Symmetry and Concurrency
(Extended Abstract)

Glynn Winskel

University of Cambridge Computer Laboratory, England
`Glynn.Winskel@cl.cam.ac.uk`
`http://www.cl.cam.ac.uk/users/gw104`

Abstract. A category of event structures with symmetry is introduced and its categorical properties investigated. Applications to the event-structure semantics of higher order processes, nondeterministic dataflow and the unfolding of higher-dimensional automata and Petri nets with multiple tokens are indicated.

Key words: Event structures, symmetry, pseudo monads, spans, higher order processes, nondeterministic dataflow, unfolding, Petri nets, higher dimensional automata.

1 Introduction

In the paper introducing event structures [15] a 'curious mismatch' was noted. There event structures represent domains, so types. But they also represent processes which belong to a type. How are we to reconcile these two views?

One answer has arisen in recent work under the banner of 'domain theory for concurrency' (see [17] for a summary). This slogan stands for an attempt to push the methodology of domain theory and denotational semantics into the areas of interactive/concurrent/distributed computation, where presently more syntactic, operational or more informal methodologies prevail. Certain generalized relations (profunctors [4]) play a strong unifying role and it was discovered that in several contexts that they could be represented in a more informative operational way by spans of event structures [16, 28, 19].

A span of event structures is typically of the form

$$\begin{tikzcd} & E \arrow[dl, "in"'] \arrow[dr, "out"] & \\ A & & B \end{tikzcd}$$

where *in* and *out* are maps of event structures—the maps are not necessarily of the same kind. The event structure E represents a process computing from an input type, represented by the event structure A, to output type represented by B. A span with no input amounts to just a single map $E \xrightarrow{out} B$ which we can read as expressing that the process E has type B. So spans are a way to reconcile the double role that event structures can take, as processes and as types.

Of course spans should compose. So one would like systematic ways to vary the *in* and *out* maps of spans which ensure they do. One way is to derive the maps by a Kleisli construction from monads on a fundamental category of event structures. With respect to suitable monads S and T satisfying a suitable distributivity law, one can form a bicategory of more general spans

$$\begin{tikzcd} & E \arrow[dl] \arrow[dr] & \\ S(A) & & T(B)\ . \end{tikzcd}$$

It becomes important that event structures are able to support a reasonable repertoire of monads, including monads which produce multiple, essentially similar, copies of an event structure. For this the introduction of symmetry seems essential.[1]

In fact, there are several reasons for introducing symmetry to event structures and related models:

– It's there—at least informally. Symmetry often plays a role in the analysis of distributed algorithms. In particular, symmetry has always been present at least informally in the model of strand spaces, and has recently been exploited in exploring their behaviour [8], and was used to understand their expressivity [6]. Strand spaces are forms of event structures used in the analysis of security protocols. They comprise a collection of strands of input and output events, possibly with the generation of fresh values. Most often there are collections of strands which are essentially indistinguishable and can be permuted one for another without changing the strand space's behaviour.
– To obtain categorical characterizations of unfoldings of higher-dimensional automata [7], and more specifically Petri nets in which places may hold with multiplicity greater than one. There are well-known ways to unfold such general nets; for example by distinguishing the tokens through 'colours,' splitting the places and events accordingly and reducing the problem to the unfolding in [15]. But the folding maps are not unique (w.r.t. an obvious cofreeness property). They are however unique 'up to symmetry.'
– Event structures are sometimes criticized for not being abstract enough. One precise way in which this manifests itself is that the category of event structures does not support monads and comonads of the kind discovered for more general presheaf models [4]. The computation paths of an event structure, its configurations, are ordered by inclusion. In contrast the paths of presheaf models can be related more generally by maps. Some (co)monads used for presheaf models allow the explicit copying of processes and produce a proper category of paths even when starting with a partial order of paths—this arises because of the similarity of one copy of a process with another.

[1] Symmetry was introduced into game semantics specifically to support a 'copying' comonad [1].

The last point is especially pertinent to the versatility of spans of event structures. This paper presents a definition of a symmetry on an event structure. Roughly a symmetry will express the similarity of *finite* behaviours of an event structure. The introduction of symmetries to event structures will, in effect, put the structure of a category on their finite configurations, and so broaden the structure of computation paths event structures can represent. The ensuing category of event structures with symmetries will support a much richer class of (pseudo) monads, from which we can then obtain more general kinds of span. The category of event structures with symmetry with rigid maps emerges as fundamental; other maps on event structures can be obtained by a Kleisli construction or as instances of general spans starting from rigid maps.

Several applications, to be developed in future work, are outlined in Section 6:

– *Event types:* One reason why so-called 'interleaving' models for concurrency have gained prevalence is that they support definitions by cases on the initial actions processes can do; another is that they readily support higher-order processes. Analogous facilities are lacking, at least in any reasonable generality, in 'true-concurrency' models—models like Petri nets and event structures, in which causal dependence and independence are represented explicitly. It is sketched how processes can be associated with 'event types' which specify the kinds of events they can do, and how event types can support definitions by cases on events. There are difficulties and much more needs to be done. But the examples do demonstrate the key role that symmetry and the copying of processes could play in obtaining flexible event types and event-based definitions.
– *Nondeterministic dataflow and affine-HOPLA:* 'Stable' spans of event structures, a direct generalisation of Berry's stable functions [2], have been used to give semantics to nondeterministic dataflow [19] and the higher-order process language affine-HOPLA [16]. Stable spans can be obtained as instances of general spans. The realization of the 'demand' maps used there as a Kleisli construction on rigid maps provides a striking example of the power of symmetry.
– *Unfoldings:* One obvious application is to the unfolding of a general Petri net to an event structure with symmetry; the symmetry reflects that present in the original net through the interchangeability of tokens. Another related issue is the unfolding of higher-dimensional automata, where identifications of edges are reflected in the symmetry of the events to which they unfold.

This presentation concentrates on the model of (prime) event structures. But the same techniques apply to many other models, including more algorithmically-amenable models such as Petri nets or versions of transition systems. The model of stable families [23] plays a significant, if hidden role, in the proofs—they deserve a more forthright treatment in future. The work reported is based on an extended article which appears in the Gordon Plotkin Festschrift [29], where further details may be found. As well as streamlining the presentation, I have taken the opportunity here to make corrections (chiefly in the unfinished work on 'Event types', Section 6.2), additions (on 'Event types', Section 6.2 and on

'Unfoldings,' Section 6.4), and replaced the condition of countability on event structures by the weaker condition of 'consistent-countable,' which suffices for the proofs in [29] and allows extra results, *e.g.* in Sections 6.2 and 6.4.

2 Event structures

Event structures [15, 22, 25, 26] are a model of computational processes. They represent a process as a set of event occurrences with relations to express how events causally depend on others, or exclude other events from occurring. In one of their simpler forms they consist of a set of events on which there is a consistency relation expressing when events can occur together in a history and a partial order of causal dependency—writing $e' \leq e$ if the occurrence of e depends on the previous occurrence of e'.

An *event structure* comprises $(E, \mathrm{Con}, \leq)$, consisting of a set E, of *events* which are partially ordered by $\leq$, the *causal dependency relation*, and a *consistency relation* Con consisting of finite subsets of E, which satisfy

$$\begin{aligned}
&\{e' \mid e' \leq e\} \text{ is finite for all } e \in E,\\
&\{e\} \in \mathrm{Con} \text{ for all } e \in E,\\
&Y \subseteq X \in \mathrm{Con} \Rightarrow Y \in \mathrm{Con}, \quad \text{and}\\
&X \in \mathrm{Con} \;\&\; e \leq e' \in X \Rightarrow X \cup \{e\} \in \mathrm{Con}.
\end{aligned}$$

Here we insist that an event structure is *consistent-countable*,[2] *i.e.* that there is a function χ from its events to the natural numbers ω such that $\{e_1, e_2\} \in \mathrm{Con}$ and $\chi(e_1) = \chi(e_2)$ implies $e_1 = e_2$.

The events are to be thought of as event occurrences; in any history an event is to appear at most once. A configuration is a set of events which have occurred by some stage in a process. According to our understanding of the consistency predicate and causal dependency relations a configuration should be consistent and such that if an event appears in a configuration then so do all the events on which it causally depends. Here we restrict attention to finite configurations.

The *(finite) configurations*, $\mathcal{C}^o(E)$, of an event structure E consist of those finite subsets $x \subseteq E$ which are

Consistent: $x \in \mathrm{Con}$ and
Down-closed: $\forall e, e'.\ e' \leq e \in x \Rightarrow e' \in x$.

The configurations of an event structure are ordered by inclusion, where $x \subseteq x'$, *i.e.* x is a sub-configuration of x', means that x is a sub-history of x'. Note that an individual configuration inherits an order of causal dependency on its events from the event structure so that the history of a process is captured through a partial order of events. For an event e the set $\{e' \in E \mid e' \leq e\}$ is a configuration describing the whole causal history of the event e.

