating an address in a design. This results in the fact that, during the interaction, some designs can be copied and used several times. Non-determinism is needed in order to have enough *tests* against these non-linear designs, the same way ✠ gives enough tests to interact with linear ones. Indeed, consider for example the non-linear design $\mathfrak{p}$ of Figure 8. If $\mathfrak{p}$ interacts only against deterministic designs, two branches of the tree can never be visited, since choosing either b or c at the first branching will force the next choice to be the same. On the other hand, if we make it interact with the non-deterministic design $\mathfrak{n}$ – where non-determinism is expressed by the conjunction of positive terms – then all the branches can be visited. From this point of view, non-determinism corresponds to the ability for a program to give differ-

ent results when asked successively the same question. The non-determinism we consider here is *universal* (must testing) rather than *existential* (may testing): an interaction converges if and only if all possible choices made at positive conjunctions lead to ✠.

This chapter is much more a work in progress than the previous ones. In Section 6.1, we upgrade the definitions of designs, behaviours and multi-designs to non-linear ludics. This allows us to investigate internal completeness in Section 6.2: although it holds for negative connectives, it does not for positive ones, so we provide alternative results that still give some interesting information on the structure of behaviours. In Section 6.3 we conjecture that n-path is the right notion to capture an interaction trace in the non-linear setting. We end the chapter by giving, in Section 6.4, some directions for future work.

6.1 Basic Definitions

We still fix a set of variables $\mathcal{V}_0$ and a signature $\mathcal{S}$, and we suppose $\mathcal{S}$ contains the names $\blacktriangle, \pi_1, \pi_2, \wp$ for the connectives.

6.1.a Designs

We recall, in this subsection, the definitions from [BT10b]. From now on, a design can be non-linear (i.e., no restriction on the occurrences of free variables) and non-deterministic (i.e., conjunctions of positive terms).

Definition 6.1.1 (*Design*)

The class of **positive designs** $\mathfrak{p}, \mathfrak{q}, \ldots$, that of **predesigns** $\mathfrak{s}, \mathfrak{t}, \ldots$, and that of **negative designs** $\mathfrak{n}, \mathfrak{m}, \ldots$ are coinductively defined as follows:

$$\begin{array}{lll}
\mathfrak{p} ::= & \Omega & \textbf{(divergence)}, \\
\quad | & \bigwedge\{\mathfrak{s}_i : i \in I\} & \textbf{(conjunction)}, \\
\mathfrak{s} ::= & x|\overline{a}\langle \mathfrak{n}_1, \ldots, \mathfrak{n}_{\mathrm{ar}(a)}\rangle & \textbf{(simple predesign)}, \\
\quad | & \mathfrak{n}_0|\overline{a}\langle \mathfrak{n}_1, \ldots, \mathfrak{n}_{\mathrm{ar}(a)}\rangle & \textbf{(cut)}, \\
\mathfrak{n} ::= & \displaystyle\sum_{a\in\mathcal{S}} a(\overrightarrow{x}).\mathfrak{p}_a & \textbf{(abstraction)},
\end{array}$$

where $I \subseteq \mathbb{N}\mathrm{at}$ is an index set.

The daimon is now encoded by the empty conjunction: ✠ $= \bigwedge \emptyset$. Several definitions of Chapter 1 are adapted in a straightforward way, to the new setting: **free variables**, **substitution**, **total** design, **subdesign**, **linear** design, **atomic** design, **closed** design.

Definition 6.1.2 (*Deterministic design*)

A design is **deterministic** if, in any occurrence of subdesign $\mathfrak{p} = \bigwedge\{\mathfrak{s}_i : i \in I\}$, the set I is either empty (i.e., $\mathfrak{p} =$ ✠) or a singleton (i.e., $\mathfrak{p}$ is a predesign).

Notation

- By abuse, given a predesign $\mathfrak{s}$ and a positive design $\mathfrak{p} = \bigwedge\{\mathfrak{s}_i : i \in I\}$, we write $\mathfrak{s} \in \mathfrak{p}$ if there exists $i \in I$ such that $\mathfrak{s} = \mathfrak{s}_i$.
- Following the same idea, we write $\mathfrak{p} \rightsquigarrow^* \ni \mathfrak{s}$ if $\mathfrak{p} \rightsquigarrow^* \mathfrak{q}$ and $\mathfrak{s} \in \mathfrak{q}$, where the reduction step $\rightsquigarrow$ is defined below.

Definition 6.1.3 (*Normalisation / interaction*)

The reduction step on (non-linear non-deterministic) designs is defined as follows:

$$\text{if} \quad (\sum_{a\in\mathcal{S}} a(\overrightarrow{x^a}).\mathfrak{p}_a) \mid \overline{b}\langle\overrightarrow{\mathfrak{n}}\rangle \quad \in \mathfrak{p} \quad \text{then} \quad \mathfrak{p} \rightsquigarrow \mathfrak{p}_b[\overrightarrow{\mathfrak{n}/x^b}] \ .$$

There are indeed several possibilities of reduction from a conjunction. For example if $\mathfrak{p} = x_0|\overline{a}\langle\rangle \wedge x_0|\overline{b}\langle\rangle$ and $\mathfrak{n} = a().\mathfrak{p}_1 + b().\mathfrak{p}_2$ then we have

$$\mathfrak{p}[\mathfrak{n}/x_0] \rightsquigarrow \mathfrak{p}_1 \qquad \text{and} \qquad \mathfrak{p}[\mathfrak{n}/x_0] \rightsquigarrow \mathfrak{p}_2 \ .$$

Definition 6.1.4 (*Normal form*)

The **normal form** of a design $\mathfrak{d}$ is defined by:

$$\begin{aligned}
(\!|\mathfrak{p}|\!) &= \Omega && \text{if } \mathfrak{p} \rightsquigarrow^* \Omega \text{ or if there is an infinite reduction sequence starting from } \mathfrak{p};\\
&= \bigwedge\{\ x|\overline{a}\langle(\!|\overrightarrow{\mathfrak{n}}|\!)\rangle \ \mid\ \mathfrak{p} \rightsquigarrow^* \ni x|\overline{a}\langle\overrightarrow{\mathfrak{n}}\rangle\ \} && \text{otherwise;}\\
(\!|\sum_{a\in\mathcal{S}} a(\overrightarrow{x^a}).\mathfrak{p}_a|\!) &= \sum_{a\in\mathcal{S}} a(\overrightarrow{x^a}).(\!|\mathfrak{p}_a|\!) \ .
\end{aligned}$$

In particular, given a closed design $\mathfrak{p}$, we have:

- $(\!|\mathfrak{p}|\!) = \maltese$ if all the reduction sequences from $\mathfrak{p}$ end with $\maltese$,
- $(\!|\mathfrak{p}|\!) = \Omega$ otherwise.

This is the reason why the non-determinism here is universal: overall convergence requires that any possible choice converges. The notions of orthogonality and behaviour then correspond to the linear case.

Definition 6.1.5 (*Orthogonality, behaviour*)

Two atomic designs $\mathfrak{p}$ and $\mathfrak{n}$ are **orthogonal** if $(\!|\mathfrak{p}[\mathfrak{n}/x_0]|\!) = \maltese$. A **behaviour** $\mathbf{B}$ is a set of cut-free atomic designs of same polarity such that $\mathbf{B}^{\perp\perp} = \mathbf{B}$.

Finally, recall that the associativity theorem (Theorem 1.1.7) states that, given designs $\mathfrak{d}, \mathfrak{n}_1, \ldots, \mathfrak{n}_k$, we have:

$$(\!|\mathfrak{d}[\mathfrak{n}_1/y_1, \ldots, \mathfrak{n}_k/y_k]|\!) = (\!|(\!|\mathfrak{d}|\!)[(\!|\mathfrak{n}_1|\!)/y_1, \ldots, (\!|\mathfrak{n}_k|\!)/y_k]|\!) \ .$$

This theorem still holds with non-linear non-deterministic designs; it is in this extended setting that Basaldella and Terui [BT10b] proved it. We will need associativity for some proofs in the following.

6.1.b Multi-Designs

Again, we need a notion of multi-design, introduced for the linear case in Chapter 2. Indeed the internal completeness results for $\parr$ and $\otimes$ we give in the next section refer to behaviours of multi-designs. We just recall the main definitions, with slight modifications to fit the non-linear setting. The designs in a multi-design are no longer required to have disjoint sets of free variables, since this requirement corresponds to a linearity condition.

Definition 6.1.6 (*Multi-design*)

- A **negative multi-design** is a set
$$\{(x_1, \mathfrak{n}_1), \dots, (x_k, \mathfrak{n}_k)\}$$
where $x_1, \dots, x_k$ are distinct variables and $\mathfrak{n}_1, \dots, \mathfrak{n}_k$ are negative designs, such that for all i with $1 \leq i \leq k$ we have $\mathrm{fv}(\mathfrak{n}_i) \cap \{x_1, \dots, x_k\} = \emptyset$.
- A **positive multi-design** is a set
$$\{\mathfrak{p}, (x_1, \mathfrak{n}_1), \dots, (x_k, \mathfrak{n}_k)\}$$
where $\{(x_1, \mathfrak{n}_1), \dots, (x_k, \mathfrak{n}_k)\}$ is a negative multi-design and $\mathfrak{p}$ is a positive design such that $\mathrm{fv}(\mathfrak{p}) \cap \{x_1, \dots, x_k\} = \emptyset$.

The **normal form** of a multi-design, as well as its **free variables** and **negative places**, are defined exactly like in the linear case.

Given two multi-designs $\mathfrak{D}$ and $\mathfrak{E}$, we define the directed graph $\mathcal{G}(\mathfrak{D}, \mathfrak{E})$ with:

- $\mathrm{np}(\mathfrak{D}) \cup \mathrm{np}(\mathfrak{E})$ as set of vertices,
- an edge from x to y if $y \in \mathrm{fv}(\mathfrak{n})$ where $(\mathfrak{n}/x) \in \mathfrak{D} \cup \mathfrak{E}$.

Definition 6.1.7 (*Compatible, closed-compatible*)

Let $\mathfrak{D}$ and $\mathfrak{E}$ be multi-designs.

- $\mathfrak{D}$ and $\mathfrak{E}$ are **compatible** if they satisfy the following conditions:
  - $\mathrm{np}(\mathfrak{D}) \cap \mathrm{np}(\mathfrak{E}) = \emptyset$,
  - they are either both negative or of opposite polarities,
  - the directed graph $\mathcal{G}(\mathfrak{D}, \mathfrak{E})$ is acyclic.
- $\mathfrak{D}$ and $\mathfrak{E}$ are **closed-compatible** if they are of opposite polarities, compatible, and satisfying $\mathrm{fv}(\mathfrak{D}) = \mathrm{np}(\mathfrak{E})$ and $\mathrm{fv}(\mathfrak{E}) = \mathrm{np}(\mathfrak{D})$.

