Technical Report

UCAM-CL-TR-710
ISSN 1476-2986

Number 710

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Event structures with persistence

Lucy G. Brace-Evans

February 2008

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500

http://www.cl.cam.ac.uk/

This technical report is based on a dissertation submitted October 2007 by the author for the degree of Doctor of Philosophy to the University of Cambridge, St. John's College.

Technical reports published by the University of Cambridge Computer Laboratory are freely available via the Internet:

http://www.cl.cam.ac.uk/techreports/

ISSN 1476-2986

Abstract

Increasingly, the style of computation is changing. Instead of one machine running a program sequentially, we have systems with many individual agents running in parallel. The need for mathematical models of such computations is therefore ever greater.

There are many models of concurrent computations. Such models can, for example, provide a semantics to process calculi and thereby suggest behavioural equivalences between processes. They are also key to the development of automated tools for reasoning about concurrent systems. In this thesis we explore some applications and generalisations of one particular model – event structures. We describe a variety of kinds of morphism between event structures. Each kind expresses a different sort of behavioural relationship. We demonstrate the way in which event structures can model both processes and types of processes by recalling a semantics for Affine HOPLA, a higher order process language. This is given in terms of asymmetric spans of event structures. We show that such spans support a trace construction. This allows the modelling of feedback and suggests a semantics for non-deterministic dataflow processes in terms of spans. The semantics given is shown to be consistent with Kahn's fixed point construction when we consider spans modelling deterministic processes.

A generalisation of event structures to include *persistent* events is proposed. Based on previously described morphisms between classical event structures, we define several categories of event structures with persistence. We show that, unlike for the corresponding categories of classical event structures, all are isomorphic to Kleisli categories of monads on the most restricted category. Amongst other things, this provides us with a way of understanding the asymmetric spans mentioned previously as symmetric spans where one morphism is modified by a monad. Thus we provide a general setting for future investigations involving event structures.

Acknowledgements

Special thanks go to my supervisor Glynn Winskel for his ideas and advice. I wish to express my gratitude to Sam Staton and Samy Abbes for their help during the write-up period. I would like to acknowledge the great research environment provided by the University of Cambridge Computer Laboratory and to thank all its members for making it what it is. Thanks also go to my examiners Thomas Hildebrandt and Marcelo Fiore for their excellent suggestions. Finally, I wish to thank my friends and family, without whose support this thesis would not have been possible.

Contents

Chapter 1

Introduction

The domain theory and denotational semantics developed by Scott and Strachey ([31]) has long provided a global mathematical setting for sequential computation. Denotational semantics provides us with a way of relating programs to the mathematical functions between domains that they represent. This allows us to compare programs of different languages and structures in a precise way as well as providing inspiration for new languages and type systems. It also allows the well-known mathematical results to be used to prove properties of programs. In fact the possible applications are endless and diverse, ranging from analysis in optimising compilers to proving properties of programs in theorem-provers.

As technology has progressed, the style of computation has changed from one machine running one program which takes an input and terminates to produce output. Real computer systems now run programs that do not terminate but rather interact with their environment continuously. For example, consider a Windows-style operating system. It waits for input from the user and responds accordingly. It does not terminate but runs continuously. Also, parallel-processing and multi-machine systems are becoming increasingly relevant, especially as we begin to hit the limits of the speed at which a single processor can be run. So, functions from inputs to outputs are no longer appropriate to model this kind of behaviour. Rather than sets and functions between them to base a denotational semantics upon, we now require richer structures and constructions upon them that can capture the additional behaviour.

Theories of concurrent computation aim to describe the behaviour of systems of communicating but autonomous processes. This is done with a view to providing a mathematical framework for reasoning about them similar to that provided by domain theory for sequential computations. However, there are many possibilities. Expressivity often comes at the price of simplicity. Often the relationships between the various models are not clear.

In this thesis we describe a particular model for concurrent processes – event structures and their spans. They are easily understandable from an operational point of view and yet expressive enough to describe the behaviour of many kinds of concurrent processes. We investigate how this model could be extended to capture additional behaviours without sacrificing the simplicity that makes it so easy to reason about.

In this chapter we recall some of the possibilities for modelling concurrent computation, making a note of some of the choices available. We discuss some of the issues associated with modelling non-deterministic dataflow and observe that it cannot be modelled adequately with relations or powerdomains. We then introduce event structures. We describe spans of event structures based on two different kinds of morphism, attempting to give an intuitive description of the behavioural relationships between event structures that they express. Some other kinds of morphism are described. We give an example of how the universal constructions in the categories formed from these morphisms can be of operational relevance. Next we discuss the possibility of varying the kinds of morphism from which spans are constructed to allow additional behaviours to be captured. We consider how such alterations might be done systematically. Finally, we give an overview of each chapter of the thesis and its contribution.

1.1 Modelling Concurrent Computation

Models for sequential computations have been studied in great depth and include Turing Machines, the Lambda Calculus and many others. These models have been shown to be equivalent in the sense that their behaviours in terms of functions from inputs to outputs are the same. Concurrent computations exhibit very different behaviours. They may be viewed as a number of spatially separated activities communicating with the environment and each other to accomplish a joint task. Models for concurrent computations need to provide a representation of the patterns of actions the entire system can perform in order to capture aspects such as mutual exclusion, starvation and deadlock.

Having developed such models we can use them as a foundation upon which to build. For example, such models can provide a semantics to process calculi, thereby providing a way to relate them and suggesting suitable behavioural equivalences. They can also be used to formally define specification logics and therefore are key to the development of automated tools for reasoning about concurrent systems.

One of the key behaviours of such processes is non-determinism, i.e., for a particular input there may be several possible outputs. Clearly, continuous functions between domains cannot express this. This suggests replacing continuous functions with relations between domains. Another alternative is moving from domains to Plotkin's *powerdomains* [28]. Roughly, powerdomains allow us to represent relations between domains as set-valued functions where the set contains all outputs that could be produced for a particular input. The Hoare powerdomains $\mathcal{P}[D]$ are constructed from domains D by taking the set of downwards closed subsets of D that are also closed under existing least upper bounds of directed sets in D. These are ordered by inclusion. It can be shown that $\mathcal{P}[D]$ is itself a domain. This suggests we might be able to remain in the world of continuous functions and domains. However, as will be shown in the next section, there are still certain behaviours that cannot be modelled adequately in this setting.

1.2 Models for Dataflow

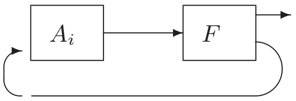
The dataflow paradigm for concurrent computation was introduced in the work of Jack Dennis and others [19, 12]. The key idea is that there is a flow of data between many asynchronous computing agents. These agents are interconnected by communication channels which can be viewed as unbounded buffers. Such behaviour is becoming increasingly common in the real world where the desire for low power and low electrical interference from circuits is encouraging the development of clockless systems.

If the computing agents in the system are deterministic then, as shown in [19], a denotational semantics can be given in terms of the fixed point of a set of equations describing each of the components. This result was used by several authors, for example in the work of Faustini [13], and Lynch and Stark [21].

Brock and Ackerman showed in [5] that giving a semantics to dataflow processes where the components were non-deterministic was far from easy. In particular, *input-output* relations between sequences of values on input and output channels carry too little information about the behaviour of networks to support a compositional semantics such as that given by Kahn. The following example is essentially that presented in [29]. Let A_1 be the non-deterministic process that either outputs a token and stops, or outputs a token, waits for a token on input and then outputs another token. Let A_2 be the process that either outputs a token and stops, or waits for a token on input and then outputs two tokens. The input-output relations between strings of tokens of both A_1 and A_2 are the same:

$$\{(\varepsilon,\ t),\ (t,\ t),\ (t,\ tt)\}\ ,$$

where ε represents the empty string of tokens and the presence of a token is represented by the symbol t. Let F be the process that copies every input to two outputs through which the output of the process A_i is fed back to the input channel.

$C[A_i] =$ 

Observe that the process A_1, placed in this context, produces a process which can output two tokens whereas the process A_2 results in a process that can only output a single token. This confirms that there is no denotational semantics of non-deterministic dataflow in terms of traditional input-output relations. For similar reasons traditional uses of powerdomains also fall short.

Although traditional relations are insufficient, it was shown in [15, 16] that a compositional form of relational semantics is possible, but at the cost of moving to generalised relations - profunctors.

As will be shown in Chapter 4, another alternative is to use spans of *event structures*

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

where $d : E \to A$ models the demands on the input channels and $out : E \to B$ the output on the output channels. In fact, such spans correspond to profunctors of the kind used in [15, 16]. In the following section, we make the definition of spans precise by defining event structures and two kinds of morphism between them – demand morphisms and rigid morphisms.

1.3 Event Structures

In this thesis we concentrate on one particular independence model: *event structures*. Event structures, an invention of Nielson, Plotkin and Winskel, were first seen in [35] and [36]. Often there is no point in analysing exactly where and when events occur in concurrent computations. Instead what is important is to select the significant events and describe how the occurrence of an event causally depends on those that have already occurred. Event structures model concurrent computations as a tuple

$$(E,\ \leq,\ \#),$$

a set of events E accompanied by a causal dependency relation $\leq$ on E and a conflict relation $\#$ between events in E to express how the occurrence of certain events prevents the occurrence of others. The tuple must have the following properties.

- The relation $\leq$ is a partial order.
- The set $\{e' \in E \mid e' \leq e\}$ is finite for all $e \in E$.
- If $e_1 \# e_2$ and $e_2 \leq e_3$ then $e_1 \# e_3$.

An event in a computation can be anything that can be viewed as atomic, i.e., once started, it must finish and it has no internal structure. An example of an event in a computation is two independent agents synchronising. What constitutes an event in a computation depends entirely on the level of abstraction required.

Event structures are members of the collection of models that describe the behaviour of concurrent computations in time. Their future behaviour is dependent entirely on which of their events have occurred so far. They therefore have a built in notion of state consisting of the set of events that have already occurred. The set of states of an event structure E is known as its configurations and is denoted by $\mathcal{C}(E)$.

Event structures provide a simple and intuitive model of concurrent computation. It is easy to define constructions to describe non-determinism, parallel composition and many other relevant behaviours (see Section 2.3).

Processes are often constructed from other processes. Also, one process may be the refinement of a second process. The corresponding relationships between behaviours are often expressed as morphisms in a category with objects that are instances of the model. Often, different kinds of morphism can be chosen according to which kinds of relationship need to be described. We now describe two possible choices of morphism between event structures.

Let $(E, \leq,\ \#)$ be an event structure. Define $[e]$ for $e \in E$ by

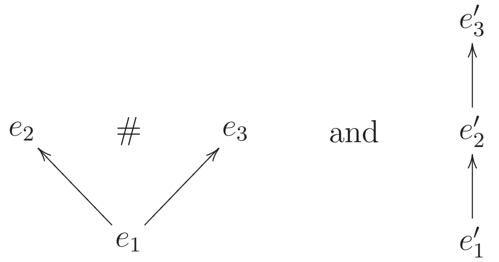
$$[e] \overset{def}{=} \{e' \in E \mid e' \leq e\}.$$

Rigid morphisms: Let $(E_1,\ \leq_1,\ \#_1)$ and $(E_2,\ \leq_2,\ \#_2)$ be event structures. A function $f : E_1 \to E_2$ is a rigid morphism between them if f is total and, for all $e_1,\ e_2 \in E_1$,

- $[f(e_1)] = f[e_1]$,
- if $f(e_1) = f(e_2)$ or $f(e_1) \# f(e_2)$ then $e_1 = e_2$ or $e_1 \# e_2$.

In other words, f preserves the causal dependency relation and downwards closure when extended to act over sets, reflects conflict and can only map two distinct events to the same event when they are in conflict.

If an event structure E_1 has a rigid morphism into E_2 then E_2 can behave exactly like E_1 although it may have fewer non-deterministic choices. For example, there is a rigid morphism between the event structures



(where the causal dependency partial order is shown graphically such that $e \leq e'$ is represented as $e \to e'$) mapping e_1 to e'_1 and mapping e_2 and e_3 to e'_2.

We can further restrict rigid morphisms to preserve conflict. They are then called *rigid embeddings*. Taking only those rigid embeddings that preserve events, we can view event structures as a large CPO [36]. We can therefore make use of the fixed point of any operation on event structures based on continuous constructions with respect to this ordering.

Demand morphisms: Let $(E_1,\ \leq_1,\ \#_1)$ and $(E_2,\ \leq_2,\ \#_2)$ be event structures. Let $\mathcal{C}^0(E_2,\ \leq_2,\ \#_2)$ be the set of finite downwards closed subsets of E_2 with respect to $\leq_2$ and where no two events in any one subset are related by $\#_2$. A function d between E_1 and $\mathcal{C}^0(E_2,\ \leq_2,\ \#_2)$ is a demand morphism between them if

- $e_1 \leq_1 e_2$ implies $d(e_1) \subseteq d(e_2)$,
- if there exist $e'_1,\ e'_2 \in d(e_1) \cup d(e_2)$ such that $e'_1 \#_2 e'_2$ then $e_1 \#_1 e_2$.

Unlike the rigid morphisms which can be viewed as expressing the way in which one event structure can simulate another, demand morphisms relate events of an event structure to the states of another event structure. A demand morphism between two event structures E_1 and E_2 can be thought of as relating an event e in E_1 to the minimum set of events in E_2 that must have occurred in order for e to occur. If $d(e) = x$ for some $e \in E_1$ and x a configuration of E_2, we say e *demands* x.

So, recalling Section 1.2, dataflow processes can be modelled as spans of event structures

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

where d is a demand morphism and *out* is a rigid morphism. For an event $e \in E$ to occur in a process, we require input $d(e)$ and our output will be $out(e)$.

Another example of the use of such spans can be found in the work of Nygaard and Winskel [26]. The powerful meta-language Affine HOPLA can be given an event structure semantics that corresponds to the profunctor semantics given in [26] for first order processes. An Affine HOPLA process $\Gamma \vdash t : \mathbb{P}$ can be represented as a span of event structures

$$\begin{tikzcd} & E_t \arrow[dl, "d"'] \arrow[dr, "out"] & \\ \llbracket \Gamma \rrbracket & & \llbracket \mathbb{P} \rrbracket \end{tikzcd}$$

where $\llbracket \Gamma \rrbracket$ and $\llbracket \mathbb{P} \rrbracket$ are the event structures representing the types in the context Γ and the type $\mathbb{P}$. This is explained fully in Chapter 3. So, spans of event structures can be used to model higher-order processes. We can interpret the span as follows. For an event e to occur in E_t, we demand $d(e)$ from the environment and output $out(e)$.

1.4 Alternative Choices of Morphism

As well as rigid and demand morphisms, there are many other choices of morphism. We also discuss augmentation morphisms and partial morphisms. The definitions arose naturally during the study of event structures representing the synchronous and partially synchronous parallel composition of processes of the style found in CCS and CSP. For example, consider relating the event structure E_3 representing P_3, the partially synchronous parallel composition of two processes P_1 and P_2, to the event structure E_1 representing P_1. Not all events in E_3 will correspond to an event in E_1. Therefore, if we wish to base the relationship on a function between the event sets, partiality is required. Also, the causal dependency relation of E_3 will be a combination of that of E_1 and the event structure representing P_2. This means there may not be a function from the events of E_3 to those of E_1 that preserves causality.

We define partial and augmentation morphisms below. The reader is referred to Section 2.1.1 for more details.

Partial morphisms: Given event structures $(E_1,\ \leq_1,\ \#_1)$ and $(E_2,\ \leq_2,\ \#_2)$, $f : E_1 \to E_2$ is a partial morphism between them iff it is a partial function and, for all $e_1,\ e_2 \in E_1$,

- if $f(e_1)$ is defined then $[f(e_1)] \subseteq f[e_1]$,
- if $f(e_1)$ and $f(e_2)$ are defined then if either $f(e_1) = f(e_2)$ or $f(e_1)\#f(e_2)$ then $e_1 = e_2$ or $e_1\#e_2$.

So, partial morphisms may be seen as a relaxation of rigid morphisms to allow partial functions such that the causal dependency relation is not necessarily preserved.

For example, there is a partial morphism between

$$\begin{tikzcd}
e_3 & & & \\
e_2 \arrow[u] & \text{and} & e'_2 & e'_3 \\
e_1 \arrow[u] & & &
\end{tikzcd}$$

which is undefined for e_1, maps e_2 to e'_2, and e_3 to e'_3.

Augmentation morphisms: These are partial morphisms with underlying functions that are total. Like partial morphisms, augmentation morphisms $f : E_1 \to E_2$ express when one event structure E_2 can behave like E_1 although the events in E_2 may be subject to fewer dependency and conflict constraints. Their name derives from the fact that E_1 may be thought of as having an augmented causal dependency relation between its events in comparison to E_2.

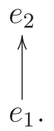
As an example, consider the following. There is an augmentation morphism between

$$\begin{tikzcd}
e_2 & & & \\
e_1 \arrow[u] & \text{and} & e'_1 & e'_2
\end{tikzcd}$$

that maps e_1 to e'_1 and e_2 to e'_2. Observe that e_1 must occur before e_2 but the events they map to can occur concurrently.

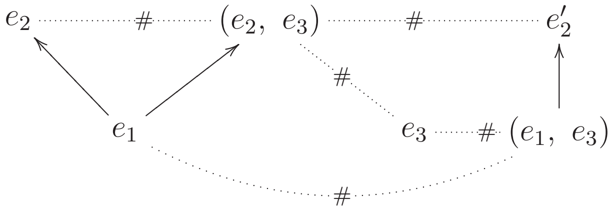
The universal constructions in the categories of event structures formed from rigid, augmentation and partial morphisms are often of interest operationally. For example, the coproduct of event structures in all three categories behaves like the non-deterministic sum of two processes. Below we show an example of a product in the category of event structures and partial morphisms and compare it to a sort of CCS-style (see [23]) partially synchronous parallel composition.

Let E_1 be the event structure



Let E_2 be the event structure consisting of a single event e_3.

The product $E_1 \times_\star E_2$ in the category of partial morphisms is $(E,\ \leq,\ \#)$ which we define graphically below.



We can relate $(E,\ \leq,\ \#)$ to E_1 and E_2 by observing that the events e_1 and e_3 in E correspond to themselves in E_1 and E_2. The events e_2 and e_2' in E correspond to e_2 in E_1. The events $(e_2,\ e_3)$ and $(e_1,\ e_3)$ correspond to synchronisations of events in E_1 and E_2. So, the event structure $(E,\ \leq,\ \#)$ behaves like E_1 and E_2 placed in parallel where their events may or may not synchronise.

The product $E_1 \times E_2$ in the category of augmentation morphisms consists of a single event. It forces all events in E_1 to synchronise with events in E_2 and therefore e_1 synchronises with e_3 but no other synchronisations are possible.

In general, it is difficult to describe products in categories of event structures. To see why, observe how the event e_2 in E_1 is split into three different events in $(E,\ \leq,\ \#)$ according to what events come before it and how it synchronises with other events. In Section 2.2 we describe a way of constructing these products in terms of another structure – the *stable family*. Products in categories of stable families are easy to construct. To summarise, the categories of rigid, augmentation and partial morphisms and event structures are related to categories of stable families by a series of coreflections. We can therefore construct products by including the event structures in the category of stable families, constructing the product and using the right adjoint to produce an event structure from it.

As well as stable families, there are other extensions and generalisations of event structures and their morphisms. In [34], probability is added to event structures. The possibility of allowing multiple consistent causes for events is explored in [37, 7, 4]. In [33], event structures are generalised in order to model disjunctive causality and resolvable conflict. A new kind of morphism is described in [1] that provides us with a notion of quotient event structures. Event structures appear in the study of security where they are known as strand spaces [10, 11]. Also, event structures underlie concrete data-structures and sequential algorithms [20] and later development in game semantics.

In this thesis, we concentrate on event structures of the kind described in Section 1.3 and rigid, augmentation, partial and demand morphisms.

1.5 Varying Spans of Event Structures

In the previous section, we discussed how the product of event structures in the category of augmentation morphisms behaved like a synchronous parallel composition of processes. Such a construction on spans based on rigid and demand morphisms does not fit well. This is because the projections out of the product in the category of augmentation morphisms may not be rigid. This together with other examples suggests that varying the kind of morphisms in spans may yield a more expressive model for higher order concurrent processes. In order to do this systematically we can adopt a similar approach to that of Moggi [24] and consider varying the types of morphisms via monads (and comonads) on the bicategory of spans of rigid morphisms. For example we would seek to model the kind of span

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

consisting of a demand morphism and a rigid morphism as a span of rigid morphisms

$$\begin{tikzcd}
& E \arrow[dl, "d'"'] \arrow[dr, "out"] & \\
T(A) & & B.
\end{tikzcd}$$

Such monads (and comonads) on spans can be induced by cartesian monads on the category of event structures with rigid morphisms (a consequence of Burroni's work in [6]). Let $\mathcal{R}$, $\mathcal{A}$, $\mathcal{P}$ and $\mathcal{D}$ be the categories of rigid, augmentation, partial and demand morphisms. A sensible approach to constructing monads on $\mathcal{R}$ is to attempt to find adjunctions between the categories. The monads can then be constructed as the composition of the adjoint functors. Clearly $\mathcal{R}$ is a subcategory of $\mathcal{A}$. In Chapter 5 we show that there is a right adjoint to the inclusion functor from $\mathcal{R}$ into $\mathcal{A}$.

$$\begin{tikzcd}
\mathcal{R} \arrow[r, hook, bend right] \arrow[r, phantom, "\top"] & \mathcal{A} \arrow[l, bend right]
\end{tikzcd}$$

We can construct a monad T on $\mathcal{R}$ such that the category $\mathcal{A}$ is isomorphic to $\mathcal{R}_T$ where $\mathcal{R}_T$ is the Kleisli category of T. However, there is no adjunction between $\mathcal{R}$ and $\mathcal{P}$ or $\mathcal{D}$ as $\mathcal{R}$ has no terminal object unlike $\mathcal{P}$ and $\mathcal{D}$. In this thesis we seek to address this issue by extending the definition of event structures to include *persistence* such that the corresponding categories of rigid, augmentation, partial and demand morphisms are related by adjunctions. Let $\mathcal{R}_P$, $\mathcal{A}_P$, $\mathcal{P}_P$ and $\mathcal{D}_P$ be the categories of event structures with persistence and rigid, augmentation, partial and demand morphisms. We show that there are adjunctions between $\mathcal{R}_P$ and the other categories.

$$\begin{tikzcd}
\mathcal{R}_P \arrow[r, hook, bend right] \arrow[r, phantom, "\top"] \arrow[d, hook, bend left] \arrow[d, phantom, "\vdash"] & \mathcal{A}_P \arrow[l, bend right] \arrow[r, hook, bend right] \arrow[r, phantom, "\top"] & \mathcal{P}_p \arrow[l, bend right] \\
\mathcal{D}_P \arrow[u, bend left] & &
\end{tikzcd}$$

1.6 Synopsis

We give an overview of each chapter of the thesis. We describe the subjects covered and discuss the overall contribution of each chapter.

Chapter 2: In this chapter we introduce the main categories and constructions that are required for the rest of the thesis. We define event structures together with several different sorts of morphism between them including rigid, augmentation, partial and demand morphisms. We recall how adjunctions with the related categories of stable families allow limits in the categories of event structures to be easily constructed. We then go on to describe asymmetric spans of event structures consisting of a demand morphism and a rigid morphism as discussed in Section 1.3. A composition operation is defined and we confirm that spans form a bicategory with event structures as objects and spans as morphisms.

Chapter 3: We recall the work of Nygaard and Winskel in [26] where spans are used to give a denotational semantics to Affine HOPLA. After giving the types, grammar and term formation rules for Affine HOPLA, we present the constructions described in [26] in greater detail than previously seen. This semantics is consistent with the profunctor semantics described in [27] when only first-order terms are considered. Finally, in Section 3.3, we show several example processes and describe the spans that represent them.

The purpose of this chapter is to demonstrate how event structures can be seen as both types and processes. We also familiarise the reader with the behaviours captured by the various constructions on event structures and their morphisms introduced in Chapter 2. The examples section makes clear the operational nature of the way in which event structures can model concurrent computations.

Chapter 4: We give another example of the types of process that can be modelled by spans of event structures – dataflow processes. Event structures are shown representing types of data at channels in a dataflow process as well as the process itself. In order to model dataflow processes, we need to define a feedback construction. Such constructions must obey certain axioms and are called *trace* operations. We describe a *trace* operation on spans of event structures. We also describe the relationship between event structures and certain sorts of profunctor and use this relationship, together with the results in [15, 16] to demonstrate that the operation obeys the axioms required. We define deterministic spans in order to model deterministic dataflow, describing how they relate to functions from input to output. We then show that the trace construction applied to these spans is identical to using the fixed point construction described in [19] on the functions they represent. We thereby show that the semantics given to processes with feedback by event structures matches that given by other models but is more operational in nature than the general relations given in [15, 16].

Chapter 5: We consider the possibility of extending event structures to capture additional behaviour. As discussed in Section 1.5, the behaviours that can be expressed in terms of constructions on a category of event structures depend on the kind of morphisms chosen. It is desirable to model the different morphisms between event structures as rigid morphisms modified via monads. This would enable all the different behaviours expressible to be defined as constructions based on one kind of morphism. The development of a symmetric span category on which various asymmetric and symmetric span categories could be built would then be possible. This would allow the use of spans obtained from various morphisms while maintaining the ability to relate them to the simple symmetric spans and therefore the spans previously described. In Section 5.1 we show that such monads do not always exist for classical event structures.

In [14], a method for describing weak bisimulation in a categorical manner is given in terms of a monad on the category of presheaves. After describing the presheaves that represent event structures using the properties given in [38], we show that the application of the monad results in a presheaf that does not represent an event structure and give examples illustrating the extra behaviour that event structures cannot express.

The results described above are used to justify an alteration of the definition of event structure to allow *persistent events*. These structures are known as *event structures with persistence*. It is then shown that at least one of the desired adjunctions, that between the category of augmentation morphisms and the category of partial morphisms, now exists. The difficulties of constructing the other monads are discussed.

Chapter 6: We show an interesting relationship between events in an event structure with persistence given by rigid morphisms from it into another event structure. There is a correspondence between rigid morphisms where the domain has an empty conflict relation and *rigid pairs*. These comprise an equivalence relation and a predicate on the events of E which have certain properties. The equivalence relations may be reasonably seen as providing some of the inspiration for the development of event structures with symmetry [39]. Using these rigid pairs, we show how to construct adjunctions between the category of rigid morphisms $\mathcal{R}_P$ and the category of demand morphisms $\mathcal{D}_P$, and also between $\mathcal{R}_P$ and the category of augmentation morphisms $\mathcal{A}_P$. Finally, we use rigid pairs to prove the existence of all finite limits in $\mathcal{R}_P$. This confirms that we can model behaviours like the CCS-style parallel composition described in Section 2.2.2 as constructions in $\mathcal{R}_P$. It also confirms that spans of morphisms of $\mathcal{R}_P$ do indeed form a bicategory.

Chapter 7: We summarise the results described in the thesis and discuss what we can conclude from them. We then discuss further work on event structures that addresses some of the remaining open questions.

Chapter 2

Event Structures and Spans

In this chapter we introduce event structures and their categories. We also define some constructions that will be used later in the thesis. Finally, we introduce spans of event structures.

2.1 An Introduction to Event Structures

Event structures model a process as a set of event occurrences with relations to express how events causally depend on others or exclude other events from occurring. An event structure consists of a set of events, an ordering on the events and a conflict relation.

Definition 2.1.1. *The tuple $(E, \le, \#)$ is an event structure if it obeys the following:*

- *E is a set of events,*
- *$\le$ is a partial order on E,*
- *$[e]$ is a finite set for all e in E (where $[e]$ is the set of events, e', in E such that $e' \le e$),*
- *$\#$ is a binary, irreflexive, symmetric relation between the events in E such that if $e_1 \le e_2$ and $e_1 \# e_3$ then $e_2 \# e_3$.*

We use the name of the set of events interchangeably with the name of the event structure. We also overload $\le$ and $\#$ where it is clear from the context which relation is being referred to.

Definition 2.1.2. *The event structure $(E, \le, \#)$ is an* elementary event structure *iff $\#$ is the empty relation. It is described as being a* prime *elementary event structure iff there exists $e \in E$ such that $e' \le e$ for all $e' \in E$.*

We write $(E, \le)$ for the elementary event structure $(E, \le, \emptyset)$.

If E is a prime elementary event structure then we define $max(E)$ to be the maximal event of E with respect to $\le$.

Definition 2.1.3. *Let E be an event structure. Let $\mathcal{C}(E)$ be the set of subsets of E that are downwards closed with respect to $\leq$ and contain no conflicting events. These are known as configurations of E. We write $\mathcal{C}^0(E)$ for the set of finite configurations of E.*

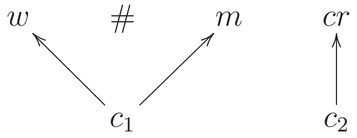
The set of configurations of an event structure when ordered by inclusion forms a prime algebraic domain [25]; the complete primes of this domain have the form $[e]$ for e ranging over E. It follows that the set of primes is isomorphic to the set of events of an event structure.

Definition 2.1.4. *Two configurations x and y of E are consistent, denoted $x \uparrow y$, iff there exists $z \in \mathcal{C}(E)$ that contains them both, i.e., they have a common upper bound when $\mathcal{C}(E)$ is ordered by set-theoretic inclusion. This can be extended to a set of configurations X where we write $X\uparrow$.*

Observe that $e_1 \leq e_2$ in an event structure E iff $[e_1] \subseteq [e_2]$ and $e_1 \# e_2$ iff $[e_1] \not\uparrow [e_2]$ in $\mathcal{C}(E)$.

A behavioural interpretation of the definition of event structures is to see $\leq$ as describing when one event must occur before another and $\#$ as describing when the occurrence of one event prevents the occurrence of another. The conditions imposed on these relations ensure that they behave consistently. The reader is referred to [36] for further details.

The following example illustrates the graphical representation used throughout the document.



The above event structure represents two vending machines, running concurrently. Each machine can accept a coin corresponding to the events c_1 and c_2. One machine can then dispense either a bar of white chocolate (w) or a bar of milk chocolate (m) and the other a bag of crisps (cr).

The example also demonstrates an event structure being used to model a process.

2.1.1 Categories of Event Structures

Taking event structures as the objects, it is possible to form categories of event structures where the morphisms are functions between the sets of events with particular properties. The morphisms express relationships between the behaviours of the event structures as explained in Sections 1.3 and 1.4.

Let $\star$ represent *undefined*.

Definition 2.1.5. *Given event structures $(E_1,\ \leq_1,\ \#_1)$ and $(E_2,\ \leq_2,\ \#_2)$, $f : E_1 \to E_2$ is a partial morphism between them iff it is a partial function and*

- *if $f(e) \neq \star$ then $[f(e)] \subseteq f[e]$ for all e in E_1,*
- *$(f(e_1) = f(e_2) \neq \star$ or $f(e_1)\#f(e_2)) \;\Rightarrow\; (e_1 = e_2$ or $e_1\#e_2)$ for all e_1 and $e_2 \in E_1$.*

Composition of morphisms is simply function composition.

The category given by partial morphisms is referred to as $\mathcal{P}$.

Definition 2.1.6. *Given event structures* $(E_1,\ \leq_1,\ \#_1)$ *and* $(E_2,\ \leq_2,\ \#_2)$*,* $f : E_1 \to E_2$ *is an augmentation morphism between them iff it is a partial morphism whose underlying function is total.*

The category given by augmentations is referred to as $\mathcal{A}$.

Definition 2.1.7. *Given event structures* $(E_1,\ \leq_1,\ \#_1)$ *and* $(E_2,\ \leq_2,\ \#_2)$*,* $f : E_1 \to E_2$ *is a rigid morphism between them iff it is an augmentation morphism and*

$$[f(e)] = f[e]$$

for all e *in* E_1*.*

The category given by rigid morphisms will be referred to as $\mathcal{R}$.

The following property of rigid morphisms proves useful several times in future chapters.

Lemma 2.1.8. *Let* $f : E_1 \to E_2$ *be a rigid morphism that preserves conflict and also an isomorphism between the set* E_1 *and the set* E_2*. Then* $f^{-1} : E_2 \to E_1$ *is a rigid morphism.*

Proof. We show that $[f^{-1}(e)] = f^{-1}[e]$ for all $e \in E_2$.

Suppose $e_1 \leq e_2$ in E_2. Then, as f is an isomorphism between the event sets, there exists an e_1' and an e_2' in E_1 such that $f(e_1') = e_1$ and $f(e_2') = e_2$. As f is rigid, $e_1' \leq e_2'$ and therefore $f^{-1}(e_1) \leq f^{-1}(e_2)$ and so $f^{-1}[e] \subseteq [f^{-1}(e)]$ for all $e \in E_2$. Suppose $e_1 \leq f^{-1}(e)$ in E_1. Then, as f is rigid, $f(e_1) \in [e]$. As $f^{-1} \circ f(e_1) = e_1$ we have that $e_1 \in f^{-1}[e]$ and therefore that $[f^{-1}(e)] \subseteq f^{-1}[e]$.

Let e and e' be events in E_2. Finally we show that $e = e'$ or $e\#e'$ if $f^{-1}(e) = f^{-1}(e')$ or $f^{-1}(e)\#f^{-1}(e')$. If $f^{-1}(e) = f^{-1}(e')$ then $e = e'$ as f is an isomorphism between the sets of events. If $f^{-1}(e)\#f^{-1}(e')$ then, as f preserves conflict, $e\#e'$. □

Definition 2.1.9. *Given event structures* $(E_1,\ \leq_1,\ \#_1)$ *and* $(E_2,\ \leq_2,\ \#_2)$*, the function* $f : E_1 \to \mathcal{C}^0(E_2)$ *is a demand morphism between them iff it is total and*

- *if* $e_1 \leq e_2$ *then* $f(e_1) \subseteq f(e_2)$ *for all* e_1 *and* e_2 *in* E_1*,*
- *if* $f(e_1) \not\uparrow f(e_2)$ *then* $e_1\#e_2$ *for all* e_1 *and* e_2 *in* E_1*.*

Let $d : E \to E'$ be a demand morphism. Define the function $d^\dagger : \mathcal{C}(E) \to \mathcal{C}(E')$ by

$$d^\dagger(x) = \bigcup_{e \in x} d(e)$$

for all configurations x of E. The composition of two demand morphisms $d_1 : E \to E'$ and $d_2 : E' \to E''$ is defined to be $d_2^\dagger \circ d_1$.

The category of event structures with demand morphisms will be referred to as $\mathcal{D}$.

It is possible to view rigid morphisms as demand morphisms. We say that a rigid morphism $f : E_1 \to E_2$ corresponds to a demand morphism $f' : E_1 \to E_2$ where f' is defined by $f'(e) \overset{def}{=} f([e])$. So, $\mathcal{R}$ is included in $\mathcal{A}$ which is included in $\mathcal{P}$ and clearly there is a faithful functor $F : \mathcal{R} \to \mathcal{D}$.

$$\begin{tikzcd}
\mathcal{D} & \mathcal{R} \arrow[l, "F"'] \arrow[r, hook] & \mathcal{A} \arrow[r, hook] & \mathcal{P}
\end{tikzcd}$$

2.1.2 Labelled Event Structures

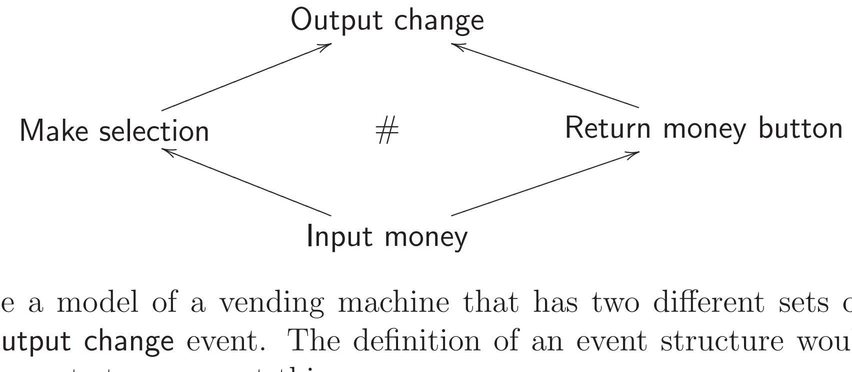
Often a process can perform an action multiple times. An event structure can only perform each event once. So, if an event structure is used to model such a process, each occurrence of an action must be modelled by a different event. It is therefore sometimes useful to label events in an event structure. We can do this by extending event structures from being of the form $(E,\ \leq,\ \#)$ to being a tuple $(E,\ \leq,\ \#,\ l)$ where l is a function from E to a set of labels L. As for event structures without labels, we can form categories of these event structures where the morphisms have the properties described above but in addition preserve the labelling function, i.e., if $f : (E_1,\ \leq_1,\ \#_1,\ l_1) \to (E_2,\ \leq_2,\ \#_2,\ l_2)$ is a morphism then $l_1(e_1) = l_2(f(e_1))$ if $f(e_1)$ is defined. We write $\mathcal{R}_L$ for the category of labelled-event structures with labelling set L and rigid, label-preserving morphisms.

2.2 Stable Families

2.2.1 An Introduction to Stable Families

Instead of defining a global causality on events, it is possible to make the causality local, i.e., a single event may be dependent upon two inconsistent sets of events, requiring only one set to be enabled. Consider the diagram below for an example, presented graphically in the same way as for event structures.

Events: Output change, Make selection, Input money, Return money button.



Above we see a model of a vending machine that has two different sets of events that trigger the output change event. The definition of an event structure would necessitate two different events to represent this.

The purpose of this section is to define several categories of stable families and constructions, and to relate them to categories of event structures via adjunctions. For a more detailed description of these structures, the reader is referred to [36].

Definition 2.2.1. *A stable family $\mathcal{F}$ is a set of sets with the following properties.*

- Coherence*:* $\forall X \subseteq \mathcal{F}.\ (\forall x,\ y \in X. x \uparrow y) \Rightarrow \bigcup X \in \mathcal{F}$.
- Stability*:* $\forall X \subseteq \mathcal{F}.\ X \uparrow \Rightarrow \bigcap X \in \mathcal{F}$.
- Coincidence-freeness*:*

$$\begin{aligned}&\forall x \in \mathcal{F},\ e_1,\ e_2 \in x.\ e_1 \neq e_2 \Rightarrow \\ &\quad \exists y \in \mathcal{F}.\ y \subseteq x \text{ and } ((e_1 \in y \text{ and } e_2 \notin y) \text{ or } (e_2 \in y \text{ and } e_1 \notin y)).\end{aligned}$$

- Finiteness*:* $\forall x \in \mathcal{F}. \forall e \in x. \exists y \in \mathcal{F}.\ y \subseteq x$ and $e \in y$ and y is finite.

Here $\uparrow$ has the same meaning it had for the set of configurations of an event structure.

The set of events of a stable family $\mathcal{F}$ is $\bigcup \mathcal{F}$.

It can be deduced that a stable family, ordered by inclusion, forms a *prime algebraic* domain [25, 36]. If x is a member of a stable family $\mathcal{F}$ and $e \in x$ let

$$[e]_x \overset{def}{=} \bigcap \{y \in \mathcal{F} | e \in y \text{ and } y \subseteq x\}.$$

Then $[e]_x \in \mathcal{F}$ and is called a *prime configuration* of $\mathcal{F}$. (It is a complete prime of $(\mathcal{F},\ \subseteq)$ in the sense of [25].) Because of coincidence-freeness, taking $max([e]_x) \overset{def}{=} e$ is well-defined.

Event structures, due to their simplicity, are, in general, more easily understood than stable families. However, as will later be demonstrated, certain useful constructions are more concisely described in terms of stable families.

2.2.2 Categories of Stable Families

As for event structures, it is possible to define three categories of stable families, $\mathcal{P}_F$, $\mathcal{A}_F$ and $\mathcal{R}_F$. Each of the categories consists of stable families as objects and functions between events as morphisms. Let $\mathcal{F}_1$ and $\mathcal{F}_2$ be stable families.

The partial function $f : \bigcup \mathcal{F}_1 \to \bigcup \mathcal{F}_2$ is a morphism between $\mathcal{F}_1$ and $\mathcal{F}_2$ in $\mathcal{P}_F$ iff

i) $f(x) \in \mathcal{F}_2$, for all $x \in \mathcal{F}_1$,

ii) $\forall e_1,\ e_2 \in x.\ f(e_1) = f(e_2) \neq \star \ \Rightarrow\ e_1 = e_2$ for all $x \in \mathcal{F}_1$. (We again use $\star$ to represent *undefined*.)

The function $f : \bigcup \mathcal{F}_1 \to \bigcup \mathcal{F}_2$ is a morphism between $\mathcal{F}_1$ and $\mathcal{F}_2$ in $\mathcal{A}_F$ iff it is a morphism between $\mathcal{F}_1$ and $\mathcal{F}_2$ in $\mathcal{P}_F$ and is total.