[2] The condition of consistent-countability replaces the stronger condition of countability of event structures in [29]. Proofs there still go through with the weaker condition, while the extra generality makes new results possible—see Sections 6.2, 6.4.

When the consistency relation is determined by the pairwise consistency of events we can replace it by a binary relation or, as is more usual, by a complementary binary conflict relation on events. It can be awkward to describe operations such as certain parallel compositions directly on the simple event structures here, because an event determines its whole causal history. One closely related and more versatile model is that of stable families, described in Appendix B.

Let E and E' be event structures. A *partial map* of event structures $f : E \rightharpoonup E'$ is a partial function on events $f : E \rightharpoonup E'$ such that for all configurations x of E its direct image fx is a configuration of E' for which

$$\text{if } e_1, e_2 \in x \text{ and } f(e_1) = f(e_2) \in E', \text{ then } e_1 = e_2.$$

The map expresses how the occurrence of an event e in E induces the coincident occurrence of the event $f(e)$ in E' whenever it is defined. The partial function f respects the instantaneous nature of events: two distinct event occurrences which are consistent with each other cannot both coincide with the occurrence of a common event in the image. Maps of event structures compose as partial functions.

We will say the map is *total* iff the function f is total. Notice that for a total map f the condition on maps now 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 f to a function from x to fx is thus bijective.

We say the map f is *rigid* iff it is total and for all $x \in \mathcal{C}^o(E)$ and $y \in \mathcal{C}^o(E')$

$$y \subseteq f(x) \Rightarrow \exists z \in \mathcal{C}^o(E).\ z \subseteq x \text{ and } fz = y\ .$$

(The configuration z is necessarily unique.)

A rigid map of event structures preserves the causal dependency relation "rigidly," so that the causal dependency relation on the image fx is a copy of that on a configuration x of E; this is not so for general maps where x may be augmented with extra causal dependency over that on fx. (Special forms of rigid maps appeared as *rigid embeddings* in Kahn and Plotkin's work on concrete domains [12].)

Here we concentrate on the category of event structures with total maps.

Definition 1. *Write $\mathcal{E}$ for the category of event structures with total maps. (In future, unless further specified, by a map of event structures we will mean a total map.)*

Proposition 1. *The category $\mathcal{E}$ of event structures with total maps of event structures has (binary) products and pullbacks (though no terminal object).*

In defining symmetries on event structures we will make use of open maps w.r.t. finite elementary event structures (*i.e.* finite event structures in which all subsets of events are consistent) as the particular choice of paths [11].

Say a map $h : A \to B$, between event structures A and B, is *open* iff for all maps $j : p \to q$ between finite elementary event structures, any commuting square

$$\begin{tikzcd}
p \arrow[r, "x"] \arrow[d, "j"'] & A \arrow[d, "h"] \\
q \arrow[r, "y"'] & B
\end{tikzcd}$$

can be split into two commuting triangles

$$\begin{tikzcd}
p \arrow[r, "x"] \arrow[d, "j"'] & A \arrow[d, "h"] \\
q \arrow[r, "y"'] \arrow[ur, dashed, "z"] & B.
\end{tikzcd}$$

That the square commutes means that the path $h \circ x$ in B can be extended via j to a path y in B. That the two triangles commute means that the path x can be extended via j to a path z in A which matches y.

Open maps are a generalisation of functional bisimulations, known from transition systems.

Proposition 2. *A map $h : A \to B$ of event structures is open iff h is rigid and satisfies: $\forall x \in \mathcal{C}^o(A), y' \in \mathcal{C}^o(B).\ hx \subseteq y' \Rightarrow \exists x' \in \mathcal{C}^o(E).\ x \subseteq x'\ \&\ hx' = y'$.*

3 Event structures with symmetry

We shall present a general definition of symmetry, concentrating on the category $\mathcal{E}$ of event structures with total maps. This category has (binary) products and pullbacks (though no terminal object) and supports a notion of open map. For the definition of symmetry we are about to give this is all we require.

A symmetry on an event structure should specify which events are similar in such a way that similar events have similar pasts and futures. This is captured, somewhat abstractly, by the following definition.

Definition 2. *An* event structure with symmetry (E, l, r) *comprises an event structure E together with* open *maps $l : S \to E$ and $r : S \to E$ from a common event structure S such that the map $\langle l, r\rangle : S \to E \times E$ is an equivalence relation (i.e., the map $\langle l, r\rangle$ is monic—equivalently, l, r are jointly monic—and satisfies the standard diagramatic properties of reflexivity, symmetry and transitivity [10]. See Appendix A).*

A bisimulation is given by a span of open maps [11], in the case of the above definition by the pair of open maps l and r. So the definition expresses a symmetry on an event structure as a bisimulation equivalence. The definition has the advantage of being abstract in that it readily makes sense for any category with binary products and pullbacks for which there is a sensible choice of paths in order to define open maps. It is sensible for the categories of event structures with rigid and partial maps, for stable families, transition systems, trace languages and Petri nets [21], because these categories also have products, pullbacks and open maps; both categories of event structures with rigid and partial maps would have the same class of open maps and so lead to precisely the same event structures with symmetry as objects. We shall mainly concentrate on the category with total maps to connect directly with the particular examples we shall treat here.[3]

For the specific model of event structures there is an alternative way to present a symmetry. We can express a symmetry $l, r : S \to E$ on an event structure E equivalently as a relation of similarity between its finite configurations. More precisely, two finite configurations x, y of E are related by a bijection $\theta_z =_{\rm def} \{(l(s), r(s)) \mid s \in z\}$ if they arise as images $x = l\,z$ and $y = r\,z$ of a common finite configuration z of S; because l and r are locally injective θ_z is a bijection between x and y. Because l and r are rigid the bijection is an order isomorphism between x and y with the order of causal dependency inherited from E. In this way a symmetry on E will determine an *isomorphism family* expressing when and how two finite configurations are similar, or symmetric, in the sense that one can replace the other. As expected, such similarity forms an equivalence relation, and if two configurations are similar then so are their pasts (restrictions to subconfigurations) and futures (extensions to larger configurations).

Definition 3. *An* isomorphism family *of an event structure E consists of a family $\mathbb{S}$ of bijections*

$$\theta : x \cong y$$

between pairs of finite configurations of E such that:
(i) the identities $\mathrm{id}_x : x \cong x$ are in $\mathbb{S}$ for all $x \in \mathcal{C}^o(E)$; if $\theta : x \cong y$ is in $\mathbb{S}$, then so is the inverse $\theta^{-1} : y \cong x$; and if $\theta : x \cong y$ and $\varphi : y \cong z$ are in $\mathbb{S}$, then so is their composition $\varphi \circ \theta : x \cong z$.
(ii) for $\theta : x \cong y$ in $\mathbb{S}$ whenever $x' \subseteq x$ with $x' \in \mathcal{C}^o(E)$, then there is a (necessarily unique) $y' \in \mathcal{C}^o(E)$ with $y' \subseteq y$ such that the restriction of θ to $\theta' : x' \cong y'$ is in $\mathbb{S}$.
(iii) for $\theta : x \cong y$ in $\mathbb{S}$ whenever $x \subseteq x'$ for $x' \in \mathcal{C}^o(E)$, then there is an extension of θ to $\theta' : x' \cong y'$ in $\mathbb{S}$ for some (not necessarily unique) $y' \in \mathcal{C}^o(E)$ with $y \subseteq y'$.
[Note that (i) implies that the converse forms of (ii) and (iii) also hold. Note too that (ii) implies that the bijections in the family $\mathbb{S}$ respect the partial order of causal dependency on configurations inherited from E; the bijections in an isomorphism family are isomorphisms between the configurations regarded as elementary event structures.]

Theorem 1. *Let E be an event structure.*
(i) A symmetry $l, r : S \to E$ determines an isomorphism family $\mathbb{S}$: defining $\theta_z = \{(l(s), r(s)) \mid s \in z\}$ for z a finite configuration of S, yields a bijection

[3] There is a strong case for regarding rigid maps as *the* fundamental maps of event structures, in that other maps on event structures can then ultimately be obtained as Kleisli maps w.r.t. suitable pseudo monads once we have introduced symmetry.

$\theta_z : l\,z \cong r\,z$; the family $\mathbb{S}$ consisting of all bijections $\theta_z : l\,z \cong r\,z$, for z a finite configuration of S.
(ii) An isomorphism family $\mathbb{S}$ of E determines a symmetry $l, r : S \to E$: the family $\mathbb{S}$ forms a stable family; the event structure S is obtained as $\mathrm{Pr}(\mathbb{S})$ *for which the events are primes $[(e_1, e_2)]_\theta$ for θ in $\mathbb{S}$ and $(e_1, e_2) \in \theta$; the maps l and r send a prime $[(e_1, e_2)]_\theta$ to e_1 and e_2 respectively.*

The operations of (i) and (ii) are mutually inverse (regarding relations as subobjects).

