Making Concurrency Functional

GLYNN WINSKEL, Edinburgh Research Centre, Central Software Institute, Huawei;
University of Strathclyde, Glasgow, United Kingdom

The article bridges between two major paradigms in computation, the *functional*, at basis computation from input to output, and the *interactive*, where computation reacts to its environment while underway. Central to any compositional theory of interaction is the dichotomy between a system and its environment. Concurrent games and strategies address the dichotomy in fine detail, very locally, in a distributed fashion, through distinctions between Player moves (events of the system) and Opponent moves (those of the environment). A functional approach has to handle the dichotomy more ingeniously, via its blunter distinction between input and output. This has led to a variety of functional approaches, specialised to particular interactive demands. Through concurrent games we can see what separates and connects the differing paradigms, and show how:
• to lift functions to strategies; how to turn functional dependency to causal dependency.
• several paradigms of functional programming and logic arise naturally as full subcategories of concurrent games, including stable domain theory; nondeterministic dataflow; geometry of interaction; the dialectica interpretation; lenses and optics, and their extensions to containers in dependent lenses and optics.
• the enrichments of strategies (*e.g.* to probabilistic, quantum or real-number computation) specialise to the functional cases.

1 INTRODUCTION

The view of computation as functions is at the very foundation of computer science: the Church-Turing thesis expresses the coincidence of different notions of computable function; programming with higher-order functions is now taken for granted.

In contrast the view of computation as interaction is more recent and less settled, and often obscured by adherence to one syntax or another, perhaps each with its own mechanism of interaction. Instead our approach is maths-driven. Its tools are those of distributed/concurrent games and strategies [53], a causal model which allows for highly distributed interaction. Concurrent games and strategies are built on the mathematical foundations of categories of models for interaction [65], chiefly on the central model of event structures [57].[1]

Whereas the basic mechanism of interaction of functions is clear—ultimately by function composition—a functional approach can struggle with finding quite the right way to approach computation which isn't simply from input to output. The literature includes approaches via lenses, optics, combs, containers, dependent lenses, open games and learners [2, 14, 25–27, 46, 50]. The difficulties are compounded by enrichments to, say, probabilistic, quantum or real-number computation.

In functional approaches new patterns of interaction are often achieved by extending the usual input/output of functions with extra parameters to permit exchanges with the environment while computation is underway; the environment may comprise another similar parameterised function. But the types of functions tend only to give a static, rather rigid, partial picture of the dynamics of interaction. This handicaps the expression of and search for more complicated patterns of interaction within functional languages: for instance, patterns of interaction that may change over time, perhaps with one pattern of interaction replacing another, or perhaps being chosen nondeterministically or probabilistically. And if we are to allow very general interactions how are we to avoid functional loops which may not be sensible for the functions of interest?

By adopting a model which addresses interaction from the outset, we can better understand and explore the space of possible interactions, functional or otherwise. Concurrent games and

---

[1]A core language for concurrent strategies derives from the mathematical structure, although we shall only glimpse it here in Section 4.7: it is *higher-order* and an interesting hybrid of *dataflow*, *cf.* TensorFlow [1], *concurrent process calculi*, *cf.* CSP, CCS and *Session Types* [9, 13, 41].

strategies provide a way to describe and orchestrate temporal patterns of interaction between functions, their fine-grained dependencies and dynamic linkage—Sections 6 and 7. They support enrichments to strategies for probabilistic, quantum and real-number computation.

This article bridges between the two paradigms of computation, the functional and the interactive. In broad terms it shows:

- How to convert a general class of functions to concurrent strategies—Section 5; this helps in the programming of strategies via functional techniques and is of potential further use in describing sub(bi)categories of strategies through functions.
- How in many cases we can describe concurrent strategies as interacting patterns of functions; it reveals many paradigms in functional programming arise as full subcategories associated with special cases of concurrent games; in these cases composition of strategies can be described via simpler function composition—Section 6.
- How concurrent strategies enriched in a symmetric monoidal category $\mathcal{M}$ determine interacting patterns of "functions" (maps in $\mathcal{M}$) and how these compose through the composition of strategies; in this sense a sub(bi)category of strategies determines its own functional paradigm. This can be used to systematise the way we explore interaction between functions—Section 7.

Amplifying the second point above, it was a surprise to the author how neatly and automatically many functional paradigms arise simply by specialising to full subcategories of concurrent games. For example:

- We shall see how by restricting to deterministic strategies between concurrent games where all moves are Player moves we rediscover *stable functions* and Berry's stable domain theory, of which Girard's qualitative domains and coherence spaces are special cases. For such restricted games, general, possibly nondeterministic, strategies correspond to *stable spans,* a model discovered and rediscovered in compositional accounts of nondeterministic dataflow.
- Only marginally more complicated than those purely Player games are games which consist of two parallel components, one a purely Player game and the other with purely Opponent moves. Strategies between such games yield models for *Geometry of Interaction* built on stable functions and stable spans [3, 4, 6, 28].
- Adjoining winning conditions and imperfect information to these games, so Opponent can see the moves of Player but not the converse, we recover a *dialectica category* [21], so Gödel's dialectica interpretation [7], from deterministic strategies. We obtain from Gödel's work an interpretation of proofs in first-order arithmetic as winning strategies. Dialectica categories, studied by Valeria de Paiva in her Cambridge PhD [21], mark an early occurrence of *lenses* used in functional programming, where they were invented independently to make composable local changes on data-structures [26, 46].
- The newer paradigm of optics appears in characterising arbitrary, not just deterministic, strategies between dialectica games and when we move to more general container games, associated with container types [2]. Deterministic strategies between container games amount to *dependent lenses* and nondeterministic strategies to a form of *dependent optics*. The definition of dependent optic is derived as a characterisation of general strategies between container games; it appears to be new [30].

After the basics on event structures, the tools of stable families, and concurrent strategies, the new contribution comes in three parts which can roughly be described as: how to describe strategies by functions; how to describe functions and functional paradigms by strategies; and, how enriched strategies describe interacting patterns of functions. The first, rather technical part, Section 5, introduces a powerful method for lifting a very broad class of functions to strategies, turning functional into causal

dependency. It makes essential use of stable families and the Scott order intrinsic to a concurrent game. The second part, Section 6, concerns how causal dependency determines functional dependency, and shows how many paradigms discovered in making functions interactive arise as subcategories of concurrent games. The final, much shorter, third part, Section 7, shows how to enrich strategies in a symmetric monoidal category. An enriched strategy imposes a dynamic pattern of interaction between arrows in the monoidal category; the pattern of interaction has the form of an event structure. Such patterns of interaction compose well and won't contain loops of functional dependency because they are determined by strategies.

2 EVENT STRUCTURES

An *event structure* [57] comprises $(E, \leq, \mathrm{Con})$, consisting of a set E of *events* which are partially ordered by $\leq$, the *causal dependency relation*, and a nonempty *consistency* relation Con consisting of finite subsets of E. The relation $e' \leq e$ expresses that event e causally depends on the previous occurrence of event e'; the consistency relation, those events which may occur together. We insist that the partial order is *finitary*, *i.e.*

- $[e] := \{e' \mid e' \leq e\}$ is finite for all $e \in E$,

and that consistency satisfies

- $\{e\} \in \mathrm{Con}$ for all $e \in E$,
- $Y \subseteq X \in \mathrm{Con}$ implies $Y \in \mathrm{Con}$, and
- $X \in \mathrm{Con} \;\&\; e \leq e' \in X$ implies $X \cup \{e\} \in \mathrm{Con}$.

There is an accompanying notion of state or history. A *configuration* is a, possibly infinite, subset $x \subseteq E$ which is:

- *consistent,* $X \subseteq x \;\&\; X$ is finite implies $X \in \mathrm{Con}$; and
- *down-closed,* $e' \leq e \in x$ implies $e' \in x$.

Two events e, e' are called *concurrent* if the set $\{e, e'\}$ is in Con and neither event is causally dependent on the other; then we write $e \; co \; e'$. In games the relation of *immediate* dependency $e \rightarrowtail e'$, meaning e and e' are distinct with $e \leq e'$ and no event in between, plays a very important role. We write $[X]$ for the down-closure of a subset of events X. Write $\mathscr{C}(E)$ for the configurations of E and $\mathscr{C}(E)^o$ for its *finite configurations*. (Sometimes we shall need to distinguish the precise event structure to which a relation is associated and write, for instance, $\leq_E$, $\rightarrowtail_E$ or co_E.)

Let E and E' be event structures. A *map of event structures* $f : E \rightarrow E'$ is a partial function on events $f : E \rightharpoonup E'$ such that for all $x \in \mathscr{C}(E)$ its direct image $fx \in \mathscr{C}(E')$ and

$$\text{if } e, e' \in x \text{ and } f(e) = f(e') \text{ (with both defined), then } e = e'.$$

Maps of event structures compose as partial functions. Notice that for a *total* map f, *i.e.* when the function f is total, the condition on maps says it is *locally injective*, in the sense that w.r.t. any configuration x of the domain the restriction of f to a function from x is injective; the restriction of total f to a function from x to fx is thus bijective.

Although a map $f : E \rightarrow E'$ of event structures does not generally preserve causal dependency, it does reflect causal dependency locally: whenever $e, e' \in x$, a configuration of E, and $f(e)$ and $f(e')$ are both defined with $f(e') \leq f(e)$, then $e' \leq e$. Consequently, f preserves the concurrency relation: if $e \; co \; e'$ in E then $f(e) \; co \; f(e')$, when defined.

A total map of event structures is *rigid* when it preserves causal dependency. Rigid maps induce discrete fibrations:

PROPOSITION 2.1. *A total map $f : E \to E'$ of event structures is rigid iff for all $x \in \mathscr{C}(E)$ and $y \in \mathscr{C}(E')$,*

$$y \subseteq fx \implies \exists z \in \mathscr{C}(E). \, z \subseteq x \text{ and } fz = y \, .$$

The configuration z is necessarily unique by local injectivity.

3 STABLE FAMILIES

In an event structure, defined above, an event e has a unique causal history, the prime configuration $[e]$. Constructions directly on such event structures can be unwieldy, as often an event is more immediately associated with several mutually inconsistent causal histories. In this case the broader model of stable families is apt, especially so, as any stable family yields an event structure [56, 57].

A subset X of a family of sets $\mathscr{F}$ is *compatible* if there is an element of $\mathscr{F}$ which includes all elements of X; we say X is *finitely compatible* if every finite subset of X is compatible.
A *stable family* is a non-empty family of sets $\mathscr{F}$ which is

• *Complete:* $\forall Z \subseteq \mathscr{F}$. if Z is finitely compatible, $\bigcup Z \in \mathscr{F}$;
• *Stable:* $\forall Z \subseteq \mathscr{F}$. $Z \neq \emptyset \, \& \, Z$ is compatible $\implies \bigcap Z \in \mathscr{F}$;
• *Finitary:* $\forall x \in \mathscr{F}, e \in x \exists x_0 \in \mathscr{F}$. x_0 is finite $\& \, e \in x_0 \subseteq x$;
• *Coincidence-free:* For all $x \in \mathscr{F}, e, e' \in x$ with $e \neq e'$,

$$\exists x_0 \in \mathscr{F}. \, x_0 \subseteq x \, \& \, (e \in x_0 \iff e' \notin x_0) \, .$$

We call elements of $\mathscr{F}$ its *configurations*, $\bigcup \mathscr{F}$ its *events* and write $\mathscr{F}^o$ for its *finite configurations*.

A *map* $f : \mathscr{F} \to \mathscr{G}$ between stable families $\mathscr{F}$ and $\mathscr{G}$ is a partial function f from the events of $\mathscr{F}$ to those of $\mathscr{G}$ such that for all $x \in \mathscr{F}$ its direct image $fx \in \mathscr{G}$ and if $e, e' \in x$ and $f(e) = f(e')$ then $e = e'$. The choice of map ensures an inclusion functor from the category of event structures to that of stable families. The inclusion functor has a right adjoint Pr giving a coreflection (an adjunction with unit an isomorphism). The construction $\mathrm{Pr}(\mathscr{F})$ essentially replaces the original events of a stable family $\mathscr{F}$ by the minimal, prime configurations at which they occur. Let x be a configuration of a stable family $\mathscr{F}$. Define the *prime* configuration of e in x by

$$[e]_x := \bigcap \{y \in \mathscr{F} \mid e \in y \, \& \, y \subseteq x\} \, .$$

By coincidence-freeness, the function $top : \mathscr{C}(\mathrm{Pr}(\mathscr{F})) \to \mathscr{F}$ which takes a prime configuration $[e]_x$ to e is well-defined; it is the counit of the adjunction [56, 57].

THEOREM 3.1. *Let $\mathscr{F}$ be a stable family. Then, $\mathrm{Pr}(\mathscr{F}) := (P, \mathrm{Con}, \leq)$ is an event structure where*

$$P = \{[e]_x \mid e \in x \, \& \, x \in \mathscr{F}\} \, ,$$

$$Z \in \mathrm{Con} \text{ iff } Z \subseteq P \, \& \, \bigcup Z \in \mathscr{F} \, , \text{ and}$$

$$p \leq p' \text{ iff } p, p' \in P \, \& \, p \subseteq p' \, .$$

There is an order isomorphism $\theta : (\mathscr{C}(\mathrm{Pr}(\mathscr{F})), \subseteq) \cong (\mathscr{F}, \subseteq)$ where $\theta(y) := top \, y = \bigcup y$ for $y \in \mathscr{C}(\mathrm{Pr}(\mathscr{F}))$; its mutual inverse is φ where $\varphi(x) = \{[e]_x \mid e \in x\}$ for $x \in \mathscr{F}$.

The partial orders represented by configurations under inclusion are the same whether for event structures or stable families. They are Gérard Berry's *dI-domains* [8, 56, 57].

3.1 Hiding—the defined part of a map

Let $(E, \leq, \mathrm{Con})$ be an event structure. Let $V \subseteq E$ be a subset of 'visible' events. Define the *projection* on V, by $E \downarrow V := (V, \leq_V, \mathrm{Con}_V)$, where $v \leq_V v'$ iff $v \leq v' \, \& \, v, v' \in V$ and $X \in \mathrm{Con}_V$ iff $X \in \mathrm{Con} \, \& \, X \subseteq V$. The operation *hides* all events outside V. It is associated with a *partial-total factorization system*. Consider a partial map of event structures $f : E \rightharpoonup E'$. Let

$$V := \{e \in E \mid f(e) \text{ is defined}\} \, .$$

Then f clearly factors into the composition

$$E \xrightarrow{\;f_0\;} E{\downarrow}V \xrightarrow{\;f_1\;} E'$$

of f_0, a partial map of event structures taking $e \in E$ to itself if $e \in V$ and undefined otherwise, and f_1, a total map of event structures acting like f on V. Note that any $x \in \mathscr{C}(E{\downarrow}V)$ is the image under f_0 of a *minimum* configuration, *viz.* $[x]_E \in \mathscr{C}(E)$. We call f_0 a *projection* and f_1 the *defined part* of the map f.

3.2 Pullbacks

The coreflection from event structures to stable families is a considerable aid in constructing limits in the former from limits in the latter. The *pullback* of total maps of event structures is essential in composing strategies. We can define it via the pullback of stable families, obtained as a stable family of *secured bijections*. Let $\sigma : S \to B$ and $\tau : T \to B$ be total maps of event structures. There is a composite bijection

$$\theta : x \cong \sigma x = \tau y \cong y\,,$$

between $x \in \mathscr{C}(S)$ and $y \in \mathscr{C}(T)$ such that $\sigma x = \tau y$; because σ and τ are total they induce bijections between configurations and their image. The bijection is *secured* when the transitive relation generated on θ by $(s, t) \leq (s', t')$ if $s \leq_S s'$ or $t \leq_T t'$ is a finitary partial order.

THEOREM 3.2. *Let $\sigma : S \to B$, $\tau : T \to B$ be total maps of event structures. The family $\mathscr{R}$ of secured bijections between $x \in \mathscr{C}(S)$ and $y \in \mathscr{C}(T)$ such that $\sigma x = \tau y$ is a stable family. The functions $\pi_1 : \mathrm{Pr}(\mathscr{R}) \to S$, $\pi_2 : \mathrm{Pr}(\mathscr{R}) \to T$, taking a secured bijection with top to, respectively, the left and right components of its top, are maps of event structures. $\mathrm{Pr}(\mathscr{R})$ with π_1, π_2 is the pullback of σ, τ in the category of event structures.*

NOTATION 3.3. W.r.t. $\sigma : S \to B$ and $\tau : T \to B$, define $x \wedge y$ to be the configuration of their pullback which corresponds via this isomorphism to a secured bijection between $x \in \mathscr{C}(S)$ and $y \in \mathscr{C}(T)$, necessarily with $\sigma x = \tau y$; any configuration of the pullback takes the form $x \wedge y$ for unique x and y.

4 CONCURRENT GAMES AND STRATEGIES

The driving idea is to replace the traditional role of game trees by that of event structures. Both games and strategies will be represented by *event structures with polarity*, which comprise (A, pol_A) where A is an event structure and a polarity function $pol_A : A \to \{+, -, 0\}$ ascribing a polarity + (Player) or − (Opponent) or 0 (neutral) to its events. The events correspond to (occurrences of) moves. It will be technically useful to allow events of neutral polarity; they arise, for example, in a play between a strategy and a counterstrategy. Maps are those of event structures which preserve polarity. A *game* is represented by an event structure with polarities restricted to + or −, with no neutral events.

Definition 4.1. In an event structure with polarity, with configurations x and y, write $x \subseteq^- y$ to mean inclusion in which all the intervening events $y \setminus x$ are Opponent moves. Write $x \subseteq^+ y$ for inclusion in which the intervening events are neutral or Player moves. For a subset of events X we write X^+ and X^- for its restriction to Player and Opponent moves, respectively. The *Scott order* will play a central role: between $x, y \in \mathscr{C}(A)$, where A is a game, define

$$y \sqsubseteq_A x \iff \exists z \in \mathscr{C}(A).\ y \supseteq^- z \ \&\ z \subseteq^+ x$$

—it is not hard to show that z is unique and equal to $x \cap y$. The Scott order is also characterised by

$$y \sqsubseteq_A x \iff y^- \supseteq x^- \ \&\ y^+ \subseteq x^+\,,$$

which makes clear why it is a partial order. The Scott order is so named because it reduces to Scott's order on functions in special cases and plays a central role in relating games to Scott domains and "generalised domain theory" [32].

There are two fundamentally important operations on games. One is that of forming the *dual game*. On a game A this amounts to reversing the polarities of events to produce the dual $A^{\perp}$. The other operation, a *simple parallel composition $A\|B$*, is achieved on games A and B by simply juxtaposing them, ensuring a finite subset of events is consistent if its overlaps with the two games are individually consistent; any configuration x of $A\|B$ decomposes into $x_A\|x_B$ where x_A and x_B are configurations of A and B respectively.