The function $f : \bigcup \mathcal{F}_1 \to \bigcup \mathcal{F}_2$ is a morphism between $\mathcal{F}_1$ and $\mathcal{F}_2$ in $\mathcal{R}_F$ iff it is a morphism in $\mathcal{A}_F$ and

$$y \subseteq f(x) \ \Rightarrow\ \exists x' \in \mathcal{F}_1.\ x' \subseteq x \text{ and } f(x') = y$$

for all $y \in \mathcal{F}_2$ and $x \in \mathcal{F}_1$.

All the categories have products. Behaviourally, these can be thought of as describing several sorts of parallel composition.

Let X and Y be sets. Define $X \times_* Y$ to be $(X \cup \{\star\} \times Y \cup \{\star\}) \backslash \{(\star,\ \star)\}$.

The set x is a member of $\mathcal{F}_1 \times \mathcal{F}_2$, the product of $\mathcal{F}_1$ and $\mathcal{F}_2$ in $\mathcal{P}_F$ iff

i) $x \subseteq (\bigcup \mathcal{F}_1) \times_* (\bigcup \mathcal{F}_2)$,

ii) $\pi_1(x) \in \mathcal{F}_1$ and $\pi_2(x) \in \mathcal{F}_2$,

iii) $\forall e,\ e' \in x.\ \pi_1(e) = \pi_1(e') \neq \star$ or $\pi_2(e) = \pi_2(e') \neq \star \ \Rightarrow\ e = e'$,

iv) $\forall e,\ e' \in x.\ e \neq e' \ \Rightarrow\ \exists y \subseteq x.\ \pi_1(y) \in \mathcal{F}_1$ and $\pi_2(y) \in \mathcal{F}_2$ and $((e \in y$ and $e' \notin y)$ or $(e \notin y$ and $e' \in y))$,

v) $\forall e \in x.\ \exists y \subseteq x.\ \pi_1(y) \in \mathcal{F}_1$ and $\pi_2(y) \in \mathcal{F}_2$ and $e \in y$ and y is finite.

This product in $\mathcal{P}_F$ is referred to as the *partially synchronous product*. It behaves in much the same way as the parallel composition of two CCS processes [23]. Events in $\mathcal{F}_1$ and $\mathcal{F}_2$ can occur independently, i.e., be paired up with $\star$ or synchronise with each other.

If we have two morphisms $f : \mathcal{F}_1 \to \mathcal{F}_3$ and $g : \mathcal{F}_2 \to \mathcal{F}_3$ then we can construct the pullback by taking the product of $\mathcal{F}_1$ and $\mathcal{F}_2$ and restricting it to those events $(e_1,\ e_2)$ for which $f(e_1) = g(e_2)$.

The product and pullback in $\mathcal{A}_F$ are defined in a similar way. Here, $x \subseteq \bigcup \mathcal{F}_1 \times \bigcup \mathcal{F}_2$ is a member of $\mathcal{F}_1 \times \mathcal{F}_2$ iff conditions (ii), (iv) and (v) hold and

$$\pi_1(e) = \pi_1(e') \text{ or } \pi_2(e) = \pi_2(e') \ \Rightarrow\ e = e'$$

holds for all $e,\ e' \in x$. This construction is referred to as the synchronous product. Here all events in $\mathcal{F}_1$ must synchronise (be paired up) with an event in $\mathcal{F}_2$.

The product and pullback in $\mathcal{R}_F$ are defined in the same way as for $\mathcal{A}_F$ but with the additional requirement that

$$\exists x' \subseteq x.\ \pi_1(x') \in \mathcal{F}_1 \text{ and } \pi_2(x') \in \mathcal{F}_2 \text{ and } \pi_i(x') = y$$

holds for all $y \in \mathcal{F}_i$ that are subsets of $\pi_i(x)$.

Other limits and colimits can be constructed based upon the corresponding limits and colimits in the category of sets.

2.2.3 Relating Stable Families and Event Structures

In Section 1.4 we mentioned the existence of adjunctions between the categories of event structures and those of stable families. We make this precise below.

There are functors, $F_R : \mathcal{R} \to \mathcal{R}_F,\ F_A : \mathcal{A} \to \mathcal{A}_F$ and $F_P : \mathcal{P} \to \mathcal{P}_F$ between the categories. Let $(E,\ \leq,\ \#)$ be an event structure and $f : E_1 \to E_2$ be a morphism. Define F_X by

$$\begin{aligned} F_X(E,\ \leq,\ \#) &\overset{def}{=} \mathcal{C}(E) \\ F_X(f)(e) &\overset{def}{=} f(e) \end{aligned}$$

for $X = \mathcal{R},\ \mathcal{A},\ \mathcal{P}$.

Each of these functors has a right adjoint $G_X : X_F \to X$. Let $\mathcal{F}$ be a stable family and $g : \mathcal{F}_1 \to \mathcal{F}_2$ be a morphism in X_F. Define $G_X(\mathcal{F})$ to be the event structure $(E,\ \leq,\ \#)$ defined below.

- E is the set of prime configurations of $\mathcal{F}$,
- $x \leq x'$ iff $x \subseteq x'$,
- $x\#x'$ iff $x \not\uparrow x'$.

We define the action of G_X on morphisms by

$$G_X(g)([e]_x) \ \overset{def}{=}\ [g(e)]_{g(x)}.$$

The reader is referred to [36] for the proof that these functors do indeed form adjunctions between the categories.

For $X = \mathcal{R},\ \mathcal{A}$ or $\mathcal{P}$ we have that F_X is a full embedding and therefore that the adjunction is a coreflection. As discussed in Section 1.4, this means that definitions of products and indeed all limits in the categories of event structures can be given in terms of stable families.

2.3 Constructions on Event Structures

We now define some constructions that prove useful when describing processes in terms of event structures.

Restriction: The restriction of an event structure $(E,\ \leq,\ \#)$ to S, where S is a subset of E downwards closed with respect to $\leq$, is $(S,\ \leq',\ \#')$ where $e_1 \leq' e_2$ iff $e_1 \leq e_2$ and $e_1\#'e_2$ iff $e_1\#e_2$ for $e_1,\ e_2 \in S$, i.e., we use the standard notion of restriction of relations. We write $E \upharpoonright S$ for $(S,\ \leq',\ \#')$. It follows that the injection of S into E is a rigid morphism.

Sum: Given a set A and a set of event structures $\{E_a \mid a \in A\}$ we write their sum as the event structure $\sum_{a\in A} E_a$. Take $\biguplus$ to be the disjoint union constructor on sets. Then define $\sum_{a\in A} E_a$ by

$$\sum_{a\in A} E_a \overset{def}{=} (\biguplus_{a\in A} E_a,\ \leq,\ \#)$$

where $(a_1,\ e_1) \leq (a_2,\ e_2)$ iff $a_1 = a_2$ and $e_1 \leq e_2$. Also, conflict in $\sum_{a\in A} E_a$ is defined by $(a_1,\ e_1)\#(a_2,\ e_2)$ iff $a_1 \neq a_2$ or $(a_1 = a_2$ and $e_1\#e_2)$. This construction allows us to model non-determinism. The occurrence of an event in one of the event structures in the set prevents any of the events in the other event structures occurring. It is easy to show that the sum construction is the coproduct in all the categories of event structures. When we are taking the sum of two event structures E_1 and E_2 we write $E_1 + E_2$.

Tensor: Given event structures E_1 and E_2, define $\otimes$ by

$$E_1 \otimes E_2 \overset{def}{=} (E_1 \uplus E_2,\ \leq,\ \#)$$

where $(i,\ e_1) \leq (j,\ e_2)$ iff $i = j$ and $e_1 \leq e_2$ and where $(i,\ e_1)\#(j,\ e_2)$ iff $i = j$ and $e_1\#e_2$. This construction acts as an asynchronous parallel composition of the event structures, i.e., it allows them to run concurrently with each other. Where there is no ambiguity we write events $(i,\ e)$ as e and write the extraction of E_i events from a configuration of $E_1 \otimes E_2$ as intersection with E_i. We can also define $\otimes$ for rigid and demand morphisms. If $f_1 : E_1 \to E_2$ and $f_2 : E_3 \to E_4$ are rigid morphisms then let $f_1 \otimes f_2 : E_1 \otimes E_3 \to E_2 \otimes E_4$ be defined by $(f_1 \otimes f_2)(i,\ e) = (i,\ f_i(e))$ for $i = 1,\ 2$. If $d_1 : E_1 \to E_2$ and $d_2 : E_3 \to E_4$ are demand morphisms then define $d_1 \otimes d_2$ by $(d_1 \otimes d_2)(i,\ e) \overset{def}{=} \{i\} \times d_i(e)$ for $i = 1,\ 2$.

Lifting: Given an event structure E we define $E_\perp$ by

$$E_\perp \overset{def}{=} (\{\emptyset\} \cup \{[e] \mid e \in E\},\ \subseteq,\ \uparrow).$$

This construction has the action of prefixing the original event structure with an event that must occur before all others. We can define a lifting operation on rigid morphisms. If $f : E \to E'$ is a rigid morphism then $f_\perp : E_\perp \to E'_\perp$ is defined by $f_\perp(\emptyset) = \emptyset$ and $f_\perp([e]) = [f(e)]$ for all $e \in E$.

Function Space: We describe this constructor in terms of a stable family. Let E_1 and E_2 be event structures. For any set $f \subseteq \mathcal{C}(E_1) \times E_2$, we write $\pi_2(f,\ E)$ for $\{e \in E_2 \mid \exists x \in \mathcal{C}(E_1).\ x \subseteq E$ and $(x,\ e) \in f\}$. We also make use of the shorthand $\pi_2(f) = \pi_2(f,\ {\pi_1}^\dagger f)$ where ${\pi_1}^\dagger f$ is defined to be $\bigcup_{(x,\ e)\in f} x$. We define $E_1 \multimap E_2$ to be $G_{\mathcal{R}}(\mathcal{F}_{E_1 \multimap E_2})$ where f is a member of $\mathcal{F}_{E_1 \multimap E_2}$ if

i) $\pi_1^\dagger f \in \mathcal{C}(E_1)$

ii) $\forall x \in \mathcal{C}(E_1).\ \pi_2(f,\ x) \in \mathcal{C}(E_2)$

iii) $\forall (x,\ e),\ (x',\ e) \in \mathcal{C}(E_1) \times E_2.\ (x,\ e) \in f$ and $(x',\ e) \in f \Rightarrow x = x'$.

Fixed Point: It was shown in [36] that the collection of event structures could be viewed as a *large CPO*, i.e., there is an ordering on event structures that has all the characteristics of a CPO except for the fact that event structures form a collection and not a set. Given event structures E_1 and E_2 we say $E_1 \sqsubseteq E_2$ iff $E_1 = E_2 \upharpoonright E_1$. The categories of event structures can therefore be viewed as domains with the empty event structure $\emptyset$ as $\perp$. It is shown in [26] that all the above constructions are continuous with respect to this ordering. This shows that fixed points exist for any operation built from these constructions. So, if F is an expression based on the constructions described and contains a variable X, then $\mu X.F$ is well-defined.

2.4 Spans of Event Structures

It has been well-known since [25] that event structures can represent both processes and datatypes, which at the time of [25] was remarked on as creating a curious mismatch with classical denotational semantics (where a process denotes an element of a domain). The double role of event structures can be resolved by working with spans of event structures. A span

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

represents a computation process from a type, represented by event structure A, to the type B, an event structure, as again an event structure E. The morphisms d and out specify how the event structure inspects input and delivers output. There are many possible variations on spans as the morphisms d and out can have different natures.

For the purposes of this thesis, we will take a span of event structures to be defined as follows.

Definition 2.4.1. *If we have event structures E, A, B, a rigid morphism $out : E \rightarrow B$ and a demand morphism $d : E \rightarrow A$ then*

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

is a span of event structures.

If the demand morphism $d : E \rightarrow A$ of $A \overset{d}{\leftarrow} E \overset{out}{\rightarrow} B$ maps every event to the empty set then we write it as $E \overset{out}{\rightarrow} B$.

Where there is no ambiguity, we refer to spans by their vertex.

For all event structures A and B we have a category of spans $Span(A,\ B)$. The morphisms between spans are rigid morphisms with certain properties.

Definition 2.4.2. *The rigid morphism $f : E_1 \rightarrow E_2$ is a morphism between the spans $A \overset{d_1}{\leftarrow} E_1 \overset{out_1}{\rightarrow} B$ and $A \overset{d_2}{\leftarrow} E_2 \overset{out_2}{\rightarrow} B$ iff $out_2 \circ f = out_1$ and $d_2 \circ f \subseteq d_1$.*

$$\begin{tikzcd}
& E_1 \arrow[dl, bend right, "d_1"'] \arrow[d, "f"] \arrow[dr, "out_1"] & \\
A \arrow[ur, phantom, "\supseteq"] & E_2 \arrow[l, "d_2"'] \arrow[r, "out_2"] & B
\end{tikzcd}$$

2.4.1 Composition of Spans

We define the composition of spans as follows.

Definition 2.4.3. *Given two spans,*

$$\begin{tikzcd}
 & E_1 \arrow[dl, "d_1"'] \arrow[dr, "out_1"] & & E_2 \arrow[dl, "d_2"'] \arrow[dr, "out_2"] & \\
A & & B & & C
\end{tikzcd}$$

their composition is the span

$$\begin{tikzcd}
 & E_2 \circ E_1 \arrow[dl, "d"'] \arrow[dr, "out"] & \\
A & & C
\end{tikzcd}$$

.

$E_2 \circ E_1$ *is defined as follows:*

- Events*:* $\{(x,\ e) \in \mathcal{C}(E_1) \times E_2 \mid out_1(x) = d_2(e)\}$,
- Causality*:* $(x_1,\ e_1) \leq (x_2,\ e_2)$ *iff* $x_1 \subseteq x_2$ *and* $e_1 \leq e_2$,
- Conflict*:* $(x_1,\ e_1)\#(x_2,\ e_2)$ *iff* $x_1 \not\uparrow x_2$ *or* $e_1\#e_2$.

In fact, viewing the rigid morphisms as demand morphisms, this corresponds to constructing a pullback in $\mathcal{D}$ (see page 127 of [26]).

Lemma 2.4.4. *Given three spans* $A \overset{d_1}{\leftarrow} E_1 \overset{out_1}{\rightarrow} B$, $B \overset{d_2}{\leftarrow} E_2 \overset{out_2}{\rightarrow} C$ *and* $C \overset{d_3}{\leftarrow} E_3 \overset{out_3}{\rightarrow} D$*, the span*

$$\begin{tikzcd}
 & & & P_2 \arrow[dl, "\pi_1"'] \arrow[ddrr, "\pi_2"] & & & \\
 & & P_1 \arrow[dl, "\pi_1"'] \arrow[dr, "\pi_2"] & & & & \\
 & E_1 \arrow[dl, "d_1"'] \arrow[dr, "out_1"] & & E_2 \arrow[dl, "d_2"'] \arrow[dr, "out_2"] & & E_3 \arrow[dl, "d_3"'] \arrow[dr, "out_3"] & \\
A & & B & & C & & D
\end{tikzcd}$$

is isomorphic to

$$\begin{tikzcd}
 & & & P_4 \arrow[ddll, "\pi_1"'] \arrow[dr, "\pi_2"] & & & \\
 & & & & P_3 \arrow[dl, "\pi_1"'] \arrow[dr, "\pi_2"] & & \\
 & E_1 \arrow[dl, "d_1"'] \arrow[dr, "out_1"] & & E_2 \arrow[dl, "d_2"'] \arrow[dr, "out_2"] & & E_3 \arrow[dl, "d_3"'] \arrow[dr, "out_3"] & \\
A & & B & & C & & D.
\end{tikzcd}$$

Proof. Recall that the composition of two spans corresponds to a pullback in $\mathcal{D}$, the category of demand morphisms. We show that Lemma 2.4.4 follows from the first part of Proposition 2.5.9, page 54 of [3]. The proposition states that, in a category, given the diagram

$$\begin{tikzcd}
A \arrow[r, "a"] \arrow[d, "c"'] \arrow[dr, phantom, "(I)"] & B \arrow[r, "b"] \arrow[d, "d"'] \arrow[dr, phantom, "(II)"] & C \arrow[d, "e"] \\
D \arrow[r, "f"'] & E \arrow[r, "g"'] & F ,
\end{tikzcd}$$

if squares (I) and (II) are pullbacks then the outer square is also a pullback.

Observe that $E_1 \overset{\pi_1}{\leftarrow} P_1 \overset{\pi_2}{\rightarrow} E_2$ and $E_2 \overset{\pi_1}{\leftarrow} P_3 \overset{\pi_2}{\rightarrow} E_3$ are also spans and can therefore be composed. We show that the vertex X of this span is isomorphic to both P_1 and P_2 via the above result.

Consider the diagram

$$\begin{tikzcd}
X \arrow[r, "\pi_2"] \arrow[d, "\pi_1"'] & P_3 \arrow[r, "\pi_2"] \arrow[d, "\pi_1"'] & E_3 \arrow[d, "d_3"] \\
P_1 \arrow[r, "\pi_2"'] & E_2 \arrow[r, "out_2"'] & C
\end{tikzcd} .$$

We know from the proposition that the outer square must be a pullback also and therefore, from the uniqueness of pullbacks, we have that X is isomorphic to P_2. A similar argument shows us that X is isomorphic to P_4 and therefore that P_2 is isomorphic to P_4. Composition is therefore associative up to isomorphism. □

It follows from the results in [22] pages 283-285 that the category with event structures as objects, spans as arrows and the morphisms described in Definition 2.4.2 as 2-cells is a bicategory. The identity span for an event structure E is

$$\begin{tikzcd}
 & E \arrow[dl, "{[\text{-}]}"'] \arrow[dr, "id_E"] & \\
E & & E.
\end{tikzcd}$$

In fact there is a relationship between spans of event structures and the stable functions of Berry [2]. These functions correspond to *deterministic* spans which we will give details of in Section 4.3.

Chapter 3

Affine HOPLA

Affine HOPLA is a meta-language that is sufficiently expressive to encode many kinds of processes. Affine languages are of particular interest as, in many environments, a process cannot be copied but may be discarded, i.e., a process is used at most once. In [27], Affine HOPLA was given a denotational semantics by making use of profunctors. Here we describe in detail a semantics in terms of event structure spans first seen in [26]. It can be shown that this semantics corresponds to the profunctor semantics for first-order processes. The purpose of this chapter is to illustrate the use of spans of event structures to give a semantics to a higher-order process calculus.

3.1 Modelling the Types of Affine HOPLA

In this section, we show how event structures can be used to model Affine HOPLA types.

The types of Affine HOPLA are given by the following grammar.

$$\mathbb{T} ::= \mathbb{T}_1 \multimap \mathbb{T}_2 \mid \mathbb{T}_1 \otimes \mathbb{T}_2 \mid \sum_{\alpha \in A} \mathbb{T}_\alpha \mid \mathbb{T}_\perp \mid \mathbf{T} \mid \mu_j \vec{\mathbf{T}}.\vec{\mathbb{T}}.$$

The variable T is taken from a set of type variables and

$$\mu_j \vec{T}.\vec{\mathbb{T}} \text{ abbreviates } \mu_j T_1, ..., T_k.(\mathbb{T}_1, ..., \mathbb{T}_k)$$

and represents the j^{th}-component of the least solution to the equations,

$$(T_1 = \mathbb{T}_1), ..., (T_k = \mathbb{T}_k),$$

where $\mathbb{T}_1, ...,\ \mathbb{T}_k$ may contain the T_js.

The types model the behaviours of which processes are capable. A process is of type

- $\mathbb{T}_1 \otimes \mathbb{T}_2$ if it behaves like the asynchronous parallel composition of a process of type $\mathbb{T}_1$ with a process of type $\mathbb{T}_2$,
- $\sum_{a \in A} \mathbb{T}_a$ if it behaves like the non-deterministic sum of a set of processes with types $\mathbb{T}_a$ for $a \in A$,

- $\mathbb{T}_\perp$ if it can perform an action and then behaves like a process of type $\mathbb{T}$,
- $\mathbb{T}_1 \multimap \mathbb{T}_2$ if, given a process of type $\mathbb{T}_1$, it behaves as a process of type $\mathbb{T}_2$, i.e., it is a function type.

Recursive types can be used to express the repeated combinations of the other constructions.

Making use of the constructions described in Section 2.3, we can model these types as event structures. So, the empty event structure $\emptyset$ with no events is a type of processes that can do nothing. We build up the more complex types as follows. Let $[\![\mathbb{T}]\!]$ be the event structure that models $\mathbb{T}$. We define this by

$$\begin{aligned}
[\![\mathbb{T}_1 \otimes \mathbb{T}_2]\!] &= [\![\mathbb{T}_1]\!] \otimes [\![\mathbb{T}_2]\!] \\
[\![\sum_{a \in A} \mathbb{T}_a]\!] &= \sum_{a \in A} [\![\mathbb{T}_a]\!] \\
[\![\mathbb{T}_\perp]\!] &= [\![\mathbb{T}]\!]_\perp \\
[\![\mu_j \vec{T}.\vec{\mathbb{T}}]\!] &= \mu_j \vec{T}.[\![\vec{\mathbb{T}}]\!] \\
[\![\mathbb{T}_1 \multimap \mathbb{T}_2]\!] &= [\![\mathbb{T}_1]\!] \multimap [\![\mathbb{T}_2]\!] \,.
\end{aligned}$$

In the earlier cases, it is easy to see that the constructions on event structures capture the correct behaviour. In order to see that the function type behaves in the correct way, consider the following.

The adjunction between event structures and stable families defined in Section 2.2.3 yields a bijective correspondence between rigid morphisms between E and $A \multimap B$ in $\mathcal{R}$ and morphisms in $\mathcal{R}_F$ from $\mathcal{C}(E)$ to $\mathcal{F}_{A \multimap B}$. Making use of this we can show that for all spans $A \overset{d}{\leftarrow} E \overset{out}{\rightarrow} B$ there is a unique morphism $m : E \to A \multimap B$ for which the following diagram commutes.

$$\begin{tikzcd}
& E \arrow[dl, "d"'] \arrow[d, dotted, "m"] \arrow[dr, "out"] & \\
A & A \multimap B \arrow[l, "G_R(\pi_1)"] \arrow[r, "G_R(\pi_2)"'] & B
\end{tikzcd}$$

So, the span $A \overset{d}{\leftarrow} E \overset{out}{\rightarrow} B$ corresponds to a rigid morphism from E to $A \multimap B$ and therefore to a span $\emptyset \overset{\emptyset}{\leftarrow} E \overset{out'}{\rightarrow} (A \multimap B)$. This shows how $A \multimap B$ internalises spans from A to B.

Also, it is the case that $(A \otimes B) \multimap C \;\cong\; A \multimap (B \multimap C)$ holds for all event structures A, B and C. For further details, the reader is referred to [26].

3.2 A Compositional Semantics

We now describe a compositional semantics for Affine HOPLA in terms of spans of event structures.

Affine HOPLA has terms given by

$$\begin{aligned} t,\ u \ ::=\ & x \mid rec\ x.t \mid \sum_{i\in I} t_i \mid \lambda x.t \mid t\ u \mid \beta t \mid \pi_\beta t \mid !t \mid t\otimes u \mid \\ & [u\ >\ !x\ \Rightarrow\ t] \mid [u > w\otimes x\ \Rightarrow\ t] \mid abs\ t \mid rep\ t. \end{aligned}$$

Note that the x in $[u\ >\ !x\ \Rightarrow\ t]$ binds all free occurrences of x in t and, similarly, w and x are binding occurrences in $[u > w\otimes x\ \Rightarrow\ t]$.

The term formation rules are shown below.

Structural rules:
$$\frac{}{x:\mathbb{P}\vdash x:\mathbb{P}} \qquad \frac{\Gamma\vdash t{:}\mathbb{Q}}{\Gamma,x{:}\mathbb{P}\vdash t{:}\mathbb{Q}}$$

$$\frac{\Gamma,y{:}\mathbb{Q},x{:}\mathbb{P},\Delta\vdash t{:}\mathbb{R}}{\Gamma,x{:}\mathbb{P},y{:}\mathbb{Q},\Delta\vdash t{:}\mathbb{R}}$$

Recursive definition:
$$\frac{\Gamma,x{:}\mathbb{P}\vdash t{:}\mathbb{P}}{\Gamma\vdash rec\ x.t{:}\mathbb{P}}$$

Non-deterministic sum:
$$\frac{\Gamma\vdash t_j{:}\mathbb{P}\quad \forall j{\in}I}{\Gamma\vdash\sum_{i\in I} t_i{:}\mathbb{P}}$$

Function space:
$$\frac{\Gamma,x{:}\mathbb{P}\vdash t{:}\mathbb{Q}}{\Gamma\vdash\lambda x.t{:}\mathbb{P}\multimap\mathbb{Q}} \qquad \frac{\Gamma\vdash t{:}\mathbb{P}\multimap\mathbb{Q}\quad \Delta\vdash u{:}\mathbb{P}}{\Gamma,\Delta\vdash t\ u{:}\mathbb{Q}}$$

Sum type:
$$\frac{\Gamma\vdash t{:}\mathbb{P}_\beta\quad \beta{\in}A}{\Gamma\vdash\beta t{:}\sum_{\alpha\in A}\mathbb{P}_\alpha} \qquad \frac{\Gamma\vdash t{:}\sum_{\alpha\in A}\mathbb{P}_\alpha\quad \beta{\in}A}{\Gamma\vdash\pi_\beta t{:}\mathbb{P}_\beta}$$

Prefixing:
$$\frac{\Gamma\vdash u{:}\mathbb{P}}{\Gamma\vdash !u{:}\mathbb{P}_\perp} \qquad \frac{\Gamma,x{:}\mathbb{P}\vdash t{:}\mathbb{Q}\quad \Delta\vdash u{:}\mathbb{P}_\perp}{\Gamma,\Delta\vdash[u{>}!x{\Rightarrow}t]{:}\mathbb{Q}}$$

Tensor:
$$\frac{\Gamma\vdash t{:}\mathbb{P}\quad \Delta\vdash u{:}\mathbb{Q}}{\Gamma,\Delta\vdash t{\otimes}u{:}\mathbb{P}{\otimes}\mathbb{Q}} \qquad \frac{\Gamma,x{:}\mathbb{P},y{:}\mathbb{Q}\vdash t{:}\mathbb{R}\quad \Delta\vdash u{:}\mathbb{P}{\otimes}\mathbb{Q}}{\Gamma,\Delta\vdash[u{>}x{\otimes}y{\Rightarrow}t]{:}\mathbb{R}}$$

Recursive type definitions:
$$\frac{\Gamma\vdash t{:}\mathbb{T}_j[\mu\vec{T}.\vec{\mathbb{T}}/\vec{T}]}{\Gamma\vdash abs\ t{:}\mu_j\vec{T}.\vec{\mathbb{T}}} \qquad \frac{\Gamma\vdash t{:}\mu_j\vec{T}.\vec{\mathbb{T}}}{\Gamma\vdash rep\ t{:}\mathbb{T}_j[\mu\vec{T}.\vec{\mathbb{T}}/\vec{T}]}$$

Define $[\![\Gamma]\!]$ to be the event structure consisting of the types in Γ tensored together. We use Γ as shorthand for $[\![\Gamma]\!]$. Similarly, $[\![\mathbb{P}]\!]$ is confused with $\mathbb{P}$.

For clarity, as described in the definition of the tensor construction in Section 2.3, we make the following assumption. In an event structure $A\otimes C$ constructed from event structures A and C, we assume A and C are disjoint and therefore that $A\otimes C$ is given by union and that the projection of a configuration $y\in\mathcal{C}(A\otimes C)$ to its component in C can be written as $y\cap C$.

It is useful to extend the tensor operation to spans. We define

$$(A \xleftarrow{d_1} E_1 \xrightarrow{out_1} B) \ \otimes \ (C \xleftarrow{d_2} E_2 \xrightarrow{out_2} D)$$

by

$$\begin{tikzcd} & E_1 \otimes E_2 \arrow[dl, "d_1 \otimes d_2"'] \arrow[dr, "out_1 \otimes out_2"] & \\ A \otimes C & & B \otimes D. \end{tikzcd}$$

Using the above and previously defined constructions on event structures, we can now give a semantics for Affine HOPLA. Throughout the rest of the chapter, $\Gamma \xleftarrow{d_t} E_t \xrightarrow{out_t} \mathbb{P}$ is taken to be the span $[\![\Gamma \vdash t : \mathbb{P}]\!]$.

Variable: $[\![x : \mathbb{P} \vdash x : \mathbb{P}]\!]$

$$\begin{tikzcd} & \mathbb{P} \arrow[dl, "{[\text{-}]}"'] \arrow[dr, "id_{\mathbb{P}}"] & \\ \mathbb{P} & & \mathbb{P} \end{tikzcd}$$

Other Structural Rules: If the span $[\![\Gamma \vdash t : \mathbb{Q}]\!]$ is $\Gamma \xleftarrow{d} E_t \xrightarrow{out} \mathbb{Q}$ then we define $[\![\Gamma, x : \mathbb{P} \vdash t : \mathbb{Q}]\!]$ to be

$$\begin{tikzcd} & E_t \arrow[dl, "d'"'] \arrow[dr, "out"] & \\ \Gamma \otimes \mathbb{P} & & \mathbb{Q} \end{tikzcd}$$

where d' is defined by $d'(e) = d(e)$ for all $e \in E_t$ (making use of the previously explained assumption that $\mathbb{P}$ and Γ are disjoint).

If the span $[\![\Gamma, y : \mathbb{Q}, x : \mathbb{P} \vdash t : \mathbb{R}]\!]$ is $\Gamma \otimes \mathbb{Q} \otimes \mathbb{P} \xleftarrow{d} E_t \xrightarrow{out} \mathbb{R}$ then we define $[\![\Gamma, x : \mathbb{P}, y : \mathbb{Q} \vdash t : \mathbb{R}]\!]$ to be

$$\begin{tikzcd} & E_t \arrow[dl, "d'"'] \arrow[dr, "out"] & \\ \Gamma \otimes \mathbb{P} \otimes \mathbb{Q} & & \mathbb{R} \end{tikzcd}$$

where d' is defined by $d'(e) = d(e)$ for all $e \in E_t$ (making use of our assumption that Γ, $\mathbb{P}$ and $\mathbb{Q}$ are disjoint).

Recursion: $[\![\Gamma \vdash \mathit{rec}\ x.t : \mathbb{P}]\!]$

We construct this span in terms of the span $[\![\Gamma,\ x : \mathbb{P} \vdash t : \mathbb{P}]\!]$. Let $S = \Gamma \xleftarrow{[\text{-}]} \Gamma \xrightarrow{id_\Gamma} \Gamma$. Define the $(i{+}1)^{\text{th}}$ approximation S_{i+1} to $[\![\Gamma \vdash \mathit{rec}\ x.t : \mathbb{P}]\!]$ to be

$$[\![\Gamma,\ x : \mathbb{P} \vdash t : \mathbb{P}]\!] \text{ composed with } S \otimes S_i.$$

Let S_0 be $\emptyset \xrightarrow{\emptyset} \mathbb{P}$. The span $[\![\Gamma,\ x : \mathbb{P} \vdash t : \mathbb{P}]\!]$ is defined to be S_∞. Observe that $\lambda x.t$ can be viewed as a continuous operation on event structures and therefore the vertex of S_∞ is isomorphic to $\mu x.t$.

Sum: $[\![\Gamma \vdash \sum_{i\in I} t_i : \mathbb{P}]\!]$
If the span $[\![\Gamma \vdash t_i : \mathbb{P}]\!]$ is $\Gamma \overset{d_i}{\leftarrow} E_i \overset{out_i}{\rightarrow} \mathbb{P}$ for all $i \in I$ then the sum is

$$\begin{tikzcd}
& \sum_{i\in I} E_i \arrow[dl, "d"'] \arrow[dr, "out"] & \\
\Gamma & & \mathbb{P}
\end{tikzcd}$$

where $d(j,\ e) \overset{def}{=} d_j(e)$ and $out(j,\ e) \overset{def}{=} out_j(e)$ for all $(j,\ e) \in \sum_{i\in I} E_i$.

Function: $[\![\Gamma \vdash \lambda x.t : \mathbb{P} \multimap \mathbb{Q}]\!]$
If $[\![\Gamma,\ x : \mathbb{P} \vdash t : \mathbb{Q}]\!]$ is $\Gamma \otimes \mathbb{P} \overset{d_t}{\leftarrow} E_t \overset{out_t}{\rightarrow} \mathbb{Q}$ then we define $[\![\Gamma \vdash \lambda x.t : \mathbb{P} \multimap \mathbb{Q}]\!]$ to be

$$\begin{tikzcd}
& E_t \arrow[dl, "d'"'] \arrow[dr, "out'"] & \\
\Gamma & & \mathbb{P} \multimap \mathbb{Q}
\end{tikzcd}$$

where $d'(e)$ is $d_t(e) \cap \Gamma$ and $out'(e)$ is $\{(d_t(e') \cap \mathbb{P},\ out_t(e')) \mid e' \in [e]\}$.

Application: $[\![\Gamma, \Delta \vdash t\ u : \mathbb{P}]\!]$
If $[\![\Gamma \vdash t : \mathbb{Q} \multimap \mathbb{P}]\!]$ is the span $\Gamma \overset{d_t}{\leftarrow} E_t \overset{out_t}{\rightarrow} \mathbb{Q} \multimap \mathbb{P}$ and $[\![\Delta \vdash u : \mathbb{Q}]\!]$ is the span $\Delta \overset{d_u}{\leftarrow} E_u \overset{out_u}{\rightarrow} \mathbb{Q}$ then $[\![\Gamma, \Delta \vdash t\ u : \mathbb{P}]\!]$ is the span

$$\begin{tikzcd}
& & (\Gamma \otimes E_u) \circ E_t \arrow[dl, "\pi_1"'] \arrow[dr, "\pi_2"] & & \\
& (\Gamma \otimes E_u) \arrow[dl, "{[\text{-}]} \otimes d_u"'] \arrow[dr, "id_\Gamma \otimes out_u"] & & E_t \arrow[dl, "d'_t"'] \arrow[dr, "out'_t"] & \\
\Gamma \otimes \Delta & & \Gamma \otimes \mathbb{Q} & & \mathbb{P}.
\end{tikzcd}$$

We define d'_t and out'_t as below, making use of the previously explained assumption that $\mathbb{P}$ and $\mathbb{Q}$ are disjoint. We let e' be $max(out_t(e))$, viewing $out_t(e)$ as a complete prime for the stable family $\mathcal{F}_{\mathbb{Q}\multimap\mathbb{P}}$ (see Sections 2.2.1 and 2.3). Define d'_t by $d'_t(e) = d_t(e) \cup \pi_1(e')$ and define $out'_t(e) = \pi_2(e')$ for all $e \in E_t$.

Injection: $[\![\Gamma \vdash \beta t\ : \sum_{\alpha\in A} \mathbb{P}_\alpha]\!]$
If the span $[\![\Gamma \vdash t : \mathbb{P}_\beta]\!]$ is $\Gamma \overset{d_t}{\leftarrow} E_t \overset{out_t}{\rightarrow} \mathbb{P}_\beta$ then the injection into $\sum_{\alpha\in A} \mathbb{P}_\alpha$ where $\beta \in A$ is

$$\begin{tikzcd}
& E_t \arrow[dl, "d_t"'] \arrow[dr, "out'_t"] & \\
\Gamma & & \sum_{\alpha\in A} \mathbb{P}_\alpha
\end{tikzcd}$$

where $out'_t(e) = (\beta,\ out_t(e))$ for all $E \in E_t$.

Projection: $[\![\Gamma \vdash \pi_\beta t : \mathbb{P}_\beta]\!]$

$$\begin{tikzcd} & E_t \upharpoonright S \arrow[dl, "d_t \upharpoonright S"'] \arrow[dr, "out'_t"] & \\ \Gamma & & \mathbb{P}_\beta \end{tikzcd}$$

where $S = \{e \in E_t \mid \exists e' \in \mathbb{P}_\beta .\ out_t(e) = (\beta,\ e')\}$ and $out'_t(e) = \pi_2(out_t(e))$.

Prefixing: $[\![\Gamma \vdash !u : \mathbb{P}_\perp]\!]$

$$\begin{tikzcd} & (E_u)_\perp \arrow[dl, "{d_u}'"'] \arrow[dr, "out_{u\perp}"] & \\ \Gamma & & \mathbb{P}_\perp \end{tikzcd}$$

We define d'_u by $d'_u(\emptyset) = \emptyset$ and $d'_u([e]) = d_u(e)$. (Recall the definition of lifting in Section 2.3.)

Prefix Match: $[\![\Gamma,\ \Delta \vdash [u\ > !x\ \Rightarrow\ t] : \mathbb{Q}]\!]$
We can construct a span from $\mathbb{P}_\perp$ to $\mathbb{P}$:

$$\begin{tikzcd} & \mathbb{P} \arrow[dl, "{[[\text{-}]]}"'] \arrow[dr, "id_{\mathbb{P}}"] & \\ \mathbb{P}_\perp & & \mathbb{P}. \end{tikzcd}$$

If $[\![\Delta \vdash u : \mathbb{P}_\perp]\!]$ and $[\![\Gamma,\ x : \mathbb{P} \vdash t : \mathbb{Q}]\!]$ are $\Delta \overset{d_u}{\leftarrow} E_u \overset{out_u}{\rightarrow} \mathbb{P}_\perp$ and $\Gamma \otimes \mathbb{P} \overset{d_t}{\leftarrow} E_t \overset{out_t}{\rightarrow} \mathbb{Q}$ then the prefix match is the composition of

$$\begin{tikzcd} & \Gamma \otimes E_u \arrow[dl, "{[-]\otimes d_u}"'] \arrow[dr, "id_\Gamma \otimes out_u"'] & & \Gamma \otimes \mathbb{P} \arrow[dl, "{[-]\otimes[[-]]}"'] \arrow[dr, "id_{\Gamma\otimes\mathbb{P}}"'] & & E_t \arrow[dl, "d_t"'] \arrow[dr, "out_t"'] & \\ \Gamma \otimes \Delta & & \Gamma \otimes \mathbb{P}_\perp & & \Gamma \otimes \mathbb{P} & & \mathbb{Q}. \end{tikzcd}$$

Tensor: $[\![\Gamma,\ \Delta \vdash t \otimes u : \mathbb{P} \otimes \mathbb{Q}]\!]$

$$\begin{tikzcd} & E_t \otimes E_u \arrow[dl, "d_t \otimes d_u"'] \arrow[dr, "out_t \otimes out_u"] & \\ \Gamma \otimes \Delta & & \mathbb{P} \otimes \mathbb{Q} \end{tikzcd}$$

Tensor Match: $[\![\Gamma,\ \Delta \vdash [u\ >\ x \otimes y\ \Rightarrow\ t] : \mathbb{R}]\!]$
If the span $[\![\Delta \vdash u : \mathbb{P} \otimes \mathbb{Q}]\!]$ is

$$\begin{tikzcd} & E_u \arrow[dl, "d_u"'] \arrow[dr, "out_u"] & \\ \Delta & & \mathbb{P} \otimes \mathbb{Q} \end{tikzcd}$$

and $[\![\Gamma,\ x : \mathbb{P},\ y : \mathbb{Q} \vdash t : \mathbb{R}]\!]$ is

$$\begin{tikzcd} & E_t \arrow[dl, "d_t"'] \arrow[dr, "out_t"] & \\ \Gamma \otimes \mathbb{P} \otimes \mathbb{Q} & & \mathbb{R} \end{tikzcd}$$

then $[\![\Gamma,\ \Delta \vdash [u\ >\ x \otimes y\ \Rightarrow\ t] : \mathbb{R}]\!]$ is

$$\begin{tikzcd}
 & & & (\Gamma \otimes E_u) \circ E_t \arrow[dll, "\pi_1"'] \arrow[drr, "\pi_2"] & & & \\
 & (\Gamma \otimes E_u) \arrow[dl, "{[\text{-}]\otimes d_u}"'] \arrow[drr, "id_\Gamma \otimes out_u"] & & & & E_t \arrow[dll, "d_t"'] \arrow[dr, "out_t"] & \\
\Gamma \otimes \Delta & & & \Gamma \otimes \mathbb{P} \otimes \mathbb{Q} & & & \mathbb{R}.
\end{tikzcd}$$

Recursive Types: $[\![\Gamma \vdash abs\ t\ :\ \mu_j \vec{T}.\vec{\mathbb{T}}]\!]$ and $[\![\Gamma \vdash rep\ t : \mathbb{T}_j[\mu\vec{T}.\vec{\mathbb{T}}/\vec{T}]]\!]$
Recall that, from the properties of fixed points, $\mu_j \vec{T}.\vec{\mathbb{T}}$ and $\mathbb{T}_j[\mu\vec{T}.\vec{\mathbb{T}}/\vec{T}]$ are equal. The semantics for *rep* and *abs* is therefore given by

$$\begin{tikzcd}
 & E_t \arrow[dl, "d_t"'] \arrow[dr, "out_t"] & \\
\Gamma & & \mu_j \vec{T}.\vec{\mathbb{T}},
\end{tikzcd}$$

$$\begin{tikzcd}
 & E_t \arrow[dl, "d_t"'] \arrow[dr, "out_t"] & \\
\Gamma & & \mathbb{T}_j[\mu\vec{T}.\vec{\mathbb{T}}/\vec{T}].
\end{tikzcd}$$

When only the first order fragment of Affine HOPLA is considered, the event structure semantics corresponds to the presheaf semantics (see [26]).