Through the addition of symmetry event structures can represent a much richer class of 'path categories' [4] than mere partial orders. The finite configurations of an event structure with symmetry can be extended by inclusion or rearranged bijectively under an isomorphism allowed by the symmetry. In this way an event structure with symmetry determines, in general, a *category* of finite configurations with maps obtained by repeatedly composing the inclusions and allowed isomorphisms. By property (ii) in Definition 3 any such map factors uniquely as an isomorphism of the symmetry followed by an inclusion. While by property (iii) any such map factors (not necessarily uniquely) as an inclusion followed by an isomorphism of the symmetry.

Example 1. Any event structure E can be identified with the event structure with the identity symmetry $(E, \mathrm{id}_E, \mathrm{id}_E)$. Its isomorphism family consists of all identities $\mathrm{id}_x : x \cong x$ on finite configurations $x \in \mathcal{C}^o(E)$.

Example 2. Identify the natural numbers ω with the event structure with events ω, trivial causal dependency given by the identity relation and in which all finite subsets of events are in the consistency relation. Define S to be the product of event structures $\omega \times \omega$ in $\mathcal{E}$; the product comprises events all pairs $(i, j) \in \omega \times \omega$ with trivlal causal dependency, and consistency relation consisting of all finite subsets of $\omega \times \omega$ which are bijective (so we take two distinct pairs (i, j) and (i', j') to be in conflict iff $i = i'$ or $j = j'$.) Define l and r to be the projections $l : S \to E$ and $r : S \to E$. Then $\varpi =_{\mathrm{def}} (\omega, l, r)$ forms an event structure with symmetry. The corresponding isomorphism family in this case coincides with all finite bijections between finite subsets of ω. Any finite subset of events of ϖ is similar to any other. Of course, an analogous construction works for any countable, possibly finite, set.

Example 3. Let $E = (E, l : S \to E, r : S \to E)$ be an event structure with symmetry. Define an event structure with symmetry $!E = (E_!, l_! : S_! \to E_!, r_! : S_! \to E_!)$ comprising ω similar copies of E as follows. The event structure $E_!$ has the set of events $\omega \times E$ with causal dependency

$$(i, e) \leq_! (i', e') \text{ iff } i = i' \ \&\ e \leq_E e'$$

and consistency relation

$$C \in \mathrm{Con}_! \text{ iff } C \text{ is finite } \&\ \forall i \in \omega.\ \{e \mid (i, e) \in C\} \in \mathrm{Con}_E\ .$$

The symmetry $S_!$ has events $\omega \times \omega \times S$ with causal dependency

$$(i,j,s) \leq_{S_!} (i',j',s') \text{ iff } i = i' \ \& \ j = j' \ \& \ s \leq_S s' \ .$$

A finite subset $C \subseteq S_!$ is in the consistency relation $\mathrm{Con}_{S_!}$ iff

$$\{(i,j) \mid \exists s.\ (i,j,s) \in C\} \text{ is bijective } \ \& \ \forall i,j \in \omega.\ \{s \mid (i,j,s) \in C\} \in \mathrm{Con}_S \ .$$

Define $l_!(i,j,s) = (i, l(s))$ and $r_!(i,j,s) = (j, r(s))$ for $i, j \in \omega$, $s \in S$.

The finite configurations of $E_!$ correspond to tuples (or indexed families) $\langle x_i \rangle_{i \in I}$ of nonempty-finite configurations $x_i \in \mathcal{C}^o(E)$ indexed by $i \in I$, where I is a finite subset of ω. With this view of the configurations of $E_!$, the isomorphism family corresponding to $S_!$ specifies isomorphisms between tuples

$$(\sigma, \langle \theta_i \rangle_{i \in I}) : \langle x_i \rangle_{i \in I} \cong \langle y_j \rangle_{j \in J}$$

consisting of a bijection between indices $\sigma : I \cong J$ together with $\theta_i : x_i \cong y_{\sigma(i)}$ from the isomorphism family of S, for all $i \in I$.

The event structure with symmetry ϖ reappears as the special case $!1$, where 1 is the event structure with a single event.

We conclude this section with a general method for constructing symmetries. Just as there is a least symmetry on an event structure, *viz.* the identity symmetry, so is there a greatest. Moreover any bisimulation on an event structure generates a symmetry on it. We take a *bisimulation* on an event structure A to be a pair of open maps $l, r : R \to A$ from an event structure R for which $\langle l, r \rangle$ is monic. (In general we might specify a bisimulation on an event structure just by a pair of open maps from a common event structure, and not insist that the pair is monic. But here, no real generality is lost as such a pair of open maps on event structures will always factor through its image, a bisimulation with monicity.) In fact, the proof proceeds most easily by first establishing an analogous property for isomorphism families, a property which depends on the notion of a *bisimulation family*, defined to be a family of bijections between finite configurations of A which satisfy (ii) and (iii) in Definition 3.

Proposition 3. *Let A be an event structure.*
(i) For any bisimulation family $\mathcal{R}$ on A there is a least isomorphism family $\mathbb{S}$ for which $\mathcal{R} \subseteq \mathbb{S}$.
(ii) For any bisimulation $\langle l_0, r_0 \rangle : R \to A$ there is a least symmetry $\langle l, r \rangle : S \to A$ (understood as a subobject) for which R is a subobject of S. There is a greatest symmetry on A (which coincides with the greatest bisimulation on A).

4 Maps preserving symmetry

Maps between event structures with symmetry are defined as maps between event structures which preserve symmetry. Let (A, l_A, r_A) and (B, l_B, r_B) be event structures with symmetry. A map $f : (A, l_A, r_A) \to (B, l_B, r_B)$ is a map of event structures $f : A \to B$ such that there is a (necessarily unique) map of event structures $h : S_A \to S_B$ ensuring

$$\langle l_B, r_B \rangle \circ h = (f \times f) \circ \langle l_A, r_A \rangle \ .$$

Maps between event structures with symmetry compose as maps of event structures and share the same identity maps.

Definition 4. *We define $\mathcal{SE}$ to be category of event structures with symmetry.*

We can characterize when maps of event structures preserve symmetry in terms of isomorphism families. A map preserving symmetry should behave as a functor both w.r.t. the inclusion between finite configurations and the isomorphisms of the symmetry.

Proposition 4. *A map of event structures $f : A \to B$ is a map $f : (A, l_A, r_A) \to (B, l_B, r_B)$ of event structures with symmetry iff whenever $\theta : x \cong y$ is in the isomorphism family of A then $f\theta : f\,x \cong f\,y$ is in the isomorphism family of B, where $f\theta =_{\mathrm{def}} \{(f(e_1), f(e_2)) \mid (e_1, e_2) \in \theta\}$.*

We explore properties of the category $\mathcal{SE}$. It is more fully described as a category enriched in the category of equivalence relations and so, because equivalence relations are a degenerate form of category, as a 2-category in which the 2-cells are instances of the equivalence $\sim$. This view informs the constructions in $\mathcal{SE}$ which are often very simple examples of the (pseudo- and bi-) constructions of 2-categories.

Definition 5. *Let $f, g : (A, l_A, r_A) \to (B, l_B, r_B)$ be maps of event structures with symmetry between (A, l_A, r_A) and (B, l_B, r_B). Define $f \sim g$ iff there is a (necessarily unique) map of event structures $h : A \to S_B$ such that*

$$\langle f, g \rangle = \langle l_B, r_B \rangle \circ h \ .$$

Straightforward diagrammatic proofs show:

Proposition 5. *The relation $\sim$ is an equivalence relation on maps $\mathcal{SE}(A, B)$ between event structures with symmetry A and B. The relation $\sim$ respects composition in the sense that if $f \sim g$ then $h \circ f \circ k \sim h \circ g \circ k$, for composable maps h and k.*

The category $\mathcal{SE}$ is enriched in the category of equivalence relations (comprising equivalence relations with functions which preserve the equivalence).

We can characterize the equivalence of maps between event structures with symmetry in terms of isomorphism families which makes apparent how $\sim$ is an instance of natural isomorphism between functors.