A *strategy in* a game A is a total map $\sigma : S \to A$ of event structures with polarity such that

(i) if $\sigma x \subseteq^{-} y$, for $x \in \mathscr{C}(S), y \in \mathscr{C}(A)$, there is a unique $x' \in \mathscr{C}(S)$ with $x \subseteq x'$ and $\sigma x' = y$;

(ii) if $s \rightarrowtail_S s'$ and $(pol(s) = +$ or $pol(s') = -)$, then $\sigma(s) \rightarrowtail_A \sigma(s')$.

The conditions prevent Player from constraining Opponent's behaviour beyond the constraints of the game. Condition (i) is *receptivity*, ensuring that the strategy is open to all moves of Opponent permitted by the game. Condition (ii), called *innocence* in [22], ensures that the only additional immediate causal dependencies a strategy can enforce beyond those of the game are those in which a Player move awaits moves of Opponent. A map $f : \sigma \Rightarrow \sigma'$ of strategies $\sigma : S \to A$ and $\sigma' : S' \to A$ is a map $f : S \to S'$ such that $\sigma = \sigma' f$; this determines when strategies are isomorphic.

Following [18, 35], a *strategy from* a game A *to* a game B is a strategy in the game $A^{\perp}\|B$. Given a strategy from B to a game C, so in $B^{\perp}\|C$, we compose the two strategies essentially by playing them against each other in the common game B, where if one strategy makes a Player move the other sees it as a move of Opponent. The conditions of receptivity and innocence precisely ensure that the copycat strategy behaves as identity w.r.t. composition, detailed below [53].

4.1 Copycat

Let A be a game. The copycat strategy $\propto_A : \mathbb{CC}_A \to A^{\perp}\|A$ is an instance of a strategy from A to A. The event structure $\mathbb{CC}_A$ is based on the idea that Player moves in one component of the game $A^{\perp}\|A$ always copy corresponding moves of Opponent in the other component. For $c \in A^{\perp}\|A$ we use $\bar{c}$ to mean the corresponding copy of c, of opposite polarity, in the alternative component. The event structure $\mathbb{CC}_A$ comprises $A^{\perp}\|A$ with extra causal dependencies $\bar{c} \le c$ for all events c with $pol_{A^{\perp}\|A}(c) = +$; with the original causal dependency they generate a partial order; a finite subset is consistent in $\mathbb{CC}_A$ iff its down-closure w.r.t. $\le$ is consistent in $A^{\perp}\|A$. The map $\propto_A$ acts as the identity function. In characterising the configurations of $\mathbb{CC}_A$ we recall the Scott order of Defn 4.1.

LEMMA 4.2. *Let A be a game. Let $x \in \mathscr{C}(A^{\perp})$ and $y \in \mathscr{C}(A)$. Then*

$$x\|y \in \mathscr{C}(\mathbb{CC}_A) \quad \text{iff} \quad y \sqsubseteq_A x.$$

4.2 Composition

Two strategies $\sigma : S \to A^{\perp}\|B$ and $\tau : T \to B^{\perp}\|C$ compose via pullback and hiding, summarised below.

Ignoring polarities, by forming the pullback of $\sigma \| C$ and $A \| \tau$ we obtain the synchronisation of complementary moves of S and T over the common game B; subject to the causal constraints of S and T, the effect is to instantiate the Opponent moves of T in $B^\perp$ by the corresponding Player moves of S in B, and *vice versa*. Reinstating polarities we obtain the *interaction* of σ and τ

$$\tau \circledast \sigma : T \circledast S \to A^\perp \| B^0 \| C,$$

where we assign neutral polarities to all moves in or over B. Neutral moves over the common part B^0 remain unhidden. The map $A^\perp \| B^0 \| C \rightharpoonup A^\perp \| C$ is undefined on B^0 and otherwise mimics the identity. Pre-composing this map with $\tau \circledast \sigma$ we obtain a partial map $T \circledast S \rightharpoonup A^\perp \| C$; it is undefined on precisely the neutral events of $T \circledast S$. The defined parts of its partial-total factorization yields

$$\tau \odot \sigma : T \odot S \to A^\perp \| C$$

—this is the *composition* of σ and τ.

NOTATION 4.3. For $x \in \mathscr{C}(S)$ and $y \in \mathscr{C}(T)$, let $\sigma x = x_A \| x_B$ and $\tau y = y_B \| y_C$ where $x_A \in \mathscr{C}(A)$, $x_B, y_B \in \mathscr{C}(B)$, $y_C \in \mathscr{C}(C)$. Define $y \circledast x = (x \| y_C) \wedge (x_A \| y)$. This is a partial operation only defined if the $\wedge$-expression is. It is defined and glues configurations x and y together at their common overlap over B provided $x_B = y_B$ and a finitary partial order of causal dependency results. Any configuration of $T \circledast S$ has the form $y \circledast x$, for unique $x \in \mathscr{C}(S), y \in \mathscr{C}(T)$.

4.4 A bicategory of strategies

We obtain a bicategory Strat for which the objects are games, the arrows $\sigma : A \rightarrowtail B$ are strategies $\sigma : S \to A^\perp \| B$; with 2-cells $f : \sigma \Rightarrow \sigma'$ maps of strategies. The vertical composition of 2-cells is the usual composition of maps. Horizontal composition is the composition of strategies $\odot$ (which extends to a functor via the universality of pullback and partial-total factorisation). We can restrict the 2-cells to be rigid maps and still obtain a bicategory. The bicategory of strategies is compact-closed, though with the addition of winning conditions—Section 4.5—this weakens to $*$-autonomous.

A strategy $\sigma : S \to A$ is *deterministic* if S is deterministic, *viz.*

$$\forall X \subseteq_{\mathrm{fin}} S. \ [X]^- \in \mathrm{Con}_S \implies X \in \mathrm{Con}_S,$$

where $[X]^- := \{s' \in S \mid \exists s \in X. \ pol_S(s') = - \ \& \ s' \leq s\}$. So, a strategy is deterministic if consistent behaviour of Opponent is answered by consistent behaviour of Player. Copycat $\mathfrak{cc}_A$ is deterministic iff the game A is *race-free*, *i.e.* if $x \subseteq^- y$ and $x \subseteq^+ z$ in $\mathscr{C}(A)$ then $y \cup z \in \mathscr{C}(A)$. The bicategory of strategies restricts to a bicategory of deterministic strategies between race-free games.

There are several ways to reformulate strategies. Deterministic strategies coincide with the *receptive* ingenuous strategies of Melliès and Mimram based on asynchronous transition systems [40, 60]. Via the Scott order, we can see strategies as a refinement of profunctors: a strategy in a game A induces a discrete fibration, so presheaf, on $(\mathscr{C}(A)^o, \sqsubseteq_A)$, a construction which extends to strategies between games [63].

4.5 Winning conditions

Winning conditions of a game A specify a subset W of its configurations, an outcome in which is a win for Player. Informally, a strategy (for Player) is *winning* if it always prescribes moves for Player to end up in a winning configuration, no matter what the activity or inactivity of Opponent.

Formally, a game with winning conditions (A, W_A) comprises a concurrent game A with winning conditions $W_A \subseteq \mathscr{C}(A)$. A strategy $\sigma : S \to A$ is *winning* if σx is in W_A for all $+$-maximal configurations x of S; in general, a configuration is $+$-maximal if no additional Player, or neutral,

moves can occur from it. That σ is winning can be shown equivalent to: all plays of σ against any counterstrategy of Opponent result in a win for Player [16, 64].

As the dual of a game with winning conditions (A, W_A) we again reverse the roles of Player and Opponent, and take its winning conditions to be the set-complement of W_A, *i.e.* $(A, W_A)^\perp = (A^\perp, \mathscr{C}(A) \setminus W_A)$.

In a parallel composition of games with winning conditions, we deem a configuration x of $A \| B$ winning if its component x_A is winning in A or its component x_B is winning in B: $(A, W_A) \| (B, W_B) :=$ $(A \| B, W)$ where $W = \{ x \in \mathscr{C}(A \| B) \mid x_A \in W_A \text{ or } x_B \in W_B \}$.

With these extensions, we take a winning strategy from a game (A, W_A) to a game (B, W_B), to be a winning strategy in the game $A^\perp \| B$ —its winning conditions form the set

$$\{ x \in \mathscr{C}(A^\perp \| B) \mid x_A \in W_A \Rightarrow x_B \in W_B \} .$$

When games are race-free, copycat will be a winning strategy. The composition of winning strategies is winning [16, 64]. In the proof the following lemma is critical:

LEMMA 4.3. *Let* $\sigma : S \to A^\perp \| B$ *and* $\tau : T \to B^\perp \| C$ *be strategies. Suppose* $y \circledast x \in \mathscr{C}(T \circledast S)$ *where* $x \in \mathscr{C}(S)$ *and* $y \in \mathscr{C}(T)$. *Then,* $y \circledast x$ *is +-maximal iff both* x *and* y *are +-maximal.*

One can extend winning conditions to payoff functions [17] or to allow draws, where neither player wins [61].

4.6 Imperfect information

In a game of *imperfect information* some moves are masked, or inaccessible, and strategies with dependencies on unseen moves are ruled out. One can extend games with imperfect information in a way that respects the operations of concurrent games and strategies [61]. Each move of a game is assigned a level in a global order of access levels; moves of the game or its strategies can only causally depend on moves at equal or lower levels.

In more detail, a fixed preorder of *access levels* $(\Lambda, \leq)$ is pre-supposed. A Λ-*game* comprises a game A with a *level function* $l : A \to \Lambda$ such that if $a \leq_A a'$ then $l(a) \leq l(a')$ for all moves a, a' in A. A Λ-*strategy* in the Λ-game is a strategy $\sigma : S \to A$ for which if $s \leq_S s'$ then $l\sigma(s) \leq l\sigma(s')$ for all s, s' in S. The access levels of moves in a game are left undisturbed in forming the dual and parallel composition of Λ-games. As before, a Λ-strategy from a Λ-game A to a Λ-game B is a Λ-strategy in the game $A^\perp \| B$. It can be shown that Λ-strategies compose [61].

4.7 A language for strategies

We recall briefly the language for strategies introduced in [12]. Games $A, B, C, \cdots$ play the role of types. Operations on games include forming the dual $A^\perp$, simple parallel composition $A \| B$, a sum $\Sigma_{i \in I} A_i$ as well as recursively-defined games.

Terms, denoting strategies, have typing judgements

$$x_1 : A_1, \cdots, x_m : A_m \vdash t \dashv y_1 : B_1, \cdots, y_n : B_n ,$$

where all the variables are distinct, interpreted as a strategy from $\vec{A} = A_1 \| \cdots \| A_m$ to $\vec{B} = B_1 \| \cdots \| B_n$. We can picture the term t as a box with input and output wires for the variables $\vec{x}$ and $\vec{y}$:

The term t denotes a strategy $\sigma : S \to \vec{A}^\perp \| \vec{B}$. It does so by describing witnesses, configurations of S, to a relation between configurations $\vec{x}$ of $\vec{A}$ and $\vec{y}$ of $\vec{B}$. For example, the term

$$x : A \vdash y \sqsubseteq_A x \dashv y : A$$

denotes the copycat strategy on a game A; it describes configurations of copycat, $\mathbb{CC}_A$, as witnesses, *viz.* those configurations $x \| y$ of $\mathbb{CC}_A$ for which $y \sqsubseteq_A x$ in the Scott order. There are other operations, such as sum $[]$ and pullback $\wedge$ on strategies of the same type.

Duality is caught by the rules

$$\frac{\Gamma, x : A \vdash t \dashv \Delta}{\Gamma \vdash t \dashv x : A^\perp, \Delta} \qquad \frac{\Gamma \vdash t \dashv x : A, \Delta}{\Gamma, x : A^\perp \vdash t \dashv \Delta}$$

and composition of strategies by

$$\frac{\Gamma \vdash t \dashv \Delta \qquad \Delta \vdash u \dashv H}{\Gamma \vdash \exists \Delta . \, [\, t \, \| \, u \,] \dashv H}$$

which, in the picture of strategies as boxes, joins the output wires of one strategy to input wires of the other. Simple parallel composition of strategies arises when Δ is empty.

5 FROM FUNCTIONS TO STRATEGIES

The language for strategies in [12] included a judgement

$$x : A \vdash g(y) \sqsubseteq_C f(x) \dashv y : B$$

for building strategies out of expressions $f(x)$ and $g(y)$ denoting "affine functions." It breaks down into a composition

$$x : A \vdash \exists z : C. \, [\, g(y) \sqsubseteq_C z \, \| \, z \sqsubseteq_C f(x) \,] \dashv y : B \,.$$

Here we present a considerably broader class of affine-stable functions f and "co-affine-stable" functions g with which to define strategies in this manner. It hinges on the Scott order to convert functional dependency to causal dependency, in the sense captured by Theorem 5.2 below.

5.1 Affine-stable maps and their strategies

Definition 5.1. An *affine-stable map* between games from A to B, is a function $f : \mathscr{C}(A) \to \mathscr{C}(B)$ which is

• *polarity-respecting:* for $x, y \in \mathscr{C}(A)$,

$$x \subseteq^- y \Rightarrow f(x) \subseteq^- f(y) \text{ and } x \subseteq^+ y \Rightarrow f(x) \subseteq^+ f(y) \,;$$

• *+-continuous:* for $x \in \mathscr{C}(A)$,

$$b \in f(x) \,\&\, pol_B(b) = + \Rightarrow \exists x_0 \in \mathscr{C}(A)^o. \, x_0 \subseteq x \,\&\, b \in f(x_0) \,;$$

• *−-image finite:* for all finite configurations $x \in \mathscr{C}(A)^o$ the set $f(x)^-$ is finite;

• *affine:* for all compatible families $\{x_i \mid i \in I\}$ in $\mathscr{C}(A)$,

$$\bigcup_{i \in I} f(x_i) \subseteq^+ f(\bigcup_{i \in I} x_i)$$

—when I is empty this amounts to $\emptyset \subseteq^+ f(\emptyset)$; and

• *stable:* for all compatible families $\{x_i \mid i \in I\} \neq \emptyset$ in $\mathscr{C}(A)$,

$$f(\bigcap_{i \in I} x_i) \subseteq^- \bigcap_{i \in I} f(x_i) \,.$$

When all the moves of games A and B are those of Player, the definition reduces to that of stable function. If all moves are those of Opponent, it becomes that of demand maps—see Section 6.2 [59]. Affine-stable maps include maps of event structures with polarity, including partial maps between games, and the affine maps of [12]. They are the most general maps out of which we can construct a corresponding strategy, in a way we now describe.

THEOREM 5.2. *Let $f : \mathscr{C}(A) \to \mathscr{C}(B)$ be an affine-stable map between games A and B. Then*

$$\mathscr{F} := \{x \| y \in \mathscr{C}(A^\perp \| B) \mid y \sqsubseteq_B f(x)\}$$

is a stable family. The map $top : \mathrm{Pr}(\mathscr{F}) \to A^\perp \| B$ is a strategy $f_! : A \nrightarrow B$. The strategy $f_!$ is deterministic if A and B are race-free and f reflects $-$-compatibility, i.e. $x \subseteq^- x_1$ and $x \subseteq^- x_2$ in $\mathscr{C}(A)$ and $f x_1 \cup f x_2 \in \mathscr{C}(B)$ implies $x_1 \cup x_2 \in \mathscr{C}(A)$.

The theorem above explains how to convert functional dependency, expressed as $y \sqsubseteq_B f(x)$, to causal dependency between moves $\mathrm{Pr}(\mathscr{F})$, obtained as primes of the stable family $\mathscr{F}$. The expression of functional dependency as causal dependency is quite subtle; the direction of causal dependency hinges critically on the polarities of events.

For f an affine-stable map from A to B we can write $f_!$ as

$$x : A \vdash y \sqsubseteq_B f(x) \dashv y : B.$$

Through suitable f we can create strategies from structural maps, injections and projections as strategies, for conditional and case statements and, generally, much of the causal wiring that is often explained informally in diagrammatic reasoning. If σ is a strategy in A then $f_! \odot \sigma$ is its "push-forward" to a strategy in B. Some basic examples:

Example 5.3. (*Projectors*) Let $f : A \| B \to B$ be the function undefined on game A but acting as identity on game B. Let σ be a strategy in the game $A \| B$. The strategy $f_! \odot \sigma$ is its projection to a strategy in B. □

Example 5.4. (*Duplicators*) Let A be a game. Consider the function $d_A : x \mapsto x \| x$ from $\mathscr{C}(A)$ to $\mathscr{C}(A \| A)$. It is easily checked to be affine-stable. Hence there is a *duplicator* strategy $\delta_A = d_{A!} : A \nrightarrow A \| A$. (The strategy δ_A is not natural in A as $\|$ is not a product, except in subcategories.) □

Example 5.5. (*Detectors*) Let A be a game. Let $X \in \mathrm{Con}_A$ with $X \subseteq A^+$. Let $\boxplus$ be a single "detector" event, of +ve polarity. Let

$$d_X : \mathscr{C}(A) \to \mathscr{C}(\boxplus)$$

be the function such that

$$d_X(x) = \begin{cases} \boxplus & \text{if } X \subseteq x, \\ \emptyset & \text{otherwise.} \end{cases}$$

The function d_X is affine-stable. There is a *detector* strategy

$$d_{X!} : A \nrightarrow \boxplus.$$

The strategy simply adjoins extra causal dependencies $a \rightarrow \boxplus$ from $a \in X$. It detects the presence of X. In a similar way, one can extend detectors to detect the occurrence of one of a family $\langle X_i \rangle_{i \in I}$ of $X_i \in \mathrm{Con}_A$ provided $X_i \cup X_j \in \mathrm{Con}_A \implies i = j$ for $i, j \in I$. □

Example 5.6. (*Blockers*) Let A be a game and $Y \subseteq A^-$. Let

$$h_Y : \mathscr{C}(A) \to \mathscr{C}(\boxminus)$$

be the function which acts so

$$h_Y(x) = \begin{cases} \boxminus & \text{if } x \cap Y \neq \emptyset, \\ \emptyset & \text{otherwise.} \end{cases}$$

h_Y is a map of event structures so affine-stable. The *blocker* strategy $h_{Y!}$ adjoins causal dependencies $\boxminus \rightarrow a$ from $\boxminus$ to each $a \in Y$. The absence of move $\boxminus$ blocks all moves Y. □

THEOREM 5.7. *The operation $(\_)_!$ is a (pseudo) functor from the category of affine-stable maps to concurrent strategies* Strat.