3.3 Examples

To give a greater insight into the way spans of event structures can model higher order processes, we discuss the spans representing five simple higher-order Affine HOPLA terms. We also show the results when they are applied to other terms.

Example 1: The empty process — $\lambda y.y : \emptyset \multimap \emptyset$.
Following the construction in Section 3.2, we construct $[\![\lambda y.y\ :\ \emptyset\ \multimap\ \emptyset]\!]$ in terms of $[\![y : \emptyset\ \vdash\ y : \emptyset]\!]$.

$$\begin{tikzcd}
 & \emptyset \arrow[dl, "\emptyset"'] \arrow[dr, "\emptyset"] & \\
\emptyset & & \emptyset
\end{tikzcd}$$

So, $[\![\lambda y.y : \emptyset \multimap \emptyset]\!]$ is $\emptyset \xrightarrow{\emptyset} (\emptyset \multimap \emptyset)$. As $(\emptyset \multimap \emptyset) = \emptyset$, the span is equal to $\emptyset \xrightarrow{\emptyset} \emptyset$. The vertex of this span is not capable of any events and therefore may be thought of as the nil process.

We next show the constructors for prefixing and asynchronous parallel composition.

Example 2: Prefixing a process — $\lambda x.!x : \mathbb{P} \multimap \mathbb{P}_\perp$.
The span $[\![\lambda x.!x : \mathbb{P} \multimap \mathbb{P}_\perp]\!]$ is constructed from the span $[\![x : \mathbb{P} \vdash !x : \mathbb{P}_\perp]\!]$ which is itself constructed from $[\![x : \mathbb{P} \vdash x : \mathbb{P}]\!]$.

$$\begin{tikzcd} & \mathbb{P} \arrow[dl, "{[-]}"'] \arrow[dr, "id"] & \\ \mathbb{P} & & \mathbb{P} \end{tikzcd}$$

So $[\![x : \mathbb{P} \vdash !x : \mathbb{P}_\perp]\!]$ is

$$\begin{tikzcd} & \mathbb{P}_\perp \arrow[dl, "d"'] \arrow[dr, "id_\perp"] & \\ \mathbb{P} & & \mathbb{P}_\perp \end{tikzcd}$$

where d is defined by $d(e) = e$ for all $e \in \mathbb{P}_\perp$.

The span $[\![\lambda x.!x : \mathbb{P} \multimap \mathbb{P}_\perp]\!]$ is therefore

$$\begin{tikzcd} \mathbb{P}_\perp \arrow[d, "out"] \\ \mathbb{P} \multimap \mathbb{P}_\perp \end{tikzcd}$$

where $out : \mathbb{P}_\perp \to (\mathbb{P} \multimap \mathbb{P}_\perp)$ is defined by $out(e) = \{(e',\ id_\perp(e')) \mid e' \in [e]\}$ for all $e \in \mathbb{P}_\perp$. (The vertex of the span has no demands for any of it's events.)

Below we show an example of the application of $\lambda x.!x : \mathbb{P} \multimap \mathbb{P}_\perp$. Let $[\![t : \mathbb{P}]\!]$ be the span $E_t \xrightarrow{out_t} \mathbb{P}$ where E_t consists of two independent events e_1 and e_2. Then $[\![(\lambda x.!x)\ t : \mathbb{P}_\perp]\!]$ will be the composition

$$\begin{tikzcd} & E \arrow[dl, "\pi_1"'] \arrow[dr, "\pi_2"] & & \\ E_t \arrow[dr, "out_t"] & & \mathbb{P}_\perp \arrow[dl, "d"'] \arrow[dr, "id"] & \\ & \mathbb{P} & & \mathbb{P}_\perp . \end{tikzcd}$$

The event structure E will be

$$\begin{tikzcd} (\{e_1\},\ [out_t(e_1)]) & & (\{e_2\},\ [out_t(e_2)]) \\ & (\emptyset,\ \emptyset) \arrow[ul] \arrow[ur] & . \end{tikzcd}$$

So, the vertex of the span E can perform an event and then behave like the vertex of the span E_t. Thus we have demonstrated the way in which our function takes a process and prefixes it with an extra event that must occur before all the others.

Example 3: Asynchronous parallel composition — $\lambda x.(x \otimes t) : \mathbb{P} \multimap (\mathbb{P} \otimes \mathbb{Q})$.
Let $[\![t : \mathbb{Q}]\!]$ be $E_t \xrightarrow{out_t} \mathbb{Q}$. Assume that $\mathbb{P}$ and E_t are disjoint and therefore that the action of $\otimes$ on $\mathbb{P}$ and E_t can be defined as union. We construct $[\![\lambda x.(x \otimes t) : \mathbb{P} \multimap (\mathbb{P} \otimes \mathbb{Q})]\!]$ from $[\![x : \mathbb{P} \vdash x \otimes t : \mathbb{P} \otimes \mathbb{Q}]\!]$.

$$\begin{tikzcd}
 & \mathbb{P} \otimes E_t \arrow[dl, "{[\_]\otimes(\_\mapsto\emptyset)}"'] \arrow[dr, "id\otimes out_t"] & \\
\mathbb{P} & & \mathbb{P} \otimes \mathbb{Q}
\end{tikzcd}$$

So $[\![\lambda x.(x \otimes t) : \mathbb{P} \multimap (\mathbb{P} \otimes \mathbb{Q})]\!]$ is

$$\begin{tikzcd}
\mathbb{P} \otimes E_t \arrow[d, "out"] \\
\mathbb{P} \multimap (\mathbb{P} \otimes \mathbb{Q})
\end{tikzcd}$$

where *out* is defined by

$$\begin{aligned} out(e) &= \{([e'],\ e') \mid e' \in [e]\} \quad \text{if } e \in \mathbb{P} \\ &= \{(\emptyset,\ out_t(e')) \mid e' \in [e]\} \quad \text{otherwise.} \end{aligned}$$

If we apply this function to a process $s : \mathbb{P}$ with corresponding span $E_s \xrightarrow{out_s} \mathbb{P}$ then the resulting process is represented by the composition

$$\begin{tikzcd}
 & E \arrow[dl, "\pi_1"'] \arrow[dr, "\pi_2"] & & \\
E_s \arrow[dr, "out_s"'] & & \mathbb{P} \otimes E_t \arrow[dl, "{[\_]\otimes(\_\mapsto\emptyset)}"'] \arrow[dr, "id\otimes out_t"] & \\
 & \mathbb{P} & & \mathbb{P} \otimes \mathbb{Q}.
\end{tikzcd}$$

The events of E will either be of the form $(\emptyset,\ e)$ where $e \in E_t$ or $([e'],\ out_s(e))$. It is easy to show that E is isomorphic to $E_s \otimes E_t$ and indeed that the span E is isomorphic to

$$\begin{tikzcd}
E_s \otimes E_t \arrow[d, "out_s\otimes out_t"] \\
\mathbb{P} \otimes \mathbb{Q}.
\end{tikzcd}$$

So the function $\lambda x.(x \otimes t) : \mathbb{P} \multimap (\mathbb{P} \otimes \mathbb{Q})$ has the action of placing a process in parallel with the process $t : \mathbb{Q}$.

We complete this section by showing examples of the use of the destructors for prefixing and asynchronous parallel composition.

Example 4: Matching a prefixed event — $\lambda y.[y >!x => x] : \mathbb{P}_\perp \multimap \mathbb{P}$.
We construct $\llbracket \lambda y.[y >!x => x] : \mathbb{P}_\perp \multimap \mathbb{P} \rrbracket$ from $\llbracket y : \mathbb{P}_\perp \ \vdash \ [y >!x => x] : \mathbb{P} \rrbracket$, i.e., the composition of

$$\begin{tikzcd}
 & \mathbb{P}_\perp \arrow[dl, "{[\text{-}]}"'] \arrow[dr, "id_{\mathbb{P}_\perp}"] & & \mathbb{P} \arrow[dl, "{[[\text{-}]]}"'] \arrow[dr, "id_{\mathbb{P}}"] & & \mathbb{P} \arrow[dl, "{[\text{-}]}"'] \arrow[dr, "id_{\mathbb{P}}"] & . \\
\mathbb{P}_\perp & & \mathbb{P}_\perp & & \mathbb{P} & & \mathbb{P}
\end{tikzcd}$$

However, $\mathbb{P} \xleftarrow{[\text{-}]} \mathbb{P} \xrightarrow{id_{\mathbb{P}}} \mathbb{P}$ is the identity span for $\mathbb{P}$ and $\mathbb{P}_\perp \xleftarrow{[\text{-}]} \mathbb{P}_\perp \xrightarrow{id_{\mathbb{P}_\perp}} \mathbb{P}_\perp$ is the identity span for $\mathbb{P}_\perp$ and so $\llbracket y : \mathbb{P}_\perp \ \vdash \ [y >!x => x] : \mathbb{P} \rrbracket$ is isomorphic to $\mathbb{P}_\perp \xleftarrow{[[\text{-}]]} \mathbb{P} \xrightarrow{id_{\mathbb{P}}} \mathbb{P}$. The span $\llbracket \lambda y.[y >!x => x] : \mathbb{P}_\perp \multimap \mathbb{P} \rrbracket$ is therefore isomorphic to

$$\begin{tikzcd}
\mathbb{P} \arrow[d, "out"] \\
\mathbb{P}_\perp \multimap \mathbb{P}
\end{tikzcd}$$

where $out : \mathbb{P} \to (\mathbb{P}_\perp \multimap \mathbb{P})$ is defined by $out(e) = \{([[e']],\ e') \mid e' \in [e]\}$ for all $e \in \mathbb{P}$.

Let us consider the application $(\lambda y.[y >!x => x])t : \mathbb{P}$ where $\llbracket t : \mathbb{P}_\perp \rrbracket$ is $E_t \xrightarrow{out_t} \mathbb{P}_\perp$. This will be isomorphic to the composition

$$\begin{tikzcd}
 & E \arrow[dl, "\pi_1"'] \arrow[dr, "\pi_2"] & & \\
E_t \arrow[dr, "out_t"] & & \mathbb{P} \arrow[dl, "{[[\text{-}]]}"'] \arrow[dr, "id_{\mathbb{P}}"] & \\
 & \mathbb{P}_\perp & & \mathbb{P}.
\end{tikzcd}$$

The events of E will be of the form $([e],\ max(out_t(e)))$ such that $e \in E_t$ and $out_t(e) \neq \emptyset$. For events $([e_1],\ max(out_t(e_1)))$ and $([e_2],\ max(out_t(e_2)))$ in E, we have

$$\begin{aligned} &([e_1],\ max(out_t(e_1))) \leq ([e_2],\ max(out_t(e_2))) \text{ iff} \\ &\quad [e_1] \subseteq [e_2] \text{ and } max(out_t(e_1)) \leq max(out_t(e_2)). \end{aligned}$$

There is a bijective correspondence between the prime configurations of an event structure and its events. Also $[e_1] \subseteq [e_2]$ iff $e_1 \leq e_2$. A similar argument shows the relationship between the conflict relation of E_t and that of E. It follows that E behaves exactly as E_t would after the occurrence of its minimal (prefixed) event. So, our function has been shown to remove the minimal event of a process it is applied to.

Example 5: Tensor matching — $\lambda y.[y > w \otimes z => w] : (\mathbb{P} \otimes \mathbb{Q}) \multimap \mathbb{P}$.
We construct $\llbracket \lambda y.[y > w \otimes z => w] : (\mathbb{P} \otimes \mathbb{Q}) \multimap \mathbb{P} \rrbracket$ from the span representing

$y : \mathbb{P}\otimes\mathbb{Q} \;\vdash\; [y > w \otimes z => w] : \mathbb{P}$.

$$\begin{tikzcd}
& & E \arrow[dl, "\pi_1"'] \arrow[dr, "\pi_2"] & & \\
& \mathbb{P}\otimes\mathbb{Q} \arrow[dl, "{[\text{-}]}"'] \arrow[dr, "id_{\mathbb{P}\otimes\mathbb{Q}}"] & & \mathbb{P} \arrow[dl, "{[\text{-}]}"'] \arrow[dr, "id_{\mathbb{P}}"] & \\
\mathbb{P}\otimes\mathbb{Q} & & \mathbb{P}\otimes\mathbb{Q} & & \mathbb{P}
\end{tikzcd}$$

(We again make use of the assumption that $\mathbb{P}$ and $\mathbb{Q}$ are disjoint and therefore that $\otimes$ can be defined in terms of union.)

As $\mathbb{P}\otimes\mathbb{Q} \xleftarrow{[\text{-}]} \mathbb{P}\otimes\mathbb{Q} \xrightarrow{id_{\mathbb{P}\otimes\mathbb{Q}}} \mathbb{P}\otimes\mathbb{Q}$ is the identity span for $\mathbb{P}\otimes\mathbb{Q}$, the span E will be isomorphic to

$$\begin{tikzcd}
& \mathbb{P} \arrow[dl, "{[\text{-}]}"'] \arrow[dr, "id_{\mathbb{P}}"] & \\
\mathbb{P}\otimes\mathbb{Q} & & \mathbb{P}.
\end{tikzcd}$$

So $\llbracket \lambda y.[y > w \otimes z => w] : (\mathbb{P}\otimes\mathbb{Q}) \multimap \mathbb{P} \rrbracket$ is isomorphic to the span

$$\begin{tikzcd}
\mathbb{P} \arrow[d, "out"] \\
(\mathbb{P}\otimes\mathbb{Q}) \multimap \mathbb{P}
\end{tikzcd}$$

where $out : \mathbb{P} \to (\mathbb{P}\otimes\mathbb{Q}) \multimap \mathbb{P}$ is defined by $out(e) = \{([e'],\ e') \mid e' \in [e]\}$ for all $e \in \mathbb{P}$.

We now show that the application of this function to a term $t : \mathbb{P}\otimes\mathbb{Q}$ acts to extract the part of t that behaves like a process of type $\mathbb{P}$ and discards the rest.

Let $\llbracket t : \mathbb{P}\otimes\mathbb{Q} \rrbracket$ be $E_t \xrightarrow{out_t} \mathbb{P}\otimes\mathbb{Q}$. Then $\llbracket (\lambda y.[y > w\otimes z => w])t : \mathbb{P} \rrbracket$ is the composition

$$\begin{tikzcd}
& E \arrow[dl, "\pi_1"'] \arrow[dr, "\pi_2"] & & \\
E_t \arrow[dr, "out_t"] & & \mathbb{P} \arrow[dl, "{[\text{-}]}"'] \arrow[dr, "id_{\mathbb{P}}"] & \\
& \mathbb{P}\otimes\mathbb{Q} & & \mathbb{P}.
\end{tikzcd}$$

The events in E will be of the form $([e],\ out_t(e))$ for $e \in E_t$ such that $out_t(e) \in \mathbb{P}$. It is clear that E is isomorphic to the restriction of E_t to those events that are mapped into $\mathbb{P}$ by out_t, i.e., E is isomorphic to $E \upharpoonright S$ where $S \overset{def}{=} \{e' \in E_t \mid out_t(e') \in \mathbb{P}\}$. Indeed, the span E is isomorphic to the span

$$\begin{tikzcd}
E_t \upharpoonright S \arrow[d, "out'_t"] \\
\mathbb{P}
\end{tikzcd}$$

where out'_t is the restriction of out_t.

3.4 Concluding Remarks

In this chapter, by recalling the work of Nygaard and Winskel, we have demonstrated event structures as both types and processes with the aim of helping the reader to become familiar with event structures and their spans. The event structure semantics for Affine HOPLA illustrates the way in which spans of event structures are an operational and intuitive model of higher order processes. We have shown that they support many useful constructions and, via our examples, given an intuition of the behaviours they capture.

Chapter 4

Event Structure Spans for Non-Deterministic Dataflow

Recall that, in order to model dataflow, we require a feedback operation. In more recent years feedback of the kind found in dataflow has reappeared in a variety of new contexts, which are condensed in a more abstract and general formulation of the properties a feedback operation, called *trace*, should satisfy. Let $\mathcal{C}$ be a category with a symmetric monoidal structure, $\otimes$. A trace operation for $\mathcal{C}$ is defined to be a family of operations,

$$Tr^{C}_{A,B} : \mathcal{C}(A \otimes C,\ B \otimes C) \rightarrow \mathcal{C}(A,\ B)$$

satisfying a number of axioms.

i) *Bekic*: If $f : A \otimes U \otimes V \rightarrow B \otimes U \otimes V$ is a morphism in $\mathcal{C}$ then

$$Tr^{U\otimes V}_{A,B}(f) = Tr^{U}_{A,B}(Tr^{V}_{A\otimes U,B\otimes U}(f)).$$

ii) *Yanking*: Define $\sigma_{A,B} : A \otimes B \rightarrow B \otimes A$ to be the isomorphism between $A \otimes B$ and $B \otimes A$ for $A,\ B$ in $\mathcal{C}$. Then, if U is an object in $\mathcal{C}$,

$$Tr^{U}_{U,U}(\sigma_{U,U}) = id_U$$

where $id_U : U \rightarrow U$ is the identity morphism of U.

iii) *Superposing*: If $f : A \otimes U \rightarrow B \otimes U$ and $g : V \rightarrow W$ are morphisms in $\mathcal{C}$

$$Tr^{U}_{A\otimes V,B\otimes W}((id_B \otimes \sigma_{U,W}) \circ (f \otimes g) \circ (id_A \otimes \sigma_{V,U})) = Tr^{U}_{A,B}(f) \otimes g.$$

iv) *Naturality I*: If $f : C \otimes U \rightarrow B \otimes U$ and $g : A \rightarrow C$ are morphisms in $\mathcal{C}$

$$Tr^{U}_{A,B}(f \circ (g \otimes id_U)) = Tr^{U}_{C,B}(f) \circ g.$$

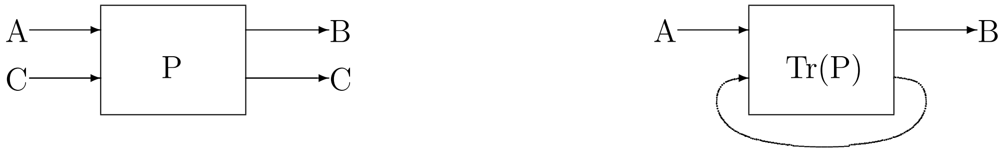
v) *Naturality II*: If $f : A \otimes U \rightarrow C \otimes U$ and $g : C \rightarrow B$ are morphisms in $\mathcal{C}$

$$Tr^{U}_{A,B}((g \otimes id_U) \circ f) = g \circ Tr^{U}_{A,C}(f).$$

vi) *Dinaturality*: If $f : A \otimes U \to B \otimes V$ and $g : V \to U$ are morphisms in $\mathcal{C}$

$$Tr^U_{A,B}((id_B \otimes g) \circ f) = Tr^V_{A,B}(f \circ (id_A \otimes g)).$$

The intuition behind this operation is illustrated in the following diagram.



The reader is referred to the work of Verity, Joyal and Street [18] for more information about the trace construction and to [15, 16] for a fuller set of references to this rich area.

In this chapter, which covers much of the same material as [30], we show that the bicategory of spans of event structures supports a trace construction. Event structure spans can therefore be used to model non-deterministic dataflow processes. Having defined the trace operation, we argue that it possesses the correct properties. This is done by relating spans of event structures to certain kinds of profunctor and showing that the definition of trace for profunctors in [15, 16] corresponds to that for event structures. Finally, we define *deterministic spans*, show that they correspond to continuous functions and that, for these spans, the trace operation corresponds to the fixed point construction given by Kahn in [19].

4.1 Trace for Event Structures

This section is devoted to defining a trace operation $Tr^C_{A,B}$ which takes a span from $A \otimes C$ to $B \otimes C$ to a span from A to B. Consider a span

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

We first build a stable family out of those configurations of E which are *secure*. Roughly, a configuration x is secure if the demand $d(e)$ of each event $e \in x$ is either met by the input in A, or by previous output to C, which we imagine is fed back as input. This idea is formalised below.

In defining the trace it is convenient, as in previous sections, to assume that both pairs of sets A, C, and B, C are disjoint, so that we can treat the parallel compositions $A \otimes C$ and $B \otimes C$ as given by union and write $y \cap C$ for the 'projection' of a configuration $y \in \mathcal{C}(A \otimes C)$ to its component in the event structure C.

Definition 4.1.1. *Let x be a configuration of E. A* securing sequence *in x with respect to the span $A \otimes C \overset{d}{\leftarrow} E \overset{out}{\rightarrow} B \otimes C$ consists of a sequence of events $e_1, \cdots, e_n$ in x such that*

$$\{e_1, \cdots, e_i\} \in \mathcal{C}(E) \quad \text{and} \quad d(e_i) \cap C \subseteq out\{e_1, \cdots, e_{i-1}\}$$

for all $i \leq n$. An event e is secured *in x iff there is a securing sequence $e_1, \cdots, e_n$ in x with $e_n = e$. Finally, the configuration x is* secure *if each of its events is* secured *in x.*

The subset of $\mathcal{C}(E)$ consisting of all the secure configurations will be shown to be a stable family, from which we then obtain an event structure. Our proofs will make use of a relation $\prec_x$ expressing the extra causal dependency on a configuration x of E which is introduced through feedback.

Definition 4.1.2. *Let x be a configuration of E. For all events e_1 and e_2 in x, define*

$$\begin{aligned} e_1 \rightarrow_x e_2 \quad &\text{iff} \quad out(e_1) \in d(e_2) \cap C, \text{ and} \\ e_1 \prec_x e_2 \quad &\text{iff} \quad e_1 < e_2 \text{ or } e_1 \rightarrow_x e_2. \end{aligned}$$

We define $\{e\}_x \subseteq x$ to be the set $\{e' \in x \mid e' \prec_x{}^\star e\}$ for all $e \in x$. (Note that because $\{e\}_x$ must be a downwards closed subset of x with respect to causal dependency it is necessarily a configuration.)

Lemma 4.1.3. *An event e is secured in configuration x iff*

i) $\prec_x$ *is well-founded on* $\{e\}_x$, *and*

ii) $(d^\dagger(\{e\}_x)) \cap C \subseteq out(x)$.

Proof. if: Assume (i) and (ii). To prove that event e is secured in x, we require a securing sequence for e. First note that the set $\{e\}_x$ is finite as $\prec_x$ is well-founded and the set of $\prec_x$-predecessors of an event is also finite. Thus, tentatively we may take the securing sequence to be a choice of order $\{e_1, \cdots, e_n\}$ of the set $\{e\}_x$ which respects $\prec_x$, *i.e.*, for $0 < i,\ j \leq n$,

$$(e_i \prec_x e_j \Rightarrow i < j) \text{ and } e_n = e\ .$$

We now check that the chosen sequence is indeed securing. Observe that the set $\{e_1, \cdots, e_i\}$, with $i \leq n$, is a configuration of E because it is a downwards closed subset of x – this follows immediately from the definition of $\prec_x$. It remains to confirm that $d(e_i) \cap C \subseteq out\{e_1, \cdots, e_{i-1}\}$, for all $i \leq n$. Consider an event $c \in d(e_i) \cap C$ for some $i \leq n$. As $d^\dagger(\{e\}_x) \cap C \subseteq out\, x$, we have that $c = out(e')$ for some $e' \in x$. By definition, $e' \rightarrow_x e_i$, so $e' \prec_x e_i$. Thus $e' \in \{e\}_x$ and, as the sequence respects $\prec_x$, we see that $e' = e_j$ for some $j < i$. This confirms that $c \in out\{e_1, \cdots, e_{i-1}\}$. Hence $\{e_1, \cdots, e_n\}$ is a securing sequence.

only if: Assume that $e_1, \cdots, e_n = e$ is a securing sequence in x.

We first show that for $i \leq n$

$$e' \prec_x e_i \Rightarrow e' \in \{e_1, \cdots, e_{i-1}\} \tag{$\dagger$}$$

and therefore that $e' = e_j$ for some $j < i$.

By definition, if $e' \prec_x e_i$ then either $e' < e_i$ or $e' \rightarrow_x e_i$. As $\{e_1, \cdots, e_i\}$ is a configuration and therefore downwards closed with respect to $<$, if $e' < e_i$ then $e' \in \{e_1, \cdots, e_{i-1}\}$.

If $e' \rightarrow_x e_i$ then $out(e') \in d(e_i) \cap C$. But we have $d(e_i) \cap C \subseteq out\{e_1, \cdots, e_{i-1}\}$ by the property of a securing sequence. So $out(e') = out(e_j)$ where $e_j \in \{e_1, \cdots, e_{i-1}\}$. As both e' and e_j are in x and out is a rigid morphism, we have $e' = e_j$ and so $e' \in \{e_1, \cdots, e_{i-1}\}$.

Now we can show (i) and (ii).

(i) By (†), a $\prec_x$-descending chain in $\{e\}_x$ induces a strictly descending chain of finite sets under inclusion. This chain is certainly well-founded. Hence $\prec_x$ is well-founded on $\{e\}_x$.

(ii) From (†) we have $\{e\}_x \subseteq \{e_1, \cdots, e_n\}$, so $d^\dagger(\{e\}_x) \subseteq d^\dagger(\{e_1, \cdots, e_n\})$. Thus

$$\begin{aligned}
(d^\dagger(\{e\}_x)) \cap C &\subseteq (d^\dagger(\{e_1, \cdots, e_n\}) \cap C \\
&\subseteq (\bigcup_{i \leq n} d(e_i)) \cap C \\
&\subseteq \bigcup_{i \leq n} (d(e_i) \cap C) \\
&\subseteq \bigcup_{i \leq n} out\{e_1, \cdots, e_{i-1}\} \quad \text{as } e_1,\ldots,e_n \text{ is a securing sequence,} \\
&\subseteq out(x) \ .
\end{aligned}$$

□

Corollary 4.1.4. *A configuration x is secure iff*

i) $\prec_x$ is well-founded on x, and

ii) $d^\dagger(x) \cap C \subseteq out(x)$.

Proof. if: Assume that a configuration x satisfies (i) and (ii) above. Let $e \in x$. Then certainly $\prec_x$ is well-founded on $\{e\}_x$ and

$$(d^\dagger(\{e\}_x)) \cap C \subseteq d^\dagger(x) \cap C \subseteq out(x).$$

Thus e is secured in x by Lemma 4.1.3. But e was an arbitrary event in x. Hence x is secure.

only if: Assume the configuration x is secure. By Lemma 4.1.3, $\prec_x$ is well-founded on x, *i.e.*, (i) . To show (ii):

$$\begin{aligned}
d^\dagger(x) \cap C &= (\bigcup_{e \in x} d^\dagger(\{e\}_x)) \cap C \\
&= \bigcup_{e \in x} (d^\dagger(\{e\}_x) \cap C) \subseteq out(x) \ , \qquad \text{by Lemma 4.1.3.}
\end{aligned}$$

□

In proving that the family of secure configurations forms a stable family, we will make use of the following lemma.

Lemma 4.1.5. *Suppose x and y are secure configurations of E with $x \uparrow y$. Let $e \in x \cap y$. Then,* *i)* $e' \prec_x e$ *iff* $e' \prec_y e$*, for all events* e'*, and*

ii) $\{e\}_x = \{e\}_y$.

If x is a secure configuration and $e \in x$, then $\{e\}_x$ is the least secure sub-configuration of x containing e.

Proof. (i) Assume $e \in x \cap y$ where $x \uparrow y$. Suppose $e' \prec_x e$. Then either $e' < e$ or $e' \rightarrow_x e$. In the former case, $e \in y$ and $e' \prec_y e$. In the latter case, $\mathit{out}(e') \in d(e) \cap C$ with $e' \in x$. As y is secure,

$$d(e) \cap C \subseteq (d^\dagger(y)) \cap C \subseteq \mathit{out}(y) \ .$$

Thus $\mathit{out}(e') = \mathit{out}(e'')$ with $e' \in x$ and some $e'' \in y$. But $x \uparrow y$ and out is a rigid morphism, so $e' = e''$ making $e' \in y$. It follows that $e' \rightarrow_y e$, and $e' \prec_y e$. In either case, $e' \prec_x e$ implies $e' \prec_y e$. The same argument proves the converse implication, establishing (i).

(ii) Obvious from (i).

Assume $e \in x$ where x is a secure configuration. From Corollary 4.1.4, to establish that $\{e\}_x$ is secure it suffices to check (i) that $\prec_{\{e\}_x}$ is well-founded on $\{e\}_x$ and (ii) that $(d^\dagger(\{e\}_x)) \cap C \subseteq \mathit{out}\,\{e\}_x$. Now (i) follows directly because $\prec_x$, which includes $\prec_{\{e\}_x}$, is well-founded on $\{e\}_x$ by Lemma 4.1.3. We now concentrate on showing (ii). By Lemma 4.1.3, as x is secure, we have that $(d^\dagger(\{e\}_x)) \cap C \subseteq \mathit{out}(x)$. Thus any event of $(d^\dagger(\{e\}_x)) \cap C$ takes the form $\mathit{out}(e')$ for some $e' \in x$. But supposing $\mathit{out}(e') \in (d^\dagger(\{e\}_x)) \cap C$, then $\mathit{out}(e') \in (d(e'')) \cap C$, *i.e.,* $e' \rightarrow_x e''$, for some $e'' \in \{e\}_x$. Consequently, $e' \in \{e\}_x$. Hence $(d^\dagger(\{e\}_x)) \cap C \subseteq \mathit{out}\,\{e\}_x$.

Suppose $e \in y \subseteq x$ where y is a secure configuration. Then certainly $x \uparrow y$, so $\{e\}_x = \{e\}_y \subseteq y$. Thus, $\{e\}_x$ is the least secure sub-configuration of x containing e. □

Theorem 4.1.6. *The family consisting of all secure configurations of E is a stable family. For any $e \in x$, a secure configuration, $[e]_x = \{e\}_x$.*

Proof. Let $\mathcal{S} = \{x \in \mathcal{C}(E) \mid x \text{ is secure}\}$. We show $\mathcal{S}$ is a stable family.

Coherence: $\forall X \subseteq \mathcal{S}.\ (\forall x,\ y \in X.\ x \uparrow y) \Rightarrow \bigcup X \in \mathcal{S}$.
Assume X is a pairwise compatible subset of $\mathcal{S}$. It is clear that $\bigcup X$ is a configuration of E. If $e \in \bigcup X$ then $e \in x$ for some x in X. As e is secured in x and $x \subseteq \bigcup X$, there is a securing sequence for e in $\bigcup X$.

Stability: $\forall X \subseteq \mathcal{S}.\ X\!\uparrow\ \Rightarrow\ \bigcap X \in \mathcal{S}$.
Suppose $X \subseteq \mathcal{S}$ and $X \uparrow$. Then there exists a secure configuration y of E for which $\forall x \in X.\ x \subseteq y$ holds. For such a y, consider the set

$$Y \overset{def}{=} \{\{e\}_y \mid e \in \bigcap X\ \} \ .$$

It consists of secure configurations which are certainly pairwise compatible, with upper bound y. By coherence, $\bigcup Y$ is a secure configuration. Clearly, $\bigcap X \subseteq \bigcup Y$ as each $e \in \bigcap X$ is a member of $\{e\}_y$. For any $e \in \bigcap X$ and $x \in X$, then $\{e\}_y = \{e\}_x \subseteq x$, by Lemma 4.1.5. So we also have the reverse inclusion $\bigcup Y \subseteq \bigcap X$ ensuring the equality $\bigcap X = \bigcup Y$, and that $\bigcap X$ is a secure configuration.

Finiteness: $\forall x \in \mathcal{S} \forall e \in x \exists y \in \mathcal{S}.\ y \subseteq x\ \ \&\ e \in y\ \&\ |y| < \infty$.
If x is secure then each event e in x must have a securing sequence. This determines a finite secure configuration in $\mathcal{S}$ which contains e.

Coincidence-freeness: $\forall x \in \mathcal{S},\ e_1,\ e_2 \in x.\ e_1 \neq e_2 \Rightarrow$

$$\exists y \in \mathcal{S}.\ y \subseteq x\ \&\ ((e_1 \in y)\ \&\ (e_2 \notin y)) \text{ or } ((e_2 \in y)\ \&\ (e_1 \notin y)).$$

Assume $e_1, e_2 \in x \in \mathcal{S}$ and $e_1 \neq e_2$. Consider the secure configurations $\{e_1\}_x$ and $\{e_2\}_x$. If e_2 is a member of $\{e_1\}_x$ then $e_2 \prec_x^+ e_1$. As x is secure, it cannot be the case that $e_1 \prec_x^+ e_2$ – otherwise we would contradict the well-foundedness of $\prec_x$. Therefore $e_1 \notin \{e_2\}_x$ if $e_2 \in \{e_1\}_x$ and *vice versa.*

Finally, let $e \in x$, a secure configuration. Both $[e]_x$ (by definition) and $\{e\}_x$ (by Lemma 4.1.5) are the smallest secure sub-configurations of x which contain e, and hence equal. □

We can now define the trace operation.

Definition 4.1.7. *We define* $Tr(A \otimes C \overset{d}{\leftarrow} E \overset{out}{\rightarrow} B \otimes C)$ *to be* $A \overset{d'}{\leftarrow} P \overset{out'}{\rightarrow} B$ *where* P *is the event structure with*

- Events, *the prime configurations* p *of* $\mathcal{S}$*, the stable family of secure configurations of* E*, for which* $out(max(p)) \in B$*, with*
- Causal dependency, $p_1 \leq p_2$ *iff* $p_1 \subseteq p_2$*, and*
- Conflict, $p_1 \# p_2$ *iff* $p_1 \not\uparrow p_2$ *in* $\mathcal{S}$*,*

and where we define $d'(p) \overset{def}{=} d^{\dagger}(p) \cap A$ *and* $out'(p) \overset{def}{=} out(max(p))$ *for* $p \in P$*.*

We should check that the definition of trace does indeed yield a span. In order to show that out' is rigid we observe the following.

Lemma 4.1.8. *For* e_1 *and* e_2 *in* $x \in \mathcal{C}(E)$*, if* $e_1 {\prec_x}^{\star} e_2$ *and* $e_1 \not\leq e_2$ *then* $out(e_1) \in C$*.*

Proof. By a simple induction on the length of chain $e_1 \prec_x \cdots \prec_x e_2$.
Basis: Suppose that $e_1 \prec_x e_2$. In this case $e_1 \not\leq e_2$ implies that $e_1 \rightarrow_x e_2$, so $out(e_1) \in C$.
Induction step: Suppose that the property holds for all chains of length less than or equal to k, *i.e.*, assume that if $e_1 {\prec_x}^k e_2$ and $e_1 \not\leq e_2$ then $out(e_1) \in C$. Suppose $e_1 {\prec_x}^{k+1} e_2$ and $e_1 \not\leq e_2$. Then there exists an e' with $e_1 {\prec_x}^k e'$ and $e' \prec_x e_2$. If $e_1 \not\leq e'$ then, by the induction hypothesis, $out(e_1) \in C$. If $e_1 \leq e'$ then, as $e_1 \not\leq e_2$, we have $e' \not\leq e_2$ and so $out(e') \in C$. But then $out(e_1) \in C$ as out is a rigid morphism. □

Lemma 4.1.9. *The map out' is a rigid morphism:*

Proof. We first show that out' preserves causal dependency. Suppose $p' \leq p$ in P. Write $max(p') = e'$ and $max(p) = e$. Then, $e' \prec_p e$ as $[e]_p = \{e\}_p$. Now, by Lemma 4.1.8, $e' \leq e$ as $out(e') \in B$. Thus $out(e') \leq out(e)$, as out is rigid, which directly yields $out'(p') \leq out'(p)$.

Now let X be a configuration of P. First observe that

$$out'\, X = B \cap out(\bigcup X)\ .$$

Clearly we have the inclusion $out'(X) \subseteq B \cap out \bigcup X$. To show the reverse inclusion, suppose $e \in \bigcup X$ and $out(e) \in B$. Then, $e \in p$ for some $p \in X$. Write $p' \overset{def}{=} [e]_p$. Because $p' \leq p$ and X is downwards closed, $p' \in X$, with $out'(p') = out(e)$ – as required.

As $out'(X) \in \mathcal{C}(B)$ for all $X \in \mathcal{C}(P)$ we have $out'[p] \subseteq [out'(p)]$ for all $p \in P$. So, as causality is preserved, we have $out'[p] = [out'(p)]$.

The preservation of configurations by out' also implies that it must reflect conflict.

Suppose $p, p' \in X$ and $out'(p) = out'(p')$. The union $\bigcup X$ is a secure configuration, compatible with both p and p'. Thus $p = [e]_{\bigcup X}$ and $p' = [e']_{\bigcup X}$ for some $e, e' \in \bigcup X$. From the definition of out' it follows that $out(e) = out(e')$. Because out is a rigid morphism sending both $e, e' \in \bigcup X$ to a common event we must have $e = e'$. Hence $p = p'$. □

Theorem 4.1.10. *The trace $Tr(A \otimes C \overset{d}{\leftarrow} E \overset{out}{\rightarrow} B \otimes C)$ is a span.*

Proof. In the definition of the trace $Tr(E)$ as the span $A \overset{d'}{\leftarrow} P \overset{out'}{\rightarrow} B$ it is easily seen that P is an event structure as it is formed from $\mathcal{S}$ as described in Section 2.2.3. Lemma 4.1.9 confirms that out' is indeed a rigid morphism. That d' is a demand morphism follows directly from its definition. □

In [15, 16], a trace operation was given for *stable rooted port profunctors* and shown to obey the trace axioms. In the next section we will show that spans of event structures represent certain stable, rooted profunctors and that the representation respects both sequential and parallel composition as well as the trace operation. It follows that, for the limited case where the spans represent port profunctors, the trace axioms hold up to isomorphism for our definition of trace. In fact it is believed that the definition of trace for profunctors obeys the axioms for all stable rooted profunctors. If this is indeed proved to be the case, the results in the next section are sufficient to show that the axioms hold for our definition of trace in general.

4.2 Spans and Rooted Profunctors

In this section, we show how spans of event structures correspond to certain profunctors and that the trace operation defined in the previous section coincides with the trace defined on stable profunctors in [15, 16]. In this case, the profunctors will be between partial orders of finite configurations of event structures (regarded as categories). We shall write $\mathcal{C}^0(E)$ for the partial order of finite configurations of an event structure E under inclusion.

It is helpful to give an alternative characterisation of rigid morphisms between event structures before describing the profunctors determined by event structure spans.

Proposition 4.2.1. *Let E_1 and E_2 be event structures. A function f from the set of events E_1 to the set of events E_2 is a rigid morphism $f : E_1 \to E_2$ iff*

$$\forall x \in \mathcal{C}(E_1).\ f(x) \in \mathcal{C}(E_2)\ \&\ \forall e, e' \in x.\ f(e) = f(e') \Rightarrow e = e'\ ,\ \textit{and}$$
$$\forall x' \in \mathcal{C}(E_1), y \in \mathcal{C}(E_2).\ y \subseteq f(x')\ \Rightarrow \exists x \in \mathcal{C}(E_1).\ x \subseteq x'\ \&\ f(x) = y\ .$$

(x is necessarily unique and given by $x = \{e \in x' \mid f(e) \in y\}$.)

We now define the operation $\overline{(\_)}$ that maps a span $A \overset{d}{\leftarrow} E \overset{out}{\rightarrow} B$ to a functor

$$\overline{E} : \mathcal{C}^0(A) \times \mathcal{C}^0(B)^{op} \to \mathbf{Set},$$

i.e., a profunctor between $\mathcal{C}^0(A)$ and $\mathcal{C}^0(B)$.

Definition 4.2.2. *Define $\overline{E}$ for an event structure E by*

$$\begin{aligned} \overline{E}(a,\ b) &\overset{def}{=} \{x \in \mathcal{C}^0(E) \mid d^\dagger(x) \subseteq a\ \&\ out(x) = b\} \\ \overline{E}(a \subseteq a',\ b' \subseteq b)(x) &\overset{def}{=} \{e \in x \mid out(e) \in b'\} \end{aligned}$$

for $a,\ a' \in \mathcal{C}^0(A)$ and $b,\ b' \in \mathcal{C}^0(B)$.

We show that $\overline{E}$ is a rooted stable profunctor in the sense of [15, 16].

Proposition 4.2.3. *The operation $\overline{E}$ is a rooted stable profunctor.*

Proof. We first show that $\overline{E}$ is a functor from $\mathcal{C}^0(A) \times \mathcal{C}^0(B)^{op}$ to **Set**.