Proposition 6. *Let $f, g : (A, l_A, r_A) \to (B, l_B, r_B)$ be maps of event structures with symmetry. Then, $f \sim g$ iff $\theta_x : f\,x \cong g\,x$ is in the isomorphism family of (B, l_B, r_B) for all $x \in \mathcal{C}^o(A)$, where $\theta_x =_{\mathrm{def}} \{(f(a), g(a)) \mid a \in x\}$.*

Equivalence on maps yields an equivalence on objects:

Definition 6. *Let A and B be event structures with symmetry. An* equivalence *from A to B is a pair of maps $f : A \to B$ and $g : B \to A$ such that $f \circ g \sim \mathrm{id}_B$ and $g \circ f \sim \mathrm{id}_A$; then we say A and B are equivalent and write $A \simeq B$.*

The category $\mathcal{SE}$ has products.

Theorem 2. *Let (A, l_A, r_A) and (B, l_B, r_B) be event structures with symmetry. Their product in $\mathcal{SE}$ is given by $(A \times B, l_A \times l_B, r_A \times r_B)$, based on the product $A \times B$ of their underlying event structures in $\mathcal{E}$, and sharing the same projections, $\pi_1 : A \times B \to A$ and $\pi_2 : A \times B \to B$.*

The isomorphism family of the product consists of all order isomorphisms $\theta : x \cong x'$ between finite configurations x, x' of $A \times B$, with order inherited from the product, for which $\theta_A = \{(\pi_1(p), \pi_1(p')) \mid (p, p') \in \theta\}$ is in the isomorphism family of A and $\theta_B = \{(\pi_2(p), \pi_2(p')) \mid (p, p') \in \theta\}$ is in the isomorphism family of B.

Let $f, f' : C \to A$ and $g, g' : C \to B$ in $\mathcal{SE}$. If $f \sim f'$ and $g \sim g'$, then $\langle f, g \rangle \sim \langle f', g' \rangle$.

The category $\mathcal{SE}$ does not have a terminal object. However, the event structure with symmetry ϖ defined in Example 2 satisfies an appropriately weakened property (it is a simple instance of a biterminal object):

Proposition 7. *For any event structure with symmetry A there is a map $f : A \to \varpi$ in $\mathcal{SE}$ and moreover for any two maps $f, g : A \to \varpi$ we have $f \sim g$.*

The category $\mathcal{SE}$ does not have pullbacks and equalizers in general. However:

Theorem 3.
(i) Let $f, g : A \to B$ be two maps between event structures with symmetry. They have a pseudo equalizer, i.e. an event structure with symmetry E and map $e : E \to A$ such that $f \circ e \sim g \circ e$ which satisfies the further property that for any event structure with symmetry E' and map $e' : E' \to A$ such that $f \circ e' \sim g \circ e'$, there is a unique map $h : E' \to E$ such that $e' = e \circ h$.
(ii) Let $f : A \to C$ and $g : B \to C$ be two maps between event structures with symmetry. They have a pseudo pullback, i.e. an event structure with symmetry D and maps $p : D \to A$ and $q : D \to B$ such that $f \circ p \sim g \circ q$ which satisfies the further property that for any event structure with symmetry D' and maps $p' : D' \to A$ and $q' : D' \to B$ such that $f \circ p' \sim g \circ q'$, there is a unique map $h : D' \to D$ such that $p' = p \circ h$ and $q' = q \circ h$.

There are obvious weakenings of the conditions of (i) and (ii) in which the uniqueness is replaced by uniqueness up to $\sim$ and equality by $\sim$—these are simple special cases of bilimits called biequalizers and bipullbacks when we regard $\mathcal{SE}$ as a 2-category. As in Theorem 3, we follow tradition and call the stricter construction described in (ii) a *pseudo pullback*. In Theorem 2, that pairing of maps preserves $\sim$ means that the products described are 2-products in $\mathcal{SE}$ regarded as a 2-category. For an accessible introduction to limits in 2-categories see [18].

5 Functors and pseudo monads

Certain functors on $\mathcal{E}$, the category of event structures, straightforwardly induce functors on $\mathcal{SE}$, the enriched category of event structures with symmetry. Say a functor $F : \mathcal{A} \to \mathcal{B}$ has *monic mediators for products* when for all products $A \times A, \pi_1, \pi_2$ in $\mathcal{A}$ and $F(A) \times F(A), p_1, p_2$ in $\mathcal{B}$ the unique mediating map h in the commuting diagram

$$\begin{tikzcd}
 & F(A \times A) \arrow[dl, "F(\pi_1)"'] \arrow[d, dashed, "h"] \arrow[dr, "F(\pi_2)"] & \\
F(A) & F(A) \times F(A) \arrow[l, "p_1"] \arrow[r, "p_2"'] & F(A)
\end{tikzcd}$$

is monic. A functor on several, even infinitely many, arguments $F : \mathcal{E} \times \cdots \times \mathcal{E} \times \cdots \to \mathcal{E}$ which preserves pullbacks, open maps and has monic mediatiors for products will induce a functor on event structures with symmetry respecting $\sim$ on homsets. (A map in a product of categories, such as $\mathcal{E} \times \cdots \times \mathcal{E} \times \cdots$, is taken to be open iff it is open in each component.) We consider some examples.

5.1 Operations

Simple parallel composition For example, consider the functor $\|: \mathcal{E} \times \mathcal{E} \to \mathcal{E}$ which given two event structures puts them in parallel. Let $(A, \mathrm{Con}_A, \leq_A)$ and $(B, \mathrm{Con}_B, \leq_B)$ be event structures. The events of $A \| B$ are $(\{0\} \times A) \cup (\{1\} \times B)$; with $(0, a) \leq (0, a')$ iff $a \leq_A a'$ and $(1, b) \leq (1, b')$ iff $b \leq_B b'$; and with a subset of events C consistent in $A \| B$ iff $\{a \mid (0, a) \in C\} \in \mathrm{Con}_A$ and $\{b \mid (1, b) \in C\} \in \mathrm{Con}_B$. The operation extends to a functor—put the two maps in parallel. It is not hard to check that the functor $\|$ preserves pullbacks and open maps, and that the mediating maps $(A \times A) \| (B \times B) \to (A \| B) \times (A \| B)$ are monic. Consequently it induces a functor $\|: \mathcal{SE} \times \mathcal{SE} \to \mathcal{SE}$ which preserves $\sim$ on homsets. On the same lines the functor giving the parallel composition $\|_{i \in I} A_i$ of countably-indexed event structures $A_i, i \in I$, extends to a functor on event structures with symmetry.

Sum Similarly, the coproduct or sum of two event structures extends to the sum of event structures with symmetry. Let $(A, \mathrm{Con}_A, \leq_A)$ and $(B, \mathrm{Con}_B, \leq_B)$ be event structures. The events of the sum $A + B$ are $(\{0\} \times A) \cup (\{1\} \times B)$; with $(0, a) \leq (0, a')$ iff $a \leq_A a'$ and $(1, b) \leq (1, b')$ iff $b \leq_B b'$; but now a subset of events C is consistent in $A + B$ iff there is $C_0 \in \mathrm{Con}_A$ such that $C = \{(0, a) \mid a \in C_0\}$ or there is $C_1 \in \mathrm{Con}_B$ such that $C = \{(1, a) \mid a \in C_1\}$. We can also form a sum $\Sigma_{i \in I} A_i$ of event structures A_i indexed by a set I. Again this extends to a functor on event structures with symmetry.

5.2 Pseudo monads

That $\mathcal{SE}$ is enriched over equivalence relations ensures that it supports the definitions pseudo functors and pseudo natural transformations, which here parallel those of functor and natural transformation, but with equality replaced by $\sim$. In the same spirit a pseudo monad on $\mathcal{SE}$ satisfies variants of the usual monad laws but expressed in terms of $\sim$ rather than equality (we can ignore the extra coherence conditions [5] as they trivialize in the simple situation here). As examples we consider two particular pseudo monads which we can apply to the semantics of higher-order nondeterministic processes.

The copying pseudo monad The copying operation ! of Example 3 extends to a functor on $\mathcal{SE}$. Let $f : A \to B$ be a map of event structures with symmetry. Define $!f :!A \to !B$ by taking $!f(i, a) = (i, f(a))$ for all events a of A. The functor ! preserves $\sim$ on homsets. (It is not induced by a functor on $\mathcal{E}$.)

The component of the unit $\eta^!_E : E \to !E$ acts so $\eta^!_E(e) = (0, e)$ for all events $e \in E$—it takes an event structure with symmetry E into its zeroth copy in $!E$.

The multiplication map relies on a subsidiary pairing function on natural numbers $[\_, \_] : \omega \times \omega \to \omega$ which we assume is injective. The component of the multiplication $\mu^!_E :!!E \to !E$ acts so $\mu^!_E(i, j, e) = ([i, j], e)$.

It can be checked that the unit and the multiplication are natural transformations and that the usual monad laws, while they do not hold up to equality, do hold up to $\sim$. The somewhat arbitrary choice of the zeroth copy in the definition of the unit and pairing function on natural numbers in the definition of the multiplication don't really matter in the sense that other choices would lead to components $\sim$-equivalent to those chosen. (Different choices lead to natural transformations related by modifications with $\sim$ at all components.)

