Section 2: The scientific proposal

a. State-of-the-art and objectives

Denotational semantics and domain theory of Scott and Strachey set the standard for semantics of computation. The theory provided a global mathematical setting for sequential computation, and thereby placed programming languages in connection with each other; connected with the mathematical worlds of algebra, topology and logic; and inspired programming languages, type disciplines and methods of reasoning. Despite the many striking successes it has become very clear that many aspects of computation do not fit within the traditional framework of denotational semantics and domain theory. In particular, classical domain theory has not scaled up to the more intricate models used in interactive/distributed computation. Nor has it been as operationally informative as one could hope.

How are we to extend the methodology of denotational semantics to the much broader forms of computational processes we need to design, understand and analyze today? How are we to maintain clean algebraic structure and abstraction alongside the operational nature of computation?

Here we expand on the Extended Synopsis and describe in more detail the problems with traditional semantics that lead us to seek a next generation of semantics; why this should be an intensional theory, concerned also with the ways values are computed; and how this leads us to seek a comprehensive theory of causal models. As we will see, this is because causal models, such as event structures, have appeared unexpectedly in the role of expressing the ways in which values are computed. But causal models, as they are currently understood and used, have themselves several problems, mainly due to their overly-concrete nature, which has obstructed their expressivity and range of use. A closer examination points to the introduction of a formal treatment of behavioural symmetry as the means to increase expressivity of causal models. The way forward to the next generation of semantics is through the key elements of *events*, *causality* and *symmetry*.

Recall the objectives of ECSYM:

Objective 1. A comprehensive semantic theory—one which includes that of causal models—together with rich metalanguage(s) and structured operational semantics, extracted from the denotational semantics;

Objective 2. New techniques for event-based, causal reasoning, the beginnings of which are suggested by the mathematics;

Objective 3. To incorporate quantitative reasoning through enrichment with probability and time—the mathematics and applications suggest a way;

Objective 4. To develop application methods, especially where causal models (have the potential to) play a central role, including distributed and parallel computation, and systems biology.

To these ends, ECSYM assembles a world-leading team of theoretical computer scientists and mathematicians: Principal Investigator: Glynn Winskel; Senior researchers: Marcelo Fiore, Martin Hyland, Andy Pitts; Junior researchers: Richard Garner, Jonathan Hayman, Chung-Kil Hur, Sam Staton. The PI's discoveries in semantics of computation, old and recent, underpin the foundation on which ECSYM builds.

1 Background

1.1 Causal models

One reason why ECSYM is especially timely is because of the current rebirth of interest in causal models. Causal models are alternatively described as: causal-dependence models; independence models; non-interleaving models; true-concurrency models; and partial-order models. They include Petri nets, event structures, Mazurkiewicz trace languages, transition systems with independence, multiset rewriting, and many more. The models share the central feature that they represent processes in terms of the events they can perform, and that they make explicit the causal dependency and conflicts between events.

Causal models have arisen, and have sometimes been rediscovered as *the* natural model, in many diverse and often unexpected areas of application:

• *Security protocols:* for example, forms of event structure, strand spaces, support reasoning about secrecy and authentication through causal relations and freshness of names [51, 13, 15];

• *Systems biology:* ideas from Petri nets and event structures are used in taming the state-explosion in the stochastic simulation of biochemical processes and in the analysis of biochemical pathways [16];

• *Hardware:* in the design and analysis of asynchronous circuits [32];

• *Types and proof:* event structures appear as representations of propositions as types, and of proofs [22, 14];

• *Nondeterministic dataflow:* where numerous researchers have used or rediscovered causal models in providing a compositional semantics—see [44] and its references;

• *Network diagnostics:* in the patching together local of fault diagnoses of communication networks [5];

• *Logic of programs:* in concurrent separation logic where artificialities in Brookes' pioneering soundness proof are obviated through a Petri-net model [28];

• *Partial order model checking:* following the seminal work of McMillan [36] the unfolding of Petri nets is exploited in recent automated analysis of systems [18];

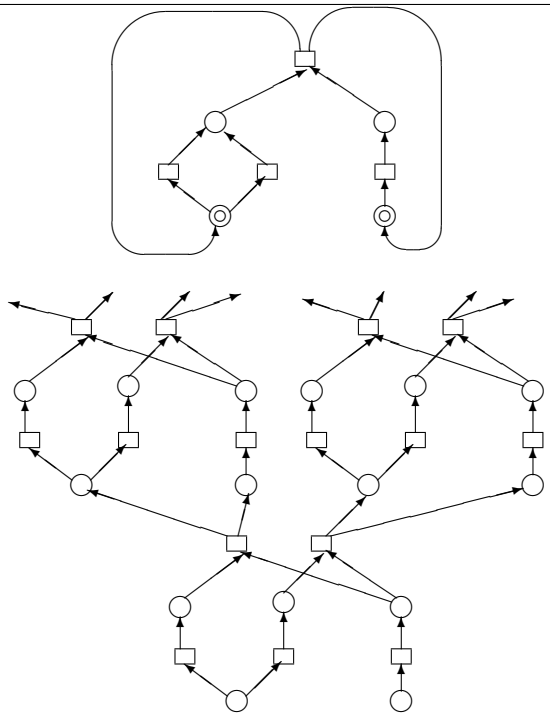
• *Distributed computation:* event structures appear both classically [33] and recently in the Bayesian analysis of trust [38] and modelling multicore memory [49].

To illustrate the close relationship between Petri nets and the 'partial-order models' of occurrence nets and event structures, we sketch how a (1-safe) Petri net can be unfolded first to a net of occurrences and from there to an event structure [39]. The unfolding construction (due to the PI, Nielsen and Plotkin) is analogous to the well-known method of unfolding a transition system to a tree, and is central to several analysis tools in the applications above. In the figure, the net on top has loops. The net below it is its *occurrence-net unfolding*. It consists of all the occurrences of conditions and events of the original net, and is infinite because of the original repetitive behaviour. The occurrences keep track of what enabled them. The simplest form of event structure arises by abstracting away the conditions in the occurrence net and capturing their role in relations of causal dependency and conflict on event occurrences.

The relations between the different forms of causal models are well understood. Despite this and their often very successful, specialized applications, causal models lack a *comprehensive* theory which would support:

• Their systematic use in giving *structured operational semantics* to a broad range of programming and process languages; while many examples exist of Petri-net semantics of processes and languages there are presently no *generally-accepted* standard techniques for describing operational semantics using Petri nets on similar lines to Plotkin's 'Structural Operational Semantics.'

• An expressive '*domain theory*' with rich higher-order type constructions needed by mathematical semantics. It should for example cover the standard



A Petri net and its occurrence-net unfolding

event-structure semantics of CCS [56], extend to higher-order CCS, and support the formalization and analysis of distributed algorithms. Such a domain theory would go beyond traditional domain theory, in which types are represented as partial orders of information, in that causal models (perhaps enriched, *e.g.* with probability) would feature as denotations.

1.1.1 Problems in traditional causal models

Unfoldings of general Petri nets: In general nets conditions can hold with multiplicities. While their occurrence net unfoldings can be defined, there can be no universal characterisation like that for unfoldings of 1-safe nets. The symmetry intrinsic to nets with multiplicities spoils uniqueness.