The acyclicity condition prevents from having negative designs $\mathfrak{n}_1, \dots, \mathfrak{n}_k$ such that, for all $1 \leq i < k$, $\mathfrak{n}_i$ is substituted in $\mathfrak{n}_{i+1}$ and $\mathfrak{n}_k$ is substituted in $\mathfrak{n}_1$. Such a situation would not necessarily be problematic, since designs are defined coinductively, but for simplicity we do not want to consider this because we will not need it. Notice that acyclicity was always

satisfied in the linear version of the definition of compatible (Definition 2.1.5), thanks to linearity and the additional condition in the case both multi-designs were negative.

The multi-design $\mathfrak{Cut}_{\mathfrak{D}|\mathfrak{E}}$, which operates the substitutions between two multi-designs, is then defined as one would expect, leading naturally to the notions of orthogonality and behaviour.

Definition 6.1.8 ($\mathfrak{Cut}$)

Let $\mathfrak{D}$ and $\mathfrak{E}$ be compatible multi-designs. $\mathfrak{Cut}_{\mathfrak{D}|\mathfrak{E}}$ is a multi-design defined by induction on the number of designs in $\mathfrak{E}$:

$$\mathfrak{Cut}_{\mathfrak{D}|\emptyset} = \mathfrak{D} \ , \tag{1}$$

$$\mathfrak{Cut}_{\mathfrak{D}|\mathfrak{E}} = \mathfrak{Cut}_{(\mathfrak{D}\setminus S')\cup\{\mathfrak{p}[S]\} \mid \mathfrak{E}\setminus\{\mathfrak{p}\}} \qquad \text{if } \mathfrak{p} \in \mathfrak{E} \ , \tag{2}$$

$$\mathfrak{Cut}_{\mathfrak{D}|\mathfrak{E}} = \mathfrak{Cut}_{(\mathfrak{D}\setminus S')\cup\{\mathfrak{n}[S]/x\} \mid \mathfrak{E}\setminus\{\mathfrak{n}/x\}} \qquad \text{if } (\mathfrak{n}/x) \in \mathfrak{E} \text{ and } x \notin \mathrm{fv}(\mathfrak{D}) \ , \tag{3}$$

$$\mathfrak{Cut}_{\mathfrak{D}|\mathfrak{E}} = \mathfrak{Cut}_{(\mathfrak{D}\setminus S')[\mathfrak{n}[S]/x] \mid \mathfrak{E}\setminus\{\mathfrak{n}/x\}} \qquad \text{if } (\mathfrak{n}/x) \in \mathfrak{E} \text{ and } x \in \mathrm{fv}(\mathfrak{D}) \ , \tag{4}$$

where, if we let $\mathfrak{d} = \mathfrak{p}$ in (2) and $\mathfrak{d} = \mathfrak{n}$ in (3) and (4),

$$S = \{(\mathfrak{m}/y) \in \mathfrak{D} \mid y \in \mathrm{fv}(\mathfrak{d})\} \qquad \text{and} \qquad S' = S \setminus \{(\mathfrak{m}/y) \mid y \in \mathrm{fv}(\mathfrak{E} \setminus \{\mathfrak{d}\})\} \ .$$

Definition 6.1.9 (*Orthogonality, behaviour*)

Two closed-compatible multi-designs $\mathfrak{D}$ and $\mathfrak{E}$ are **orthogonal** if $(\!|\mathfrak{Cut}_{\mathfrak{D}|\mathfrak{E}}|\!) = \maltese$. A set $\mathbf{B}$ of cut-free multi-designs of same polarity is a **behaviour** if $\mathbf{B}^{\perp\perp} = \mathbf{B}$.

In the rest of this chapter, behaviours of designs will be called "atomic behaviours".

6.2 About Internal Completeness

This section presents new results about the atomic behaviours constructed by connectives in a non-linear setting. In non-linear ludics, internal completeness holds for negative connectives, but it does not hold anymore for positive ones. To overcome this issue, we prove alternative – and still meaningful – results for $\downarrow$, $\oplus$ and $\otimes$ (Propositions 6.2.6, 6.2.7 and 6.2.8 respectively); these results are similar to what Basaldella and Faggian obtain [BF11] in a different non-linear ludics setting than ours, but they do not take additives into account. Before this, we need preliminaries about the conjunction of designs.

6.2.a Conjunction of Designs

The non-determinism of designs allows us to define the conjunction $\mathfrak{p} \wedge \mathfrak{q}$ of two positive designs, such that a design $\mathfrak{n}$ is orthogonal to $\mathfrak{p} \wedge \mathfrak{q}$ if and only if it is orthogonal both to $\mathfrak{p}$ and $\mathfrak{q}$. Such a conjunction does not work that well on negative designs, but one implication is still preserved (see Lemma 6.2.2).

Definition 6.2.1 (*Conjunction of designs*)

Let $\mathfrak{p}, \mathfrak{q}$ be positive designs. The **conjunction** of $\mathfrak{p}$ and $\mathfrak{q}$, noted $\mathfrak{p} \wedge \mathfrak{q}$ is defined by:

$$\begin{aligned} \mathfrak{p} \wedge \mathfrak{q} &= \Omega && \text{if } \mathfrak{p} = \Omega \text{ or } \mathfrak{q} = \Omega \ , \\ &= \bigwedge\{\mathfrak{s}_k : k \in I \cup J\} && \text{if } \mathfrak{p} = \bigwedge\{\mathfrak{s}_i : i \in I\} \text{ and } \mathfrak{q} = \bigwedge\{\mathfrak{s}_j : j \in J\} \\ & && \text{with } I \text{ and } J \text{ disjoint} \ . \end{aligned}$$

Note that, given two total positive designs, we can always suppose that their index set is disjoint modulo renaming, thus the conjunction of two positive designs is always defined.

The next lemma justifies the definition; we state it in a generalised form, where opponents can be multi-design.

Lemma 6.2.2

Let $\mathfrak{p}_1, \mathfrak{p}_2$ be positive designs.

1. *For every negative multi-design $\mathfrak{N}$, we have*

$$\mathfrak{p}_1 \wedge \mathfrak{p}_2 \perp \mathfrak{N} \qquad \textit{if and only if} \qquad (\mathfrak{p}_1 \perp \mathfrak{N} \quad \textit{and} \quad \mathfrak{p}_2 \perp \mathfrak{N}) \ .$$

2. *For every positive multi-design $\mathfrak{P}$, every $a \in \mathcal{S}$ and all variables $\overrightarrow{x}$, we have*

$$a(\overrightarrow{x}).(\mathfrak{p}_1 \wedge \mathfrak{p}_2) \perp \mathfrak{P} \qquad \textit{implies} \qquad (a(\overrightarrow{x}).\mathfrak{p}_1 \perp \mathfrak{P} \quad \textit{and} \quad a(\overrightarrow{x}).\mathfrak{p}_2 \perp \mathfrak{P}) \ .$$

***Proof*.**

1. Note that $\mathfrak{Cut}_{(\mathfrak{p}_1 \wedge \mathfrak{p}_2)|\mathfrak{N}} = (\mathfrak{p}_1 \wedge \mathfrak{p}_2)[\mathfrak{N}]$, and by definition we have

$$(\!|(\mathfrak{p}_1 \wedge \mathfrak{p}_2)[\mathfrak{N}]|\!) = (\!|\mathfrak{p}_1[\mathfrak{N}] \wedge \mathfrak{p}_2[\mathfrak{N}]|\!) = (\!|\mathfrak{p}_1[\mathfrak{N}]|\!) \wedge (\!|\mathfrak{p}_2[\mathfrak{N}]|\!)$$

   if neither $\mathfrak{p}_1[\mathfrak{N}]$ nor $\mathfrak{p}_2[\mathfrak{N}]$ reduces to Ω. Moreover $\mathfrak{Cut}_{\mathfrak{p}_1|\mathfrak{N}} = \mathfrak{p}_1[\mathfrak{N}]$ and $\mathfrak{Cut}_{\mathfrak{p}_2|\mathfrak{N}} = \mathfrak{p}_2[\mathfrak{N}]$. Therefore

$$(\!|\mathfrak{Cut}_{(\mathfrak{p}_1 \wedge \mathfrak{p}_2)|\mathfrak{N}}|\!) = \maltese \qquad \text{if and only if} \qquad (\!|\mathfrak{Cut}_{\mathfrak{p}_1|\mathfrak{N}}|\!) = (\!|\mathfrak{Cut}_{\mathfrak{p}_2|\mathfrak{N}}|\!) = \maltese \ .$$

2. We prove the result for $\mathfrak{P} = \mathfrak{p}$ an atomic design, the multi-design case being similar. Let us note

$$\begin{aligned} \mathfrak{n} &= a(\overrightarrow{x}).(\mathfrak{p}_1 \wedge \mathfrak{p}_2) \\ \mathfrak{n}_1 &= a(\overrightarrow{x}).\mathfrak{p}_1 \\ \mathfrak{n}_2 &= a(\overrightarrow{x}).\mathfrak{p}_2 \end{aligned}$$

   and suppose $\mathfrak{n} \perp \mathfrak{p}$, that is

$$\mathfrak{p}[\mathfrak{n}/x_0] \rightsquigarrow^* \maltese \ .$$

This means that each sequence of reductions starting from $\mathfrak{p}[\mathfrak{n}/x_0]$ and following a particular sequence of choices ends on a daimon. In particular, some of these choices are between $\mathfrak{p}_1$ and $\mathfrak{p}_2$, when reaching a copy of design $\mathfrak{n}$.
But any sequence of reduction starting from $\mathfrak{p}[\mathfrak{n}_1/x_0]$ or from $\mathfrak{p}[\mathfrak{n}_2/x_0]$ matches a reduction sequence from $\mathfrak{p}[\mathfrak{n}/x_0]$, since it corresponds to making the same choice (either always $\mathfrak{p}_1$ or always $\mathfrak{p}_2$) every time we reach a copy of $\mathfrak{n}$. Let us explain this more precisely, reasoning by contradiction. Suppose there exists a reduction sequence from $\mathfrak{p}[\mathfrak{n}_i/x_0]$ (where $i = 1$ or 2) which is not a sequence of reduction from $\mathfrak{p}[\mathfrak{n}/x_0]$. Thus, at some point during interaction, the reduction sequences split, and this happens necessarily in a situation involving $\mathfrak{n}$ vs. $\mathfrak{n}_i$ (interacting against a design of the form $\mathfrak{q} = x_0|\overline{a}\langle\overrightarrow{\mathfrak{m}}\rangle$) since this is the only difference between the two interactions. In such a situation, we have

$$\mathfrak{q}[\mathfrak{n}/x_0] \rightsquigarrow \mathfrak{p}_1'[\mathfrak{n}/x_0] \qquad \text{and} \qquad \mathfrak{q}[\mathfrak{n}/x_0] \rightsquigarrow \mathfrak{p}_2'[\mathfrak{n}/x_0]$$

where

$$\mathfrak{p}_1' = \mathfrak{p}_1[\overrightarrow{\mathfrak{m}_i/x_i}] \qquad \text{and} \qquad \mathfrak{p}_2' = \mathfrak{p}_2[\overrightarrow{\mathfrak{m}_i/x_i}] \ ;$$

on the other hand,

$$\mathfrak{q}[\mathfrak{n}_i/x_0] \rightsquigarrow \mathfrak{p}_i'[\mathfrak{n}_i/x_0] \ .$$

Therefore, each time we reach this point during the interaction of $\mathfrak{p}[\mathfrak{n}/x_0]$, it suffices to choose $\mathfrak{p}_i$, and the interaction will continue the same way it does for $\mathfrak{p}[\mathfrak{n}_i/x_0]$: contradiction. We deduce that all the reduction sequences from $\mathfrak{p}[\mathfrak{n}_i/x_0]$ end with ✠, hence $\mathfrak{n}_i \perp \mathfrak{p}$.