The partiality pseudo monad Let E be an event structure with symmetry. Define $E_* =_{\text{def}} E \parallel \varpi$, *i.e.* it consists of E and ϖ put in parallel.

The component of the unit $\eta^*_E : E \to E_*$ acts so $\eta^*_E(e) = (0, e)$ for all events $e \in E$—so taking E to its copy in $E \parallel \varpi$.

The component of the multiplication $\mu^*_E : (E_*)_* \to E_*$ acts so $\mu^*_E(0, (0, e)) = (0, e)$ and $\mu^*_E(0, (1, j)) = [0, j]$ and $\mu^*_E(1, k) = [1, k]$, where we use the pairing function on natural numbers above to map the two disjoint copies of ω injectively into ω.

Both η^* and μ^* are natural transformations and the usual monad laws hold up to $\sim$ making a pseudo monad. Again, the definition of multiplication is robust; if we used some alternative way to inject $\omega + \omega$ into ω the resulting multiplication would be $\sim$-related at each component to the one we have defined.

The category of event structures with partial maps has played a central role in the event structure semantics of synchronizing processes [23]. It readily generalizes to accommodate symmetry and reappears as the Kleisli bicategory of $(\_)_*$.

Definition 7. *Let (A, l_A, r_A) and (B, l_B, r_B) be event structure with symmetry. A* partial map *of event structures with symmetry $f : (A, l_A, r_A) \rightharpoonup (B, l_B, r_B)$ consists of a partial map of event structures $f : A \rightharpoonup B$ for which there is a (necessarily unique) partial map of event structures $h : S_A \rightharpoonup S_B$ ensuring*

$$\langle l_B, r_B\rangle \circ h = (f \times f) \circ \langle l_A, r_A\rangle \ .$$

Partial maps of event structures with symmetry form a category; they compose as partial maps of event structures and share the same identity maps. We can define an equivalence relation $\sim$ on partial maps of event structures with symmetry by the obvious analogue of Definition 5. The category is enriched over equivalence relations. (The full subcategory of event structures with identity symmetry is isomorphic to the category of event structures with partial maps.)

Proposition 8. *The Kleisli bicategory of the pseudo monad $(-)_*$ and the category of event structures with symmetry and partial maps (regarded as a 2 category) are biequivalent; the biequivalence is the identity on objects and takes maps $f : A \to B_*$ in the Kleisli bicategory to partial maps $\bar{f} : A \rightharpoonup B$, undefined precisely when the image is in ϖ.*

Equivalences We have enough operations to derive some useful equivalences. Below we use 1 to denote the single-event event structure with symmetry and $\otimes$ for the product of event structures with symmetry with partial maps.

Proposition 9. *For event structures with symmetry:*

(i) $!A \, \| !B \simeq !(A + B)$ *and* $\|_{k\in K} !A_k \simeq !\Sigma_{k\in K} A_k$ *where K is a countable set.*
(ii) $\varpi \simeq !1$ *and* $A \times \varpi \simeq A$.
(iii) $A_* \simeq A \parallel \varpi$, $(!A)_* \simeq !(A + 1)$ *and* $(A \otimes B)_* \simeq A_* \times B_*$.

The equivalence $!A \, \| !B \simeq !(A + B)$, and its infinite version in (i), express the sense in which copying obviates choice. More importantly, they and the other the equivalences enable definitions by case analysis on events, also in the presence of asynchrony.

6 Applications

Here we present some unfinished applications, the subject of current work.

6.1 Spans

Because $\mathcal{SE}$ has pseudo pullbacks—Theorem 3, we can imitate the standard construction of the bicategory of spans (see [14]) to produce a bicategory $\mathrm{Span}_{\mathcal{SE}}$. Its objects are event structures with symmetry. Its maps $\mathrm{Span}_{\mathcal{SE}}(A, B)$, from A to B, are spans

$$\begin{tikzcd} & E \arrow[dl] \arrow[dr] & \\ A & & B \end{tikzcd}$$

composed using the pseudo pullbacks of of Theorem 3 (ii). $\text{Span}_{\mathcal{SE}}$ has a tensor and function space given by the product of $\mathcal{SE}$.

An individual span can be thought of as a process computing from input of type A to output of type B. But given the nature of maps in $\mathcal{SE}$ such a process is rather restricted; from a computational view the process is unnaturally symmetric and 'ultra-linear' because any output event is synchronized with an event of input.

We wish to modify the maps of a span to allow for different regimes of input and output. A systematic way to do this is through the use of pseudo monads on $\mathcal{SE}$ and build more general spans

$$\begin{tikzcd} & E \arrow[dl] \arrow[dr] & \\ S(A) & & T(B) \end{tikzcd}$$

for pseudo monads S and T. For example a span in which $S = (\_)_*$ and $T = !(\_)$ would permit output while ignoring input and allow the output of arbitrarily many similar events of type B. But for such general spans to compose, we require that S and T satisfy several conditions, which we can only indicate here:

– in order to lift to pseudo comonads and monads on spans, S and T should be 'cartesian' pseudo monads, now w.r.t. pseudo/bipullbacks (adapting [3]);
– in order to obtain a comonad-monad distributive law for the liftings of S and T to spans it suffices to have a 'cartesian' distributive law for S and T, with commutativity up to $\sim$, with extra pseudo/bipullback conditions on two of the four diagrams (adapting [13]).

The two pseudo monads $S = (\_)_*$ and $T = !(\_)$ do satisfy these requirements with a distributive law with components $\lambda_E : (!E)_* \to !(E_*)$ such that $\lambda_E(0,(j,e)) = (j,(0,e))$ and $\lambda_E(1,k) = (0,(1,k))$.

The paper has concentrated on the categories of event structures $\mathcal{E}$ and $\mathcal{SE}$ with total maps. In particular, general spans have been described for maps in $\mathcal{SE}$. Analogous definitions and results hold for rigid maps, and for spans in $\mathcal{SE}_r$—event structures with symmetry and rigid maps. Total maps on event structures with symmetry can be obtained as Kleisli maps w.r.t. a monad $\mathcal{S}aug$ on $\mathcal{SE}_r$—see [29]. It appears that we can ground all the maps and spans of event structures of interest in $\mathcal{SE}_r$. The category $\mathcal{SE}_r$ is emerging as *the* fundamental category of event structures.

6.2 Event types

The particular bicategory of spans

$$\begin{tikzcd} & E \arrow[dl] \arrow[dr] & \\ A_* & & !B \end{tikzcd}$$

is already quite an interesting framework for the semantics of higher-order processes. It supports types including:

- Prefix types $\bullet !T$: in which a single event $\bullet$ prefixes $!T$ for an event structure with symmetry T.
- Sum types $\Sigma_{\alpha \in A} T_\alpha$: the sum of a collection T_α, for $\alpha \in A$, of event structures with symmetry—the sum functor is described in Section 5.1. Sum types may also be written $a_1 T_1 + \cdots + a_n T_n$ when the indexing set is finite. The empty sum type is the empty event structure $\emptyset$.
- Tensor types $T_1 \otimes T_2$: the product in $\mathcal{SE}_p$.
- Function types $T_1 \multimap T_2$: a form of function space, defined as the product $(T_1)_* \times !T_2$ in $\mathcal{SE}$.[4]
- Recursively defined types: treated for example as in [23, 25].

The types describe the events and basic causalities of a process, and in this sense are examples of *event types*, or *causal types*, of a process. (One can imagine other kinds of spans and variations in the nature of event types.)

As an example, the type of a process only able to do actions within $a_1, \cdots, a_k$ could be written

$$a_1 \bullet !\emptyset + \cdots + a_k \bullet !\emptyset \,,$$

which we condense to $a_1 + \cdots + a_k$, as it comprises the event structure with events $a_1, \cdots, a_k$ made in pairwise-conflict, with the identity relation of causal dependency. The judgement that a closed process, represented by an event structure with symmetry E, has this type would be associated with a degenerate span from the biterminal $\emptyset_*$ to $!(a_1 + \cdots + a_k)$, so essentially with a map

$$l : E \to !(a_1 + \cdots + a_k)$$

in $\mathcal{SE}$, 'labelling' events by their actions. By Proposition 9 (i), there is an equivalence

$$!a_1 \parallel \cdots \parallel !a_k \simeq !(a_1 + \cdots + a_k) \,,$$

and a process of this type can only do actions $a_1, \cdots, a_k$, though with no bound on how many times any action can be done.