5.2 co-Affine-stability

We examine the dual, or co-notion, to affine-stability. An affine-stable map f from $A^\perp$ to $B^\perp$ yields a strategy $f_! : A^\perp \rightarrow B^\perp$, so by duality a strategy $f^* : B \rightarrow A$. We obtain the dual to Theorems 5.2, 5.7 as a corollary:

COROLLARY 5.8. *Let* $g : \mathscr{C}(A) \to \mathscr{C}(B)$ *be such that* $g : \mathscr{C}(A^\perp) \to \mathscr{C}(B^\perp)$ *is affine-stable, then*

$$\mathscr{G} := \{y \| x \in \mathscr{C}(B^\perp \| A) \mid g(x) \sqsubseteq_B y\}$$

is a stable family. The map $top : \mathrm{Pr}(\mathscr{G}) \to B^\perp \| A$ *is a strategy* $g^* : B \rightarrow A$. *The strategy* g^* *is deterministic if* A *is race-free and* g *reflects +-compatibility. The operation* $(\_)^*$ *is a contravariant (pseudo) functor from the category of affine-stable maps to* Strat.

For g an affine-stable map from $A^\perp$ to $B^\perp$ we can write g^* as

$$y : B \vdash g(y) \sqsubseteq_B x \dashv x : A.$$

For a strategy σ in game B the operation $g^* \odot \sigma$ yields the strategy in A got as the pullback of σ along g. In particular, if A prefixed game B by some initial move, $g^* \odot \sigma$ would be a prefix operation on strategies.

5.3 An adjunction

An affine-stable map f from A to B is not generally an affine-stable map from $A^\perp$ to $B^\perp$. The next definition, of an *additive-stable* map f from A to B, bluntens affine-stability to ensure f is also a additive-stable map from $A^\perp$ to $B^\perp$; and hence is associated with both a strategy $f_! : A \rightarrow B$ and a converse strategy $f^* : B \rightarrow A$. Together they form an adjunction.

Definition 5.9. A *additive-stable map* between event structures with polarity, from A to B, is a function $f : \mathscr{C}(A) \to \mathscr{C}(B)$ which is
• *polarity-respecting:* for $x, y \in \mathscr{C}(A)$,

$$x \subseteq^- y \Rightarrow f(x) \subseteq^- f(y) \text{ and } x \subseteq^+ y \Rightarrow f(x) \subseteq^+ f(y);$$

• *image finite:* if $x \in \mathscr{C}(A)^o$ then $f(x) \in \mathscr{C}(B)^o$;
• *additive:* for all compatible families $\{x_i \mid i \in I\}$ in $\mathscr{C}(A)$,

$$\bigcup_{i \in I} f(x_i) = f(\bigcup_{i \in I} x_i);$$

• *stable:* for all compatible families $\{x_i \mid i \in I\} \neq \emptyset$ in $\mathscr{C}(A)$,

$$f(\bigcap_{i \in I} x_i) = \bigcap_{i \in I} f(x_i).$$

The usual maps of games are additive-stable, including those which are partial, as are Girard's linear maps. Additive-stability is indifferent to a switch of polarities:

PROPOSITION 5.10. *An additive-stable map* f *from* A *to* B *is an additive-stable map* f *from* $A^\perp$ *to* $B^\perp$ *and vice versa.*

Given an additive-stable map f from A to B we obtain a strategy $f_! : A \rightarrow B$ and, via f from $A^\perp$ to $B^\perp$, $f^* : B \rightarrow A$.

THEOREM 5.11. *Let* f *be an additive-stable map from* A *to* B *between event structures with polarity. The strategies* $f_!$ *and* f^* *form an adjunction* $f_! \dashv f^*$ *in the bicategory* Strat.

This says Strat forms a *pseudo double category* [34, 48]. In Section 6.3.1 we apply Theorem 5.11 to relate deterministic strategies in general, to those of Geometry of Interaction. The adjunction in Strat of Theorem 5.11 yields a traditional adjunction:

Corollary 5.12. *Let f be an additive-stable map from game A to game B. Let Strat_A be the category of strategies in the game A, and Strat_B that in B. Then there are functors $f_! \odot (\_) : \mathrm{Strat}_A \to \mathrm{Strat}_B$ and $f^* \odot (\_) : \mathrm{Strat}_B \to \mathrm{Strat}_A$ with $f_! \odot (\_)$ left adjoint to $f^* \odot (\_)$.*

6 FROM STRATEGIES TO FUNCTIONS

We recover familiar notions of games from those based on event structures. A game is *tree-like* when any two events are either inconsistent or causally dependent. When such a game is race-free, at any finite configuration, the next possible moves, if there are any, belong purely to Player, or purely to Opponent. Then, at each position where Player may move, a deterministic strategy either chooses a unique move or to stay put. In contrast to many presentations of games, in a concurrent strategy Player isn't forced to make a move, though that can be encouraged through suitable winning conditions. A counterstrategy, as a strategy in the dual game, picks moves for Opponent at their configurations. The interaction $\tau \circledast \sigma$ of a deterministic strategy σ with a deterministic counterstrategy τ determines a finite or infinite branch in the tree of configurations, which in the presence of winning conditions will be a win for one of the players.

On tree-like games we recover familiar notions. More surprising is that by exploiting the richer structure of concurrent games we can recover other familiar paradigms, not traditionally tied to games, or if so only somewhat informally. We start by rediscovering Berry's stable domain theory, of which Jean-Yves Girard's qualitative domains and coherence spaces are special cases. The other examples, from dataflow, logic and functional programming, concern ways of handling interaction within a functional approach. We shall restrict to race-free games, so guaranteeing that deterministic strategies have an identity w.r.t. composition, given by copycat.

6.1 Stable functions

Consider games in which all moves are Player moves. Consider a strategy σ from one such purely Player game A to another B. This is a map $\sigma : S \to A^\perp \| B$ which is receptive and innocent. Notice that in $A^\perp \| B$ all the Opponent moves are in $A^\perp$ and all the Player moves are in B. By receptivity any configuration of A can be input. The only new immediate causal connections, beyond those in $A^\perp$ and B, that can be introduced in a strategy are those from Opponent moves of $A^\perp$ to a Player move in B. Beyond the causal dependencies of the games, a strategy σ can only make a Player move in B causally depend on a finite subset of moves in $A^\perp$.
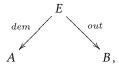
When σ is deterministic, all conflicts are inherited from conflicts between Opponent moves. Then the strategy σ gives rise to a stable function from the configurations of A to the configurations of B. Conversely, such a stable function f yields a deterministic strategy $f_! : A \longrightarrow B$, by Theorem 5.2.

Theorem 6.1. *The category dI of dI-domains and stable functions, enriched by the stable order, is equivalent to the bicategory of deterministic strategies between purely Player games with rigid 2-cells. (The bicategory of deterministic strategies between purely Player games with* all *2-cells is equivalent to the category of dI-domains enriched by the Scott—or pointwise—order.)*
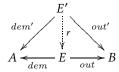
The category of dI-domains and stable functions is well-known to be cartesian-closed; its function space and product are realised by constructions $[A \to B]$ and $A \| B$ on event structures. When the games are further restricted to have trivial causal dependency we recover Girard's *qualitative domains* and, with conflict determined in a binary fashion, his *coherence spaces*. Girard's models for polymorphism there generalise to dI-domains, with dependent types $\Pi_{x:A} B(x)$ and $\Sigma_{x:A} B(x)$ on event structures [19, 20, 57]—see Appendices A.1, A.2.

6.2 Stable spans

When between games in which all the moves are Player moves, a general, nondeterministic, strategy corresponds to a *stable span*, a form of many-valued stable function which has been discovered, and rediscovered, in giving semantics to higher-order processes and especially nondeterministic dataflow [44, 55, 59]; the trace of strategies, derived from their compact closure, specialises to the feedback operation of dataflow. Recall a *stable span* comprises

$$
\begin{array}{ccc}
 & E & \\
{\scriptstyle dem}\swarrow & & \searrow{\scriptstyle out} \\
A & & B\,,
\end{array}
$$

with event structure E relating input given by an event structure A and output by an event structure B. The map $out : E \to B$ is a *rigid* map. The map $dem : E \to A$, associated to input, is of a different character. It is a *demand* map, *i.e.*, a function from $\mathscr{C}(E)$ to $\mathscr{C}(A)$ which preserves unions and finite configurations; $dem(x)$ is the minimum input for x to occur and is the union of the demands of its events. The occurrence of an event e in E demands minimum input $dem([e])$ and is observed as the output event $out(e)$. Spans from A to B are related by the usual 2-cells, here (necessarily) rigid maps r making the diagram below commute:

$$
\begin{array}{ccc}
 & E' & \\
{\scriptstyle dem'}\swarrow & \Big\downarrow {\scriptstyle r} & \searrow{\scriptstyle out'} \\
A \xleftarrow{\ dem\ } & E & \xrightarrow{\ out\ } B
\end{array}
$$

Stable spans compose via the usual pullback construction of spans, as both demand and output maps extend to functions between configurations. A stable span E corresponds to a (special) profunctor

$$
\tilde{E}(x,y) = \{w \in \mathscr{C}(E)^o \mid dem(w) \subseteq x \ \& \ out\, w = y\}\,,
$$

between the partial-order categories $\mathscr{C}(A)^o$ and $\mathscr{C}(B)^o$ —a correspondence that respects composition. Recalling the view of profunctors as Kleisli maps w.r.t. the presheaf construction [24], we borrow from Moggi [42] and describe the composition of stable spans $F : A \rightarrow\!\!\!\!+ B$, $G : B \rightarrow\!\!\!\!+ C$ as

$$
G\odot F(x) = \text{ let } y \Leftarrow F(x) \text{ in } G(y)
$$

—which, via the correspondence with profunctors, stands for the coend $\int^{y \in \mathscr{C}(B)^o} \tilde{F}(x,y) \times \tilde{G}(y, \_)$. In using let-notation we can take account of the shape of the configuration y in the definition of G, in effect an informal pattern matching.

Stable spans are monoidal closed [44, 45]: w.r.t. an event structure A, the functor $(\_\|A)$ has a right adjoint, the function space $[A \multimap \_]$. The construction $[A \multimap B]$ is recalled in Appendix B along with a more general dependent product $\Pi^s_{x:A}B(x)$ for stable spans: the type of stable spans which on input $x : A$ yield output $y : B(x)$ nondeterministically. Stable spans are trace monoidal closed; their trace is described in [55].

Let A and B be purely Player games. A strategy $\sigma : S \to A^\perp\|B$ gives rise to a stable span

$$
A \xleftarrow{\ dem\ } E \xrightarrow{\ out\ } B\,,
$$

where $E = S^+$, and out gives the image of its events in B and dem those events in A on which they casually depend. Conversely given a stable span, as above, we obtain a strategy as the composition $out_!\odot dem^*$, by the results of Section 5; as regarding E and A as purely Player games, both out and $dem : \mathscr{C}(E^\perp) \to \mathscr{C}(A^\perp)$ are affine-stable. In the strategy $out_!\odot dem^* : S \to A^\perp\|B$ so obtained,

S comprises the disjoint union of A and E with the additional causal dependencies of $e \in E$ on $a \in A^\perp$ prescribed by dem.

THEOREM 6.2. *The bicategory* Stab *of stable spans is equivalent to the bicategory of strategies between purely Player games with rigid 2-cells.*

We show that in a similar way, we obtain geometry of interaction, dialectica categories, containers, lenses, open games and learners, optics and dependent optics by moving to slightly more complicated subcategories of games, sometimes with winning conditions and imperfect information.

6.3 Geometry of Interaction

Let's now consider slightly more complex games. A *GoI game* comprises a parallel composition $A := A_1 \| A_2$ of a purely Player game A_1 with a purely Opponent game A_2. Consider a strategy σ from a GoI game $A := A_1 \| A_2$ to a GoI game $B := B_1 \| B_2$. Rearranging the parallel compositions,

$$A^\perp \| B = A_1^\perp \| A_2^\perp \| B_1 \| B_2 \cong (A_1 \| B_2^\perp)^\perp \| (A_2^\perp \| B_1).$$

So σ, as a strategy in $A^\perp \| B$, corresponds to a strategy from the purely Player game $A_1 \| B_2^\perp$ to the purely Player game $A_2^\perp \| B_1$. We are back to the simple situation considered in the previous section, of strategies between purely Player games.

Strategies between GoI games, from A to B, correspond to stable spans from $A_1 \| B_2^\perp$ to $A_2^\perp \| B_1$. The maps are familiar from models of geometry of interaction built as free compact-closed categories from traced monoidal categories [3, 6], though here lifted to the *bicategory* Stab of stable spans.

THEOREM 6.3. *The bicategory of strategies on GoI games with rigid 2-cells is equivalent to the free compact-closed bicategory built on the trace monoidal bicategory* Stab.

When deterministic, strategies from GoI game A to GoI game B correspond to a stable function from $\mathscr{C}(A_1 \| B_2^\perp)$ to $\mathscr{C}(A_2^\perp \| B_1)$. Note that a configuration of a parallel composition of games splits into a pair of configurations:

$$\mathscr{C}(A_1 \| B_2^\perp) \cong \mathscr{C}(A_1) \times \mathscr{C}(B_2), \quad \mathscr{C}(A_2^\perp \| B_1) \cong \mathscr{C}(A_2) \times \mathscr{C}(B_1).$$

Thus deterministic strategies from A to B correspond to stable functions

$$S = \langle g, f \rangle : \mathscr{C}(A_1) \times \mathscr{C}(B_2) \to \mathscr{C}(A_2) \times \mathscr{C}(B_1),$$

associated with a pair of stable functions $g : \mathscr{C}(A_1) \times \mathscr{C}(B_2) \to \mathscr{C}(A_2)$ and $f : \mathscr{C}(A_1) \times \mathscr{C}(B_2) \to \mathscr{C}(B_1)$, summarised diagrammatically by:

Such maps are obtained by Abramsky and Jagadeesan's GoI construction, here starting from *stable* domain theory [4].

The composition of deterministic strategies between GoI games, σ from A to B and τ from B to C coincides with the composition of GoI given by "tracing out" B_1 and B_2. Precisely, supposing σ corresponds to the stable function

$$S : \mathscr{C}(A_1) \times \mathscr{C}(B_2) \to \mathscr{C}(A_2) \times \mathscr{C}(B_1)$$

and τ to the stable function

$$T : \mathscr{C}(B_1) \times \mathscr{C}(C_2) \to \mathscr{C}(B_2) \times \mathscr{C}(C_1),$$

we see a loop in the functional dependency at B:

Accordingly, the composition $\tau \odot \sigma$ corresponds to the stable function taking $(x_1, z_2) \in \mathscr{C}(A_1) \times \mathscr{C}(C_2)$ to $(x_2, z_1) \in \mathscr{C}(A_2) \times \mathscr{C}(C_1)$ in the least solution to the equations

$$(x_2, y_1) = S(x_1, y_2) \ \text{ and } \ (y_2, z_1) = T(y_1, z_2)$$

—given, as in Kahn networks, by taking a least fixed point.

THEOREM 6.4. *The bicategory of deterministic strategies on GoI games with rigid 2-cells is equivalent to the free compact-closed category* $\mathrm{Int}(\mathrm{dI})$ *of [6] and the Geometry of Interaction category* $\mathscr{G}(\mathrm{dI})$ *of [3] built on the category* dI *of dI-domains and stable functions.*

Geometry of Interaction started as an investigation of the nature of proofs of linear logic, understood as networks [28]. It has subsequently been tied to optimal reduction in the λ-calculus [29], and inspired implementations via token machines on networks [39, 43]; when the two components of a GoI game match events of exit and entry of a token at a link.

It is straightforward to extend GoI games with winning conditions. A winning condition on a GoI game $A = A_1 \| A_2$ picks out a subset of the configurations $\mathscr{C}(A)$, so amounts to specifying a property $W_A(x_1, x_2)$ of pairs (x_1, x_2) in $\mathscr{C}(A_1) \times \mathscr{C}(A_2)$. That a deterministic strategy from GoI game A to GoI game $B = B_1 \| B_2$ is winning means

$$W_A(x, g(x, y)) \implies W_B(f(x, y), y),$$

for all $x \in \mathscr{C}(A_1), y \in \mathscr{C}(B_2)$, when expressed in terms of the pair of stable functions the strategy determines. In particular, a deterministic winning strategy in the individual GoI game B, with winning conditions W_B, corresponds to a stable function $f : \mathscr{C}(B_2) \to \mathscr{C}(B_1)$ such that $\forall y \in \mathscr{C}(B_2). W_B(f(y), y)$.

With stable spans, unlike with dI-domains with stable functions, the operation of parallel composition $\|$ is no longer a product; stable spans are monoidal-closed and not a cartesian-closed. While general, not just deterministic, strategies $\sigma : A {\longrightarrow} B$ between GoI games are expressible as stable spans $A_1 \| B_2^{\perp} {\longrightarrow} A_2^{\perp} \| B_1$, their expression doesn't project to an equivalent pair of separate components as with lenses.

6.3.1 The GoI adjunctions. For any game A there is a map of event structures with polarity

$$f_A : A \to A^+ \| A^-,$$

where A^+ is the projection of A to its +ve events and A^- is the projection to its −ve events: the map f_A acts as the identity function on events; it sends a configurations $x \in \mathscr{C}(A)$ to $f_A x = x^+ \| x^-$. It determines an adjunction $f_! \dashv f^*$ from A to $A^+ \| A^-$. Because the game A is race-free, both $f_{A!}$ and $f_A{}^*$ are deterministic strategies. This provides a lax functor from deterministic strategies in general, to those between GoI games. Let $\sigma : A {\longrightarrow} B$ be a deterministic strategy between games A and B. Defining $goi(\sigma) = f_{B!} \odot \sigma \odot f_A{}^*$ we obtain a deterministic strategy

$$goi(\sigma) : A^+ \| A^- {\longrightarrow} B^+ \| B^-.$$

Then, the strategy $goi(\sigma)$ corresponds to a stable function from $A^+\|B^-$ to $A^-\|B^+$, so to a GoI map. The operation goi only forms a lax functor however: for $\sigma : A \longrightarrow B$ and $\tau : B \longrightarrow C$, there is, in general, a nontrivial 2-cell $goi(\tau\odot\sigma) \Rightarrow goi(\tau)\odot goi(\sigma)$. This puts pay to goi being right adjoint to the inclusion functor in a pseudo adjunction from the category of GoI games to deterministic strategies. But, there is a *lax* pseudo adjunction, of potential use in abstract interpretation.

6.4 Dialectica games

Dialectica categories were devised in the late 1980's by Valeria de Paiva in her Cambridge PhD work with Martin Hyland [21]. The motivation then was to provide a model of linear logic underlying Kurt Gödel's *dialectica interpretation* of first-order logic [7]. They have come to prominence again recently because of a renewed interest in their maps in a variety of contexts, in formalisations of reverse differentiation and back propagation, open games and learners, and as an early occurrence of maps as lenses. The dialectica interpretation underpins most proof-mining techniques [23, 37].

We obtain a particular dialectica category, based on Berry's stable functions, as a full subcategory of deterministic strategies on dialectica games. Dialectica games are obtained as GoI games of imperfect information, intuitively by not allowing Player to see the moves of Opponent.