Weak bisimulation: Just as for labelled transition systems weak bisimulation between labelled event structures (in which we abstract from invisible actions, generally labelled τ) can be explained as strong bisimulation between the results of 'hiding' the invisible actions. Whereas the 'hiding' operation on a transition system is again a transition system, the hiding operation on an event structures does not always yield an event structure.

Name generation: There are methods to represent the generation of new/fresh names in causal models, but the methods are overly concrete in the sense that they ignore the implicit symmetry on names. Presently there are difficulties in extending work on an event-structure semantics of the pi-Calculus to the whole

language because of the absence of a key algebraic operation—a form of new-name abstraction on event structures.

Varying maps: The basic maps of event structures are 'event-linear' in the sense that an event of output has depended on precisely one event of input. This is sometimes too restrictive. But changing the maps generally changes important categorical constructions. One would like to settle on some category of basic maps and then have a systematic way to vary the nature of maps within it.

Higher-order processes: causal models, as used traditionally, do not represent general higher-order processes (higher-order in the sense that they could treat processes themselves as input and output values).

Although it is far from obvious, *all* these anomalies stem from ignoring the symmetry intrinsic to the constructions needed. We will see that once symmetry is taken account of ia comprehensive theory, including a 'domain theory' which encompasses causal models, is within reach.

1.2 Domain theory

In the earliest days of computer science it became accepted that a computation was essentially an (effective) partial function $f : \mathbb{N} \to \mathbb{N}$ between the natural numbers. This view underpins the Church-Turing thesis on the universality of computability.

As computer science matured it demanded increasingly sophisticated mathematical representations of processes. The pioneering work of Strachey and Scott in the denotational semantics of programs assumed a view of a process still as a function $f : D \to D'$, but now acting in a continuous fashion between datatypes represented as special topological spaces, 'domains' D and D'; reflecting the fact that computers can act on complicated, conceptually-infinite objects, but only by virtue of their finite approximations.

In the 1960's, around the time that Strachey started the programme of denotational semantics, Petri advocated his radical view of a process, expressed in terms of its events and their effect on local states—a model which addressed directly the potentially distributed nature of computation, but which, in common with many other current models, ignored the distinction between data and process implicit in regarding a process as a function. Here it is argued that today an adequate notion of process requires a marriage of Petri's view of a process and the vision of Scott and Strachey. An early hint in this direction came in answer to the following question.

What is the information order in domains? There are essentially two answers in the literature, the '*topological*,' the most well-known from Scott's work, and the '*temporal*,' arising from the work of Berry:

• *Topological*: the basic units of information are *propositions* describing finite properties; more information corresponds to more propositions being true. Functions are ordered pointwise.

• *Temporal*: the basic units of information are *events*; more information corresponds to more events having occurred over time. Functions are restricted to 'stable' functions and ordered by the intensional 'stable order,' in which common output has to be produced for the same minimal input. Berry's specialized domains 'dI-domains' are represented by event structures.

In truth, Berry developed 'stable domain theory' by a careful study of how to obtain a suitable category of domains with stable rather than all continuous functions. He arrived at the axioms for his 'dI-domains' because he wanted function spaces (so a cartesian-closed category). The realization that dI-domains were precisely those domains which could be represented by event structures, came later in work of the PI [57, 58].

1.2.1 Event structures

An *event structure* comprises $(E, \leq, \mathrm{Con})$, consisting of a set E, of *events* (event occurrences), partially ordered by $\leq$, the *causal dependency relation* which satisfies $\{e' \mid e' \leq e\}$ is finite for all $e \in E$, and a family Con of finite subsets of E, the *consistency relation*, which satisfy the natural axioms. The relation $e' \leq e$ expresses that the event e can only occur after the event e'. The consistency relation picks out those events which can occur together.

A configuration of the event structure gives a record of the events that have occurred. If an event e has occurred then so must all events e' on which e depends have occurred previously, and any finite set of events that have occurred should be consistent. Accordingly, the *configurations*, $\mathcal{C}(E)$, of an event structure E consist of those subsets $x \subseteq E$ which are down-closed w.r.t. $\leq$ and for which every finite subset belongs to Con.

Configurations stand for histories, given in terms of the events that have occurred; the events inherit an order from the event structure. In particular, for an event e the set $[e] =_{\mathrm{def}} \{e' \in E \mid e' \leq e\}$ is a configuration including the whole causal history of the event e. The inclusion relation $x \subseteq x'$ means that x is a sub-history of x'. Ordered by inclusion the configurations form

a domain $(\mathcal{C}(E), \subseteq)$—in fact, when E is countable, a dI-domain as discovered by Berry, and all such are so obtained. In this way event structures have been used to represent a rich variety of types, even for polymorphism.

1.2.2 Problems in domain theory

Nondeterminism: For traditional ('topological') domain theory the problem of adjoining nondeterminism was solved by Plotkin through the introduction of powerdomains. But for stable domain theory the information order of all but the simplest powerdomain fail to be temporal.

Concurrency/interaction: The intricacy of models for distributed computation such as Petri nets and event structures (modelling both processes and types), means that they don't fit comfortably within a partial order of information. Rather their intricacy suggests that they belong more rightfully to an extended notion of domain as a category. Only rarely do the wanted equivalences on processes arise from traditional domain theory.

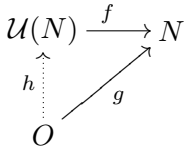
Probability and *nondeterminism:* There are powerdomains both for nondeterminism and probability. Sometimes one needs both. Combining probability and nondeterminism is problematic because the two forms of powerdomain together do not satisfy a distributive law (their combination forces extra laws to be imposed). However, if one works with the intensional *indexed probabilistic powerdomain* where the probability distribution is carried by the *ways* in which values are computed, one recovers a distributive law.

Nondeterministic dataflow: While, as Kahn was early to show, deterministic dataflow is a shining application of simple domain theory, nondeterministic dataflow is beyond its scope. The compositional semantics of nondeterministic dataflow needs a form of generalized relation which specifies the *ways* input-output pairs are realized.

Traditional denotational semantics and domain theory appear to have abstracted away from operational concerns too early. The problems point towards a more intensional 'domain theory' which expresses the *set of ways* of computing. Causal models will reappear in providing an operational reading of the ways computations are realized, for example as the finite configurations of an event structure.

1.3 An abundance of models

Partly in reaction to the difficulties of traditional denotational semantics and domain theory, today we find a range of ways to model a process in computer science. Fortunately many models can be formally related by adjunctions whose adjoints give translations of one model into another. The adjunctions make explicit how to translate from one kind of model (say Petri nets) to another (say occurrence nets, or event structures); this relies on regarding a kind of model (say Petri nets) as a category (a category of Petri nets) with suitable 'simulation' maps (we define the maps on event structures below). For example the unfolding of a Petri net illustrated in the Introduction is the right adjoint to the inclusion functor from the category of occurrence nets to the category of (1-safe) Petri nets. There is an intuitively obvious map $f : \mathcal{U}(N) \to N$ from the occurrence net $\mathcal{U}(N)$ back to the original net N; it takes occurrences of conditions and events back to those conditions and events in the original net of which they are occurrences. A way to express the adjunction is through the following *universal characterization* of the unfolding. Given any map $g : O \to N$ from an occurrence net O to the original net, there is a unique map $h : O \to \mathcal{U}(N)$ such that $f \circ h = g$:



As a right adjoint, unfolding automatically preserves limits, for example products and pullbacks, in the category, and this can be useful in relating parallel compositions in one model to those in another (and in network diagnostics [19]). There is a further adjunction between event structures and occurrence nets; its right adjoint 'strips' away the conditions of an occurrence net to reveal its underlying event structure, while its left adjoint 'saturates' an event structure with conditions to make an occurrence net. Adjunctions compose so we obtain an adjunction between event structures and 1-safe nets—its right adjoint acts as the operation from Petri nets to event structures sketched earlier.