The type of CCS, with channels A, can be written as

$$Act = \tau \bullet !\emptyset \; + \; \Sigma_{\bar{a} \in \bar{A}} \bullet !\emptyset \; + \; \Sigma_{a \in A} \bullet !\emptyset \,.$$

We can describe the parallel composition of CCS by a partial function from the events $Act \otimes Act$ to the events $!Act$, expressing how events combine to form synchronization events (the second line), or can occur asynchronously (the first):

$$\begin{aligned} &(\alpha, *) \mapsto \mu^!_{Act}(0, \eta^!_{Act}(\alpha)) \,, \quad (*, \alpha) \mapsto \mu^!_{Act}(1, \eta^!_{Act}(\alpha)) \,, \\ &(a, \bar{a}), \; (\bar{a}, a) \mapsto \eta^!_{Act}(\tau), \quad \text{and undefined otherwise.} \end{aligned}$$

[4] Although this function space seems hard to avoid for this choice of span and tensor, we don't quite have $\_ \otimes B$ a left biadjoint to $B \multimap \_$.

This partial function is also a partial map of event structures from $Act \otimes Act$ to $!Act$—it would have violated local injectivity and not been a map of event structures, had we chosen simply $\eta^!_{Act}(\alpha)$ as the resulting events in the first two clauses. The partial function is readily interpreted as a span from $Act \otimes Act$ to $!Act$—its vertex is essentially the domain of definition of the partial function. Post-composing its left 'leg' with $\eta^*_{Act \otimes Act}$ we obtain a span from $(Act \otimes Act)_*$ to $!Act$ which denotes the parallel composition of CCS. Given two CCS processes represented by degenerate spans, we can combine them to a process with event type $Act \otimes Act$, denoting a degenerate span ending in $!(Act \otimes Act)$. Its composition with the span for parallel composition can be shown to give the traditional event-structure semantics of parallel composition in CCS [23, 25, 26, 21].

In fact there is a general way to define spans from partial functions on events which respect symmetry. There is a functor from event structures with symmetry $\mathcal{SE}$ to equivalence relations; it takes an event structure with symmetry A to the equivalence relation $|A|$ induced by the symmetry on the set of events. The functor is enriched in equivalence relations and has a right biadjoint \$ which takes an equivalence relation $(L, R \subseteq L \times L)$ to the event structure with symmetry $!(L, l, r : R \to L)$, where we understand L as an event structure with events in pairwise conflict with trivial causal dependency, R similarly, and with symmetry maps given as the obvious projections from R to L. (The biadjunction between event structures with symmetry and equivalence relations relies on the event structures being consistent-countable.) For event structures with symmetry A and B, a partial function respecting equivalence relations from $|A|$ to $|B|$ can be regarded as a span

$$\begin{tikzcd}
& D \arrow[dl, hook'] \arrow[dr] & \\
{|A|} & & {|B|}
\end{tikzcd}$$

in the category of equivalence relations—the equivalence relation D being where the partial function is defined. The unit of the biadjunction with equivalence relations has components $A \to \$|A|$ and $B \to \$|B|$, so by applying \$ to the span above and taking successive pseudo pullbacks we obtain a span from A to B:

$$\begin{tikzcd}
& & \cdot \arrow[dl] \arrow[dr] & & \\
& \cdot \arrow[dl] \arrow[dr] & & \cdot \arrow[dl] \arrow[dr] & \\
A \arrow[dr] & & \$D \arrow[dl] \arrow[dr] & & B \arrow[dl] \\
& \$|A| & & \$|B| &
\end{tikzcd}$$

A partial function between on events may not be so simple to define directly by case analysis on events. This is because the events that arise in products of event structures can be quite complicated; the events of a product $A \otimes B$ of event structures A and B are perhaps best seen as prime configurations of a product of stable families—see Appendix B. Their complexity contrasts with the simplicity of the events arising in constructions on stable families; the events of the corresponding product of stable families are simple pairs $(a, *)$, $(*, b)$ and (a, b), where a and b are events of the components. For this reason it can be easiest to define a partial map on event structures (so a partial function on their events) via a partial map between their representations as stable families. This is so below, in a putative 'true concurrency' definition of a version of higher-order CCS and its parallel composition.

A form of higher-order CCS could reasonably be associated with the recursive type

$$T = \tau \bullet !T \ + \ \Sigma_{\bar{a} \in \bar{A}} \bullet !(T \otimes T) \ + \ \Sigma_{a \in A} \bullet !(T \multimap T)\,,$$

specifying that an event of a higher-order CCS process is either a 'process' event following a τ-event, a 'concretion' event following an output synchronization $\bar{a} \in \bar{A}$, or an 'abstraction' event following an input synchronization $a \in A$. Why is the first component in the type T of the form $\tau \bullet !T$ and not just $\tau \bullet !\emptyset$? Without the present choice I cannot see how to ensure that in the parallel composition an interaction between a concretion and abstraction event always follows a corresponding synchronization at their channels.

Parallel composition in higher-order CCS would be associated with a typing judgment $x : T, y : T \vdash (x \,|\, y) : T$. The typing judgment should denote a span from $(T \otimes T)_*$ to $!T$. As above, we can define a tentative parallel composition via a partial function from $|T \otimes T|$ to $|!T|$. The partial function should describe when and how events of T combine. Because events of the product $T \otimes T$ are quite complicated we must face the difficulties outlined above. However, first we need a makeshift syntax for events in T. Events of higher-order CCS are either internal events τ, subsequent process events $\tau.(i, t)$, output synchronizations $\bar{a}$, subsequent concretion events $\bar{a}.(i, c)$, input synchronizations a, or subsequent abstraction events $a.(j, f)$—the natural numbers i, j index the copies in !-types. In the notation for events of the product $T \otimes T$ we exploit the way it is built from a product of stable families; in the product of stable families out of which $T \otimes T$ is constructed events have the simple form of pairs $(t, *)$, $(*, t)$ or (t, t'), where t and t' are events of T. We can define a partial map from this stable family to the stable family of $!T$ by case analysis on events:

$$t \,|\, * \ = \ \mu^!_T(0, \eta^!_T(t))\,, \quad * \,|\, t \ = \ \mu^!_T(1, \eta^!_T(t))\,,$$
$$a \mid \bar{a} \ = \ \bar{a} \mid a \ = \ \eta^!_T(\tau)\,,$$
$$a.(i, f) \mid \bar{a}.(j, c) \ = \ \bar{a}.(i, c) \mid a.(j, f) \ = \ \eta^!_T(\tau.\mu^!_T([i, j],\ (f \,|\, c)))$$

provided $(f \,|\, c)$ is defined,

$$\tau.(i, t) \mid \tau.(j, t') \ = \ \mu^!_T([i, j], (t \,|\, t')) \ \text{ provided } (t \,|\, t') \text{ is defined,}$$
$$\tau.(i, t) \mid \alpha \ = \ \mu^!_T(i, (t \,|\, \alpha))\,, \quad \alpha \mid \tau.(j, t) \ = \ \mu^!_T(j, (\alpha \,|\, t))$$

provided α is not of the form $\tau.(k, t'')$, and undefined otherwise.

We have combined indices i, j using an injective pairing $[i, j]$ of natural numbers. The definition above relies on our simultaneously defining not just how process events combine, but also how 'abstraction' events f in type $T \multimap T$ and 'concretion' events c in type $T \otimes T$ combine to form a process event $(f \mid c)$ in type $!T$. We postpone the full definition. Although provisional, I hope the example helps illustrate the aims and present difficulties—there may be difficulties that I'm not aware of.

Clearly the syntax of operations to accompany the types is unfinished and really needed. But I believe the examples indicate the potential of a more thorough study of event types and give a flavour of the style of definition they might support, a method of definition which breaks away from traditional 'interleaving' approaches to concurrency.

6.3 Nondeterministic dataflow and affine-HOPLA

'Stable' spans of event structures have been used to give semantics to nondeterministic dataflow [19] and the higher-order process language affine-HOPLA [16]. They are generalisations of Berry's stable functions [2]: deterministic stable spans correspond to stable functions—see [19]. A stable span

$$\begin{tikzcd} & E \arrow[dl, "dem"'] \arrow[dr, "out"] & \\ A & & B \end{tikzcd}$$

consists of a 'demand' map $dem : E \to A$ and a rigid map $out : E \to B$. That dem is a demand map means that it is a function from $\mathcal{C}^o(A)$ to $\mathcal{C}^o(B)$ which preserves unions of configurations when they exist. An equivalent way to view the demand map dem is as a function from the events of E to finite configurations of A such that if $e \le e'$ then $dem(e) \subseteq dem(e')$, and if $X \in \mathrm{Con}$ then $demX \uparrow$, *i.e.*, the demands are compatible. The intuition is that $dem(e)$ is the minimum input required for the event e to occur; when it does $out(e)$ is observed in the output. (The stable span is *deterministic* when $demX \uparrow$ implies $X \in \mathrm{Con}$, for X a finite subset of events in E.)

