EVENT STRUCTURES

by
Glynn Winskel
University of Cambridge,
Computer Laboratory,
Corn Exchange Street,
Cambridge CB2 3QG.

Abstract

Event structures are a model of computational processes. They represent a process as a set of event occurrences with relations to express how events causally depend on others. This paper introduces event structures, shows their relationship to Scott domains and Petri nets, and surveys their role in denotational semantics, both for modelling languages like CCS and CSP and languages with higher types.

Introduction.

Event structures are models of processes as events constrained by relations of consistency and enabling. Their study in denotational semantics first arose as a biproduct in the pioneering work of G.Kahn and G.Plotkin on some foundational questions in denotational semantics (See 1.5). The concrete data structures of Kahn and Plotkin were later realised to be closely related to confusion–free Petri nets (see part 3 and [NPW]) and this led to the more general definitions discussed here. Since then they have been developed as a model in their own right and for certain applications (*e.g.* see part 4) they are easier and less clumsy to use than Petri nets to which they are closely related however. These notes are intended to present the mathematical theory of event structures, show how they are related to Petri nets and Scott domains, and how they can be used to provide semantics to programming languages for parallel processes as well as languages with higher types.

A goal in working with event structures has been to develop a theory of concurrency which incorporates both the insights of C.A.Petri and D.S.Scott. To some extent this has been achieved. On the one hand, event structures consist of relations on events and bear a close relationship to Petri nets. On the other, the configurations or states of an event structure naturally reflect information about what events have occurred and determine a Scott domain of information. Because of this dual nature event structures stand as an intermediary between the theories Petri nets and denotational semantics, sharing ideas with both. As such they can serve a bridge between the two theories. For example the insight of Scott that computable functions induce continuous functions on domains appears as a finiteness axiom on event structures (1.4), which can be readily interpreted for Petri nets, while the restriction of confusion–freeness on a Petri net translates to concreteness on a domain naturally associated with it (1.5, 3.3). There remains the curious mismatch noted in [NPW]: a computation which is described by an event structure, or Petri net, gives rise to a whole domain whereas usually in denotational semantics a computation denotes a single element of a domain. This indicates, I believe, that we are still some way from the comprehensive theory of events in computation envisaged in [W].

The notes are organised in four parts. The first introduces event structures and their relations to families of configurations and certain kinds of domains which are viewed as different presentations of essentially the same idea. It develops the framework in which event structures can be defined recursively. Here the closeness of event structures to domains has another pay–off. It is easy to adapt ideas from denotational semantics to provide a smooth framework for recursion. In parts 2 and 4 this work is extended to particular applications. In part 2, event structures are used to

provide a non–interleaving model of languages like CCS and CSP. The approach is quite abstract and mathematical, using some category theory, but has the benefit of establishing once and for all, in a uniform way, a variety of semantics, interleaving and non–interleaving, and the relations between them. We can now move on, use the semantics, and try, for example, to advance our understanding of the relationship of models of "true concurrency" with operational semantics and the logic of concurrent programs. The guts of the work of this part appeared in [W1]. Part 3 shows how the same ideas can be carried through for Petri nets, giving rise to a more algebraic treatment of nets than usual, and gives a formal translation between nets and event structures. (Sections 3.1 to 3.3 can be read without any knowledge of the earlier sections.) In part 4 some work of G.Berry is presented in a new light. It is shown how event structures can be made into cartesian closed category and so be used to model programming languages with higher types. This part is meant as a preparation and indicator to further work, both Berry and Curien's work (see [C] especially) and some recent work of Girard on a model for his System F, the polymorphic λ-calculus (see [G] and [CGW]). The work after part 1 will use some basic ideas from category theory. Our main reference is [Mac].

1. EVENT STRUCTURES, CONFIGURATIONS AND DOMAINS.

This part gives the definition of event structures, focusses on special forms, and shows how particular kinds of Scott domains of information are formed by their configurations (or states). Scott's thesis is related to the axiom of finite causes and the machinery is established for defining event structures recursively. In addition, the relationship between event structures and concrete domains is exhibited with a brief indication of the relevance of concrete domains to denotational semantics.

1.1. Event structures.

Picture a process as performing events as time goes on. What we choose to regard as events of the process will depend on the level of abstraction at which we view the process. For the moment let us not worry about what kinds of events they are. Suppose that this is settled on and we have decided that the events of interest to us come from a set E of events or more strictly event occurrences. Generally for various reasons some events exclude some others from occurring so not all subsets of events can occur together in a history of the process. For example one event may exclude another for physical reasons, you just cannot have two values at the same time at some place or they may be in conflict because they compete for the same resource. Whatever the reason we can only expect certain subsets of events to be able to occur in the same history. We can express this as as a consistency predicate $\mathrm{Con} \subseteq \mathrm{Fin}(E)$ on the finite subsets of E. And of course if a set X of events can occur together in the same history then so can a subset $Y \subseteq X$ so we can put $Y \in \mathrm{Con}$ too. There is a additional constraint on the occurrence of events. Generally an event can occur only after certain other events have already occurred, and naturally we can assume they are consistent. We capture this by use of an enabling relation $\vdash \subseteq \mathrm{Con} \times E$ where intuitively an event e can only occur after a set X, with $X \vdash e$, has occurred previously.

1.1.1 Definition. An *event structure* is a triple $(E, \mathrm{Con}, \vdash)$ where

(i) E is a set of *events*,

(ii) Con is nonempty subset of $\mathrm{Fin}E$, the finite subsets of E, called the *consistency* predicate which satisfies

$$X \in \mathrm{Con} \ \& \ Y \subseteq X \Rightarrow Y \in \mathrm{Con}, \text{ and}$$

(iii) $\vdash \subseteq \mathrm{Con} \times E$ is the *enabling* relation which satisfies

$$X \vdash e \ \& \ X \subseteq Y \in \mathrm{Con} \Rightarrow Y \vdash e.$$

Our intuitive understanding of the consistency predicate and the enabling relation are expressed in the notion of configuration (=state) we adopt for event structures. A configuration is a set of events which have occurred by some stage in a process. According to our understanding of the consistency predicate a configuration should be consistent in the sense that any finite subset is in the consistency predicate. And according to our understanding of the enabling relation every event in a configuration should have been enabled by events which have occurred previously. However the chain of enablings should not be infinite but eventually end with events which are enabled by the null set, and so need no events to occur previously.

1.1.2 Definition. Let $\mathrm{E} = (E, \mathrm{Con}, \vdash)$ be an event structure. Define a *configuration* of E to be a subset of events $x \subseteq E$ which is

(i) *consistent:* $\forall X \subseteq_{fin} x.\ X \in \mathrm{Con}$,

(ii) *secured:* $\forall e \in x \exists e_0, \cdots, e_n \in x.\ e_n = e \ \& \ \forall i \leq n.\ \{e_0, \cdots, e_{i-1}\} \vdash e_i$.

The set of all configurations of an event structure is written as $\mathcal{F}(\mathrm{E})$.

It is helpful to unwrap condition (ii) a little. It says an event e is secured in a set x iff there is a sequence of events $e_0, \cdots, e_n = e$ in x such that

$$\emptyset \vdash e_0,\ \{e_0\} \vdash e_1, \cdots, \{e_0, \cdots, e_{i-1}\} \vdash e_i, \cdots, \{e_0, \cdots, e_{n-1}\} \vdash e_n.$$

We call such a sequence $e_0, e_1, \ldots, e_n = e$ a *securing* for e in x. The following proposition expresses when an event can be added to a configuration to obtain another configuration. We use $X \subseteq_{fin} Y$ to mean X is a finite subset of Y.

1.1.3 Proposition. *Let $E = (E, Con, \vdash)$ be an event structure. Suppose $x \in \mathcal{F}(E)$ and $e \in E$. Then $x \cup \{e\} \in \mathcal{F}(E)$ iff*

(i) $\forall X \subseteq_{fin} x.\ X \cup \{e\} \in Con$ *and*

(ii) $\exists X \subseteq_{fin} x.\ X \vdash e$.

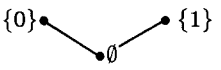
Proof. Clearly (i) and (ii) are necessary for $x \cup \{e\}$ to be a configuration. Conversely, assume (i) and (ii) hold. Then by (i), $x \cup \{e\}$ is consistent. By (ii) there is some $X \subseteq_{fin} x$ such that $X \vdash e$. Write X as $\{e_0, \ldots, e_{n-1}\}$. Each e_i has a securing s_i in x. Form the chain $s_0 \frown s_1 \frown \cdots \frown s_{n-1} e$ by concatenation. Then this chain is a securing for e in $x \cup \{e\}$. ∎

1.1.4 Example. Event structures may be infinite. For example, define Ω to be the event structure with events the nonnegative integers ω, with any finite subset consistent and enabling relation

$$X \vdash n \Leftrightarrow \{n' \mid n' < n\} \subseteq X.$$

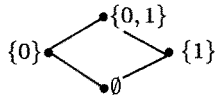
Then Ω represents a process, like a "ticking clock", which can perform the events $0, 1, \cdots, n, \cdots$ in sequence.

1.1.5 Example. Event structures can exhibit nondeterminism, or conflict. Consider the event structure with two events 0, 1 in which $\emptyset \vdash 0$ and $\emptyset \vdash 1$, $\{0\}, \{1\} \in \mathrm{Con}$ and yet $\{0, 1\} \notin \mathrm{Con}$. Its configurations have the form:



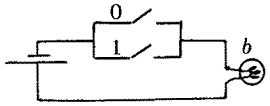
Nondeterminism appears as "branching" in the partial order of configurations ordered by inclusion.

1.1.6 Example. Event structures can exhibit parallelism, or concurrency. The event structure with two events $0, 1$ in which $\emptyset \vdash 0$ and $\emptyset \vdash 1$ and this time $\{0,1\} \in Con$, has configurations of the form:

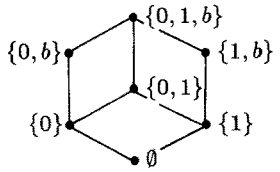


Concurrency of events appears as a "little square" in the partial order of configurations.

1.1.7 Example. A parallel switch:



An event may be enabled in more than one way even in a single configuration. Assume initially both switches are open. Closing either one enables the event of the bulb lighting up. The configurations have the form:



Thus each event structure determines a family of subsets of events, the configurations of the event structure. Such families have a simple characterisation.

1.1.8 Definition.
Let $(P, \sqsubseteq)$ be a partial order.
Say a subset X of P is *compatible*, and write $X\uparrow$, iff

$$\exists p \in P \forall x \in X.\ x \sqsubseteq p.$$

We can use this notion in the particular case where P is a family of sets ordered by inclusion. In the special case where X is a set of two elements $\{x, y\}$ we write $x \uparrow y$ for $X\uparrow$.

For families of configurations we shall use a more delicate notion of compatibility. Say a subset X is *finitely compatible*, and write $X\uparrow^{fin}$, iff

$$\forall X_0 \subseteq_{fin} X.\ X_0\uparrow,$$

i.e. when every finite subset is compatible.

1.1.9 Theorem. *Let $E = (E, Con, \vdash)$ be an event structure. Its configurations $F = \mathcal{F}(E)$ form a set of subsets of E which satisfy*

(i) finite-completeness:

$$A \subseteq F \ \ \& \ \ A\uparrow^{fin} \Rightarrow \bigcup A \in F,$$

(ii) finiteness:

$$\forall x \in F \forall e \in x \exists z \in F.\ (z \text{ is finite} \ \ \& \ \ e \in z \ \ \& \ \ z \subseteq x),$$

(iii) coincidence-freeness:

$$\forall x \in F \forall e, e' \in x.\ e \neq e' \ \Rightarrow \ (\exists y \in F.\ y \subseteq x \ \ \& \ \ (e \in y \Leftrightarrow e' \notin y)).$$

Proof. The proof is a routine exercise using the definition of configuration of an event structure. ∎

1.1.10 Definition. Let F be a set of subsets. Say F is a *family of configurations* when it satisfies the axioms of finite–completeness, finiteness and coincidence–freeness above. Say F is a family of configurations of E when $E = \bigcup \mathrm{F}$.

1.1.11 Lemma. *Let F be a family of configurations. For all $x, y \in F$*

$$x \subset y \Rightarrow \exists e \in y \setminus x.\ x \cup \{e\} \in F.$$

Proof.

Suppose $x \subset y$ for $x, y \in \mathrm{F}$. Then there is some event $e \in y \setminus x$. By finiteness $e \in z \subseteq_{fin} y$ for some finite config z. By finite–completeness $x \cup z \in \mathrm{F}$. Of course, $x \subset x \cup z \subset y$. Thus it is sufficient to prove the lemma in the case where the set $y \setminus x$ is finite. We do this by induction on the size $|y \setminus x|$ of the set difference, taking the statement of the lemma as the induction hypothesis.

If $|y \setminus x| = 1$ then obviously $y = x \cup \{e\}$ for the unique event e with $e \in y \setminus x$.