The use of categories exposes a uniformity across different models. Semantics of synchronising processes, whether they be in transition systems, Petri nets, event structures or many other models, are given in precisely the same way in terms of the categorical constructions used. Presented as categories, models support a general, diagrammatic definition of an important equivalence, *strong bisimulation* and its extension beyond transition systems, via open maps.

Despite the abundance of categories of pre-existing models, they form a rather patchy landscape and are, for example, insufficient to represent *higher-order* pro-

cesses (which might take a process itself as input and deliver another process as output). Presheaf categories fill out the landscape of models to provide a versatile range of models for processes [11]. The idea is to build models for processes directly out of computation paths, regarding a nondeterministic process as a presheaf on a category of paths. Essentially, a presheaf is a glueing together of computation paths, the computation paths corresponding to the ways a path can be realiized. Presheaf categories are as versatile as the notion of computation path. With suitable choices of paths, presheaf categories subsume existing models such as event structures, while supporting a range of type constructions, also for higher-order processes and name generation [60, 11, 40, 10]. Presheaf categories and the relations between them, expressed as profunctors, connect with the rich world of higher-dimensional algebra. In particular, the representation of processes as 'bundles' is crucial in the general treatment of weak bisimulation, and its extension to causal models [20]. There are fundamental results such as that application of profunctors to presheaves preserves open-map bisimulation [11]. But the mathematical advantages come at a cost, that of finding an operational reading.

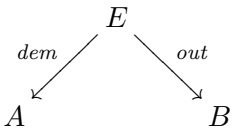
b. Methodology

2 Relations rediscovered

Whereas closed process terms denote presheaves, process terms with free variables denote profunctors. Profunctors are relations between categories in which the category of sets takes over the role of truth values; instead of simply saying whether or not an object of input is related to an object of output, a profunctor provides a set of ways in which that particular input-output instance is realized. Profunctors have arisen independently in a range of areas: in logic and types, e.g. Girard's normal functors and 'container types;'combinatorics through Joyal's theory of species and its extensions [21]; nondeterministic dataflow [44]; higher-order programming languages and processes [11]; categories of models for concurrent computation [11]; as a starting point for the theory of operads.

The surprise is that 'relations' arising from computation can often be represented, in a more computationally informative way, in terms of event structures, with event structures playing both the role of input and output *types*, as well as the *process* of computation between them. A compelling example comes from solutions to the problem of giving a compositional semantics to nondeterministic dataflow.

The many solutions to giving a compositional semantics to nondeterministic dataflow all hinge on some form of generalized relation, to distinguish the different ways in which output is produced from input. A compositional semantics can be given using *stable spans* of event structures, an extension of Berry's stable functions to include nondeterminism [46, 47]. A process of nondeterministic dataflow, with input type given by an event structure A and output by an event structure B, is captured by a pair of maps (a span)



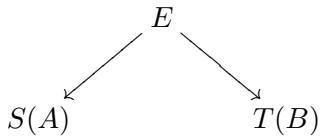
where E is also an event structure. The map $out : E \rightarrow B$ is a *rigid* map, a total function on events that preserves causal dependency and sends configurations to configurations in a locally injective way. The map $dem : E \rightarrow A$, associated to input, is of a different character. It is a *demand* map, *i.e.* a function from $\mathcal{C}(E)$ to $\mathcal{C}(A)$ which preserves finite configurations and unions. The occurrence of an event e in E demands minimum input $dem\,[e]$ and is observed as the output event $out(e)$. *Deterministic* stable spans, where consistent demands in A lead to consistent behaviour in E, correspond to Berry's stable functions.

In fact, stable spans were first discovered as a way to represent, and give operational meaning to, the profunctors that arose as denotations of (open) terms in affine-HOPLA, an affine Higher Order Process LAnguage [40, 41, 42]. The spans helped explain the tensor of affine-HOPLA as the parallel juxtaposition of event structures and a form of entanglement which appeared there as patterns of consistency and inconsistency on events. The use of stable spans in nondeterministic dataflow came later as a representation of the profunctors used in an earlier semantics [44, 46].

This is the start of a key idea, processes as spans. But stable spans are insufficient in various ways; for example, because output maps of stable spans are rigid, stable spans are too restrictive to support a broad range of parallel compositions without resorting to interleaving. What is required is a systematic method to vary the nature of the maps in and out. A systematic way to modify maps is through Kleisli maps associated with monads [37]. Starting from an original category—a good choice would be the category of event structures with rigid maps—one could hope to obtain other kinds maps from an object E to an object B as original maps from E to $T(B)$, where T is an

appropriate monad. This suggests that stable spans be generalized to general spans of event structures

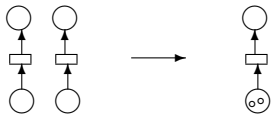
General span



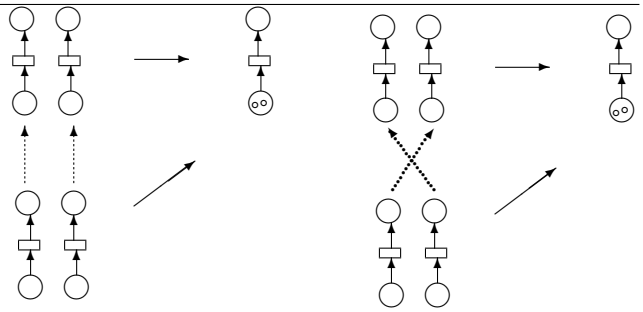
with input the event structure A, output B and process E, w.r.t. suitable monads S and T to moderate the regimes of input and output. More should hold for the spans to compose in a relation-like way—the monads should preserve pullbacks and satisfy a distributive law [61].

In the span shown above, one would hope in particular for a monad S such that demand maps from E to A are realized as Kleisli maps from E to $S(A)$. It becomes important that event structures are able to support a reasonable repertoire of monads. It is here we run into difficulties. For example the useful operation of replication $!E$, forming the parallel composition of countably many copies of an event structure E, does not carry the structure of a monad because the monad laws fail to hold on the nose. But in $!E$ one copy of E is similar to another. Up to this symmetry, allowing one copy to swap with another, the monad laws do hold. In answer to the first and second anomalies at least, a formal treatment of symmetry is needed.

It is illustrative to consider how symmetry is also important in the third anomaly, that of obtaining a universal characterisation of unfoldings of general Petri nets. In general a condition of a Petri net may not just hold or not hold, but hold with a certain multiplicity. For example, below, the net on the right has an initial marking in which a condition holds with multiplicity two. It is generally agreed that its unfolding should be the occurrence net on the left, the two components of which correspond to the two ways in which the conditions and the events of the original general net can occur. There is a folding map taking the occurrences back to the conditions and events of the original net of which they are occurrences:



Earlier, we saw a universal characterization of the unfolding of (1-safe) Petri nets without multiply-holding conditions. An analogous result here would require that any map from an occurrence net to the original net factored *uniquely* through the folding map. But this does not hold of the folding map itself, where both the identity and the map 'swapping' the components in

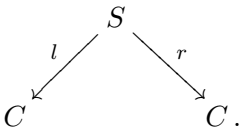


the unfolding provide a factorization—see the figure above. However the two maps, identity and 'swap,' are equal up to the symmetry implicit in the unfolding. The two components of the unfolding inherit their symmetry from the essentially symmetric marking of the initial condition in the original net.