□

Example 6.2.3

Let us illustrate the fact that the converse of Lemma 6.2.2(2) does not hold in general. Consider the designs depicted on Figure 9, where

$$\mathfrak{n}_1 = a(y).(y|\overline{b}\langle\rangle) \qquad \text{and} \qquad \mathfrak{n}_2 = a(y).(y|\overline{c}\langle\rangle) \ .$$

We have $\mathfrak{n}_1 \perp \mathfrak{p}$ and $\mathfrak{n}_2 \perp \mathfrak{p}$, but if we consider the conjunction

$$\mathfrak{n} = a(y).(y|\overline{b}\langle\rangle \wedge y|\overline{c}\langle\rangle)$$

of $\mathfrak{n}_1$ and $\mathfrak{n}_2$, then $\mathfrak{n} \not\perp \mathfrak{p}$.

Remark that design $\mathfrak{n}$ from the previous example is represented on Figure 8. Actually, the designs of Figures 8 and 9 shed light on the fact that non-deterministic opponents make it possible to explore a non-linear design throughout, leading to interaction traces (i.e., paths) that we could not get in a deterministic setting. This corresponds to a non-uniform semantics for programs, as remarked by Maurel [Mau04]. Indeed, if we let `true` $= \mathfrak{n}_1$ and `false` $= \mathfrak{n}_2$, then $\mathfrak{n}$ is an argument that, when interrogated, can output either `true` or `false`. Such a term cannot be defined in a uniform (that is, deterministic) world.

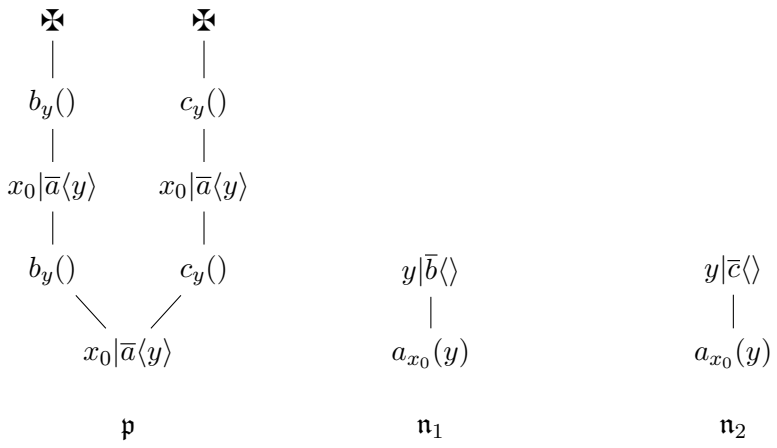


Figure 9: A counterexample to the converse of Lemma 6.2.2(2)

Note that, with respect to interaction, the conjunction of designs $\wedge$ is close to the intersection of designs $\cap$ that we defined in the linear deterministic setting (Definition 1.1.13). We could extend this definition to embed non-linear non-deterministic designs, and if we did so we would have the following: for all positive designs $\mathfrak{p}$ and $\mathfrak{q}$ such that $\mathfrak{p} \cap \mathfrak{q}$ is defined, and for any negative multi-design $\mathfrak{N}$,

$$(\mathfrak{p} \cap \mathfrak{q}) \perp \mathfrak{N} \qquad \text{if and only if} \qquad (\mathfrak{p} \wedge \mathfrak{q}) \perp \mathfrak{N} \ .$$

In other words, $\mathfrak{p} \cap \mathfrak{q}$ and $\mathfrak{p} \wedge \mathfrak{q}$ are not *separable*. However $\mathfrak{p} \cap \mathfrak{q} \neq \mathfrak{p} \wedge \mathfrak{q}$, and there would also be cases when $\mathfrak{p} \cap \mathfrak{q}$ is not defined but $\mathfrak{p} \wedge \mathfrak{q}$ is.

6.2.b Logical Connectives and Negative Internal Completeness

The logical connectives $\downarrow$, $\uparrow$, $\oplus$, $\otimes$ and $\multimap$ are defined as in Definition 1.1.20, except that now our notion of design has been widened. We can also consider the negative connectives $\&$ (**with**) and $\parr$ (**par**) defined dualy to $\oplus$ and $\otimes$ respectively:

$$\mathbf{P} \,\&\, \mathbf{Q} = (\mathbf{P}^\perp \oplus \mathbf{Q}^\perp)^\perp \ ,$$
$$\mathbf{P} \parr \mathbf{Q} = (\mathbf{P}^\perp \otimes \mathbf{Q}^\perp)^\perp \ .$$

Similarly to Lemma 3.2.3, we have

$$\uparrow\mathbf{P} = (\downarrow\mathbf{P}^\perp)^\perp \ .$$

Note that we do not have explicit connectives ! and ? though we said that non-linearity in ludics could model exponentials. In fact, non-linear ludics corresponds to logical systems where formulas are implicitly exponential, typically LLP in which the structural rules are extended to all the negative formulas. Without drawing a precise correspondence (others did [BT10b]), the next subsection will still give an idea about how the interaction process in behaviours constructed by logical connectives can now duplicate designs.

For the three negative connectives, we have the following internal completeness result.

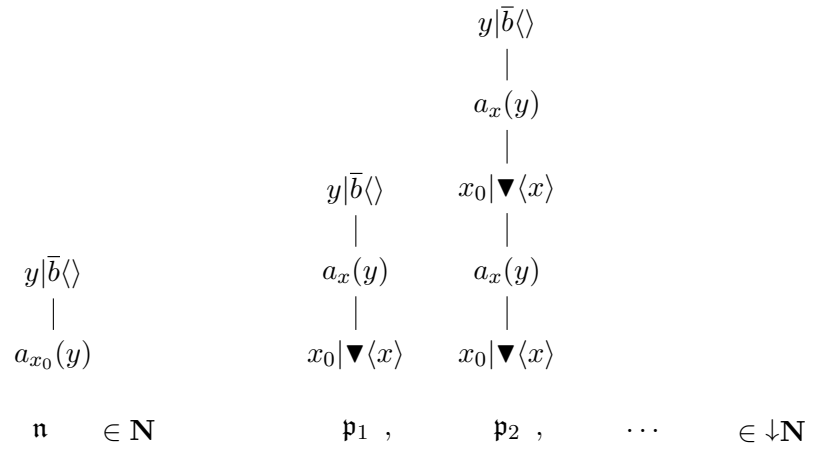


Figure 10: A counterexample to internal completeness for $\downarrow$

Theorem 6.2.4 (*Internal completeness, negative non-linear case*)

Let $\mathbf{P}, \mathbf{Q}$ *be positive atomic behaviours.*

$$\begin{aligned} \uparrow\mathbf{P} &= \{\mathfrak{n} \mid \mathfrak{n}\uparrow\blacktriangle \in \blacktriangle(x).\mathbf{P}^x\} \ , \\ \mathbf{P} \,\&\, \mathbf{Q} &= \{\mathfrak{n} \mid \mathfrak{n}\uparrow\pi_1 \in \pi_1(x).\mathbf{P}^x \textit{ and } \mathfrak{n}\uparrow\pi_2 \in \pi_2(x).\mathbf{Q}^x\} \ , \\ \mathbf{P} \parr \mathbf{Q} &= \{\mathfrak{n} \mid \mathfrak{n}\uparrow\wp \in \wp(x,y).\mathbf{R}\} \ , \end{aligned}$$

where $\mathbf{R} = \{\{\mathfrak{n}_1/x, \mathfrak{n}_2/y\} \mid \mathfrak{n}_1 \in \mathbf{P}^\perp, \mathfrak{n}_2 \in \mathbf{Q}^\perp\}^\perp$ *is a behaviour of multi-designs.*

This result comes directly from Theorem 2.17 of [BT10b]. The incarnated form of the theorem would follow easily, if we had defined incarnation for non-linear designs and multi-designs.

6.2.c About Positive Internal Completeness

A problem is that, in this non-linear setting, internal completeness does not hold anymore for the positive connectives, as shown for $\downarrow$ in the following example.

Example 6.2.5

Consider the behaviour $\mathbf{N} = \{\mathfrak{n}\}^{\perp\perp}$ where $\mathfrak{n} = a(y).(y|\bar{b}\langle\rangle)$. The positive atomic behaviour $\downarrow\mathbf{N}$ contains in particular all the designs $\mathfrak{p}_i$ defined by:

$$\begin{aligned} \mathfrak{p}_1 &= x_0|\blacktriangledown\langle\mathfrak{n}\rangle \ , \\ \mathfrak{p}_{i+1} &= x_0|\blacktriangledown\langle a(y).\mathfrak{p}_i\rangle \ . \end{aligned}$$

If $i > 1$ then $\mathfrak{p}_i \notin \blacktriangledown\langle\mathbf{N}\rangle$, contradicting the internal completeness theorem for the linear case (Theorem 1.1.21). All these designs are represented in Figure 10.

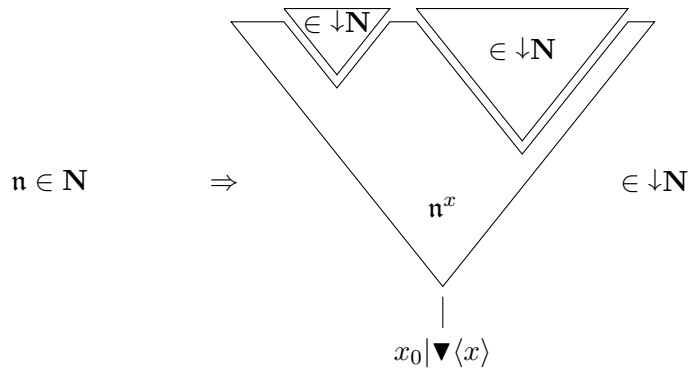


Figure 11: Construction of $\downarrow\mathbf{N}$

We however have alternative results to internal completeness for $\downarrow$, $\oplus$ and $\otimes$ (Propositions 6.2.6, 6.2.7 and 6.2.8 respectively). Let us start with the shift.

Proposition 6.2.6

Given a negative atomic behaviour $\mathbf{N}$*, the following are equivalent:*

1. $\mathfrak{p} \in \downarrow\mathbf{N} \setminus \{\maltese\}$.
2. $\mathfrak{p} = x_0|\blacktriangledown\langle\mathfrak{n}\rangle$ *such that for every* $\mathfrak{m} \in \uparrow\mathbf{N}^\perp$, $([\mathfrak{n}[\mathfrak{m}/x_0]]) \in \mathbf{N}$.
3. $\mathfrak{p} = x_0|\blacktriangledown\langle\mathfrak{n}\rangle$ *such that for every* $\mathfrak{q} \in \mathbf{N}^\perp$, $([\mathfrak{q}[\mathfrak{n}/x_0]]) \in \downarrow\mathbf{N}$.