Suppose $a \subseteq a'$ in $\mathcal{C}^0(A)$. Then $\overline{E}(a, b) \subseteq \overline{E}(a', b)$ because any configuration x of E with demand $d^\dagger(x) \subseteq a$ will make $d^\dagger(x) \subseteq a'$. So, $\overline{E}(a \subseteq a',\ b)$ is an inclusion map from $\overline{E}(a,\ b)$ into $\overline{E}(a',\ b)$. Therefore $\overline{E}$ respects the inclusion order on configurations in $\mathcal{C}^0(A)$ covariantly.

Suppose $b' \subseteq b$ in $\mathcal{C}^0(B)$. Then $\overline{E}(a,\ b' \subseteq b)$ is a map from $\overline{E}(a,\ b)$ to $\overline{E}(a,\ b')$. Let x be a member of $\overline{E}(a,\ b')$. It follows from Proposition 4.2.1 and *out* being rigid that $\{e \in x \mid out(e) \in b'\}$ is a member of $\overline{E}(a,\ b')$. So, $\overline{E}$ respects the inclusion order on configurations in $\mathcal{C}^0(B)$ contravariantly.

It is easy to see that $\overline{E}$ respects identities and compositions of inclusions, so we have a functor to the category of sets

$$\overline{E} : \mathcal{C}^0(A) \times \mathcal{C}^0(B)^{\mathrm{op}} \to \mathbf{Set}\ ,$$

i.e., a *profunctor* from $\mathcal{C}^0(A)$ to $\mathcal{C}^0(B)$. As $\emptyset \in \mathcal{C}^0(B)$ and only $\emptyset$ is mapped to $\emptyset$ by *out* we have $\overline{E}(a,\ \emptyset) = \{\emptyset\}$ for all $a \in \mathcal{C}^0(A)$. It is therefore *rooted*.

It can easily be shown from the stability of *out* that $\overline{E}$ preserves pullbacks in its second argument. It is therefore *stable* in the sense of [15, 16]. □

We now define the action of $\overline{(\_)}$ on span morphisms (Definition 2.4.2).

Definition 4.2.4. *Let $f : E_1 \rightarrow E_2$ be a morphism between spans.*

$$\begin{tikzcd}
 & E_1 \arrow[ddl, "d_1"'] \arrow[d, "f"] \arrow[ddr, "out_1"] & \\
 & E_2 \arrow[dl, "d_2"] \arrow[dr, "out_2"'] & \\
A & & B
\end{tikzcd}$$

Define the family of operations $\overline{f}$ by

$$\overline{f}_{a,b}(x) \overset{def}{=} f(x)$$

for all $x \in \overline{E}_1(a,\ b)$

Proposition 4.2.5. *For all morphisms $f : E_1 \rightarrow E_2$ between spans of event structures $A \overset{d_1}{\leftarrow} E_1 \overset{out_1}{\rightarrow} B$ and $A \overset{d_2}{\leftarrow} E_2 \overset{out_2}{\rightarrow} B$, we have that $\overline{f}$ is a natural transformation from $\overline{E_1}$ to $\overline{E_2}$.*

Proof. We show that the following diagram commutes.

$$\begin{tikzcd}
\overline{E_1}(a,\ b) \arrow[r, "\overline{f}_{a,b}"] \arrow[d, "\overline{E_1}(a\subseteq a',\ b'\subseteq b)"'] & \overline{E_2}(a,\ b) \arrow[d, "\overline{E_2}(a\subseteq a',\ b'\subseteq b)"] \\
\overline{E_1}(a',\ b') \arrow[r, "\overline{f}_{a',b'}"'] & \overline{E_2}(a',\ b')
\end{tikzcd}$$

Let x be a member of $\overline{E_1}(a,\ b)$. As x is a configuration of E_1 we know that $f(x)$ is a configuration of E_2. It follows from Definition 2.4.2 that we have $d_2^\dagger(f(x)) \subseteq {d_1}^\dagger(x)$ and $out_2(f(x)) = out_1(x)$. So, $\overline{f}_{a,b}(x)$ is a member of $\overline{E_2}(a,\ b)$. It is easy to show that

$$\overline{E_2}(a \subseteq a',\ b' \subseteq b) \circ \overline{f}_{a,b} = \overline{f}_{a',\ b'} \circ \overline{E_1}(a \subseteq a',\ b \subseteq b').$$

□

In fact, spans embed faithfully in the bicategory of profunctors and the embedding reflects isomorphisms.

The tensor operation for profunctors given in [15, 16] is also a simple parallel composition.

Definition 4.2.6. *Let $X : \mathcal{C}^0(A_1) \times \mathcal{C}^0(B_1)^{op} \rightarrow$ **Set** and $Y : \mathcal{C}^0(A_2) \times \mathcal{C}^0(B_2)^{op} \rightarrow$ **Set** be profunctors. As before, assume without loss of generality that A_1 and A_2 are disjoint and also that B_1 and B_2 are disjoint. Define $X \otimes Y$ by*

$$(X \otimes Y)(a,\ b) \overset{def}{=} X(a \cap A_1,\ b \cap B_1) \times Y(a \cap A_2,\ b \cap B_2)$$

for all $a \subseteq A_1 \cup A_2$ and $b \subseteq B_1 \cup B_2$.

The tensor operation for profunctors corresponds to that for spans.

Proposition 4.2.7. *Let E_1 and E_2 be spans of event structures. There is a natural isomorphism between $\overline{E_1} \otimes \overline{E_2}$ and $\overline{E_1 \otimes E_2}$.*

Proof. Define $\theta : \overline{S_1 \otimes S_2} \to \overline{S_1} \otimes \overline{S_2}$ by

$$\begin{aligned} \theta_{a,b}(x) &\overset{def}{=} (x \cap E_1,\ x \cap E_2) \\ {\theta^{-1}}_{a,b}(w,\ z) &\overset{def}{=} w \cup z \end{aligned}$$

for all $x \in \overline{S_1 \otimes S_2}(a,\ b)$ and $(w,\ z) \in (\overline{S_1} \otimes \overline{S_2})(a,\ b)$. It is routine to check that θ is indeed a natural isomorphism. □

We now show that the trace construction on event structure spans coincides with that given for profunctors in [15, 16]. So, it inherits the properties of trace proved there.

We define trace on stable rooted profunctors (Definition 18 in [15]) by the usual coend operation on profunctors (see [9]) restricted to elements which are secure. Here, the concept of secure is expressed without the benefit of concepts from event structures. We recall the definitions with respect to $\overline{E}$ for a span of event structures

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

We assume this span from now on.

We recall the definition of an *element* of a profunctor.

Definition 4.2.8. *An element of the profunctor $\overline{E}$ is a triple $(y, z; x)$ where $x \in \overline{E}(y,\ z)$.*

The definition of trace on stable rooted profunctors relies on three relations between the elements.

Definition 4.2.9. *Let $(y, z; x)$ and $(y', z'; x')$ be elements of $\overline{E}$. Define*

$$\begin{aligned} (y, z; x) \xrightarrow{A} (y', z'; x') \ &\text{iff}\ \ z = z' \ \&\ x = x' \ \&\ \exists a \in A.\ y \cup \{a\} = y' \ ,\ \textit{and} \\ (y, z; x) \xrightarrow{B} (y', z'; x') \ &\text{iff}\ \ \exists b \in B.\ z \cup \{b\} = z' \ \&\ x = \{e \in x' \mid out(e) \in z\} \\ &\phantom{\text{iff}}\ \ \ \&\ y = y' \ . \end{aligned}$$

Define $(y, z; x) \rightsquigarrow (y', z'; x')$ iff $(y, z'; x')$ is an element with

$$\begin{aligned} & y \subseteq y' \text{ and } z \subseteq z' \text{ and} \\ & y \cap A = y' \cap A \text{ and } z \cap B = z' \cap B \text{ and} \\ & y \cap C = z \cap C \text{ and } y' \cap C = z' \cap C \text{ and} \\ & x = \{e \in x' \mid out(e) \in z\} \ . \end{aligned}$$

The relation $\xrightarrow{A}$ describes the change of element associated with an input in A. Similarly, $\xrightarrow{B}$ describes the change of element associated with an output on B. The relation $\rightsquigarrow$ may be thought of as describing the input in C that is matched by prior output on C. It is used in the definition of trace to identify those elements requiring no more input in C than they output.

The proposition below is used several times in the remainder of this section.

Proposition 4.2.10. *A finite configuration x of E is secure iff there is a securing sequence $e_1, ..., e_n$ such that $x = \{e_1, ..., e_n\}$.*

Proof. If: This follows directly from the definition of a securing sequence.
Only if: Assume that x is a secure configuration. From Corollary 4.1.4, the relation $\prec_x$ is well-founded and $d^\dagger(x) \cap C \subseteq out\ x$. As x is finite, we can order its events in a way that respects $\prec_x$, i.e., $e_1, ..., e_n$, such that $e_i \prec_x e_j$ implies $i < j$. This will be a securing sequence. □

The definition of *secure element* as described in [15, 16] is recalled below.

Definition 4.2.11. *Let $(y, z; x)$ be an element of $\overline{E}$. Then $(y, z; x)$ is a* secure element *iff $(\emptyset, \emptyset; \emptyset)(\xrightarrow{A} \cup \xrightarrow{B} \cup \rightsquigarrow)^\star(y, z; x)$.*

We now show that the notion of *secure element* corresponds to that of secure configuration.

Lemma 4.2.12. *Let $(y, z; x)$ be an element of $\overline{E}$. Then $(y, z; x)$ is a secure element iff the configuration x is secure.*

Proof. Only if: Define the property Q of an element $(y, z; x)$ of $\overline{E}$ by

$$Q(y, z; x) \text{ iff } x \text{ is secure and } y \cap C \subseteq out\ x.$$

Clearly we have $Q(\emptyset, \emptyset; \emptyset)$. In order to show that Q holds for all secure elements it therefore suffices to show the following. In the cases

i) $(y, z; x) \xrightarrow{A} (y', z'; x')$,

ii) $(y, z; x) \xrightarrow{B} (y', z'; x')$ or

iii) $(y, z; x) \rightsquigarrow (y', z'; x')$,

if $Q(y, z; x)$ holds then $Q(y', z', x')$ holds.
i) If $(y, z; x) \xrightarrow{A} (y', z'; x')$ then, from the definition, $x = x'$ and $y' \cap C = y \cap C$. So $Q(y, z; x)$ clearly implies $Q(y', z'; x')$.
ii) Suppose $(y, z; x) \xrightarrow{B} (y', z'; x')$ and $Q(y, z; x)$. Then, from the definition of $\xrightarrow{B}$, we have $y = y'$ and $x' = x \cup \{e'\}$ for some $e' \in x$ such that $out(e') \in B$. As $out(x')$ therefore contains $out(x)$ and $y \cap C = y' \cap C$, we have that $y' \cap C \subseteq out(x')$. From Proposition 4.2.10 and the fact that x is secure, a securing sequence $e_1, ..., e_n$ exists such that $\{e_1, ..., e_n\} = x$.

If we extend the sequence to $e_1, ..., e_n, e'$ we also have a securing sequence. This is the case because

$$\begin{aligned} d(e') \cap C &\subseteq d^\dagger(x') \cap C \\ &\subseteq y' \cap C \\ &= y \cap C \\ &\subseteq \mathit{out}(x) \\ &\subseteq \mathit{out}\{e_1, ..., e_n\}. \end{aligned}$$

Hence x' is secure. So, $Q(y', z'; x')$ holds.
iii) Assume $(y, z; x) \rightsquigarrow (y', z'; x')$ and $Q(y, z; x)$. As before, take a securing sequence $e_1, ...e_n$ such that $x = \{e_1, ..., e_n\}$. As x' is a finite configuration and $x \subseteq x'$, we can choose a sequence of events $e'_1, ..., e'_m$ such that $x' = x \cup \{e'_1, ..., e'_m\}$ and $x \cup \{e'_1, ..., e'_i\}$ is a configuration of E for all $0 < i \leq m$. We now show that the concatenated sequence is a securing sequence. It follows from the definition of $\rightsquigarrow$ that $(y, z'; x')$ must be an element of $\overline{E}$ and therefore we have $d^\dagger(x') \subseteq y$. Hence

$$\begin{aligned} d(e'_i) \cap C &\subseteq y \cap C \\ &\subseteq \mathit{out}(x) \text{ (from the holding of } Q(y, z; x)) \\ &\subseteq \{e_1, ..., e_n, e'_1, ..., e'_{i-1}\}. \end{aligned}$$

So, from Definition 4.1.1 and Proposition 4.2.10, we have that x' is secure. From the definition of $\rightsquigarrow$, we have $y' \cap C = z' \cap C$. Also, because $(y', z'; x')$ is an element of $\overline{E}$, we have $z' \cap C = \mathit{out}(x') \cap C$. So, $y' \cap C \subseteq \mathit{out}(x')$.

If: Define the relation $\xrightarrow{e}$ between configurations of E by $x \xrightarrow{e} x'$ iff x and x' are secure configurations of E such that $e \notin x$ and $x' = x \cup \{e\}$.

We first show that if $x \xrightarrow{e} x'$ and $(y, z; x)$ is an element with $z \cap C \subseteq y$, then

$$(y, z; x)(\xrightarrow{A} \cup \xrightarrow{B} \cup \rightsquigarrow)^\star(y', z'; x')$$

for some element $(y', z'; x')$ with $z' \cap C \subseteq y'$.

Assume $x \xrightarrow{e} x'$ and $(y, z; x)$ is an element with $z \cap C \subseteq y$. As x is secure, there must exist a securing sequence $e_1, ..., e_n$ such that $x = \{e_1, ..., e_n\}$. As x' is secure and $x' = x \cup \{e\}$, we have $d(e) \cap C \subseteq \mathit{out}(x)$. In proving that $(\emptyset, \emptyset; \emptyset)(\xrightarrow{A} \cup \xrightarrow{B} \cup \rightsquigarrow)^\star(y, z; x)$, we must consider two cases: (*i*) when $\mathit{out}(e) \in B$ and (*ii*) when $\mathit{out}(e) \in C$.
(*i*) Suppose $\mathit{out}(e) \in B$. Observe that $d(e) \backslash y \subseteq A$ because

$$\begin{aligned} d(e) \cap C &\subseteq (\mathit{out}(x)) \cap C \\ &= z \cap C \\ &\subseteq y. \end{aligned}$$

So, we have

$$(y, z; x) \xrightarrow{A}{}^\star (y \cup d(e), z; x) \xrightarrow{B} (y \cup d(e), z \cup \{\mathit{out}(e)\}; x').$$

Also, as $out(e) \in B$, we have $(z \cup \{out(e)\}) \cap C = z \cap C$ and therefore $(z \cup \{out(e)\}) \cap C$ is a subset of y and therefore of $y \cup d(e)$.
(ii) Suppose $out(e) \in C$. Again, we have $d(e) \backslash y \subseteq A$, via the same argument as before. So, we have

$$(y, z; x) \overset{A}{\rightarrow}^{\star} (y \cup d(e), z; x) \rightsquigarrow (y \cup d(e) \cup \{out(e)\}, z \cup \{out(e)\}; x').$$

Also, we have

$$\begin{array}{rcl} (z \cup \{out(e)\}) \cap C & = & (z \cap C) \cup \{out(e)\} \\ & \subseteq & y \cup d(e) \cup \{out(e)\}. \end{array}$$

Assume that x is a secure configuration of E. As $(y, z; x)$ is an element of $\overline{E}$, x is necessarily finite. From Proposition 4.2.10, we can choose a securing sequence $e_1, ..., e_n$ such that $\{e_1, ..., e_n\}$ equals x. Define x_i to be the set $\{e_1, ..., e_i\}$, for $0 \leq i \leq n$. So, each x_i is secure and, from the definition of $\overset{e}{\rightarrow}$, we have

$$\emptyset = x_0 \overset{e_1}{\rightarrow} x_1 \overset{e_2}{\rightarrow} x_2 ... \overset{e_n}{\rightarrow} x_n = x.$$

From this chain and the previous result, we can inductively produce a chain

$$(\emptyset, \emptyset; \emptyset)(\overset{A}{\rightarrow} \cup \overset{B}{\rightarrow} \cup \rightsquigarrow)^{\star}(y, z; x)$$

to the element $(y, z; x)$ of $\overline{E}$. □

We now define the trace of a stable rooted profunctor $\overline{E}$ from [15, 16]. This construction mimics the definition of the construction of a coend in the category of sets but is restricted to secure elements. It is defined in terms of an equivalence relation $\sim$ between secure elements. For the benefit of readers unfamiliar with coends, we define the trace operation without the notation associated with them.

Definition 4.2.13. *Define $\sim$ to be the symmetric, transitive closure of $\rightsquigarrow$.*

Define the stable rooted profunctor

$$Tr(\overline{E}) : \mathcal{C}^0(A) \times \mathcal{C}^0(B)^{op} \rightarrow \mathbf{Set}$$

as follows.

$$\begin{array}{l} Tr(\overline{E})(y_0,\ z_0) \overset{def}{=} \\ \{[y, z; x]_{\sim} \mid (y, z; x) \textit{ is a secure element and } y \cap A = y_0 \textit{ and } z \cap B = z_0\} \end{array}$$

for all $y_0 \in \mathcal{C}^0(A)$ and $z_0 \in \mathcal{C}^0(B)$.

If $y_0 \subseteq y_0'$ in $\mathcal{C}^0(A)$ then

$$Tr(\overline{E})(y_0 \subseteq y_0',\ z_0) : Tr(\overline{E})(y_0,\ z_0) \rightarrow Tr(\overline{E})(y_0',\ z_0)$$

is defined to map an equivalence class of elements $[y, z; x]_\sim \in Tr(\overline{E})(y_0,\ z_0)$ *to the equivalence class* $[y \cup y_0', z; x]_\sim$.

If $z_0 \subseteq z_0'$ *in* $\mathcal{C}^0(B)$ *then*

$$Tr(\overline{E})(y_0,\ z_0 \subseteq z_0') : Tr(\overline{E})(y_0,\ z_0') \to Tr(\overline{E})(y_0,\ z_0)$$

is defined to map an equivalence class of elements $[y, z; x]_\sim$ *in* $Tr(\overline{E})(y_0,\ z_0')$ *to* $[y, z'; x']_\sim$ *where* $z' = (z \cap C) \cup z_0$ *and* $x' = \{e \in x \mid out(e) \in (z \cap C) \cup z_0\}$.

To see that $Tr(\overline{E})(y_0 \subseteq y_0',\ z_0)$ is well-defined, consider the following. Let $[y, z; x]_\sim$ be a member of $Tr(\overline{E})(y_0,\ z_0)$. To see that $[y \cup y_0', z; x]_\sim$ is a member of $Tr(\overline{E})(y_0',\ z_0)$, observe that we have $(y, z; x) \overset{A}{\rightarrow}^{\star} (y \cup y_0', z; x)$. Therefore, as $(y, z; x)$ is a secure element, we know $(y \cup y_0', z; x)$ is secure. It is also clear that $(y \cup y_0') \cap A = y_0'$ as $y_0 \subseteq y_0'$. Finally, suppose $(y', z'; x') \in [y, z; x]_\sim$. Observe that $(y, z; x) \sim (y', z'; x')$ implies $(y \cup y_0', z; x) \sim (y' \cup y_0', z'; x')$.

To see that $Tr(\overline{E})(y_0,\ z_0 \subseteq z_0')$ is well-defined, consider the following. Let $[y, z; x]_\sim$ be a member of $Tr(\overline{E})(y_0,\ z_0')$ and let $[y, z'; x']_\sim$ be $Tr(\overline{E})(y_0,\ z_0 \subseteq z_0')[y, z; x]_\sim$. That x' is a configuration of E follows from Proposition 4.2.1. Clearly, *out* maps x' to z'. We have $(y, z'; x') \overset{B}{\rightarrow}^{\star} (y, z; x)$ and so $(y, z'; x')$ is secure. So, $[y, z'; x']_\sim$ is indeed a member of $Tr(\overline{E})(y_0,\ z_0)$. Finally, suppose $(y'', z''; x'') \in [y, z; x]_\sim$. Let $z''' = (z'' \cap C) \cup z_0$ and $x''' = \{e \in x'' \mid out(e) \in (z'' \cap C) \cup z_0\}$. Observe that $(y, z; x) \sim (y'', z''; x'')$ implies $(y, z'; x') \sim (y'', z'''; x''')$.

Definition 4.2.14. *Let* x *be a secure configuration of* E. *Define* $min(x)$ *by*

$$min(x) \overset{def}{=} \bigcap \{s \subseteq x \mid s \in \mathcal{C}(E) \text{ and } s \text{ secure and } out(s) \cap B = out(x) \cap B\}.$$

Observe that, because the secure configurations of E form a stable family (see Theorem 4.1.6), $min(x)$ is the minimum secure sub-configuration of x that outputs the same events in B as x.

Lemma 4.2.15. *For secure elements, we have*

$$\begin{aligned}&(y, z; x) \sim (y', z'; x') \textit{ iff} \\ &\quad y \cap A = y' \cap A \textit{ and } z \cap B = z' \cap B \textit{ and } min(x) = min(x').\end{aligned}$$

Proof. If: Assume $y \cap A = y' \cap A$ and $z \cap B = z' \cap B$ and $min(x) = min(x')$. Let $x_0 = min(x)$, $y_0 = d^\dagger(x_0) \cup (y \cap A)$ and $z_0 = out(x_0)$. It follows from the definition of $\rightsquigarrow$ that we have $(y_0, z_0; x_0) \rightsquigarrow (y, z; x)$ and $(y_0, z_0; x_0) \rightsquigarrow (y', z'; x')$. Hence $(y, z; x) \sim (y', z'; x')$.
Only if: Assume $(y, z; x) \rightsquigarrow (y', z'; x')$. From the definition of $\rightsquigarrow$ (Definition 4.2.9), we have $y \cap A = y' \cap A$ and $z \cap B = z' \cap B$ and that x is a subset of x'. From Definition 4.2.14, $min(x)$ is the least secure configuration s such that $s \subseteq x$ and $out(s) \cap B = z \cap B$. As $z' \cap B = z \cap B$, we know $out(x' \backslash x) \cap B = \emptyset$. So, $min(x)$ is also the least secure configuration s such that $s \subseteq x'$ and $out(s) \cap B = z' \cap B$. It must therefore be equal to $min(x')$. □

Theorem 4.2.16. *There is a natural isomorphism*

$$\theta : Tr(\overline{E}) \cong \overline{Tr(E)}$$

with components $\theta_{y_0,z_0} : Tr(\overline{E})(y_0,\ z_0) \to \overline{Tr(E)}(y_0,\ z_0)$ *defined by*

$$\theta_{y_0,z_0}([y,z;x]_\sim) \overset{def}{=} \{p \in Tr(E) \mid p \subseteq x\}.$$

Proof. Let x be a secure configuration of E. As the secure configurations of E form a stable family $\mathcal{S}$, it follows from the properties of stable families that x is the union of all the prime configurations p in $\mathcal{S}$ below it. As $Tr(E)$ has the prime configurations of $\mathcal{S}$ for which $out(max(p)) \in B$ as its events, we have

$$min(x) = \bigcup\{p \in Tr(E) \mid p \subseteq x\}.$$

We argue that this implies θ_{y_0,z_0} is a well-defined function. We have that $\theta_{y_0,z_0}([y,z;x]_\sim)$ consists of those $p \in Tr(E)$ for which $p \subseteq min(x)$. It is therefore a configuration of $Tr(E)$. It follows directly from the definition of $Tr(E)$ that, as $min(x) \subseteq x$, the demand of $\theta_{y_0,z_0}([y,z;x]_\sim)$ is a subset of $y_0 = y \cap A$. Also, because $out(min(x)) \cap B = out(x) \cap B$, the output for $\theta_{y_0,z_0}([y,z;x]_\sim)$ is equal to $z_0 = z \cap B$. Finally, the definition of θ_{y_0,z_0} is independent of the choice of element in the equivalence class. Suppose $(y',z';x') \in [y,z;x]_\sim$. It is easy to show $min(x) = min(x')$ and therefore

$$\{p \in Tr(E) \mid p \subseteq x\} = \{p \in Tr(E) \mid p \subseteq x'\}.$$

Define $\varphi_{y_0,z_0} : \overline{Tr(E)}(y_0,z_0) \to Tr(\overline{E})(y_0,z_0)$ by

$$\varphi_{y_0,z_0}(X) \overset{def}{=} [y,z;x]_\sim$$

where $x = \bigcup X$, $y = d^\dagger(x) \cup y_0$ and $z = out(x)$. Then we have

$$\begin{aligned} \varphi_{y_0,z_0}\theta_{y_0,z_0}([y,z;x]_\sim) &= \varphi_{y_0,z_0}(\{p \in Tr(E) \mid p \subseteq x\}) \\ &= [d^\dagger(min(x)) \cup y_0, out\ min(x); min(x)]_\sim \\ &= [y,z;x]_\sim, \end{aligned}$$

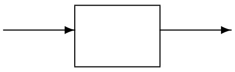
from the above observations. Also

$$\theta_{y_0,z_0}\varphi_{y_0,z_0}(X) = \{p \in Tr(E) \mid p \subseteq \bigcup X\} = X,$$

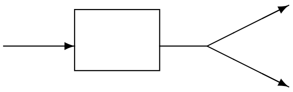
from the properties of primes as X is downwards closed. So, φ_{y_0,z_0} is the inverse of θ_{y_0,z_0}. It is routine to verify the naturality of θ_{y_0,z_0}. □

4.3 Deterministic Dataflow

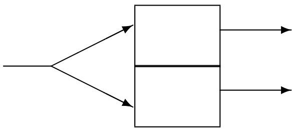
Selinger ([32]) describes a deterministic dataflow process as a dataflow process



for which the process



is indistinguishable from the process



where the splitting arrows represent the fork process.

To connect with Kahn's classic treatment of deterministic dataflow, we define the notion of a *deterministic span.* Such spans represent stable functions between domains of configurations, and can be used to model deterministic dataflow. The trace construction on event structures is shown to correspond to the least fixed point construction detailed in [19].

Definition 4.3.1. *A span*

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

is deterministic *iff* $d(e_1) \uparrow d(e_2) \Rightarrow \neg(e_1 \# e_2)$ *for all* $e_1,\ e_2 \in E$.

The extra requirement of the demand morphism ensures that no two conflicting events of E can require consistent configurations of A. Such spans represent stable functions (see [2]) between domains of configurations.

Definition 4.3.2. *Let* $A \overset{d}{\leftarrow} E \overset{out}{\rightarrow} B$ *be a deterministic span. Define*

$$\widetilde{E}(y) \overset{def}{=} out\{e \in E \mid d(e) \subseteq y\}\ ,\ \text{for all } y \in \mathcal{C}(A).$$

Theorem 4.3.3. *The mapping* $\widetilde{E}$ *determines a function* $\mathcal{C}(A) \rightarrow \mathcal{C}(B)$ *that is continuous and stable for all deterministic spans* E. *Moreover, any stable function from* $\mathcal{C}(A)$ *to* $\mathcal{C}(B)$ *can be represented in this way.*

Proof. To show $\widetilde{E}(y) \in \mathcal{C}(B)$, as out preserves configurations, it is sufficient to demonstrate that the set $\{e \in E \mid d(e) \subseteq y\}$ is a configuration of E. Let $x = \{e \in E \mid d(e) \subseteq y\}$. To see that x is downwards closed, let $e \in x$ and suppose $e' \leq e$. Then $d(e') \subseteq d(e)$ due to the properties of demand morphisms and so $d(e') \subseteq y$. So, we have $e' \in x$. That all the events are consistent follows from the definition of deterministic spans.

As the demand of an event is a finite configuration, it follows straightforwardly that $\widetilde{E}$ is continuous. To see it is stable we observe that $\widetilde{E}$ factors into the composition $out \circ F$ where F is defined by

$$F(y) = \{e \in E \mid d(e) \subseteq y\}\ ,\ \text{for } y \in \mathcal{C}(A)\ .$$

The function $F : \mathcal{C}(A) \to \mathcal{C}(E)$ preserves all intersections so it is certainly stable, while *out* is a rigid morphism so a stable function on configurations, from $\mathcal{C}(E)$ to $\mathcal{C}(B)$. Their composition $\widetilde{E}$ is therefore stable.

To see that any stable function $f : \mathcal{C}(A) \to \mathcal{C}(B)$ can be represented by a deterministic span, define a corresponding event structure E as follows. Its events E are pairs (y, b) where y is a finite configuration of A and b is an event of B for which y is a minimal configuration such that $b \in f(y)$, i.e., b is a member of $f(y)$ and for no configuration $y' \subset y$ do we have $b \in f(y')$. (Because f is stable, if $b \in f(w)$ then there is a unique, minimal $y \subseteq w$ with $(y, b) \in E$.) Define

$$\begin{aligned}&(y, b) \leq (y', b') \text{ iff } y \subseteq y' \ \&\ b \leq b'\ ,\ \text{and}\\ &(y, b)\#(y', b') \text{ iff } y \not\uparrow y' \ \text{ or }\ b\#b'\ .\end{aligned}$$

That $\leq$ is a partial order follows immediately from the subset relation and the causality relation being partial orders and the continuity of f. Let $(y_1,\ b_1)$, $(y_2,\ b_2)$ and $(y_3,\ b_3)$ be events in E. That $[(y,\ b)]$ is finite for all $(y,\ b) \in E$ follows immediately from the finiteness of y and $[b]$. Clearly the conflict relation is irreflexive and symmetric. In order to confirm that E is an event structure, we must show that $(y_1,\ b_1)\#(y_2,\ b_2)$ and $(y_2,\ b_2) \leq (y_3,\ b_3)$ implies $(y_1,\ b_1)\#(y_3,\ b_3)$. There are two cases to consider – $y_1 \not\uparrow y_2$ and $b_1\#b_2$. In the first case, as $y_2 \subseteq y_3$, we have $y_1 \not\uparrow y_3$ and therefore $(y_1,\ b_1)\#(y_3,\ b_3)$. In the second case we have $b_2 \leq b_3$ and $b_1\#b_2$ so, as B is an event structure, $b_1\#b_3$. So E is an event structure.

Define $d : E \to \mathcal{C}(A)$ by $d(y,\ b) \overset{def}{=} y$. We now show that d is a demand morphism. Let $(y_1,\ b_1)$ and $(y_2,\ b_2)$ be events in E. That $(y_1,\ b_1) \leq (y_2,\ b_2)$ implies $d(y_1,\ b_1) \subseteq d(y_2,\ b_2)$ and also that $d(y_1,\ b_1) \not\uparrow d(y_2,\ b_2)$ implies $(y_1,\ b_1)\#(y_2,\ b_2)$ is obvious from the definitions of d and E. So, d is indeed a demand morphism. We also show $d(y_1,\ b_1) \uparrow d(y_2,\ b_2)$ implies $\neg(b_1\#b_2)$. Assume $d(y_1,\ b_1) \uparrow d(y_2,\ b_2)$. Then we have $y_1 \uparrow y_2$. It remains to check $\neg(b_1\#b_2)$. If $y_1 \uparrow y_2$ then there exists $z \in \mathcal{C}(A)$ such that $y_1 \subseteq z$ and $y_2 \subseteq z$. As f is continuous, $f(y_1) \subseteq f(z)$ and $f(y_2) \subseteq f(z)$ and $f(z) \in \mathcal{C}(B)$. So b_1 and b_2 are members of $f(z)$ and therefore are not in conflict.

Define $out : E \to B$ by $out(y,\ b) \overset{def}{=} b$. We show that *out* is a rigid morphism. First we show $[out(y,\ b)] = out[(y,\ b)]$ for all $(y,\ b) \in E$. Let $b' \in [out(y,\ b)]$. Then $b' \leq b$ so $b' \in f(y)$. From the stability of f, there is a unique minimal $y' \subseteq y$ with $(y',\ b') \in E$ and clearly $(y',\ b') \leq (y,\ b)$. So $b' \in out[(y,\ b)]$. Let $b' \in out[(y,\ b)]$. Then there exists a y' such that $(y',\ b') \leq (y,\ b)$. So $b' \leq b$ and therefore $b' \in [out(y,\ b)]$. It is obvious from the definition of conflict in E that *out* reflects conflict. Finally we show $out(y,\ b) = out(y',\ b')$ implies $(y,\ b) = (y',\ b')$ or $(y,\ b)\#(y',\ b')$. Assume $out(y,\ b) = out(y',\ b')$ for distinct $(y,\ b)$ and $(y',\ b')$ in E. Then $b = b'$ but $y \neq y'$. As f is stable, there cannot be $z \in \mathcal{C}(A)$ containing both y and y' and therefore $y \not\uparrow y'$. So $(y,\ b)\#(y',\ b')$.

We therefore have that E, d and *out* together form a deterministic span. It is clear that $\widetilde{E} = f$. □

Theorem 4.3.3 and its proof provide two operations, one from deterministic spans to stable functions, and an inverse operation from stable functions to deterministic spans. These operations are part of a bi-equivalence between the bicategory of deterministic spans and the order-enriched category of stable functions on coherent prime algebraic domains.

Another interesting point to note is that the morphisms between deterministic spans now correspond to a continuous ordering of the continuous functions to which the spans correspond. Let f_1 and f_2 be continuous functions between the configurations of two event structures A and B. The ordering given by span morphisms corresponds to defining f_1 as being below f_2 iff $f_1(a) \subseteq f_2(a)$ for all $a \in \mathcal{C}(A)$.

Proposition 4.3.4. *The composition of two deterministic spans is deterministic.*

Proof. Let $A \overset{d_1}{\leftarrow} E_1 \overset{out_1}{\rightarrow} B$ and $B \overset{d_2}{\leftarrow} E_2 \overset{out_2}{\rightarrow} C$ be deterministic spans with composition $A \overset{d}{\leftarrow} E_3 \overset{out}{\rightarrow} C$, which we check is deterministic.

Let (x_1, e_2) and (x_1', e_2') be events in E_3 and assume $d(x_1, e_2) \uparrow\ d(x_1', e_2')$. This is true iff ${d_1}^{\dagger}(x_1) \uparrow {d_1}^{\dagger}(x_1')$. As the span E_1 is deterministic, $x_1 \uparrow x_1'$. Consequently, $d_2(e_2) \uparrow d_2(e_2')$, because, by the definition of composition, $d_2(e_2) = out_1(x_1)$ and $d_2(e_2') = out_1(x_1')$ and out_1 reflects conflict. This implies $\neg(e_2\#e_2')$, as the span E_2 is deterministic. As $x_1 \uparrow x_1'$ and $\neg(e_2\#e_2')$ it follows that we have $\neg((x_1, e_2)\#(x_1', e_2'))$. □

For the remainder of this section, let

$$A \otimes C \overset{d}{\leftarrow} E \overset{out}{\rightarrow} B \otimes C$$

be a deterministic span. We explore the properties of its trace. As usual, we assume that $A \cap C = \emptyset$ and $B \cap C = \emptyset$ and take the parallel compositions $A \otimes C$ and $B \otimes C$ to be unions as described in Section 2.3.

Lemma 4.3.5. *Let $y \in \mathcal{C}(A)$. For $s \subseteq E$ define φ by*

$$\varphi(s) \overset{def}{=} \{e \in E \mid d(e) \subseteq y \cup (C \cap out\, s)\}\ .$$

This determines a continuous function $\varphi : \mathcal{C}(E) \to \mathcal{C}(E)$ such that $Sec(y) \overset{def}{=} \mu s.\varphi(s)$, the least fixed point of φ, is a secure configuration of E.

Moreover, $Sec(y)$ is the maximum secure configuration x of E such that

$$d^{\dagger}(x) \cap A \subseteq y\ .$$

Proof. That φ is a function between configurations follows from determinism, while continuity follows from finiteness of demands, as in the proof of Theorem 4.3.3.

Then $\mu s.\varphi(s) = \bigcup_{i\in\omega} s_i$, a union of an $\subseteq$-increasing chain of configurations of E given inductively by

$$\begin{array}{rcl} s_0 & = & \emptyset \\ s_{i+1} & = & \varphi(s_i). \end{array}$$

Observe that, as φ is a function from configuration to configuration, s_i is a member of $C(E)$ for all $i \in \omega$.

We show by induction that s_i is secure for all $i \in \omega$.

Base case: The property trivially holds for s_0.

Inductive step: Assume that s_k is secure. Each event in s_k is secured and therefore has a securing sequence within s_k. Any additional events e in s_{k+1} have $d(e) \cap C \subseteq out\, s_k$. We can therefore give a securing sequence for e by combining the securing sequences of the events upon which it depends according to $\prec_{s_{k+1}}$. Therefore s_{k+1} is secure. This shows that $Sec(y)$ is secure for all $y \in \mathcal{C}$ and it is clear that $d^\dagger(Sec(y)) \cap A \subseteq y$.

Suppose $x \in \mathcal{C}(E)$ is secure and $d^\dagger(x) \cap A \subseteq y$. By Corollary 4.1.4, $\prec_x$ is well-founded. We show by well-founded induction on $\prec_x$ that

$$\forall e \in x.\ e \in Sec(y)\ .$$

Suppose $e' \in Sec(y)$ for all $e' \prec_x e$. Then $d(e) \cap C \subseteq out(Sec(y))$ as x is secure and by assumption $d(e) \cap A \subseteq y$; because x is secure $d(e) \cap C \subseteq out(x)$ (Corollary 4.1.4) and if $out(e') \in d(e) \cap C$ then $e' \to_x e$, so $e' \in Sec(y)$ inductively. Hence $d(e) \subseteq y \cup (C \cap out(Sec(y)))$ and therefore $e \in \varphi(Sec(y)) = Sec(y)$. □

Corollary 4.3.6. *The trace $Tr(E)$, of a deterministic span E, is deterministic.*

Proof. Suppose p_1, $p_2 \in Tr(E)$ have compatible demands $d'(p_1)$ and $d'(p_2)$, *i.e.,* there is a configuration y of A such that

$$d^\dagger(p_1) \cap A \subseteq y \text{ and } d^\dagger(p_2) \cap A \subseteq y\ .$$

From Lemma 4.3.5, $Sec(y)$ is the maximum secure configuration x of E such that

$$d^\dagger(x) \cap A \subseteq y.$$

Both p_1 and p_2 are secure configurations. Hence both $p_1 \subseteq Sec(y)$ and $p_2 \subseteq Sec(y)$, so p_1 and p_2 are compatible as secure configurations, and hence not in conflict as events of the trace. □

Lemma 4.3.7. *For all $y \in \mathcal{C}(A)$,*

$$\widetilde{Tr(E)}(y) = B \cap (out(Sec(y)))\ .$$

Proof. By definition

$$\widetilde{Tr(E)}(y) = out'\{p \in Tr(E) \mid d^\dagger(p) \cap A \subseteq y\}\ ,$$

where $out'(p) = out(max(p)) \in B$ for $p \in Tr(E)$. If $d^\dagger(p) \cap A \subseteq y$ with $p \in Tr(E)$ then p is secure, so, from Lemma 4.3.5, $p \subseteq Sec(y)$ by the maximum property of $Sec(y)$. It follows that $\widetilde{Tr(E)}(y) \subseteq B \cap (out(Sec(y)))$. To see the reverse inclusion, note that for any $e \in Sec(y)$ with $out(e) \in B$ we have $[e]_{Sec(y)} \in Tr(E)$. □

In conclusion we can understand the trace of the deterministic span in terms of the least fixed point of its associated function; the trace of deterministic event structures reduces to the trace known to Kahn [19].

Theorem 4.3.8. *For all $w \in \mathcal{C}(A)$,*

$$\widetilde{Tr(E)}(w) = B \cap (\mu z \in \mathcal{C}(B \otimes C).\, \widetilde{E}(w \cup (C \cap out\, z)))\ .$$

Proof. By Lemma 4.3.7,

$$\widetilde{Tr(E)}(w) = B \cap out(\mu x.\, \varphi(x))$$

where $\varphi : \mathcal{C}(E) \to \mathcal{C}(E)$ is the continuous function given by

$$\varphi(x) = \{e \in E \mid d(e) \subseteq w \cup (C \cap out(x))\}$$

for $x \in \mathcal{C}(E)$.

Define the continuous function $\psi : \mathcal{C}(B \otimes C) \to \mathcal{C}(B \otimes C)$ by taking

$$\psi(u) = \widetilde{E}(w \cup (C \cap u))$$

for $u \in \mathcal{C}(B \otimes C)$.

It is easy to see that $out : \mathcal{C}(E) \to \mathcal{C}(B \otimes C)$ induces a strict continuous function between the domains of configurations. Directly from their definition we see that

$$out\, \varphi(x) = \psi(out\, x)$$

for $x \in \mathcal{C}(E)$. So both expressions equal

$$out\{e \in E \mid d(e) \subseteq w \cup (C \cap out\, x)\}\ .$$

By the well-known uniformity property of least fixed points (making use of the fact that *out* is strict) we now know that $\mu u.\, \psi(u) = out(\mu x.\, \varphi(x))$, which gives the result directly. □

4.4 Concluding Remarks

Spans of event structures provide an intuitive semantics for both deterministic and nondeterministic dataflow. The semantics provides another good example of the way in which event structures can be used to represent both types and processes acting between types.