Suppose $|y \setminus x| > 1$ and assume the induction hypothesis for strictly smaller sizes. There are then two distinct events $e_0, e_1 \in y \setminus x$. By coincidence–freeness there is a configuration z containing one and not the other. (Without loss of generality assume $e_0 \in w$ and $e_1 \notin w$.) Hence $x \subset x \cup w \subset y$. Therefore by the induction hypothesis there is some $e \in (x \cup w) \setminus x$ for which $x \cup \{e\} \in \mathrm{F}$, and clearly $e \in y \setminus x$, as required. ∎

1.1.12 Definition. Let F be a family of configurations of a set E. Define a structure $\mathcal{E}(\mathrm{F}) = (E, \mathrm{Con}, \vdash)$ on E by taking

$X \in \mathrm{Con} \Leftrightarrow_{def} X$ is finite $\ \&\ \exists x \in \mathrm{F}.\ X \subseteq x$,

$X \vdash e \Leftrightarrow_{def} X \in \mathrm{Con}\ \&\ \exists x \in \mathrm{F}.\ e \in x\ \&\ x \subseteq X \cup \{e\}$.

1.1.13 Theorem. *If F is a family of configurations then $\mathcal{E}(F)$ is an event structure such that $\mathcal{F}\mathcal{E}(F) = F$.*

Proof. Let F be a family of configurations. It is easy to see that $\mathcal{E}(\mathrm{F})$ is an event structure.

Suppose $x \in \mathcal{F}\mathcal{E}(\mathrm{F})$. Then, by the definition of the enabling relation of $\mathcal{E}(\mathrm{F})$, for each $e \in x$ there is a configuration $x_e \in \mathrm{F}$ such that $e \in x_e \subseteq_{fin} x$. By the definition of the consistency predicate of $\mathcal{E}(\mathrm{F})$, the set $\{x_e \mid e \in x\}$ is a finitely compatible subset of F. Therefore

$$x = \bigcup \{x_e \mid e \in x\} \in \mathrm{F}.$$

Suppose $x \in \mathrm{F}$. We show $x \in \mathcal{F}\mathcal{E}(\mathrm{F})$. Certainly x is consistent. Suppose $e \in x$. By the finiteness property of F there is a finite configuration y for which $e \in y \subseteq_{fin} x$. Repeatedly applying lemma 1.1.11, starting with the interval $\emptyset \subset y$, we obtain a sequence $e_1, \ldots, e_i, \ldots, e_n$ such that

$$\{e_1\}, \cdots, \{e_1, \ldots, e_i\}, \cdots, \{e_1, \ldots, e_i, \ldots, e_n\} \in \mathrm{F}$$

with $\{e_1, \ldots, e_i, \ldots, e_n\} = y$. As e occurs in some stage of the sequence this provides us with a securing for e in x. Hence x is a configuration of $\mathcal{E}(\mathrm{F})$. ∎

Notice we do not have $\mathcal{E}\mathcal{F}(\mathrm{E})$ and E equal in general for event structures E.

1.1.14 Corollary. *Let F_0 and F_1 be families of configurations. If $\mathcal{E}(F_0) = \mathcal{E}(F_1)$ then $F_0 = F_1$.*

Proof. If $\mathcal{E}(\mathrm{F}_0) = \mathcal{E}(\mathrm{F}_1)$ then $\mathrm{F}_0 = \mathcal{FE}(\mathrm{F}_0) = \mathcal{FE}(\mathrm{F}_1) = \mathrm{F}_1$, by the theorem above. ∎

Of course the configurations of an event structure form a partial order when ordered by inclusion. It is sensible to think of the points of this partial order as elements of information expressing how far the process has progressed; the computation has progressed further when more events have occurred. The idea of information is familiar from Dana Scott's work and in fact the configurations of an event structure do form a domain when ordered by inclusion. Though note it is a rather special kind of domain. In particular it satisfies the finiteness axiom that an finite element dominates only a finite number of elements. This is because the concept of more information is tied very closely to the progress of the process over time. The associated domains are closely related to the concrete domains of Kahn and Plotkin (see [W, KP] and section 1.5) Recall:

1.1.15 Definition. Let $(D, \sqsubseteq)$ be a partial order.

Say D is *consistently complete* iff all finitely compatible subsets $X \subseteq D$ have least upper bounds $\bigsqcup X$.

Note a consistently complete partial order has a least element, *viz.* $\bot = \bigsqcup \emptyset$, though it may not have a greatest.

Say a subset S of D is directed iff all $S_0 \subseteq_{fin} S$ have upper bounds in S. (So S is finitely compatible and cannot be empty.) An element e of D is said to be *finite* iff for all directed sets S, if $e \sqsubseteq \bigsqcup S$ then $e \sqsubseteq s$ for some $s \in S$.

A consistently complete partial order is *algebraic* iff for every element d

$$d = \bigsqcup \{e \sqsubseteq d \mid e \text{ is finite}\}.$$

We call a consistently complete algebraic partial order a *Scott domain* (or simply a domain).

A *finitary* domain is one in which every finite element dominates only a finite number of elements, *i.e.* $\{d \mid d \sqsubseteq e\}$ is finite.

1.1.16 Theorem. *Let F be a family of configurations. The partial order $(F, \subseteq)$ is a finitary Scott domain with finite elements the finite configurations.*

Proof. As the family F is finitely-complete, the partial order $(\mathrm{F}, \subseteq)$ is consistently complete. Clearly every configuration which forms a finite set is a finite element. Let $x \in \mathrm{F}$. Each $e \in x$ is contained in some finite configuration $x_e \subseteq x$ by the finiteness axiom. Obviously $x = \bigcup \{x_e \mid e \in x\}$. Hence $(\mathrm{F}, \subseteq)$ is algebraic and so a Scott domain. ∎

The thesis [W] contains a characterisation of the domains that result from event structures in the case when the consistency relation is induced by a binary conflict relation between events (see later, 2.3). I am not sure of the characterisation of domains associated with the more general event structures presented here. However such representation results are particularly smooth for the slightly more restrictive class of stable event structures, introduced in the next section, which are suitable in most cases.

Thus when we picture a process as an event structure we can choose, if we wish, to regard it more abstractly as determining a family of configurations—when we abstract from the precise nature of the consistency and enabling relations—or more abstractly still as a domain of configurations—when we abstract from the precise nature of the names we use for events. Conversely we can regard families of configurations and domains of configurations as special kinds of event structures. As we shall see we can abstract in other ways too, and see these means of abstraction in a categorical light. (The trees which underly the interleaving models of CCS and CSP are a similar abstraction from the extra detail present in the non–interleaving model of event structures.)

1.2. Stable event structures.

Many people [Pe, He, La, Ma, NPW, W, Sh, MS, F, Pr] represent concurrent processes as partial orders of events where an event e_0 precedes an event e_1 if the occurrence of the event e_0 is necessary in order for e_1 to occur, in other words if the event e_1 causally depends on the event e_0. Often in these treatments all possible events of the process are put in the partial order whether or not they are in can occur in the same history; there is a global partial order of causal dependency. We shall treat models like these in the next section. It is useful to look at a more general class of structures for which there need not be one global partial order but where each configuration has its own local partial order of causal dependency.

We look for a special class of event structures for which there is a partial order of causal dependency on each configuration. This can not be done so obviously for all event structures. Consider the event structure of example 1.1.7, representing a parallel switch where the event b causally depends not on a unique set of events but rather on *either* the occurrence of 0 *or* on the occurrence of 1. It is incorrect to say b causally depends on both 0 and 1 because the occurrence of only one of them enables the occurrence of b. The difficulty arises because there is a configuration $\{0, 1, b\}$ in which there is an event b which is not enabled by a unique minimal set of event occurrences. We can rule out such possibilities by insisting event structures satisfy the following *stability* axiom.

1.2.1 Definition. Let $\mathrm{E} = (E, \mathrm{Con}, \vdash)$ be an event structure. Say E is *stable* if it satisfies the following axiom

$$X \vdash e \ \ \& \ \ Y \vdash e \ \ \& \ \ X \cup Y \cup \{e\} \in \mathrm{Con} \Rightarrow X \cap Y \vdash e.$$

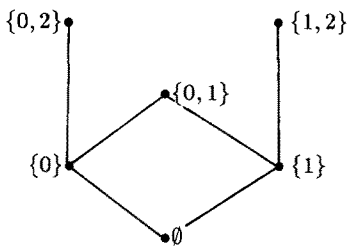
1.2.2 Example. Let E be the event structure with events $\{0, 1, 2\}$ with consistency predicate the least one such that

$$\{0,1\}, \{0,2\}, \{1,2\} \in \mathrm{Con},$$

so $\{0, 1, 2\} \notin \mathrm{Con}$, and enabling relation the least one such that

$$\emptyset \vdash 0, \ \emptyset \vdash 1, \ \{0\} \vdash 2, \ \{1\} \vdash 2.$$

Then E is a stable event structure and the configurations $\mathcal{F}(\mathrm{E})$ have the form



The stability axiom ensures that an event in a configuration is enabled in an essentially unique way. Assume e belongs to a configuration x of a stable event structure. Suppose $X \vdash e$ and $X \subseteq x$. Then $X \cup \{e\} \in \mathrm{Con}$—the enabling $X \vdash e$ is consistent. Take

$$X_0 = \bigcap \{Y \mid Y \subseteq X \ \ \& \ \ Y \vdash e\}.$$

Because X is finite this is an intersection of a finite number of sets and we see by the stability axiom that $X_0 \vdash e$. Moreover X_0 is the unique minimal subset of X which enables e. More formally, for any event structure, stable or otherwise, we can define the *minimal enabling* relation $\vdash_{min}$ by

$$X \vdash_{min} e \Leftrightarrow X \vdash e \ \ \& \ \ (\forall Y \subseteq X.\ Y \vdash e \Rightarrow Y = X).$$

Then for any event structure

$$Y \vdash e \Rightarrow \exists X \subseteq Y.\ X \vdash_{min} e.$$

But for stable event structures we have uniqueness too, at least for consistent enablings:

$$Y \vdash e \ \ \& \ \ Y \cup \{e\} \in \mathrm{Con} \Rightarrow \exists! X \subseteq Y.\ X \vdash_{min} e.$$

It follows that for stable event structures

$$X \vdash_{min} e \ \ \& \ \ Y \vdash_{min} e \ \ \& \ \ X \cup Y \cup e \in \mathrm{Con} \Rightarrow X = Y.$$

Consequently the families of configurations of stable event structures satisfy the following intersection property.

1.2.3 Theorem. *Let E be a stable event structure. Then its family of configurations $\mathcal{F}(E)$ satisfies*

$$\forall X \subseteq \mathcal{F}(E).\ X \neq \emptyset \ \ \& \ \ X\uparrow \Rightarrow \bigcap X \in \mathcal{F}(E).$$

Proof. Suppose X is a nonempty compatible subset of configurations. Then $\forall x \in X.\ x \sqsubseteq z$ for some configuration z. Clearly $\bigcap X$ is consistent. Suppose $e \in \bigcap X$. Then $e \in z$ so there is some securing $e_0, e_1, \ldots, e_n = e$ for e in z. By stability, for any $e \in \bigcap X$ if $Y \vdash_{min} e$ and $Y \subseteq z$ then $Y \subseteq \bigcap X$. Therefore the securing for e in z becomes a securing for e in $\bigcap X$ by omitting all members of the sequence not in $\bigcap X$. Thus $\bigcap X$ is secured, and so a configuration. ∎

1.2.4 Definition. Say a family of configurations F is *stable* when it satisfies the following axiom (in addition to those in 1.1.9)

(stability) $$\forall X \subseteq \mathrm{F}.\ X \neq \emptyset \ \ \& \ \ X\uparrow \Rightarrow \bigcap X \in \mathrm{F}.$$

Thus the configurations of a stable event structure form a stable family. For a stable family there is a partial order of causal dependency on each configuration of events.

1.2.5 Definition. Let F be a stable family of configurations. Let x be a configuration. For $e, e' \in x$ define

$$e' \leq_x e \Leftrightarrow \forall y \in \mathrm{F}.\ e' \in y \ \ \& \ \ y \subseteq x \Rightarrow e \in y.$$

When $e \in x$ define

$$\lceil e \rceil_x = \bigcap \{y \in \mathrm{F} \mid e \in y \ \ \& \ \ y \subseteq\!\!\!\cdot\ x\}.$$

We say a set y is $\leq_x$-left closed when it satisfies

$$e' \leq_x e \ \ \& \ \ e \in y \Rightarrow e' \in y.$$

As usual, we write $e' <_x e$ for $e \leq_x e' \ \ \& \ \ e \neq e'$.

1.2.6 Proposition. *Let x be a configuration of a stable family F. Then $\leq_x$ is a partial order and $\lceil e \rceil_x$ is a configuration such that*

$$\lceil e \rceil_x = \{e' \in x \mid e' \leq_x e\}.$$

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

Proof. Let x be a configuration of a stable family F.

The relation $\leq_x$ is clearly a preorder. Further it is a partial order by coincidence–freeness.

The fact that the set $\lceil e \rceil_x$ is a configuration follows directly from its definition as the family is stable. Suppose $e' \leq_x e$. Then $e \in \lceil e \rceil_x \subseteq x$ so $e' \in \lceil e \rceil_x$. Thus $\lceil e \rceil_x$ is $\leq_x$–left closed. Suppose $e' \in \lceil e \rceil_x$. Then from the definition of $\lceil e \rceil_x$ we see directly that $e' \leq_x e$. Hence

$$\lceil e \rceil_x = \{e' \mid e' \leq_x e\}.$$

Suppose $y \in \mathrm{F}$ and $y \subseteq x$. Assume $e' \leq_x e$ and $e \in y$. Then by the definition of $\leq_x$ we see $e' \in y$. Thus y is left closed. The converse also holds. Suppose y is left closed and $y \subseteq x$. Then clearly

$$y = \bigcup \{\lceil e \rceil_x \mid e \in y\},$$

and $\{\lceil e \rceil_x \mid e \in y\} \uparrow$, each element being a configuration included in x. Therefore by finite–completeness of the family we see $y \in \mathrm{F}$. ∎

Let x be a configuration of a stable family. Intuitively an event e in x can only occur once all its predecessors $\{e' \in x \mid e' <_x e\}$ have occurred.