A *dialectica game* is a GoI game $A = A_1\|A_2$ with winning conditions, and with imperfect information given as follows. The imperfect information is determined by particularly simple order of access levels: $1 \prec 2$. All Player moves, those in A_1, are assigned to 1 and all Opponent moves, those in A_2, are assigned to 2. It is helpful to think of the access levels 1 and 2 as representing two rooms separated by a one-way mirror allowing anyone in room 2 to see through to room 1. In a dialectica game, Player is in room 1 and Opponent in room 2. Whereas Opponent can see the moves of Player, and in a counterstrategy make their moves dependent on those of Player, the moves of Player are made blindly, in that they cannot depend on Opponent's moves.

Although we are mainly interested in strategies *between* dialectica games it is worth pausing to think about strategies *in* a single dialectica game $A = A_1\|A_2$ with winning conditions W_A. Because Player moves cannot causally depend on Opponent moves, a deterministic strategy in A corresponds to a configuration $x \in \mathscr{C}(A_1)$; that it is winning means $\forall y \in \mathscr{C}(A_2). W_A(x,y)$. So to have a winning strategy for the dialectica game means

$$\exists x \in \mathscr{C}(A_1)\forall y \in \mathscr{C}(A_2). W_A(x,y).$$

Consider now a deterministic winning strategy σ from a dialectica game $A = A_1\|A_2$ with winning conditions W_A to another $B = B_1\|B_2$ with winning conditions W_B. Ignoring access levels, σ is also a deterministic strategy between GoI games, so corresponds to a pair of stable functions

$$f : \mathscr{C}(A_1) \times \mathscr{C}(B_2) \to \mathscr{C}(B_1) \text{ and } g : \mathscr{C}(A_1) \times \mathscr{C}(B_2) \to \mathscr{C}(A_2).$$

But moves in B_2 have access level 2, moves of B_1 access level 1; a causal dependency in the strategy σ of a move in B_1 on a move in B_2 would violate the access order $1 \prec 2$. That no move in B_1 can causally depend on a move in B_2 is reflected in the functional independence of f on its second argument. As a deterministic strategy between dialectica categories, σ corresponds to a pair of stable functions

$$f : \mathscr{C}(A_1) \to \mathscr{C}(B_1) \text{ and } g : \mathscr{C}(A_1) \times \mathscr{C}(B_2) \to \mathscr{C}(A_2),$$

which we can picture as:

$$
\begin{array}{ccc}
A_1 & \xrightarrow{\ f\ } & B_1 \\
 & \searrow\!\!{}^{g} & \\
A_2 & \longleftarrow & B_2
\end{array}
$$

That σ is winning means, for all $x \in \mathscr{C}(A_1), y \in \mathscr{C}(B_2)$,

$$W_A(x, g(x, y)) \implies W_B(f(x), y) \,.$$

Pairs of functions f, g satisfying this winning condition are precisely the maps of de Paiva's construction of a dialectica category from Berry's stable functions.

Such pairs of functions are the *lenses* of functional programming where they were invented to make composable local changes on data-structures [26, 46]. We recover their at-first puzzling composition from the composition of strategies. Let σ be a deterministic strategy from dialectica game A to dialectica game B; and τ a deterministic strategy from B to another dialectica game C. Assume σ corresponds to a pair of stable functions f and g, as above, and analogously that τ corresponds to stable functions f' and g'. Then, the composition of strategies $\tau \odot \sigma$ corresponds to the composition of lenses: with first component $f' \circ f$ and second component taking $x \in \mathscr{C}(A_1)$ and $y \in \mathscr{C}(C_2)$ to $g(x, g'(f(x), y))$.

THEOREM 6.5. *The bicategory of deterministic strategies on dialectica games with rigid 2-cells is equivalent to the dialectica category of [21] built on dI-domains and stable functions.*

Girard's variant. In the first half of de Paiva's thesis she concentrates on the construction of dialectica categories. In the second half, she follows up on a suggestion of Girard to explore a variant. This too is easily understood in the context of concurrent games: imitate the work of this section, with GoI games extended with imperfect information, but now with access levels modified to the *discrete* order on $1, 2$. Then the causal dependencies of strategies are further reduced and deterministic strategies from $A = A_1 \| A_2$ to $B = B_1 \| B_2$ correspond to pairs of stable functions

$$f : \mathscr{C}(A_1) \to \mathscr{C}(B_1) \text{ and } g : \mathscr{C}(B_2) \to \mathscr{C}(A_2) \,.$$

Combs. Discussions of causality in science, and quantum information in particular, are often concerned with what causal dependencies are feasible; then structures similar to orders of access levels are used to capture *one-way signalling*, as in dialectica games, and *non-signalling*, as in Girard's variant. In this vein, through another variation of games with imperfect information, we obtain the generalisation of lenses to *combs*, used in quantum architecture and information [14, 36]. Combs provide a common method for imposing higher-order structure on quantum circuits or string diagrams.

Combs arise as strategies between *comb games* which, at least formally, are an obvious generalisation of dialectica games; their name comes from their graphical representation as structures that look like (hair) combs, with teeth representing successive transformations from input to output. An *n-comb game*, for a natural number n, is an n-fold parallel composition $A_1 \| A_2 \| \cdots \| A_n$ of purely Player or purely Opponent games A_i of alternating polarity; it is a game of imperfect information associated with access levels $1 \prec 2 \prec \cdots \prec n$ with moves of component A_i having access level i. Dialectica games are 2-comb games with winning conditions.

Open games and learners. Open games and learners [25, 27] have recently been presented as *parameterised* lenses or optics, in the case of open games with some concept of equilibrium or winning condition [10]. As an example, we obtain a form of open game between dialectica games A and B as a strategy $A \| P \longrightarrow B$, where P is a dialectica game of which the configurations specify *strategy profiles*. A variation based on optimal strategies between dialectica games with payoff, following [17], introduces Nash equilibria and takes us into game-theory territory, and to a testing ground for open games and the notions being developed there.

6.5 Optics

Now we show that general, possibly nondeterministic, strategies between dialectica games are precisely *optics* [50, 54] based on stable spans [44, 55, 59]. Recall that a dialectica game comprises $A_1 \| A_2$ where A_1 is a purely Player game, all events of which have access level 1 and A_2 is a purely Opponent game with all events of access level 2, w.r.t. access order Λ specifying $1 \prec 2$. We ignore winning conditions.

Let A and B be dialectica games. Let Q be a purely Player Λ-game. Recall that nondeterministic strategies between purely Player games correspond to stable spans. Consider strategies

$$F : A_1 \rightarrowtail B_1 \| Q \text{ and } G : Q \| B_2^{\perp} \rightarrowtail A_2^{\perp} \,.$$

Then the strategies F and G are between purely Player games, so correspond to stable spans— Appendix B. As any causal dependencies of F or G respect Λ, they are Λ-strategies.

Hence the composition

$$A_1 \| B_2^{\perp} \xrightarrow{\ F \| B_2^{\perp}\ } B_1 \| Q \| B_2^{\perp} \xrightarrow{\ B_1 \| G\ } B_1 \| A_2^{\perp}$$

is also a Λ-strategy and, being between purely Player games, corresponds to a stable span. The composition, rearranges to a strategy

$$\sigma : A_1 \| A_2 \rightarrowtail B_1 \| B_2 \,,$$

which is a Λ-strategy, so to a strategy between the original dialectica games A and B. We call this strategy $\mathrm{optic}(F, G)$ and call (F, G) its *presentation* from A to B with *residual* Q. The terminology is apt, as we'll show strategies obtained in this way coincide with *optics* as usually defined. Presentations can be represented diagrammatically:

$$
\begin{array}{ccc}
A_1 & \xrightarrow{\quad F \quad} & B_1 \\
 & \searrow & \\
 & Q & \\
 & \nearrow & \\
A_2 & \xleftarrow{\quad G \quad} & B_2 \,,
\end{array}
$$

illustrating how F and G are "coupled" via the residual Q.

As usually defined, an optic is an equivalence class of presentations. Let (F, G) and (F', G') be presentations from A to B with residuals Q and Q' respectively. The equivalence relation $\sim$ on presentations is that generated by taking $(F, G) \sim (F', G')$ if, for some $f : Q \rightarrowtail Q'$, the following triangles commute

$$
\begin{array}{ccccc}
A_1 \xrightarrow{\ F'\ } B_1 \| Q' & & & Q' \| B_2 \xrightarrow{\ G'\ } B_1 & \\
\ \ \diagdown \quad \uparrow {\scriptstyle B_1 \| f} & & {\scriptstyle f \| B_2} \uparrow \quad \diagup & & \\
{\scriptstyle F} \ \diagdown \ \uparrow & & \uparrow \ \diagup \ {\scriptstyle G} & & \\
B_1 \| Q & & & Q \| B_2 \,. &
\end{array}
\qquad (\sim \text{def})
$$

Presentations of optics compose. Let A, B and C be dialectica games. Given a presentation (F, G) from A to B with residual Q and another (F', G') from B to C with residual P we obtain a presentation from A to C with residual $P \| Q$ guided by the diagram

$$
\begin{array}{ccccc}
A_1 & \xrightarrow{\ F\ } B_1 & \xrightarrow{\ F'\ } & C_1 \\
 & \searrow & \searrow & \\
 & Q & P & \\
 & \nearrow & \nearrow & \\
A_2 & \xleftarrow{\ G\ } B_2 & \xleftarrow{\ G'\ } & C_2 \,,
\end{array}
$$

precisely, as $((F' \| Q) \odot F, \ G \odot (Q \| G') \odot (s_{PQ} \| C_2))$, where s_{PQ} expresses the symmetry $P \| Q \cong Q \| P$.

Composition preserves $\sim$ and has the evident identity presentation, with residual the empty game. It follows that optic is functorial and that if $(F, G) \sim (F', G')$ then

$$\mathrm{optic}(F, G) \cong \mathrm{optic}(F', G') \,.$$

To show any strategy between container games is an optic, we exploit the monoidal-closure of stable spans—see Appendix B. A presentation (F, G) is $\sim$-equivalent to a *canonical presentation* (F', G') with residual $Q' = [B_2^\perp \multimap A_2^\perp]$ and G' as *application* apply: in ($\sim$def), take $f = \mathrm{curry}\, G$ and $F' = (B_1 \| f) \odot F$.

Now, strategies $\sigma : A \longmapsto B$, between dialectica games A and B, correspond to canonical presentations. To see this, ignoring the access levels for the moment, a general strategy

$$\sigma : A_1 \| A_2 \longmapsto B_1 \| B_2$$

corresponds to a strategy between purely Player games

$$\sigma_1 : A_1 \| B_2^\perp \longmapsto B_1 \| A_2^\perp \,,$$

so to a stable span. From the monoidal-closure of stable spans we can curry σ_1, to obtain a corresponding strategy

$$\sigma_2 : A_1 \longmapsto [B_2^\perp \multimap (B_1 \| A_2^\perp)]$$

with the property

$$\sigma_1 \cong \mathrm{apply}_{B_1 \| A_2^\perp} \odot (\sigma_2 \| B_2^\perp) \,.$$

Recalling the access levels, no event of B_1 can causally depend on an event of B_2, ensuring that σ_2 corresponds to

$$\sigma^+ : A_1 \longmapsto B_1 \| [B_2^\perp \multimap A_2^\perp]$$

where

$$\sigma_1 \cong \mathrm{apply}_{B_1 \| A_2^\perp} \odot (\sigma_2 \| B_2^\perp) \cong (B_1 \| \mathrm{apply}_{A_2^\perp}) \odot (\sigma^+ \| B_2^\perp) \,.$$

It follows that $(\sigma^+, \mathrm{apply}_{A_2^\perp})$ is a canonical presentation for which

$$\sigma \cong \mathrm{optic}(\sigma^+, \mathrm{apply}_{A_2^\perp}) \,,$$

giving a correspondence between strategies $\sigma : A \longmapsto B$ between dialectica games and canonical presentations $(\sigma^+, \mathrm{apply}_{A_2^\perp})$.

Via canonical presentations we obtain a bicategory of optics. Its objects are dialectica games. Its maps are stable spans $A_1 \longmapsto B_1 \| [B_2^\perp \multimap A_2^\perp]$, with the associated 2-cells, from dialectica game A to dialectica game B.

THEOREM 6.6. *The bicategories of strategies on dialectica games with rigid 2-cells and that of optics built on stable spans are equivalent.*

6.6 Containers

A *container game* is a game of imperfect information A w.r.t. access levels $1 < 2$; each Player move of A is sent to 1 and each Opponent move to 2. So in A the only causal dependencies between moves of different polarity are $\boxplus \le \boxminus$.

The configurations of a container game A have a dependent-type structure. Opponent moves can causally depend on Player moves, but not conversely. Let A_1 denote the subgame comprising the initial substructure of purely Player moves of A. A configuration $x \in \mathscr{C}(A_1)$ determines a subgame $A_2(x)$ comprising the substructure of A based on all those Opponent moves for which all the Player moves on which they depend appear in x. A configuration of A breaks down uniquely into a union $x \cup y$, so a pair (x, y), where $x \in \mathscr{C}(A_1)$ and $y \in \mathscr{C}(A_2(x))$. We can see the configurations of a container game A as forming a dependent sum $\Sigma_{x:A_1} A_2(x)$. In this way a container game

represents a container type, familiar from functional programming [2]; configurations x of A_1 are its "shapes," indexing "positions" $y \in A_2(x)$.[2]

We can of course extend a container game A with winning conditions which we identify with a property W_A of the dependent sum $\Sigma_{x:A_1} A_2(x)$. A deterministic winning strategy in the container game corresponds to a configuration $x \in \mathscr{C}(A_1)$ such that $\forall y \in \mathscr{C}(A_2(x)).\ W_A(x, y)$.

Strategies between container games respect $\preceq$ on access levels. A deterministic strategy σ from a container game A to a container game B corresponds to a map of container types, also called a *dependent lens*, having type

$$\Sigma_{f:[A_1 \to B_1]} \Pi_{x:A_1} \left[B_2(f(x)) \to A_2(x) \right] ; \qquad (*)$$

so σ corresponds to a pair of stable functions

$$f : [A_1 \to B_1] \ \text{ and } \ g : \Pi_{x:A_1} \left[B_2(f(x)) \to A_2(x) \right],$$

where we are using the function space, dependent sum and product of stable functions—see Appendix A.2. With winning conditions W_A and W_B, the strategy from A to B would be winning iff, for all $x \in \mathscr{C}(A_1), y \in \mathscr{C}(B_2(f(x)))$,

$$W_A(x, g_x(y)) \implies W_B(f(x), y).$$

The correspondence respects composition. Container types built on dI-domains and stable functions arise as a full subcategory of deterministic concurrent games.

THEOREM 6.7. *The bicategory of deterministic strategies on container games with rigid 2-cells is equivalent to a full subcategory of containers of dI-domains and stable functions [2].*

6.7 Dependent optics

What about general, nondeterministic, strategies between container games? A way to motivate their characterisation is to observe the isomorphism of the type of a dependent lens $(*)$ above with

$$\Pi_{x:A_1} \Sigma_{y:B_1} \left[B_2(y) \to A_2(x) \right].$$

It is this nonstandard way to present the type of lenses that generalises to the monoidal-closed bicategory of stable spans, once we move to the dependent product Π^s of stable spans—Appendix A.2.

Ignoring winning conditions, a general strategy between container games corresponds to a new form of optic. A *dependent optic* between container games, from A to B, is a stable span of type

$$\mathrm{dOp}[A, B] = \Pi^s_{x:A_1} \Sigma_{y:B_1} \left[B_2(y) \multimap A_2(x) \right],$$

so a rigid map into $\mathrm{dOp}[A, B]$. A 2-cell $f : F \Rightarrow F'$ between dependent optics $F, F' : \mathrm{dOp}[A, B]$ is a 2-cell of stable spans. Composition of dependent optics is the stable span

$$\circ : \mathrm{dOp}[B, C] \, \| \, \mathrm{dOp}[A, B] \rightarrowtail \mathrm{dOp}[A, C]$$

described by

$$G \circ F \coloneqq \lambda x : A_1.\ \text{let } (y, F') \Leftarrow F(x) \text{ in}$$
$$\text{let } (z, G') \Leftarrow G(y) \text{ in } (z, F' \odot G'),$$

where $F' \odot G' : [C_2(z) \multimap A_2(x)]$ is the composition of stable spans $G' : [C_2(z) \multimap B_2(y)]$ and $F' : [B_2(y) \multimap A_2(x)]$. The identity optic of container game A acts on $x : A_1$ to return the identity at the x-component of $\Sigma_{x:A_1} [A_2(x) \multimap A_2(x)]$.

The equivalence of strategies between container games with dependent optics, hinges on recasting $\mathrm{dOp}[A, B]$ as a strategy $do[A, B] : A \rightarrowtail B$ between container games A and B. Any strategy

---

[2]We won't treat symmetry in concurrent games at all here, but it is important in many applications. With the addition of symmetry, configurations form a nontrivial category, not merely a partial order based on inclusion [11].

between container games is of course a strategy where we forget the access levels. We can express that a strategy $\sigma : A \rightarrowtail B$ respects the access levels, so is truly a strategy between container games, precisely through the presence of a rigid 2-cell

$$A \underset{do[A,B]}{\overset{\sigma}{\Longrightarrow r}} B\,.$$

The 2-cell r is unique, making the strategy $do[A, B]$ terminal amongst strategies σ between container games, from A to B. By restricting r to Player moves we obtain the dependent optic $\sigma^+ :$ $\mathrm{dOp}[A, B]$ which corresponds to σ.

THEOREM 6.8. *The bicategory of strategies between container games, with rigid 2-cells, is equivalent to the bicategory of dependent optics.*

PROOF. (Sketch)

The proof relies on position functions of strategies [64]—see §4.5.1 of *op. cit.*. The *position function* d of a strategy $\sigma : S \to A^\perp \| B$ is given by $d(x) = \sigma[x]_S$ for $x \in \mathscr{C}(S^+)^o$; the event structure S^+ is the projection of S to its Player moves. A position function d is characterised as a union-preserving function $d : \mathscr{C}(S^+)^o \to \mathscr{C}(A)^o$ which restricts to a map $f : S^+ \to A^+$ of event structures that on $s \in S^+$ gives the unique event of Player amongst the $\leq_A$-maximal events in $d([s]_S)$.

We describe the strategy $do[A, B]$ associated with $\mathrm{dOp}[A, B]$ via its position function.

Recall that $\mathrm{dOp}[A, B]$ is built using Pr out of primes of the stable family

$$\Pi^s_{x \in \mathscr{C}(A_1)} \Sigma_{y \in \mathscr{C}(B_1)} \left[\mathscr{C}(B_2(y)) \multimap \mathscr{C}(A_2(x)) \right],$$

whose events take the form (x, b) or $(x, (y, a))$ where $x \in \mathscr{C}(A_1)^o$, $y \in \mathscr{C}(B_2)^o$, $b \in B_1$ and $a \in A_2$. The position function d_o takes a prime with top element (x, b) to $x\|[b]_B$ and one with top element $(x, (y, a))$ to $x\|(y \cup [a]_A)$.