Chapter 5

Extending Event Structures

We have shown in the previous two chapters that event structure spans are expressive enough to model a variety of processes. However, there are some behaviours that cannot be modelled. For example, synchronous or partially synchronous parallel composition can be modelled by event structures as the product in the category of augmentation morphisms $\mathcal{A}$ or partial morphisms $\mathcal{P}$ as demonstrated in Section 2.2.2 but not by spans based on the categories of demand morphisms and rigid morphisms $\mathcal{D}$ and $\mathcal{R}$ (see Section 1.5). Roughly, this is because the projections out of these constructions are not rigid morphisms. In order to capture these behaviours, we would like to vary the kinds of morphisms used in the spans but be able to relate them. In this chapter we examine how to relate different sorts of morphism as a step towards this. We use these investigations and other results to justify an extension to the definition of event structures to allow *persistent events*.

In Section 5.1 we discuss how categories of event structures can be related. In Section 5.2, labelled event structures are related to certain kinds of presheaf. The results recalled therein are used in Section 5.3 to highlight some behaviours that event structures cannot model. Section 5.4 introduces a new kind of event structure and describes rigid, augmentation, partial and demand morphisms relating them. Using the new definitions, in Section 5.5, an adjunction between the category of augmentation morphisms and the category of partial morphisms is described. Finally, in Section 5.6, we discuss how to relate the other categories. We describe some of the associated difficulties and give an intuition for how they can be overcome.

5.1 Relating Categories of Event Structures

Let $\mathcal{R}$ and $\mathcal{A}$ be the categories of event structures defined in Section 2.1.1. In this section we define a coreflection between the categories $\mathcal{R}$ and $\mathcal{A}$. This is used to construct a monad for which the Kleisli category is isomorphic to $\mathcal{A}$. We would like to be able to do the same for the other event structure categories $\mathcal{P}$ and, in particular, $\mathcal{D}$. This would provide us with an alternative way of understanding the spans described in Section 2.4. However, we show that no right adjoints exist to the inclusion functors from $\mathcal{R}$ into $\mathcal{P}$ and $\mathcal{D}$. A small alteration to the definition of event structures is shown to be sufficient for an adjunction to exist between the category with augmentation morphisms and the category with partial morphisms. However, we end the section by showing that, for this alteration, we no longer have a right adjoint to the inclusion of the category of rigid morphisms into the category with augmentation morphisms. Thus, we demonstrate the requirement for an additional alteration to the definition of event structures.

Observe that the conditions on morphisms in $\mathcal{A}$ are a relaxation of those on morphisms in $\mathcal{R}$. It follows that $\mathcal{R}$ is a subcategory of $\mathcal{A}$. In this section we define a right adjoint to the inclusion functor $I : \mathcal{R} \rightarrow \mathcal{A}$.

Definition 5.1.1. *An* augmentation *of an event structure E is an elementary event structure $(x,\ \leq)$ where $x \in \mathcal{C}^0(E)$ and $e_1 \leq e_2$ in E implies that $e_1 \leq e_2$ in x for all $e_1,\ e_2 \in x$.*

Observe that there may be more dependencies between events in x than between those events in E. Hence we can think of the causality relation as having been *augmented*. Following Definition 2.1.2, we say that an augmentation is *prime* if it has a maximum element.

We now define a functor $Aug : \mathcal{A} \rightarrow \mathcal{R}$.

Definition 5.1.2. *Let the action of Aug on objects in $\mathcal{A}$ be defined by*

$$Aug(E,\ \leq,\ \#) \overset{def}{=} (E',\ \leq',\ \#')$$

where

- *the set E' consists of the prime augmentations of $(E,\ \leq,\ \#)$,*
- *$(x,\ \leq_x)\ \leq'\ (x',\ \leq_{x'})$ iff $x \subseteq x'$ and $\leq_x \subseteq \leq_{x'}$ and $e_1 \leq_x e_2$ iff $e_1 \leq_{x'} e_2$ for all $e_1,\ e_2 \in x$,*
- *$(x,\ \leq_x)\#'(x',\ \leq_{x'})$ iff $x \not\uparrow x'$ or there exists $e \in x \cap x'$ such that $[e]$ for $(x,\ \leq_x)$ is not equal to $[e]$ for $(x',\ \leq_{x'})$.*

Observe that the causality relation could be described equivalently by

$$(x,\ \leq_x) \leq' (x',\ \leq_{x'}) \text{ iff } (x,\ \leq_x) = ((x',\ \leq_{x'}) \restriction x).$$

It is therefore a partial order.

Let $f : E_1 \rightarrow E_2$ be a morphism in $\mathcal{A}$. Let the action of Aug on morphisms be defined by

$$Aug(f)(x,\ \leq) \overset{def}{=} (f(x),\ f(\leq))$$

for all prime augmentations $(x,\ \leq)$ of E_1 and where $f(e_1)\ f(\leq)\ f(e_2)$ iff $e_1 \leq e_2$ holds for all $e_1,\ e_2 \in x$. (This is well-defined as, from the definition of augmentation morphisms, we know that f is injective when restricted to domain x.)

It is routine to show that $Aug(E)$ is an event structure for all event structures E.

Lemma 5.1.3. *For all morphisms $f : E_1 \to E_2$ in $\mathcal{A}$, we have that $Aug(f)$ is a rigid morphism between $Aug(E_1)$ and $Aug(E_2)$.*

Proof. We first show $Aug(f)(x,\ \leq_x) \in Aug(E_2)$ for all $(x,\ \leq_x) \in Aug(E_1)$. Let $(x,\ \leq_x)$ be an event in $Aug(E_1)$. As f preserves configurations we have that $f(x) \in \mathcal{C}(E_2)$. As f is an augmentation morphism it reflects causality and so if $f(e_1) \leq f(e_2)$ in E_2 for $e_1,\ e_2$ in x then $f(e_1)\ f(\leq)\ f(e_2)$. It remains to check that $f(\leq)$ is a partial order with a maximum event. Recall that, as f is an augmentation morphism, if $f(e_1) = f(e_2)$ for non-conflicting events in E_1 then $e_1 = e_2$. It follows that, as x is a configuration of E_1, f will be injective on x. This implies that $(x,\ \leq)$ is isomorphic to $(f(x),\ f(\leq))$ and therefore that $f(\leq)$ is a partial order with a maximum event. We therefore have that $(f(x),\ f(\leq))$ is an event in $Aug(E_2)$.

We next show $[Aug(f)(e)] = Aug(f)[e]$ for all $e \in Aug(E_1)$. As explained above, $Aug(f)(e)$ is isomorphic to e for all $e \in Aug(E_1)$. It follows immediately from the definition of causality for $Aug(E_1)$ and $Aug(E_2)$ that $[Aug(f)(e)] = Aug(f)[e]$ holds for all $e \in Aug(E_1)$.

In the following let $(x_1,\ \leq_1)$ and $(x_2,\ \leq_2)$ be events in $Aug(E_1)$.

We now show that $Aug(f)$ reflects conflict. Suppose $Aug(f)(x_1,\ \leq_1)$ is in conflict with $Aug(f)(x_2,\ \leq_2)$ in $Aug(E_2)$. There are two cases to consider – $f(x_1) \not\uparrow f(x_2)$ or there exists $e \in f(x_1) \cap f(x_2)$ such that $[e]$ for $(f(x_1),\ f(\leq_1))$ is not equal to $[e]$ for $(f(x_2),\ f(\leq_2))$. In the first case, as f reflects conflict, it must be that we have $x_1 \not\uparrow x_2$. Recall that $(f(x_i),\ f(\leq_i))$ is isomorphic to $(x_i,\ \leq_i)$ for $i = 1,\ 2$. So, assuming the second case, even if x_1 is compatible with x_2, then as f is injective on $x_1 \cup x_2$, there exists $e \in x_1 \cap x_2$ for which $[e]$ in $(x_1,\ \leq_1)$ is not equal to $[e]$ in $(x_2,\ \leq_2)$. Therefore $(x_1,\ \leq_1)\#(x_2,\ \leq_2)$. Hence conflict is reflected by $Aug(f)$.

Finally, we show that if $Aug(f)(x_1,\ \leq_1) = Aug(f)(x_2,\ \leq_2)$ then $(x_1,\ \leq_1)$ equals $(x_2,\ \leq_2)$ or $(x_1,\ \leq_1)\#(x_2,\ \leq_2)$. Assume $(x_1,\ \leq_1) \neq (x_2,\ \leq_2)$. As $(x_1,\ \leq_1)$ is isomorphic to $Aug(f)(x_1,\ \leq_1)$ and therefore to $(x_2,\ \leq_2)$ it must be the case that $x_1 \neq x_2$. So, as $f(x_1) = f(x_2)$ there exist events in x_1 and x_2 that are distinct but map to the same thing. This implies $x_1 \not\uparrow x_2$ given the properties of f and therefore that $(x_1, \leq_1)\#(x_2,\ \leq_2)$. □

Proposition 5.1.4. *The operation Aug is a functor between $\mathcal{A}$ and $\mathcal{R}$.*

Proof. The result follows from Lemma 5.1.3 and that it is obvious from its definition that *Aug* will preserve identities and distribute over composition. We therefore have that *Aug* is a functor. □

Theorem 5.1.5. *The functor $Aug : \mathcal{A} \to \mathcal{R}$ is right adjoint to the inclusion functor $I : \mathcal{R} \to \mathcal{A}$.*

Proof. Define the family of operations ϵ by

$$\epsilon_E(x,\ \leq_x) \overset{def}{=} max(x,\ \leq_x)$$

for all event structures E and $(x,\ \leq_x) \in Aug(E)$.

We now show that $\epsilon_E : Aug(E) \to E$ is an augmentation morphism.

We first show $[\epsilon_E(x,\ \leq_x)] \subseteq \epsilon_E[(x,\ \leq_x)]$ for $(x,\ \leq_x) \in Aug(E)$. Suppose $e \leq \epsilon_E(x,\ \leq_x)$ for some $e \in E$. Then, as $e \leq max(x,\ \leq_x)$, we know that e must be in x as $(x,\ \leq_x)$ is an augmentation of E. The smallest restriction of $(x,\ \leq_x)$ that contains e will be a prime elementary event structure and, from the definition of $Aug(E)$, will be a member of $[(x,\ \leq_x)]$. Also, its maximum element will be e. We therefore have $e \in \epsilon_E[(x,\ \leq_x)]$. So, we have

$$[\epsilon_E(x,\ \leq_x)] \subseteq \epsilon_E[(x,\ \leq_x)].$$

Let $(x_1,\ \leq_1)$ and $(x_2,\ \leq_2)$ be events in $Aug(E)$.

We now show that ϵ_E reflects conflict. If $\epsilon_E(x_1,\ \leq_1)$ is in conflict with $\epsilon_E(x_2, \leq_2)$ then we have $max(x_1,\ \leq_1)\#max(x_2,\ \leq_2)$ and therefore $x_1 \not\uparrow x_2$. So, $(x_1, \leq_1)\#(x_2,\ \leq_2)$ in $Aug(E)$. Hence, ϵ_E reflects conflict.

Finally, we demonstrate that $\epsilon_E(x_1,\ \leq_1) = \epsilon_E(x_2,\ \leq_2)$ implies $(x_1,\ \leq_1) = (x_2,\ \leq_2)$ or $(x_1,\ \leq_1)\#(x_2,\ \leq_2)$. Assume that $\epsilon_E(x_1,\ \leq_1) = \epsilon_E(x_2,\ \leq_2)$ and $(x_1,\ \leq_1) \neq (x_2,\ \leq_2)$. So, we have $x_1 \neq x_2$ or $\leq_1 \neq \leq_2$. We know $x_1 \neq x_2$ implies $\leq_1 \neq \leq_2$. Also, $x_1 \cap x_2$ is non-empty because $max(x_1,\ \leq_1)$ equals $max(x_2,\ \leq_2)$ and, by the assumption $(x_1,\ \leq_1) \neq (x_2,\ \leq_2)$, it cannot be the case that x_1 or x_2 equals $\emptyset$. So, there must exist e in $x_1 \cap x_2$ for which $[e]$ for $(x_1,\ \leq_1)$ is not equal to $[e]$ for $(x_2,\ \leq_2)$. From the definition of $Aug(E)$, it follows that $(x_1,\ \leq_1)\#(x_2,\ \leq_2)$.

We now demonstrate $I \dashv Aug$ by showing the following. Let $f : E' \rightarrow E$ be a morphism in $\mathcal{A}$. We show that the diagram below commutes for a unique rigid morphism $f^{\#}$.

$$\begin{tikzcd}
Aug(E) \arrow[r, "\epsilon_E"] & E \\
E' \arrow[u, dotted, "f^{\#}"] \arrow[ur, "f"'] &
\end{tikzcd}$$

Define $f^{\#} : E' \rightarrow Aug(E)$ by

$$f^{\#}(e) \overset{def}{=} (f[e],\ f(\leq_e))$$

for all $e \in E'$ where $\leq_e$ is the restriction of the causality relation of E' to $[e]$.

We show that $f^{\#}$ is indeed a rigid morphism. We first demonstrate that $[f^{\#}(e)] = f^{\#}[e]$. Let e_1 and e_2 be events in E'. If $e_1 \leq e_2$ then $[e_1] \subseteq [e_2]$ and also $\leq_{e_1} \subseteq \leq_{e_2}$. This implies $(f[e_1],\ f(\leq_{e_1})) \leq (f[e_2],\ f(\leq_{e_2}))$ in $Aug(E)$ and therefore $f^{\#}(e_1) \leq f^{\#}(e_2)$. So we have

$$f^{\#}[e] \subseteq [f^{\#}(e)]$$

for all $e \in E'$.

Suppose we have $(x,\ \leq_x) \leq f^{\#}(e_1)$. Then, from the definition of causality for $Aug(E)$, $(x,\ \leq_x)$ is a restriction of $(f[e_1],\ f(\leq_{e_1}))$. As f is injective on $[e_1]$ we know that $(f[e_1],\ f(\leq_{e_1}))$ is isomorphic to $([e_1],\ \leq_{e_1})$ and therefore there is a restriction $(x',\ \leq_{x'})$ of $([e_1],\ \leq_{e_1})$ for which $(f(x'),\ f(\leq_{x'})) = (x,\ \leq_x)$. So, $f^{\#}(max(x',\ \leq'_x)) = (x,\ \leq_x)$. Also, as $(x',\ \leq_{x'})$ is prime and a restriction of $([e_1],\ \leq_{e_1})$, we know $max(x',\ \leq_{x'}) \leq e_1$. It follows that $(x,\ \leq_x) \in f^{\#}[e_1]$ and therefore

$$[f^{\#}(e)] \subseteq f^{\#}[e]$$

for all $e \in E'$.

We next show that $f^{\#}$ reflects conflict. Assume $f^{\#}(e_1)\#f^{\#}(e_2)$. Then $f[e_1] \not\uparrow f[e_2]$ as $\leq_{e_1}$ and $\leq_{e_2}$ are restrictions of the same causality relation. As f reflects conflict, this implies $[e_1] \not\uparrow [e_2]$ and therefore $e_1\#e_2$. So, conflict is reflected.

Finally, we show that $f^{\#}(e_1) = f^{\#}(e_2)$ implies $e_1 = e_2$ or $e_1\#e_2$. Assume $f^{\#}(e_1) = f^{\#}(e_2)$ and $e_1 \neq e_2$. As $f[e_1] = f[e_2]$ but $[e_1] \neq [e_2]$ we know that f is mapping an event in $[e_1]$ and a different event in $[e_2]$ to the same thing. From the properties of f, this implies that these events are in conflict and therefore $[e_1] \not\uparrow [e_2]$. So, we have $e_1\#e_2$.

Clearly we have $f = \epsilon_E \circ f^{\#}$. Having proved that $f^{\#} : E' \to Aug(E)$ is a rigid morphism for which the diagram commutes, we now show that it is unique. Suppose that the diagram commutes for another rigid morphism $g : E' \to Aug(E)$. We show by induction on the number of causal dependencies of events in E' that $g = f^{\#}$.

Let e be an event in E'.

Base Case: Assume $|[e]| = 1$. As g is rigid and $max(g(e)) = f(e)$ we have $g(e) = (\{f(e)\},\ \{(f(e),\ f(e))\})$ as this is the only event in $Aug(E)$ that ϵ_E maps to $f(e)$ and that only depends on itself. So, $g(e) = f^{\#}(e)$.

Inductive Step: Assume $g(e') = f^{\#}(e')$ for all events e' in E' for which we have $1 \leq |[e']| \leq k$. Suppose $|[e]| = k + 1$. From the assumption, $g(e)$ and $f^{\#}(e)$ must depend on precisely the same events as g and $f^{\#}$ are both rigid and $e' < e$ implies $g(e') = f^{\#}(e')$. We also know that $max(g(e)) = max(f^{\#}(e)) = f(e)$. This uniquely defines an event in $Aug(E)$, given by the union of the events and causality relations of all the events upon which $g(e)$ and $f^{\#}(e)$ depend with $f(e)$ added as the maximum event. To see that this is so, recall that, for an event $(x,\ \leq_x)$ in $Aug(E)$, we have that $[(x,\ \leq_x)]$ is the set of prime restrictions of $(x,\ \leq_x)$. This implies $|[(x,\ \leq_x)]| = |x|$. So, $(x,\ \leq_x)$ can be defined as the union of all its restrictions not equal to it with $max(x,\ \leq_x)$ added as the maximum event. So, as g is rigid, there is only one event in $Aug(E)$ to which it can map the event e. This event is $f^{\#}(e)$.

By induction, $g = f^{\#}$ and therefore $f^{\#}$ is unique. □

Using this adjunction we can construct a monad $(Aug \circ I, \eta,\ Aug \circ \epsilon \circ I)$ where η is the unit of the adjunction and ϵ is the counit. We overload Aug to be this monad as well as the right adjoint.

Recall that the Kleisli category $\mathcal{C}_T$ of a monad $(T,\ \eta,\ \mu)$ on a category $\mathcal{C}$ is defined to have the objects of $\mathcal{C}$ and morphisms $f : A \to B$ where f is a morphism between A and $T(B)$ in $\mathcal{C}$. The composition of a morphism $f : A \to B$ and $g : B \to C$ in $\mathcal{C}_T$ is defined to be $\mu \circ T(g) \circ f$. (See page 147 of [22] for further details.)

Proposition 5.1.6. *Consider an adjunction $F \dashv G$ where F is a functor from $\mathbf{A}$ to $\mathbf{B}$ that is identity on objects, (i.e., $\mathbf{A}$ and $\mathbf{B}$ contain the same objects and F maps objects to themselves). If $T = (GF,\ \eta,\ G\epsilon F)$ then the comparison functor $\mathbf{A}_T \to \mathbf{B}$ is an isomorphism.* □

Proposition 5.1.6 is a special case of Exercise 2, page 148 of [22].

Corollary 5.1.7. *The Kleisli category $\mathcal{R}_{Aug}$ of Aug is isomorphic to $\mathcal{A}$.* □

So, augmentations can be described as certain rigid morphisms.

There are clearly functors $I_{\mathcal{D}} : \mathcal{R} \to \mathcal{D}$ and $I_{\mathcal{P}} : \mathcal{R} \to \mathcal{P}$ where $I_{\mathcal{P}}$ maps objects and arrows to themselves and $I_{\mathcal{D}}$ maps objects to themselves and the action of $I_{\mathcal{D}}$ on morphisms is defined by $I_{\mathcal{D}}(f)(e) \overset{def}{=} f[e]$ for $f : E_1 \to E_2$ in $\mathcal{R}$ and $e \in E_1$.

Proposition 5.1.8. *There are no right adjoints to $I_{\mathcal{D}}$ and $I_{\mathcal{P}}$.*

Proof. We show that $\mathcal{D}$ and $\mathcal{P}$ both have terminal objects whereas $\mathcal{R}$ does not. As any right adjoints to I_D and $I_{\mathcal{P}}$ must preserve limits, this implies that they cannot exist.

The terminal object in both $\mathcal{D}$ and $\mathcal{P}$ will be the empty event structure $\emptyset$. If E is an event structure then the unique arrow in $\mathcal{D}$ from E into $\emptyset$ will map e to $\emptyset$ for all $e \in E$ and the unique arrow in $\mathcal{P}$ will be undefined for all $e \in E$.

Suppose there exists a terminal object E in $\mathcal{R}$. Then there must be a unique morphism f from the event structure E' containing two independent events e_1 and e_2 into it. As e_1 and e_2 are not in conflict and are distinct, $f(e_1) \neq f(e_2)$. Also, as f reflects conflict, $f(e_1)$ and $f(e_2)$ must be independent. However, this means that there exists another morphism $g : E' \to E$ in $\mathcal{R}$ that maps e_1 to $f(e_2)$ and e_2 to $f(e_1)$. So, f is not unique and therefore the terminal object cannot exist. □

Observe that this implies that, if $I'_{\mathcal{P}}$ is the inclusion functor from $\mathcal{A}$ to $\mathcal{P}$ that maps objects and morphisms to themselves, there can be no right adjoint to $I'_{\mathcal{P}}$.

If we wish such right adjoints to exist, we must consider altering the definition of event structures. An initial idea was to allow *independence events.*

Definition 5.1.9. *Define $(E,\ I,\ \leq,\ \#)$ to be an event structure with independence with set of events E and set of independence events I if*

- *$I \subseteq E$,*
- *$(E,\ \leq,\ \#)$ obeys the conditions listed in Definition 2.1.1,*
- *$e_1 \# e_2$ implies e_1 and e_2 are not members of I and*
- *$e_1 < e_2$ implies e_1 and e_2 are not members of I.*

Observe that the events in I in the above definition may be thought of as being completely causally unrelated to and consistent with all other events in the event structure.

We can define categories $\mathcal{R}_I$, $\mathcal{A}_I$, $\mathcal{P}_I$ and $\mathcal{D}_I$ with event structures as objects and where the morphisms $f : E_1 \to E_2$ have the properties defined previously in Section 2.1.1 but where f preserves independence events and

$$f(e_1) = f(e_2) \ \Rightarrow\ f(e_1) \text{ is an independence event}$$

holds for $e_1,\ e_2 \in E_1$ where $e_1 \neq e_2$ and $\neg(e_1 \# e_2)$.

It is routine to verify that the categories $\mathcal{R}_I$ and $\mathcal{A}_I$ have a terminal object consisting of a single independence event.

It is now possible to define an adjunction between $\mathcal{A}_I$ and $\mathcal{P}_I$.

We define a functor $Ptl : \mathcal{P}_I \to \mathcal{A}_I$ as follows.

Definition 5.1.10. *Let the action of Ptl on objects be defined by*

$$Ptl(E,\ I,\ \leq,\ \#) \overset{def}{=} (E \uplus \{\star\},\ I \uplus \{\star\},\ \leq',\ \#).$$

For clarity, we assume that $\star$ *is not a member of* E *and therefore write* $(1,\ e)$ *as* e *and* $(2,\ \star)$ *as* $\star$ *and define* $\leq'$ *to be* $\leq \cup\{(\star,\ \star)\}$. *Let the action of Ptl on morphisms be defined by*

$$\begin{aligned} Ptl(f)(e) &\overset{def}{=} f(e) \text{ if } f(e) \text{ defined} \\ &\overset{def}{=} \star \text{ otherwise.} \end{aligned}$$

for all $f : E_1 \to E_2$ *in* $\mathcal{P}_I$.

Proposition 5.1.11. *The operation Ptl is a functor.*

Proof. That $Ptl(E)$ is an event structure for all event structures in $\mathcal{P}_I$ follows directly from its definition.

Let $f : E_1 \to E_2$ be a morphism in $\mathcal{P}_I$. We now show that $Ptl(f)$ is a morphism from $Ptl(E_1)$ to $Ptl(E_2)$ in $\mathcal{A}_I$. Clearly, from its definition, $Ptl(f)$ is total. We first show $[Ptl(f)(e)] \subseteq Ptl(f)[e]$ for all $e \in Ptl(E_1)$. Suppose $e_1 \leq Ptl(f)(e)$ for some $e \in E_1$. Then there are three cases to consider – $e = \star$, $e \neq \star$ and f is defined for e, $e \neq \star$ and f is undefined for e. If $e \neq \star$ and f is defined for e we have that $e_1 \leq f(e)$ which, given the properties of f, implies $e_1 \in f[e]$ and therefore $e_1 \in Ptl(f)[e]$. Otherwise, $Ptl(f)(e) = \star$ and so $e_1 = \star$ and is therefore trivially a member of $Ptl(f)[e]$. So, we have

$$[Ptl(f)(e)] \subseteq Ptl(f)[e]$$

for all $e \in Ptl(E_1)$.

We next show that $Ptl(f)$ preserves independence. Suppose e is an independence event in $Ptl(E_1)$. If $e = \star$ then $Ptl(f)(e)$ is defined to be $\star$ which is an independence event. Otherwise, as f must preserve independence where defined, $Ptl(f)(e)$ must also produce an independence event. Hence, independence is preserved.

Let e_1 and e_2 be events in $Ptl(E_1)$.

We show that $Ptl(f)$ reflects conflict. Suppose $Ptl(f)(e_1)\#Ptl(f)(e_2)$. From the properties of event structures with independence, this implies that neither $Ptl(f)(e_1)$ nor $Ptl(f)(e_2)$ can be $\star$. So, $f(e_1)\#f(e_2)$ in E_2 and therefore $e_1\#e_2$ in E_1. From the definition of Ptl, this implies that we have $e_1\#e_2$ in $Ptl(E_1)$. So, conflict is reflected.

Finally, we show that $Ptl(f)(e_1) = Ptl(f)(e_2)$ implies $e_1 = e_2$ or $e_1\#e_2$ or $Ptl(f)(e_1)$ is an independence event. Suppose $Ptl(f)(e_1) = Ptl(f)(e_2)$ and $e_1 \neq e_2$. If $Ptl(f)(e_1) \neq \star$ then $f(e_1)$ and $f(e_2)$ must be defined and therefore the correct property follows from the properties of f. Otherwise, $\star$ is an independence event as required.

It is trivial to check that Ptl preserves identities and distributes over composition. □

Theorem 5.1.12. *The functor* $Ptl : \mathcal{P}_I \to \mathcal{A}_I$ *is right adjoint to the inclusion functor from* $\mathcal{A}_I$ *into* $\mathcal{P}_I$.

Proof. Define $\epsilon_E : \mathit{Ptl}(E) \to E$ by

$$\epsilon_E(e) \overset{def}{=} e \text{ if } e \neq \star$$

for all event structures E and e in $\mathit{Ptl}(E)$. It is clear that ϵ_E is a partial morphism as it behaves exactly like the identity morphism except for being undefined for $\star$.

In order to show $I \dashv \mathit{Ptl}$, we demonstrate the following. Let $f : E' \to E$ be a morphism in $\mathcal{P}_I$. We show that the following diagram commutes for a unique augmentation morphism $f^{\#} : E' \to \mathit{Ptl}(E)$.

$$\begin{tikzcd}
\mathit{Ptl}(E) \arrow[r, "\epsilon_E"] & E \\
E' \arrow[u, dotted, "f^{\#}"] \arrow[ur, "f"'] &
\end{tikzcd}$$

Define $f^{\#}$ by

$$\begin{aligned} f^{\#}(e) &\overset{def}{=} f(e) \text{ if } f(e) \text{ is defined} \\ &\overset{def}{=} \star \text{ otherwise} \end{aligned}$$

for all $e \in E'$. That $f^{\#}$ is indeed an augmentation morphism follows from the properties of f and the independence of $\star$.

Clearly $f = \epsilon_E \circ f^{\#}$. We now show that it is the unique augmentation morphism for which this is true. If this was true for another morphism f' then we would have $\epsilon_E \circ f' = \epsilon_E \circ f^{\#}$. As ϵ_E is injective in the sense that it maps all events to themselves except for $\star$ for which it is undefined, this implies that $f' = f^{\#}$ and therefore that $f^{\#}$ is unique. So, *Ptl* is right adjoint to the inclusion functor.

□

However, we will now show that the addition of independence events means that there is no right adjoint to the inclusion functor from $\mathcal{R}_I$ to $\mathcal{A}_I$.

Theorem 5.1.13. *Let $I_{\mathcal{A}_I} : \mathcal{R}_I \to \mathcal{A}_I$ be the functor that maps objects and morphisms in $\mathcal{R}_I$ to themselves in $\mathcal{A}_I$. There is no right adjoint to $I_{\mathcal{A}_I}$.*

Proof. Suppose that a right adjoint $G : \mathcal{A}_I \to R_I$ to $I_{\mathcal{A}_I}$ exists and we will derive a contradiction.

Let E be the event structure consisting of two independence events i_1 and i_2. Then

$$\mathcal{A}_I(E,\ E) \cong \mathcal{R}_I(E,\ G(E)).$$

There are four different augmentation morphisms from E to E so

$$|\mathcal{R}_I(E,\ G(E))| = |\mathcal{A}_I(E,\ E)| = 4.$$

As independence must be preserved, this implies that $G(E)$ must contain at least two independence events.

Consider the event structure E' that consists of two independence events and another non-independence event. Then $|\mathcal{A}_I(E',\ E)| = 8$. Suppose $G(E)$ contains another event e in addition to the two independence events. Then there are more than eight rigid morphisms from E' into $G(E)$ and so $|\mathcal{R}_I(E',\ G(E))| \neq |\mathcal{A}_I(E',\ E)|$, contradicting the initial assumption. It follows that, if G exists, $G(E)$ is the event structure only containing two independence events.

Let E'' be

$$\begin{tikzcd} e_2 \\ e_1 \arrow[u] \end{tikzcd} \quad .$$

Then, $|\mathcal{A}_I(E'',\ E)| = 4$ and $|\mathcal{R}_I(E'',\ G(E))| = 2$. This contradicts the assumption that a right adjoint exists. □

Observe that the lack of existence of a right adjoint is caused by disallowing causal relationships between independence events and other events. This suggests a further extension to the definition of event structure which we explore in Section 5.4. Before this, we describe another justification for extending event structures. To do this we must first describe the presheaves that are represented by labelled event structures.

5.2 Labelled Event Structures as Presheaves

In this section we recall that labelled event structures represent certain kinds of presheaf. We use this relationship in the following section to demonstrate some of the behaviours that presheaves are capable of expressing but that cannot be captured by event structures.

Recall that a presheaf is simply a contravariant functor into the category of sets. Presheaves over a path category can be used to model concurrent processes [8]. With the correct choice of path category, we can form a category of presheaves for which the category of labelled event structures is isomorphic to a sub-category.

Definition 5.2.1. *Let* $\mathbf{Pom}_L$ *be the skeleton of the category of labelled elementary event structures with labelling set L and rigid, label-preserving morphisms (see Definition 2.1.2 and Section 2.1.2).*

We refer to the objects in $\mathbf{Pom}_L$ as *pomsets* – partially ordered multisets. We make use of a shorthand notation when reasoning about pomsets, i.e., an event's label will be used instead of the name of the event itself where it is clear from the context which event is being discussed.

Let $\widehat{\mathbf{Pom}_L}$ be the category of presheaves over the category $\mathbf{Pom}_L$. An object in $\widehat{\mathbf{Pom}_L}$ represents a labelled event structure E if it is isomorphic to the presheaf

$$\widehat{E} : {\mathbf{Pom}_L}^{op} \to \mathbf{Set}$$

where $\overset{\frown}{E}$ is defined by

$$\begin{aligned} \overset{\frown}{E}(x) &\overset{def}{=} \mathcal{R}_L(x,\ E) \\ \overset{\frown}{E}(f)(g) &\overset{def}{=} g \circ f \end{aligned}$$

where $f : p \to q$ is a morphism in $\mathbf{Pom}_L$ and $g \in \overset{\frown}{E}(q)$.

In [38], Winskel demonstrated that such presheaves $X : {\mathbf{Pom}_L}^{op} \to \mathbf{Set}$ possess the following two properties.
mono: For all morphisms $j_1,\ j_2 : p \to q$ in $\mathbf{Pom}_L$ where p is prime

$$\forall x \in X(q).X(j_1)(x) = X(j_2)(x) \ \Rightarrow\ j_1 = j_2.$$

separated: For all $x,\ x' \in X(q)$

$$(\forall j : p \to q \text{ such that } p \text{ is prime } X(j)(x) = X(j)(x')) \ \Rightarrow x = x'.$$

5.3 Weak Bisimulation from Open Maps

In this section, we recall a categorical characterisation of weak bisimulation in terms of open maps and demonstrate a curious discrepancy where event structures are concerned.

In [17], the notion of a bisimulation being a span of open maps is described and shown to correspond to strong, history-preserving bisimulation for event structures. Fiore, Winskel and Cattani extended this idea to give a categorical description of weak bisimulation and observational congruence [14]. Firstly, presheaves are generalised to *bundles* via the Grothendieck fibration. A bundle is an object in $Cat/\mathbb{P}$ for some category $\mathbb{P}$. We now give an intuition into the way in which a bundle models a process. Let $J : \mathcal{C} \to \mathbb{P}$ be a bundle in $Cat/\mathbb{P}$. We can view the objects of $\mathcal{C}$ as computation states and the morphisms as transitions between states. The category $\mathbb{P}$ may be viewed as representing possible observations. For example, let x be an object in $\mathcal{C}$, i.e., a computation state. Then $J(x)$ represents what has been observed at that point in the computation.

Let $\mathbb{P}$ and $\mathbb{Q}$ be categories of observations. Let $H : \mathbb{P} \to \mathbb{Q}$ be a functor. Then H induces a functor $(H \circ \_) : Cat/\mathbb{P} \to Cat/\mathbb{Q}$. Both the Grothendieck fibration and $(H \circ \_)$ have right adjoints.

$$\begin{tikzcd} \hat{\mathbb{P}} \arrow[r, bend left, "F"] & Cat/\mathbb{P} \arrow[l, bend left, "G"] \arrow[r, bend left, "hide"] & Cat/\mathbb{Q} \arrow[l, bend left, "sat"] \end{tikzcd}$$

(Where $\hat{\mathbb{P}}$ is the category of presheaves over $\mathbb{P}$, F is the Grothendieck fibration and $hide = (H \circ \_)$.)

Let $T : \hat{\mathbb{P}} \to \hat{\mathbb{P}}$ be the monad on $\hat{\mathbb{P}}$ given by the composition of these functors. Then, in order to determine whether two presheaves X and Y are weakly bisimilar, it is sufficient to check that there is a span of open maps between $T(X)$ and $T(Y)$ (see [17] for more details about open maps). We provide more precise definitions of these functors below.

First, we recall the Grothendieck construction.

Definition 5.3.1. *Let X be an object in $\hat{\mathbb{P}}$. Then $\mathcal{E}l(X)$ is the category with objects $(x,\ y)$ where $x \in \mathbb{P}$ and $y \in X(x)$. Define the morphisms $f : (x_1,\ y_1) \to (x_2,\ y_2)$ in $\mathcal{E}l(X)$ to be morphisms $f : x_1 \to x_2$ in $\mathbb{P}$ such that $X(f)(y_2) = y_1$.*

Further details on this construction can be found in [14].

The functor F maps a presheaf $X \in \hat{\mathbb{P}}$ to the bundle in $Cat/\mathbb{P}$ given by the functor $\pi_X : \mathcal{E}l(X) \to \mathbb{P}$ that maps $(p,\ x) \in \mathcal{E}l(X)$ to p and morphisms $f : (p_1,\ x_1) \to (p_2,\ x_2)$ to the underlying morphism from p_1 to p_2. Its right adjoint $G : Cat/\mathbb{P} \to \hat{\mathbb{P}}$ is defined by

$$G(\pi : Y \to \mathbb{P})(p) \ = \ Cat/\mathbb{P}(\mathbb{P}/p \to \mathbb{P}, \pi : Y \to \mathbb{P})$$

for a bundle π in $Cat/\mathbb{P}$ and $p \in \mathbb{P}$.

Let $\mathbb{P}$ and $\mathbb{Q}$ be categories of pomsets with rigid morphisms and labelling sets L and L' such that L' is a subset of L. Having defined the adjunction $F \dashv G$, we now define the adjunction $hide \dashv sat$ for our particular choice of $\mathbb{P}$ and $\mathbb{Q}$. The functor $hide : Cat/\mathbb{P} \to Cat/\mathbb{Q}$ acts to remove invisible events. We define $hide$ in terms of another functor $H : \mathbb{P} \to \mathbb{Q}$. Let $(x,\ \leq,\ l) \in \mathbb{P}$. Define the action of the functor $H : \mathbb{P} \to \mathbb{Q}$ on objects by

$$H(x,\ \leq,\ l) \ \overset{def}{=} \ (x',\ \leq',\ l')$$

where $x' = \{e \in x \mid l(e) \in l'\}$ and $\leq'$ and l' are the corresponding restrictions of $\leq$ and l. Its action on morphisms $f : (x,\ \leq,\ l) \to (x',\ \leq',\ l')$ in $\mathbb{P}$ is to restrict their domain to x'. Let $\pi : Y \to \mathbb{P}$ be a bundle in $Cat/\mathbb{P}$. The functor $hide : Cat/\mathbb{P} \to Cat/\mathbb{Q}$ is defined by

$$hide(\pi : Y \to \mathbb{P}) \ \overset{def}{=} \ H \circ \pi.$$

Its right adjoint sat takes a bundle in $Cat/\mathbb{Q}$ and pulls it back along H to form a bundle in $Cat/\mathbb{P}$. Using these adjunctions, we can define the monad T on $\hat{\mathbb{P}}$ where the functor $T : \hat{\mathbb{P}} \to \hat{\mathbb{P}}$ of the monad is defined by

$$T \ \overset{def}{=} \ G \circ sat \circ hide \circ F.$$

When T is applied to a presheaf that represents an event structure, the resulting presheaf may not represent an event structure, i.e., it may be capable of behaviour that cannot be expressed in terms of an event structure. We give two examples of event structures for which this is the case together with an intuition for the additional behaviours introduced.

For the examples, let $\mathbb{P}$ be the category of pomsets and rigid, label-preserving morphisms with labelling set $L \cup \{\tau\}$ where $\tau \notin L$ and let $\mathbb{Q}$ be the sub-category of $\mathbb{P}$ with labelling set L.

Example 1: The empty event structure $\emptyset$.

The presheaf $\widehat{\emptyset} \in \hat{\mathbb{P}}$ representing this event structure produces a singleton when given the empty pomset and gives $\emptyset$ for all other arguments. Applying T to $\widehat{\emptyset}$ produces a presheaf X that gives a singleton for any pomset consisting entirely of τ labelled events. Consider the two possible morphisms j_1 and $j_2 : \tau \rightarrow (\tau\tau)$ in $\mathbb{P}$. While these are distinct, it is the case that $X(j_1) = X(j_2)$ and therefore the (*mono*) property is violated.

The presheaf $T(\widehat{\emptyset})$ may be thought of as describing a process that can do the same τ event multiple independent times. It is clear that such a process cannot be expressed as an event structure.

Example 2: The event structure E defined to be

$$e_1 \quad e_2 \quad e_3 \# e_4$$

where e_4 is labelled τ and the other three events are labelled a, b and c respectively. The set $T(\widehat{E})(ab)$ contains two items x and y where x represents the occurrence of e_1 and e_2, and y the occurrence of e_1, e_2 and e_4. There are morphisms $f_1 : a \rightarrow (ab)$ and $f_2 : b \rightarrow (ab)$ in $\mathbb{P}$ from prime pomsets into ab. Observe that

$$\begin{aligned} T(\widehat{E})(f_1)(x) &= T(\widehat{E})(f_1)(y) \text{ and} \\ T(\widehat{E})(f_2)(x) &= T(\widehat{E})(f_2)(y) \end{aligned}$$

but x and y are distinct. This demonstrates that $T(\widehat{E})$ does not possess the (*separated*) property.

The additional behaviour beyond that of an event structure can be seen by considering whether or not e_3 can occur after observing two events labelled a and b. The state of an event structure, i.e., its future capabilities, is determined entirely by its history. This is not the case with the process represented by $T(\widehat{E})$.

This provides yet another justification for extending the definition of event structures. In the next section, we describe event structures with *persistence* in an effort to capture some of the extra required behaviour.

5.4 Event Structures with Persistence

In this section we describe event structures with persistence. The definitions of rigid, augmentation, partial and demand morphism are extended accordingly. As will be shown in Section 5.5 and Chapter 6, unlike for classical event structures, the categories of augmentation, partial and demand morphisms are isomorphic to Kleisli categories of monads on the category of rigid morphisms.

Definition 5.4.1. *Define an event structure with persistence to be a tuple*

$$(E,\ P,\ \leq,\ \#)$$

where $(E,\ \leq,\ \#)$ is an event structure as defined in Definition 2.1.1 and $P \subseteq E$ is a set of persistent *events.*

The configurations for an event structure with persistence $(E,\ P,\ \leq,\ \#)$ are to be the configurations of $(E,\ \leq,\ \#)$ (see Definition 2.1.3).