1.2.7 Example. Refer to example 1.2.2. Let $x = \{0, 2\}$ and $y = \{1, 2\}$ be particular configurations. Then $0 \leq_x 2$ and $1 \leq_y 2$ but $0 \not\leq_y 2$ and $1 \not\leq_x 2$. The orderings $\leq_x$ and $\leq_y$ cannot be the restrictions of a "global" partial order on events.

1.2.8 Theorem.
Let E be a stable event structure. Then its family of configurations $\mathcal{F}E$ is stable.
Let F be a stable family of configurations. Then $\mathcal{E}(F)$ is a stable event structure.

Proof. The first part is simply a restatement of 1.2.3. We show the second part. By 1.1.13 we already know $\mathcal{E}(\mathrm{F})$ is an event structure. Suppose $X \vdash e$ and $Y \vdash e$ with $X \cup Y \cup \{e\} \in \mathrm{Con}$ in $\mathcal{E}(\mathrm{F})$. Then

$$e \in x \ \ \& \ \ x \subseteq X \cup \{e\} \text{ and}$$
$$e \in y \ \ \& \ \ y \subseteq Y \cup \{e\}$$

for $x, y \in \mathrm{F}$. Thus $x \cup y \subseteq X \cup Y \cup \{e\}$, a consistent set, so $x \uparrow y$. Therefore $x \cap y \in \mathrm{F}$ and clearly $e \in x \cap y \ \ \& \ \ x \cap y \subseteq (X \cap Y) \cup \{e\}$. Thus $X \cap Y \vdash e$, as required to show $\mathcal{E}(\mathrm{F})$ is a stable event structure. ∎

1.3. Prime algebraic domains and partial orders of events.

We consider the form of domain associated with stable event structures. Firstly we define the relevant properties.

1.3.1 Definition.
Let $\mathrm{D} = (D, \sqsubseteq)$ be a consistently complete partial order.
Say D is *distributive* iff it satisfies

$$y \uparrow z \Rightarrow x \sqcap (y \sqcup z) = (x \sqcap y) \sqcup (x \sqcap z).$$

Say D is *infinitely distributive* iff it satisfies the following two laws:

$$(\bigsqcup X) \sqcap y = \bigsqcup \{x \sqcap y \mid x \in X\},$$

where X is a compatible subset of D and $y \in D$, and

$$(\prod X) \sqcup y = \prod\{x \sqcup y \mid x \in X\}$$

where $\emptyset \neq X \subseteq D$ and $y \in D$.

A *complete prime* of D is an element $p \in D$ such that

$$p \sqsubseteq \bigsqcup X \Rightarrow \exists x \in X.\ p \sqsubseteq x$$

for any compatible set X.

D is a *prime algebraic domain* iff

$$x = \bigsqcup\{p \sqsubseteq x \mid p \text{ is a complete prime}\},$$

for all $x \in D$.

Thus a prime algebraic domain is a Scott domain of information which possesses a special kind of sub–basis.

Prime algebraic domains have a characterisation as familiar structures, a result which follows directly from [W2].

1.3.2 Theorem. *Let D be a consistently complete partial order.*
D is a prime algebraic domain iff it is infinitely distributive and algebraic.
If D is finitary then it is prime algebraic iff it is distributive.

Proof. The proofs are quite lengthy and so are omitted. They can be found for lattices in [W2] from which the results follow for consistently complete orders. ∎

Families of configurations of stable event structures are prime algebraic. The axiom of stability on event structures has as its counterpart the axiom of distributivity on domains.

1.3.3 Theorem. *Let F be a stable family of configurations. The partial order $(F, \subseteq)$ is a finitary and prime algebraic domain; the complete primes are the set $\{\lceil e \rceil_x \mid e \in x \ \&\ x \in \mathcal{F}(E)\}$.*

Proof.

By 1.1.16 we know $(\mathrm{F}, \subseteq)$ forms a finitary Scott domain.

Suppose $e \in x \in \mathrm{F}$. Assume $\lceil e \rceil_x \subseteq \bigsqcup W$. Then $e \in w$ for some $w \in W$. By stability and the fact that $\lceil e \rceil_x \uparrow w$ we see $\lceil e \rceil_x \subseteq w$. Hence $\lceil e \rceil_x$ is a complete prime.

Let $x \in \mathrm{F}$. Clearly $\{\lceil e \rceil_x \mid e \in x\} \uparrow^{fin}$. Thus $x = \bigcup\{\lceil e \rceil_x \mid e \in x\}$. Therefore $(\mathrm{F}, \subseteq)$ is prime algebraic. ∎

Thus stability of event structures appears as distributivity of the domains of configurations. The fact that events must be secured in configurations, expressing the intuition that an event's occurrence can only depend on a finite number of previous occurrences, reappears as the fact that domains of configurations are finitary.

Conversely, given a finitary prime algebraic domain we can easily generate a stable event structure which has an isomorphic domain of configurations. There is a natural choice of events associated with a finitary prime algebraic domain, *viz.* the complete primes. They inherit the ordering from D and this partial order can be viewed as a causal dependency relation. Unlike the local causal

dependency relations of the previous section which were defined with respect to particular configurations this is one global relation. There is an obvious consistency relation on complete primes; take a finite subset of primes to be consistent iff they are compatible. The family of configurations is easily generated from these relations. Structures $(P, \mathrm{Con}, \leq)$ can be thought of as another kind of event structure in which the enabling relation can be expressed in an especially simple form, as a global partial order of causal dependency.

1.3.4 Definition. Define a *prime event structure* to be a structure $\mathrm{E} = (E, \mathrm{Con}, \leq)$ consisting of a set E, of *events* which are partially ordered by $\leq$, the *causal dependency relation*, and a predicate $\mathrm{Con} \subseteq \mathrm{Fin}E$, the *consistency relation*, which satisfy

$$\begin{aligned}
&\{e' \mid e' \leq e\} \text{ is finite,}\\
&\{e\} \in \mathrm{Con},\\
&Y \subseteq X \in \mathrm{Con} \Rightarrow Y \in \mathrm{Con},\\
&X \in \mathrm{Con} \ \ \& \ \ \exists e' \in X.\ e \leq e' \Rightarrow X \cup \{e\} \in \mathrm{Con}
\end{aligned}$$

for all $e \in E$, and finite subsets X, Y of E.

Define its *consistent left-closed subsets*, $\mathcal{L}(\mathrm{E})$, to consist of those subsets $x \subseteq E$ which are

consistent: $\forall X \subseteq_{fin} x.\ X \in \mathrm{Con}$ and

left-closed: $\forall e, e'.\ e' \leq e \in x \Rightarrow e' \in x$.

In particular, define $\lceil e \rceil = \{e' \in E \mid e' \leq e\}$.

1.3.5 Theorem.

Let E be a prime event structure. Then $\mathcal{L}(E)$ is a stable family of configurations. The domain $(\mathcal{L}(E), \subseteq)$ has complete primes those elements of the form $\lceil e \rceil$ for $e \in E$.

Proof. Routine. ∎

Conversely, as we have indicated, any prime algebraic domain is associated with a prime event structure in which the events are its complete primes.

1.3.6 Definition. Let D be a finitary prime algebraic domain. Define $\mathcal{P}r(\mathrm{D}) = (P, \mathrm{Con}, \leq)$, where P consists of the complete primes of D,

$$p \leq p' \Leftrightarrow p \sqsubseteq p',$$

for $p, p' \in P$, and

$$X \in \mathrm{Con} \Leftrightarrow X\uparrow$$

for a finite subset X of P.

1.3.7 Theorem. *Let D be a finitary prime algebraic domain. Then $\mathcal{P}r(D)$ is a prime event structure, with $\phi : D \cong (\mathcal{L}\mathcal{P}r(D), \subseteq)$ giving an isomorphism of partial orders where $\phi(d) = \{p \sqsubseteq d \mid p \text{ is a complete prime}\}$ with inverse $\theta : \mathcal{L}\mathcal{P}r(D) \to D$ given by $\theta(x) = \bigsqcup x$.*

Proof.

It is easy to see that $\mathcal{P}r(\mathrm{D}) = (P, \mathrm{Con}, \leq)$ as defined is a prime event structure.

Obviously the maps θ and ϕ are monotonic *i.e.* order preserving. We show they are mutual inverses and so give the required isomorphism.

It is easy to see that the maps ϕ and θ are well-defined.

Firstly we show $\theta\phi = 1$. Thus we require $d = \bigsqcup\{p \in P \mid p \sqsubseteq d\}$ for all $d \in D$. But this is just the condition of prime algebraicity.

Now we show $\phi\theta = 1$. Let $x \in \mathcal{LPr}(\mathrm{D})$. We require $x = \phi\theta(x)$ *i.e.* $x = \{p \in P \mid p \sqsubseteq \bigsqcup x\}$. Clearly $x \subseteq \{p \in P \mid p \sqsubseteq \bigsqcup x\}$. Conversely if $p \sqsubseteq \bigsqcup x$, where p is a complete prime, then certainly $p \sqsubseteq q$ for some $q \in x$. However x is left–closed so $p \in x$, showing the converse inclusion.

Thus we have established the required isomorphism. ∎

Thus finitary prime algebraic domains and prime event structures are equivalent; one form of structure can be used to represent the other. Prime event structures are very simple and determine the same domains of configuration as the stable event structures so why do we not work solely with them? The reason is that prime event structures do not always combine very easily. Constructions on stable event structures are generally easy whereas it can often be quite awkward and clumsy to make the constructions yield prime event structures directly. For example the product (see section 2.3) and function space (see section 4.2) of two prime event structures are complicated when defined directly. By introducing the more general class of stable event structures we get the best of both worlds; constructions are easy and we can always obtain prime event structures with isomorphic domains of configurations by theorems 1.3.5, 1.3.7. We should remark that finitary prime algebraic domains have appeared in the context of Berry's work; in [B] he considers *dI-domains* which are finitary distributive domains, which by the results above are exactly the finitary prime algebraic domains.

As example 1.2.7 shows the local partial orders of causal dependency are not necessarily part of a global partial order on events. The above theorems show that at the cost of renaming events they can made to be so. Suppose $\mathrm{E} = (E, \mathrm{Con}, \vdash)$ is a stable event structure. Instead of taking the events as E we might change our view and regard the events as being $P = \{\lceil e \rceil_x \mid e \in x \in \mathcal{F}(\mathrm{E})\}$, so a new event is a complete prime which includes the information about how it occurs. What causal dependency relation should be put on the events P? An event p can only occur once all events p' strictly included in p have occurred. The global causal dependency relation $\leq$ on P given by

$$p' \leq p \Leftrightarrow p' \subseteq p.$$

And when can a finite set of events $X \subseteq_{fin} E$ occur together in a configuration? When they are compatible as configurations of E. This is the consistency predicate on events P:

$$X \in \mathrm{Con}_P \Leftrightarrow X \subseteq_{fin} P \ \ \& \ \ X\uparrow .$$

In this way, by renaming events, a stable event structure E determines a prime event structure $(P, \mathrm{Con}_P, \leq)$. Of course, the configurations of $(P, \mathrm{Con}_P, \leq)$ are not the same as the configurations of the original event structure—the events are different. Still, the two domains of configurations are isomorphic as partial orders. This just expresses the fact that the domain of configurations of a stable event structure is *prime algebraic*.

In [NPW] and [W] it is pointed out that events also manifest themselves in a domain as prime intervals. We say d is *covered* by d' in a domain, written $d \prec d'$ iff

$$d \sqsubseteq d' \ \ \& \ \ d \neq d' \ \ \& \ \ (\forall z.\ d \sqsubseteq z \sqsubseteq d' \Rightarrow d = z \ \text{ or } \ z = d).$$

The relation $\prec$ is called the *covering* relation. A *prime interval* is a pair $[d, d']$ such that $d \prec d'$. In a domain of configurations a prime interval is associated with the occurrence of an event at some configuration; in a domain of configurations $(\mathrm{F}, \subseteq)$, the relation $x \prec x'$ holds iff there is an event e such that $e \notin x$ and $x' = x \cup \{e\}$ with $x, x' \in \mathrm{F}$. Define

$$[c, c'] \leq [d, d'] \Leftrightarrow c = c' \sqcap d.$$

Form the equivalence relation $\sim$ as the symmetric, transitive closure of $\leq$, and write $[d,d']_\sim$ for the equivalence class of $[d,d']$ with respect to $\sim$. In a domain of configurations, $[c,c'] \sim [d,d']$ implies $c' \setminus c = d' \setminus d = \{e\}$ for the same event e. So $\sim$–classes are associated with unique events. For domains represented as families of configurations of complete primes this association is a 1–1 correspondence.

1.3.8 Proposition. *Let D be a finitary prime algebraic domain. Let $\phi : D \cong \mathcal{LP}r(D)$ be the isomorphism $d \mapsto \{p \sqsubseteq d \mid p \text{ is a complete prime}\}$. Define the following map from $\sim$–classes to complete primes:*

$$[d,d']_\sim \mapsto p$$

where p is the unique member of $\phi(d') \setminus \phi(d)$. This map is a 1–1 correspondence with inverse

$$p \mapsto [d,d']_\sim$$

where $d = \bigsqcup\{c \mid c \sqsubseteq p \ \ \& \ \ c \neq p\}$ and $d' = p$.