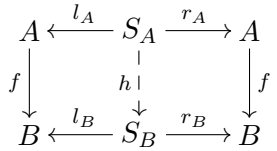
The two examples, replication and unfolding, are alike and perhaps deceptively simple; symmetry in the behaviour of a process can be much less structurally apparent than in these examples. Still, they suggest that the extra structure of symmetry should express when a computation run, or path, can be swapped with another in the behaviour of a process. The method of adjoining symmetry should be tuned to process behaviour, and applicable to a wide range of models.

2.1 Symmetry

The treatment of symmetry on models makes use of a general method of open maps in defining bisimulation in a variety of models. Briefly, symmetry in an object C (be it an event structure, Petri net or transition system or some other model) is expressed as a bisimulation pseudo equivalence, a span of open maps

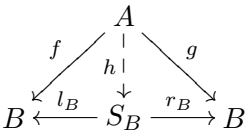


that expresses when paths in C are similar according to the symmetry; its being a bisimulation ensures that similar paths will have similar pasts and futures. That it forms a pseudo equivalence ensures that similarity is reflexive, symmetric and transitive. Maps $f : (A; S_A) \to (B; S_B)$ between objects with symmetry must preserve symmetry in the sense that



commutes for some map h. Two maps $f, g : (A; S_A) \to (B; S_B)$ between objects with symme-

try are regarded as equivalent *up to symmetry*, written $f \sim g$, if



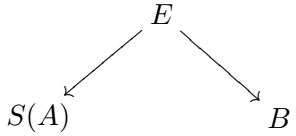
commutes for some map h. The equivalence relation $\sim$ plays a central role. It allows the relaxation of concepts normally defined using equality on maps to analogous concepts *up to symmetry*.

For particular models we can often take advantage of more concrete characterizations of symmetry equivalence; for example, a symmetry equivalence on an event structure corresponds to a certain kind of family of isomorphisms between its finite configurations.

2.2 Consequences of symmetry

A major consequence is that many wished-for monads on event structures, such as replication, do indeed become monads up to symmetry (technically forms of pseudo monad) once event structures are extended with symmetry. Starting with the category of event structures with rigid maps, its extension with symmetry has monads with which to realize demand maps (including the variant in [1]), total non-rigid maps, partial maps and replication [61]. The monads adjust the notion of event, including atomicity, so that events can now have duration. For example, the monad for demand maps creates events in the form of 'input histories,' themselves special demand maps, describing the way that input is explored—for details see [61].

This opens the way to general spans providing semantics to potentially rich process languages and event types, supporting case analysis on events. This work is incomplete, but see [62] for an example of how such a higher-order language can induce the usual event-structure semantics for CCS, as well as an event-structure semantics for a higher-order variant of CCS—here CCS parallel composition appears as a higher-order process taking pairs of event structures with symmetry to their parallel composition. In particular, the stable spans used in the semantics of nondeterministic dataflow can be realized as particular general spans, with only one monad to modify the input map, a case studied by Burroni. Such spans comprise a pair of rigid maps between event structures with symmetry



where the monad S makes events of $S(A)$ input histories, in such a way that rigid maps from E to $S(A)$ correspond to demand maps.

By adjoining symmetry to Petri nets we can obtain a universal characterisation of unfoldings of *general* Petri nets, similar to that for 1-safe nets, but where instead of uniqueness we achieve uniqueness up to symmetry [29].

By extending event structures with symmetry we are able to represent a broader category of presheaves [59, 62, 50]. Operations that previously only worked on presheaves now work on event structures with symmetry. We can now obtain a universal characterisation of an unfolding of higher-dimensional automata as an event structure with symmetry [62]. Now, in principle, the hiding operation on event structures with symmetry yields an event structure with symmetry, bringing a central operation of weak bisimulation back into causal models. Work on a domain theory [52] within the theory of nominal sets [45], was designed with the idea of extending it to presheaf models. Viewing event structures as presheaves (now in nominal sets) would yield constructs such as new-name abstraction on event structures, the missing key to an algebraic treatment of name generation in causal models.

In summary, the introduction of symmetry is bringing a new expressive power to causal models: previously unthought-of semantics to higher-order types and languages; mechanisms for event abstraction, allowing the switch from atomic to compound events; extensions to the usual Petri-net unfolding and its preservation results; new constructions, equivalences and new-name abstraction; and possibilities of new enrichments with further structure through the presentation of generalized relations as spans of causal models. It exposes the path forward to the next generation of semantics.

3 Research plan

Objective 1. Intensional semantics

The core objective is a comprehensive intensional semantics in which it is possible to read the operational semantics from the denotational semantics. A key vehicle will be high-level syntax for general spans and associated types.

1.1 Intensional domain theory A start point is the theory of general spans of event structures with symmetry. Their form is determined by input and output monads, subject to laws to ensure the spans compose.

It is important to understand the algebraic nature of general spans: characterize the profunctors they induce; discover the type constructions and operations they lead to. The relation with profunctors is important technically, but also because profunctors support a rich calculus (based on ends and coends) which can then be imported into the syntax of process languages. Different choices of input and output monads determine different process languages. A particular choice leads to, for example, stable spans and affine HOPLA, and, with a variation on the input monad, to HOPLA, where input can be replicated. Other choices are leading to completely new process languages, some with their own challenges of syntax and operational semantics—examples are described in [61]. The mathematics suggests we look for an encompassing metalanguage accommodating both an input and output monad, with a suitable distributive law, on spans of rigid maps. One guideline is that the metalanguage should subsume Moggi's metalanguage, where in effect there is only a nontrivial output monad. *Goals:*

1. *characterizations of profunctors represented by general spans;*

2. *type constructions and operations of general spans, the corresponding process languages;*

3. *a metalanguage and its equational theory.*

1.2 Algebra of operational methods: An aim in moving to a more intensional domain theory and denotational semantics is so that the denotational semantics can prescribe an operational reading. As a bridge over to operational methods we will lift to general spans earlier results [40, 43, 42] on the strong correspondence between derivations in an operational semantics and elements of denotations as generalized relations. For this there are more robust alternatives to earlier well-foundedness proofs [40, 43, 42], based instead on generalized logical relations. An operational semantics for the fundamental higher-order process metalanguage introduced in [60], inspired by profunctors and their input-output duality, would yield a solution to the open problem of giving a traditional coinductive definition of open-map bisimulation for higher-order processes. The models will lead to new techniques to show key equivalences are congruences, extending the result that any operation expressed as a profunctor automatically preserves open-map bisimulation [11]. *Goals:*

1. *results and techniques for strong correspondence for the languages of 1.1, in particular generalizing logical relations;*

2. *an operational semantics (via general spans) for the profunctor language of [60].*

These should lead to:

3. *an operational account of open-map bisimulation for higher-order processes;*

4. *operational (coinductive) characterizations of weak bisimulation on event structures and other key equivalences;*