On the face of it demand maps are radically different from rigid maps of event structures. They can however be recovered as Kleisli maps associated with a pseudo monad $\mathcal{H}$ on event structures with symmetry and rigid maps.

Roughly the pseudo monad $\mathcal{H}$ adjusts the nature of events so that they record the demand history on the input. This enables stable spans to be realized as spans

$$\begin{tikzcd} & E \arrow[dl, "in"'] \arrow[dr, "out"] & \\ \mathcal{H}(A) & & B \end{tikzcd}$$

of rigid maps in $\mathcal{SE}_r$. Such spans are a special case of the general spans of Section 6.1, with the identity monad on the right-hand-side. Because of 'Seely conditions' $\mathcal{H}(E \parallel F) \simeq \mathcal{H}(E) \times \mathcal{H}(F)$ and $\mathcal{H}(\emptyset) \simeq \top$ relating parallel composition $\parallel$ and its unit, the empty event structure $\emptyset$, to product $\times$ and the biterminal object $\top$ in $\mathcal{SE}_r$, we obtain a description of the function space, w.r.t. parallel composition $A \parallel B$, as $A \multimap B = \mathcal{H}(A) \times B$. A very different route to the definition of function space using stable families is described in the PhD thesis [16]. The pseudo monad $\mathcal{H}$ and the biadjunction which induces it are described in [29].

6.4 Unfoldings

Another application of symmetry is to the unfolding of Petri nets with multiple tokens, and the unfolding of higher-dimensional automata (hda's) [7]. Unfoldings of 1-safe Petri nets to occurrence nets and event structures were introduced in [15], and have since been applied in a variety of areas from model checking to self-timed circuits and the fault diagnosis of communication networks. The unfoldings were given a universal characterisation a little later in [24] (or see [21]) and this had the useful consequence of providing a direct proof that unfolding preserved products and so many parallel compositions. There is an obstacle to an analogous universal characterisation of the unfolding of nets in which places/conditions hold with multiplicities: the symmetry between the multiple occurrences in the original net is lost in unfoldings to standard occurrence nets or event structures, and this spoils universality through non-uniqueness. However through the introduction of symmetry uniqueness up to symmetry obtains, and a universal characterisation can be regained [9].

We can illustrate the role symmetry plays in the unfolding of nets and hda's through a recent result relating event structures with symmetry to certain presheaves.[5] Let $\mathbb{P}$ be the category of finite elementary event structures (so essentially finite partial orders) with rigid maps. Form the presheaf category $\widehat{\mathbb{P}}$ which by definition is the functor category $[\mathbb{P}^{op}, \mathbf{Set}]$. From [27] we obtain that event structures with rigid maps (called 'strong' in [27]) embed fully and faithfully in $\widehat{\mathbb{P}}$ and are equivalent to those presheaves which are *separated* w.r.t. the Grothendieck topology with basis collections of jointly surjective maps in $\mathbb{P}$, and satisfy a further *mono* condition. Presheaves over $\widehat{\mathbb{P}}$ are thus a kind of generalised event structure.

There is clearly an inclusion functor $I : \mathbb{P} \hookrightarrow \mathcal{SE}_r$ of finite elementary event structures into event structures with symmetry and rigid maps. Thus there is a functor $F : \mathcal{SE}_r \to \widehat{\mathbb{P}}$ taking an event structure with symmetry E to the presheaf $\mathcal{SE}_r(I(\_), E)/\!\sim$. Event structures with symmetry yield more than just separated presheaves, and quite which presheaves they give rise to is not yet understood. But by restricting to event structures with symmetry $(E, l, r : S \to E)$ for which the symmetry is *strong*, in the sense that the mono $\langle l, r\rangle : S \to E \times E$ reflects consistency, we will always obtain nonempty separated presheaves. Let $\mathcal{SSE}_r$ be the category of event structures with strong symmetry and rigid maps. Let $\mathrm{Sep}(\mathbb{P})$ be the full subcategory of non-empty separated presheaves.

[5] The result is inspired by joint work with the Sydney Concurrency Group: Richard Buckland, Jon Cohen, Rob van Glabbeek and Mike Johnstone.

So restricted, we obtain a functor $F : \mathcal{SSE}_r \to \widehat{\mathbb{P}}$ taking an event structure with strong symmetry E to the nonempty separated presheaf $\mathcal{SSE}_r(J(\_), E)/\sim$. The functor F can be shown to have a right biadjoint, a functor G, producing an event structure with strong symmetry from a nonempty separated presheaf. The right biadjoint G is full and faithful (once account is taken of the the equivalence $\sim$ on maps). (The existence of G relies on the event structures being consistent-countable.) It shows how separated presheaves embed via a reflection fully and faithfully in event structures with symmetry:

$$\begin{tikzcd} \mathcal{SSE}_r \arrow[r, bend left, "F"] \arrow[r, phantom, "\perp"] & \mathrm{Sep}(\mathbb{P}) \arrow[l, bend left, "G"] \end{tikzcd} . \qquad (\dagger)$$

The proof of the biadjunction has only been carried out for rigid maps, the reason why we have insisted that the maps of event structures in this section be rigid. (One could hope for a similar biadjunction without restricting F to strong symmetries.)

Higher-dimensional automata [7] are most concisely described as cubical sets, *i.e.* as presheaves over $\mathbb{C}$, a category of cube shapes of all dimensions with maps including *e.g.* 'face' maps, specifying how one cube may be viewed as a (higher-dimensional) face of another. We can identify the category of hda's with the presheaf category $\widehat{\mathbb{C}}$. There are some variations in the choice of maps in $\mathbb{C}$, according to whether the cubes are oriented and whether degeneracy maps are allowed. For simplicity we assume here that the cubes are not oriented and have no degeneracy maps, so the maps are purely face maps. Roughly, then the maps of $\mathbb{P}$ and $\mathbb{C}$ only differ in that maps in $\mathbb{P}$ fix the initial empty configuration whereas face maps in $\mathbb{C}$ are not so constrained. By modifying the maps of $\mathbb{P}$ to allow the initial configuration to shift under maps, we obtain a category $\mathbb{A}$ into which both $\mathbb{P}$ and $\mathbb{C}$ include:

$$\begin{tikzcd} \mathbb{P} \arrow[r, hook, "J"] & \mathbb{A} & \mathbb{C} \arrow[l, hook', "K"'] \end{tikzcd}$$

Now we can construct a functor from $H : \mathbb{P} \to \widehat{\mathbb{C}}$; it takes p in $\mathbb{P}$ to the presheaf $\mathbb{A}(K(\_), J(p))$. Taking its left Kan extension over the Yoneda embedding of $\mathbb{P}$ in $\widehat{\mathbb{P}}$ we obtain a functor

$$H_! : \widehat{\mathbb{P}} \to \widehat{\mathbb{C}} \ .$$

For general reasons [4], the functor $H_!$ has a right adjoint H^* taking an hda Y in $\widehat{\mathbb{C}}$ to the presheaf $\widehat{\mathbb{C}}(H(\_), Y)$ in $\widehat{\mathbb{P}}$:

$$\begin{tikzcd} \widehat{\mathbb{P}} \arrow[r, bend left, "H_!"] \arrow[r, phantom, "\perp"] & \widehat{\mathbb{C}} \arrow[l, bend left, "H^*"] \end{tikzcd} . \qquad (\ddagger)$$

We cannot quite compose the biadjunctions (†) and the adjunction (‡) because (†) is only for separated presheaves. However restricting to hda's which are separated, now w.r.t. a basis of jointly surjective maps in $\mathbb{C}$,[6] will ensure that they are sent to separated presheaves over $\mathbb{P}$ and so to event structures with symmetry. General Petri nets give rise to separated hda's (for example, with the 'self-concurrent individual token interpretation' of [7]). So we obtain a rather abstract construction of an unfolding of general nets to event structures with symmetry. Again, much more needs to be done, both mathematically in seeking a generalisation of the biadjunction (†) to all event structures with symmetry, and in understanding unfoldings concretely so that they can be made amenable algorithmically.

Acknowledgments. I'm grateful to Marcelo Fiore, Martin Hyland, Gordon Plotkin, Lucy Saunders-Evans, Pawel Sobocinski, Sam Staton, Dominic Verity and the Sydney Concurrency Group for discussions and encouragement. I acknowledge the partial support of EPSRC grant GR/T22049/01.