Proof. Routine—or see [NPW]. ∎

Later in some proofs we shall make use of the fact that if d is a finite element of a finitary prime algebraic domain D then there is a *covering chain*

$$\bot = d_0 \prec d_1 \prec \cdots \prec d_n = d$$

in D up to d. This is obvious because we can represent any such domain as the left closed consistent subsets of some prime event structure.

We turn now to one special kind of prime algebraic domain. Trees form a basic model of computation. Often branching represents nondeterminism as for example in Milner's synchronisation trees. We show how such trees can be taken to be particular kinds of prime algebraic domains and hence can be identified with certain kinds of event structure.

1.3.9 Definition. A *tree* is a prime algebraic domain which satisfies

$$x \uparrow y \Rightarrow (x \sqsubseteq y \ \text{ or } \ y \sqsubseteq x).$$

Thus for our purposes a tree is a special kind of domain whose order structure is that of a tree in the conventional sense but with limit points at the end of every infinite branch. Of course such trees are in 1–1 correspondence with certain forms of prime event structures and a tree T, as we have defined it, can be identified with its image $\mathcal{P}r(T)$ as a prime event structure. Its events are complete primes which are in 1–1 correspondence with prime intervals which are the arcs of the tree.

A finitary prime algebraic domain determines a tree in a natural way, a construction which will be important later in part 2.

1.3.10 Definition. Let D $= (D, \sqsubseteq)$ be a finitary prime algebraic domain. Define a *covering sequence* of D to be a sequence $\langle d_0, d_1, \ldots, d_{n-1}, \ldots\rangle$, which may be empty, finite or infinite, in which

$$\bot = d_0 \prec d_1 \prec \cdots \prec d_{n-1} \prec \cdots.$$

Define $\mathcal{T}$(D) to consist of all the covering sequences in D ordered by extension.

1.3.11 Proposition. *Let D be a finitary prime algebraic domain. Then $\mathcal{T}(D)$ is a tree.*

Proof. Clear. ∎

The translation from event structures to domains has perhaps seemed rather formal. However as was argued in [NPW, W] it does provide a bridge between concepts expressed in terms of Scott's idea of information and the ideas of Petri and others. And of course as we pointed out domains of configurations can be associated with certain kinds of event structures in a natural way.

1.4. Scott's thesis and the axiom of finite causes.

Dana Scott proposed the thesis that computable functions are continuous. Here it is understood that datatypes are associated with domains of information and that computable functions between datatypes are associated with functions between their domains of information. Recall a function $f : D \to E$ from one cpo D to another E is *continuous* iff it preserves least upper bounds of directed sets *i.e.* for all directed sets S

$$\bigsqcup fS = f(\bigsqcup S).$$

Note a continuous function is *monotonic*, *i.e.*

$$\forall x, y \in D.\ x \sqsubseteq y \Rightarrow f(x) \sqsubseteq f(y),$$

In particular, a continuous function should preserve least upper bounds of ω–chains, *i.e.* for all chains $x_0 \sqsubseteq x_1 \sqsubseteq \cdots \sqsubseteq x_n \sqsubseteq \cdots$ in D we have

$$\bigsqcup_{n\in\omega} f(x_n) = f(\bigsqcup_{n\in\omega} x_n).$$

Intuitively the ultimate output value should be no more than the limit of the values determined at finite stages in delivering the input, so we can approximate the ultimate output value arbitrarily closely by the output values at finite stages. Scott's thesis has an intuitive justification (see *e.g.* [St]), and plays a key part in the mathematical basis of denotational semantics. We show how Scott's thesis implies the thesis that for a computable process the occurrence of an event depends on the previous occurrence of a finite number of events.

We need first to motivate some definitions. For simplicity we assume a process is modelled by a partial order on events, $\mathrm{E} = (E, \leq)$ say, and show how the process will obey Scott's thesis iff it satisfies the *axiom of finite causes*:

$$\forall e \in E.\ \{e' \in E \mid e' \leq e\} \text{ is finite.}$$

Of course we need to make clear what we mean by "obey Scott's thesis". This hinges on associating datatypes and continuous functions with E.

We can choose to imagine some of the events of E as being events of input E_0 from some datatype, some as internal events, and others as events of output E_1 to some datatype. The datatypes may have their own causal dependencies, which contribute to the dependency of the full process, so the input datatype can carry an partial order $\mathrm{E}_0 = (E_0, \leq_0)$ and the output datatype a partial order $\mathrm{E}_1 = (E_1, \leq_1)$. The orderings of the datatypes should be sub–partial orders of that of the process, *i.e.*

$$\mathrm{E}_0 \subseteq \mathrm{E} \ \ \& \ \ \mathrm{E}_1 \subseteq \mathrm{E},$$

meaning $\leq_0 \subseteq \leq$ and $\leq_1 \subseteq \leq$. There are natural domains of information associated with the two datatypes, *viz.* their domains of left–closed sets of events. The process induces a function between the domains. Define

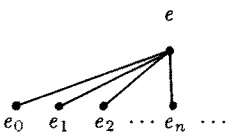
$$f_{E_0,E_1} : \mathcal{L}(\mathrm{E}_0) \to \mathcal{L}(\mathrm{E}_1) \text{ to map } x \mapsto \{e \in E_1 \mid \lceil e \rceil \cap E_0 \subseteq x\}.$$

The idea is that an event of E occurs once the necessary input events have occurred. It is clear that:

1.4.1 Lemma. *The function f_{E_0,E_1} is monotonic.*

Proof. Obvious. ∎

However for partial orders in general the function may not be continuous. Consider, for example, the partial order



with $E_0 = \{e_n \mid n \in \omega\}$ and $E_1 = \{e\}$ ordered by the identity relation. Then taking S to be the directed set consisting of all finite subsets of E_0 we see (as in the proof of the theorem below) that the least upper bound of S is not preserved by f_{E_0,E_1}. If E is to represent a computable process, according to Scott's thesis, f_{E_0,E_1} should be continuous. Furthermore it should be for any choice of n events for the input and output datatypes. We say E *obeys Scott's thesis* iff

$$\forall \mathrm{E}_0, \mathrm{E}_1.\ (\mathrm{E}_0 \subseteq \mathrm{E}\ \ \&\ \ \mathrm{E}_1 \subseteq \mathrm{E} \Rightarrow f_{E_0,E_1} \text{ is continuous}).$$

Now by an elementary argument we can show those partial orders of causal dependency E which obey Scott's thesis are precisely those which satisfy the axiom of finite causes.

1.4.2 Theorem. *The partial order E obeys Scott's thesis iff*

$$\forall e \in E.\ \{e' \in E \mid e' \leq e\} \text{ is finite.}$$

Proof.

"⇒" Suppose E obeys Scott's thesis. Suppose for some e in E we had $\lceil e \rceil$ infinite. Take

$$E_0 = \{e' \in E \mid e' < e\} \text{ and } E_1 = \{e\},$$

with both ordered by the identity relation. Define S to consist of all finite subsets of E_0. Then S is a directed subset of $\mathcal{L}(\mathrm{E}_0)$. Moreover no element of S is E_0 as E_0 is infinite. However now $f_{E_0,E_1}(\bigcup S) = \{e\}$ while $\bigcup f_{E_0,E_1} S = \emptyset$. Thus f_{E_0,E_1} is not continuous which contradicts the assumption that E obeys Scott's thesis. Thus $\lceil e \rceil$ is finite for all $e \in E$.

"⇐" Suppose $\lceil e \rceil$ is finite for all e in E. Assume $E_0 \subseteq E$ and $E_1 \subseteq E$. Let S be a directed subset of $\mathcal{L}(\mathrm{E}_0)$. Abbreviate f_{E_0,E_1} to f. As f is always monotonic we have $\bigcup fS \subseteq f(\bigcup S)$. Suppose $e \in f(\bigcup S)$. Then $\lceil e \rceil \cap E_0 \subseteq \bigcup S$. As $\lceil e \rceil$ is finite so is $\lceil e \rceil \cap E_0$. Thus because S is directed $\lceil e \rceil \cap E_0 \subseteq s$ for some $s \in S$. Then $e \in f(s)$. This shows $f(\bigcup S) \subseteq \bigcup fS$ so $f(\bigcup S) = \bigcup fS$. Therefore f is continuous. Hence $(E, \leq)$ obeys Scott's thesis, as required. ∎

1.5. Concrete domains.

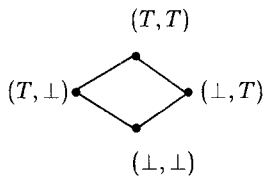
Event structures first arose in denotational semantics through the work of Kahn and Plotkin on *concrete domains* [KP]. They were interested in extending the definitions of sequential functions used by Milner and Vuillemin. It had become clear that often there was a mismatch between denotational semantics and operational semantics because the denotational semantics failed to take

account adequately of the sequential nature of the evaluation performed by machines. The problem was realised in its most acute form in [P], where the failure of full–abstraction for the denotational semantics of languages with higher type was traced to an inadequate treatment of sequential functions. For much more on these notions of sequentiality and full–abstraction, their importance, and work which stemmed from them see P.L.Curien's book [C]. (For a little more see section 4.1.)

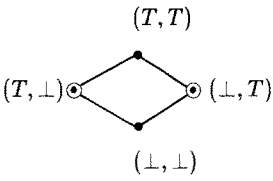
Let O be the simple domain consisting of two points $\bot \sqsubseteq T$. Then the product O $\times$ O, as a domain, is got by taking all ordered pairs $(x, y) \in \{\bot, T\}^2$ ordered coordinatewise:

$$(x, y) \sqsubseteq (x', y') \Leftrightarrow x \sqsubseteq x' \ \ \& \ \ y \sqsubseteq y'.$$

This yields a domain which may be pictured thus:



Consider the least monotonic function giving $(T, \bot) \mapsto T$ and $(\bot, T) \mapsto T$ which can be drawn as



encircling the minimal points at which T is output. This function cannot be realised according to the operational semantics of many languages because often they are deterministic and so cannot express functions like this one which examines its two arguments in parallel. It is not a sequential function.

We seek a definition of sequential function between domains based solely on the structure of the domains themselves. Two early definitions of sequential function were proposed independently by R.Milner and J.Vuillemin. These depend on viewing a function $f : D_1 \times \cdots \times D_n \to E$ between domains as having n arguments $x = (x_1, \ldots, x_i, \ldots, x_n)$ (viewing the function as having more or less arguments may change its character according to these definitions!) Assume f is a continuous function.

Then f is *M-sequential* (Milner) iff either it is constant or there is an integer i (with $1 \leq i \leq n$) such that f is strict in its ith argument (*i.e.* $x_i = \bot \Rightarrow f(x) = \bot$) and the function obtained by fixing its ith argument is M-sequential.

On the other hand, f is *V-sequential* (Vuillemin) iff it is a constant or there is an integer i (with $1 \leq i \leq n$) such that

$$x \sqsubseteq y \ \ \& \ \ x_i = y_i \Rightarrow f(x) = f(y)$$

for any $x, y \in D_0 \times \cdots \times D_1$.

Note the definitions depend on the grouping of argument places, and in particular that if we regard x as occupying a single argument place the function f would then be both M and V sequential. The two above definitions of sequential do not agree in general. However, importantly,

they do coincide and appear correct in the situation where $D_0, \ldots, D_1$ and E are flat domains, *i.e.* those for which $d = \perp$ or $\perp \prec d$ for all elements d.

G.Kahn and G.Plotkin sought a very general definition of sequential function which unlike M and V sequentiality was independent of the way that the function was viewed as having arguments. Reasonably, the definition should agree with M and V sequentiality in the case where the domains $D_1, \ldots, D_n$ and E are flat. They achieved this by axiomatising a wide class of domains for which there was a natural definition of places accessible from a point. Places are a generalisation of argument–places of functions. Unlike argument places, however, places are defined independently of the way the domain is viewed as a product. Their definition of sequential then agrees locally with M or V sequentiality. Recognising that the notion of sequential depended on the nature of the program terms denoted in the domains they chose to axiomatise only the first–order domains consisting of basic input or output values and so include domains of integers, truth values, tapes and trees.

Kahn and Plotkin first axiomatised the concrete domains and then discovered they could be represented by a *concrete data structure* (rather like a Petri net). Our presentation is the other way round. A concrete data structure consists of places which can be occupied by at most one of a set of events. In general a place may not be occupied immediately but must wait until this is enabled by certain events. A place may be thus enabled by several different sets of events. (As an example the nth place of a list is enabled by the event of making the $(n-1)$th entry. We now give the formal definition of a concrete data structure M and its configurations.

A *concrete data structure* C is a quadruple $(P, E, l, \vdash)$ where:

P is a set of *places*,

E is a countable set of *events*,

l is a function from E onto P locating events at places,

$\models$ is a subset of $\mathrm{Fin}E \times P$ called the *enabling* relation.

Such a concrete data structure determines an event structure and so a family of configurations. The events are the same. Define the consistency predicate by

$$X \in \mathrm{Con} \Leftrightarrow X \subseteq_{fin} E \ \ \& \ \ \forall e, e' \in X.\ l(e) = l(e') \Rightarrow e = e'.$$

Thus events are not allowed to occur together if they occupy the same place. Define the enabling on the event structure by

$$X \vdash e \Leftrightarrow \exists Y \subseteq X, p \in P.\ Y \models p \ \ \& \ \ l(e) = p,$$

for $X \in \mathrm{Con}$ and $e \in E$. The configurations of C, written $\mathcal{F}(\mathrm{C})$, are taken to be the configurations of the associated event structure. Say C is *stable* iff the associated event structure is.