5. *techniques to establish congruences.*

1.3 Game semantics: Game semantics [3, 30] provides an intensional semantics of sequential programs. To import its lessons we can exploit the PI's recent discovery that basic games (and so probably all game semantics) fit within spans of event structures. This puts game semantics within a new space of possibilities, where nondeterministic and concurrent games are naturally situated. The more general setting should expose some 'artificialities' in standard formulations of games and their properties, traditionally based on *sequences* of moves. For instance, subtle notions such as 'views' of innocent strategies can be explained more directly through causal dependencies. There are already promising connections: the combinatorially-defined schedules of Harmer, Hyland and Melliès, important for abstract machines, now appear automatically as prime configurations of an event structure denoting the strategy as a span. *Goals:*

1. *present the games of [3] in terms of general spans of event structure—this will make essential use of the replication monad on event structures with symmetry;*

2. *present the games of [30] in terms of general spans of event structure—this can exploit the result [27] expressing the games of [30] in terms of a 2-sided Kleisli construction on basic games;*

3. *nondeterministic and concurrent games in the setting of general spans;*

4. *extensions to algorithmic game theory.*

1.4 Names: We will investigate the mathematics of name generation, in particular to support the need for fresh-name assumptions and new-name abstraction within causal models. There are new mechanisms for name generation supported by event structures with symmetry. For example, a datatype of names can be represented by a countable discrete event structure with symmetry; its events would correspond to names and its symmetry allow the permutation of names. The intrinsic event-linearity of maps of event structures would ensure the initial privacy of names. There are projects in nominal sets [45] designed to import name generation into profunctors and event structures—the former will proceed by analogy with nominal domain theory [52]. It is an intriguing research issue to decide whether the effect of using nominal sets can be real-

ized through event structures with symmetry. *Goals:*
1. *develop profunctors in nominal sets, the ensuing semantics of nominal/new HOPLA [52];*
2. *develop event structures in nominal sets;*
3. *mechanisms for name generation directly from event structures with symmetry;*
4. *applications to event-structure semantics of the pi-calulus and SPL [12, 13].*

1.5 Higher-dimensional algebra: The connection with higher-dimensional algebra is woven into the development of the new domain theory as several threads (presheaves, profunctors, pseudo monads, symmetry and the enriched categories it leads to) and has already been fruitful [11, 21]. It is likely to lead to the refinement and development of causal models, as we seek to give computational interpretation to mathematical constructions. We need to understand better the relationship between the two kinds of generalized relations, spans of event structures and (certain kinds of) profunctors. We begin to see methods to side-step the sometimes difficult verification of distributive laws between (pseudo) monads, an example where the calculational demands of CS can lead to new mathematical techniques (as has happened earlier [9]). Methods for establishing the strong correspondence of semantics using generalized relations and operational semantics are taking us tohigher-dmensional logical relations.

Objective 2. Event-based reasoning

The addition of symmetry to causal models paves the way to new equivalences and methods of reasoning about processes.

2.1 Equivalences and equational theories: Equational theories of equivalences will play an essential role in reasoning, in counterbalancing the inbuilt intensionality of the models. Through the greater expressivity of event structures with symmetry [50], weak bisimulation can now in principle be developed purely within event structures with symmetry—doing this is a primary task. In fact the results of [20] extend beyond weak bisimulation: they show how to obtain weakenings of biimulation on presheaves w.r.t. a wide range of observations. Now presheaves are representable by event structures with symmetry, we can define weak bisimulations on event structures with symmetry w.r.t. the observations specified by an event structure with symmetry. This could have enormous potential, giving us a mathematical powerful and intuitive tool for all manner of equivalences. More distantly, the mathematical connections move us closer to topology and geometry and the potential use of their equivalences' use in reasoning about processes [26, 34]. *Goals:*
1. *weak bisimulation directly on event structures with symmetry;*
2. *observation-based equivalences on event structures with symmetry; their equational theories and scope.*

2.2 Event types: Event types express not just the type of events a process can perform but also constraints of causality, linearity and atomicity. They suggest specification logics by analogy with existing logic for domains and how the verification of such properties might be reduced to type-checking. Present examples show event types support a basic proof and definitional method of event induction. We are beginning to see similar causal logics, designed specifically for security protocols, to support reasoning along causal dependency in the manner of strand spaces [15]. *Goals:*
1. *a type-directed syntax for events, their causal relations and logic (by analogy with 'logic of domains');*
2. *formal event-induction for types (via the syntax for events);*
3. *reasoning via type checking;*
4. *express causal logics for security protocols.*

2.3 Program logics: Through a causal semantics of concurrent separation logic (the first was developed by Hayman and the PI), we are in a strong position to settle an old conjecture of John Reynolds on robustness of the logic under command-refinement. *Goal:*
1. *tackle Reynolds' conjecture, possibly revising the causal semantics of concurrent separation logic [28].*

Objective 3. Quantitative semantics

The use of spans provides, relatively unexplored, methods to represent types and computation with probabilistic, stochastic and even quantum behaviour. Although strictly not a part of quantitative semantics the development of mechanisms for name generation will provide mathematical guidelines: name generation has features in common with random variables.

3.1 Probability: Spans can be enriched with probability, essentially by taking the vertex in a span of event structures to be a probabilistic event structure [55, 2]. The probabilistic event structure expresses both the ways, and with which probability, output is obtained. The output event structure (to the right of the span) would now stand for a type of probabilistic processes. In special cases the idea relates to *categorical* versions of the indexed-probability powerdomains [53, 54]. The development will be guided by uses of probabilistic event structures in security, Bayesian models

of trust and network diagnostics, and by the appearance of stochastic event structures in systems biology. *Goals:*
1. *basic properties of probabilistic and stochastic event structures* with symmetry*;*
2. *probabilistic spans of event structures and their application in simple probabilistic languages;*
3. *spans for probability* and *nondeterminism;*
In the longer term we are keen to:
4. *relate languages with name generation and to their analogues based on probabilistic generation.*

3.2 Quantum event structures: It is planned to investigate a tentative definition of 'quantum event structure,' comprising an event structure where the events (regarded as occurrences of observations) are labelled by projectors in a Hilbert space, with concurrent events labelled by commuting projectors. Each finite configuration is associated with an operator, that got by composing any sequence of projectors from which the configuration arises. In the manner of consistent histories [23], we can investigate 'decoherence conditions:' those sets of configurations over which the operator weights (got via the trace inner product) determine a probability distribution. *Goal:*
1. *obtain decoherence conditions on quantum event structures, exhibit their relation to probabilistic event structures;*
2. *semantics of a quantum-process languages [48].*

Objective 4. Application methods

The introduction of symmetry to event structures opens up a new landscape of models in which event structures both stand for types and the process of computation between them. Event types can express local causal constraints and symmetries, and provide semantics to name generation. This calls out for experiments, sometimes on a small scale, in the semantics and analysis of distributed and parallel algorithms; in developing new methods there is an art in choosing experiments at the right scale, to reveal a verification technique or highlight a missing key feature in the theory.