As for the other types of event structure, we can define partial, augmentation, rigid and demand morphisms between event structures with persistence.

Definition 5.4.2. *Let $f : E_1 \to E_2$ be a partial function. It is a partial morphism $f : (E_1,\ P_1,\ \leq_1,\ \#_1) \to (E_2,\ P_2,\ \leq_2,\ \#_2)$ if*

i) $e \in P_1$ and $f(e)$ defined $\Rightarrow\ f(e) \in P_2$, i.e., f preserves persistent events where defined,

ii) if $f(e)$ and $f(e')$ are defined then

$$\begin{aligned} f(e) = f(e') \text{ or } f(e)\#f(e') \ \Rightarrow& \\ e = e' \text{ or } e\#e' \text{ or } f(e) = f(e') \in P_2,& \end{aligned}$$

iii) if $f(e)$ defined then $[f(e)] \subseteq f\,[e]$.

Let $\mathcal{P}_P$ be the category of event structures with persistence and partial morphisms.

Definition 5.4.3. *Let f be an augmentation morphism between the event structures with persistence $(E_1,\ P_1,\ \leq_1,\ \#_1)$ and $(E_2,\ P_2,\ \leq_2,\ \#_2)$ if it is a partial morphism between them whose underlying function is total.*

Let $\mathcal{A}_P$ be the category of event structures with persistence and augmentation morphisms.

Definition 5.4.4. *Let f be a rigid morphism between $(E_1,\ P_1,\ \leq_1,\ \#_1)$ and $(E_2,\ P_2,\ \leq_2$ $,\ \#_2)$ if it is an augmentation morphism between them such that*

$$[f(e)] = f[e].$$

Let $\mathcal{R}_P$ be the category of event structures with persistence and rigid morphisms.

Again, composition in $\mathcal{P}_P$, $\mathcal{A}_P$ and $\mathcal{R}_P$ is simply function composition.

Lemma 5.4.5. *A rigid morphism $f : (E,\ P,\ \leq,\ \#) \to (E',\ P',\ \leq',\ \#')$ between event structures with persistence is a rigid isomorphism iff it is a bijection and it reflects persistence and preserves conflict.*

Proof. We can use the same argument as that in the proof of Lemma 2.1.8 to show that f can be viewed as an isomorphism between the event structures without persistence given by $(E,\ \leq,\ \#)$ and $(E',\ \leq',\ \#')$. This argument gives us a definition of a possible inverse f' of f. In order to show that f^{-1} is a rigid morphism between $(E',\ P',\ \leq',\ \#')$ and $(E,\ P,\ \leq,\ \#)$ it remains to show f^{-1} preserves persistence. This follows immediately from f reflecting persistence. So, f is an isomorphism between $(E,\ P,\ \leq,\ \#)$ and $(E',\ P',\ \leq',\ \#')$. □

Definition 5.4.6. *Let d be a demand morphism between event structures with persistence $(E_1,\ P_1,\ \leq_1,\ \#_1)$ and $(E_2,\ P_2,\ \leq_2,\ \#_2)$ iff it is a demand morphism between $(E_1,\ \leq_1,\ \#_1)$ and $(E_2,\ \leq_2,\ \#_2)$ in $\mathcal{D}$.*

Composition of demand morphisms is defined in the same way as for morphisms in $\mathcal{D}$, i.e., the composition of two demand morphisms $d_1 : E \rightarrow E'$ and $d_2 : E' \rightarrow E''$ is $d_2^\dagger \circ d_1$.

Let $\mathcal{D}_P$ be the category of event structures with persistence and demand morphisms.

Observe that $\mathcal{D}$ is in fact equivalent to $\mathcal{D}_P$.

It is trivial to see that, for all the categories, identities exist and composition is defined and associative. The constructions detailed in Section 2.3 can easily be extended to constructions on event structures with persistence. For the next chapter, it is particularly relevant to note that restriction can be defined exactly as for classical event structures.

In order to gain an intuition for the extra behaviour captured by event structures with persistence, observe that the key difference is the loss of local injectivity of event structure morphisms, i.e., two non-conflicting events can be mapped to the same event if it is persistent. We can therefore interpret persistent events as being in some sense repeatable as, viewing morphisms as simulations, one persistent event can simulate multiple events, possibly at the same time. Alternatively, we can think of persistent events as happening continuously over a period as opposed to non-persistent events which happen instantaneously.

We can now define an adjunction between $\mathcal{A}_P$ and $\mathcal{P}_P$. In Section 5.6 we discuss the problems involved with defining adjunctions between $\mathcal{R}_P$ and $\mathcal{A}_P$ and $\mathcal{R}_P$ and $\mathcal{D}_P$ that will be the topic of Chapter 6.

5.5 An Adjunction between $\mathcal{A}_P$ and $\mathcal{P}_P$

It is clear that $\mathcal{A}_P$ is a subcategory of $\mathcal{P}_P$. We define a right adjoint $Par : \mathcal{P}_P \rightarrow \mathcal{A}_P$ to the inclusion functor $I : \mathcal{A}_P \rightarrow \mathcal{P}_P$.

Definition 5.5.1. *Let the action of Par on objects $(E,\ P,\ \leq,\ \#)$ in $\mathcal{P}_P$ be defined by*

$$Par(E,\ P,\ \leq,\ \#) \overset{def}{=} (E \uplus \{\star\},\ P \uplus \{\star\},\ \leq',\ \#').$$

For clarity, we assume that $\star \notin E$ and therefore treat $E \uplus \{\star\}$ as $E \cup \{\star\}$ and define $\leq'$ and $\#'$ such that, for $e,\ e' \in E \cup \{\star\}$,

- *$e \leq' e'$ iff $e \leq e'$ or $e = e' = \star$ and*
- *$e\#'e'$ iff $e\#e'$.*

Let the action of Par on morphisms be defined by

$$\begin{aligned} Par(f)(e) &\overset{def}{=} f(e) \text{ if } e \neq \star \text{ and } f(e) \text{ is defined} \\ &\overset{def}{=} \star \text{ otherwise.} \end{aligned}$$

for all $f : E_1 \rightarrow E_2$ in $\mathcal{P}_P$.

We could alternatively define $Par(E)$ to be the event structure $E \otimes \star$ (where we use $\star$ to represent the event structure that consists only of the independence event $\star$).

It is clear that $Par(E)$ is an event structure with persistence for all event structures with persistence E and therefore is a member of $\mathcal{A}_P$. Observe that this definition corresponds to that for event structures with independence events in that we are placing the original event structure in parallel with a completely independent persistent event. In fact, $\mathcal{P}_I$ is a subcategory of $\mathcal{P}_P$.

Lemma 5.5.2. *If $f : E_1 \to E_2$ is a morphism in $\mathcal{P}_P$ then*

$$Par(f) : Par(E_1) \to Par(E_2)$$

is a morphism in $\mathcal{A}_P$.

Proof. From the definition, $Par(f)$ is clearly total. We now show that $[Par(f)(e)]$ is a subset of $Par(f)[e]$ for all $e \in Par(E_1)$. Suppose $e' \in [Par(f)(e)]$. We must show $e' \in Par(f)[e]$. Now, either $e \in E_1$ or $e = \star$. In the former case, then if $f(e)$ is defined then $e' \in f[e]$ by the properties of partial morphisms. Therefore $e' \in Par(f)[e]$. Otherwise, if $e \in E_1$ but $f(e)$ is undefined or $e = \star$, then $Par(f)(e) = \star$. As $[\star] = \{\star\}$, we have that $e' = \star$ and e' is therefore a member of $Par(f)[e]$. So, $[Par(f)(e)] \subseteq Par(f)[e]$.

We now explain why f preserves persistence. Suppose that e is a persistent event in $Par(E_1,\ P_1,\ \leq_1,\ \#_1)$. Then either $e \in P_1$ or $e = \star$. In the first case, $f(e)$ is persistent or undefined according to the properties of partial morphisms. If $f(e)$ is persistent then $Par(f)(e)$ is persistent. If $f(e)$ is undefined then $Par(f)(e) = \star$ which is persistent. Finally, if $e = \star$ then $Par(f)(e) = \star$. So, persistence is preserved.

In the following, let e_1 and e_2 be events in $Par(E_1)$.

We show that conflict is reflected. Suppose $Par(f)(e_1)\#Par(f)(e_2)$ holds. Then, as it is defined that $\star$ is not in conflict with anything, $f(e_1)\#f(e_2)$ and therefore $e_1\#e_2$ in E_1 and in $Par(f)(E_1)$. Conflict is therefore reflected.

Suppose that $Par(f)(e_1) = Par(f)(e_2)$. We must show that either $e_1 = e_2$ or $e_1\#e_2$ or $Par(f)(e_1) \in P_2$. There are two cases to consider – $Par(f)(e_1) = \star$ and $Par(f)(e_1) \neq \star$. If $Par(f)(e_1) = \star$ then the property holds as $\star$ is persistent. If $Par(f)(e_1) \neq \star$ then $f(e_1)$ and $f(e_2)$ are defined so $f(e_1) = f(e_2)$ so, from the properties of partial morphisms, $e_1 = e_2$ or $e_1\#e_2$ or $f(e_1)$ is persistent. From the definition, if $f(e_1)$ is persistent then $Par(f)(e_1)$ is persistent. The condition is therefore satisfied. □

Proposition 5.5.3. *The operation Par is a functor.*

Proof. It is clear that, for all event structures E with persistence, $Par(E)$ is also an event structure with persistence and therefore a member of $\mathcal{A}_P$. From Lemma 5.5.2, if $f : E_1 \to E_2$ is a morphism in $\mathcal{P}_P$ then $Par(f)$ is a morphism between $Par(E_1)$ and $Par(E_2)$ in $\mathcal{A}_P$.

We show that Par preserves identities. Let $id_E : E \to E$ be the identity morphism for the event structure with persistence E in $\mathcal{R}_P$. It follows directly from the definition of Par that $Par(id_E)(e) = e$ for all $e \in Par(E)$ and so

$$Par(id_E) = id_{Par(E)}.$$

Let $f : E_1 \to E_2$ and $g : E_2 \to E_3$ be morphisms in $\mathcal{P}_P$. Then

$$\begin{aligned} Par(g \circ f)(e) &= g \circ f(e) \text{ if } g \circ f(e) \text{ is defined for } e \\ &= \star \text{ otherwise.} \end{aligned}$$

We show that $Par(g \circ f) = Par(g) \circ Par(f)$. There are three cases to consider: *i)* $e = \star$; *ii)* $e \neq \star$ but $f(e)$ is undefined; *iii)* $e \neq \star$ and $f(e)$ is defined. If *(i)* or *(ii)* then $Par(f)(e) = \star$ and $Par(g)(\star) = \star$ so $Par(g) \circ Par(f)(e) = \star$. If *(iii)* then $Par(g) \circ Par(f)(e) = Par(g)(f(e))$. So, $Par(g)(f(e)) = g \circ f(e)$ if g is defined for $f(e)$ or $\star$ otherwise. It follows that $Par(g) \circ Par(f) = Par(g \circ f)$ and therefore that $Par : \mathcal{P}_P \to \mathcal{A}_P$ is a functor. □

Theorem 5.5.4. *The functor $Par : \mathcal{P}_P \to \mathcal{A}_P$ is right adjoint to the inclusion functor $I : \mathcal{A}_P \to \mathcal{P}_P$.*

Proof. Let E be an event structure with persistence and let the family of partial morphisms ϵ be defined by

$$\begin{aligned} \epsilon_E(e) &\overset{def}{=} e \text{ if } e \neq \star \\ \epsilon_E(\star) & \quad \text{undefined.} \end{aligned}$$

Observe that $\epsilon_E : Par(E) \to E$ is indeed a morphism in $\mathcal{P}_P$ as $Par(E)$ is isomorphic to E placed in parallel with the event structure consisting of a single event $\star$ and ϵ_E maps all events to themselves in E except for $\star$.

To show $I \dashv Par$, we show that, if $f : E_1 \to E_2$ is a morphism in $\mathcal{P}_P$, then the diagram below commutes for a unique morphism $f^{\#}$ in $\mathcal{A}_P$.

$$\begin{tikzcd} E_2 & Par(E_2) \arrow[l, "\epsilon_{E_2}"'] \\ & E_1 \arrow[ul, "f"] \arrow[u, dotted, "f^{\#}"'] \end{tikzcd}$$

Indeed, let $f^{\#}$ be defined by

$$\begin{aligned} f^{\#}(e) &\overset{def}{=} f(e) \text{ if } e \neq \star \text{ and } f(e) \text{ is defined} \\ &\overset{def}{=} \star \text{ otherwise.} \end{aligned}$$

We now show that $f^{\#}$ is indeed an augmentation morphism. It follows immediately from the definition that $f^{\#}$ is a total function. First we show $[f^{\#}(e)] \subseteq f^{\#}[e]$ for all $e \in E_1$. Let e be an event in E_1. Consider $e' \in [f^{\#}(e)]$. We must show that $e' \in f^{\#}[e]$. There are two cases to consider – $f^{\#}(e) = f(e)$ and $f^{\#}(e) = \star$. If $f^{\#}(e) = \star$ then $e' = \star$ as $\star$ has no dependencies and so $e' \in f^{\#}[e]$. Otherwise $f^{\#}(e) = f(e)$ and $[f(e)]$ is a subset of $f[e]$ implying that $e' \in f^{\#}[e]$. So,

$$[f^{\#}(e)] \subseteq f^{\#}[e]$$

holds for all $e \in E_1$.

In the following, let $e_1,\ e_2 \in E_1$.

We now show that $f^{\#}$ reflects conflict. As $\star$ is not in conflict with anything, the holding of $f^{\#}(e_1)\#f^{\#}(e_2)$ implies $f(e_1)\#f(e_2)$ and therefore, as f reflects conflict, that $e_1\#e_2$ holds. Conflict is therefore reflected.

We next show that $f^{\#}(e_1) = f^{\#}(e_2)$ implies $e_1 = e_2$ or $\neg(e_1\#e_2)$ or $f^{\#}(e_1)$ is persistent. Suppose $f^{\#}(e_1) = f^{\#}(e_2)$ for distinct e_1 and e_2 in E_1. There are two cases to consider – $f^{\#}(e_1) = \star$ and $f^{\#}(e_1) = f(e_1)$. If $f^{\#}(e_1) = \star$ then, as $\star$ is defined to be persistent, the correct condition is met. Otherwise, $f^{\#}(e_1) = f(e_1) = f^{\#}(e_2) = f(e_2)$ and therefore, from the properties of f, either $e_1\#e_2$ holds or $f(e_1)$ is persistent, implying that $f^{\#}(e_1)$ is persistent as required.

We now show that persistence is preserved. Let e be a persistent event in E_1. Then either $f^{\#}(e) = \star$, a persistent event, or $f^{\#}(e) = f(e)$ and, as f preserves persistence, $f(e)$ is a persistent event, confirming the preservation of persistence by $f^{\#}$ and therefore that $f^{\#}$ is a morphism in $\mathcal{A}_P$.

Observe that ϵ_{E_2} is injective and therefore that $f^{\#}$ must be unique. It therefore follows that *Par* is right adjoint to I.

□

The result below follows from Proposition 5.1.6.

Corollary 5.5.5. *The Kleisli category of* $(Par \circ I,\ \eta,\ Par \circ \epsilon \circ I)$ *(where* η *is the unit of the adjunction) is isomorphic to* $\mathcal{P}_P$. □

5.6 Towards other Adjunctions

In this section we discuss the difficulties involved in constructing right adjoints to the inclusions of $\mathcal{R}_P$ into $\mathcal{A}_P$, and $\mathcal{R}_P$ into $\mathcal{D}_P$. This acts as a justification for the method described in the following chapter.

An augmentation x (as defined in Definition 5.1.1) of an event structure E without persistence may be viewed as a morphism $f : x \to E$ in $\mathcal{A}$ such that $f(e) = e$ for all $e \in x$. We extend this idea to define the augmentations of an event structure with persistence.

Definition 5.6.1. *An elementary event structure with persistence is a triple* $(x,\ y,\ \leq)$ *with* $y \subseteq x$ *such that* $(x,\ \leq)$ *is an elementary event structure. An* augmentation *of an event structure with persistence* E *is a pair*

$$((x,\ y,\ \leq),\ f)$$

where $(x,\ y,\ \leq)$ *is an elementary event structure with persistence and* f *is a morphism in* $\mathcal{A}_P$ *between* $(x,\ y,\ \leq)$ *and* E.

As before, an augmentation may be seen as adding causality between the events of E but now it is possible for an augmentation to have two events corresponding to the same event in E. For example, let E be the event structure consisting of a single persistent event e. The elementary event structure consisting of two persistent events e_1 and e_2 paired with a morphism mapping both e_1 and e_2 to e is an augmentation of E. Consider the event structure

$$e_1 \qquad e_2 \qquad e_3$$

where e_1 is persistent. The event structure

$$\begin{tikzcd}
e_1 & e'_1 \\
e_2 \arrow[u] & e_3 \arrow[u] ,
\end{tikzcd}$$

where e_1 and e'_1 are persistent, together with a morphism mapping e_1 and e'_1 to e_1, e_2 to e_2 and e_3 to e_3, is an augmentation. This is an example of how augmentations, in addition to having extra dependencies, can split persistent events.

Definition 5.6.2. *Let the rigid morphism $g : (x,\ y,\ \leq) \rightarrow (x',\ y',\ \leq')$ be a morphism between augmentations $((x,\ y,\ \leq),\ f)$ and $((x',\ y',\ \leq'),\ f')$ of E iff $f' \circ g = f$*

Define $\mathcal{A}_E$ to be the category with objects the augmentations of E and morphisms as above.

A *prime augmentation* is an augmentation $((x,\ y,\ \leq),\ f)$ where $(x,\ y,\ \leq)$ is a prime elementary event structure, i.e., x has a maximum event with respect to $\leq$.

As before, we wish to define our right adjoint in terms of prime augmentations. We wish to select those prime augmentations p with the property that, if x maps into p in $\mathcal{A}_E$ then it does so uniquely. To gain an intuition as to why this is required, consider the following. To obtain the correct universal property for the adjunction, we require that, if $Aug_P : \mathcal{A}_P \rightarrow \mathcal{R}_P$ is the right adjoint, each morphism $f : E_1 \rightarrow E_2$ in $\mathcal{A}_P$ induces a unique morphism $f' : E_1 \rightarrow Aug_P(E_2)$ in $\mathcal{R}_P$. So, there must be only one choice of prime augmentation that an event in E_1 can be mapped to for f' to be rigid. When attempting to develop a right adjoint to the functor from $\mathcal{R}_P$ into $\mathcal{D}_P$, a similar problem occurs. In the next chapter we describe a method of characterising such *extremal* elementary event structures and use them to define adjunctions between $\mathcal{R}_P$ and $\mathcal{A}_P$ and between $\mathcal{R}_P$ and $\mathcal{D}_P$.

Chapter 6

The Rigid-Pairs Method

In the previous chapter we discussed the difficulties involved in defining right adjoints to the inclusions of the category of event structures with persistence and rigid morphisms $\mathcal{R}_P$ into the categories with augmentation morphisms and demand morphisms $\mathcal{A}_P$ and $\mathcal{D}_P$. Also, if spans of rigid morphisms are to form a bicategory, we require $\mathcal{R}_P$ to have pullbacks. Without corresponding categories of stable families in which to construct limits, it is by no means obvious how to construct these pullbacks. We address these issues in this chapter.

Let x be an elementary event structure with persistence and let $f : x \to E$ be a morphism of any kind (partial, augmentation, demand or rigid). In this chapter we describe a method for factorising f into a rigid epimorphism $g : x \to x_0$ and a morphism $h : x_0 \to E$ of the same kind as f with the following property. If $h' \circ g' = f$ and $g' : x \to x'$ is a rigid epimorphism and $h' : x' \to E$ is of the same kind as f, the following diagram commutes for a unique rigid epimorphism $m : x' \to x_0$.

$$\begin{tikzcd}
x \arrow[rr, "f"] \arrow[dr, "g"] \arrow[ddr, bend right, "g'"'] & & E \\
& x_0 \arrow[ur, "h"] & \\
& x' \arrow[u, dotted, "m"'] \arrow[uur, bend right, "h'"'] &
\end{tikzcd}$$

In Section 6.3, we use these factorisations to define an adjunction between $\mathcal{R}_P$ and $\mathcal{D}_P$. We do the same for $\mathcal{R}_P$ and $\mathcal{A}_P$ in Section 6.4. Finally, in Section 6.5, we show how these factorisations can be used to construct limits in $\mathcal{R}_P$.

6.1 Pre-rigid and Rigid Pairs

We introduce *pre-rigid pairs* and *rigid pairs.* These consist of a relation and a property on the events of an elementary event structure x. We describe the properties these pairs must have. We define an ordering on pairs based on the subset relation and show that for any pre-rigid pair $(R,\ P)$ there is a maximal rigid pair $(X,\ \Phi)$ below it in this ordering.

We begin by defining pre-rigid pairs.

Definition 6.1.1. *Let x be an elementary event structure. A* pre-rigid pair *for x is a pair $(R,\ P)$ where R is an equivalence relation on the events of x and P is a property of events of x that holds of all persistent events.*

In order to define rigid pairs we require the following.

Definition 6.1.2. *A relation R on an elementary event structure x is a* $\leq$-bisimulation *if*

$$\begin{aligned} e_1 R e_2 \ \Rightarrow\ & i)\ \forall e_1' \leq e_1 . \exists e_2' \leq e_2 .\ e_1' R e_2' \text{ and} \\ & ii)\ \forall e_2' \leq e_2 . \exists e_1' \leq e_1 .\ e_1' R e_2'. \end{aligned} \tag{6.1}$$

A property P on x respects a relation *R on x if*

$$e_1 R e_2 \text{ and } e_1 \neq e_2 \ \Rightarrow\ P(e_1) \text{ and } P(e_2). \tag{6.2}$$

Definition 6.1.3. *A* rigid pair *for an elementary event structure x is a pre-rigid pair $(R,\ P)$ for x such that R is a $\leq$-bisimulation and P respects R.*

We define an ordering on pairs of relations and properties for an elementary event structure in terms of the subset relation.

Definition 6.1.4. *Let R' be a relation on the events of an elementary event structure x and let P' be a property on these events. We say a pair $(R',\ P')$ is a* sub-pair *of $(R,\ P)$ if R' is a subset of R and $P'(e) \Rightarrow P(e)$ holds for all $e \in x$. We write $(R',\ P') \subseteq (R,\ P)$.*

Proposition 6.1.5. *Let $(R,\ P)$ be a pre-rigid pair for an elementary event structure x.*

Let $X = \bigcup\{R' \subseteq R \mid R' \text{ is a } \leq\!-\text{bisimulation and } P \text{ respects } R'\}$. We have that $(X,\ P)$ is a rigid pair and indeed is the greatest rigid sub-pair of $(R,\ P)$.

Proof. We first show that X is a $\leq$-bisimulation and that P respects X. We will use Tarski's Fixed Point Theorem to do this. Define the function $\phi : Pow(R) \rightarrow Pow(R)$ for $R' \subseteq R$ by

$$\begin{aligned} e_1\ \phi(R')\ e_2 \quad \text{iff} \quad & i)\ \forall e_1' \leq e_1 . \exists e_2' \leq e_2 .\ e_1' R' e_2', \\ & ii)\ \forall e_2' \leq e_2 . \exists e_1' \leq e_1 .\ e_1' R' e_2' \text{ and} \\ & iii)\ e_1 \neq e_2 \ \Rightarrow\ P(e_1) \text{ and } P(e_2). \end{aligned}$$

Clearly ϕ is monotonic. Also R' is a $\leq$-bisimulation and P respects R' iff R' is a post-fixed point of ϕ, i.e., $R' \subseteq \phi(R')$. By Tarski's fixed point theorem, X is the greatest post-fixed point of ϕ. It is therefore a $\leq$-bisimulation and P respects X. It is also the greatest such relation.

It remains to show that X is an equivalence relation.

Reflexivity: Follows immediately from R being reflexive and the identity relation being a $\leq$-bisimulation that P respects.

Symmetry: If R' is a $\leq$-bisimulation that P respects then R'^{op} is clearly also a $\leq$-bisimulation that P respects. So, the symmetry of X follows from the symmetry of R.
Transitivity: We show that, if $R',\ R'' \subseteq R$ are both $\leq$-bisimulations respected by P then so is $R' \circ R''$. It is clear that P respects $R' \circ R''$ and that $R' \circ R'' \subseteq R$ because R is transitive. It remains to check that $R' \circ R''$ is a $\leq$-bisimulation. Assume $e_1\ (R' \circ\ R'')\ e_2$, i.e., there exists e_3 such that $e_1\ R'\ e_3$ and $e_3\ R''\ e_2$. Assume $e_1' \leq e_1$ in x. We show

$$\exists e_2' \leq e_2.\ e_1'\ (R' \circ R'')\ e_2'.$$

Given that R' and R'' are $\leq$-bisimulations, we have the following. If $e_1' \leq e_1$ then there exists $e_3' \leq e_3$ such that $e_1'\ R'\ e_3'$. Since $e_3' \leq e_3$ then there exists $e_2' \leq e_2$ such that $e_3'\ R''\ e_2'$. So there exists $e_2' \leq e_2$ such that $e_1'\ (R' \circ R'')\ e_2'$. Because $R' \circ R'' \subseteq R$ and therefore $R' \circ R'' \subseteq X$, it follows that X is transitive.

So $(X,\ P)$ is indeed the greatest rigid sub-pair of $(R,\ P)$. □

Definition 6.1.6. *Let $(R,\ P)$ be a pre-rigid pair. Its maximal rigid sub-pair is the* extremal *rigid sub-pair of $(R,\ P)$.*

6.2 Rigid Morphisms as Rigid Pairs

We show a bijective correspondence between rigid epimorphisms $f : x \to x'$, where x is an elementary event structure, and the rigid pairs for x. We then show that the morphism corresponding to the extremal rigid sub-pair of a pre-rigid pair $(R,\ P)$ factors uniquely through the morphisms corresponding to rigid sub-pairs of $(R,\ P)$.

We first define the pair to which a rigid epimorphism corresponds.

Definition 6.2.1. *Let $f : x \to x'$ be a rigid morphism. Define $(R_f,\ P_f)$ by*

$$\begin{array}{rll} e_1\ R_f\ e_2 & \text{if} & f(e_1) = f(e_2) \text{ and} \\ P_f(e) & \text{if} & f(e) \text{ is a persistent event} \end{array}$$

for all $e,\ e_1,\ e_2 \in x$.

Proposition 6.2.2. *If f is a rigid epimorphism then $(R_f,\ P_f)$ is a rigid pair.*

Proof. It is clear from the definition that R_f is an equivalence relation. We know that P_f holds for all persistent events because f preserves persistence. So, $(R_f,\ P_f)$ is a pre-rigid pair. We must next show that R_f is a $\leq$-bisimulation. Assume $e_1 R_f e_2$ so that $f(e_1) = f(e_2)$ and $e_1' \leq e_1$ for some $e_1,\ e_2 \in x$. If $f(e_1) = f(e_2)$ then $[f(e_1)] = [f(e_2)]$ and, as f is rigid, $f[e_1] = f[e_2]$. It must therefore be the case that there exists an $e_2' \in [e_2]$ such that we have $f(e_1') = f(e_2')$. A symmetric argument suffices for the second part of the property.

Finally we show that P_f respects R_f. If $f(e_1) = f(e_2)$ and $e_1 \neq e_2$ then, as f is a rigid morphism, it must be the case that $f(e_1)$ is persistent. □

In fact, every rigid pair of an elementary event structure x corresponds to a rigid epimorphism with domain x.

Definition 6.2.3. *Let $(R,\ P)$ be a rigid pair for an elementary event structure x. Define $(x',\ p',\ \leq)$ by*

- $x' \overset{def}{=} \{[e]_R \mid e \in x\}$ *where $[e]_R$ is the equivalence class of e according to R,*
- $[e]_R \in p'$ *iff* $P(e)$,
- $[e]_R \leq [e']_R$ *iff* $\exists e_1 \in [e]_R,\ e_2 \in [e']_R.\ e_1 \leq e_2$.

Define $\langle R,\ P\rangle : x \to x'$ by

$$\langle R,\ P\rangle(e) = [e]_R.$$

The following property is used in the remainder of the chapter.

Lemma 6.2.4. *Let $g : x \to y$ be a rigid morphism. The following are equivalent.*

- *The morphism g is a surjective function.*
- *The morphism g is an epimorphism.* □

It is trivial to see that, if g is surjective, it is an epimorphism. To see the reverse, recall that $h \circ g = h' \circ g$ implies $h = h'$ for all rigid morphisms h and h' with domain y. If g was not also a surjective function, we could easily construct morphisms h and h' for which $h \circ g = h' \circ g$ and $h \neq h'$.

Theorem 6.2.5. *Let x be an elementary event structure. The constructions in Definitions 6.2.1 and 6.2.3 define a bijective correspondence between rigid pairs for x and rigid epimorphisms with domain x up to isomorphism of the codomain.*

Proof. In order to prove the theorem we must show that, if $(R,\ P)$ is a rigid pair for x, then

i) $\langle R,\ P\rangle : x \to x'$ is a rigid epimorphism and

ii) it is the unique epimorphism corresponding to $(R,\ P)$ up to isomorphism of the codomain.

i) We must show that x' is an elementary event structure and that $\langle R, P\rangle$ has the correct properties to be a rigid epimorphism. We begin by proving that the causality relation of x' is a partial order.
Reflexivity: Obvious.
Transitivity: Suppose $[e_1]_R \leq [e_2]_R$ and $[e_2]_R \leq [e_3]_R$. Then there exist $e_1' \in [e_1]_R$, $e_2',\ e_2'' \in [e_2]_R$ and $e_3' \in [e_3]_R$ for which we have $e_1' \leq e_2'$ and $e_2'' \leq e_3'$. We have $e_2' R e_2''$ and $e_1' \leq e_2'$ so, as R is a $\leq$-bisimulation, there must exist $e_1'' \in x$ such that $e_1'' \leq e_2''$ and $e_1 R e_1''$. By transitivity of $\leq$ on x, we have $e_1'' \leq e_3'$ and therefore $[e_1]_R \leq [e_3]_R$.

Anti-symmetry: Suppose $[e_1]_R \leq [e_2]_R$ and $[e_2]_R \leq [e_1]_R$. Then there exist $e_1',\ e_1'' \in [e]_R$ and $e_2',\ e_2'' \in [e']_R$ for which we have $e_1' \leq e_2'$ and $e_2'' \leq e_1''$. It follows from R being a $\leq$-bisimulation that there exists $e_2''' \in x$, related by R to e_2'' such that $e_2''' \leq e_1'$. As e_2''' is related to e_2'' by R, we have, from the transitivity of R that e_2''' is related to e_2'. As we have $e_1' \leq e_2'$, there exists an e_1''' for which we have $e_1''' \leq e_2'''$ and $e_1''' R e_1'$. The argument can be continued similarly to produce a descending chain of events that alternate between being related to e_1 and being related to e_2. As $[e]$ is finite for all events e in x, the chain must eventually become constant and so there is an e_3 related by R to both e_1 and e_2, implying $[e_1]_R = [e_2]_R$.

Let $f = \langle R,\ P \rangle$. We next prove that $f : x \to x'$ is a rigid morphism.

We first show $[f(e)] \subseteq f[e]$ for all $e \in x$. Apologising for the notation we let $[e_1]_R \in [f(e)]$. This implies that there exist $e_1' \in [e_1]_R$ and $e' \in f(e)$ for which we have $e_1' \leq e'$. So, as $e' R e$ holds, there must exist $e'' \in x$, for which $e'' \leq e$ and $e_1' R e''$. Therefore we know $[e_1]_R \in f[e]$.

We next show $f[e] \subseteq [f(e)]$. Suppose $[e_1]_R \in f[e]$. There must exist $e' \leq e$ such that $f(e') = [e_1]_R$, i.e., such that $e' R e_1$. As $[e_1]_R$ contains an event upon which e depends, it follows from the definition that $[e_1]_R \in [f(e)]$.

That f behaves correctly concerning persistence follows directly from P respecting R.

ii) We show that f is the unique epimorphism for which $(R_f,\ P_f) = (R,\ P)$ up to isomorphism of the codomain. As the underlying function of f is surjective, it is clearly an epimorphism. Suppose there exists another epimorphism $g : x \to y$ for which $(R_g,\ P_g) = (R,\ P)$. Define $\alpha : x' \to y$ and $\alpha^{-1} : y \to x'$ by

$$\alpha([e]_R) \overset{def}{=} g(e) \text{ and } \alpha^{-1}(e') \overset{def}{=} [e'']_R$$

for all $e \in x$ and $e' \in y$ where $e'' \in x$ is such that $g(e'') = e'$ (making use of Lemma 6.2.4). We have that α and α^{-1} are well-defined as we have $e_1 R e_2$ iff $g(e_1) = g(e_2)$. To see that α is an isomorphism on the set of events consider that $(\alpha^{-1} \circ \alpha)([e]_R) = [e'']_R$ where $g(e'') = g(e)$ and therefore we have $e'' R e$ and so $[e'']_R = [e]_R$ and $(\alpha \circ \alpha^{-1})(e) = g(e')$ where $e' \in x$ with $g(e') = e$. Having shown that α is a bijection on events, we now show that α is a rigid morphism that reflects persistence. The result then follows directly from Lemma 5.4.5. As g corresponds to the same property of events as f, we have that α behaves as required with persistence. We show that $[\alpha([e]_R)] = \alpha[[e]_R]$ for $[e]_R \in x'$. Suppose $e' \in [\alpha([e]_R)]$. Then $e' \in [g(e)]$ and therefore e' is in $g[e]$ as g is rigid. So, there exists $e'' \in [e]$ such that $g(e'') = e'$. As we have $e'' \leq e$ we also have $[e'']_R \leq [e]_R$ and so e' is in $\alpha[[e]_R]$. So, $[\alpha([e]_R)] \subseteq \alpha[[e]_R]$. Assume $e' \in \alpha[[e]_R]$. Then there exists $e'' \in x$ such that $[e'']_R$ is in $[[e]_R]$ and $e'' \leq e$ holds and g maps e'' to e'. This implies $e' \in g[e]$ and therefore that e' is in $[\alpha[e]_R]$ as g is rigid. So, α is a rigid morphism that is a bijection on events, reflects persistence and (trivially) preserves conflict. From Lemma 5.4.5, it is therefore an isomorphism between x' and y. □

As all rigid pairs correspond to rigid epimorphisms, the extremal rigid sub-pair of a pre-rigid pair for x corresponds to a rigid epimorphism.

Definition 6.2.6. *Let $(R,\ P)$ be a pre-rigid pair and $(X,\ P)$ its extremal rigid sub-pair. If $\langle X,\ P\rangle : x \to x_0$ is the morphism corresponding to $(X,\ P)$ (see Definition 6.2.3) then we refer to x_0 (and all things isomorphic to it) as* extremal elementary event structures *for $(R,\ P)$.*

Throughout the rest of the chapter, we treat extremals as having events of the form $[e]_X$.

Proposition 6.2.7. *Let x be an elementary event structure and let $(R,\ P)$ be a pre-rigid pair on x. Let $f_0 : x \to x_0$ be a rigid epimorphism between elementary event structures.*

The following are equivalent.

i) $(R_{f_0},\ P_{f_0})$ is extremal for $(R,\ P)$.

ii) If $g : x \to x'$ is a rigid epimorphism and $(R_g,\ P_g) \subseteq (R,\ P)$ then there exists a unique rigid epimorphism $m : x' \to x_0$ such that the following diagram commutes.

$$\begin{tikzcd}
x \arrow[r, "f_0"] \arrow[dr, "g"'] & x_0 \\
& x' \arrow[u, dotted, "m"']
\end{tikzcd}$$

Proof. We first show that (i) implies (ii). Assume (i). Let $(X,\ P) = (R_{f_0},\ P_{f_0})$. We begin by defining m and demonstrating that it is a rigid morphism. Define $m(e)$ to be $f_0(e')$ where $e' \in x$ is such that $g(e') = e$ for all $e \in x'$ (making use of Lemma 6.2.4). This is well-defined as, because $(X,\ P)$ is extremal, if $g(e_1) = g(e_2)$ for $e_1,\ e_2 \in x$ then $e_1 X e_2$. *Rigidity of m*: We first show that $[m(e)] = m[e]$. Assume $e' \in m[e]$, i.e., that there exists $e_1 \in [e]$ such that $m(e_1) = e'$. As g is rigid and therefore reflects dependency and we have $e_1 \leq e$, there exist $e'_1,\ e'' \in x$ such that $g(e'_1) = e_1$ and $g(e'') = e$ for which $e'_1 \leq e''$ holds. As f_0 is rigid, it preserves dependencies and so we have $f_0(e'_1) \leq f_0(e'')$. This implies $m(e_1) \leq m(e)$ as $f_0(e'_1) = m(e_1)$ and $f_0(e'') = m(e)$ so $m[e] \subseteq [m(e)]$ holds for all $e \in E$.

As x_0 is an extremal elementary event structure for $(R,\ P)$, we can assume it has events of the form $[e'']_X$ (see Definition 6.2.6). Assume $e' \in [m(e)]$, i.e., assume $e' \leq m(e)$. We must show $e' \in m[e]$, i.e., that there exists $e''' \leq e$ such that $m(e''') = e'$. From the definition of x_0, if $e' \leq m(e)$ then there exist $e_1 \in e'$ and $e_2 \in m(e)$ such that $e_1 \leq e_2$ holds in x. As g is rigid, we have $g(e_1) \leq g(e_2)$. It follows from the definition of m that $(m \circ g)(e_2) = m(e)$ and that $(m \circ g)(e_1) = e'$. As $e_2 \in m(e)$ it must be that X relates e_2 to all the events $e_3 \in x$ for which $g(e_3) = e$. From the properties of $\leq$-bisimulations we can deduce that, as we have $e_1 \leq e_2$, there must exist e'_3 such that $e'_3 \leq e_3$ and $e_1 X e'_3$ hold. So, we have $g(e'_3) \leq e$ and therefore $g(e'_3) \in [e]$. Recall that $(m \circ g)(e'_3) = (m \circ g)(e_1) = e'$ and so $e' \in m[e]$ implying $[m(e)] \subseteq m[e]$ for all $e \in E$.

We next show that $m(e_1) = m(e_2)$ implies $e_1 = e_2$ or $m(e_1)$ is persistent. If $m(e_1) = m(e_2)$ for two distinct events e_1 and $e_2 \in x'$ then for all $e'_1 \in x$ such that $g(e'_1) = e_1$ and $e'_2 \in x$ such that $g(e'_2) = e_2$ we have $f_0(e'_1) = f_0(e'_2)$. As f_0 is a rigid morphism, this implies that $f_0(e'_1)$ is persistent and therefore that $m(e_1)$ is persistent.

Finally, we show that m preserves persistence. If $e_1 \in x'$ is persistent then, because $(R_g,\ P_g) \subseteq (R,\ P)$, it follows that P holds for all events e_1' in x that are mapped to e_1 by g. So $f_0(e_1')$ is persistent for all such events and therefore m preserves persistence.

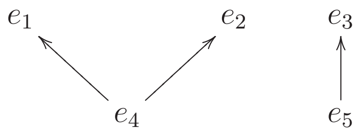
Uniqueness of m: Suppose there exists $m' : x' \to x_0$ such that $m' \circ g = m \circ g = f_0$. As g is an epimorphism, this implies that $m = m'$.

Surjectivity of m: The morphism $f_0 = m \circ g$ is surjective from its definition. Therefore m must be surjective and is therefore an epimorphism.

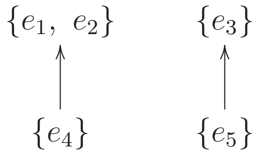
We now show that (ii) implies (i). Observe that $f_0 = m \circ g$ implies $(R_g,\ P_g)$ is a subset of $(R_{f_0},\ P_{f_0})$. Also, from Theorem 6.2.5, all rigid sub-pairs of $(R,\ P)$ correspond to rigid epimorphisms. So, from this and the assumption of (ii), all rigid sub-pairs of $(R,\ P)$ are subpairs of (R_{f_0}, P_{f_0}). It is therefore the largest and so, from Definition 6.1.6, the extremal rigid sub-pair of $(R,\ P)$. □

The following examples show the construction being applied to two different elementary event structures and pre-rigid pairs.

Example 1: Consider the case where x is given by



where e_1 is a persistent event. Let R be the equivalence relation with classes $\{e_1,\ e_2,\ e_3\}$ and $\{e_4,\ e_5\}$ and let P be true for $e_1,\ e_2,\ e_3$, and e_5. Although $(R,\ P)$ is a pre-rigid pair for x, it is not rigid because P does not respect R, i.e., $e_4 R e_5$ but $P(e_4)$ does not hold. The extremal elementary event structure for $(R,\ P)$ is shown below.



where $\{e_1,\ e_2\}$, $\{e_3\}$ and $\{e_5\}$ are persistent events.