Domains which are isomorphic to $(\mathcal{F}(\mathrm{C}), \subseteq)$ for some concrete data structure C are said to be *concrete*.

The following definitions are important in defining sequential functions.

Let C be a concrete data structure. Suppose $x \in \mathcal{F}(\mathrm{C})$ and p is a place of C.

Say x *fills* p iff $\exists e \in x.\ l(e) = p$.

Say p is *accessible* from x iff x does not fill p and $\exists X \subseteq x.\ X \models p$.

Write $p(x)$ for the set of places accessible at x.

For x, y in $\mathcal{F}(\mathrm{C})$ and a place p write $x \xrightarrow{p} y$ iff $x \sqsubseteq y$ and p is accessible from x and y fills p.

Thus we can tentatively define a function $f : \mathcal{F}(\mathrm{C}_0) \to \mathcal{F}(\mathrm{C}_1)$ to be *sequential* if it is sequential at all x in $\mathcal{F}(\mathrm{C}_0)$ where this means

$$\forall p' \in p(f(x)).\ ([\exists z.\ x \sqsubseteq z\ \ \&\ \ f(x) \xrightarrow{p'} f(y)] \Rightarrow$$
$$\exists p \in p(x).\ [\forall y.\ x \sqsubseteq y\ \ \&\ \ f(x) \xrightarrow{p'} f(y) \Rightarrow x \xrightarrow{p} y]).$$

This says to fill p' accessible from $f(x)$ there is some p accessible from x which must be filled; it generalises V-sequentiality. Of course, it is not immediately clear that this definition gives the same notion of sequential for different ways of generating isomorphic domains. This is the case however, a fact which follows from the particular representation provided for concrete domains in [W, BC, C].

We shall not give the most general representation theorem here but mention a simpler one in the case when the concrete data structure is stable. It involves an axiom called Q by Kahn and Plotkin.

1.5.1 Theorem.
Let C be a stable concrete data structure. The family of configurations ordered by inclusion forms a finitary prime algebraic domain which satisfies

$$(Q) \qquad x \sqsubseteq y\ \ \&\ \ x \prec z\ \ \&\ \ x \not\asymp y \Rightarrow \exists! t \sqsubseteq y.\ x \prec t\ \ \&\ \ t \not\asymp z.$$

Let D be a finitary prime algebraic domain which satisfies axiom (Q). Then D is a concrete domain.

Proof. It is easy to check stable concrete data structures satisfy (Q) and the other properties have already been dealt with for event structures. We omit the construction which shows that the domains mentioned are concrete. Proofs can be found in [KP], [W] or [C]. A key idea is to recover places from the domain as equivalence classes of prime intervals under the least equivalence relation $\approx$ such that

$$((c = d\ \ \&\ \ c' \not\asymp d')\ \ \text{or}\ \ [c, c'] \sim [d, d']) \Rightarrow [c, c'] \approx [d, d'].$$

As we have already seen events can be recovered as equivalence classes of prime intervals under $\sim$. ∎

1.6. A complete partial order of event structures.

There is a useful ordering on event structures which is a representation of the notion of rigid embedding in [KP]. It is useful for giving meaning to recursively defined event structures. The order is based on an idea of substructure.

1.6.1 Definition. Let $\mathrm{E}_0 = (E_0, \mathrm{Con}_0, \vdash_0)$ and $\mathrm{E}_1 = (E_1, \mathrm{Con}_1, \vdash_1)$ be event structures. Define

$$\mathrm{E}_0 \trianglelefteq \mathrm{E}_1 \Leftrightarrow E_0 \subseteq E_1,$$
$$\forall X.\ X \in \mathrm{Con}_0 \Leftrightarrow X \subseteq E_0\ \ \&\ \ X \in \mathrm{Con}_1\ \ \text{and}$$
$$\forall X, e.\ X \vdash_0 e \Leftrightarrow X \subseteq E_0\ \ \&\ \ e \in E_0\ \ \&\ \ X \vdash_1 e.$$

In this case say E_0 is a *substructure* of E_1.

The notion of substructure is closely tied to that of restriction, an important operation in its own right.

1.6.2 Definition. Let $\mathrm{E} = (E, \mathrm{Con}, \vdash)$ be an event structure. Let $A \subseteq E$. Define the *restriction* of E to A to be

$$\mathrm{E}\lceil A = (A, \mathrm{Con}_A, \vdash_A)$$

where

$$X \in \mathrm{Con}_A \Leftrightarrow X \subseteq A \ \ \& \ \ X \in \mathrm{Con},$$
$$X \vdash_A e \Leftrightarrow X \subseteq A \ \ \& \ \ e \in A \ \ \& \ \ X \vdash e.$$

1.6.3 Proposition. *Let $E = (E, Con, \vdash)$ be an event structure. Let $A \subseteq E$. Then $E\lceil A$ is an event structure.*

Let $E_0 = (E_0, Con_0, \vdash_0)$ and $E_1 = (E_1, Con_1, \vdash_1)$ be event structures. Then

$$E_0 \trianglelefteq E_1 \Leftrightarrow E_0 = E_1\lceil E_0.$$

If $E_0 \trianglelefteq E_1$ and $E_0 = E_1$ then $E_0 = E_1$.

Proof. Obvious from the definitions. ∎

This definition of substructure almost gives a complete partial order (cpo) of event structures. There is a least event structure, the unique one with the emptyset of events. Each ω–chain of event structures, increasing with respect to $\trianglelefteq$ has a least upper bound, with events, consistency and enabling relations the union of those in the chain. But of course event structures form a class and not a set and for this reason alone they do not quite form a cpo. We call structures like cpos but on a class rather than a set *large cpos*. This is all we need. (Very similar approaches for solving domain equations, or equations for structures like domains, occur in [C], [LW], [W1], [A] and [S1].)

1.6.4 Theorem. *The relation $\trianglelefteq$ is a partial order on event structures. It has a least event structure $\underline{\emptyset} =_{def} (\emptyset, \{\emptyset\}, \emptyset)$. An ω–chain of event structures $E_0 \trianglelefteq E_1 \cdots \trianglelefteq E_n \trianglelefteq \cdots$ where $E_n = (E_n, \mathrm{Con}_n, \vdash_n)$ has a least upper bound*

$$\bigcup_{n\in\omega} E_n = (\bigcup_{n\in\omega} E_n, \bigcup_{n\in\omega} Con_n, \bigcup_{n\in\omega} \vdash_n).$$

Proof. Routine. ∎

It is easy to extend the substructure relation to n–tuples of event structures. They form a large cpo too.

1.6.5 Definition. Write Π_j for the *projection* map $\Pi_j(\mathrm{E}_0, \ldots, \mathrm{E}_{n-1}) = \mathrm{E}_j$ on n–tuples of event structures. For n–tuples,

$$(\mathrm{E}_0, \ldots, \mathrm{E}_{n-1}) \trianglelefteq (\mathrm{E}'_0, \ldots, \mathrm{E}'_{n-1}) \text{ iff } \mathrm{E}_0 \trianglelefteq \mathrm{E}'_0 \ \ \& \ \ \cdots \ \ \& \ \ \mathrm{E}_{n-1} \trianglelefteq \mathrm{E}'_{n-1}.$$

1.6.6 Proposition. *For a particular integer n, the relation $\trianglelefteq$ is a partial order on n–tuples of event structures with least element $(\underline{\emptyset}, \cdots, \underline{\emptyset})$. There are least upper bounds of increasing ω-chains in n–tuples of event structures; in each coordinate j the least upper bound $\bigcup_i E_i$ of a chain $E_0 \trianglelefteq E_1 \cdots \trianglelefteq E_n \trianglelefteq \cdots E$ satisfies $\Pi_j(\bigcup_i E_i) = \bigcup_i \Pi_j(E_i)$.*

Thus, as an example, the above proposition says the projection maps Π_j are continuous on tuples of event structures ordered by $\trianglelefteq$.

Fortunately in reasoning about the monotonicity and continuity of an operation we need only consider one input coordinate and one output coordinate at a time because of the following facts, well–known for cpos.

1.6.7 Proposition. *Let* **F** *be an operation on n–tuples of event structures.*

It is monotonic, respectively continuous, (with respect to $\trianglelefteq$*) iff it is monotonic, respectively continuous, in each argument separately (i.e. considered as a function in any one of its argument, holding the others fixed).*

Similarly it is monotonic, respectively continuous, (with respect to $\trianglelefteq$*) iff it is monotonic, respectively continuous, considered as a function to each output coordinate (i.e. each function* Π_j **F** *is continuous for* $j < n$*).*

Thus in verifying that an operation is monotonic or continuous we ultimately have to show certain unary operations are continuous with respect to the substructure relation $\trianglelefteq$. The next lemma will be a great help in proving operations continuous. Generally it is very easy to show that a unary operation is monotonic with respect to $\trianglelefteq$ and continuous on the sets of events, a notion we now make precise.

1.6.8 Definition. Say a unary operation **F** on event structures is *continuous on events* iff for any ω–chain, $\mathrm{E}_0 \trianglelefteq \mathrm{E}_1 \cdots \trianglelefteq \mathrm{E}_n \trianglelefteq \cdots E$, each event of $\mathrm{F}(\bigcup_i \mathrm{E}_i)$ is a event of $\bigcup_i \mathrm{F}(\mathrm{E}_i)$.

1.6.9 Lemma. *Let F be a unary operation on event structures. Then F is continuous iff F is monotonic with respect to* $\trianglelefteq$ *and continuous on events.*

Proof.

only if: obvious.

if: Let $\mathrm{E}_0 \trianglelefteq \mathrm{E}_1 \cdots \trianglelefteq \mathrm{E}_n \trianglelefteq \cdots E$ be an ω–chain of event structures. Clearly $\bigcup_i \mathrm{F}(\mathrm{E}_i) \trianglelefteq \mathrm{F}(\bigcup_i \mathrm{E}_i)$ since F is assumed monotonic. Thus from the assumption the events of $\bigcup_i \mathrm{F}(\mathrm{E}_i)$ are the same as the events of $\mathrm{F}(\bigcup_i \mathrm{E}_i)$. Therefore they are the same event structure by proposition 1.6.3. ∎

Now we relate the substructure relation on event structures to corresponding relations on families of configurations and domains. The substructure relation represents the rigid embeddings of Kahn and Plotkin [KP].

1.6.10 Definition. Let D_0 and D_1 be domains. Let $f : \mathrm{D}_0 \to \mathrm{D}_1$ be a continuous function. Say f is an *embedding* iff there is a continuous function $g : \mathrm{D}_1 \to \mathrm{D}_0$, called a *projection*, such that

$$gf(d) = d \text{ for all } d \in D_0 \text{ and}$$
$$fg(c) \sqsubseteq c \text{ for all } c \in D_1.$$

Say f is a *rigid embedding* iff it is an embedding with projection g such that

$$c \sqsubseteq f(d) \Rightarrow fg(c) = c$$

for all $d \in D_0, c \in D_1$.

1.6.11 Proposition. *Let* E_0 *and* E_1 *be event structures such that* $E_0 \trianglelefteq E_1$*. The inclusion map* $i : \mathcal{F}(E_0) \to \mathcal{F}(E_1)$ *is a rigid embedding with projection* $j : \mathcal{F}(E_1) \to \mathcal{F}(E_0)$ *given by* $j(y) = \bigcup\{x \in \mathcal{F}(E_0) \mid x \subseteq y\}$ *for* $y \in \mathcal{F}(E_1)$*.*

Proof. Straightforward. ∎

It is well–known that continuous functions on cpos have least fixed points and the argument is virtually the same for continuous operations on large cpos.

1.6.12 Definition. Let **D** be a large cpo ordered by $\trianglelefteq$, with least upper bounds $\bigcup X$ when they exist. Let F be a continuous operation on **D**. Define *fix* F to be the least upper bound

$$\bigcup_{n\in\omega} \mathrm{F}^n(\emptyset).$$

1.6.13 Proposition. *For the situation in the above definition, the element fix F of* **D** *is the least fixed point of F.*

We finish this section with a simple example of a recursively defined event structure. The operation we consider is that of *prefixing* (sometimes called lifting, or guarding) whose effect on an event structure is to adjoin an extra initial event. Then once it has occurred the behaviour resumes as that of the original event structure.

1.6.14 Definition. Let a be an event. For an event structure $\mathrm{E} = (E, \mathrm{Con}, \vdash)$ define $a\mathrm{E}$ to be the event structure $(E', \mathrm{Con}', \vdash')$ where

$$\begin{aligned}
&E' = \{(0,a)\} \cup \{(1,e) \mid e \in E\},\\
&X \in \mathrm{Con}' \Leftrightarrow \{e \mid (1,e) \in X\} \in \mathrm{Con},\\
&X \vdash' e' \Leftrightarrow e' = (0,a) \ \text{ or } \ [e' = (1,e_1) \ \ \&\ \ (0,a) \in X \ \ \&\ \ \{e \mid (1,e) \in X\} \vdash e_1].
\end{aligned}$$

1.6.15 Proposition. *For any event a the operation $a(\ )$ is $\trianglelefteq$–continuous on event structures. The least fixed point fix $a(\ )$ has events in 1–1 correspondence with strings in the regular language 1^*0a; any finite subset of events is consistent and the enabling relation satisfies*

$$\begin{aligned}
&\emptyset \vdash 0a,\\
&X \vdash 1^n0a \Leftrightarrow \{0a, \cdots, 1^{n-1}0a\} \subseteq X,
\end{aligned}$$

for $n \geq 1$. In fact the map $1^n0a \mapsto (n+1)$ gives an isomorphism fix $a(\ ) \cong \Omega$—the two event structures are the same but for renaming of events.