Let us explain to what extent this result is close to internal completeness. It states that

$$\downarrow\mathbf{N} = \blacktriangledown\langle\mathbf{N}'\rangle \cup \{\maltese\}$$

where, though $\mathbf{N}' \neq \mathbf{N}$ (unlike linear internal completeness), we still have an interesting property which has two equivalent formulations:

- $\mathbf{N}'$ is the set of designs falling in $\mathbf{N}$ when interacting with a design of $\uparrow\mathbf{N}^\perp$.
- $\mathbf{N}'$ is the set of designs falling in $\downarrow\mathbf{N}$ when interacting with a design of $\mathbf{N}^\perp$.

In particular, the first point acknowledges that, contrarily to the linear case, a design in $\downarrow\mathbf{N}$ can force its opponent (in $\uparrow\mathbf{N}^\perp$) to duplicate, so as to interact several times with it. Put differently, $\mathbf{N}'$ is the behaviour of multi-designs defined by

$$\mathbf{N}' = \{\{\mathfrak{q}, \mathfrak{m}/x_0\} \mid \mathfrak{q} \in \mathbf{N}^\perp, \mathfrak{m} \in \uparrow\mathbf{N}^\perp\}^\perp \quad .$$

In particular, indeed, we have $\mathbf{N} \subseteq \mathbf{N}'$.

Note that we can also describe $\downarrow\mathbf{N}$ in a coinductive way, by:

- $\blacktriangledown\langle\mathbf{N}\rangle \cup \{\maltese\} \subseteq \downarrow\mathbf{N}$
- if $\mathfrak{n} \in \mathbf{N}$ then $x_0|\blacktriangledown\langle\mathfrak{n}'\rangle \in \downarrow\mathbf{N}$ where $\mathfrak{n}'$ is obtained from $\mathfrak{n}$ by replacing some of its positive subdesigns by designs of $\downarrow\mathbf{N}$ (as illustrated in Figure 11).

Let us now give the equivalent results for $\oplus$ and $\otimes$. In fact, the three propositions are very akin to each other, and so are their proofs. It would probably be possible to factorise them into a single proposition, if we considered a generic form of connective [Ter11, BT10b]. We leave it as future work.

Proposition 6.2.7

Given negative atomic behaviours $\mathbf{N}_1$ *and* $\mathbf{N}_2$*, the following are equivalent:*

1. $\mathfrak{p} \in \mathbf{N}_1 \oplus \mathbf{N}_2 \setminus \{\maltese\}$.
2. $\mathfrak{p} = x_0|\iota_i\langle\mathfrak{n}\rangle$ *with* $i \in \{1,2\}$ *and such that for every* $\mathfrak{m} \in \mathbf{N}_1^\perp \,\&\, \mathbf{N}_2^\perp$
$$(\!|\mathfrak{n}[\mathfrak{m}/x_0]|\!) \in \mathbf{N}_i \ .$$
3. $\mathfrak{p} = x_0|\iota_i\langle\mathfrak{n}\rangle$ *with* $i \in \{1,2\}$ *and such that for every* $\mathfrak{q} \in \mathbf{N}_i^\perp$
$$(\!|\mathfrak{q}[\mathfrak{n}/x_0]|\!) \in \mathbf{N}_1 \oplus \mathbf{N}_2 \ .$$

Proposition 6.2.8

Given negative atomic behaviours $\mathbf{N}_1$ *and* $\mathbf{N}_2$*, the following are equivalent:*

1. $\mathfrak{p} \in \mathbf{N}_1 \otimes \mathbf{N}_2 \setminus \{\maltese\}$.
2. $\mathfrak{p} = x_0|\bullet\langle\mathfrak{n}_1,\mathfrak{n}_2\rangle$ *such that for every* $i \in \{1,2\}$ *and every* $\mathfrak{m} \in \mathbf{N}_1^\perp \parr \mathbf{N}_2^\perp$
$$(\!|\mathfrak{n}_i[\mathfrak{m}/x_0]|\!) \in \mathbf{N}_i \ .$$
3. $\mathfrak{p} = x_0|\bullet\langle\mathfrak{n}_1,\mathfrak{n}_2\rangle$ *such that for every* $i \in \{1,2\}$ *and every* $\mathfrak{q} \in \mathbf{N}_i^\perp$
$$(\!|\mathfrak{q}[\mathfrak{n}_i/x_0]|\!) \in \mathbf{N}_1 \otimes \mathbf{N}_2 \ .$$

Similarly to the case of the shift, we have

$$\mathbf{N}_1 \oplus \mathbf{N}_2 = \iota_1\langle\mathbf{N}_1'\rangle \cup \iota_2\langle\mathbf{N}_2'\rangle \cup \{\maltese\}$$
$$\text{where } \mathbf{N}_i' = \{\{\mathfrak{q},\mathfrak{m}/x_0\} \mid \mathfrak{q} \in \mathbf{N}_i^\perp, \mathfrak{m} \in \mathbf{N}_1^\perp \,\&\, \mathbf{N}_2^\perp\}^\perp$$

and

$$\mathbf{N}_1 \otimes \mathbf{N}_2 = \bullet\langle\mathbf{N}_1',\mathbf{N}_2'\rangle \cup \{\maltese\}$$
$$\text{where } \mathbf{N}_i' = \{\{\mathfrak{q},\mathfrak{m}/x_0\} \mid \mathfrak{q} \in \mathbf{N}_i^\perp, \mathfrak{m} \in \mathbf{N}_1^\perp \parr \mathbf{N}_2^\perp\}^\perp \ .$$

We give two out of three proofs for these propositions, since they are very similar: the positive shift (Proposition 6.2.6), which is the simplest, and the tensor (Proposition 6.2.8), which leads to consider behaviours of multi-designs. An important remark is that the proofs rely on non-determinism, by taking the conjunction of some designs.

Proof (Prosition 6.2.6).

(1) $\Rightarrow$ (2) Let $\mathfrak{p} \in {\downarrow}\mathbf{N}$ such that $\mathfrak{p} \neq \maltese$, and let $\mathfrak{m} \in {\uparrow}\mathbf{N}^\perp$. We have ${\uparrow}\mathbf{N}^\perp = ({\downarrow}\mathbf{N})^\perp$, so $\mathfrak{p} \perp \mathfrak{m}$. By internal completeness (Theorem 6.2.4), there exists $\mathfrak{q} \in \mathbf{N}^\perp$ such that

$$\mathfrak{m}{\restriction}\blacktriangle = \blacktriangle(x).\mathfrak{q}^x \ .$$

Since $\mathfrak{p} \neq \Cross$, there exists a design $\mathfrak{n}$ such that

$$\mathfrak{p} = x_0 | \blacktriangledown \langle \mathfrak{n} \rangle \ .$$

Let $\mathfrak{q}' \in \mathbf{N}^\perp$, let us show that

$$(\!|\mathfrak{n}[\mathfrak{m}/x_0]|\!) \perp \mathfrak{q}' \ .$$

Let

$$\mathfrak{m}' = \blacktriangle(x).(\mathfrak{q}^x \wedge \mathfrak{q}'^x) \ .$$

Since $\mathfrak{q}, \mathfrak{q}' \in \mathbf{N}^\perp$ we have $\mathfrak{q} \wedge \mathfrak{q}' \in \mathbf{N}^\perp$ by Lemma 6.2.2(1), therefore $\mathfrak{m}' \in \uparrow\mathbf{N}^\perp$. Hence $\mathfrak{p} \perp \mathfrak{m}'$. By the unique one-step reduction possible, we deduce

$$(\!|\mathfrak{p}[\mathfrak{m}'/x_0]|\!) = (\!|(\mathfrak{q} \wedge \mathfrak{q}')[\mathfrak{n}[\mathfrak{m}'/x_0]/x_0]|\!) = \Cross \ ,$$

thus

$$\mathfrak{n}[\mathfrak{m}'/x_0] \perp (\mathfrak{q} \wedge \mathfrak{q}') \ .$$

By Lemma 6.2.2(1), this implies that

$$\mathfrak{n}[\mathfrak{m}'/x_0] \perp \mathfrak{q}' \ , \qquad \text{thus also} \qquad \mathfrak{m}' \perp \mathfrak{q}'[\mathfrak{n}/x_0]$$

(since $(\!|\mathfrak{q}'[\mathfrak{n}[\mathfrak{m}'/x_0]/x_0]|\!) = \Cross$). We deduce by Lemma 6.2.2(2) that

$$\mathfrak{m} \restriction \blacktriangle \perp \mathfrak{q}'[\mathfrak{n}/x_0] \ .$$

It follows immediately that

$$\mathfrak{m} \perp \mathfrak{q}'[\mathfrak{n}/x_0] \ , \qquad \text{thus also} \qquad \mathfrak{n}[\mathfrak{m}/x_0] \perp \mathfrak{q}' \ ,$$

and by associativity

$$(\!|\mathfrak{n}[\mathfrak{m}/x_0]|\!) \perp \mathfrak{q}' \ .$$

Hence the result.

(2) $\Rightarrow$ (1) Let

$$\mathfrak{p} = x_0 | \blacktriangledown \langle \mathfrak{n} \rangle$$

where $\mathfrak{n}$ is a design such that for every $\mathfrak{m} \in \uparrow\mathbf{N}^\perp$ we have

$$(\!|\mathfrak{n}[\mathfrak{m}/x_0]|\!) \in \mathbf{N} \ .$$

Let $\mathfrak{m} \in \uparrow\mathbf{N}^\perp$, and we show that

$$\mathfrak{p} \perp \mathfrak{m} \ .$$

By internal completeness, there exists $\mathfrak{q} \in \mathbf{N}^\perp$ such that

$$\mathfrak{m} \restriction \blacktriangle = \blacktriangle(x).\mathfrak{q}^x$$

and we have

$$\mathfrak{p}[\mathfrak{m}/x_0] \rightsquigarrow \mathfrak{q}[\mathfrak{n}[\mathfrak{m}/x_0]/x_0] \ ,$$

where $(\!|\mathfrak{n}[\mathfrak{m}/x_0]|\!) \in \mathbf{N}$ by hypothesis. Since $\mathfrak{q} \in \mathbf{N}^\perp$, by associativity we deduce

$$(\!|\mathfrak{q}[\mathfrak{n}[\mathfrak{m}/x_0]/x_0]|\!) = (\!|\mathfrak{q}[(\!|\mathfrak{n}[\mathfrak{m}/x_0]|\!)/x_0]|\!) = \maltese \ ,$$

thus

$$\mathfrak{p} \perp \mathfrak{m} \ .$$

Finally $\mathfrak{p} \in {\downarrow}\mathbf{N}$.