References

1. Abramsky, S., Jagadeesan, R., and Malacaria, P., Full Abstraction for PCF. Information and Computation vol. 163, 409–470, 2000.
2. Berry, G., Modèles completement adéquats et stables des λ-calculs typés. Thèse de Doctorat d'Etat, Université de Paris VII, 1979.
3. Burroni, A., T-catégories. Cahiers de topologie et géométrie différentielle, XII 3, 1971.
4. Cattani, G.L., and Winskel, G., Profunctors, open maps and bisimulation. MSCS, 2005.
5. Cheng, E., Hyland, J.M.E., and Power, A.J., Pseudo-distributive laws. ENTCS 83, 2004.
6. Crazzolara, F., and Winskel, G., Composing Strand Spaces. FSTTCS'02, 2002.
7. Glabbeek, R.J.van, On the expressiveness of higher dimensional automata. EXPRESS 2004, ENTCS 128(2), 2005.
8. Doghmi, S.F., Guttman, J.D., and Thayer, F.J., Searching for shapes in cryptographic protocols. TACAS'07, 2007.
9. Hayman, J., and Winskel, G., The unfolding of general Petri nets. Forthcoming.
10. Johnstone, P., Sketches of an elephant, a topos theory compendium, vol.1. *OUP*, 2002.
11. Joyal, A., Nielsen, M., and Winskel, G., Bisimulation from open maps. *LICS '93 special issue of Information and Computation*, 127(2):164–185, 1996. Available as BRICS report, RS-94-7.
12. Kahn, G., and Plotkin, G.D., Concrete domains. *TCS*, 121(1& 2):187–277, 1993.
13. Koslowski, J., A monadic approach to polycategories. *Theory and Applications of Categories*, 14(7):125–156, 2005.
14. Mac Lane, S. *Categories for the Working Mathematician.* Springer, 1971.
15. Nielsen, M., Plotkin, G.D., and Winskel, G., Petri nets, event structures and domains. *TCS*, 13(1):85–108, 1981.

[6] For a separated hda, cubes which share the same 1-dimensional edges must be equal (so 'no ravioli').

16. Nygaard, M., Domain theory for concurrency. PhD Thesis, University of Aarhus, 2003.
17. Nygaard, M., and Winskel, G., Domain theory for concurrency. *TCS* 316: 153–190, 2004.
18. Power, A.J., 2-Categories. *BRICS Lecture Notes, Aarhus University,* March, 1998.
19. Saunders-Evans, L., and Winskel, G., Event structure spans for non-deterministic dataflow. Proc. Express'06, ENTCS, 2006.
20. Saunders-Evans, L., Events with persistence. Forthcoming PhD thesis, University of Cambridge Computer Laboratory, 2007.
21. Winskel, G., and Nielsen, M., Models for Concurrency. Handbook of Logic and the Foundations of Computer Science, vol. 4, pages 1-148, OUP, 1995.
22. Winskel, G., *Events in Computation.* PhD thesis, Univ. of Edinburgh, 1980. Available from `http://www.cl.cam.ac.uk/users/gw104`.
23. Winskel, G., Event structure semantics of CCS and related languages. ICALP 82, Springer–Verlag LNCS 140, 1982. Extended version available from `http://www.cl.cam.ac.uk/users/gw104`.
24. Winskel, G., A new definition of morphism on Petri Nets. STACS'84: 140-150, 1984.
25. Winskel, G., Event structures. Invited lectures for the Advanced Course on Petri nets, September 1986. Springer LNCS, vol.255, 1987.
26. Winskel, G., An introduction to event structures. REX summerschool in temporal logic, Springer LNCS, vol.354, 1988.
27. Winskel, G., Event structures as presheaves—two representation theorems. CONCUR 1999. Springer LNCS, vol.1664, 1999.
28. Winskel, G., Relations in concurrency. Invited talk, LICS'05, 2005.
29. Winskel, G., Event structures with symmetry. In the Plotkin Festschrift. ENTCS 172, 2007. See `http://www.cl.cam.ac.uk/users/gw104` for corrections.

A Appendix: Equivalence relations [10]

Assume a category with pullbacks. Let E be an object of the category. A *relation* on E is a pair of maps $l, r : S \to E$ for which l, r are *jointly monic*, *i.e.* for all maps $x, y : D \to S$, if $lx = ly$ and $rx = ry$, then $x = y$. Equivalently, if the category has binary products, a relation on E is a pair of maps $l, r : S \to E$ for which the mediating map $\langle l, r\rangle : S \to E \times E$ is monic. The relation is an *equivalence relation* in the category iff it is:

Reflexive: there is a (necessarily unique) map ρ such that

$$\begin{tikzcd} & E \arrow[dl, "\mathrm{id}_E"'] \arrow[d, dotted, "\rho"] \arrow[dr, "\mathrm{id}_E"] & \\ E & S \arrow[l, "l"] \arrow[r, "r"'] & E \end{tikzcd}$$

commutes;

Symmetric: there is a (necessarily unique) map σ such that

$$\begin{tikzcd}
& S \arrow[dl, "r"'] \arrow[d, dotted, "\sigma"] \arrow[dr, "l"] & \\
E & S \arrow[l, "l"] \arrow[r, "r"'] & E
\end{tikzcd}$$

commutes;
Transitive: there is a (necessarily unique) map τ such that

$$\begin{tikzcd}
& & P \arrow[dl, "f"'] \arrow[d, dotted, "\tau"] \arrow[dr, "g"] & & \\
& S \arrow[dl, "l"'] \arrow[dr, "r"] & S \arrow[dll, "l"] \arrow[drr, "r"] & S \arrow[dl, "l"] \arrow[dr, "r"] & \\
E & & E & & E
\end{tikzcd}$$

commutes, where P, f, g is a pullback of r, l.

B Appendix: Stable families

Event structures can be obtained from finitary prime algebraic domains. One convenient way to construct finitary prime algebraic domains is from stable families [23]. The use of stable families facilitates constructions such as products and pullbacks of event structures.

The use of stable families facilitates definitions on event structures.

Definition A *stable family* comprises $\mathcal{F}$, a family of finite subsets, called *configurations*, satisfying:
Completeness: $Z \subseteq \mathcal{F}$ & $Z\uparrow \Rightarrow \bigcup Z \in \mathcal{F}$;
Coincidence-freeness: For all $x \in \mathcal{F}$, $e, e' \in x$ with $e \neq e'$,

$$(\exists y \in \mathcal{F}.\ y \subseteq x \ \&\ (e \in y \iff e' \notin y)) \ ;$$

Stability: $\forall Z \subseteq \mathcal{F}.\ Z \neq \emptyset$ & $Z\uparrow \Rightarrow \bigcap Z \in \mathcal{F}$.

For $Z \subseteq \mathcal{F}$, we write $Z\uparrow$ to mean compatibility, *i.e.*

$$\exists x \in \mathcal{F} \forall z \in Z.\ z \subseteq x \ .$$

Configurations of stable families each have their own local order of causal dependency, so their own prime sub-configurations generated by their events. We can build an event structure by taking the events of the event structure to comprise the set of all prime sub-configurations of the stable family.

Definitions and Proposition Let x be a configuration of a stable family $\mathcal{F}$. For $e, e' \in x$ define

$$e' \leq_x e \text{ iff } \forall y \in \mathcal{F}.\ y \subseteq x \ \&\ e \in y \Rightarrow e' \in y.$$

When $e \in x$ define the prime configuration

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

Then $\leq_x$ is a partial order and $[e]_x$ is a configuration such that

$$[e]_x = \{e' \in x \mid e' \leq_x e\}.$$

Moreover the configurations $y \subseteq x$ are exactly the down-closed subsets of $\leq_x$.

Definition and Proposition Let $\mathcal{F}$ be a stable family. Then, $\mathrm{Pr}(\mathcal{F}) =_{\mathrm{def}} (P, \mathrm{Con}, \leq)$ is an event structure where:

$$\begin{aligned}
&P = \{[e]_x \mid e \in x \ \&\ x \in \mathcal{F}\}\ ,\\
&Z \in \mathrm{Con} \text{ iff } Z \subseteq P \ \&\ \bigcup Z \in \mathcal{F} \ \text{ and,}\\
&p \leq p' \text{ iff } p, p' \in P \ \&\ p \subseteq p'\ .
\end{aligned}$$

This proposition furnishes a way to construct an event structure with events the prime configurations of a stable family. In fact we can equip the class of stable families with maps (the definitions are the same as those for event structures). The configurations of an event structure form a stable family, so in this sense event structures are included in stable families. With respect to any of the maps (rigid, total or partial), the “inclusion” functor from the category of event structures to the category of stable families has a right adjoint, which on objects is the construction we have just given, producing an event structure from a stable family. The products w.r.t. total and partial maps are hard to define directly on the event structures of this article. It is however straightforward to define the products of stable families [23, 29]. Right adjoints preserve limits, and so products in particular. Consequently we obtain products of event structures by first regarding them as stable families, and then producing the event structure from the product of the stable families. Pullbacks of event structures are obtained by restricting products to the appropriate equalizing set. See [29] for more details.