Showing $do[A, B]$ is terminal amongst strategies $\sigma : S \to A^\perp \| B$ between container games, rests on there being a unique rigid map σ^+ such that

$$\begin{array}{ccc} S^+ & \overset{\sigma^+}{\longrightarrow} & \mathrm{dOp}[A, B] \\ & {\scriptstyle d}\searrow & \downarrow{\scriptstyle d_o} \\ & & A^\perp \| B \end{array}$$

commutes, where $d : S^+ \to A^\perp \| B$ is the position function of σ. The map $\sigma^+ = \mathrm{Pr}(\sigma_0)$, where σ_0 is the map of stable families

$$\sigma_0 : \mathscr{C}(S^+) \to \Pi_{x \in \mathscr{C}(A_1)} \Sigma_{y \in \mathscr{C}(B_1)} \left[\mathscr{C}(B_2(y)) \multimap \mathscr{C}(A_2(x)) \right],$$

defined as follows. For notational simplicity, assume A and B have disjoint sets of events so we can regard the events of $A\|B$ as $A \cup B$. Let $s \in S^+$. Either $\sigma(s) \in B_1$ or $\sigma(s) \in A_2$. Accordingly, define

$$\sigma_0(s) = \begin{cases} (x, b) & \text{if } \sigma(s) = b \in B_1 \ \& \ x = \sigma[s]_S^- ; \\ (x, (y, a)) & \text{if } \sigma(s) = a \in A_2 \ \& \ x = \sigma[s]_S^- \cap A_1 \\ & \& \ y = \sigma[s]_S^- \cap B_2 \,. \end{cases}$$

The fact that $do[B, C] \odot do[A, B] \cong do[A, C]$ is key in showing the correspondence of σ with σ^+ respects composition. □

The results on optics for container games specialise to those for dialectica games.

7 ENRICHMENT

Games and strategies support enrichments, to: probabilistic strategies, also with continuous distributions [49, 62]; quantum strategies [15]; and strategies on the reals [5]. The enrichments specialise to the cases above. Work on enriched concurrent strategies transfers to situations of interest in functional programming, domain theory and geometry of interaction. In explaining how, we can take advantage of a general method for enriching strategies.

The enrichments named above were developed individually and are not always the final story. For instance, the assignment of quantum operators to configurations of strategies in [15] is not functorial w.r.t. inclusion on configurations, a defect when it comes to understanding how the operator of a configuration is built up. The authors' remedy also achieves all the enrichments just named, now uniformly by the same construction.

The construction is w.r.t. a symmetric monoidal category $(\mathscr{M}, \otimes, I)$. For example, $\mathscr{M}$ can be the monoid $([0, 1], \cdot, 1)$ comprising the unit interval under multiplication (for probabilistic strategies); measurable spaces with Markov kernels (for probabilistic strategies with continuous distributions); CPM, finite-dimensional Hilbert spaces with completely positive maps (for quantum strategies); or Euclidean spaces with smooth maps, to support (reverse) differentiation.

We first extend $\mathscr{M}$ to allow interaction beyond that from argument to result. The *parameterised category* $\mathrm{Para}(\mathscr{M})$ has the same objects, now with maps $(P, f, Q) : X \to Y$ consisting of $f : X \otimes P \to Q \otimes Y$ in $\mathscr{M}$; the parameters P and Q allow input and output with the environment. Composition accumulates parameters: $(R, g, S) \circ (P, f, Q) := (P \otimes R, (Q \otimes g) \circ (f \otimes R), Q \otimes S)$. Then,

(1) moves a of a game A are assigned objects $\mathscr{H}(a)$ in $\mathscr{M}$, extended to $X \in \mathrm{Con}_A$ by $\mathscr{H}(X) := \bigotimes_{a \in X} \mathscr{H}(a)$. (Neutral moves, appearing in interaction, are assigned the tensor unit I.)

(2) an $\mathscr{M}$-enriched strategy $\sigma : S \to A$ is accompanied by a functor $\mathscr{Q} : (\mathscr{C}(S)^o, \subseteq) \to \mathrm{Para}(\mathscr{M})$. To an interval $x \subseteq x'$ in $\mathscr{C}(S)^o$ this assigns a parameterised map $\mathscr{Q}(x \subseteq x')$ from $\mathscr{Q}(x)$ to $\mathscr{Q}(x')$ with input parameters $\mathscr{H}(\sigma(x' \setminus x)^-)$ and output parameters $\mathscr{H}(\sigma(x' \setminus x)^+)$.
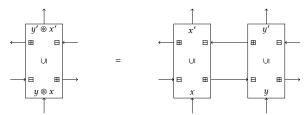
The assignment in **(2)** describes how the internal state is transformed in moving from x to x' under interaction with the environment through events $x' \setminus x$. The assignment in **(2)** is assumed *oblivious*, i.e. $\mathscr{Q}(x \subseteq^- x')$ is always an isomorphism in $\mathscr{M}$, expressing that all the input from $x' \setminus x$ is adjoined to the internal state $\mathscr{Q}(x)$ to produce a new internal state $\mathscr{Q}(x') = \mathscr{Q}(x) \otimes \mathscr{H}(\sigma(x' \setminus x))$. This is needed to ensure that the enriched version of copycat acts as identity w.r.t. composition.

In the quantum and probabilistic cases, observation is contextual, reflected in the presence of an extra *drop condition*, a form of inclusion-exclusion principle [15, 62]; it requires $\mathscr{M}$ be enriched over, at least, cancellative commutative monoids.

Moves, their positions, dependencies and polarities, orchestrate the functional dependency and dynamic linkage in composing enriched strategies. Consider an enriched strategy $\sigma : S \to A^\perp \| B$. In the enrichment the interval $x \subseteq x'$ of S is assigned a parameterised map $\mathscr{Q}(x \subseteq^- x')$ pictured below, in which the input parameters $P_A \otimes P_B$ and output parameters $Q_A \otimes Q_B$ have been factored into those over A and those over B:

Consider now the interaction of enriched strategies $\sigma : S \to A^\perp \| B$ and $\tau : T \to B^\perp \| C$. An interval $y \circledast x \subseteq y' \circledast x'$ in the interaction $T \circledast S$ breaks down into two intervals $x \subseteq x'$ of S and $y \subseteq y'$ of T. Their assignments compose together as shown to give the assignment to $y \circledast x \subseteq y' \circledast x'$:

For this composition to be well-defined we need that it involves no functional loops. But this is assured through the absence of causal loops in the interaction. Suppose $z \subseteq z'$ is an interval of $T \odot S$. The event structure $T \odot S$ is the projection of $T \circledast S$. Take $y \circledast x = [z]_{T \circledast S}$ and $y' \circledast x' = [z']_{T \circledast S}$ the down-closures of z and z' in $T \circledast S$. By definition, the interval $z \subseteq z'$ of $T \odot S$ is assigned the same parameterised map as $y \circledast x \subseteq y' \circledast x'$ in $T \circledast S$.

Enrichments achieved in this way specialise automatically to sub(bi)categories, and the functional cases we have considered, without needing extra demands on the category $\mathcal{M}$. Some of the specialisations are known. For example, stable spans when enriched by probability, via the monoid $([0, 1], \cdot, 1)$, become Markov kernels, and this enrichment extends to the various forms of optics we have uncovered. Others deserve further exploration. Enrichment w.r.t. CPM, yielding quantum strategies, specialises to nondeterministic strategies between GoI games. This provides an enrichment of Geometry of Interaction with quantum effects, and a likely candidate with which to give a semantics for the more operational, multi-token machine treatment of [38]. Concurrent games and strategies enrich with Euclidean spaces and (partial) smooth maps and via them connect with forwards and reverse differentiation. An easier subcase to explore first is the enrichment of stable functions; here one already encounters many of the issues of differential programming.

Enriched strategies provide a general framework in which to explore the interaction patterns of "functions" (maps in $\mathcal{M}$) and realisations of approaches to causal inference through string diagrams [33, 36].

8 CONCLUSION

Functional paradigms help tame the wild world of interactive computation. On the other hand, discovering the simplifying paradigms has often required considerable ingenuity, for example, by Gödel in his Dialectica Interpretation, or Girard in Geometry of Interaction.

The challenges to a functional approach are even more acute with enrichments, say to probabilistic, quantum or real number computation. The traditional categories of mathematics do not often support all the features required by computation. They often don't have function spaces or support recursion. Their extension to computational features has often to be dealt with separately, and ingeniously, for example, by replacing Borel spaces by quasi Borel spaces to support recursion and higher-order with probability [31] or the category CPM of completely positive maps by a completion for quantum lambda calculi [47]. Concurrent games and strategies provide enough computational infrastructure that traditional symmetric monoidal categories suffice (the unit interval for probability, Markov kernels for general distributions, CPM for quantum, or smooth maps for differentiation).

As a model of interaction, concurrent games and strategies are more technically challenging and require a new, more local, way of thinking. But, as has been demonstrated here, they can provide a broad general context for interaction which can be specialised to functional paradigms, also in

providing enrichments to probabilistic, quantum and real number computation, without requiring clever extensions to the traditional categories of mathematics.

Concurrent games and strategies can also provide a rationale for new definitions. The form of dependent optic described here appears to be new. It is *derived* as a characterisation of non-deterministic strategies between container games. Contrast this with the incomplete search for a categorical axiomatics of dependent optics described in the blog post [30]. This is not a criticism of axiomatisations but does make the obvious point that they are best guided by concrete examples—of which concurrent games and strategies and their enrichments are a rich source. (A broader characterisation of dependent optics would ensue if games carried symmetry; then the configurations of a game would form a proper category rather than a partial order.)

There is work to do, specifically in extending the work here to games with symmetry [11]. But a lot can be said for a single, expressive, intrinsically higher-order framework which readily adapts to enrichments.

One challenge is that of connecting concurrent games and strategies with the theory of effects [42, 51], specifically with understanding effect handlers [52] as concurrent strategies. Though superficially rather different, effect handlers and concurrent strategies have very similar roles: both are concerned with orchestrating the future of a computation contingent on its past and its environment. Through the work of this paper, the language of strategies outlined in Section 4.7 is able to express complex functional dependencies—see Section 7: how can it best be extended to existing theories of effects and effect handlers? As a beginning, the "detectors" of Example 5.5 extend to a form of "event handler." In the other direction, such an investigation should suggest ways to enhance effects and effect handlers to support richer forms of parallel computation.

ACKNOWLEDGMENTS

REFERENCES

[1] Martín Abadi, Michael Isard, and Derek Gordon Murray. 2017. A computational model for TensorFlow: an introduction. In *Proceedings of the 1st ACM SIGPLAN International Workshop on Machine Learning and Programming Languages, MAPL@PLDI 2017*. ACM, 1–7. https://doi.org/10.1145/3088525.3088527

[2] Michael Gordon Abbott, Thorsten Altenkirch, and Neil Ghani. 2005. Containers: Constructing strictly positive types. *Theor. Comput. Sci.* 342, 1 (2005), 3–27. https://doi.org/10.1016/j.tcs.2005.06.002

[3] Samson Abramsky, Esfandiar Haghverdi, and Philip J. Scott. 2002. Geometry of Interaction and Linear Combinatory Algebras. *Math. Struct. Comput. Sci.* 12, 5 (2002), 625–665. https://doi.org/10.1017/S0960129502003730

[4] Samson Abramsky and Radha Jagadeesan. 1994. New Foundations for the Geometry of Interaction. *Inf. Comput.* 111, 1 (1994), 53–119. https://doi.org/10.1006/inco.1994.1041

[5] Aurore Alcolei. 2019. *Enriched concurrent games : witnesses for proofs and resource analysis. (Jeux concurrents enrichis : témoins pour les preuves et les ressources).* Ph.D. Dissertation. University of Lyon, France. https://tel.archives-ouvertes.fr/tel-02448974

[6] Ross Street André Joyal and Dominic Verity. 1996. Traced monoidal categories. *Math. Proc. Camb. Phil. Soc. (1996), 119* (1996), 447–468.

[7] Jeremy Avigad and Solomon Feferman. 1999. Gödel's functional ("Dialectica") interpretation. (1999), 337–405.

[8] Gérard Berry. 1978. Stable Models of Typed lambda-Calculi. In *ICALP (Lecture Notes in Computer Science, Vol. 62)*. Springer, 72–89.

[9] S. Brookes, C. A. R. Hoare, and A. W. Roscoe. 1984. A Theory of Communicating Sequential Processes. *J. ACM* 31 (1984), 560–599.

[10] Matteo Capucci, Bruno Gavranović, Jules Hedges, and Eigil Fjeldgren Rischel. 2021. Towards foundations of categorical cybernetics. arXiv:2105.06332 [math.CT]

[11] Simon Castellan, Pierre Clairambault, and Glynn Winskel. 2014. Symmetry in concurrent games. In *Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS '14, Vienna, Austria, July 14 - 18, 2014.* ACM.

[12] Simon Castellan, Jonathan Hayman, Marc Lasson, and Glynn Winskel. 2014. Strategies as concurrent processes. *Electr. Notes Theor. Comput. Sci.* 308 (2014), 87–107.

[13] Simon Castellan and Nobuko Yoshida. 2019. Two sides of the same coin: session types and game semantics: a synchronous side and an asynchronous side. *Proc. ACM Program. Lang.* 3, POPL (2019), 27:1–27:29. https://doi.org/10.1145/3290340

[14] G. Chiribella, G. M. D'Ariano, and P. Perinotti. 2008. Quantum Circuit Architecture. *Physical Review Letters* 101, 6 (Aug 2008). https://doi.org/10.1103/physrevlett.101.060401

[15] Pierre Clairambault, Marc de Visme, and Glynn Winskel. 2019. Game semantics for quantum programming. *Proc. ACM Program. Lang.* 3, POPL (2019), 32:1–32:29. https://doi.org/10.1145/3290345

[16] Pierre Clairambault, Julian Gutierrez, and Glynn Winskel. 2012. The Winning Ways of Concurrent Games. In *LICS 2012: 235-244.*

[17] Pierre Clairambault and Glynn Winskel. 2013. On Concurrent Games with Payoff. *Electr. Notes Theor. Comput. Sci. 298: 71-92* (2013).

[18] John Conway. 2000. *On Numbers and Games.* Wellesley, MA: A K Peters.

[19] Thierry Coquand, Carl A. Gunter, and Glynn Winskel. 1987. DI-Domains as a Model of Polymorphism. In *Mathematical Foundations of Programming Language Semantics, 3rd Workshop, Tulane University, New Orleans, Louisiana, USA, April 8-10, 1987, Proceedings (Lecture Notes in Computer Science, Vol. 298)*, Michael G. Main, Austin Melton, Michael W. Mislove, and David A. Schmidt (Eds.). Springer, 344–363. https://doi.org/10.1007/3-540-19020-1_18

[20] Thierry Coquand, Carl A. Gunter, and Glynn Winskel. 1989. Domain Theoretic Models of Polymorphism. *Inf. Comput.* 81, 2 (1989), 123–167. https://doi.org/10.1016/0890-5401(89)90068-0

[21] Valeria de Paiva. 1988. *The Dialectica categories.* PhD Thesis, University of Cambridge.

[22] Claudia Faggian and Mauro Piccolo. 2009. Partial orders, event structures and linear strategies. In *TLCA '09 (LNCS, Vol. 5608).* Springer.

[23] Solomon Feferman. 1996. Kreisel's 'Unwinding Program', Kreiseliana: about and around Georg Kreisel. (1996), 247–273.

[24] Marcelo Fiore, Nicola Gambino, Martin Hyland, and Glynn Winskel. 2018. Relative pseudomonads, Kelisli bicategories and substitution monoidal structures. *Selecta Mathematica - New Series* 24, 3 (2018), 2791–2830.

[25] Brendan Fong, David I. Spivak, and Rémy Tuyéras. 2019. Backprop as Functor: A compositional perspective on supervised learning. In *34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019, Vancouver, BC, Canada, June 24-27, 2019.* IEEE, 1–13. https://doi.org/10.1109/LICS.2019.8785665

[26] J. Nathan Foster, Michael B. Greenwald, Jonathan T. Moore, Benjamin C. Pierce, and Alan Schmitt. 2007. Combinators for bidirectional tree transformations: A linguistic approach to the view-update problem. *ACM Trans. Program. Lang. Syst.* 29, 3 (2007), 17. https://doi.org/10.1145/1232420.1232424

[27] Neil Ghani, Jules Hedges, Viktor Winschel, and Philipp Zahn. 2018. Compositional Game Theory. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, Anuj Dawar and Erich Grädel (Eds.). ACM, 472–481. https://doi.org/10.1145/3209108.3209165

[28] Jean-Yves Girard. 1989. Geometry of interaction I: Interpretation of System F. In *Logic Colloquium '88*, C. Bonotto, S R. Ferro, Valentini, and A. Zanardo (Eds.). North-Holland, 221–260.

[29] Georges Gonthier, Martín Abadi, and Jean-Jacques Lévy. 1992. The Geometry of Optimal Lambda Reduction. In *Conference Record of the Nineteenth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Albuquerque, New Mexico, USA, January 19-22, 1992*, Ravi Sethi (Ed.). ACM Press, 15–26. https://doi.org/10.1145/143165.143172

[30] Jules Hedges. 2020. Towards dependent optics (Blog Post). https://julesh.com/2020/06/10/towards-dependent-optics/

[31] Chris Heunen, Ohad Kammar, Sam Staton, and Hongseok Yang. 2017. A convenient category for higher-order probability theory. In *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. IEEE. https://doi.org/10.1109/lics.2017.8005137

[32] Martin Hyland. 2010. Some reasons for generalising domain theory. *Mathematical Structures in Computer Science* 20, 2 (2010), 239–265.

[33] Bart Jacobs, Aleks Kissinger, and Fabio Zanasi. 2019. Causal Inference by String Diagram Surgery. In *Foundations of Software Science and Computation Structures - 22nd International Conference, FOSSACS 2019, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2019, Prague, Czech Republic, April 6-11, 2019, Proceedings (Lecture Notes in Computer Science, Vol. 11425)*, Mikolaj Bojanczyk and Alex Simpson (Eds.). Springer,

313–329. https://doi.org/10.1007/978-3-030-17127-8_18

[34] Niles Johnson and Donald Yau. 2020. 2-Dimensional Categories. arXiv:2002.06055 [math.CT]

[35] Andre Joyal. 1997. Remarques sur la théorie des jeux à deux personnes. *Gazette des sciences mathématiques du Québec, 1(4)* (1997).

[36] Aleks Kissinger and Sander Uijlen. 2017. A categorical semantics for causal structure. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017*. IEEE Computer Society, 1–12. https://doi.org/10.1109/LICS.2017.8005095

[37] Ulrich Kohlenbach. 2008. *Applied Proof Theory: Proof Interpretations and their Use in Mathematics.* Springer Monographs in Mathematics.