(2) $\Leftrightarrow$ (3) Suppose $\mathfrak{p}$ is of the form $\mathfrak{p} = x_0|\blacktriangledown\langle\mathfrak{n}\rangle$. Using associativity, we have:

$$\begin{array}{ll}
& \text{for every } \mathfrak{m} \in {\uparrow}\mathbf{N}^\perp, (\!|\mathfrak{n}[\mathfrak{m}/x_0]|\!) \in \mathbf{N} \\
\Leftrightarrow & \text{for every } \mathfrak{m} \in {\uparrow}\mathbf{N}^\perp \text{ and every } \mathfrak{q} \in \mathbf{N}^\perp, \mathfrak{q} \perp (\!|\mathfrak{n}[\mathfrak{m}/x_0]|\!) \\
\Leftrightarrow & \text{for every } \mathfrak{m} \in {\uparrow}\mathbf{N}^\perp \text{ and every } \mathfrak{q} \in \mathbf{N}^\perp, \mathfrak{q} \perp \mathfrak{n}[\mathfrak{m}/x_0] \\
\Leftrightarrow & \text{for every } \mathfrak{m} \in {\uparrow}\mathbf{N}^\perp \text{ and every } \mathfrak{q} \in \mathbf{N}^\perp, (\!|\mathfrak{q}[\mathfrak{n}[\mathfrak{m}/x_0]/x_0]|\!) = \maltese \\
\Leftrightarrow & \text{for every } \mathfrak{m} \in {\uparrow}\mathbf{N}^\perp \text{ and every } \mathfrak{q} \in \mathbf{N}^\perp, \mathfrak{q}[\mathfrak{n}/x_0] \perp \mathfrak{m} \\
\Leftrightarrow & \text{for every } \mathfrak{m} \in {\uparrow}\mathbf{N}^\perp \text{ and every } \mathfrak{q} \in \mathbf{N}^\perp, (\!|\mathfrak{q}[\mathfrak{n}/x_0]|\!) \perp \mathfrak{m} \\
\Leftrightarrow & \text{for every } \mathfrak{q} \in \mathbf{N}^\perp, (\!|\mathfrak{q}[\mathfrak{n}/x_0]|\!) \in {\downarrow}\mathbf{N} \ .
\end{array}$$

□

Proof (Prosition 6.2.8). In this proof, let $\mathbf{R} = \{\{\mathfrak{m}_1/x_1, \mathfrak{m}_2/x_2\} \mid \mathfrak{m}_1 \in \mathbf{N}_1, \mathfrak{m}_2 \in \mathbf{N}_2\}^\perp$. In particular we have $(\mathbf{N}_i^\perp)^{x_i} \subseteq \mathbf{R}$ for $i = 1, 2$.

(1) $\Rightarrow$ (2) Let $\mathfrak{p} \in \mathbf{N}_1 \otimes \mathbf{N}_2$ such that $\mathfrak{p} \neq \maltese$, and let $\mathfrak{m} \in \mathbf{N}_1^\perp \parr \mathbf{N}_2^\perp$. By definition $\mathfrak{p} \perp \mathfrak{m}$. By internal completeness, there exists $\mathfrak{q} \in \mathbf{R}$ such that

$$\mathfrak{m}\lceil_{\wp} = \wp(x_1, x_2).\mathfrak{q} \ .$$

Since $\mathfrak{p} \neq \maltese$, there exist designs $\mathfrak{n}_1, \mathfrak{n}_2$ such that

$$\mathfrak{p} = x_0| \bullet \langle \mathfrak{n}_1, \mathfrak{n}_2 \rangle \ .$$

Let $i \in \{1, 2\}$, let $\mathfrak{q}' \in \mathbf{N}_i^\perp$. We show that

$$(\!|\mathfrak{n}_i[\mathfrak{m}/x_0]|\!) \perp \mathfrak{q}' \ .$$

Let

$$\mathfrak{m}' = \wp(x_1, x_2).(\mathfrak{q} \wedge \mathfrak{q}'^{x_i}) \ ,$$

and note that $\mathfrak{q}'^{x_i} \in (\mathbf{N}_i^\perp)^{x_i} \subseteq \mathbf{R}$. Since $\mathfrak{q}, \mathfrak{q}'^{x_i} \in \mathbf{R}$ we have $\mathfrak{q} \wedge \mathfrak{q}'^{x_i} \in \mathbf{R}$ by Lemma 6.2.2(1), therefore $\mathfrak{m}' \in \mathbf{N}_1^\perp \parr \mathbf{N}_2^\perp$. Hence $\mathfrak{m}' \perp \mathfrak{p}$. By the unique one-step reduction possible, we deduce

$$(\!|\mathfrak{p}[\mathfrak{m}'/x_0]|\!) = (\!|(\mathfrak{q} \wedge \mathfrak{q}'^{x_i})[\mathfrak{n}_1/x_1, \mathfrak{n}_2/x_2][\mathfrak{m}'/x_0]|\!) = \maltese \ ,$$

hence

$$\{\mathfrak{n}_1/x_1, \mathfrak{n}_2/x_2\}[\mathfrak{m}'/x_0] \perp (\mathfrak{q} \wedge \mathfrak{q}'^{x_i}) \ .$$

By Lemma 6.2.2(1), this implies that

$$\{\mathfrak{n}_1/x_1, \mathfrak{n}_2/x_2\}[\mathfrak{m}'/x_0] \perp \mathfrak{q}'^{x_i} \ ,$$

i.e.,

$$\mathfrak{n}_i[\mathfrak{m}'/x_0] \perp \mathfrak{q}'$$

since $x_i \notin \mathrm{fv}(\mathfrak{q}'^{x_i})$. Thus also

$$\mathfrak{m}' \perp \mathfrak{q}'[\mathfrak{n}_i/x_0]$$

(since $(\!|\mathfrak{q}'[\mathfrak{n}_i[\mathfrak{m}'/x_0]/x_0]|\!) = \maltese$). We deduce by Lemma 6.2.2(2) that

$$\mathfrak{m}\restriction_{\wp} \perp \mathfrak{q}'[\mathfrak{n}_i/x_0] \ .$$

It follows that

$$\mathfrak{m} \perp \mathfrak{q}'[\mathfrak{n}_i/x_0] \ , \qquad \text{thus also} \qquad \mathfrak{n}_i[\mathfrak{m}/x_0] \perp \mathfrak{q}' \ .$$

By associativity we get

$$(\!|\mathfrak{n}_i[\mathfrak{m}/x_0]|\!) \perp \mathfrak{q}' \ .$$

Hence the result.

(2) $\Rightarrow$ (1) Let

$$\mathfrak{p} = x_0| \bullet \langle \mathfrak{n}_1, \mathfrak{n}_2 \rangle$$

where $\mathfrak{n}_1, \mathfrak{n}_2$ are such that for every $\mathfrak{m} \in \mathbf{N}_1^{\perp} \parr \mathbf{N}_2^{\perp}$ we have

$$(\!|\mathfrak{n}_1[\mathfrak{m}/x_0]|\!) \in \mathbf{N}_1 \qquad \text{and} \qquad (\!|\mathfrak{n}_2[\mathfrak{m}/x_0]|\!) \in \mathbf{N}_2 \ .$$

Let $\mathfrak{m} \in \mathbf{N}_1^{\perp} \parr \mathbf{N}_2^{\perp}$, and we show that

$$\mathfrak{p} \perp \mathfrak{m} \ .$$

By internal completeness there exists $\mathfrak{q} \in \mathbf{R}$ such that

$$\mathfrak{m}\restriction_{\wp} = \wp(x_1, x_2).\mathfrak{q} \ .$$

Let us write $\mathfrak{n}_1' = \mathfrak{n}_1[\mathfrak{m}/x_0]$ and $\mathfrak{n}_2' = \mathfrak{n}_2[\mathfrak{m}/x_0]$, we have

$$\mathfrak{p}[\mathfrak{m}/x_0] \rightsquigarrow \mathfrak{q}[\mathfrak{n}_1'/x_1, \mathfrak{n}_2'/x_2] \ ,$$

where $(\!|\mathfrak{n}_1'|\!) \in \mathbf{N}_1$ and $(\!|\mathfrak{n}_2'|\!) \in \mathbf{N}_2$ by hypothesis. Since $\mathfrak{q} \in \mathbf{R}$, by associativity we deduce

$$(\!|\mathfrak{q}[\mathfrak{n}_1'/x_1, \mathfrak{n}_2'/x_2]|\!) = (\!|\mathfrak{q}[(\!|\mathfrak{n}_1'|\!)/x_1, (\!|\mathfrak{n}_2'|\!)/x_2]|\!) = \maltese \ ,$$

thus

$$\mathfrak{p} \perp \mathfrak{m} \ .$$

Finally $\mathfrak{p} \in \mathbf{N}_1 \otimes \mathbf{N}_2$.

(2) $\Leftrightarrow$ (3) Suppose $\mathfrak{p}$ is of the form $\mathfrak{p} = x_0| \bullet \langle \mathfrak{n}_1, \mathfrak{n}_2 \rangle$. Using associativity, for every $i \in \{1, 2\}$ we have:

$$
\begin{array}{rl}
 & \text{for every } \mathfrak{m} \in \mathbf{N}_1^\perp \parr \mathbf{N}_2^\perp, (\!|\mathfrak{n}_i[\mathfrak{m}/x_0]|\!) \in \mathbf{N}_i \\
\Leftrightarrow & \text{for every } \mathfrak{m} \in \mathbf{N}_1^\perp \parr \mathbf{N}_2^\perp \text{ and every } \mathfrak{q} \in \mathbf{N}_i^\perp, \mathfrak{q} \perp (\!|\mathfrak{n}_i[\mathfrak{m}/x_0]|\!) \\
\Leftrightarrow & \text{for every } \mathfrak{m} \in \mathbf{N}_1^\perp \parr \mathbf{N}_2^\perp \text{ and every } \mathfrak{q} \in \mathbf{N}_i^\perp, \mathfrak{q} \perp \mathfrak{n}_i[\mathfrak{m}/x_0] \\
\Leftrightarrow & \text{for every } \mathfrak{m} \in \mathbf{N}_1^\perp \parr \mathbf{N}_2^\perp \text{ and every } \mathfrak{q} \in \mathbf{N}_i^\perp, (\!|\mathfrak{q}[\mathfrak{n}_i[\mathfrak{m}/x_0]/x_0]|\!) = \maltese \\
\Leftrightarrow & \text{for every } \mathfrak{m} \in \mathbf{N}_1^\perp \parr \mathbf{N}_2^\perp \text{ and every } \mathfrak{q} \in \mathbf{N}_i^\perp, \mathfrak{q}[\mathfrak{n}_i/x_0] \perp \mathfrak{m} \\
\Leftrightarrow & \text{for every } \mathfrak{m} \in \mathbf{N}_1^\perp \parr \mathbf{N}_2^\perp \text{ and every } \mathfrak{q} \in \mathbf{N}_i^\perp, (\!|\mathfrak{q}[\mathfrak{n}_i/x_0]|\!) \perp \mathfrak{m} \\
\Leftrightarrow & \text{for every } \mathfrak{q} \in \mathbf{N}_i^\perp, (\!|\mathfrak{q}[\mathfrak{n}_i/x_0]|\!) \in \mathbf{N}_1 \otimes \mathbf{N}_2 \ .
\end{array}
$$

□

6.3 Non-Linear Non-Deterministic Paths

This section introduces paths in non-linear ludics. In particular, paths are non-linear, i.e., a variable can appear free several times. After describing an interaction as a set of paths (an n-path), we give ideas on what visitable paths of behaviours constructed by logical connectives may look like.