Example 2: For any elementary event structure x, if R is the equivalence relation that relates all the events in x to each other and if P is true for all events, then the extremal elementary event structure for $(R,\ P)$ is an event structure consisting of one persistent event.

An intuition for the relationship between an elementary event structure x and the extremal elementary event structure x_0 for a particular pre-rigid pair $(R,\ P)$ for x is that x_0 merges the events of x in such a way that there is a rigid morphism respecting R into x_0 from x and that x_0 is the minimum such elementary event structure, i.e., the event structure with the fewest events but the most persistent events.

6.3 The Category of Demand Morphisms as a Kleisli Category

In this section we show how to construct an adjunction between the categories $\mathcal{R}_P$ and $\mathcal{D}_P$ via rigid pairs.

We first describe a functor from $\mathcal{R}_P$ to $\mathcal{D}_P$.

Definition 6.3.1. *Define F to map objects in $\mathcal{R}_P$ to themselves in $\mathcal{D}_P$ and define its action on arrows $f : E_1 \to E_2$ in $\mathcal{R}_P$ by*

$$F(f)(e) \overset{def}{=} f[e]$$

for all $e \in E_1$.

Proposition 6.3.2. *The operation $F : \mathcal{R}_P \to \mathcal{D}_P$ is a functor.*

Proof. We first show that, if $f : E_1 \to E_2$ is a rigid morphism, then $F(f)$ is a demand morphism between E_1 and E_2. As f is a rigid morphism, it preserves configurations and therefore $F(f)(e) \in \mathcal{C}^0(E_2)$ for all $e \in E_1$. We next show that $e_1 \leq e_2$ implies $F(f)(e_1) \subseteq F(f)(e_2)$. If we have $e_1 \leq e_2$ then we also have $[e_1] \subseteq [e_2]$ and therefore $f[e_1] \subseteq f[e_2]$. Finally, we show that $F(f)(e_1) \not\uparrow F(f)(e_2)$ implies $e_1 \# e_2$. If $f[e_1] \not\uparrow f[e_2]$ then there is no configuration of E_2 with both $f[e_1]$ and $f[e_2]$ as subsets. This implies that $[e_1] \cup [e_2]$ cannot be a configuration of E_1 as f preserves configurations and therefore $e_1 \# e_2$.

The definition of F ensures that identities are preserved. We show it also distributes over composition. Let $f : E_1 \to E_2$ and $g : E_2 \to E_3$ be morphisms in $\mathcal{R}_P$. Then $F(g \circ f)(e) = (g \circ f)[e]$ and

$$\begin{aligned}
(F(g) \circ F(f))(e) &= (F(g)^{\dagger} \circ f)[e] \\
&= \bigcup_{e' \in f[e]} g[e'] \\
&= \bigcup_{e' \in [f(e)]} g[e'] \qquad \text{(rigidity of } f\text{)} \\
&= g[f(e)] \qquad \text{(because } e' \in [f(e)] \text{ implies } [e'] \subseteq [f(e)]\text{)} \\
&= (g \circ f)[e].
\end{aligned}$$

This confirms that F is indeed a functor between $\mathcal{R}_P$ and $\mathcal{D}_P$. □

We will define a right adjoint $Dem : \mathcal{D}_P \to \mathcal{R}_P$ by giving an object $Dem(E) \in \mathcal{R}_P$ and an arrow $\epsilon_E : F(Dem(E)) \to E$ universal from F to E.

In the remainder of this section, we describe an event structure $Dem(E)$ and a morphism $\epsilon_E : F(Dem(E)) \to E$ for each $E \in \mathcal{D}_P$ with the aid of rigid pairs. In order to do this we must first derive some properties of factorisations of demand morphisms.

Definition 6.3.3. *Let $d : x \to E$ be a demand morphism. Define the pre-rigid pair $(R,\ P)$ associated to d as follows.*

$$e_1 R e_2 \overset{def}{\Leftrightarrow} d(e_1) = d(e_2).$$

Define P to be true for all events in x.

For convenience, we occasionally use demand morphisms to denote the pre-rigid pairs to which they correspond.

Proposition 6.3.4. *i) The rigid sub-pairs $(R',\ P')$ of the pre-rigid pair $(R,\ P)$ of a demand morphism $d : x \to E$ induce factorisations of d into the rigid morphism $\langle R',\ P'\rangle : x \to x'$ (see Definition 6.2.3) followed by a demand morphism $d' : x' \to E$ where we define d' by*

$$d'[e]_{R'} \overset{def}{=} d(e)$$

for all $[e]_{R'}$ in the image of $\langle R',\ P'\rangle$.
ii) Any factorisation of d into a rigid epimorphism f followed by a demand morphism induces a rigid sub-pair of $(R,\ P)$.

Proof. (i) That d' is well-defined follows from the definition. It is clear that $d' \circ \langle R', P'\rangle$ equals d. We show that d' is a demand morphism. Suppose $[e_1]_{R'} \leq [e_2]_{R'}$ in x'. Then there exists $e_1' \in [e_1]_{R'}$ such that $e_1' \leq e_2$ holds and therefore, as $d(e_1')$ is a subset of $d(e_2)$ and $e_1'\ R'\ e_1$ implies $d(e_1') = d(e_1)$, we have $d'[e_1]_{R'} \subseteq d'[e_2]_{R'}$. Observe that, as $d^\dagger x \in \mathcal{C}(E)$, we have $d'[e_1]_{R'} \uparrow d'[e_2]_{R'}$ for all $[e_1]_{R'},\ [e_2]_{R'}$ in x'. Therefore d' is a demand morphism.
(ii) The rigid epimorphism f corresponds to a rigid pair $(R_f,\ P_f)$. As P always holds, P_f is a subset of P and, as f is part of a factorisation of d, if $f(e_1) = f(e_2)$ for any $e_1,\ e_2 \in x$ then $d(e_1) = d(e_2)$ and so we have $(R_f,\ P_f) \subseteq (R,\ P)$. □

In particular, the greatest rigid sub-pair $(X,\ P)$ for a demand morphism d induces a factorisation of d into $d' \circ [\_]_X$.

Lemma 6.3.5. *Let $d : x \to E$ be a demand morphism with pre-rigid pair $(R,\ P)$. Let $(X,\ P)$ be the greatest rigid sub-pair of $(R,\ P)$ with associated factorisation*

$$d = x \overset{[\_]_X}{\to} x_0 \overset{d_0}{\to} E.$$

For any rigid epimorphism $f : x \to x'$ and any demand morphism $d' : x' \to E$, if $d' \circ f = d$ then the following diagram commutes for a unique rigid morphism $m : x' \to x_0$.

$$\begin{tikzcd}
x \arrow[rr, "d"] \arrow[dr, "{[\_]_X}"'] \arrow[ddr, bend right, "f"'] & & E \\
& x_0 \arrow[ur, "d_0"'] & \\
& x' \arrow[u, dotted, "m"'] \arrow[uur, bend right, "d'"'] &
\end{tikzcd}$$

Proof. It follows from Proposition 6.2.7 that there exists a unique rigid morphism m such that $m \circ f = [\_]_X$. It remains to check that $d_0 \circ m = d'$. We have $d_0 \circ m \circ f = d' \circ f$. As f is an epimorphism, this implies $d_0 \circ m = d'$. □

Let E be an event structure with persistence. We now describe the construction of an event structure $Dem(E)$ and a demand morphism ϵ_E such that for all demand morphisms $d : E' \rightarrow E$ in $\mathcal{D}_P$ the diagram below commutes for a unique rigid morphism $m : E' \rightarrow Dem(E)$.

$$\begin{tikzcd}
E' \arrow[rr, "d"] \arrow[dr, dotted, "m"'] & & E \\
& Dem(E) \arrow[ur, "\epsilon_E"'] &
\end{tikzcd}$$

Definition 6.3.6. *Let $\mathcal{D}_E$ be the* category of demands *for the event structure E with objects $(x,\ d)$ where x is an elementary event structure with persistence and d is a demand morphism from x into E. A morphism f between $(x_1,\ d_1)$ and $(x_2,\ d_2)$ in $\mathcal{D}_E$ is a rigid morphism $f : x_1 \rightarrow x_2$ such that $d_1 = d_2 \circ f$.*

Definition 6.3.7. *The pair $(x_0,\ d_{x_0})$ in $\mathcal{D}_E$ is extremal for $d_x : x \rightarrow E$ if there exists a rigid epimorphism $f : (x,\ d_x) \rightarrow (x_0,\ d_{x_0})$ in $\mathcal{D}_E$ such that $(R_f,\ P_f)$ is the extremal of d_x. The pair $(x_0,\ d_{x_0})$ is* extremal *in $\mathcal{D}_E$ if it is extremal for some $d_x : x \rightarrow E$. It is prime if x_0 is prime, i.e., iff there exists $e \in x_0$ such that $e' \leq e$ for all $e' \in x_0$.*

Note that, if $(x_0,\ d_{x_0})$ is extremal in $\mathcal{D}_E$, then it is extremal for d_{x_0} (follows immediately from Proposition 6.2.7).

Let $Skl(\mathcal{D}_E)$ be the skeleton of $\mathcal{D}_E$. Making use of $Skl(\mathcal{D}_E)$ removes multiple isomorphic event structures when making definitions. The properties of extremals and primality are stable under isomorphisms and so we consider them as properties of objects in $Skl(\mathcal{D}_E)$.

Definition 6.3.8. *If $(E,\ P,\ \leq,\ \#)$ is an event structure, define $Dem(E)$ to be the event structure $(E',\ P',\ \leq',\ \#')$ where*

- *E' is the set of prime extremals in $Skl(\mathcal{D}_E)$,*
- *$P' = E'$,*
- *$e \leq' e'$ iff there exists a morphism $f : e \rightarrow e'$ in $\mathcal{D}_E$,*
- *$e \#' e'$ iff there is no object in $\mathcal{D}_E$ with morphisms from e and e' into it.*

In order to prove that $Dem(E)$ is an event structure, we make use of the following properties of extremals in $\mathcal{D}_E$.

Proposition 6.3.9. *If $(x_0,\ d_{x_0}) \in \mathcal{D}_E$ is extremal, all the morphisms in the category $\mathcal{D}_E$ with domain $(x_0,\ d_{x_0})$ are injective functions on x_0 that reflect persistence. Surjections out of $(x_0,\ d_0)$ are isomorphisms.*

Proof. Let $f : (x_0,\ d_0) \rightarrow (y,\ d_y)$ be a morphism in $\mathcal{D}_E$. Let y' be the restriction of y to the image of f.

By Proposition 6.2.7, we have a unique epimorphism $m : y' \rightarrow x_0$ such that the following diagram commutes.

$$\begin{tikzcd}
x_0 \arrow[r, "d_{x_0}"] & E \\
y' \arrow[u, dotted, "m"] \arrow[ur, "d_y"'] &
\end{tikzcd}$$

Now, we also have $d_{x_0} \circ m \circ f \circ m = d_y$ since

$$\begin{aligned}
d_{x_0} \circ m \circ f \circ m &= d_y \circ f \circ m \\
&= d_{x_0} \circ m \quad \text{from the definition of } f \\
&= d_y.
\end{aligned}$$

Since m is the unique such morphism, we have $m \circ f \circ m = m$ and since m is an epimorphism, we have $m \circ f = id_{x_0}$. So, f is a split monic, hence injective.

If f is a surjection then it is an epimorphism. So, as split monomorphisms that are epimorphisms are isomorphisms, f is an isomorphism. □

Proposition 6.3.10. *If $(x_0,\ d_{x_0})$ is extremal for $d_x : x \rightarrow E$ and there is a rigid surjection $f : (x',\ d_{x'}) \rightarrow (x_0,\ d_{x_0})$ then $(x_0,\ d_{x_0})$ is extremal for $d_{x'}$.*

Proof. Follows immediately from Proposition 6.2.7. □

Definition 6.3.11. *Let x be an elementary event structure. We say x' is a restriction of x if $x' = x \upharpoonright S$ for some $S \subseteq x$ that is downwards closed with respect to $\leq$. An object $(x',\ d_{x'})$ of $\mathcal{D}_E$ is a restriction of $(x,\ d_x) \in \mathcal{D}_E$ iff x' is a restriction of x and $d_{x'}$ is d_x with domain restricted to x'.*

In the following, if x' is a restriction of the elementary event structure x and $(x,\ d_x)$ is a member of $\mathcal{D}_E$, we sometimes write $(x',\ d_x)$ for the object in $\mathcal{D}_E$ consisting of x' and the restriction of d_x to a morphism from x' to E.

Proposition 6.3.12. *The restriction of an extremal is also extremal.*

Proof. In order to show the above we first prove the following more general property.

Let $(y,\ d_y)$ be a member of $\mathcal{D}_E$ and let $(x,\ d_x)$ be a restriction of $(y,\ d_y)$. If $(R,\ P)$ is a rigid sub-pair of d_x then $(R,\ P)$ is the restriction of a rigid sub-pair of d_y, i.e., $(R,\ P) = (R' \cap (x \times x),\ P' \cap x)$ for some $(R',\ P')$ that is a rigid sub-pair of d_y.

Define $(R',\ P')$ to be $(R \cup Id_y,\ P \cup S)$ where Id_y is the identity relation on events in y and $S \overset{def}{=} \{e \in y \mid e \text{ persistent}\}$. It is clear from the definition that $(R',\ P')$ is pre-rigid and a sub-pair of d_y. We now show that it is a rigid pair.

R' is a $\leq$-bisimulation: Assume $e_1 R' e_2$ so $e_1\ (R \cup Id_y)\ e_2$ and assume $e_1' \leq e_1$. Then either e_1 and e_2 are in x and therefore part (i) of property 6.1 holds because it holds for

R or e_1 and e_2 are in y and therefore $e_1 = e_2$. A symmetric argument demonstrates that the second part of property 6.1 also holds.

P' respects R': Suppose $e_1 R' e_2$ so $e_1 \; R \cup Id_y \; e_2$ and $e_1 \neq e_2$. Then e_1 must be related to e_2 by R and therefore $P(e_1)$ and $P(e_2)$ hold and so $P'(e_1)$ and $P'(e_2)$ hold.

We can now prove that the restriction of an extremal is also extremal.

Let $(x,\ d_x) \in \mathcal{D}_E$ and let $(x_0,\ d_{x_0})$ be the extremal of d_x. As explained earlier, we assume the events of x_0 are equivalence classes of the maximal rigid pair $(X,\ P)$ for d_x. Let S be a downwards closed subset of x_0. We show that a restriction $(x_0 \restriction S,\ d_{x_0})$ is the extremal for a restriction $d_{x'} : x' \to E$ of d_x. Define x' to be the elementary event structure $x \restriction \bigcup S$. We know that x' is indeed an elementary event structure because, from X being a $\leq$-bisimulation and S being downwards closed, we know that $\bigcup S$ is a downwards closed subset of x. Define $d_{x'}$ to be d_x with domain restricted to x'. There is a rigid morphism $f : (x',\ d_{x'}) \to (x_0 \restriction S,\ d_{x_0})$ given by the restriction of $\langle X,\ P\rangle$ to domain x'. We show that $(R_f,\ P_f)$ is the maximal rigid sub-pair of $d_{x'}$ by showing that all the rigid sub-pairs of $d_{x'}$ are sub-pairs of $(R_f,\ P_f)$. Suppose $(R,\ P)$ is a rigid sub-pair of $d_{x'}$. Then, from the above, it is the restriction of a rigid sub-pair of d_x. As $(R_f,\ P_f)$ is the restriction of $(X,\ P)$ to the events x', we have $(R,\ P) \subseteq (R_f,\ P_f)$. It is therefore the maximum rigid sub-pair of $d_{x'}$, making $(x_0 \restriction S,\ d_{x_0})$ extremal. □

Theorem 6.3.13. *$Dem(E)$ is an event structure for all event structures E.*

Proof. Let $\leq$ be the causality relation of $Dem(E)$.

Properties of $\leq$: We first show that $\leq$ is a partial order. Reflexivity and transitivity follow immediately from the properties of morphisms. Let $(x_0,\ d_{x_0})$ and $(y_0,\ d_{y_0})$ be events in $Dem(E)$. In order to show anti-symmetry, we note that, if there is a morphism $f : (x_0,\ d_{x_0}) \to (y_0,\ d_{y_0})$ and a morphism $g : (y_0, d_{y_0}) \to (x_0,\ d_{x_0})$ between extremals in $\mathcal{D}_E$, then they must be isomorphic (Proposition 6.3.9 and the Schroeder-Bernstein Theorem). As $(x_0,\ d_{x_0})$ and $(y_0,\ d_{y_0})$ are in $Skl(\mathcal{D}_E)$, this implies $(x_0,\ d_{x_0}) = (y_0,\ d_{y_0})$. Therefore $\leq$ is a partial order. We now show that for all events $(x_0,\ d_{x_0}) \in Dem(E)$, we have that $[(x_0,\ d_{x_0})]$ is finite. As $(x_0,\ d_{x_0})$ is a prime elementary event structure, it has only a finite number of sub-primes, i.e., restrictions of x_0 that are prime. For all prime extremals $(y_0,\ d_{y_0})$ with morphisms f into $(x_0,\ d_{x_0})$ we have $f(y_0) = [f(max(y_0))]$, i.e., they will be mapped to one of the sub-primes within x_0 due to the rigidity of f. So, if there are an infinite number of prime extremals mapping into $(x_0,\ d_{x_0})$, two distinct prime extremals $(y_1,\ d_{y_1})$ and $(y_2,\ d_{y_2})$ with morphisms f_1 and f_2 into $(x_0,\ d_{x_0})$ must map to the same sub-prime $(x_0 \restriction [e],\ d_{x_0})$ where e is some event in x_0. We show that this implies that they are isomorphic. The restrictions of f_1 and f_2 to morphisms into $(x_0 \restriction [e],\ d_{x_0})$ are surjective. It follows from Propositions 6.3.12 and 6.3.9 that $(x_0 \restriction [e],\ d_{x_0})$ is extremal and therefore isomorphic to both $(y_1,\ d_{y_1})$ and $(y_2,\ d_{y_2})$. So $(y_1,\ d_{y_1}) = (y_2,\ d_{y_2})$ as they are objects in $Skl(\mathcal{D}_E)$ and therefore $[(x_0,\ d_{x_0})]$ is finite.

Let # be the conflict relation of $Dem(E)$.

Properties of #: It follows immediately from the definition that # is irreflexive and symmetric. Let e_1, e_2 and e_3 be events in $Dem(E)$ such that $e_1 \# e_2$ and $e_2 \leq e_3$. We now show $e_1 \# e_3$ by contradiction. If e_1 and e_3 are not in conflict then there exists $x \in \mathcal{D}_E$

with morphisms $f : e_1 \to x$ and $g : e_3 \to x$ into it. However, $e_2 \leq e_3$ means there is a morphism from e_2 to e_3 and therefore, by composition of this with g, that e_1 and e_2 are not in conflict. This is a contradiction and therefore we have $e_1 \# e_3$. □

Define $\epsilon_E : F(Dem(E)) \to E$ by

$$\epsilon_E(x,\ d) \overset{def}{=} d^\dagger(x)$$

for all $(x,\ d)$ in $Dem(E)$. Note that, as $e_1 \leq e_2$ implies $d(e_1) \subseteq d(e_2)$ for all $e_1,\ e_2 \in x$, we could equivalently define ϵ_E by $\epsilon_E(x,\ d) = d(max(x))$.

Proposition 6.3.14. *The function ϵ_E is a demand morphism from $Dem(E)$ to E.*

Proof. Let $(x,\ d_x)$ and $(y,\ d_y)$ be events in $Dem(E)$. We first show that $(x,\ d_x) \leq (y,\ d_y)$ implies $\epsilon_E(x,\ d_x) \subseteq \epsilon_E(y,\ d_y)$. If $(x,\ d_x) \leq (y,\ d_y)$ then there exists a morphism $f : (x,\ d_x) \to (y,\ d_y)$ such that $d_y \circ f = d_x$ and therefore $(d_y^\dagger \circ f)(x) = d_x^\dagger(x)$. As $f(x) \subseteq y$, this implies $\epsilon_E(x,\ d_x) \subseteq \epsilon_E(y,\ d_y)$. Finally, we show that $\epsilon_E(x,\ d_x) \not\uparrow \epsilon_E(y,\ d_y)$ implies $(x,\ d_x) \# (y,\ d_y)$. Assume $\epsilon_E(x,\ d_x) \not\uparrow \epsilon_E(y,\ d_y)$. Then there can be no object in $\mathcal{D}_E$ that both $(x,\ d_x)$ and $(y,\ d_y)$ map into. This is because, if such an object $(z,\ d_z)$ existed, $d_z(z)$ would have to contain both $d_x(x)$ and $d_y(y)$, thus contradicting the initial assumption. We therefore have that ϵ_E is a demand morphism. □

In the proof of Proposition 6.3.16, we make use of the following.

Lemma 6.3.15. *Let $(z,\ d_z) \in \mathcal{D}_E$. Let $(y,\ d_y)$ be a restriction of $(z,\ d_z)$ with corresponding morphism $r : (y,\ d_y) \to (z,\ d_z)$. Let $(y_0,\ d_{y_0})$ and $(z_0,\ d_{z_0})$ be the extremals of d_y and d_z. Then there exists a morphism $f : (y_0,\ d_{y_0}) \to (z_0,\ d_{z_0})$ such that the following diagram commutes where $m_y : (y,\ d_y) \to (y_0,\ d_{y_0})$ and $m_z : (z,\ d_z) \to (z_0,\ d_{z_0})$ are the unique morphisms from $(y,\ d_y)$ and $(z,\ d_z)$ into their extremals.*

$$\begin{tikzcd}
(y,\ d_y) \arrow[r, "r"] \arrow[d, "m_y"'] & (z,\ d_z) \arrow[d, "m_z"] \\
(y_0,\ d_{y_0}) \arrow[r, "f"'] & (z_0,\ d_{z_0})
\end{tikzcd}$$

Proof. Restrict z_0 to $m_z \circ r(y)$. Observe that $m_z \circ r$ is surjective into the restriction. From Proposition 6.3.12 and Proposition 6.3.10, this restriction is extremal for $(y,\ d_y)$ and therefore isomorphic to $(y_0,\ d_{y_0})$. There must therefore be a morphism from $(y_0,\ d_{y_0})$ into $(z_0,\ d_{z_0})$ as required. □

Proposition 6.3.16. *For every event structure E' with a demand morphism $d : E' \to E$, there exists a unique rigid morphism $m : E' \to Dem(E)$ such that the following diagram commutes.*

$$\begin{tikzcd}
Dem(E) \arrow[r, "\epsilon_E"] & E \\
E' \arrow[u, dotted, "m"] \arrow[ur, "d"'] &
\end{tikzcd}$$

Proof. We first define $m : E' \to Dem(E)$. For all $e \in E'$, the restriction of E' to $[e]$ paired with the corresponding restriction of d is clearly a member of $\mathcal{D}_E$ as d is a demand morphism. Define m by

$$m(e) \overset{def}{=} (x,\ d_x)$$

where $(x,\ d_x)$ is the extremal of d restricted to domain $[e]$ in $Skl(\mathcal{D}_E)$.

Let $h : (E' \upharpoonright [e],\ d) \to (x,\ d_x)$ be the unique morphism in $\mathcal{D}_E$ from $(E' \upharpoonright [e],\ d)$ into $(x,\ d_x)$.

As h is rigid and $max(E \upharpoonright [e]) = e$ we have $(d_x \circ h)(e) = d_x(max(x)) = d(e)$ and therefore that $d = \epsilon_E \circ m$.

Rigidity of m: That m preserves persistence holds trivially because all events in $Dem(E)$ are persistent.

In the following let e_1 and e_2 be events in E' and let $(x_1,\ d_{x_1})$ equal $m(e_1)$ and $(x_2,\ d_{x_2})$ equal $m(e_2)$.

We show that m reflects conflict. If $m(e_1)\#m(e_2)$ then there is no $(y,\ d_y)$ in $\mathcal{D}_E$ that $m(e_1)$ and $m(e_2)$ both map to. We show by contradiction that this implies that there is no configuration z of E' that contains both e_1 and e_2. Assume such a z exists. Let $(z,\ d_z)$, $([e_1],\ d_{[e_1]})$ and $([e_2],\ d_{[e_2]})$ be the objects in $\mathcal{D}_E$ given by the restrictions of E' and d to z, $[e_1]$ and $[e_2]$ respectively. Observe that $([e_1],\ d_{[e_1]})$ and $([e_2],\ d_{[e_2]})$ are restrictions of $(z,\ d_z)$. This implies from Lemma 6.3.15 that the extremals of $d_{[e_1]}$ and $d_{[e_2]}$ and therefore $m(e_1)$ and $m(e_2)$ map into the extremal of d_z, contradicting the initial assumption. It follows that we have $e_1\#e_2$ in E' and so m reflects conflict.

That $m(e_1) = m(e_2)$ implies $e_1 = e_2$ or $e_1\#e_2$ or $m(e_1)$ is persistent is trivially true as all events in $Dem(E)$ are persistent.

We next show $m[e] \subseteq [m(e)]$. If $e_1 \leq e_2$ then $(E' \upharpoonright [e_1],\ d)$ is a restriction of $(E' \upharpoonright [e_2],\ d)$ and therefore, from Lemma 6.3.15, there is a morphism from $(x_1,\ d_{x_1})$ to $(x_2,\ d_{x_2})$. This implies $m(e_1) \leq m(e_2)$ and so $m[e] \subseteq [m(e)]$ holds for all $e \in E'$.

Finally, we show $[m(e)] \subseteq m[e]$. If $(x,\ d_x) \leq m(e_1)$ for some $(x,\ d_x)$ in $Dem(E)$ then there is a morphism $f : (x,\ d_x) \to (x_1,\ d_{x_1})$ in $\mathcal{D}_E$. We show that there is a restriction $(x',\ d_{x'})$ of $(E' \upharpoonright [e_1],\ d)$ and an epimorphism $f' : (x',\ d_{x'}) \to (x,\ d_x)$. Let $m_{[e_1]} : (E' \upharpoonright [e_1],\ d) \to (x_1,\ d_{x_1}))$ be the unique morphism from $(E' \upharpoonright [e_1],\ d)$ into its extremal. Define x' to be the largest restriction of $(E' \upharpoonright [e_1])$ such that $m_{[e_1]}(e) \in f(x)$ for all $e \in x'$. Define f' by

$$f'(e) \overset{def}{=} f^{-1} \circ m_{[e_1]}(e)$$

for all $e \in x'$. This is well defined because, from Proposition 6.3.9, the morphism f is injective. We now show that f' is rigid. As f is rigid,

$$f[(f^{-1} \circ m_{[e_1]})(e)] = [m_{[e_1]}(e)] = m_{[e_1]}[e]$$

holds for all $e \in x'$ and so $(f^{-1} \circ m_{[e_1]})[e] = [f^{-1} \circ m_{[e_1]}(e)]$. Next we show that f' preserves persistence. If e is persistent in x' then it is persistent in $(E' \upharpoonright [e_1])$ and so $m_{[e_1]}(e)$ is persistent. From Proposition 6.3.9, the morphism f must reflect persistence and so f^{-1} preserves it. Therefore $f'(e)$ is persistent. Finally, we show that $f'(e) = f'(e')$

implies $e = e'$ or $f'(e)$ is persistent for all $e,\ e' \in x'$. Suppose $f'(e) = f'(e')$ for distinct $e,\ e' \in x'$. As f is injective, it must be the case that $m_{[e_1]}(e) = m_{[e_1]}(e')$. So, as $m_{[e_1]}$ is a morphism, $m_{[e_1]}(e)$ is persistent in x_1. As f reflects persistence we have that $(f^{-1} \circ m_{[e_1]})(e)$ is persistent.

Taking $d_{x'} : x' \to E$ to be the restriction of d to x' we clearly have that $d_{x'} = d_x \circ f'$ as required. It therefore follows from Proposition 6.3.10 that $(x,\ d_x)$ is the extremal for $d_{x'}$.

As $(x,\ d_x)$ is prime and we know that $f'(x') = x$, we know that there must exist $e \in x'$ such that $f[e] = x$ as f is rigid. So there is a prime restriction $(x' \upharpoonright [e],\ d_{x'})$ of $(x',\ d_{x'})$ with a surjective morphism into $(x,\ d_x)$. It follows from Proposition 6.3.10 that $(x,\ d_x)$ will be its extremal and therefore that $m(e) = (x,\ d_x)$. Also, as all elements in x' are below e_1 we have $e \leq e_1$ and therefore $(x,\ d_x) \in m[e_1]$. So, $[m(e)] \subseteq m[e]$ for all $e \in E'$.
Uniqueness of m: Suppose there exists a rigid morphism $m' : E' \to \mathit{Dem}(E)$ for which $\epsilon_E \circ m' = d$. Suppose $m'(e) = (x,\ d_x)$ for some $e \in E'$ and $(x,\ d_x) \in \mathit{Dem}(E)$. We show that $(x,\ d_x)$ must equal $m(e)$. In order to prove this, we first show the following. Let x' be the restriction of $\mathit{Dem}(E)$ to $[(x,\ d_x)]$ and let $d_{x'}$ be the restriction of ϵ_E to x'. Then $(x',\ d_{x'})$ is isomorphic to $(x,\ d_x)$. We define an isomorphism $f : x \to x'$. Define $f : x \to x'$ by $f(e) \overset{\mathit{def}}{=} (z,\ d_z)$ where $(z,\ d_z)$ is the event in x' that is isomorphic to $([e],\ d_x)$ (the restriction of $(x,\ d_x)$ to the events $[e]$). It follows from Proposition 6.3.12 that all restrictions of $(x,\ d_x)$ are extremal. It is clear that $([e],\ d_x)$ is prime and the restriction corresponds to a map from $([e]_,\ d_x)$ to $(x,\ d_x)$. So, as $\mathit{Dem}(E)$ is constructed from a skeletal category, there is a unique event $(z,\ d_z)$ in x' that is isomorphic to $([e],\ d_x)$ and a member of $[(x,\ d_x)]$. Therefore f is well-defined. We now show that f is a rigid morphism. First we demonstrate $[f(e)] \subseteq f[e]$ for all $e \in x$. Assume $(y,\ d_y) \leq f(e)$. Then there is a morphism $g : (y,\ d_y) \to ([e],\ d_x)$ in $\mathcal{D}_E$. As g is rigid and y is a prime elementary event structure, $g(y) = [e']$ where $e' \in [e]$. So g can be seen as a surjective morphism into $([e'],\ d_x)$, the restriction of $([e],\ d_x)$ to $([e'],\ d_x)$ and so $(y,\ d_y)$ is isomorphic to $([e'],\ d_x)$ from Proposition 6.3.9. As $\mathit{Skl}(\mathcal{D}_E)$ is skeletal, this gives us $f(e') = (y,\ d_y)$ and therefore $[f(e)] \subseteq f[e]$. We next show $f[e] \subseteq [f(e)]$ for all $e \in x$. Assume $e \leq e'$ for some $e,\ e' \in x$. This implies that $([e],\ d_x)$ is a restriction of $([e'],\ d_x)$. So, there must be a morphism between $f(e)$ and $f(e')$ in $\mathcal{D}_E$. Therefore $f(e) \leq f(e')$ in $\mathit{Dem}(E)$. So, $f[e] \subseteq [f(e)]$ for all $e \in x$. That f preserves persistence follows directly from the definition of $\mathit{Dem}(E)$. Finally, f is injective due to the uniqueness up to isomorphism of prime extremals. Thus f is a rigid morphism between x and x'.

In order to show that f is an isomorphism, we observe that it is a morphism between $(x,\ d_x)$ and $(x',\ d_{x'})$ in $\mathcal{D}_E$ and show that it is a surjective function. The required result then follows from Proposition 6.3.9. If $(z,\ d_z)$ is an event in x' then it is isomorphic to a prime restriction of $(x,\ d_x)$. Therefore there exists an event e in x such that $(x \upharpoonright [e],\ d_x)$ is isomorphic to $(z,\ d_z)$. So, there exists $e \in x$ such that $f(e) = (z,\ d_z)$. It follows that the morphism f is therefore surjective.

As $m'[e] = [(x,\ d_x)]$, there is a surjective morphism from $(E' \upharpoonright [e],\ d)$ into $(x,\ d_x)$. However, from Proposition 6.3.10, $(x,\ d_x)$ must be the extremal of $(E \upharpoonright [e],\ d)$ and therefore $m(e) = (x,\ d_x)$. It follows that $m = m'$. □

Definition 6.3.17. *Let $d : E \to E'$ be a demand morphism. Define the action of Dem on morphisms by letting $Dem(d) : Dem(E) \to Dem(E')$ be the unique rigid morphism that makes the following diagram commute (see Proposition 6.3.16).*

$$\begin{tikzcd}
Dem(E) \arrow[r, "\epsilon_E"] \arrow[d, dotted, "Dem(d)"'] & E \arrow[d, "d"] \\
Dem(E') \arrow[r, "\epsilon_{E'}"'] & E'
\end{tikzcd}$$

Theorem 6.3.18. *The functor Dem is right adjoint to F.*

The result below follows from Theorem 6.3.18 and Proposition 5.1.6.

Corollary 6.3.19. *The Kleisli category of $(Dem \circ F,\ \eta,\ Dem \circ \epsilon \circ F)$ (where η is the unit of the adjunction) is isomorphic to $\mathcal{D}_P$.* □

6.4 The Category of Augmentation Maps as a Kleisli Category

In this section we use rigid pairs to construct a right adjoint to the inclusion functor from the category of rigid morphisms $\mathcal{R}_P$ to the category of augmentation morphisms $\mathcal{A}_P$. We describe an object $Aug_P(E)$ in $\mathcal{R}_P$ and an augmentation morphism $\epsilon_E : I(Aug_P(E)) \to E$ that is universal from the inclusion functor $I : \mathcal{R}_P \to \mathcal{A}_P$ to E for each event structure E.

Let x be an elementary event structure and let $f : x \to E$ be an augmentation morphism. Define R by

$$e_1 R e_2 \overset{def}{\Leftrightarrow} f(e_1) = f(e_2)$$

for $e_1,\ e_2 \in x$ and define $P(e_1)$ to be true iff $f(e_1)$ is persistent in E. Clearly, $(R,\ P)$ is a pre-rigid pair for x. As for demand morphisms, we sometimes use f to denote the pre-rigid pair to which it corresponds.

Lemma 6.4.1. *Let x be an elementary event structure. Let $(R,\ P)$ be the pre-rigid pair corresponding to an augmentation morphism $f : x \to E$.*
i) The rigid sub-pairs $(R',\ P')$ of $(R,\ P)$ yield factorisations of f into $\langle R',\ P'\rangle : x \to x'$ followed by an augmentation morphism f' defined by

$$f'[e]_{R'} \overset{def}{=} f(e)$$

for all $[e]_{R'}$ in the image of $\langle R',\ P'\rangle$.
ii) Any factorisation of f into a rigid epimorphism followed by an augmentation morphism amounts to a rigid sub-pair of $(R,\ P)$.

Proof. i) It follows from the definition of R that f' is well-defined. It is obvious that $f' \circ \langle R',\ P' \rangle = f$. We show that f' is an augmentation morphism.

In the following, let $[e_1]_{R'}$ and $[e_2]_{R'}$ be events in x'.

We first show that f' preserves persistence. Suppose that $[e_1]_{R'}$ is persistent. Then $P'(e_1)$ holds and therefore, as $P'(e_1)$ implies $P(e_1)$, we have that $f(e_1)$ and therefore $f'([e_1]_{R'})$ is persistent.

Suppose $f'[e_1]_{R'} = f'[e_2]_{R'}$. We show $[e_1]_{R'} = [e_2]_{R'}$ or $f'[e_1]_{R'}$ is persistent. If $f'([e_1]_{R'}) = f'([e_2]_{R'})$ then $f(e_1) = f(e_2)$. As f is an augmentation morphism and e_1 is not in conflict with e_2, this implies $e_1 = e_2$ or $f(e_1)$ is persistent. So, either $[e_1]_{R'} = [e_2]_{R'}$ or $f'[e_1]_{R'}$ is persistent.

We now show $[f'[e_1]_{R'}] \subseteq f'[[e_1]_{R'}]$. Suppose $e \leq f'[e_1]_{R'}$ for $e \in E$. Then $e \leq f(e_1)$ and, from the properties of f, we know $e \in f[e_1]$. Let e' be an event in $[e_1]$ for which $f(e')$ equals e. As $e' \leq e_1$ holds we have $[e']_{R'} \leq [e_1]_R$ and therefore e is a member of $f'[[e_1]_{R'}]$. This concludes our proof of (i).

ii) Suppose f can be factorised into a rigid morphism g followed by an augmentation morphism h. Let $(R_g,\ P_g)$ be the rigid pair corresponding to g. We show $(R_g,\ P_g) \subseteq (R,\ P)$. As $h \circ g = f$ and h preserves persistence, P_g is a subset of P. Also, $g(e_1) = g(e_2)$ implies $f(e_1) = f(e_2)$ so R_g is a subset of R. This implies

$$(R_g,\ P_g) \subseteq (R,\ P),$$

thus concluding the proof of (ii). □

Let x be an elementary event structure, $f : x \to E$ an augmentation morphism with associated pre-rigid pair $(R,\ P)$ and $(X,\ P)$ the greatest rigid sub-pair of $(R,\ P)$ with associated rigid epimorphism $[\_]_X : x \to x_0$. The factorisation of f into $f_0 \circ [\_]_X$, where $f_0 : x_0 \to E$ is the augmentation morphism defined by $f_0[e]_X = f(e)$ for all $e \in x$, has the following property. For any rigid epimorphism $g : x \to x'$ and augmentation morphism $f' : x' \to E$, if $f' \circ g = f$ then the following diagram commutes for a unique rigid morphism m.

$$\begin{tikzcd}
x \arrow[rr, "f"] \arrow[dr, "{[\_]_X}"] \arrow[ddr, bend right, "g"'] & & E \\
& x_0 \arrow[ur, "f_0"'] & \\
& x' \arrow[u, dotted, "m"'] \arrow[uur, bend right, "f'"'] &
\end{tikzcd}$$

Proof. It follows from Proposition 6.2.7 that a unique rigid morphism $m : x' \to x_0$ exists such that $m \circ g = [\_]_X$. It remains to check that $f_0 \circ m = f'$. We have $f_0 \circ m \circ g = f' \circ g$. So, as g is an epimorphism, $f_0 \circ m = f'$. □

In Section 5.6, we defined the category of augmentations $\mathcal{A}_E$ for an event structure E (Definitions 5.6.1 and 5.6.2). We recall the definition here. The objects in $\mathcal{A}_E$ are pairs $(x,\ f)$ where x is an elementary event structure and $f : x \to E$ an augmentation morphism. These pairs are called augmentations. If $(x,\ f)$ and $(x',\ f')$ are objects of $\mathcal{A}_E$

and $g : x \to x'$ is a morphism in $\mathcal{R}_P$ then g is a morphism between $(x,\ f)$ and $(x',\ f')$ if $f' \circ g = f$.

Definition 6.4.2. *The pair $(x_0,\ f_0)$ in $\mathcal{A}_E$ is extremal for the augmentation morphism $f : x \to E$ if there exists a rigid epimorphism $g : (x,\ f) \to (x_0,\ f_0)$ in $\mathcal{A}_E$ such that $(R_g,\ P_g)$ is the extremal of f. The pair $(x_0,\ f_0)$ is* extremal *in $\mathcal{A}_E$ if it is extremal for some $f : x \to E$. It is prime if x_0 is prime, i.e., iff there exists an $e \in x_0$ such that $e' \leq e$ for all $e' \in x_0$.*

Note that, if $(x_0,\ f_0)$ is extremal in $\mathcal{A}_E$, then it is extremal for f_0 (follows immediately from Proposition 6.2.7).

The prime extremals are precisely those augmentations in $\mathcal{A}_E$ with the properties described in Section 5.6. We use them to define a functor from $\mathcal{A}_P$ to $\mathcal{R}_P$.

As for $\mathcal{D}_E$ we can define when one augmentation is the restriction of another.

Definition 6.4.3. *An object $(x',\ f')$ of $\mathcal{A}_E$ is the restriction of $(x,\ f)$ in $\mathcal{A}_E$ if x' is a restriction of x and $f'(e) = f(e)$ for all $e \in x'$.*

Let $Skl(\mathcal{A}_E)$ be the skeleton of $\mathcal{A}_E$.