Proof. Exercise. ∎

One thing may be puzzling the reader; why do we build a large cpo from the relation $\trianglelefteq$ rather than the simpler relation based on coordinatewise inclusion of an event structure in another? This is a partial order and does indeed give another large cpo and in many cases does suffice. However it suffers a drawback; the function space construction on event structures—defined in part 4—while being continuous in its right argument is not even monotonic in its left argument with respect to this inclusion order.

2. EVENT STRUCTURE SEMANTICS OF COMMUNICATING PROCESSES.

Event structures are applied to give a non–interleaving semantics to parallel programming languages like CCS and CSP, based on the idea that processes communicate by events of synchronisation. There are natural morphisms between event structures including for example morphisms which

project the events of a parallel composition to events of its components. Useful constructions like parallel composition and sum of event structures are derived simply from from categorical constructions. These yield abstract characterisations of constructions to within isomorphism. Morphisms on event structures induce morphisms on other classes of models like trees. The relationship between models can often be expressed as a coreflection between categories. Because of the way coreflections preserve limits and colimits, this leads to a smooth translation between semantics in terms of one model and semantics in terms of another. Then there are adjunctions between with other models and semantics in terms of them can be expressed as adjunctions

2.1. Morphisms to express synchronisation.

Here, in part 2, we choose a particular interpretation of events. They are to be either internal actions or actions of synchronisation of the kind that appear in CCS and CSP (see [M1, 2], [H, HBR]). Henceforth, we shall deal mainly with stable event structures.

2.1.1 Notation. We shall be working with partial functions θ on events. We indicate that θ is a partial function from E_0 to E_1 by writing $\theta : E_0 \rightarrow_* E_1$. Then it may not be the case that $\theta(e)$ is defined and sometimes we use $*$ to represent undefined, so $\theta(e) = *$ means the same as $\theta(e)$ is undefined. It is a nuissance when using predicates like $\theta(e) \in X$ to always have to say "provided $\theta(e)$ is defined". Instead we adopt the convention that the basic predicates of equality and membership are strict in the sense that if they mention $\theta(e)$ this implies $\theta(e)$ is defined. Under this convention, for example,

$$\theta(e) \in X \Rightarrow \theta(e) \text{ is defined, and}$$
$$\theta(e) = \theta(e') \Rightarrow \theta(e) \text{ is defined } \&\ \theta(e') \text{ is defined.}$$

As usual we represent the image of a set under a partial function by

$$\theta X = \{\theta(e) \mid e \in X \ \&\ \theta(e) \text{ is defined}\}.$$

Here morphisms are introduced which show the way in which the occurrences of events of in one process imply the synchronised occurrences of events in another process.

2.1.2 Definition. Let $\mathrm{E}_0 = (E_0, \mathrm{Con}_0, \vdash_0)$ and $\mathrm{E}_1 = (E_1, \mathrm{Con}_1, \vdash_1)$ be stable event structures. A *(partially synchronous) morphism* from E_0 to E_1 is a partial function $\theta : E_0 \rightarrow_* E_1$ on events which satisfies

(i) $X \in \mathrm{Con}_0 \Rightarrow \theta X \in \mathrm{Con}_1$,
(ii) $\{e, e'\} \in \mathrm{Con}_0 \ \&\ \theta(e) = \theta(e') \Rightarrow e = e'$ and
(iii) $X \vdash_0 e \ \&\ \theta(e)$ is defined $\Rightarrow \theta X \vdash_1 \theta(e)$.

Say a morphism is *synchronous* if it is a total function.
(Note by the convention stated in 2.1.1 the truth of $\theta(e) = \theta(e')$ asserts also that $\theta(e)$ and $\theta(e')$ are defined.)

For a morphism $\theta : \mathrm{E}_0 \rightarrow \mathrm{E}_1$ on event structures an event e is imagined to synchronise with the event $\theta(e)$ whenever it is defined. The partial function θ preserves consistency (i) and enabling (iii) and (ii) expresses that it preserves events in the sense that no two distinct events which are consistent with eachother can together synchronise with a common event in the image. When θ is synchronous every occurrence of an event of E_0 is linked to a synchronised occurrence of an event in E_1.

2.1.3 Proposition.
Stable event structures with morphisms of event structures form a category with composition the usual composition of partial functions and identity morphisms the identity functions on events. Stable event structures with synchronous morphisms form a subcategory.

2.1.4 Definition.
Write $\mathbf{E}$ for the category of stable event structures with morphisms of event structures.
Write $\mathbf{E}_{syn}$ for the category of stable event structures with synchronous morphisms.

As one would hope morphisms preserve configurations.

2.1.5 Proposition. *Let $\theta : E_0 \to E_1$ be a morphism of stable event structures. Then*

$$x \in \mathcal{F}(E_0) \Rightarrow (\theta x \in \mathcal{F}(E_1) \ \ \& \ \ \forall e, e' \in x.\ \theta(e) = \theta(e') \Rightarrow e = e').$$

Proof. Let $x \in \mathcal{F}(\mathrm{E}_0)$. Any finite subset of θx is the image of a finite subset of x which is consistent. Thus by property (i) in the definition of morphisms we see θx is consistent. Suppose $\theta(e) \in \theta x$. Then, by (iii), the image of a securing for e in x forms a securing for $\theta(e)$ in θx. Hence $\theta x \in \mathcal{F}(\mathrm{E}_1)$. The additional property follows directly from (ii). ∎

Similarly, morphisms between event structures induce functions on domains.

2.1.6 Definition. Let $(D_0, \sqsubseteq_0)$ and $(D_1, \sqsubseteq_1)$ be partial orders. Let f be a function $f : D_0 \to D_1$. Say f is

(i) *additive* iff
$$\forall X \subseteq D_0.\ X\uparrow \Rightarrow f(\bigsqcup X) = \bigsqcup fX,$$

(ii) *stable* iff
$$\forall X \subseteq D_0.\ X \neq \emptyset \ \ \& \ \ X\uparrow \Rightarrow f(\bigsqcap X) = \bigsqcap fX,$$

(iii) $\preceq$*-preserving* iff
$$\forall x, x' \in D_0.\ x \prec x' \Rightarrow f(x) \preceq f(x'),$$

(iv) $\prec$-preserving iff
$$\forall x, x' \in D_0.\ x \prec x' \Rightarrow f(x) \prec f(x').$$

(We use $x \preceq x'$ to mean $x = x'$ or $x \prec x'$.)

2.1.7 Proposition. *Let $\theta : E_0 \to E_1$ be a morphism of stable event structures. Then the function $x \mapsto \theta x$ from $\mathcal{F}(E_0)$ to $\mathcal{F}(E_1)$ is additive, stable and $\preceq$-preserving. If θ is synchronous then, moreover, it is $\prec$-preserving.*

Proof. Easy. ∎

Note incidentally that the substructure relation is associated with a morphism.

2.1.8 Proposition. *Suppose $E_0 \trianglelefteq E_1$. Then the inclusion map $i : E_0 \hookrightarrow E_1$ is a synchronous morphism.*

Proof. Obvious. ∎

2.2. Constructions on event structures.

The categories $\mathbf{E}$ and $\mathbf{E}_{syn}$ have products and coproducts. Of course like all limits and colimits they are determined uniquely up to isomorphism. They are intuitively natural constructions and provide a basis for defining and proving relations between different semantics for languages like $\mathbf{Proc}_L$. They generalise and make more uniform and less *ad hoc* the kind of constructions used in [F] and [MS], and elsewhere.

2.2.1 Definition. Let $\mathrm{E}_0 = (E_0, \mathrm{Con}_0, \vdash_0)$ and $\mathrm{E}_1 = (E_1, \mathrm{Con}_1, \vdash_1)$ be stable event structures. Define their *partially synchronous product* $\mathrm{E}_0 \times \mathrm{E}_1$ to be the structure $(E, \mathrm{Con}, \vdash)$ consisting of events E of the form

$$E_0 \times_* E_1 = \{(e_0, *) \mid e_0 \in E_0\} \cup \{(*, e_1) \mid e_1 \in E_1\} \cup \{(e_0, e_1) \mid e_0 \in E_0 \ \ \& \ \ e_1 \in E_1\},$$

the product in sets with partial functions with projections $\pi_i : E \rightarrow_* E_i$, given by $\pi_i(e_0, e_1) = e_i$, for $i = 0, 1$, consistency predicate Con given by

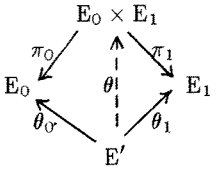
$$\begin{aligned} X \in \mathrm{Con} \Leftrightarrow (X \subseteq_{fin} E \ \ \& \ \ \pi_0 X \in \mathrm{Con}_0 \ \ \& \ \ \pi_1 X \in \mathrm{Con}_1 \ \ \& \\ \forall e, e' \in X.(\pi_0(e) = \pi_0(e') \ \ \text{or} \ \ \pi_1(e) = \pi_1(e') \Rightarrow e = e')), \end{aligned}$$

enabling relation $\vdash$ given by

$$\begin{aligned} X \vdash e \Leftrightarrow & X \in \mathrm{Con} \ \ \& \ \ e \in E \ \ \& \\ & (\pi_0(e) \text{ is defined} \Rightarrow \pi_0 X \vdash_0 \pi_0(e)) \ \ \& \ \ (\pi_1(e) \text{ is defined} \Rightarrow \pi_1 X \vdash_1 \pi_1(e)). \end{aligned}$$

2.2.2 Theorem. *The partially synchronous product $E_0 \times E_1$ of two stable event structures E_0 and E_1, with projections π_0 and π_1, is a product in the category* $\mathbf{E}$.
The product is continuous with respect to $\trianglelefteq$.

Proof. Clearly $\mathrm{E}_0 \times \mathrm{E}_1$ is an event structure, which we shall assume is $(E, \mathrm{Con}, \vdash)$. It is also stable—the proof uses both parts in the definition of Con. It is easy to see that the projections π_0 and π_1 are morphisms. Assume $\theta_0 : \mathrm{E}' \rightarrow \mathrm{E}_0$ and $\theta_1 : \mathrm{E}' \rightarrow \mathrm{E}_1$ are morphisms from a stable event structure $\mathrm{E}' = (E', \mathrm{Con}', \vdash')$. To be a product we require that there is a unique morphism $\theta : \mathrm{E}' \rightarrow \mathrm{E}_0 \times \mathrm{E}_1$ making the following diagram commute:



Because the events and projections $E_0 \times_* E_1, \pi_0, \pi_1$ are a product in the category of sets with partial functions there is no doubt about the uniqueness of θ; if it exists it is the partial function which acts on an event e of E' according to

$$\theta(e) = (\theta_0(e), \theta_1(e))$$

with the understanding that $(*, *)$ is interpreted as undefined. (Recall our use of $*$ for undefined.) Thus it only remains to show that θ as defined is a morphism $\mathrm{E}' \rightarrow \mathrm{E}_0 \times \mathrm{E}_1$, *i.e.* that conditions (i), (ii) and (iii) hold in 2.1.2:

(i) Let $X \in \mathrm{Con}'$. We require $\theta X \in \mathrm{Con}$. But certainly $\pi_k \theta X = \theta_k X \in \mathrm{Con}_k$, for $k = 0, 1$, as each θ_k is a morphism. Further if $e, e' \in \theta X$ then e, e' have the form $e = \theta(t), e = \theta(t')$. If $\pi_k(e) = \pi_k(e')$,

for $k = 0, 1$, then $\theta_k(t) = \pi_k\theta(t) = \pi_k\theta(t') = \theta_k(t')$. As both θ_0 and θ_1 are morphisms, in either case, $k = 0$ or $k = 1$, we obtain $t = t'$ so $e = e'$.

(ii) and (iii) use arguments of a similar style and are left to the reader.

Finally, we see that $\times$ is $\trianglelefteq$–continuous by an application of lemma 1.6.9. It is straightforward to check that it is monotonic and continuous on events for each argument separately, and so is $\trianglelefteq$–continuous. ∎

We characterise the configurations of the product of two event structures in terms of their configurations.

2.2.3 Proposition. *Let $E_0 \times E_1$ be the product of stable event structures with projections π_0, π_1. Let $x \subseteq E_0 \times_* E_1$, the events of the product. Then $x \in \mathcal{F}(E_0 \times E_1)$ iff*

(a) $\quad \pi_0 x \in \mathcal{F}(E_0)$ & $\pi_1 x \in \mathcal{F}(E_1)$,

(b) $\quad \forall e, e' \in x.\ \pi_0(e) = \pi_0(e')$ or $\pi_1(e) = \pi_1(e') \Rightarrow e = e'$,

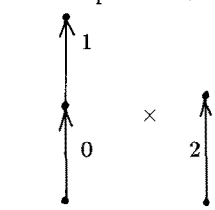
(d) $\quad \forall e \in x \exists y \subseteq x.\ \pi_0 y \in \mathcal{F}(E_0)$ & $\pi_1 y \in \mathcal{F}(E_1)$ & $e \in y$ & $|y| < \infty$ *and*

(c) $\quad \forall e, e' \in x.\ e \neq e' \Rightarrow \exists y \subseteq x.\ \pi_0 y \in \mathcal{F}(E_0)$ & $\pi_1 y \in \mathcal{F}(E_1)$ & $(e \in y \Leftrightarrow e' \notin y)$.