In a previous work [Pav14] we investigated the notion of path in non-linear but *deterministic* ludics. Then, we needed a condition of *internal coherence* in the definition of path; this condition had to do with uniformity (the fact that, when asked the same question, the answer given is the same). With non-determinism, this condition is no longer required. But an interaction is not described by a single interaction path anymore, it is now associated to a set of paths corresponding to all the possible choices.

6.3.a Views and Paths

We consider the same notion of (located) **action** as in the previous chapters (Definition 1.2.1). Thus we can still represent cut-free designs as trees/forests, but now there can be several positive nodes as roots, or above a single negative node. We will sometimes need to put annotations in order to differentiate between positive actions with the same name. The idea is the following: given a positive design $\bigwedge\{\mathfrak{s}_i : i \in I\}$, for each $i \in I$ the first positive action of $\mathfrak{s}_i$ receives the annotation i. This way, all the positive proper actions in a design receive an annotation, and we can for example differentiate between the views

$$b_{x_0}(x) \ \ x|\overline{a}\langle y\rangle_1 \qquad \text{and} \qquad b_{x_0}(x) \ \ x|\overline{a}\langle y\rangle_2$$

of design $\mathfrak{n}_1$ from Figure 12 (page 125). But we avoid writing the annotations when unnecessary.

Definition 6.3.1 (*Ajn-sequence*)

- An **ajn-sequence** is a sequence of actions satisfying all the conditions from Definition 1.2.5 except for *Linearity*.
- An **annotated ajn-sequence** is a pair (s, u) where s is an ajn-sequence and u is a sequence of *indexes* of length the number of proper positive actions in s. If κ_n^+ is the n-th positive action of s, then the n-th element of u is called the **annotation** of κ_n^+ in s.

Definition 6.3.2 (*View*)

- A **view** is an annotated ajn-sequence $(\mathbb{v}, u)$ such that each negative action of $\mathbb{v}$ which is not the first one is justified by the immediate previous action.
- Given a cut-free design $\mathfrak{d}$, the **views of** $\mathfrak{d}$, noted $\mathbb{V}[\mathfrak{d}]$, is the set of views defined recursively as in Definition 1.2.6, adding the following case for positive disjunction:
$$\mathbb{V}[\bigwedge\{\mathfrak{s}_i : i \in I\}] = \bigcup_{i\in I} \mathbb{V}[\mathfrak{s}_i] \qquad \text{if} \qquad I \neq \emptyset$$
and where each proper positive action is annotated with the index i associated to the corresponding predesign.
- The **view of** an annotated ajn-sequence (s, u) is $\ulcorner (s, u) \urcorner = (\ulcorner s \urcorner, u')$ where:
  - $\ulcorner s \urcorner$ is defined as the view of an aj-sequence (Definition 1.2.7),
  - u' is the subsequence of u containing only the annotations of the proper positive actions appearing in $\ulcorner s \urcorner$.

Notation

For annotated ajn-sequences, we may abusively write s instead of (s, u), and $\ulcorner s \urcorner$ instead of $\ulcorner (s, u) \urcorner$, the annotations being implicit.

Definition 6.3.3 (*Path*)

- A **path** is a positive-ended (or empty) P-visible and O-visible ajn-sequence.
- Given a cut-free design $\mathfrak{d}$, a path s is a **path of** $\mathfrak{d}$ if there exists a sequence u such that for all prefix s' of s, $\ulcorner (s', u) \urcorner$ is a view of $\mathfrak{d}$.

Example 6.3.4

Consider the designs of Figure 12. Designs $\mathfrak{n}_1$ and $\mathfrak{n}_2$ have different views, even up to re-indexing of positive actions. This implies in particular that the path

$$s \quad = \quad b_{x_0}(x) \quad x|\overline{a}\langle y\rangle \quad c_y(z) \quad z|\overline{d}\langle\rangle \quad c_y(z) \quad z|\overline{e}\langle\rangle$$

is a path of $\mathfrak{n}_2$ but not of $\mathfrak{n}_1$, indeed: we need to choose between annotating $x|\overline{a}\langle y\rangle$ with 1 or 2, thus we can never visit both branches of $\mathfrak{n}_1$. As a consequence, we have $\mathfrak{n}_1 \perp \mathfrak{p}$ but $\mathfrak{n}_2 \not\perp \mathfrak{p}$; a more precise explanation will be given in the next subsection (Example 6.3.10).

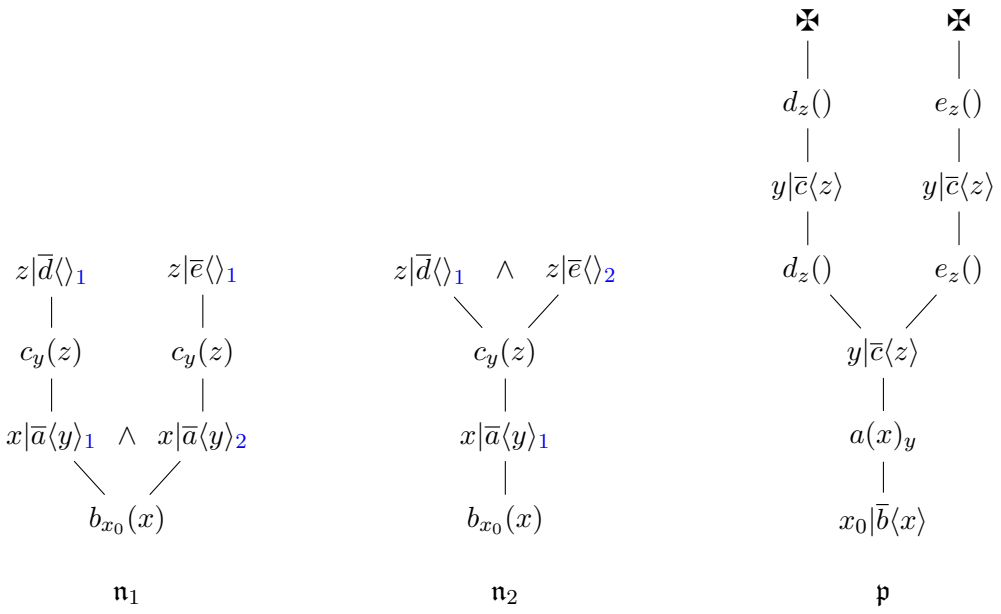


Figure 12: N-paths and orthogonality

This example also shows that a non-linear path can visit several times the same action in a design: here the negative action $c_y(z)$ of design $\mathfrak{n}_2$ appears twice in s.

Definition 6.3.5 (*Interaction path*)

Let $\mathfrak{d}$ and $\mathfrak{e}$ be cut-free atomic designs of opposite polarities. An **interaction path** of $\mathfrak{d}$ with $\mathfrak{e}$ is a path s of $\mathfrak{d}$ such that $\tilde{s}$ is a path of $\mathfrak{e}$.

Unlike the linear case, the existence of an interaction path between $\mathfrak{d}$ and $\mathfrak{e}$ does not ensure their orthogonality. The convergence of interaction now requires that we have a set of such interaction paths, coherent in a sense. Such a set is called an *n-path* and is described in the next subsection.

6.3.b N-Paths

Given two cut-free atomic designs of opposite polarities $\mathfrak{d}$ and $\mathfrak{e}$, write $\langle \mathfrak{d} \leftarrow \mathfrak{e} \rangle$ for the set of interaction paths of $\mathfrak{d}$ with $\mathfrak{e}$, and remark that $\langle \mathfrak{e} \leftarrow \mathfrak{d} \rangle = \widetilde{\langle \mathfrak{d} \leftarrow \mathfrak{e} \rangle}$. After defining n-paths, we conjecture that $\mathfrak{d} \perp \mathfrak{e}$ if and only if the sets $\langle \mathfrak{d} \leftarrow \mathfrak{e} \rangle$ and $\langle \mathfrak{e} \leftarrow \mathfrak{d} \rangle$ are respectively n-paths of $\mathfrak{d}$ and $\mathfrak{e}$.

Notation

In the following, given a set $\mathbb{S}$ of sequences, let us denote by $\mathrm{pref}(\mathbb{S})$ the set of the prefixes of sequences of $\mathbb{S}$.

Definition 6.3.6 (*N-path*)

A **non-deterministic path** or **n-path** is a non-empty set $\mathbb{S}$ of paths satisfying the following conditions:
(P-universality) For all prefixes $s\kappa^+, t \in \mathrm{pref}(\mathbb{S})$, if $\ulcorner s \urcorner = \ulcorner t \urcorner$ then $t\kappa^+ \in \mathrm{pref}(\mathbb{S})$;
(O-universality) For all prefixes $s\kappa^-, t \in \mathrm{pref}(\mathbb{S})$, if $\llcorner s \lrcorner = \llcorner t \lrcorner$ then $t\kappa^- \in \mathrm{pref}(\mathbb{S})$.

The idea, formalised in Conjecture 6.3.9, is that an n-path corresponds to a (convergent) interaction. A path in it is the interaction trace for a particular choice made at each non-deterministic branching. Thus an n-path containing a single path is for a deterministic interaction.

The P-universality condition ensures that if, when reaching a conjunction, the n-path visits a positive action κ^+ of this conjunction, then each time it reaches the same conjunction it must be able to visit κ^+. This requirement comes from the universality of our non-determinism: interaction must systematically visit all the positive actions of a conjunction it reaches. O-universality corresponds to the same requirement for the orthogonal.

Remark 6.3.7

If $\mathbb{S}$ is an n-path then its dual $\widetilde{\mathbb{S}} = \{\widetilde{s} \mid s \in \mathbb{S}\}$ is an n-path.

Definition 6.3.8 (*N-path of a design*)

Given a cut-free design $\mathfrak{d}$ and an n-path $\mathbb{S}$, $\mathbb{S}$ is an **n-path of** $\mathfrak{d}$ if all the paths in $\mathbb{S}$ are paths of $\mathfrak{d}$ and the following condition is satisfied:

$$\text{For every path } s\kappa^+ \text{ of } \mathfrak{d}, \text{ if there exists } \kappa'^+ \text{ such that } s\kappa'^+ \in \mathrm{pref}(\mathbb{S}) \text{ then } s\kappa^+ \in \mathrm{pref}(\mathbb{S}).$$

Again, the condition in this definition expresses the universality of non-determinism: it ensures that if, at some point of the interaction, at least one positive action in a conjunction of $\mathfrak{d}$ is visited by the n-path, then all the positive actions of this conjunction are.

The purpose of n-paths would be to prove the following conjecture, in order to establish the exact correspondence between orthogonality and n-paths.