Definition 6.4.4. *If $(E,\ P,\ \leq,\ \#)$ is an event structure, define $Aug_P(E)$ to be the event structure $(E',\ P',\ \leq',\ \#')$ where*

- *E' is the set of prime extremals in $Skl(\mathcal{A}_E)$,*
- *$P' = \{x \in E' \mid max(x) \in P\}$,*
- *$e \leq' e'$ iff there exists a morphism $g : e \to e'$ in $\mathcal{A}_E$,*
- *$e\#'e'$ iff there is no object in $\mathcal{A}_E$ with morphisms from e and e' into it.*

Observe that $\mathcal{A}_E$ is for augmentation morphisms what $\mathcal{D}_E$ was for demand morphisms. Many of the properties proved in the previous section do not rely on the objects in $\mathcal{D}_E$ being based on demand morphisms rather than augmentation morphisms. The following result is obtained in a similar manner to Theorem 6.3.13. The result does not depend in any way on the properties of demand morphisms; it only relies on demand morphisms being functions with events of event structures as their domains. By definition, this is also true of augmentation morphisms.

Theorem 6.4.5. *Aug_P(E) is an event structure for all event structures E.* □

Define $\epsilon_E : Aug_P(E) \to E$ by

$$\epsilon_E(x,\ g) \ \overset{def}{=}\ g(max(x))$$

for all events $(x,\ g) \in Aug_P(E)$.

Proposition 6.4.6. *The operation ϵ_E is an augmentation morphism from $Aug_P(E)$ to E.*

Proof. Let $(x_1,\ g_1)$ and $(x_2,\ g_2)$ be events in $Aug_P(E)$.

We first show that $\epsilon_E : Aug_P(E) \to E$ preserves persistence. If $(x_1,\ g_1)$ is persistent then $max(x_1)$ is persistent and, as g_1 preserves persistence, $g_1(max(x_1))$ is persistent. So, ϵ_E preserves persistence.

Next we show that ϵ_E reflects conflict. Assume $g_1(max(x_1))\#g_2(max(x_2))$ in E. There cannot be $(y,\ g) \in \mathcal{A}_E$ such that both $g_1(max(x_1))$ and $g_2(max(x_2))$ are in $g(y)$ as g would not reflect conflict. So, there cannot be an object in $\mathcal{A}_E$ that both $(x_1,\ g_1)$ and $(x_2,\ g_2)$ map to, implying that they are in conflict and therefore that ϵ_E reflects conflict.

Suppose $\epsilon_E(x_1,\ g_1) = \epsilon_E(x_2,\ g_2)$, i.e., $g_1(max(x_1)) = g_2(max(x_2))$. We show that $(x_1,\ g_1)$ equals $(x_2,\ g_2)$ or $(x_1,\ g_1)\#(x_2,\ g_2)$ or $\epsilon_E(x_1,\ g_1)$ is persistent. Suppose $(x_1,\ g_1)$ is not equal to $(x_2,\ g_2)$. If $(x_1,\ g_1)$ is not in conflict with $(x_2,\ g_2)$ then there must exist $(y,\ g)$ in $\mathcal{A}_E$ that they both map to. Therefore $g_1(max(x_1))$ and $g_2(\max(x_2))$ are in the image of g. Let h_i be the morphism mapping $(x_i,\ g_i)$ into $(y,\ g)$ for $i = 1, 2$. The h_i are injective and reflect persistence (by analogy to Proposition 6.3.9). It follows that $(x_1,\ g_1)$ is isomorphic to the restriction of $(y,\ g)$ to the image of h_1 and $(x_2,\ g_2)$ is isomorphic to the restriction to the image of h_2. So, $h_1(max(x_1))$ cannot be equal to $h_2(max(x_2))$ otherwise $(x_1,\ g_1) = (x_2,\ g_2)$, contradicting one of the initial assumptions. It follows that g maps two distinct events in y to $g_1(max(x_1))$ and therefore that $g_1(max(x_1))$ is a persistent event.

Let $(x,\ f)$ be an event of $Aug_P(E)$. We conclude our proof by showing $[\epsilon_E(x,\ f)] \subseteq \epsilon_E[(x,\ f)]$. Suppose $e \leq \epsilon_E(x,\ f)$ in E. Then e is in $[f(max(x))]$ and, as f is an augmentation morphism, e is a member of $f[max(x)]$, i.e., there exists an event $e' \in x$ that f maps to e. The restriction of $(x,\ f)$ to the events $[e']$ is an extremal (by analogy to Proposition 6.3.12) and is clearly prime. Also, ϵ_E will map it to e. So, e is a member of $\epsilon_E[(x,\ f)]$ and therefore $[\epsilon_E(x)] \subseteq \epsilon_E[x]$ holds for all $(x,\ f) \in Aug_P(E)$. □

Proposition 6.4.7. *For every augmentation morphism $f : E' \to E$, there exists a unique rigid morphism $m : E' \to Aug_P(E)$ such that the following diagram commutes.*

$$\begin{tikzcd} Aug_P(E) \arrow[r, "\epsilon_E"] & E \\ E' \arrow[u, dotted, "m"] \arrow[ur, "f"'] & \end{tikzcd}$$

This can be proved exactly as Proposition 6.3.16 is proved with the exception of showing that m preserves persistence and that $m(e_1) = m(e_2)$ implies $e_1 = e_2$ or $e_1\#e_2$ or $m(e_1)$ is persistent for all $e_1,\ e_2 \in E'$. We show this below.

Recall that $m : E' \to Aug_P(E)$ is defined for all $e \in E'$ by $m(e) \overset{def}{=} (x,\ g)$ where $(x,\ g)$ is the extremal of f restricted to $[e]$ in $Skl(\mathcal{A}_E)$.

Suppose $e \in E'$ is persistent. Then $f(e)$ is persistent in E. So, from the definition of m, there is a rigid morphism from $(E' \upharpoonright [e],\ f)$ into $m(e)$. This implies that the maximum event in the elementary event structure of $m(e)$ must be persistent. So, from the definition of $Aug_P(E)$, $m(e)$ is persistent.

Let e_1 and e_2 be events in E'. Let $(x_1,\ g_1) = m(e_1)$ and $(x_2,\ g_2) = m(e_2)$. Suppose $(x_1,\ g_1) = (x_2,\ g_2)$ but $e_1 \neq e_2$. As $\epsilon_E \circ m = f$ it follows that we have $f(e_1) = f(e_2)$.

This implies $e_1 \# e_2$ or $f(e_1)$ is persistent as f is an augmentation morphism. It remains to show that $f(e_1)$ being persistent implies $(x_1,\ g_1)$ is persistent. We know that, because $\epsilon_E \circ m = f$, we have $f(e_1) = g_1(max(x_1))$. Also, from Proposition 6.2.7, as $(x_1,\ g_1)$ is extremal, it is extremal for g_1 and therefore $g_1(max(x_1))$ being persistent implies $max(x_1)$ is persistent. Therefore, from the definition of $Aug_P(E)$, $(x_1,\ g_1)$ is persistent.

Definition 6.4.8. *Let $f : E \to E'$ be an augmentation morphism. Define $Aug_P(f)$ to be the unique morphism $Aug_P(E) \to Aug_P(E')$ that makes the diagram below commute.*

$$\begin{tikzcd}
Aug_P(E) \arrow[r, "\epsilon_E"] \arrow[d, dotted] & E \arrow[d, "f"] \\
Aug_P(E') \arrow[r, "\epsilon_{E'}"'] & E'
\end{tikzcd}$$

Theorem 6.4.9. *Aug_P is right adjoint to the inclusion functor $I : \mathcal{R}_P \to \mathcal{A}_P$.* □

The result below follows from Theorem 6.4.9 and Proposition 5.1.6.

Corollary 6.4.10. *The Kleisli category of $(Aug_P \circ I,\ \eta,\ Aug_P \circ \epsilon \circ I)$ (where η is the unit of the adjunction) is isomorphic to $\mathcal{A}_P$.* □

As there is an adjunction between $\mathcal{R}_P$ and $\mathcal{A}_P$ and an adjunction between $\mathcal{A}_P$ and $\mathcal{P}_P$ (Section 5.5), we have an adjunction between $\mathcal{R}_P$ and $\mathcal{P}_P$. In fact we can use the rigid pairs method to define the right adjoint to the inclusion functor from $\mathcal{R}_P$ into $\mathcal{P}_P$ directly, i.e., not in terms of the composition of two functors. If f is a partial morphism from an elementary event structure x to E then we can produce a pre-rigid pair on the events of x where, for all $e_1,\ e_2,\ e \in x$, R is defined by $e_1 R e_2$ iff $f(e_1) = f(e_2)$ and P is defined by $P(e)$ iff $f(e)$ is persistent or $f(e)$ is undefined. The rigid sub-pairs of $(R,\ P)$ then correspond to factorisations of f into a rigid morphism and a partial morphism. Following the same method as for demand morphisms and augmentation morphisms, the right adjoint to the inclusion functor that is produced is isomorphic to that given by the composition $Aug_P \circ Par$.

6.5 Limits in $\mathcal{R}_P$

Unlike for the categories of event structures without persistence, we do not have corresponding categories of stable families in which to construct limits. In this section we show that the category of event structures with persistence and rigid morphisms $\mathcal{R}_P$ has a terminal object. We then go on to use the method of rigid pairs to show that it has all pullbacks. It follows that $\mathcal{R}_P$ has all finite limits.

Theorem 6.5.1. *The event structure E consisting of a single persistent event p is terminal in $\mathcal{R}_P$.*

Proof. Let $E' \in \mathcal{R}_P$. We show there is a unique morphism $m : E' \to E$. Define m by $m(e) \stackrel{\mathit{def}}{=} p$ for all $e \in E'$. It is trivial to show that m is a rigid morphism. That m is unique follows immediately from there being only one possible function from E' to E. □

Theorem 6.5.2. *The category $\mathcal{R}_P$ has all pullbacks.*

The rest of this section is devoted to proving Theorem 6.5.2.

Throughout this section, we fix a cospan $E_1 \xrightarrow{h} E_3 \xleftarrow{j} E_2$ in $\mathcal{R}_P$. We construct a pullback for this cospan.

Let x be an elementary event structure. Suppose that the following diagram commutes.

$$\begin{tikzcd}
x \arrow[r, "g"] \arrow[d, "f"'] & E_2 \arrow[d, "j"] \\
E_1 \arrow[r, "h"'] & E_3
\end{tikzcd}$$

Define $(R,\ P)$ on events in x as follows.

$$\begin{array}{rll} e_1 R e_2 & \text{iff} & f(e_1) = f(e_2) \text{ and } g(e_1) = g(e_2) \\ P(e) & \text{iff} & f(e) \text{ and } g(e) \text{ are persistent.} \end{array}$$

Clearly, $(R,\ P)$ is a pre-rigid pair. The rigid sub-pairs $(R',\ P')$ of $(R,\ P)$ correspond to factorisations of f and g into $f' \circ [\_]_{R'}$ and $g' \circ [\_]_{R'}$. We define f' and g' by $f'[e]_{R'} \stackrel{\mathit{def}}{=} f(e)$ and $g'[e]_{R'} \stackrel{\mathit{def}}{=} g(e)$. The functions f' and g' are well-defined as if $e_1,\ e_2 \in [e]_{R'}$ then $f(e_1) = f(e_2)$ and $g(e_1) = g(e_2)$, because $(R',\ P')$ is a rigid subpair of $(R,\ P)$. The following diagram commutes in **Set**.

$$\begin{tikzcd}
 & x \arrow[d, "{[\_]_{R'}}"] \arrow[ddl, bend right, "f"'] \arrow[ddr, bend left, "g"] & \\
 & x' \arrow[dl, "f'"'] \arrow[dr, "g'"] & \\
E_1 \arrow[dr, "h"'] & & E_2 \arrow[dl, "j"] \\
 & C &
\end{tikzcd}$$

In fact, f' and g' are rigid morphisms. For instance, we show the rigidity of f' follows from the rigidity of f and $[\_]_{R'}$. Let $[e]_{R'} \in x'$. As f is rigid and $f' \circ [\_]_{R'} = f$ we have

$$[f'[e]_{R'}] = f[e] = (f' \circ [\_]_{R'})[e],$$

and as $[\_]_{R'}$ is rigid we have $(f' \circ [\_]_{R'})[e] = f'[[e]_{R'}]$. If $[e]_{R'}$ is persistent then $f(e)$ is persistent and therefore $f'[e]_{R'}$ is persistent. Therefore f' preserves persistence. Suppose $f'[e_1]_{R'} = f'[e_2]_{R'}$ for distinct $[e_1]_{R'}$ and $[e_2]_{R'}$ in x'. Then $e_1 \neq e_2$ but $f(e_1) = f(e_2)$. So, from the properties of f, either $e_1\#e_2$ or $f(e_1)$ and therefore $f'[e_1]_{R'}$ is persistent. (The rigidity of g' is proved via an identical argument.)

Let $\mathcal{C}$ be the category with objects $(x,\ f,\ g)$ where x is an elementary event structure and $f : x \to E_1$ and $g : x \to E_2$ are rigid morphisms such that $h \circ f = j \circ g$. The morphisms $k : (x,\ f,\ g) \to (x',\ f',\ g')$ of $\mathcal{C}$ are rigid morphisms $k : x \to x'$ such that $f = f' \circ k$ and $g = g' \circ k$.

$$\begin{tikzcd}
 & x \arrow[dl, "f"'] \arrow[d, "k"] \arrow[dr, "g"] & \\
E_1 \arrow[dr, "h"'] & x' \arrow[l, "f'"] \arrow[r, "g'"'] & E_2 \arrow[dl, "j"] \\
 & E_3 &
\end{tikzcd}$$

Observe that each $(x,\ f,\ g)$ in $\mathcal{C}$ corresponds to a pre-rigid pair as described previously.

As for $\mathcal{D}_E$ and $\mathcal{A}_E$, we can describe a notion of restriction for objects in $\mathcal{C}$. We say $(x',\ f',\ g')$ is a restriction of $(x,\ f,\ g)$ in $\mathcal{C}$ if x' is a restriction of x and $f'(e) = f(e)$ and $g'(e) = g(e)$ for all $e \in x'$. Note that we sometimes write $(x',\ f,\ g)$ for $(x',\ f',\ g')$.

Definition 6.5.3. *The triple $(x_0,\ f_0,\ g_0)$ is extremal in $\mathcal{C}$ if there exists a rigid epimorphism $h : (x,\ f,\ g) \to (x_0,\ f_0,\ g_0)$ in $\mathcal{C}$ such that $(R_h,\ P_h)$ is the extremal of the pre-rigid pair corresponding to $(x,\ f,\ g)$. It is prime if x_0 is prime, i.e., iff there exists an $e \in x_0$ such that $e' \leq e$ for all $e' \in x_0$.*

Let $\mathit{Skl}(\mathcal{C})$ be the skeleton of $\mathcal{C}$. We define the event structure E_0 in terms of this category as we did in the previous two sections.

The events of E_0 are the prime extremals in $\mathit{Skl}(\mathcal{C})$; the causality and conflict relations of E_0 are defined as follows.

$$\begin{array}{rl}
x_1 \leq x_2 & \text{if there is a morphism } k : x_1 \to x_2 \text{ in } \mathcal{C}; \\
x_1 \# x_2 & \text{if there is no object in } \mathcal{C} \text{ with a} \\
 & \text{morphism from both } x_1 \text{ and } x_2 \text{ into it.}
\end{array}$$

The event $(x,\ f,\ g)$ is persistent if $\mathit{max}(x)$ is persistent.

The following result is obtained in a similar manner to Theorem 6.3.13. The result does not depend in any way on the properties of demand morphisms; it only relies on demand morphisms being functions.

Theorem 6.5.4. *The structure E_0 is an event structure.* □

We define rigid morphisms $f_0 : E_0 \to E_1$, $g_0 : E_0 \to E_2$ by

$$\begin{aligned}
f_0(x,\ f,\ g) &\overset{def}{=} f(\mathit{max}(x)), \\
g_0(x,\ f,\ g) &\overset{def}{=} g(\mathit{max}(x)).
\end{aligned}$$

We now explain why the morphisms f_0 and g_0 are morphisms in $\mathcal{R}_P$. We prove that f_0 is rigid; the proof that g_0 is rigid is symmetric.

Let $(x,\ f,\ g)$ be an event in E_0.

We first show that $[f_0(x,\ f,\ g)] \subseteq f_0[(x,\ f,\ g)]$. Suppose $e \in [f_0(x,\ f,\ g)]$. From the definition, e is in $[f(max(x))]$ and therefore, as f is rigid, it is in $f[max(x)]$. As x is prime, $[max(x)] = x$ so e is in $f(x)$. Let $e' \in x$ be an event for which $f(e') = e$. We can restrict x to the events in $[e']$ to produce a prime extremal $(x',\ f,\ g)$ (by analogy to Proposition 6.3.12). Clearly there is a morphism from $(x',\ f,\ g)$ to $(x,\ f,\ g)$ and $f(max(x')) = e$ and so e is in $f_0[(x,\ f,\ g)]$. Thus we have shown

$$[f_0(x,\ f,\ g)] \subseteq f_0[(x,\ f,\ g)].$$

We now show that $f_0[(x,\ f,\ g)] \subseteq [f_0(x,\ f,\ g)]$. Suppose $e \in f_0[(x,\ f,\ g)]$, i.e., there exists $(x',\ f',\ g') \in [(x,\ f,\ g)]$ such that $f'(max(x')) = e$. As $(x',\ f',\ g') \leq (x,\ f,\ g)$, there is a morphism $h : (x',\ f',\ g') \to (x,\ f,\ g)$ in $\mathcal{C}$ and $f(h(max(x')) = e$. Now, $h(max(x'))$ is in x and, as x is prime, we have $h(max(x')) \leq max(x)$ and therefore $f(h(max(x'))) \leq f(max(x))$ as f is rigid. Therefore e is in $[f_0(x,\ f,\ g)]$. Thus we have shown

$$f_0[(x,\ f,\ g)] \subseteq [f_0(x,\ f,\ g)].$$

We next show that f_0 preserves persistence. If an event $(x,\ f,\ g)$ is persistent then, from the definition of E_0, it must be the case that $max(x)$ is persistent and therefore $f(max(x))$ is persistent since f is rigid. Thus $f_0(x,\ f,\ g)$ is persistent.

Let $(x,\ f,\ g)$ and $(x',\ f',\ g')$ be events in E_0. We suppose $f_0(x,\ f,\ g)$ is equal to $f_0(x',\ f',\ g')$ and we will show that either $(x,\ f,\ g) = (x',\ f',\ g')$ or $(x,\ f,\ g)\#(x',\ f',\ g')$ or $f_0(x,\ f,\ g)$ is persistent. We know $f(max(x))$ equals $f'(max(x'))$. Suppose that $(x,\ f,\ g) \neq (x',\ f',\ g')$ and that they are not in conflict. We will show that $f_0(x,\ f,\ g)$ is persistent. There is an object $(y,\ f_y,\ g_y)$ in $\mathcal{C}$ into which both $(x,\ f,\ g)$ and $(x',\ f',\ g')$ map. Therefore, because $(x,\ f,\ g) \neq (x',\ f',\ g')$, we know y contains two distinct events which are mapped to $f(max(x))$ by f_y. Hence $f(max(x))$ is persistent.

This concludes our proof that f_0 is rigid.

Theorem 6.5.5. *The diagram below is a pullback diagram in $\mathcal{R}_P$.*

$$\begin{tikzcd}
E_0 \arrow[r, "g_0"] \arrow[d, "f_0"'] & E_2 \arrow[d, "j"] \\
E_1 \arrow[r, "h"'] & E_3
\end{tikzcd}$$

Proof. Suppose there exists an event structure E and morphisms f and g such that the following diagram commutes.

$$\begin{tikzcd}
E \arrow[r, "g"] \arrow[d, "f"'] & E_2 \arrow[d, "j"] \\
E_1 \arrow[r, "h"'] & E_3
\end{tikzcd}$$

Define $m : E \to E_0$ on events of E by setting $m(e)$ as the extremal of $(E \upharpoonright [e],\ f,\ g)$. The proof that m is a rigid morphism and unique is analogous to the proof of Proposition 6.4.7. □

This concludes our proof of Theorem 6.5.2.

So, using rigid pairs, we have shown that $\mathcal{R}_P$ has all finite limits. However, the rigid pairs method does not allow us to construct limits in $\mathcal{A}_P$ or $\mathcal{P}_P$. This is because augmentation and partial morphisms may not preserve causality. We therefore have no notion of extremals for these categories. We show that $\mathcal{A}_P$ does not have all finite limits. A similar counter-example can be found for $\mathcal{P}_P$.

Proposition 6.5.6. *The category of augmentation morphisms $\mathcal{A}_P$ does not have all finite limits.*

Proof. We show the proposition by contradiction.

Let E_1 be defined by

$$\begin{tikzcd} e_2 \\ e_1 \arrow[u] \end{tikzcd}$$

where e_1 is persistent.

Let E_2 be defined by

$$e_3 \qquad e_4 \qquad e_5$$

where e_4 and e_5 are persistent.

Assume P is the product of E_1 and E_2.

$$\begin{tikzcd} E_1 & P \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] & E_2 \end{tikzcd}$$

Observe that E_1 must be isomorphic to a restriction of P. This is because there must exist a morphism $m : E_1 \to P$ such that the following diagram commutes (where $f : E_1 \to E_2$ is defined by $f(e_1) = e_4$ and $f(e_2) = e_3$).

$$\begin{tikzcd} & E_1 \arrow[dl, "id"'] \arrow[d, "m"] \arrow[dr, "f"] & \\ E_1 & P \arrow[l, "\pi_1"] \arrow[r, "\pi_2"'] & E_2 \end{tikzcd} \tag{†}$$

In addition, P must contain at least two independent persistent events. This is because there must exist a morphism $m : E_2 \to P$ such that the following diagram commutes.

$$\begin{tikzcd} & E_2 \arrow[dl, "\_ \mapsto e_1"'] \arrow[d, "m"'] \arrow[dr, "id"] & \\ E_1 & P \arrow[l, "\pi_1"] \arrow[r, "\pi_2"'] & E_2 \end{tikzcd}$$

We see that E_3 must be isomorphic to a restriction of P where E_3 is defined by

$$\begin{tikzcd} e_2 & \\ e_1 \arrow[u] & e_3 \end{tikzcd}$$

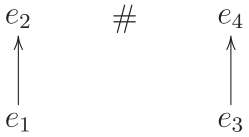
where e_1 and e_3 are persistent and, supposing without loss of generality that E_3 is precisely this restriction of P,

$$\begin{aligned}
\pi_1(e) &= e_1 \quad \text{for } e = e_1,\ e_3,\\
\pi_1(e_2) &= e_2,\\
\pi_2(e_1) &= e_4,\\
\pi_2(e_2) &= e_3,\\
\pi_2(e_3) &= e_5.
\end{aligned}$$

Observe that there are morphisms $g : E_3 \to E_1$ and $h : E_3 \to E_2$ defined by

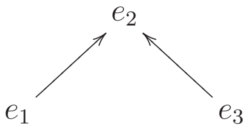
$$\begin{aligned}
g(e) &= \pi_1(e) \quad \text{for all } e \in E_3,\\
h(e_1) &= e_5,\\
h(e_2) &= e_3,\\
h(e_3) &= e_4.
\end{aligned}$$

However, if P is isomorphic to E_3, there will be no morphism $j : E_3 \to P$ such that $\pi_1 \circ j = g$ and $\pi_2 \circ j = h$. It follows from the above and the existence of $m : E_1 \to P$ in diagram (†) that E_4 must be a restriction of P where E_4 is defined by



where e_1 and e_3 are persistent and π_1 and π_2 behave as for E_3 and $\pi_1(e_4) = e_2$ and $\pi_2(e_4) = e_3$.

Let E_5 be defined by



where no events are persistent.

Define $f : E_5 \to E_1$ by

$$\begin{aligned}
f(e) &= e_1 \quad \text{for } e = e_1,\ e_3,\\
f(e_2) &= e_2.
\end{aligned}$$

Define $g : E_5 \to E_2$ by

$$\begin{aligned}
g(e_1) &= e_4,\\
g(e_2) &= e_3,\\
g(e_3) &= e_5.
\end{aligned}$$

Then there are at least two morphisms $m_1 : E_5 \to P$ and $m_2 : E_5 \to P$ such that $\pi_1 \circ m = f$ and $\pi_2 \circ m = g$ for $m = m_1,\ m_2$, i.e., $e_2 \in E_5$ can be mapped to e_2 or e_4 in E_4 within P. As there is no unique $m : E_5 \to P$, we have a contradiction and therefore $\mathcal{A}_P$ does not have all finite limits. □

6.6 Concluding Remarks

As well as allowing us to relate the four categories of event structures $\mathcal{R}_P$, $\mathcal{A}_P$, $\mathcal{P}_P$ and $\mathcal{D}_P$, the results shown in this chapter give us a new way of understanding the spans described in Section 2.4. We can view them as being spans in $\mathcal{R}_P$ of the form

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

where A and B and therefore necessarily E are event structures without persistence.

Chapter 7

Conclusions

Here we summarise the work contained in the previous chapters and suggest what can be concluded from it.

7.1 Summary

After introducing the reader to event structures in Chapter 1, we provided explicit definitions of event structures, their morphisms and the spans based upon them (Chapter 2). In Chapter 3 we recalled the work of Mykkel Nygaard's thesis on an event structure semantics for Affine HOPLA, including a more detailed description of the semantics than has previously been seen. We thus highlighted the curious ability of event structures to represent both processes and types. In the final section of this chapter, the examples of spans representing higher order processes helped to familiarise the reader with the operational nature of the way in which spans can model processes. In Chapter 4, we showed that the bicategory of spans of event structures has a trace construction that allows us to use it to model dataflow processes. This construction corresponds to that given in [15, 16] for profunctors. We also described a restriction of the bicategory of spans to deterministic spans. The trace construction applied to these was shown to be equivalent to Kahn's fixed point construction described in [19] for deterministic dataflow processes.

Several justifications for altering the definition of event structures were given in Chapter 5. We showed that, for the classical event structure category of rigid morphisms $\mathcal{R}$ and that of augmentation morphisms $\mathcal{A}$, there exists a right adjoint to the inclusion of $\mathcal{R}$ in $\mathcal{A}$

$$\begin{tikzcd}
\mathcal{R} \arrow[r, hook, bend right] \arrow[r, phantom, "\top"] & \mathcal{A} \arrow[l, bend right]
\end{tikzcd}$$

and therefore that $\mathcal{A}$ is isomorphic to the Kleisli category of a monad on $\mathcal{R}$. However, it was shown that no such adjunctions exist between $\mathcal{R}$ and the categories of partial morphisms and demand morphisms $\mathcal{P}$ and $\mathcal{D}$. Also, a curious discrepancy, highlighted by the work of Fiore, Cattani and Winskel in [14], was used to generate examples of behaviours that event structures cannot capture. We then defined event structures with persistence and extended the definitions of rigid, augmentation, partial and demand morphisms for classical event structures accordingly. This, together with the methods described in Chapter 6, allowed us to describe adjunctions between the category of event structures with persistence and rigid morphisms $\mathcal{R}_P$ and the categories $\mathcal{A}_P$, $\mathcal{P}_P$ and $\mathcal{D}_P$ of augmentation, partial and demand morphisms.

$$\begin{tikzcd}
\mathcal{R}_P \arrow[r, hook, bend right] \arrow[d, hook, bend left] & \mathcal{A}_P \arrow[l, bend right] \arrow[l, phantom, "\top"] \arrow[r, hook, bend right] & \mathcal{P}_p \arrow[l, bend right] \arrow[l, phantom, "\top"] \\
\mathcal{D}_P \arrow[u, bend left] \arrow[u, phantom, "\vdash"] & &
\end{tikzcd}$$

Thus, we showed that $\mathcal{A}_P$, $\mathcal{P}_P$ and $\mathcal{D}_P$ are isomorphic to Kleisli categories of monads on $\mathcal{R}_P$. We also demonstrated that $\mathcal{R}_P$ has all finite limits. Therefore a bicategory of symmetric spans can be constructed from it.

7.2 Further Work

We examine some remaining issues and further work in this area.

As detailed in Section 5.2, Winskel characterised the presheaves that represent event structures without persistence in [38]. At the time of writing, we do not have a characterisation of the presheaves that represent event structures with persistence. Such a characterisation would help us to relate event structures with persistence to other models more easily. It would also provide us with another way of understanding the additional behaviours that event structures with persistence can capture in comparison with those without persistence.

We have discussed the wish to experiment with spans of event structures with varying morphisms. In order to be able to relate the various kinds of span, we have proposed making use of monads on the category of rigid morphisms. For example, in Section 6.6 we discussed how a span between event structures without persistence A and B, constructed from a demand morphism and a rigid morphism, could be modelled as a span of rigid morphisms

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

where *Dem* is the monad induced by the adjunction described in Section 6.3. This suggests it might be possible to construct a pseudo-comonad on the bicategory of spans of rigid morphism based on *Dem*. As described in Section 1.5, we would like to model bicategories of asymmetric spans as Kleisli and co-Kleisli categories of pseudo-monads and co-monads on the bicategory of spans of rigid morphisms, constructed from the monads on $\mathcal{R}_P$ detailed in Chapters 5 and 6. In [6], Burroni shows the following. A monad on a category $\mathcal{C}$ induces a monad and a comonad on the span category built from the morphisms of $\mathcal{C}$

precisely when the monad is *cartesian*. (A monad $(T,\ \eta,\ \mu)$ is defined to be cartesian when T preserves pullbacks and η and μ are cartesian.)

However, the monads are not cartesian although the monad defined in Section 6.3 does induce the spans defined in Section 2.4 when we restrict our attention to spans between event structures without persistence. Roughly, the monads are not cartesian because distinct, non-conflicting events may be mapped to the same event if it is persistent. Thus the local injectivity property of morphisms between event structures without persistence no-longer exists. However, relating two non-conflicting events in this way has been shown to be necessary for the monads between the event structure categories to exist. Clearly a compromise is needed, allowing two events to be related but not be made identical. Winskel is currently studying *event structures with symmetry* [39]. Such event structures allow different events to be identified as being symmetric.

Definition 7.2.1. *An event structure with symmetry $(E,\ l,\ r)$ consists of an event structure (without persistence) together with two open maps $l : S \to E$ and $r : S \to E$ from a common event structure S such that the map $(l,\ r) : S \to E \times E$ is an equivalence relation, (i.e.,the map $(l,\ r)$ is a monomorphism and satisfies the standard diagrammatic properties of reflexivity, symmetry and transitivity).*

Maps between event structures with symmetry are defined to be maps between event structures which preserve symmetry.

Definition 7.2.2. *If $(A,\ l_A,\ r_A)$ and $(B,\ l_B,\ r_B)$ are event structures with symmetry then $f : (A,\ l_A,\ r_A) \to (B,\ l_B,\ r_B)$ is a map between them if $f : A \to B$ is a map of event structures such that there exists a map of event structures $h : S_A \to S_B$, where S_A is the domain of l_A and r_A and S_B is the domain of l_B and r_B, ensuring*

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

In fact, event structures with persistence are related by an adjunction to event structures with symmetry.

It is hoped that further study of event structures with symmetry will reveal new expressiveness. It is believed that cartesian monads relating the various categories of event structures with symmetry do exist.

7.3 Concluding Remarks

In Chapters 3 and 4 we demonstrated that event structures already provide us with a versatile model for concurrent processes. They capture complex behaviour in a highly intuitive manner, allowing us to give a formal yet still operational description of many kinds of process. Also, as shown in Chapters 5 and 6, a simple extension to the definition allows many categories of event structures to be modelled as Kleisli categories of monads on a more restricted category. This extension also enables event structures to capture additional behaviours without sacrificing their simplicity. We have simply described one step along the road. Various kinds of event structures and their spans have the potential to provide a useful and easily understandable denotational semantics to many process calculi. They, and constructions within the categories based upon them, may inspire new process calculi. Indeed, as discussed in Chapter 1, the possible applications are numerous. Event structures will provide a rich area of research both now and in the future.

Glossary of Symbols

$\mathcal{C}(E)$	The set of configurations of an event structure E, page 19
$\mathcal{P}$	The category of event structures with partial morphisms, page 20
$\mathcal{A}$	The category of event structures with augmentation morphisms, page 20
$\mathcal{R}$	The category of event structures with rigid morphisms, page 20
$\mathcal{D}$	The category of event structures with demand morphisms, page 20
$\mathcal{R}_L$	The category of labelled event structures with labelling set L and rigid, label-preserving morphisms, page 21
$\mathcal{P}_F$	The category of stable families and partial morphisms, page 22
$\mathcal{A}_F$	The category of stable families and augmentation morphisms, page 22
$\mathcal{R}_F$	The category of stable families and rigid morphisms, page 22
$Tr(E)$	The trace of an event structure span E, page 46
$\overline{E}$	The profunctor represented by the span E, page 48
$\tilde{E}$	The continuous function represented by a deterministic span E, page 56
Aug	The right adjoint to the inclusion functor from $\mathcal{R}$ into $\mathcal{A}$, page 62
$\mathcal{R}_I$	The category of event structures with independence events and rigid morphisms, page 66
$\mathcal{A}_I$	The category of event structures with independence events and augmentation morphisms, page 66
$\mathcal{P}_I$	The category of event structures with independence events and partial morphisms, page 66
$\mathcal{D}_I$	The category of event structures with independence events and demand morphisms, page 66
Ptl	The right adjoint to the inclusion functor from $\mathcal{A}_I$ into $\mathcal{P}_I$, page 67

$\mathbf{Pom}_L$ The skeleton of the category of labelled event structures with labelling set L and rigid, label-preserving morphisms, page 69

$\overset{\frown}{E}$ The presheaf represented by the labelled event structure E, page 70

$\mathcal{P}_P$ The category of event structures with persistence and partial morphisms, page 73

$\mathcal{A}_P$ The category of event structures with persistence and augmentation morphisms, page 73

$\mathcal{R}_P$ The category of event structures with persistence and rigid morphisms, page 73

$\mathcal{D}_P$ The category of event structures with persistence and demand morphisms, page 74

Par The right adjoint to the inclusion functor from $\mathcal{A}_P$ into $\mathcal{P}_P$, page 75

$\mathcal{A}_E$ The category of augmentations of the event structure E, page 78

Dem The right adjoint to the inclusion functor from $\mathcal{R}_P$ into $\mathcal{D}_P$, page 88

$\mathcal{D}_E$ The category of demands for the event structure E, page 88

Aug_P The right adjoint to the inclusion functor from $\mathcal{R}_P$ into $\mathcal{A}_P$, page 96

Bibliography

[1] S. Abbes. A cartesian closed category of event structures with quotients. *Discrete Mathematics and Theoretical Computer Science*, 8:249–272, 2006.

[2] G. Berry. *Modèles complement adéquats et stables des λ-calculs typés, Thèse de Doctorat d'Etat*. PhD thesis, Universit. Paris VII, 1979.

[3] F. Borceux. *Handbook of Categorical Algebra I: Basic Category Theory*, volume 50 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, 1994.

[4] G. Boudol and I. Castellani. Permutation of transitions: An event structure semantics for CCS and SCCS. In *Proceedings of the REX School/Workshop on Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, volume 354 of *LNCS*, pages 411–427. Springer, 1988.

[5] J. Brock and W. Ackerman. Scenarios: A model of non-determinate computation. *Formalization of programming concepts*, 107:252–259, 1981.

[6] A. Burroni. T-categories. *Cahiers de Topologie et Géométrie Différentielle*, 12(3):215–324, 1971.

[7] I. Castellani and G. Zhang. Parallel product of event structures. *Theoretical Computer Science*, 179(1–2):203–215, 1997.

[8] G. L. Cattani and G. Winskel. Presheaf models for concurrency. In *Computer Science Logic. 10th International Workshop, CSL '96, Annual Conference of the European Association for Computer Science Logic. Selected Papers*, volume 1258 of *LNCS*, pages 58–75. Springer, 1997.

[9] G. L. Cattani and G. Winskel. Profunctors, open maps and bisimulation. *Mathematical Structures in Computer Science*, 15(3):553–614, 2005.

[10] F. Crazzolara and G. Winskel. Events in security protocols. In *CCS '01, Proceedings of the 8th ACM Conference on Computer and Communication Security*, pages 96–105. ACM, 2001.

[11] F. Crazzolara and G. Winskel. Composing strand spaces. In *FSTTCS '02, Foundations of Software Technology and Theoretical Computer Science – 22nd International Conference*, volume 2556 of *LNCS*, pages 97–108. Springer, 2002.

[12] J. B. Dennis. First version of a dataflow procedure language. In *Proceedings Colloque sur la Programmation*, volume 19 of *LNCS*, pages 362–376. Springer, 1974.

[13] A. A. Faustini. An operational semantics for pure dataflow. In *ICALP'82, Proceedings of the 9th International Colloquium on Automata Languages and Programming*, volume 140 of *LNCS*, pages 212–224. Springer, 1982.

[14] M. P. Fiore, G. L. Cattani, and G. Winskel. Weak bisimulation and open maps. In *LICS '99, Proceedings of the Fourteenth Annual IEEE Symposium on Logic in Computer Science*, pages 67–76. IEEE Computer Society Press, 1999.

[15] T. Hildebrandt, P. Panangaden, and G. Winskel. Relational semantics of nondeterministic dataflow. In *CONCUR'98, Proceedings of the 9th International Conference on Concurrency Theory*, volume 1466 of *LNCS*, pages 613–628. Springer, 1998.

[16] T. Hildebrandt, P. Panangaden, and G. Winskel. A relational model of nondeterministic dataflow. *Mathematical Structures in Computer Science*, 14(5):613–649, 2003.

[17] A. Joyal, M. Nielson, and G. Winskel. Bisimulation from open maps. *Information and Computation*, 127:164–185, 1996.

[18] A. Joyal, R. Street, and D. Verity. Traced monoidal categories. *Mathematical Proceedings of the Cambridge Philosophical Society*, 119:447–468, 1996.

[19] G. Kahn. The semantics of a simple language for parallel programming. *Information Processing*, 74:471–475, 1974.

[20] G. Kahn and G. D. Plotkin. Concrete domains. *Theoretical Computer Science*, 121(1–2):187–277, 1993.

[21] N. A. Lynch and E. W. Stark. A proof of the Kahn principle for input/output automata. *Information and Computation*, 82:81–92, 1989.

[22] S. MacLane. *Categories for the Working Mathematician*, volume 5 of *Graduate Texts in Mathematics*. Springer, second edition, 1998.

[23] R. Milner. A calculus of communicating systems. volume 92 of *LNCS*. Springer, 1980.

[24] E. Moggi. Computational lambda-calculus and monads. In *LICS '89, Proceedings of the Fourth Annual IEEE Symposium on Logic in Computer Science*, pages 14–23. IEEE Computer Society Press, 1989.

[25] M. Nielson, G. D. Plotkin, and G. Winskel. Petri nets, event structures and domains. *Theoretical Computer Science*, 13(1):85–108, 1981.

[26] M. Nygaard. *Domain Theory for Concurrency.* PhD thesis, University of Aarhus, July 2003.

[27] M. Nygaard and G. Winskel. Domain theory for concurrency. *Theoretical Computer Science*, 316:153–190, 2004.

[28] G. D. Plotkin. A powerdomain construction. *Society for Industrial and Applied Mathematics Journal on Computing*, 5(3):452–487, 1976.

[29] J. R. Russell. Full abstraction for nondeterministic dataflow networks. In *FOCS '89, Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, pages 170–175. IEEE Computer Society Press, 1989.

[30] L. Saunders-Evans and G. Winskel. Event structure spans for non-deterministic dataflow. In *EXPRESS '06*, volume 175 of *ENTCS*, pages 109–129. Elsevier, 2006.

[31] D. S. Scott and C. Strachey. Towards a mathematical semantics for computer languages. In *Proceedings of the Symposium on Computers and Automata*, volume 21 of *Microwave Research Institute Symposium Series*, 1971.

[32] P. Selinger. Categorical structure of asynchrony. In *Proceedings of MFPS 15*, volume 20 of *ENTCS*, pages 158–181, 1999.

[33] R. J. van Glabbeek and G. D. Plotkin. Event structures for resolvable conflict. In *Proceedings of the 29th International Symposium on Mathematical Foundations of Computer Science*, volume 3153 of *LNCS*, pages 550–561. Springer, 2004.

[34] D. Varacca, H. Voelzer, and G. Winskel. Probabilistic event structures and domains. In *CONCUR '04, Proceedings of the 15th International Conference on Concurrency Theory*, volume 3170 of *LNCS*, pages 481–496. Springer, 2004.

[35] G. Winskel. *Events in computation.* PhD thesis, University of Edinburgh, 1980.

[36] G. Winskel. Event structure semantics for CCS and related languages. DAIMI Research Report, University of Aarhus, 1983. Full version of ICALP'82 article.

[37] G. Winskel. An introduction to event structures. In *Proceedings of the REX School/Workshop on Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, volume 354 of *LNCS*, pages 364–397. Springer, 1988.

[38] G. Winskel. Event structures as presheaves – two representation theorems. In *CONCUR '99, Proceedings of the 10th International Conference on Concurrency Theory*, volume 1664 of *LNCS*, pages 541–556. Springer, 1999.

[39] Glynn Winskel. Event structures with symmetry. In *Proceedings GDP Festschrift*, ENTCS. Elsevier, 2007.