4.1 Petri-net SOS: It is planned to design and promote an accessible structural operational semantics based on Petri nets (with symmetry), to update the traditional and highly-influential techniques of structural operational semantics (SOS) based on transition systems. The idea is to replace the rule-based inductive definition of configurations and transitions in SOS by rule-based definitions of conditions and events. *Goal:*
1. *a manual illustrating Petri-net SOS on a range of applications.*

4.2 Distributed and parallel algorithms: The introduction of symmetry to event structures opens up a new landscape of models in which event structures both stand for types and the process of computation between them. Event types can express local causal constraints and symmetries, and provide semantics to name generation. This calls out for experiments in the semantics and analysis of distributed and parallel algorithms.

There are many instances of reasoning about distributed algorithms through symmetry and chains of dependencies. Sometimes the nature of the model can obscure the analysis. CSP was used to show that algorithms deployed on the International Space Station were resilient against Byzantine failures [7, 8]. The proofs relied critically on implicit causal relationships between the processes. It is planned to show how the existing proofs in these papers can be claried through direct reasoning about causality.

One rich area for ECSYM is that of security protocols, where the PI has experience. In protocol design and analysis considerable skill is used in specifying the identity of events often through the generation of random or fresh names, which play an essential role in establishing causal dependencies. This is manifest in the verification methods of strand spaces [51, 13]. There are clear gaps to explore in the treatment and exploitation of symmetry, and in relations with truly-cryptographic protocols. The latter challenges present-day techniques, and attempts will feed back into the development of probabilistic semantics. Two other specialized areas which rely on causal models, and probabilistic event structures [55], are the distributed diagnosis of communication networks [5], and the very recent Bayesian analysis of event-based trust [38]—there are good working relations with both research teams. We are beginning to see convincing logics, designed specifically for security protocols, to support reasoning through causal dependency [15]. Expressing these will test out the semantics the and event-based reasoning it supports.

Another ripe and important testing ground—where there is considerable local expertise—is the analysis of multicore memory, recent verification of which uses a form of event structure [49]. A positive solution to Reynolds' conjecture would, for instance, have an immediate impact here as conditions of race-freedom play a key role in the verification of multicore memory [6]. *From the start we will carry out experiments in the expressivity of the semantics in:*
1. *security protocols, initially for the basic language*

SPL [13];
2. *event structure models of multicore memory (with Sewell).*
In the longer term:
3. *verify the Byzantine protocols of [7, 8].*

4.3 Unfoldings and tools: Unfoldings of Petri nets have provided strikingly successful methods for the analysis of distributed systems [18], and with some variation in systems biology [16]. The characterization of the unfolding as a right adjoint has been exploited in network diagnosis [19]. The techniques here, based on symmetry, extend unfoldings and their characterization to general nets, and begin to push Petri nets into the new territory of higher-order processes (although we have concentrated on event structures, similar ideas are working for nets). Symmetry is already exploited in model checking, and we will investigate its potential role in net-unfolding techniques (consulting Esparza).
Goal: **1.** *extend unfolding tools with symmetry.*

4.4 Systems biology: Systems biology provides new challenges to semantics, puts into action ideas from causal models, symmetry, as well as stochastic and probabilistic event structures. Biochemical reactions are by nature determined in a local fashion and so fit concepts from causal models. By invoking ideas of independence and conflict from causal models stochastic simulations can to an important degree rely on an accumulation of local updates to control the state explosion, as is done in the kappa-system, developed in Harvard, Edinburgh and Paris. Symmetry plays several important roles: in determining the stochastic rates of rules; in reductions via abstract interpretation; and in the analysis of biochemical pathways. Once account is taken of independence, rule-based simulations generate (stochastic) event structures, with symmetry induced by the similarity of molecules of the same species. One might expect that the event structures described the biochemical pathways recognised by biologists. But the requirements of biologists drove kappa to a much more compressed account of the pathways. On a recent visit to Harvard the PI provided a mathematical rationale for the 'compression algorithms' of kappa. The solution introduces maps on site-graphs to express local state, evolution and symmetry of biochemical systems. We need to push this mathematical analysis into stochastic simulation and further state-reduction methods exploiting symmetry. The experience here will test the developing semantics and its enrichment with probability and time.
Goals (with the kappa team):
1. *correctness proofs of the compression algorithms of kappa;*
2. *extension of site-graphs and maps to membranes, to locatize reactions;*
3. *semantics in terms of stochastic event structures;*
4. *state reduction methods using symmetry.*

c. Resources

As Principal Investigator **Glynn Winskel** [concerned with all areas of the project] would head a world-class team comprising the following researchers [working on the indicated areas of the project].
Marcelo Fiore [Areas 1, 2.1, 2.2, 3] has been Reader in Mathematical Foundations of Computer Science in the University of Cambridge since 2005, having been a University Lecturer in the Computer Laboratory from October 2000. From 2000 to 2005 he held an EPSRC Advanced Research Fellowship. Previously, he was a University Lecturer at the University of Sussex. Most relevant to ECSYM is his research on: axiomatic domain theory, leading to the higher-dimensional model of complete cuboidal sets (with Plotkin and Power); the construction of the first fully-abstract denotational model of the pi-calculus (with Moggi and Sangiorgi); the investigation of fibred models of processes (with Bunge), including the first conceptual treatment of weak bisimulation (with Cattani and Winskel); the development of a model theory (with Turi) leading to the first congruence rule format for name-passing mobile systems (with Staton); the investigation of generalised combinatorial models of linear logic (with Gambino, Hyland and Winskel), leading to axiomatisations of creation and annihilation operators in bosonid Fock space and of differential structure. Fiore has recently given several invited addresses on his work.
Martin Hyland [Areas 1.1, 1.3, 1.4, 1.5, 2.1, 3], is Professor of Mathematical Logic and Head of the Department of Pure Mathematics and Mathematical Statistics at Cambridge. He has wide experience of the application of abstract mathematical ideas and methods to theoretical computer science and mathematical logic. He is famous for pioneering work on the lambda calculus, models for constructive logics, synthetic domain theory and the effective topos, and game semantics. With regard to ECSYM, he has particular experience in domain theory, in game semantics and in higher dimensional algebra and category theory, and general experience of a wide range of pure mathematics. He has been closely involved in recent higher dimensional (intensional) versions of domain theory. He was the originator with Luke Ong of the innocent strategy approach in game semantics and has recently

been studying concurrent versions of that. With his student, Nathan Bowler, he is working on the algebra of games. He has a deep knowledge of homotopy-theoretic aspects of higher dimensional algebra, and with his recent student, Richard Garner, has been involved in work particularly relevant to models based on symmetry.

Andrew Pitts [Areas 1.1, 1.2, 1.4, 2] has been Professor of Theoretical Computer Science in the University of Cambridge since 2001, having previously been a Lecturer and Reader in the University's Computer Laboratory since 1989. He has received research funding from the Royal Society, the ESPRIT, SCIENCE, HCM, and TMR programmes of the European Union, SERC, and EPSRC. His work has ranged over category theory, logic, type theory and programming language semantics. His contributions to the semantics of naming and localising resources are particularly relevant to this proposal. The Pitts-Stark nu-calculus identified subtle interactions between ML-style higher order functional programming and dynamically allocated names. The operational aspect of this work has led to useful methods for reasoning about local state in programming languages; the denotational aspect of the work, involving the use of category-theoretic sheaf and presheaf models, has been applied to modelling the pi-calculus and influenced recent work on metamathematics of syntax involving binding operations. With his PhD student M.J. Gabbay he introduced the model of fresh names and name-binding based on name-permutations ("nominal sets") that has since proved very influential in dealing with issues associated with binding in semantics, metaprogramming and automated theorem proving. Symmetry lies at the heart of this novel approach to modelling anonymity of named entities and is of direct relevance to this proposal.