Conjecture 6.3.9

Let $\mathfrak{d}$ and $\mathfrak{e}$ be cut-free atomic designs of opposite polarities. $\mathfrak{d} \perp \mathfrak{e}$ if and only if there exists an n-path $\mathbb{S}$ of $\mathfrak{d}$ such that $\widetilde{\mathbb{S}}$ is an n-path of $\mathfrak{e}$; moreover, this n-path is $\mathbb{S} = \langle \mathfrak{d} \leftarrow \mathfrak{e} \rangle$.

We are strongly convinced that this holds, but we have not proved it yet. It would probably require to adapt all the results about multi-designs (Chapter 2) to the non-linear setting, indeed: the linear version of this conjecture (Proposition 1.2.11) relies on such results. Let us however give the following example as an intuition.

Example 6.3.10

Going back to the designs of Figure 12, we have $\mathfrak{n}_1 \perp \mathfrak{p}$ with

$$\langle \mathfrak{n}_1 \leftarrow \mathfrak{p} \rangle = \quad \{ \quad b_{x_0}(x) \quad x|\overline{a}\langle y\rangle \quad c_y(z) \quad z|\overline{d}\langle\rangle \quad c_y(z) \quad z|\overline{d}\langle\rangle \quad , \\ b_{x_0}(x) \quad x|\overline{a}\langle y\rangle \quad c_y(z) \quad z|\overline{e}\langle\rangle \quad c_y(z) \quad z|\overline{e}\langle\rangle \quad \} \ .$$

We can check that $\langle \mathfrak{n}_1 \leftarrow \mathfrak{p} \rangle$ is an n-path of $\mathfrak{n}_1$ while $\widetilde{\langle \mathfrak{n}_1 \leftarrow \mathfrak{p} \rangle} = \langle \mathfrak{p} \leftarrow \mathfrak{n}_1 \rangle$ is an n-path of $\mathfrak{p}$. On the other hand we have $\mathfrak{n}_2 \not\perp \mathfrak{p}$ and

$$\langle \mathfrak{n}_2 \leftarrow \mathfrak{p} \rangle = \langle \mathfrak{n}_1 \leftarrow \mathfrak{p} \rangle$$

where $\langle \mathfrak{n}_2 \leftarrow \mathfrak{p} \rangle$ is indeed an n-path but not an n-path *of* $\mathfrak{n}_2$; the problem is that:

$$\begin{array}{rll} \text{though} & b_{x_0}(x)\ x|\overline{a}\langle y\rangle\ c_y(z)\ z|\overline{d}\langle\rangle\ c_y(z)\ z|\overline{e}\langle\rangle & \text{is a path of } \mathfrak{n}_2 \\ \text{and} & b_{x_0}(x)\ x|\overline{a}\langle y\rangle\ c_y(z)\ z|\overline{d}\langle\rangle\ c_y(z)\ z|\overline{d}\langle\rangle & \in \mathrm{pref}(\langle \mathfrak{n}_2 \leftarrow \mathfrak{p}\rangle) \\ \text{we have} & b_{x_0}(x)\ x|\overline{a}\langle y\rangle\ c_y(z)\ z|\overline{d}\langle\rangle\ c_y(z)\ z|\overline{e}\langle\rangle & \notin \mathrm{pref}(\langle \mathfrak{n}_2 \leftarrow \mathfrak{p}\rangle) \end{array}$$

which contradicts the condition in Definition 6.3.8.

6.4 Outlook

We now present some possible future directions of research.

Visitable paths and connectives. There is still work to undertake concerning the relation between paths and logical connectives in a non-linear setting. In particular, as for the linear case (Section 3.2 in Chapter 3), we would like to prove the form of visitable paths of behaviours constructed by connectives. Those visitable paths are in particular non-linear, thus some parts can be repeated many times. We conjecture the way they may look like.

Definition 6.4.1 (*Visitable path*)

A path s is **visitable** in an atomic behaviour $\mathbf{B}$ if there exist $\mathfrak{d} \in \mathbf{B}$ and $\mathfrak{e} \in \mathbf{B}^\perp$ such that $s \in \langle \mathfrak{d} \leftarrow \mathfrak{e} \rangle$. The set of visitable paths of $\mathbf{B}$ is written $V_{\mathbf{B}}$.

Shuffle $\sqcup\!\sqcup$ and **anti-shuffle** $\sqcap$ on non-linear paths are defined the same way as for linear ones (Definition 1.3.1); but $s \sqcup\!\sqcup t$ is now a larger set than in the linear case, since the notion of path has been extended. We also consider unary operations $\bigsqcup\!\bigsqcup$ and $\bigsqcap$ on a set of path V, defined by:

$$\bigsqcup\!\bigsqcup V = \bigcup_{k\in\mathbb{N}} \{s_1 \sqcup\!\sqcup \ldots \sqcup\!\sqcup s_k \mid s_i \in V\} \qquad \text{and} \qquad \bigsqcap V = \bigcup_{k\in\mathbb{N}} \{s_1 \sqcap \ldots \sqcap s_k \mid s_i \in V\} \ .$$

In order to prove the conjecture below, we should redefine regularity, thus also incarnation: this is not difficult. Similarly to the visitable paths of the tensor (Propositions 3.2.6 and 3.2.8), we guess that regularity is needed to have such simple formulations.

Conjecture 6.4.2

Let $\mathbf{M}, \mathbf{N}$ be negative behaviours, let $\mathbf{P}$ be a positive behaviour, and suppose these behaviours are regular. We have:

$$\begin{aligned} V_{\downarrow\mathbf{N}} &= \prod(\kappa_{\blacktriangledown} V_{\mathbf{N}}) \ , \\ V_{\uparrow\mathbf{P}} &= \bigsqcup(\kappa_{\blacktriangle} V_{\mathbf{P}}) \ , \\ V_{\mathbf{M}\oplus\mathbf{N}} &= \prod(\kappa_{\iota_1} V_{\mathbf{M}}) \sqcap \prod(\kappa_{\iota_2} V_{\mathbf{N}}) \ , \\ V_{\mathbf{M}\otimes\mathbf{N}} &= \prod(\kappa_{\bullet}(V_{\mathbf{M}} \sqcup V_{\mathbf{N}})) \ . \end{aligned}$$

Compared to the linear version of visitable paths for connectives, the non-linear ones combine the paths with anti-shuffles, which corresponds to the possibility of repeating a location. We believe that, similarly to the linear case, the proof of this conjecture relies in particular on:

- internal completeness for the negative connectives (Theorem 6.2.4) and the weaker alternative for the positive connectives (Propositions 6.2.6, 6.2.7 and 6.2.8),
- a non-linear equivalent of the associativity for interaction paths (Proposition 2.2.12), which would require that we adapt many results concerning multi-designs to non-linear ludics.

Regularity and purity. Towards a study of the types of non-linear ludics, as we did in the linear case, we could use the form of the visitable paths to deduce regularity and purity of behaviours constructed by connectives.

Conjecture 6.4.3

In non-linear ludics:

- *regularity is stable under $\downarrow, \uparrow, \oplus, \otimes, \parr, \&, \multimap$;*
- *purity is stable under $\downarrow, \uparrow, \oplus, \otimes$.*

Recall that Fouqueré and Quatrini [FQ16] proved, in linear ludics, that finite regular behaviours correspond exactly to the interpretation of MALL formulas (Proposition 4.3.10). Following this idea, we believe that it is possible to capture LLP in non-linear ludics.

Conjecture 6.4.4

In non-linear ludics, a behaviour is the denotation of a formula of LLP if and only if it is regular and finite.

Separation vs. weak separation. It has been observed [BF11, BT10b, Mau04] that *separation*, an analytical theorem of linear ludics which is the analogue of Böhm's theorem in λ-calculus, does not hold in the non-linear setting, even if we restrict to deterministic designs. In other words, there exist designs $\mathfrak{d}_1 \neq \mathfrak{d}_2$ that are not separable, in the sense that every design $\mathfrak{e}$ is orthogonal either to both $\mathfrak{d}_1$ and $\mathfrak{d}_2$ or to none of them. The designs

$\mathfrak{p}_i$ of Figure 10 (page 117) are an example of non-separable designs; in fact, it is because we lack separation that we do not have true internal completeness for positive connectives.

But we might have an alternative, *weak separation*, that differentiates designs according to interaction paths.

Conjecture 6.4.5 (*Weak separation*)

Let $\mathfrak{d}_1$ and $\mathfrak{d}_2$ be cut-free atomic designs of same polarity such that $\mathfrak{d}_1 \neq \mathfrak{d}_2$. There exists a cut-free atomic design $\mathfrak{e}$ such that $\langle \mathfrak{d}_1 \leftarrow \mathfrak{e} \rangle \neq \langle \mathfrak{d}_2 \leftarrow \mathfrak{e} \rangle$.

Conclusion

The research conducted in this thesis has consisted in exploring ludics to highlight and deconstruct some behaviours with an interesting computational or logical meaning. More precisely, we focused on the behaviours representing data types and functional types, and on the behaviours of non-linear ludics.

A first step has been to gather all the necessary material into the framework of computational ludics, where designs are terms. The properties of behaviours with respect to interaction can be characterised by their visitable paths, and we have followed this approach to analyse the behaviours we were interested in. We have expressed the visitable paths of the behaviours constructed by logical connectives in a compositional way. This, together with internal completeness, has led us to prove that these behaviours were regular – the key notion for the characterisation of MALL in ludics – and those constructed without $\multimap$ were pure – that is, type safe.

Interpreting the inductive data types in ludics requires that we consider a least fixed point operator in addition to the usual connectives. Adopting a constructive approach, we have provided an internal completeness result for fixed points, unveiling the structure of the behaviours corresponding to inductive types. As a consequence, we could show that data behaviours were regular and pure. But the behaviours interpreting types of functions taking functions as argument are impure: a bad interruption of the execution can happen in the case of a ludics program which is not strictly functional (in the sense of functional programming). However, under regularity assumption, well-bracketedness keeps from getting such errors.

The work achieved so far lies mainly in linear ludics, but we have also started to study non-linearity, which is interesting from both the "proof" and the "program" perspective: indeed, it allows to consider proofs that can use their hypotheses several times, as well as programs that can call their arguments several times. In non-linear ludics, internal completeness does not hold anymore for the positive connectives, but we could recover a weaker form. We have also described how the paths could be adapted to describe an interaction in a setting that is both non-linear and non-deterministic, with the notion of n-path.

Some ideas for future work then arise naturally:

- We plan to extend our study of data types with greatest fixed points $\nu X.A$, i.e., coinduction (discussed in Section 4.4).
- Getting a real characterisation of μMALL in ludics should be possible, by proving

that a behaviour is regular if and only if it is the denotation of a μMALL formula (Conjecture 4.3.11).
- It would be interesting to consider fixed points together with functional types (discussed in Remark 5.1.3).
- Another future goal is to generalise the study of data and functions to non-linear ludics. This would require to lift the results on multi-designs to the non-linear setting, in order to prove that the existence of an n-path matches orthogonality (Conjecture 6.3.9) and to make explicit the form of the visitable paths for the connectives (Conjecture 6.4.2). From this, we believe we could deduce regularity and purity (Conjecture 6.4.3).
- In non-linear ludics, we would also like to characterise the behaviours corresponding to LLP as the regular and finite ones (Conjecture 6.4.4)
- Finally, we believe that we can recover a weaker form of separation in non-linear ludics (Conjecture 6.4.5), which may have some interesting outcomes.