[38] Ugo Dal Lago, Claudia Faggian, Benoît Valiron, and Akira Yoshimizu. 2016. The Geometry of Parallelism. Classical, Probabilistic, and Quantum Effects. *CoRR* abs/1610.09629 (2016). arXiv:1610.09629 http://arxiv.org/abs/1610.09629

[39] Ian Mackie. 1995. The Geometry of Interaction Machine. In *Conference Record of POPL'95: 22nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, San Francisco, California, USA, January 23-25, 1995*, Ron K. Cytron and Peter Lee (Eds.). ACM Press, 198–208. https://doi.org/10.1145/199448.199483

[40] Paul-André Melliès and Samuel Mimram. 2007. Asynchronous games : innocence without alternation. In *CONCUR '07 (LNCS, Vol. 4703)*. Springer.

[41] Robin Milner. 1980. *A Calculus of Communicating Systems*. Lecture Notes in Computer Science, Vol. 92. Springer. https://doi.org/10.1007/3-540-10235-3

[42] Eugenio Moggi. 1989. Computational Lambda-Calculus and Monads. In *Proceedings of the Fourth Annual Symposium on Logic in Computer Science (LICS '89), Pacific Grove, California, USA, June 5-8, 1989*. IEEE Computer Society, 14–23. https://doi.org/10.1109/LICS.1989.39155

[43] Koko Muroya and Dan R. Ghica. 2019. The Dynamic Geometry of Interaction Machine: A Token-Guided Graph Rewriter. *Log. Methods Comput. Sci.* 15, 4 (2019). https://doi.org/10.23638/LMCS-15(4:7)2019

[44] Mikkel Nygaard. 2003. *Domain theory for concurrency.* PhD Thesis, Aarhus University. https://www.brics.dk/DS/03/13/BRICS-DS-03-13.pdf

[45] Mikkel Nygaard and Glynn Winskel. 2002. Linearity in Process Languages. In *LICS'02*. IEEE Computer Society.

[46] Frank J. Oles. 1982. *A category theoretic approach to the semantics of programming languages*. PhD Thesis, University of Syracuse.

[47] Michele Pagani, Peter Selinger, and Benoît Valiron. 2014. Applying Quantitative Semantics to Higher-Order Quantum Computing. *POPL '14* (2014), 647–658. https://doi.org/10.1145/2578855.2535879

[48] Hugo Paquet. 2020. *Probabilistic concurrent game semantics.* Ph. D. Dissertation. Computer Laboratory, University of Cambridge, UK.

[49] Hugo Paquet and Glynn Winskel. 2018. Continuous Probability Distributions in Concurrent Games. In *Proceedings of the Thirty-Fourth Conference on the Mathematical Foundations of Programming Semantics, MFPS 2018, Dalhousie University, Halifax, Canada, June 6-9, 2018 (Electronic Notes in Theoretical Computer Science, Vol. 341)*, Sam Staton (Ed.). Elsevier, 321–344. https://doi.org/10.1016/j.entcs.2018.11.016

[50] Matthew Pickering, Jeremy Gibbons, and Nicolas Wu. 2017. Profunctor Optics: Modular Data Accessors. *Art Sci. Eng. Program.* 1, 2 (2017), 7. https://doi.org/10.22152/programming-journal.org/2017/1/7

[51] Gordon D. Plotkin and A. John Power. 2004. Computational Effects and Operations: An Overview. *Electron. Notes Theor. Comput. Sci.* 73 (2004), 149–163. https://doi.org/10.1016/j.entcs.2004.08.008

[52] Gordon D. Plotkin and Matija Pretnar. 2009. Handlers of Algebraic Effects. In *Programming Languages and Systems, 18th European Symposium on Programming, ESOP 2009, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2009, York, UK, March 22-29, 2009. Proceedings (Lecture Notes in Computer Science, Vol. 5502)*, Giuseppe Castagna (Ed.). Springer, 80–94. https://doi.org/10.1007/978-3-642-00590-9_7

[53] Silvain Rideau and Glynn Winskel. 2011. Concurrent Strategies. In *LICS 2011*.

[54] Mitchell Riley. 2018. Categories of Optics. arXiv:1809.00738 [math.CT]

[55] Lucy Saunders-Evans and Glynn Winskel. 2007. Event Structure Spans for Nondeterministic Dataflow. *Electr. Notes Theor. Comput. Sci. 175(3): 109-129* (2007).

[56] Glynn Winskel. 1982. Event Structure Semantics for CCS and Related Languages. In *ICALP'82 (LNCS, Vol. 140)*. Springer, A full version is available from Winskel's Cambridge homepage.

[57] Glynn Winskel. 1986. Event Structures. In *Advances in Petri Nets (LNCS, Vol. 255)*. Springer, 325–392.

[58] Glynn Winskel. 2009. Prime algebraicity. *Theor. Comput. Sci.* 410, 41 (2009), 4160–4168. https://doi.org/10.1016/j.tcs.2009.06.015

[59] Glynn Winskel. 2011. Events, Causality and Symmetry. *Comput. J.* 54, 1 (2011), 42–57.

[60] Glynn Winskel. 2012. Deterministic concurrent strategies. *Formal Asp. Comput.* 24, 4-6 (2012), 647–660.

[61] Glynn Winskel. 2012. Winning, Losing and Drawing in Concurrent Games with Perfect or Imperfect Information. In *Festschrift for Dexter Kozen (LNCS, Vol. 7230)*. Springer.

[62] Glynn Winskel. 2013. Distributed Probabilistic and Quantum Strategies. *Electr. Notes Theor. Comput. Sci. 298: 403-425* (2013).

[63] Glynn Winskel. 2013. Strategies as profunctors. In *FOSSACS 2013 (Lecture Notes in Computer Science)*. Springer.

[64] Glynn Winskel. 2017. *ECSYM Notes: Event Structures, Stable Families and Concurrent Games.* http://www.cl.cam.ac.uk/~gw104/ecsym-notes.pdf

[65] Glynn Winskel and Mogens Nielsen. 1995. *Handbook of Logic in Computer Science 4.* OUP, Chapter Models for Concurrency, 1–148.

A DI-DOMAINS AND STABLE FUNCTIONS

That dI-domains are exactly the partial orders of configurations of an event structure was first published in [56]—see the extended version or [58] for the proof. A *stable* function between dI-domains is a Scott continuous function (*i.e.* preserves least upper bounds of directed sets) which preserves greatest lower bounds of compatible pairs of elements. Gérard Berry developed stable domain theory axiomatically, following operational guidelines [8]. For the reader's convenience, we include the constructions on event structures which realise the cartesian-closure of dI-domains and their dependent types.

A.1 Stable function space

Berry's cartesian-closed category of dI-domains[3] and stable functions can be presented as an equivalent category of event structures [57]. We summarise the product and stable function space constructions on stable families and event structures.

Let $\mathscr{A}$ and $\mathscr{B}$ be stable families with events A and B respectively. The product of their domains of configurations is easily realised as a simple parallel composition: $\mathscr{A} \| \mathscr{B} := \{x\|y \mid x \in \mathscr{A}\ \&\ y \in \mathscr{B}\}$.

We construct the stable function space of domains as a stable family $[\mathscr{A} \to \mathscr{B}]$. The stable family $[\mathscr{A} \to \mathscr{B}]$ comprises those $f \subseteq \mathscr{A}^o \times B$ for which, for all $x \in \mathscr{A}$,

- $\{b \mid \exists x' \subseteq x.\ (x', b) \in f\} \in \mathscr{B}$ and
- if $(x', b), (x'', b) \in f$ with $x', x'' \subseteq x$ then $x' = x''$.

THEOREM A.1. *The construction $[\mathscr{A} \to \mathscr{B}]$ above is a stable family with $([\mathscr{A} \to \mathscr{B}], \subseteq)$ order isomorphic to $[(\mathscr{A}, \subseteq) \to (\mathscr{B}, \subseteq)]$, the stable function space of stable functions, ordered by the stable order, between dI-domains $(\mathscr{A}, \subseteq)$ and $(\mathscr{B}, \subseteq)$.*

Given event structures A and B, we define

$$[A \to B] := \Pr([\mathscr{C}(A) \to \mathscr{C}(B)]).$$

The configurations of $[A \to B]$ under inclusion are isomorphic to the stable function space of dI-domains $(\mathscr{C}(A), \subseteq)$ and $(\mathscr{C}(B), \subseteq)$.

A.2 Dependent-type constructions

We base the constructions here on [19, 20] (though with the simplification w.l.o.g. of using the substructure relation $\unlhd$ between stable families [56, 57] in place of rigid embeddings between dI-domains). Recall the definition $\mathscr{A} \unlhd \mathscr{B}$, where $\mathscr{A}$ and $\mathscr{B}$ are stable families with events A and B respectively:

$$\mathscr{A} \unlhd \mathscr{B}\ \text{iff}\ A \subseteq B\ \text{and}$$

$$\forall x.\ x \in \mathscr{A} \iff x \subseteq A\ \&\ x \in \mathscr{B}.$$

---

[3]Strictly speaking, Berry defined dI-domains to have a countable basis of finite elements. Countability plays no role in the work here and we shall not impose it.

The relation $\mathscr{A} \trianglelefteq \mathscr{B}$ specifies a rigid embedding from the dI-domain $(\mathscr{A}, \subseteq)$ to the dI-domain $(\mathscr{B}, \subseteq)$ with projection $y \mapsto y \cap A$ from $\mathscr{B}$ to $\mathscr{A}$.[4]

For event structures A and B, we write $A \trianglelefteq B$ when $\mathscr{C}(A) \trianglelefteq \mathscr{C}(B)$. On event structures, $A \trianglelefteq B$ is equivalent to the events of A being included in those of B with

$$\forall a \in A.\ [a]_A = [a]_B \ \text{ and } \ \forall X \subseteq A.\ X \in \text{Con}_A \iff X \in \text{Con}_B\,.$$

Let $\mathscr{A}$ be a stable family. Let $\mathscr{B}(\_)$ be a *stable functor* from the partial-order category $(\mathscr{A}, \subseteq)$ to the (large) partial-order category of stable families related by $\trianglelefteq$. We shall write $B(x)$ for the events of $\mathscr{B}(x)$, where $x \in \mathscr{A}$. That the functor is *stable* means it is continuous and preserves pullbacks, which in this case means it is a function which preserves least upper bounds of directed sets and greatest lower bounds of compatible pairs. Correspondingly, for an event structure A, a functor from $x \in \mathscr{C}(A)$ to event structures $B(x)$ is *stable* when it is continuous and preserves pullbacks w.r.t. $\trianglelefteq$ on event structures.

Dependent sum

$\Sigma_{x \in \mathscr{A}}\, \mathscr{B}(x)$ is the stable family

$$\{x \| y \mid x \in \mathscr{A} \ \& \ y \in \mathscr{B}(x)\}\,.$$

PROPOSITION A.2. $\Sigma_{x \in \mathscr{A}}\, \mathscr{B}(x)$ *is a stable family with configurations corresponding to pairs* (x, y), *where* $x \in \mathscr{A}$ *and* $y \in \mathscr{B}(x)$; *the order of configurations corresponds to the coordinatewise order on pairs.*

We shall describe a typical configuration of $\Sigma_{x \in \mathscr{A}}\, \mathscr{B}(x)$ as a pair (x, y) where $x \in \mathscr{A}$ and $y \in \mathscr{B}(x)$. It's often convenient to describe an operation on configurations of the dependent sum in terms of their decomposition into pairs.

For an event structure A and $B(x)$, stable in $x \in \mathscr{C}(A)$,

$$\Sigma_{x:A}\, B(x) \coloneqq \text{Pr}(\Sigma_{x \in \mathscr{C}(A)}\, \mathscr{C}(B(x)))\,.$$

By analysing the structure of the prime configurations of $\Sigma_{x \in \mathscr{A}}\, \mathscr{B}(x)$, we can see that the event structure $\Sigma_{x:A}.\, B(x)$ is isomorphic to the event structure comprising

- *events*, consisting of the set of $a \in A$ in disjoint union with the set of pairs (x, b), where $x \in \mathscr{C}(A)^o$ is a smallest configuration for which $b \in B(x)$;
- *causal dependency*, that generated by the relations on events

$$a' \leq_A a\,,$$
$$(x', b') \leq (x, b) \ \text{ if } \ x' \subseteq x \ \& \ b' \leq_B b\,, \text{ and}$$
$$a \leq (x, b) \ \text{ if } \ a \in x\,;$$

- *consistency*, a finite subset of events,

$$\{a_i \mid i \in I\} \cup \{(x_j, b_j) \mid j \in J\} \in \text{Con iff}$$
$$\{a_i \mid i \in I\} \in \text{Con}_A \ \&$$
$$\bigcup_{j \in J} x_j \in \mathscr{C}(A)^o \ \& \ \{b_j \mid j \in J\} \in \text{Con}_B \ \&$$
$$\forall j, k \in J.\ b_j = b_k \implies x_j = x_k\,.$$

---

[4] The relation $\trianglelefteq$ does not have least upper bounds in general; there can be distinct minimal upper bounds.

Dependent product

The obvious projection from $\Sigma_{x\in\mathscr{A}}\,\mathscr{B}(x)$ to $\mathscr{A}$ is a simple form of Grothendieck fibration. We obtain $\Pi_{x\in\mathscr{A}}\,\mathscr{B}(x)$ as a stable family whose configurations correspond to stable sections of the fibration, *i.e.* stable functions from $(\mathscr{A}, \subseteq)$ to $(\Sigma_{x\in\mathscr{A}}\,\mathscr{B}(x), \subseteq)$ which send $x \in \mathscr{A}$ to a configuration $x\|y$ where $y \in \mathscr{B}(x)$. To this purpose, we can refashion the construction of the stable function space of Section A.1 to restrict to stable functions which are sections.

$\Pi_{x\in\mathscr{A}}\,\mathscr{B}(x)$ is the stable family comprising those sets

$$f \subseteq \{(x, b) \mid x \in \mathscr{A}^o \ \& \ b \in B(x)\}$$

for which, for all $x \in \mathscr{A}$,

- $\{b \mid \exists x' \subseteq x.\ (x', b) \in f\} \in \mathscr{B}(x)$ and
- if $(x', b), (x'', b) \in f$ with $x', x'' \subseteq x$ then $x' = x''$.

When $\mathscr{B}(x)$ is constantly $\mathscr{B}$, for all $x \in \mathscr{A}$, we observe that

$$\Pi_{x\in\mathscr{A}}\,\mathscr{B}(x) = [\mathscr{A} \to \mathscr{B}]\,.$$

THEOREM A.3. *The configurations of $\Pi_{x\in\mathscr{A}}\,\mathscr{B}(x)$ correspond to stable sections of $\Sigma_{x\in\mathscr{A}}\,\mathscr{B}(x)$; inclusion between configurations corresponds to the stable order on sections.*

Hence we can describe a typical configuration of $\Pi_{x\in\mathscr{A}}\,\mathscr{B}(x)$ as a stable section, using λ-notation, as $\lambda x \in \mathscr{A}.\ f(x)$, provided $f(x) \in \mathscr{B}(x)$ is stable in $x \in \mathscr{A}$; we obtain its components by function application.

For event structures A and $B(x)$, stable in $x \in \mathscr{C}(A)$, define

$$\Pi_{x:A}\,B(x) \coloneqq \mathrm{Pr}(\Pi_{x\in\mathscr{C}(A)}\,\mathscr{C}(B(x)))\,.$$

B STABLE SPANS

Stable spans are monoidal-closed—see [44]§7.5. Their tensor is given by the simple parallel composition of event structures. We define the function space in slightly greater generality, between stable families.

Let $\mathscr{A}$ and $\mathscr{B}$ be stable families. We construct the function space of stable spans as a stable family. The stable family $[\mathscr{A} \multimap \mathscr{B}]$ comprises those $F \subseteq \mathscr{A}^o \times B$ for which

- $\bigcup \{x \mid \exists b.\ (x, b) \in F\} \in \mathscr{A}$,
- $\forall x \in \mathscr{A}.\ \{b \mid \exists x' \subseteq x.\ (x', b) \in F\} \in \mathscr{B}$ and
- $\forall (x, b), (x', b) \in F.\ x = x'$.

It can be checked that $[\mathscr{A} \multimap \mathscr{B}]$ is a stable family. For event structures A and B, define $[A \multimap B] \coloneqq \mathrm{Pr}([\mathscr{C}(A) \multimap \mathscr{C}(B)])$. The configurations of $[A \multimap B]$ represent the possible paths the computation of output in B from input in A can follow.

By broadening to nondeterministic computation we can often regard types as special maps. For example $[A \multimap B]$ becomes a stable span with the obvious demand and rigid map. As such it is terminal within all stable spans from A to B: for any span S, d, r there is a unique 2-cell as shown

$$
\begin{array}{ccc}
 & S & \\
d \swarrow & \downarrow & \searrow r \\
A \leftarrow & [A \multimap B] & \rightarrow B\,.
\end{array}
$$

Deterministic stable spans coincide with stable functions.

B.1 Dependent product for stable spans

The dependent product for stable spans is a refashioning of the definition of their function space, to take account of the dependency of $\mathscr{B}(x)$ on $x \in \mathscr{A}$. The stable sections of the previous dependent product above are replaced by stable spans, so giving a form of nondeterministic dependent product.

$\Pi^s_{x \in \mathscr{A}} \mathscr{B}(x)$ is the stable family comprising those sets

$$F \subseteq \{(x,b) \mid x \in \mathscr{A}^o \ \& \ b \in B(x)\}$$

for which

- $\bigcup \{x \mid \exists b. \ (x,b) \in F\} \in \mathscr{A}$,
- $\forall x \in \mathscr{A}. \ \{b \mid \exists x' \subseteq x. \ (x',b) \in F\} \in \mathscr{B}(x)$ and
- $\forall (x,b), (x'b) \in F. \ x = x'$.

When $\mathscr{B}(x)$ is constantly $\mathscr{B}$, for all $x \in \mathscr{A}$, we observe that

$$\Pi^s_{x \in \mathscr{A}} \mathscr{B}(x) = [\mathscr{A} \multimap \mathscr{B}].$$

PROPOSITION B.1. *The configurations of $\Pi^s_{x \in \mathscr{A}} \mathscr{B}(x)$ correspond to stable sections $f : X_0 \to \Sigma_{x \in X_0} \mathscr{B}(x)$, where $X_0 = \{x \in \mathscr{A} \mid x \subseteq x_0\}$ for some $x_0 \in \mathscr{A}$, and for all $x \in X_0$, writing $f(x) = (x, f'(x))$ and $f(x_0) = (x_0, f'(x_0))$, if $f'(x) = f'(x_0)$ then $x = x_0$.*

For event structures A and $B(x)$, stable in $x \in \mathscr{C}(A)$, define

$$\Pi^s_{x:A} B(x) := \Pr(\Pi^s_{x \in \mathscr{C}(A)} \mathscr{C}(B(x))).$$

C PROOFS FOR SECTION 5

THEOREM C.1. *Let $f : \mathscr{C}(A) \to \mathscr{C}(B)$ be an affine-stable map between event structures with polarity A and B. Then*

$$\mathscr{F} := \{x \| y \in \mathscr{C}(A^\perp \| B) \mid y \sqsubseteq_B f(x)\}$$

is an infinitary stable family. The map $top : \Pr(\mathscr{F}) \to A^\perp \| B$ is a strategy $f_! : A \rightarrowtail B$. The strategy $f_!$ is deterministic if A and B are race-free and f reflects $-$-compatibility, i.e. $x \subseteq^- x_1$ and $x \subseteq^- x_2$ in $\mathscr{C}(A)$ and $fx_1 \cup fx_2 \in \mathscr{C}(B)$ implies $x_1 \cup x_2 \in \mathscr{C}(A)$.