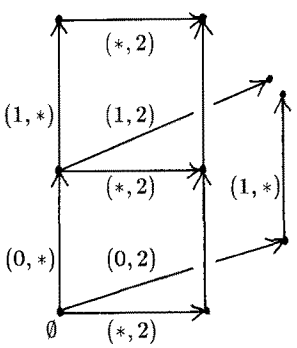
Proof. Routine. ∎

Remark. Refer to [W1](examples 3.11 and 3.12) for examples which show the necessity of properties (c) and (d) for the "if" direction of the proof.

2.2.4 Example. The configurations of the product of two trees



look like:



2.2.5 Definition. Let $\mathrm{E}_0 = (E_0, \mathrm{Con}_0, \vdash_0)$ and $\mathrm{E}_1 = (E_1, \mathrm{Con}_1, \vdash_1)$ be stable event structures. Their *synchronous product*, $\mathrm{E}_0 \otimes \mathrm{E}_1$, is defined to be $(\mathrm{E}_0 \times \mathrm{E}_1)\lceil E_0 \times E_1$.

2.2.6 Theorem. *The synchronous product $E_0 \times E_1$ of two stable event structures E_0 and E_1, with projections the restrictions of π_0 and π_1, is a product in the category of event structures with synchronous morphisms, $\mathbf{E}_{syn}$.*

The operation $\otimes$ is $\trianglelefteq$-continuous.

Proof. This proof is similar to the proof for the product but this time the underlying category of events is that of sets with total functions. ∎

2.2.7 Example. Let $\mathrm{E} = (E, \mathrm{Con}, \vdash)$ be a stable event structure. Let Ω be the event structure defined in 1.1.4 (the "ticking clock"). Then $\mathrm{E} \otimes \Omega$ has events $E \times \omega$, consistent sets those $X \subseteq_{fin} E \times \omega$ such that

$$\pi_0 X \in \mathrm{Con} \ \ \& \ \ (\forall (e,n),(e',n') \in X.\ e = e' \ \text{ or } \ n = n' \Rightarrow (e,n) = (e',n'),$$

and enabling

$$X \vdash (e,n) \Leftrightarrow \lceil n-1 \rceil \subseteq \pi_1 X \ \ \& \ \ \pi_0 X \vdash e.$$

Thus the configurations are "sequences"

$$\{(e_0,0),(e_1,1),\cdots,(e_n,n),\cdots\}$$

of distinct events from E such that $\{e_0,e_1,\cdots,e_n\} \in \mathcal{F}(\mathrm{E})$ for all n.

2.2.8 Definition. Let $\mathrm{E}_0 = (E_0, \mathrm{Con}_0, \vdash_0)$ and $\mathrm{E}_1 = (E_1, \mathrm{Con}_1, \vdash_1)$ be stable event structures. Their *sum*, $\mathrm{E}_0 + \mathrm{E}_1$, is defined to be the structure $(E, \mathrm{Con}, \vdash)$ with events $E = \{(0,e) \mid e \in E_0\} \cup \{(0,e) \mid e \in E_1\}$, the disjoint union of sets E_0 and E_1, with injections $\iota_k : E_k \to E$, given by $\iota_k(e) = (k,e)$, for $k = 0,1$, consistency predicate

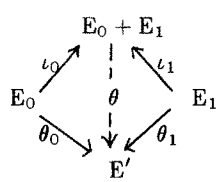
$$X \in \mathrm{Con} \Leftrightarrow (\exists X_0 \in \mathrm{Con}_0.\ X = \iota_0 X_0) \ \text{ or } \ (\exists X_1 \in \mathrm{Con}_1.\ X = \iota_1 X_1),$$

and enabling relation

$$\begin{aligned} X \vdash e \Leftrightarrow & X \in \mathrm{Con} \ \ \& \ \ e \in E \ \ \& \\ & [(\exists X_0 \in \mathrm{Con}_0, e_0 \in E_0.\ X = \iota_0 X_0 \ \ \& \ \ e = \iota_0(e_0) \ \ \& \ \ X_0 \vdash_0 e_0) \ \text{ or} \\ & (\exists X_1 \in \mathrm{Con}_1, e_1 \in E_1.\ X = \iota_1 X_1 \ \ \& \ \ e = \iota_1(e_1) \ \ \& \ \ X_1 \vdash_1 e_1)]. \end{aligned}$$

2.2.9 Theorem. *The sum $E_0 + E_1$ of two stable event structures E_0 and E_1, with injections ι_0 and ι_1, is a coproduct in both the categories $\mathbf{E}$ and $\mathbf{E}_{syn}$.*
The operation $+$ is $\trianglelefteq$-continuous.

Proof. It is easy to check that the sum is a stable event structure and that the injections are synchronous morphisms. Assume $\theta_0 : \mathrm{E}_0 \to \mathrm{E}'$ and $\theta_1 : \mathrm{E}_1 \to \mathrm{E}'$ are morphisms from a stable event structure E'. To be a coproduct we require that there is a unique morphism $\theta : \mathrm{E}_0 + \mathrm{E}_1 \to \mathrm{E}'$ making the following diagram commute:



Because the disjoint union of events with injections is a coproduct in the underlying category of sets with partial functions the uniqueness of θ is guaranteed. It is a simple matter to check that this unique θ is a morphism. Moreover if θ_0 and θ_1 are synchronous then so is θ ensuring that the sum is also the coproduct in $\mathbf{E}_{syn}$.

The continuity of + follows directly by lemma 1.6.9. ∎

It will be useful to consider more general sums as is done for transition systems and trees in the work on CCS and SCCS (see *e.g.* [M1,2]); this will help in relating our work to Milner's.

2.2.10 Definition. Let $\mathrm{E}_k = (E_k, \mathrm{Con}_k, \vdash_k)$, for $k \in K$, be a set of stable event structures indexed by a set K. Define their *indexed sum*, $\Sigma_{k\in K}\mathrm{E}_k$, to consist of events $E = \{(k,e) \mid e \in E_k\}$, the disjoint union of events, with injections $\iota_k : E_k \to E$, for $k \in K$, consistency predicate Con, where

$$X \in \mathrm{Con} \Leftrightarrow \exists k \in K.\ \exists X_k \in \mathrm{Con}_k.\ X = \iota_k X_k$$

and enabling relation $\vdash$, where $X \vdash e$ iff

$$X \in \mathrm{Con}\ \ \&\ \ e \in E\ \ \&\ \ (\exists k \in K \exists X_k, e_k.\ X = \iota_k X_k\ \ \&\ \ e = \iota_k(e_k)\ \ \&\ \ X_k \vdash_k e_k).$$

We understand the empty sum to be the null event structure $\underline{\emptyset}$.

2.2.11 Proposition. *Let $E_k = (E_k, Con_k, \vdash_k)$, for $k \in K$, be a set of stable event structures indexed by a set K with injections ι_k for $k \in K$.*
It is a coproduct in $\mathbf{E}$ and $\mathbf{E}_{syn}$.
It is a continuous K-ary operation with respect to $\trianglelefteq$.
Also

$$x \in \mathcal{F}(\Sigma_{k\in K}E_k) \Leftrightarrow (\exists k \in K \exists x_k \in \mathcal{F}(E_k).\ x = \iota_k x_k).$$

Proof. Obvious. ∎

Sums of event structures induce simple operations on families of configurations; for example configurations of $\mathrm{E}_0 + \mathrm{E}_1$ consist of copies, after renaming events, of the configurations of E_0 and E_1. Intuitively a sum has the capabilities of its components.

2.3. Synchronisation.

Individually a process P_0 is thought of as capable of performing certain events. Some of them may be communications with the environment and others may be internal events. Set in parallel with another process P_1 an event e_0 of P_0 might synchronise with an event e_1 of P_1. Whether they do or not will of course depend on what kinds of events e_0 and e_1 are because P_0 and P_1 can only perform certain kinds of synchronisation with their environments. But if they do synchronise we can think of them as forming a synchronisation event (e_0, e_1). The synchronisation event (e_0, e_1) has the same effect on the process P_0 as the component event e_0 and similarly on P_1 has the same effect as the event e_1.

Of course generally not all events of P_0 will synchronise with events of P_1; there might be an internal event of P_0 for example which by its very nature cannot synchronise with any event of P_1. So we cannot expect all events of the parallel composition to have the form (e_0, e_1). Some will have no component event from one process or the other. We can represent these events in the form $(e_0, *)$ if the event e_0 of P_0 occurs unsynchronised with any event of P_1 or $(*, e_1)$ if the event e_1 of P_1 occurs unsynchronised. The $*$ stands for the absence of an event from the corresponding component.

Thus we can view synchronisation as forming compound events from component events; a synchronisation event is viewed as a combination of events from the processes set in parallel. Whether or not synchronisations can occur is determined by the nature of the events. We use the idea of a *synchronisation algebra* to specify how events synchronise. We label events of processes to specify

how they interact with the environment, so associated with any particular sychronisation algebra is a particular parallel composition. By specialising to particular synchronisation algebras we can obtain a wide range of parallel compositions.

A *synchronisation algebra*, $(L, \bullet, *, 0)$, consists of a binary, commutative, associative operation $\bullet$ on a set of labels which always includes two distinguished elements $*$ and 0. The binary operation $\bullet$ says how labelled events combine to form synchronisation events and what labels such combinations carry. No real events are ever labelled by $*$ or 0. However their introduction allows us to specify the way labelled events synchronise without recourse to partial operations on labels. It is required that $L \setminus \{*, 0\} \neq \emptyset$.

The constant 0 is used to specify when sychronisations are disallowed. If two events labelled λ and λ' are not supposed to synchronise then their composition $\lambda \bullet \lambda'$ is 0. For this reason 0 does indeed behave like a zero with respect to the "multiplication" $\bullet$ *i.e.*

$$\forall \lambda \in L.\ \lambda \bullet 0 = 0.$$

In a synchronisation algebra, the constant $*$ is used to specify when a labelled event can or cannot occur asynchronously. An event labelled λ can occur asynchronously iff $\lambda \bullet *$ is not 0. We insist that the only divisor of $*$ is $*$ itself, essentially because we do not want a synchronisation event to disappear. We require

$$\forall \lambda, \lambda' \in L.\ \lambda \bullet \lambda' = * \Leftrightarrow \lambda = \lambda' = *.$$

We present two synchronisation algebras as examples—more can be found in [W1,2].

2.3.1 Example. *The synchronisation algebra for CCS—no value passing:* In CCS [M1] events are labelled by $\alpha, \beta, \cdots$ or their complements $\overline{\alpha}, \overline{\beta}, \cdots$ or by the label τ. The idea is that only two events bearing complementary labels may synchronise to form a synchronisation event labelled by τ. Events labelled by τ cannot synchronise further; in this sense they are invisible to processes in the environment, though their occurrence may lead to internal changes of state. All labelled events may occur asynchronously. Hence the synchronisation algebra for CCS takes the following form. The resultant parallel composition, of processes p and q say, is represented as $p|q$ in CCS.

$\bullet$	$*$	α	$\overline{\alpha}$	β	$\overline{\beta}$	$\cdots$	τ	0
$*$	$*$	α	$\overline{\alpha}$	β	$\overline{\beta}$	$\cdots$	τ	0
α	α	0	τ	0	0	$\cdots$	0	0
$\overline{\alpha}$	$\overline{\alpha}$	τ	0	0	0	$\cdots$	0	0
β	β	0	0	0	τ	$\cdots$	0	0
$\overline{\beta}$	$\overline{\beta}$	0	0	τ	0	$\cdots$	0	0
τ	τ	0	0	0	0	$\cdots$	0	0
0	0	0	0	0	0	$\cdots$	0	0
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$

2.3.2 Example. *The synchronisation algebra for $\|$ in CSP:* In the form of CSP in [H, HBR, Bk] events are labelled by $\alpha, \beta, \cdots$. There are also silent moves and following the more operational semantics in [Bk] we label them by τ. For its parallel composition $\|$ events must "synchronise on" $\alpha, \beta, \cdots$. In other words non-τ-labelled events cannot occur asynchronously. Rather, an α-labelled event in one component of a parallel composition must synchronise with an α-labelled event from the other component in order to occur; the two events must synchronise to form a synchronisation event again labelled by α. The synchronisation algebra for this parallel composition takes the following form.

$\bullet$	$*$	α	β	$\cdots$	τ	0
$*$	$*$	0	0	$\cdots$	τ	0
α	0	α	0	$\cdots$	0	0
β	0	0	β	$\cdots$	0	0
.	.	.	.	$\cdots$	.	.

Using synchronisation algebras one can define a generic programming language, inspired by CCS, SCCS and CSP but parameterised by the synchronisation algebra. For a synchronisation algebra L, the language $\mathbf{Proc}_L$ is given by the following grammar:

$$t ::= nil \mid x \mid \lambda t \mid t + t \mid t \,(\!|L|\!)\, t \mid t\lceil\Lambda \mid t[\Xi] \mid recx.t$$

where x is in some set of variables X over processes, $\lambda \in L \setminus \{*,0\}$, $\Lambda \subseteq L \setminus \{*,0\}$, and $\Xi : L \to L$ is a relabelling function preserving $*$ and 0 and such that $\Xi(\lambda) = * \Rightarrow \lambda = *$ and $\Xi(\lambda) = 0 \Rightarrow \lambda = 0$— otherwise it would not lead to a sensible labelling of events.