Junior researchers:

Richard Garner [Areas 1.1, 1.2, 1.4, 1.5, 2.1, 2.2] is a Research Fellow at St John's College, Cambridge, and has recently held a two-year European Union Marie Curie Intra-European Fellowship at Uppsala University, Sweden. His PhD dissertation at Cambridge under Martin Hyland studied polycategories—which are of importance to the linear logic view of concurrency [14, 22]—using the higher-dimensional algebra of profunctors, extending the work of [21]. His subsequent research experience has concentrated on three areas: higher-dimensional category theory, abstract homotopy theory and the dependent type theory of Martin-Löf. Of particular relevance to this proposal are his investigations into *weak factorisation systems*, which are the central abstract concept underlying the open map notion of bisimulation [31], and his contributions to a categorical understanding of the *identity types* of Martin-Löf type theory; these share many formal similarities with the notion of *symmetry* discussed above. He has also worked on infinite-dimensional category theory of the kind which features prominently in higher-order rewriting [34].

Jonathan Hayman [Areas 2, 4] is an expert in the use of causal models in defining the semantics of programming languages and their associated logics [28, 29]. His recent PhD thesis, supervised by Glynn Winskel, studied the use of Petri nets in defining the semantics of programming languages, suggesting a general framework for defining the structural Petri net semantics of programming languages analogously to the way that Plotkin's structural operational semantics defines the semantics of terms using transition systems. The net semantics obtained was used to capture properties on the independence of actions arising from concurrent separation logic. His thesis also studied how symmetry may be adjoined to Petri nets and that this has a fundamental role in obtaining the important unfolding operation on general forms of Petri net. He is presently working as a software consultant. In the future, he hopes to extend the use of causal models in the analysis of concurrent systems, for example by studying their use as efficient models to found techniques such as abstract interpretation.

Chung-Kil Hur [Areas 1.1, 1.2, 1.4, 2, 4] is presently a visiting researcher at the laboratory PPS, University of Paris 7. He was awarded his PhD from the Computer Laboratory of the University of Cambridge in 2010. He previously obtained Bachelor's Degrees in both Computer Science and Mathematics from the Korean Advanced Institute of Science and Technology (KAIST). The excellence of his grades there led to the funding of his PhD studies at Cambridge by the Samsung Scholarship Foundation. During his PhD he developed a categorical framework for defining equational theories and devised a general sound equational logic for the theories of the framework. Examples include the (enriched) algebraic theories of Kelly and Power and the nominal equational theories of Clouston and Pitts and, independently, Gabbay and Mathijssen. He recently worked with Nick Benton at Microsoft Research to develop a proof technique for showing observational equivalence of low-level machine code programs, based on a novel combination of biorthogonality, step-indexed logical relations and domain theory. At PPS he has become skilled in using the COQ theorem prover.

Sam Staton [Areas 1, 2.1, 2.2, 3.1, 4.2.1] has held the position of Research Associate at the University of Cambridge Computer Laboratory since July 2007. He currently holds a EPSRC Postdoctoral Research Fellowship entitled "Mathematical operational semantics for data-passing processes". Sam's main research interests centre around mathematical frameworks as foundations for the semantics of programming languages. Sam was awarded his PhD from the University of Cambridge Computer Laboratory in June 2007. His PhD thesis work was on the semantics of name-passing process languages, for which the pi-calculus is a simple example. The central result of his thesis is a syntactic criterion (a 'rule format') for specifications of well-behaved name-passing languages. Sam has used his thesis work as a springboard for further research in a number of different areas, including algebraic theories for local state. His work on bialgebraic semantics helps to connect progress in operational and denotational semantics. He is currently interested in languages for cryptographic protocols, and in foundations for the semantics of higher-order and functional programming languages.

Costs: See the table on Page 15. It is important to point out that this project would be impossible with the level of funding available in the UK. Salary is requested: to employ the four junior researchers (full-time, 4 years each—they all lack funding at present); for the Principal Investigator for 8 months each year over 5 years (the PI plans to concentrate on ECSYM while continuing some teaching to attract new PhD students—the reduction also acknowledges that he will still have some Lab administration); for Fiore and Pitts to work full-time on the project for 6 months each—Hyland is Head of Cambridge Maths so while intending to be active on ECSYM throughout its duration recognizes that continuous leave of 6 months is unrealistic in his case. It is important to note that there will be several PhD students, colleagues and visitors who will contribute to ECSYM without its funding (*e.g.* the PI's EPSRC-funded PhD student Chris Thompson-Walsh has begun work on system biology, Areas 4.3,4.4, his Gates-funded PhD student Steffen Loesch will work on nominal semantics for name generation 1.4, his intern Silvain Rideau from ENS, Paris, will work on expressing game semantics as spans of event structures 1.3, as will visitor P-L. Curien, and colleague Peter Sewell is an expert in Area 4.2.6). Funding is also requested for conferences and research visits to: Bath (McCusker, Power); Berkeley (Dana Scott); Birmingham (Ghica,Levy); Bologna (Asperti, Gorrieri, Sangiorgi); Boston (Fontana); Braunschweig (Koslowski); Copenhagen (Birkedal, Hildebrandt); Cornell (Kozen); Darmstadt (Streicher); Dublin (Hennessy); Edinburgh (Bradfield, Danos, Hillston, Plotkin, Stirling); London (Robinson, O'Hearn); Marseilles (Girard, Regnier, Santocanale); Montreal (Joyal, Panangaden, Selinger); Mitre (Guttman); Munich (Esparza); Ottawa (Scott); Oxford (Abramsky,Coecke, Ong); Paris (Abbes, Curien, Feret, Krivine, Mellies, Varacca, Zappa-Nardelli); Pisa (Montanari, Degano, Gadducci); Pittsburgh (Brookes, Griffiths); Rennes (Benveniste, Fabre, Haar, Jard); Sophia Antipolis (Boudol, Castellani); Southampton (Sassone,Sobocinski); Stanford (Mitchell, Pratt) Strathclyde (Ghani); Sussex (Laird); Swansea(Roggenbach) Sydney (van Glabbeek, Johnson, Street, Verity); Aarhus (Nielsen). We can expect some costs to be covered by external institutions (*e.g.* the PI has been invited to visit Fontana's Lab at the Harvard Medical School for 6 weeks). Additional funds are requested for some computer support in the form of laptops for the researchers employed on ECSYM and the cost of broadband to the PI's home (25 Euro per month, without VAT).

References

[1] Abbes, S., A Cartesian closed category of event structures with quotients. DMTCS, 2006.

[2] Abbes, S., and Benveniste, A., Probabilistic models for true-concurrencyInf. and Comp. 204(2), 2006.

[3] Abramsky, S., Jagadeesan, R., and Malacaria, P., Inf. and Comp. 163, 2000.

[4] Benson, D.B., Counting Paths: Nondeterminism as Linear Algebra. IEEE Trans. Software Eng. 10(6), 1984.

[5] Aghasaryan, A., Fabre, E., Benveniste, A., Boubour, R., and Jard, C., Fault detection and diagnosis in distributed systems Discrete event dynamic systems, theory and applications, 8(2), 1998.