More generally, studying non-regularity in ludics could also be interesting. Indeed, our exploration has only led to regular behaviours for the moment. The non-regular ones have more surprising interactions, and could model logical or programming features with a notion of orientation. For example, we believe that non-regular behaviours could represent processes executions in concurrent computing, with notions like synchronisation, deadlocks, etc. On a logical side, non-regularity can lead to new *oriented* logical connectives, for example the oriented tensor studied by Fouqueré and Quatrini [FQ16].

✠

Bibliography

[AJM00] Samson Abramsky, Radha Jagadeesan, and Pasquale Malacaria. Full Abstraction for PCF. *Information and Computation*, 163(2):409 – 470, 2000.

[AM99] Samson Abramsky and Guy McCusker. Game semantics. *Computational Logic: Proceedings of the 1997 Marktoberdorf Summer School*, pages 1–56, 1999.

[And92] Jean-Marc Andreoli. Logic Programming with Focusing Proofs in Linear Logic. *Journal of Logic and Computation*, 2:297–347, 1992.

[Bae12] David Baelde. Least and greatest fixed points in linear logic. *ACM Transactions on Computational Logic*, 13(1), January 2012.

[BDS15] David Baelde, Amina Doumane, and Alexis Saurin. Least and greatest fixed points in ludics. In *Proceedings of the 24th Annual EACSL Conference on Computer Science Logic (CSL'15)*, pages 549–566, September 2015.

[BF11] Michele Basaldella and Claudia Faggian. Ludics with repetitions (exponentials, interactive types and completeness). *Logical Methods in Computer Science*, 7(2), 2011.

[BM07] David Baelde and Dale Miller. Least and greatest fixed points in linear logic. In *Proceedings of the 14th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'07)*, pages 92–106, October 2007.

[BS98] Richard Blute and Philip J. Scott. The Shuffle Hopf Algebra and Noncommutative Full Completeness. *Journal of Symbolic Logic*, 63(4):1413–1436, 1998.

[BT10a] Michele Basaldella and Kazushige Terui. Infinitary Completeness in Ludics. In *Proceedings of the 25th Annual IEEE Symposium on Logic in Computer Science (LICS'10)*, pages 294–303, 2010.

[BT10b] Michele Basaldella and Kazushige Terui. On the meaning of logical completeness. *Logical Methods in Computer Science*, 6(4), 2010.

[CH07] Pierre-Louis Curien and Hugo Herbelin. Abstract machines for dialogue games. *CoRR*, abs/0706.2544, 2007.

[Chu41] Alonzo Church. *The Calculi of Lambda-Conversion.* Princeton University Press, 1941.

[Cla09] Pierre Clairambault. Least and greatest fixpoints in game semantics. In *Proceedings of the 12th International Conference on Foundations of Software Science and Computational Structures (FOSSACS'09)*, pages 16–31, 2009.

[Cur34] Haskell B. Curry. Functionality in Combinatory Logic. In *Proceedings of the National Academy of Sciences of the United States of America*, volume 20, pages 584–590, November 1934.

[Cur05] Pierre-Louis Curien. Introduction to linear logic and ludics, part II. *CoRR*, abs/cs/0501039, 2005.

[ER08] Thomas Ehrhard and Laurent Regnier. Uniformity and the Taylor expansion of ordinary lambda-terms. *Theoretical Computer Science*, 403(2-3):347–372, 2008.

[Fag01] Claudia Faggian. *On the dynamics of ludics: a study of interaction.* PhD thesis, Université Aix–Marseille II, 2001.

[FH02] Claudia Faggian and Martin Hyland. Designs, disputes and strategies. In *Proceedings of the 11th Annual EACSL Conference on Computer Science Logic (CSL'02)*, pages 442–457, September 2002.

[FQ13] Christophe Fouqueré and Myriam Quatrini. Incarnation in ludics and maximal cliques of paths. *Logical Methods in Computer Sciences*, 9(4), 2013.

[FQ16] Christophe Fouqueré and Myriam Quatrini. Study of behaviours via visitable paths. *CoRR*, abs/1403.3772v3, 2016.

[Gen34] Gerhard Gentzen. Untersuchungen über das logische Schließen I. *Mathematische Zeitschrift*, 39:176–210, 1934.

[Gen35] Gerhard Gentzen. Untersuchungen über das logische Schließen II. *Mathematische Zeitschrift*, 39:405–431, 1935.

[Gir87] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50(1):1–102, January 1987.

[Gir89] Jean-Yves Girard. Geometry of interaction. I. Interpretation of system F. In *Proceedings of the Logic Colloquium '88*, pages 221–260, 1989.

[Gir01] 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.

[Gir06] Jean-Yves Girard. *Le point aveugle: cours de logique*, volume 1. Hermann, 2006.

[HMU07] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computation.* Pearson Addison-Wesley, Boston, 3rd edition, 2007.

[HO00] Martin Hyland and Luke Ong. On full abstraction for PCF: I, II, and III. *Information and Computation*, 163(2):285–408, 2000.

[How80] William A. Howard. The Formulas-as-types Notion of Construction. In *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus, and Formalism*, pages 479–490. Academic Press, 1980. Original paper manuscript from 1969.

[HS03] Martin Hyland and Andrea Schalk. Glueing and orthogonality for models of linear logic. *Theoretical Computer Science*, 294(1-2):183–231, 2003.

[Kle52] Stephen C. Kleene. *Introduction to Metamathematics.* Bibliotheca Mathematica. Wolters-Noordhoff, 1952.

[Kri09] Jean-Louis Krivine. Realizability in classical logic. *Panoramas et synthèses*, 27:197–229, 2009.

[Lau02] Olivier Laurent. *Étude de la polarisation en logique.* PhD thesis, Université Aix–Marseille II, March 2002.

[Lau04] Olivier Laurent. Polarized games. *Annals of Pure and Applied Logic*, 130(1-3):79–123, 2004.

[Mau04] François Maurel. *Un cadre quantitatif pour la Ludique.* PhD thesis, Université Paris-Diderot – Paris VII, 2004.

[Maz17] Damiano Mazza. Infinitary Affine Proofs. *Mathematical Structures in Computer Science*, 27(5):581–602, 2017.

[McC98] Guy McCusker. *Games and full abstraction for a functional metalanguage with recursive types.* CPHC/BCS distinguished dissertations. Springer, 1998.

[Mil78] Robin Milner. A theory of type polymorphism in programming. *Journal of Computer and System Sciences*, 17:348–375, 1978.

[MV05] Paul-André Melliès and Jerome Vouillon. Recursive polymorphic types and parametricity in an operational framework. In *Proceedings of the 20th Annual IEEE Symposium on Logic in Computer Science (LICS'05)*, pages 82–91, June 2005.

[Nic94] Hanno Nickau. Hereditarily Sequential Functionals. In *Proceedings of the 3rd International Symposium on Logical Foundations of Computer Science (LFCS'94)*, pages 253–264, July 1994.

[Pav14] Alice Pavaux. Chemins en C-Ludique. Master's thesis, 2014.

[Pav17] Alice Pavaux. Inductive and Functional Types in Ludics. In *Proceedings of the 26th EACSL Annual Conference on Computer Science Logic (CSL'17)*, pages 34:1–34:20, August 2017.

[Rog67] Hartley Rogers. *Theory of Recursive Functions and Effective Computability*. McGraw-Hill, 1967.

[San09] Davide Sangiorgi. On the origins of bisimulation and coinduction. *ACM Transactions on Programming Languages and Systems*, 31(4):15:1–15:41, May 2009.

[Sau08] Alexis Saurin. Towards Ludics Programming: Interactive Proof Search. In *Proceedings of the 24th International Conference on Logic Programming (ICLP'08)*, pages 253–268, December 2008.

[Sir15] Eugenia Sironi. *Types in Ludics*. PhD thesis, Université d'Aix–Marseille, January 2015.

[SS71] Dana Scott and Christopher Strachey. Toward a Mathematical Semantics for Computer Languages. Oxford programming Research Group Technical Monograph PRG-6, 1971.

[Tar55] Alfred Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific journal of Mathematics*, 5(2):285–309, 1955.

[Ter11] Kazushige Terui. Computational ludics. *Theoretical Computer Science*, 412(20):2048–2071, 2011.

[Tur36] Alan M. Turing. On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 2(42):230–265, 1936.

Inductive, Functional and Non-Linear Types in Ludics

Abstract

This thesis investigates the types of ludics. Within the context of the Curry–Howard correspondence, ludics is a framework in which the dynamic aspects of both logic and programming can be studied. The basic objects, called designs, are untyped infinitary proofs that can also be seen as strategies from the perspective of game semantics, and a type or behaviour is a set of designs well-behaved with respect to interaction. We are interested in observing the interactive properties of behaviours. Our attention is particularly focused on behaviours representing the types of data and functions, and on non-linear behaviours which allow the duplication of objects. A new internal completeness result for infinite unions unveils the structure of inductive data types. Thanks to an analysis of the visitable paths, i.e., the possible execution traces, we prove that inductive and functional behaviours are regular, paving the way for a characterisation of μMALL in ludics. We also show that a functional behaviour is pure, a property ensuring the safety of typing, if and only if it is not a type of functions taking functions as argument. Finally, we set the bases for a precise study of non-linearity in ludics by recovering a form of internal completeness and discussing the visitable paths.

Types inductifs, fonctionnels et non-linéaires en ludique

Résumé

Cette thèse est consacrée à une exploration des types de la ludique. S'inscrivant dans un contexte marqué par la correspondance de Curry–Howard, la ludique est un cadre permettant d'étudier l'aspect dynamique de la logique et de la programmation. Les objets de base, appelés desseins, sont des preuves infinitaires non-typées qui peuvent également être vues comme des stratégies sous l'angle de la sémantique des jeux, et un type ou comportement est un ensemble de desseins se conduisant de la même manière du point de vue de l'interaction. On s'intéresse aux propriétés interactives des comportements. Notre attention se porte en particulier sur les comportements représentant les types de données et de fonctions, et sur les comportements non-linéaires qui permettent la duplication d'objets. Un nouveau résultat de complétude interne pour les unions infinies dévoile la structure des types de données inductifs. Grâce à une analyse des chemins visitables, c'est-à-dire des possibles traces d'exécution, on prouve que les comportements inductifs et fonctionnels sont réguliers, ouvrant la voie pour une caractérisation de μMALL en ludique. On montre également qu'un comportement fonctionnel est pur, une propriété garantissant la sûreté du typage, si et seulement si ce n'est pas un type de fonctions prenant des fonctions en argument. Enfin, on pose les bases d'une étude précise de la non-linéarité en ludique en retrouvant une forme de complétude interne et en discutant des chemins visitables.