PROOF. In the proof we make frequent use of the following observations. Let B be an event structure with polarity. Let $y_i \sqsubseteq_B y'_i$, for all $i \in I$. Then, (with I nonempty),

$$\bigcap_{i \in I} y_i \sqsubseteq_B \bigcap_{i \in I} y'_i.$$

When both $\{y_i \mid i \in I\}$ and $\{y'_i \mid i \in I\}$ are compatible in $\mathscr{C}(B)$,

$$\bigcup_{i \in I} y_i \sqsubseteq_B \bigcup_{i \in I} y'_i.$$

We first show $\mathscr{F}$ is a stable family.
Completeness: Let $\{x_i \| y_i \mid i \in I\}$ be a finitely compatible subset in $\mathscr{F}$. From compatibility, it follows that $\bigcup_{i \in I} x_i$ and $\bigcup_{i \in I} y_i$ are configurations. By assumption $y_i \sqsubseteq_B f(x_i)$, for all $i \in I$, so

$$\bigcup_{i \in I} y_i \sqsubseteq_B \bigcup_{i \in I} f(x_i) \subseteq^+ f(\bigcup_{i \in I} x_i).$$

As the relation $\subseteq^+$ is included in $\sqsubseteq_B$, by the latter's transitivity we obtain

$$\bigcup_{i \in I} y_i \sqsubseteq_B f(\bigcup_{i \in I} x_i),$$

so

$$\bigcup_{i \in I} (x_i \| y_i) = (\bigcup_{i \in I} x_i \| \bigcup_{i \in I} y_i) \in \mathscr{F} \,.$$

Stability: Let $\{x_i \| y_i \mid i \in I\}$ be a nonempty compatible subset in $\mathscr{F}$. By assumption $y_i \sqsubseteq_B f(x_i)$, for all $i \in I$, so

$$\bigcap_{i \in I} y_i \sqsubseteq_B \bigcap_{i \in I} f(x_i) \supseteq^- f(\bigcap_{i \in I} x_i)$$

—it follows from the assumptions that $\{x_i \mid i \in I\}$ is a nonempty compatible family in $\mathscr{C}(A)$, as is required to apply the stability of f. As $\supseteq^-$ is included in $\sqsubseteq_B$, we deduce

$$\bigcap_{i \in I} (x_i \| y_i) = (\bigcap_{i \in I} x_i \| \bigcap_{i \in I} y_i) \in \mathscr{F} \,.$$

Finiteness: If $x \| y$ in the family $\mathscr{F}$, then $x \in \mathscr{C}(A)$ and $y \in \mathscr{C}(B)$ with $y \sqsubseteq_B f(x)$. An element in $x \| y$ is either $(1, a)$ where $a \in x$ or $(2, b)$ where $b \in y$. We analyse these two cases.

Case $a \in x$. Observe the set $f([a])^-$ is finite by $-$-image finiteness. It follows that $[f([a])^-] \in \mathscr{C}(B)^o$ is a finite configuration of B for which

$$[f([a])^-] \subseteq^+ f[a] \,, \text{ so } [f([a])^-] \sqsubseteq_B f[a] \,.$$

As also $y \sqsubseteq_B f(x)$ we have

$$y \cap [f([a])^-] \sqsubseteq_B f(x) \cap f[a] = f[a] \,,$$

whence

$$[a] \| (y \cap [f([a])^-]) \in \mathscr{F}$$

creating a finite subconfiguration of $x \| y$ containing $(1, a)$.

Case $b \in y$. We prove a stronger result than is strictly needed for this part of the proof, in preparation for the proof of coincidence-freeness later. Letting $b \in y$, take

$$x_0 := \bigcap \{x' \in \mathscr{C}(A) \mid [b]^+ \subseteq f(x') \ \& \ x' \subseteq x\} \,.$$

By the stability of f,

$$f(x_0) \subseteq^- \bigcap \{f(x') \mid x' \in \mathscr{C}(A) \ \& \ [b]^+ \subseteq f(x') \ \& \ x' \subseteq x\} \,.$$

Thus

$$[b]^+ \subseteq f(x_0) \,,$$

and x_0 is the minimum subconfiguration of x for which $[b]^+ \subseteq f(x_0)$. By $+$-continuity, x_0 is a finite configuration. Also

$$[f(x_0)^-] \subseteq^+ f(x_0)$$

where the configuration $[f(x_0)^-]$ is also finite by $-$-image finiteness. We observe that all the $\leq$-maximal events in x_0 are $+$ve: supposing otherwise, there is a $\leq$-maximal $-$ve event in x_0 so a configuration $x_0' \subsetneq^- x_0$; then, as f preserves polarity, $[b]^+ \subseteq f(x_0) \subseteq^- f(x_0')$ so $[b]^+ \subseteq f(x_0')$, contradicting the minimality of x_0. Whatever the polarity of b we obtain

$$[f(x_0)^-] \cup [b] \supseteq^- [f(x_0)^-] \cup [[b]^+] \subseteq^+ f(x_0) \,,$$

so

$$[f(x_0)^-] \cup [b] \sqsubseteq_B f(x_0) \,.$$

We now show that $b \notin [f(x_0)^-]$ by cases on the polarity of b.

Suppose $pol_b(b) = +$. In this case $[b] = [[b]^+]$ and x_0 is the minimum subconfiguration of x such that $b \in f(x_0)$. If $x_0 = \emptyset$, by affinity, in the case of the empty family, we have $\emptyset \subseteq^+ f(\emptyset)$ which ensures $[f(x_0)^-]$ is empty, so does not contain b. Otherwise, the $\leq$-maximal events in x_0

are +ve and there is a subconfiguration $x_0' \subseteq^+ x_0$. As f respects polarity, $f(x_0') \subseteq^+ f(x_0)$. Hence $f(x_0)^- \subseteq f(x_0')$ so $[f(x_0)^-] \subseteq^+ f(x_0')$. From the minimality of x_0, we must have $b \notin f(x_0')$, so we also have $b \notin [f(x_0)^-]$, as required.

Suppose $pol_B(b) = -$. We show $b \notin f(x_0)$, from which $b \notin [f(x_0)^-]$ follows directly. Suppose otherwise that $b \in f(x_0)$. If x_0 is empty, we have $\emptyset \subseteq^+ f(\emptyset) = f(x_0)$, contradicting the polarity of b. When x_0 is nonempty, as the $\leq$-maximal events in x_0 are +ve, we must have a strictly smaller subconfiguration $x_0' \subseteq^+ x_0$. But then as f respects polarity $f(x_0') \subseteq^+ f(x_0)$. As b is −ve, $b \in f(x_0')$ making $[b]^+ \subseteq f(x_0')$, which contradicts the minimality of x_0. This shows $b \notin f(x_0)$, as required to obtain $b \notin [f(x_0)^-]$.

To complete the proof of the finiteness property, observe that $y \sqsubseteq_B f(x)$ with $[f(x_0)^-] \cup [b] \sqsubseteq_B f(x_0)$ entail

$$y \cap ([f(x_0)^-] \cup [b]) \sqsubseteq_B f(x) \cap f(x_0) = f(x_0) \,.$$

It follows that

$$x_0 \| (y \cap ([f(x_0)^-] \cup [b])) \in \mathscr{F} \,,$$

so yielding a finite subconfiguration of $x \| y$ containing $(2, b)$. We note for later that x_0 is the minimum subconfiguration of x for which $[b]^+ \subseteq f(x_0)$ and from this it follows that

$$b \notin [f(x_0)^-] \quad \text{with} \quad [f(x_0)^-] \cup [b] \sqsubseteq_B f(x_0) \,.$$

Coincidence-free: Let $x \| y \in \mathscr{F}$. Consider two distinct events in $x \| y$. There are three cases: they belong to the same component x; they belong to the same component y; or they belong to different components.

If they both belong to the same x-component, from the argument above they are $(1, a_1)$ and $(1, a_2)$ and belong to the respective subconfigurations

$$[a_1] \| (y \cap [f([a_1])^-]) \quad \text{and} \quad [a_2] \| (y \cap [f([a_2])^-])$$

of $x \| y$. If a_1 and a_2 are distinct, one of the subconfigurations must separate them in the sense of containing one but not the other.

Assume they both belong to the same y-component, one being $(2, b_1)$ and the other $(2, b_2)$, with $b_1, b_2 \in y$. From the proof of the finiteness part above, they belong to respective subconfigurations of $x \| y$ of the form

$$x_1 \| (y \cap ([f(x_1)^-] \cup [b_1])) \quad \text{and} \quad x_2 \| (y \cap ([f(x_2)^-] \cup [b_2]))$$

where x_1 is the minimum subconfiguration of x for which $[b_1]^+ \subseteq f(x_1)$ and x_2 is the minimum subconfiguration of x for which $[b_2]^+ \subseteq f(x_2)$. Recall from earlier that

$$b_1 \notin [f(x_1)^-] \quad \text{with} \quad [f(x_1)^-] \cup [b_1] \sqsubseteq_B f(x_1) \quad \text{and}$$
$$b_2 \notin [f(x_2)^-] \quad \text{with} \quad [f(x_2)^-] \cup [b_2] \sqsubseteq_B f(x_2) \,.$$

Imagine the two subconfigurations of $x \| y$ above do not separate $(2, b_1)$ and $(2, b_2)$, *i.e.*

$$(2, b_2) \in x_1 \| (y \cap ([f(x_1)^-] \cup [b_1])) \quad \text{and}$$
$$(2, b_1) \in x_2 \| (y \cap ([f(x_2)^-] \cup [b_2])) \,.$$

Then

$$b_2 \in [f(x_1)^-] \cup [b_1] \sqsubseteq_B f(x_1) \quad \text{and}$$
$$b_1 \in [f(x_2)^-] \cup [b_2] \sqsubseteq_B f(x_2) \,.$$

By the properties of $\sqsubseteq_B$, we see that $[b_2]^+ \subseteq f(x_1)$ and $[b_1]^+ \subseteq f(x_2)$. From the minimality properties of x_1 and x_2 we deduce that $x_1 = x_2$. Writing $x_0 := x_1 = x_2$ and recalling $b_1, b_2 \notin [f(x_0)^-]$ we obtain $b_1 \in [b_2]$ and $b_2 \in [b_1]$, so $b_1 = b_2$. Hence distinct $(2, b_1)$ and $(2, b_2)$ are separated by the chosen subconfigurations of $x \| y$.

Assume the two distinct events in $x\|y$ belong to different components, one being $(1, a)$, with $a \in x$, and the other $(2, b)$, with $b \in y$. If $b \notin f([a])$ then one argues, as frequently above, that $f([a]) \sqsubseteq_B f([a])$ together with $y \sqsubseteq_B f(x)$ gives $y \cap f([a]) \sqsubseteq_B f([a])$ yielding $[a]\|(y \cap f([a]))$ a subconfiguration of $x\|y$, which moreover contains $(1, a)$ but not $(2, b)$. Thus suppose $b \in f([a])$. If $b \in f([a))$ then $[a]\|(y \cap f([a)))$ is a subconfiguration of $x\|y$ which contains $(2, b)$ but not $(1, a)$. The remaining case is when $b \in f([a])$ and $b \notin f([a))$. Then $[a) \xrightarrow{a} \subset [a]$ and $b \in f([a]) \setminus f([a))$.

If $pol_A(a) = +$ then, as f respects polarity,

$$f([a)) \subseteq^+ f([a]), \text{ so } f([a)) \sqsubseteq_B f([a]).$$

By the now familiar argument, this yields $[a]\|(y \cap f[a))$ a subconfiguration of $x\|y$ containing $(1, a)$ but not $(2, b)$.

Similarly, if $pol_A(a) = -$ then

$$f([a)) \subseteq^- f([a]), \text{ so } f([a]) \sqsubseteq_B f([a)),$$

yielding a subconfiguration $[a)\|(y \cap f[a])$ of $x\|y$ which contains $(2, b)$ but not $(1, a)$.

This completes the proof of coincidence-freeness.

We check the map $top : \mathrm{Pr}(\mathscr{F}) \to A^\perp\|B$ is a strategy. Observe that

$$x' \sqsupseteq_A x \; \& \; x\|y \in \mathscr{F} \; \& \; y \sqsupseteq_B y' \implies x'\|y' \in \mathscr{F}$$

as the l.h.s. clearly entails

$$y' \sqsubseteq_B y \sqsubseteq_B f(x) \sqsubseteq_B f(x'),$$

so the r.h.s.. In particular, when $x\|y \in \mathscr{F}$ and $(x'\|y') \in \mathscr{C}(A^\perp\|B)$,

if $(x\|y) \subseteq^- (x'\|y')$, then $(x'\|y') \in \mathscr{F}$; and

if $(x'\|y') \subseteq^+ (x\|y)$, then $(x'\|y') \in \mathscr{F}$.

Thus the composite map

$$\mathscr{C}(\mathrm{Pr}(\mathscr{F})) \to \mathscr{F} \hookrightarrow \mathscr{C}(A^\perp\|B)$$

of stable families, where the first map is top and the second is an inclusion, satisfies the "lifting" conditions needed of a strategy—see [64], ensuring that $top : \mathrm{Pr}(\mathscr{F}) \to A^\perp\|B$ is a strategy.

Assume now that A and B are race-free and that f reflects $-$-compatibility. As $A^\perp\|B$ is now also race-free, to show $f_!$ a deterministic strategy it suffices to show that any two +ve event increments of a configuration in $\mathscr{F}$ are compatible in $\mathscr{F}$, i.e. if $x\|y \subset^+ x_1\|y_1$ and $x\|y \subset^+ x_2\|y_2$ in $\mathscr{F}$, then $(x_1 \cup x_2)\|(y_1 \cup y_2) \in \mathscr{F}$. Consider cases.

If the increments are $y \xrightarrow{b_1} \subset y_1$ and $y \xrightarrow{b_2} \subset y_2$, then b_1 and b_2 are +ve in B. Because each $y_i \sqsubseteq_B f(x)$, i.e. $y_i \sqsupseteq^- z \subseteq^+ f(x)$ where $z = y \cap f(x)$, we see both $b_1 \in f(x)$ and $b_2 \in f(x)$. Hence $z \cup \{b_1, b_2\} \in \mathscr{C}(B)$. Because B is race-free we obtain $y_1 \cup y_2 \in \mathscr{C}(B)$. Checking $y_1 \cup y_2 \sqsubseteq_B f(x)$, ensures $x\|(y_1 \cup y_2) \in \mathscr{F}$.

If the increments are $x \xrightarrow{a_1} \subset x_1$ and $x \xrightarrow{a_2} \subset x_2$ then a_1 and a_2 are $-$ve in A with $y \sqsubseteq_B f(x_1)$ and $y \sqsubseteq_B f(x_2)$. It follows that each $f(x_i) \setminus f(x)$ consists of solely $-$ve events in B and so are included in y. This ensures the compatibility of $f(x_1)$ and $f(x_2)$. That $(x_1 \cup x_2)\|y \in \mathscr{F}$ now follows from f reflecting $-$-compatibility and its affinity.

The final case is when the increments are, w.l.o.g. $x \xrightarrow{a_1} \subset x_1$ and $y \xrightarrow{b_2} \subset y_2$, when a_1 is $-$ve in A and b_2 +ve in B. Then $y \sqsubseteq_B f(x_1)$ and $y_2 \sqsubseteq_B f(x)$, so $y_2 \sqsubseteq_B f(x_1)$, making $x_1\|y_2 \in \mathscr{F}$. □

C.1 A functor

Let $f : A \to B$ and $g : B \to C$ be affine stable maps. They determine stable families

$$\mathscr{F} = \{x \| y \mid f(x) \sqsupseteq_B y\} \text{ and}$$
$$\mathscr{G} = \{y \| z \mid g(y) \sqsupseteq_C z\},$$

respectively. Consider the stable family determined by the composition of functions gf, *viz.*

$$\{x \| z \mid gf(x) \sqsupseteq_C z\}.$$

One can show straightforwardly that

$$\{x \| z \mid gf(x) \sqsupseteq_C z\} = \{x \| z \mid \exists y \in \mathscr{C}(B). \, f(x) \sqsupseteq_B y \ \& \ g(y) \sqsupseteq_C z\}$$
$$\{x \| z \mid \exists y \in \mathscr{C}(B). \, x \| y \in \mathscr{F} \ \& \ y \| z \in \mathscr{G}\}$$
$$= \mathscr{G} \circ \mathscr{F},$$

where the last composition is essentially the composition of stable families as relations: for instance, regarding the stable family $\mathscr{F}$ as

$$\{(x, y) \in \mathscr{C}(A) \times \mathscr{C}(B) \mid f(x) \sqsupseteq_B y\},$$

observing the isomorphism $\mathscr{C}(A) \times \mathscr{C}(B) \cong \mathscr{C}(A^\perp \| B)$. We shall show that

$$\mathrm{Pr}(\mathscr{G}) \odot \mathrm{Pr}(\mathscr{F}) \cong \mathrm{Pr}(\mathscr{G} \circ \mathscr{F}),$$

so reducing the composition of strategies of affine-stable maps to relational composition; by definition, it follows directly that

$$g_! \odot f_! \cong (gf)_!.$$

For functoriality of $(\_)_!$ we also require preservation of identities. However, the stable family determined by $\mathrm{id}_A : \mathscr{C}(A) \to \mathscr{C}(A)$ is, by definition,

$$\{x \| y \mid x \sqsupseteq_A y\} = \mathscr{C}(\mathbb{C}_A),$$

ensuring that $\mathrm{id}_{A!} \cong \mathbb{C}_A$.

LEMMA C.2. *Let $\sigma : A \nrightarrow B$ and $\tau : B \nrightarrow C$ be strategies. Suppose τ_1 is partial rigid (i.e., the component $\tau_1 : T \to B$ preserves causal dependency when defined). Letting $x \in \mathscr{C}(S)^o$, $y \in \mathscr{C}(T)^o$,*

$$y \circledast x \text{ is defined } \textit{iff } \sigma_2 x = \tau_1 y.$$

PROOF. Write $x_A = \sigma_1 x$, $x_B = \sigma_2 x$, $y_B = \tau_1 y$ and $y_C = \tau_2 y$. Recall $y \circledast x$ is defined to be the bijection

$$x \| y_C \cong x_A \| x_B \| x_C \cong x_A \| y$$

induced by σ and τ provided $x_B = y_B$, *i.e.* $\sigma_2 x = \tau_1 y$, and the bijection is secured—see Theorem 3.2. To simplify notation we can present the bijection as $x \cup y$ in which we identify the two sets x and y at their parts $\sigma^{-1} x_B$ and $\tau^{-1} y_B$ via the common image $x_B = y_B$.