[6] Boudol, G., and Petri, G., A theory of speculative computation. ESOP 2010.

[7] Buth, B., Kouvaras, M., Peleska, J., and Shi, H., Deadlock analysis for a fault-tolerant system. AMAST1997.

[8] Buth, B., Peleska, J., and Shi, H., Combining methods for the live-lock analysis of a fault-tolerant system. AMAST1999.

[9] Caccamo, M., and Winskel, G., Limit preservation from naturality. CTCS'04, ENTCS 122, 2005.

[10] Cattani, G.L., Stark, I., and Winskel, G., Presheaf models for for pi-Calculus. Proc. CTCS'97, LNCS 1290, 1997.

[11] Cattani, G.L., and Winskel, G., Profunctors, open maps and bisimulation. MSCS, 2005.

[12] Crafa, S., Varacca, D., and Yoshida, N., Event Structure Semantics for the Internal pi-calculus. CONCUR 2007.

[13] Crazzolara, F., and Winskel, G., Events in security protocols. ACM Conference on Computer and Communications Security, 2001.

[14] Curien, P-L., and Faggian, C., L-Nets, Strategies and Proof-Nets. CSL 2005.

[15] Datta, A., Derrick, A., Mitchell, J.C., and Roy, A., Protocol composition logic (PCL). ENTCS 172, 2007.

[16] Danos, V., Feret, J., Fontana, W., Krivine, J., Scalable Simulation of Cellular Signaling Networks. APLAS 2007.

[17] Doghmi, S.F., Guttman, J.D., and Thayer, F.J., Searching for shapes in cryptographic protocols. TACAS'07, 2007.

[18] Esparza, J., and Heljanko, K., Unfoldings: A Partial-Order Approach to Model Checking. EATCS Monographs in TCS, 2008.

[19] Fabre, E., Bayesian Networks of Dynamic Systems. Habilitation thesis, IRISA Rennes, 2007.

[20] Fiore, M.P, Cattani, G.L., and Winskel, G., Weak Bisimulation and Open Maps. LICS 1999.

[21] Fiore, M., Gambino, N., Hyland, J.M.E., and Winskel, G., The cartesian closed bicategory of generalised species of structures. Journ. of the London Math. Soc., 77 2, 2007.

[22] Girard, J-Y., Linear Logic. Theoretical Computer Science, 50, 1987.

[23] Griffiths, R.B., Consistent Quantum Theory. CUP, 2003.

[24] Glabbeek, R.J. van, The Individual and Collective Token Interpretations of Petri Nets. CONCUR 2005.

[25] Glabbeek, R.J. van, On the expressiveness of higher dimensional automata. EXPRESS 2004,

[26] Gunawardena, J., Homotopy and concurrency. Bulletin EATCS, 1994.

[27] Harmer, R., Hyland, J.M.E., and Melliès, P-A., Categorical combinatorics for innocent strategies. LiCS 2007.

[28] Hayman, J., and Winskel, G., Independence and Concurrent Separation. LICS 2006.

[29] Hayman, J., and Winskel, G., The unfolding of general Petri nets. FSTCS 2008.

[30] Hyland, J.M.E., and Luke Ong, C-H., On Full Abstraction for PCF: I, II, and III. Inf. Comput. 163(2), 2000.

[31] Joyal, A., Nielsen, M., and Winskel, G., Bisimulation from open maps. Inf. and Comp., 127(2), 1996.

[32] Khomenko, V., Koutny, M., and Yakovlev, A., Logic Synthesis for Asynchronous Circuits Based on STG Unfoldings and Incremental SAT, Fundamenta Informaticae 70 (1-2), 2006.

[33] Lamport, L., Time, clocks and the ordering of events in a distributed system. CACM 21, 1978.

[34] Lafont, Y., Algebra and geometry of rewriting. Applied Categorical Structures 15, 2007.

[35] Lynch, N., Distributed Algorithms. Morgan Kaufmann Publishers, San Mateo, CA, 1996.

[36] McMillan, K.L., A technique of state space search based on unfolding. Formal Methods in System Design 6(1), 1995.

[37] Moggi, E., Computational lambda-calculus and monads. LICS 1989.

[38] Nielsen, M., Krukow, K., and Sassone, V., A Bayesian Model for Event-based Trust. ENTCS 172, 2007.

[39] Nielsen, M., Plotkin, G.D., and Winskel, G., Petri nets, event structures and domains. *TCS*, 13(1), 1981.

[40] Nygaard, M., Domain theory for concurrency. PhD Thesis, University of Aarhus, 2003.

[41] Nygaard, M., and Winskel, G., Linearity in Process Languages. In proc. LICS'02, 2002.

[42] Nygaard, M., and Winskel, G., Domain theory for concurrency. *TCS* 316, 2004.

[43] Nygaard,M., Strong correspondence for HOPLA. 2004. Available *http://www.daimi.au.dk/∼nygaard/pub/strongcorrespondence.pdf*

[44] Hildebrandt, T., Panangaden, P., Winskel, G., A relational model of non-deterministic dataflow. MSCS 14(5), 2004.

[45] Pitts, A.M., Nominal Logic, A First Order Theory of Names and Binding. Inf. and Comp. 186, 2003.

[46] Saunders-Evans, L., and Winskel, G., Event structure spans for non-deterministic dataflow. Express'06, 2006.

[47] Saunders-Evans, L., Events with persistence. PhD thesis, University of Cambridge Computer Laboratory, 2007.

[48] Selinger, P., and Valiron, B., On a fully abstract model for a quantum linear functional language. ENTCS 210, 2008.

[49] Sarkar, S., et al. The Semantics of x86-CC Multiprocessor Machine Code (to appear in POPL 2009).

[50] Staton, S., and Winskel, G., On the expressivity of symmetry in event structures. Submitted, January, 2010.

[51] Thayer, J., Herzog, J., and Guttman, J., Strand spaces: Why is a security protocol correct? In IEEE Symposium on Security and Privacy, 1998.

[52] Turner, D., and Winskel, G., Nominal domain theory for concurrency. CSL 2009.

[53] Varacca, D., Two Denotational Models for Probabilistic Computation. PhD University of Aarhus, 2003.

[54] Varacca, D., and Winskel, G., Distributing Probabililty over Nondeterminism. MSCS 16(1), 2006.

[55] Varacca, D., Voelzer, H., and Winskel, G., Probabilistic event structures and domains. TTCS 358(2-3), 2006.

[56] Winskel, G., and Nielsen, M., Models for Concurrency. Handbook of Logic and the Foundations of Computer Science, vol. 4, pages 1-148, OUP, 1995.

[57] Winskel, G., Event structure semantics of CCS and related languages. ICALP 82, LNCS 140, 1982. Extended version available from *http://www.cl.cam.ac.uk/∼gw104*.

[58] Winskel, G., Event structures. LNCS 255, 1987.

[59] Winskel, G., Event structures as presheaves—two representation theorems. CONCUR 1999. LNCS 1664, 1999.

[60] Winskel, G., Relations in concurrency. Invited talk, LICS'05, 2005.

[61] Winskel, G., Event structures with symmetry. ENTCS 172, 2007. See *http://www.cl.cam.ac.uk/∼gw104* for corrections.

[62] Winskel, G., Symmetry and Concurrency. CALCO 2007.