To obtain a contradiction, suppose that the bijection were not secured, that there were a causal loop in $x \cup y$, *i.e.* that there were a chain

$$u_1 \rightarrowtail u_2 \rightarrowtail \cdots \rightarrowtail u_n = u_1$$

of events in $x \cup y$, with $n > 1$, w.r.t. causal dependency $\rightarrowtail$ which is either $\rightarrowtail_S$ or $\rightarrowtail_T$. The events of $x \circledast y$ and so of the chain are either over A, B or C. As there are no causal loops in S or T the causal loop must contain events over each of A, B and C. W.l.o.g., we may assume u_1 is over B.

Part of the chain is over C. The whole chain has the form

$$u_1 \rightarrowtail \cdots \rightarrowtail u_{i-1} \rightarrowtail_T u_i \rightarrowtail_T \cdots \rightarrowtail_T u_j \rightarrowtail_T u_{j+1} \rightarrowtail \cdots \rightarrowtail u_n = u_1$$

where u_{i-1} and u_{j+1} are over B and $u_i, \cdots, u_j$ are all over C. Clearly $u_{i-1} <_T u_{j+1}$. As τ_1 is partial rigid, we obtain $\tau(u_{i-1}) <_B \tau(u_{j+1})$. With the identification of events over B in x and y, we have $\sigma(u_{i-1}) <_B \sigma(u_{j+1})$. As σ locally reflects causal dependency, we see that $u_{i-1} <_S u_{j+1}$. We now have a causal loop

$$ u_1 \to \cdots \to u_{i-1} <_S u_{j+1} \to \cdots \to u_n = u_1 $$

from which the events $u_i, \cdots, u_j$ over C have been excised. Continuing in this way we can remove all events over C from the causal loop, obtaining a causal loop in S —a contradiction. □

Now to the isomorphism. First, a key observation, expressing that the strategy obtained from an affine-stable map doesn't disturb the causality of input:

PROPOSITION C.3. *Let $g : B \to C$ be an affine-stable map which determines the stable family $\mathscr{G} = \{y\|z \mid g(y) \sqsupseteq_C z\}$. Let $y\|z \in \mathscr{G}$. Then,*

$$ \forall b, b' \in y. \ (1, b') \leq_{y\|z} (1, b) \iff b' \leq_B b. $$

In the strategy $g_! = top : \mathrm{Pr}(\mathscr{G}) \to B^\perp \| C$, the component $(g_!)_1 : \mathrm{Pr}(\mathscr{G}) \to B^\perp$ is partial rigid.

PROOF. Recall $(1, b') \leq_{y\|z} (1, b)$ iff every subconfiguration of $y\|z$ in $\mathscr{G}$ which contains $(1, b)$ also contains $(1, b')$.

Any subconfiguration of $y\|z$ necessarily takes the form $y'\|z'$ where y' is a subconfiguration of y in B and z' is a subconfiguration of z in C with $g(y') \sqsupseteq_B z'$. From $b' \leq_B b$ it therefore follows that $(1, b') \leq_{y\|z} (1, b)$.

Conversely, given a subconfiguration y' of y we have $y'\|g(y') \in \mathscr{G}$ whence $y'\|g(y') \cap z$ is a subconfiguration of $y\|z$ in $\mathscr{G}$. From this the converse implication follows: if $(1, b') \leq_{y\|z} (1, b)$ then $b' \leq_B b$.

Thus $(1, b') \leq_{y\|z} (1, b)$ iff $b' \leq_B b$, for all $b, b' \in y$. That $(g_!)_1$ is partial rigid is a direct consequence. □

LEMMA C.4. *Let $f : A \to B$ and $g : B \to C$ be affine stable maps which determine stable families $\mathscr{F} = \{x\|y \mid f(x) \sqsupseteq_B y\}$ and $\mathscr{G} = \{y\|z \mid g(y) \sqsupseteq_C z\}$, respectively. Then, $\mathrm{Pr}(\mathscr{G}) \odot \mathrm{Pr}(\mathscr{F}) \cong \mathrm{Pr}(\mathscr{G} \circ \mathscr{F})$.*

PROOF. Recall, $\mathrm{Pr}(\mathscr{G}) \odot \mathrm{Pr}(\mathscr{F})$ is obtained as $\mathrm{Pr}(\mathscr{G} \circledast \mathscr{F})$ followed by hiding the synchronisations over B. First consider $\mathscr{G} \circledast \mathscr{F}$.

A finite configuration of $\mathscr{G} \circledast \mathscr{F}$, built as a pullback of stable families, has the form $x\|y\|z$ where $x\|y \in \mathscr{F}$ and $y\|z \in \mathscr{G}$ and the causal dependencies from $\mathscr{F}$ and $\mathscr{G}$ do not jointly introduce any causal loops. However, from the observation of Proposition C.3 and Lemma C.2 above, it follows that there are no causal loops for such particular stable families.

It follows that for all $x\|y \in \mathscr{F}$ and $y\|z \in \mathscr{G}$ we have $x\|y\|z$ is a configuration of $\mathscr{G} \circledast \mathscr{F}$. Thus we have a simple characterisation of the the stable family $\mathscr{G} \circledast \mathscr{F}$:

$$ \mathscr{G} \circledast \mathscr{F} = \{x\|y\|z \in \mathscr{C}(A^\perp \| B \| C) \mid x\|y \in \mathscr{F} \ \& \ y\|z \in \mathscr{G}\}. $$

It remains to consider the effect of hiding the synchronisations over B and show

$$ \mathrm{Pr}(\mathscr{G}) \odot \mathrm{Pr}(\mathscr{F}) \cong \mathrm{Pr}(\mathscr{G} \circ \mathscr{F}), $$

where

$$ \mathscr{G} \circ \mathscr{F} = \{x\|z \in \mathscr{C}(A^\perp \| C) \mid \exists y \in \mathscr{C}(B). \ x\|y \in \mathscr{F} \ \& \ y\|z \in \mathscr{G}\}. $$

(As we saw in the discussion preceding this lemma, this is the stable family obtained from the composition gf.) To this end we define

$$ \theta : \mathrm{Pr}(\mathscr{G}) \odot \mathrm{Pr}(\mathscr{F}) \to \mathrm{Pr}(\mathscr{G} \circ \mathscr{F}) $$

and its putative mutual inverse

$$\phi : \mathrm{Pr}(\mathscr{G} \circ \mathscr{F}) \to \mathrm{Pr}(\mathscr{G}) \odot \mathrm{Pr}(\mathscr{F}) .$$

For simplicity of notation, to avoid indices, throughout this proof assume that the events A, B and C are pairwise disjoint and identify $x\|y\|z$ with $x \cup y \cup z$.

The events of $\mathrm{Pr}(\mathscr{G}) \odot \mathrm{Pr}(\mathscr{F})$ have the form $[a]_{x\|y\|z}$, where $a \in x$, or $[c]_{x\|y\|z}$, where $c \in z$, and $x\|y\|z \in \mathscr{G} \circledast \mathscr{F}$. The events of $\mathrm{Pr}(\mathscr{G} \circ \mathscr{F})$ have the form $[a]_{x\|z}$, where $a \in x$, or $[c]_{x\|z}$, where $c \in z$, and $x\|z \in \mathscr{G} \circ \mathscr{F}$. Define

$$\theta([d]_{x\|y\|z}) = [d]_{x\|z} \text{ and } \phi([d]_{x\|z}) = [d]_{x\|f(x)\|z} ,$$

on typical events $[d]_{x\|y\|z} \in \mathrm{Pr}(\mathscr{G} \circ \mathscr{F})$ and $[d]_{x\|z} \in \mathrm{Pr}(\mathscr{G} \circ \mathscr{F})$. We should check θ and ϕ are well-defined functions. In showing that θ is well-defined we use that $x\|y\|z$ is a configuration of $\mathscr{G} \circledast \mathscr{F}$ directly implies $x\|z$ is a configuration of $\mathscr{G} \circ \mathscr{F}$. In showing ϕ is well-defined we need that $x\|z \in \mathscr{G} \circ \mathscr{F}$ implies $x\|f(x)\|z \in \mathscr{G} \circledast \mathscr{F}$. Assuming $x\|z \in \mathscr{G} \circ \mathscr{F}$, we have $x\|y \in \mathscr{F}$ and $y\|z \in \mathscr{G}$ for some $y \in \mathscr{C}(B)$. Then $f(x) \sqsupseteq_B y$ and $g(y) \sqsupseteq_C z$. Thus $gf(x) \sqsupseteq_C g(y) \sqsupseteq_C z$ whence $g(f(x)) \sqsupseteq_C z$ ensuring $f(x)\|z \in \mathscr{G}$. Clearly $x\|f(x) \in \mathscr{F}$, so $x\|f(x)\|z \in \mathscr{G} \circledast \mathscr{F}$, as needed.

We show θ and ϕ are mutual inverses. It is easy to see that $\theta\phi([d]_{x\|z}) = [d]_{x\|z}$. By definition, $\phi\theta([d]_{x\|y\|z}) = [d]_{x\|f(x)\|z}$, where $x\|y\|z \in \mathscr{G} \circledast \mathscr{F}$ and d is an event of x or z. We require

$$[d]_{x\|y\|z} = [d]_{x\|f(x)\|z} .$$

To this end we show $x\|(y \cap f(x))\|z \in \mathscr{G} \circledast \mathscr{F}$; once this is shown we have

$$[d]_{x\|y\|z} = [d]_{x\|(y \cap f(x))\|z} = [d]_{x\|f(x)\|z}$$

—using twice the general fact that $[e]_v = [e]_w$ when e is an event of compatible configurations v and w of a stable family. To show $x\|(y \cap f(x))\|z \in \mathscr{G} \circledast \mathscr{F}$ we require

$$x\|(y \cap f(x)) \in \mathscr{F} \text{ and } (y \cap f(x))\|z \mathscr{G} .$$

From $f(x) \sqsupseteq_B y$ with $f(x) \sqsupseteq_B f(x)$ we obtain $f(x) \sqsupseteq_B (y \cap f(x))$; so $x\|(y \cap f(x)) \in \mathscr{F}$. From $f(x) \sqsupseteq_B y$ we get $g(f(x) \cap y) = g(f(x)) \cap g(y)$. But $g(f(x)) \sqsupseteq_C z$ and $g(y) \sqsupseteq_C z$ ensuring $g(f(x)) \cap g(y) \sqsupseteq_C z$. Hence $g(f(x) \cap y) \sqsupseteq_C z$ and $(y \cap f(x))\|z \in \mathscr{G}$, as required. This establishes a bijection between the events of $\mathrm{Pr}(\mathscr{G}) \odot \mathrm{Pr}(\mathscr{F})$ and those of $\mathrm{Pr}(\mathscr{G} \circ \mathscr{F})$.

For an isomorphism, we require the bijection respects causal dependency and consistency. The matching of a configuration $x\|z$ in $\mathscr{G} \circ \mathscr{F}$ with a configuration $x\|f(x)\|z$ in $\mathscr{G} \circledast \mathscr{F}$ clearly respects inclusion. This implies

$$d' \leq_{x\|z} d \iff d' \leq_{x\|f(x)\|z} d ,$$

for d, d' in $x \in \mathscr{C}(A)$ or $z \in \mathscr{C}(C)$. This entails that the bijection on events given by θ and ϕ respects causal dependency.

Via the matching of configurations, both θ and its inverse ϕ may be shown to preserve consistency. This establishes the isomorphism of the lemma. □

COROLLARY C.5. *The operation $(\_)_!$ is a (pseudo) functor from the category of affine-stable maps to concurrent strategies.*

C.2 For Section 5.3, the adjunction

PROPOSITION C.6. *Let f be an additive-stable function from A to B between event structures with polarity. Define*

$$F_! := \{x\|y \in \mathscr{C}(A^\perp \| B) \mid fx \sqsupseteq_B y\} ,$$

$$F^* := \{y\|x \in \mathscr{C}(B^\perp \| A) \mid y \sqsupseteq_B fx\} .$$

Define $f_! : \mathrm{Pr}(F_!) \xrightarrow{top} A^\perp \| B$ *and* $f^* : \mathrm{Pr}(F^*) \xrightarrow{top} B^\perp \| A$. *Then the composition of strategies* $f^* \odot f_!$ *is isomorphic to*

$$\mathrm{Pr}(F^* \circ F_!) \xrightarrow{top} A^\perp \| A$$

and $f_! \odot f^*$ *to*

$$\mathrm{Pr}(F_! \circ F^*) \xrightarrow{top} B^\perp \| B \ ,$$

based on the relational composition of the stable families.

THEOREM C.7. *Let* f *be an additive-stable function from* A *to* B *between event structures with polarity. In the bicategory of strategies the strategies* $f_!$ *and* f^* *form an adjunction* $f_! \dashv f^*$.

PROOF. It is easiest to carry out the arguments by considering the associated constructions on stable families. We obtain the compositions $f^* \odot f_!$ and $f_! \odot f^*$ from "relational" compositions of the stable families

$$F_! := \{x \| y \in \mathscr{C}(A^\perp \| B) \mid fx \sqsupseteq_B y\}$$

for $f_!$ and

$$F^* := \{y \| x \in \mathscr{C}(B^\perp \| A) \mid y \sqsupseteq_B fx\}$$

for f^*.

By Proposition C.6, the composition $f^* \odot f_!$ is the event structure $\mathrm{Pr}(F^* \circ F_!)$ derived from the stable family

$$F^* \circ F_! = \{x \| x' \in \mathscr{C}(A^\perp \| A) \mid fx \sqsupseteq_B fx'\}$$

—obtained as the relational composition of the stable families $F_!$ and F^*. Recall, from Lemma 4.2, that the stable family of cc_A is

$$C_A := \{x \| x' \in \mathscr{C}(A^\perp \| A) \mid x \sqsupseteq_A x'\} \,.$$

Define the unit $\eta : \mathit{cc}_A \Rightarrow f^* \odot f_!$ to be the map $\mathrm{Pr}(I)$ of event structures with polarity got from the inclusion of stable families

$$I : C_A \hookrightarrow F^* \circ F_! \,;$$

clearly, $x \| x' \in C_A$, i.e. $x \sqsupseteq_A x'$, implies $fx \sqsupseteq_B fx'$, so $x \| x' \in F^* \circ F_!$.

By Proposition C.6, the composition $f_! \odot f^*$ is the event structure $\mathrm{Pr}(F_! \circ F^*)$ got from the stable family
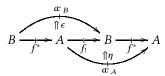
$$F_! \circ F^* = \{y \| y' \in \mathscr{C}(B^\perp \| B) \mid \exists x \in \mathscr{C}(A). \ y \sqsupseteq_B fx \ \& \ fx \sqsupseteq_B y'\}$$

—obtained as the relational composition of the stable families F^* and $F_!$. The counit $\epsilon : f_! \odot f^* \Rightarrow \mathit{cc}_B$ is the the map $\mathrm{Pr}(J)$ got from the inclusion of stable families
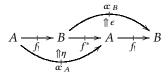
$$J : F_! \circ F^* \hookrightarrow C_B \,;$$

clearly, $y \| y' \in F_! \circ F^*$, i.e. $y \sqsupseteq_B fx$ and $fx \sqsupseteq_B y'$, implies $y \sqsupseteq_B y'$, so $y \| y' \in C_B$.

To obtain an adjunction $f_! \dashv f^*$ we require (i) $(f^* \epsilon)(\eta f^*) = \mathrm{id}_{f^*}$, i.e. the composition of the 2-cells
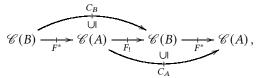
is the identity 2-cell $\mathrm{id}_{f^*} : f^* \Rightarrow f^*$; and (ii) $(\epsilon f_!)(f_! \eta) = \mathrm{id}_{f_!}$, *i.e.* the composition of the 2-cells

$$
\begin{array}{c}
& \overset{\alpha_B}{\Uparrow \epsilon} & \\
A \xrightarrow{f_!} B \xrightarrow{f^*} A \xrightarrow{f_!} B \\
& \underset{\alpha_A}{\Uparrow \eta} &
\end{array}
$$

is the identity 2-cell $\mathrm{id}_{f_!} : f_! \Rightarrow f_!$.

We establish (i) and (ii) by considering the companion diagrams for stable families—the diagrams (i) and (ii) are got by applying Pr to the diagrams for stable families. Consider the diagram for (i). It takes the form

$$
\begin{array}{c}
& \overset{C_B}{\cup \mathsf{I}} & \\
\mathscr{C}(B) \xrightarrow{F^*} \mathscr{C}(A) \xrightarrow{F_!} \mathscr{C}(B) \xrightarrow{F^*} \mathscr{C}(A)\,, \\
& \underset{C_A}{\cup \mathsf{I}} &
\end{array}
$$

yielding the inclusion $C_A \circ F^* \subseteq F^* \circ C_B$. We check this is the identity inclusion, from which (i) follows, by showing the converse inclusion $F^* \circ C_B \subseteq C_A \circ F^*$. Suppose $y \| x \in F^* \circ C_B$, *i.e.*
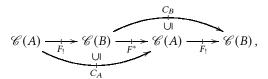
$$
y \sqsupseteq_B y' \ \& \ y' \sqsupseteq_B fx \,,
$$

for some $y' \in \mathscr{C}(B)$. Then,

$$
y \sqsupseteq_B fx \ \& \ x \sqsupseteq_A x \,,
$$

so $y \| x \in C_A \circ F^*$.

The diagram for (ii) takes the form

$$
\begin{array}{c}
& \overset{C_B}{\cup \mathsf{I}} & \\
\mathscr{C}(A) \xrightarrow{F_!} \mathscr{C}(B) \xrightarrow{F^*} \mathscr{C}(A) \xrightarrow{F_!} \mathscr{C}(B)\,, \\
& \underset{C_A}{\cup \mathsf{I}} &
\end{array}
$$

yielding the inclusion $F_! \circ C_A \subseteq C_B \circ F_!$. To show (ii), we check that the converse inclusion $C_B \circ F_! \subseteq F_! \circ C_A$ also holds. Suppose $x \| y \in C_B \odot F_!$, *i.e.*

$$
fx \sqsupseteq_B y' \ \& \ y' \sqsupseteq y \,,
$$

for some $y' \in \mathscr{C}(B)$. Then,

$$
x \sqsupseteq_A x \ \& \ fx \sqsupseteq_B y \,,
$$

so $x \| y \in F_! \circ C_A$. $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad\square$