We explain informally the behaviour of the constructs in the language $\mathbf{Proc}_L$. The behaviour can be described accurately by the models presented in the next sections. Roughly, a process of $\mathbf{Proc}_L$ determines a pattern of event occurrences over time. The nature of the events, how they interact with the environment, is specified by associating each event with a label from the synchronisation algebra L. The term *nil* represents the *nil* process which has stopped and refuses to perform any event. A *prefixed* process λt first performs an event of kind λ to become the process t. A *sum* $t + t'$ behaves like t or t'; which branch of a sum is followed will often be determined by the context and what kinds of events the process is restricted to. A *parallel composition* process $t \,(\!|L|\!)\, t'$ behaves like t and t' set in parallel. Their events of synchronisation are those pairs of events (e_0, e_1), one from each process, where e_0 is of kind λ_0 and e_1 is of kind λ_1 so that $\lambda_0 \bullet \lambda_1 \neq 0$; the synchronisation event is then of kind $\lambda_0 \bullet \lambda_1$. The *restriction* $t\lceil\Lambda$ behaves like the process p but with its events restricted to those with labels which lie in the set Λ. A relabelled process $t[\Xi]$ behaves like p but with the events relabelled according to Ξ. A closed term $recx.t$ recursively defines a process x with body t.

2.4. Denotational semantics.

We sketch how to give denotational semantics to a range of simple parallel programming languages $\mathbf{Proc}_L$ which despite their simplicity, by varying the synchronisation algebra L, include pure CCS (just synchronisation, no value-passing [M1]), SCCS (synchronous CCS [M2]) and the better part of (theoretical) CSP of [HBR] but with just one parallel composition.

To pin down the intuitions given earlier we can take each closed term in $\mathbf{Proc}_L$ as denoting a *labelled event structure*. This is simply an event structure E, with events E labelled by elements of L, and so a structure (E, l) where $l : E \to L \setminus \{*,0\}$. Parallel compositions of event structures are defined with respect to a synchronisation algebra which specifies those pairs of events which can synchronise, those which cannot and those which may occur asynchronously.

2.4.1 Definition. Let (E_0, l_0) and (E_1, l_1) be labelled event structures with events E_0 and E_1 respectively. Assume their labels lie in a synchronisation algebra $L = (L, \bullet, *, 0)$. Define their *parallel composition*

$$(\mathrm{E}_0, l_0) \,(\!|L|\!)\, (\mathrm{E}_1, l_1) = ([\mathrm{E}_0 \times \mathrm{E}_1]\lceil S, l)$$

where

$$S = \{e \in E_0 \times_{\nearrow} E_1 \mid l_0\pi_0(e) \bullet l_1\pi_1(e) \neq 0\} \text{ and}$$
$$l(e) = l_0\pi_0(e) \bullet l_1\pi_1(e)$$

the set of allowed events in the composition, and for any event e of the composition. The other operations are simple to define; prefixing, sum and restriction are just as before but taking account of labels, and the operation of relabelling simply alters the labelling function.

In order to give a meaning to the recursively defined processes of the form *recx.t* we use the fact that the operations are continuous with respect to a large c.p.o. of labelled event structures. The large c.p.o. of event structures $\trianglelefteq$ extends naturally to labelled event structures in such a way that operations like parallel composition are continuous.

Let L be a synchronisation algebra. Define the ordering $\trianglelefteq_L$ on labelled event structures by:

$$(\mathrm{E}_0, l_0) \trianglelefteq_L (\mathrm{E}_1, l_1) \Leftrightarrow \mathrm{E}_0 \trianglelefteq \mathrm{E}_1 \ \ \& \ \ l_0 = l_1 \lceil E_0,$$

where E_0 is the set of events of E_0. The null labelled event structure $(\underline{\emptyset}, \emptyset)$ is the least L–labelled event structure with respect to $\trianglelefteq_L$. Of course, $\trianglelefteq_L$ has least upper bounds of ω–chains; the lub of a chain $(\mathrm{E}_0, l_0), (\mathrm{E}_1, l_1), \ldots, (\mathrm{E}_n, l_n), \ldots$ takes the form $(\bigcup_n \mathrm{E}_n, \bigcup_n l_n)$. All the operations prefixing, sum, restriction, relabelling and parallel composition are continuous with respect to $\trianglelefteq_L$. Thus we can give a denotational semantics to $\mathbf{Proc}_L$ by representing recursively defined processes as the least fixed points of continuous operation.

2.4.2 Definition. *Denotational semantics for* $\mathbf{Proc}_L$*:* Let L be a synchronisation algebra. Define an *environment* for process variables to be a function ρ from process variables X to labelled event structures. For a term t and an environment ρ, define the denotation of t with respect to ρ written $[\![t]\!]\rho$ by the following structural induction. Note syntactic operators appear on the left and their semantic counterparts on the right.

$$\begin{array}{llll} [\![nil]\!]\rho & = (\underline{\emptyset}, \emptyset) & [\![t\lceil \Lambda]\!]\rho & = [\![t]\!]\rho \lceil \Lambda \\ [\![x]\!]\rho & = \rho(x) & [\![t[\Xi]]\!]\rho & = [\![t]\!]\rho[\Xi] \\ [\![\lambda t]\!]\rho & = \lambda([\![t]\!]\rho) & [\![t_1 \textcircled{L} t_2]\!]\rho & = [\![t_1]\!]\rho \textcircled{L} [\![t_2]\!]\rho \\ [\![t_1 + t_2]\!]\rho & = [\![t_1]\!]\rho + [\![t_2]\!]\rho & [\![recx.t]\!]\rho & = fix\ \Gamma \end{array}$$

where Γ is an operation on labelled event structures given by $\Gamma(\mathrm{E}) = [\![t]\!]\rho[\mathrm{E}/x]$ and *fix* is the least–fixed–point operator.

Remark. A straightforward structural induction shows that Γ above is indeed continuous with respect to $\trianglelefteq_L$ so the denotation of a recursively defined process is really the least fixed point of the associated functional Γ.

Choosing L to be the appropriate synchronisation algebra we immediately obtain denotational semantics for CCS, SCCS and CSP with one parallel composition. Of course, in the semantics of CCS, for example, denotations of processes carry far more detail than the semantics generally given. In particular they include information about the concurrency or causal dependence of events, information which is missing from the interleaving semantics in [M1, 2]. Results from the next section show how the semantics relates to Milner's in [M1]; as you would expect Milner's synchronisation tree semantics is obtained by serialising, or interleaving, the denotations of the event structure semantics.

2.5. Other categories.

We have given denotational semantics to $\mathbf{Proc}_L$ in terms of event structures. In a similar way we might give semantics using families of configurations, domains, prime event structures, or trees. All such classes of structures form categories too with morphisms induced by those on stable event structures.

2.5.1 Definition.
A morphism between stable families of configurations $\mathrm{F}_0, \mathrm{F}_1$, of events E_0, E_1 respectively, is a partial function $\theta : E_0 \rightarrow_* E_1$ such that

$$\forall x \in \mathrm{F}_0.\ [\theta x \in \mathrm{F}_1 \ \ \& \ \ (\forall e, e' \in x.\ \theta(e) = \theta(e') \Rightarrow e = e')].$$

It is *synchronous* when θ is total.

A morphism between prime event structures $(E_0, \mathrm{Con}_0, \leq_0)$ and $(E_1, \mathrm{Con}_1, \leq_1)$ is a partial function $\theta : E_0 \rightarrow_* E_1$ such that

$$\forall e \in E_0.\ \theta(e) \text{ is defined} \Rightarrow [\theta(e)] \subseteq \theta[e] \ \ \&$$
$$\forall X \in \mathrm{Con}_0.\ [\theta X \in \mathrm{Con}_1 \ \ \& \ \ (\forall e, e' \in X.\ \theta(e) = \theta(e') \Rightarrow e = e')].$$

It is *synchronous* when θ is total.

A morphism between finitary prime algebraic domains D_0 and D_1 is a function $f : D_0 \rightarrow D_1$ which is additive, stable and $\preceq$-preserving. It is *synchronous* when f is $\prec$-preserving.

A morphism between trees T_0 and T_1 is a function $f : T_0 \rightarrow T_1$ which is $\preceq$-preserving and such that $f(\emptyset) = \emptyset$. It is *synchronous* when f is $\prec$-preserving.

Morphisms on prime event structures can be characterised in a slightly different way which recalls the simple way in which their configurations are generated.

2.5.2 Proposition. *Let P_0 and P_1 be prime event structures with events P_0 and P_1. A partial function $\theta : P_0 \rightarrow_* P_1$ is a morphism $\theta : P_0 \rightarrow P_1$ of prime event structures iff*

$$\forall x \in \mathcal{L}(P_0).\ (\theta x \in \mathcal{L}(P_1) \ \ \& \ \ (\forall e, e' \in x.\ \theta(e) = \theta(e') \Rightarrow e = e')).$$

Proof. Routine. ∎

The classes of structures with the appropriate morphisms under function composition give rise to categories.

2.5.3 Definition.
Let $\mathbf{F}$ be the category of stable families of configurations with morphisms of families composed as functions. Let $\mathbf{F}_{syn}$ to be subcategory with synchronous morphisms.
Let $\mathbf{P}$ and $\mathbf{P}_{syn}$ be corresponding categories of prime event structures.
Let $\mathbf{D}$ and $\mathbf{D}_{syn}$ be corresponding categories of finitary prime algebraic domains.
Let $\mathbf{T}$ and $\mathbf{T}_{syn}$ be corresponding categories of trees.

We have defined the categories above in such a way that there is a natural chain of functors

$$\mathbf{E} \xrightarrow{\mathcal{F}} \mathbf{F} \xrightarrow{D} \mathbf{D} \xrightarrow{T} \mathbf{T}.$$

The functor $\mathcal{F}$ acts on an event structure E to give $\mathcal{F}(\mathrm{E})$ and on morphisms $\theta : \mathrm{E}_0 \rightarrow \mathrm{E}_1$ to give $\mathcal{F}(\theta) : \mathcal{F}(\mathrm{E}_0) \rightarrow \mathcal{F}(\mathrm{E}_1)$ which is the partial function θ restricted to the events of $\mathcal{F}(\mathrm{E}_0)$. It is easily checked that $\mathcal{F}$ preserves identity morphisms and composition and so is indeed a functor.

The functor $\mathcal{D}$ acts on a family F to give the domain $(\mathrm{F}, \subseteq)$ and on morphisms $\theta : \mathrm{F}_0 \to \mathrm{F}_1$ to give the function $\mathcal{D}(\theta) : \mathcal{D}(\mathrm{F}_0) \to \mathcal{D}(\mathrm{F}_1)$ which acts $(\mathcal{D}(\theta))(x) = \theta x$ for $x \in \mathrm{F}_0$. It is a trivial matter to verify that $\mathcal{D}$ is well-defined and a functor.

Morphisms on trees are clearly the same as morphisms on them when regarded as domains; trees **T** form a subcategory of domains **D**. Morphisms are also induced by a "sequentialisation" functor from domains to trees. The functor $\mathcal{T}$ acts on a domain D to give the tree $\mathcal{T}\mathrm{D}$ consisting of all the covering sequences in D and on morphisms $f : \mathrm{D}_0 \to \mathrm{D}_1$ to give the function $\mathcal{T}(f) : \mathcal{T}(\mathrm{D}_0) \to \mathcal{T}(\mathrm{D}_1)$ which acts

$$(\mathcal{T}(f))\langle d_0, d_1, \ldots, d_{n-1}, \ldots\rangle = \langle f(d_0), f(d_1), \ldots, f(d_{n-1}), \ldots\rangle$$

on a covering sequence of D_0. It is easy to see $\mathcal{T}$ is a functor.

There are a number of categories now, each could be used to give denotational semantics in a manner very similar to the last section; again because the behaviour of parallel compositions should be that allowed when we project into the components we expect to model it as a restriction of a product. At first sight we face the laborious task of defining parallel compositions and sums in each category and showing how they relate to eachother. This is needed in order to verify that all the semantics are compatible. Fortunately however, the categories bear a simple relationship with one another; there is a coreflection between any two. This fact, established next, gives us, as a corollary, the form of parallel compositions and sums in the different categories and yields smooth translations between the various semantics.

A coreflection is a special form of adjunction. An adjunction between two categories **A** and **B** involves a pair of functors

$$F : \mathbf{A} \to \mathbf{B},\ G : \mathbf{B} \to \mathbf{A}$$

between them. Recall one way of determining an adjunction between two categories (see [Mac] p.81). Let $G : \mathbf{B} \to \mathbf{A}$ be a functor between categories **A** and **B**. Suppose for an object $A \in \mathbf{A}$ there is an object $F(A) \in \mathbf{B}$ and a morphism $\eta_A : A \to GF(A)$ in **A** which is universal in the following sense: For any morphism $f : A \to G(B)$ in **A** with $B \in \mathbf{B}$ there is a unique morphism $h : F(A) \to B$ in **B** such that $(G(h))\ \eta_A = f$, *i.e.* so the diagram below commutes.

In this situation we say $F(A)$, η_A is *free over* A, with respect to G. In the case where for each A, we have such $F(A)$, η_A free over A there is an adjunction from **A** to **B**. Then F extends to a functor $F : \mathbf{A} \to \mathbf{B}$ by taking $F(f)$, for $f : A \to A'$ in **A**, to be the unique morphism $F(A) \to F(A')$ in **B** such that $GF(f)\ \eta_A = \eta_{A'}\ f$. The functor F is called the *left adjoint* of G, while G is called the *right adjoint* of F. If each morphism η_A, for $A \in \mathbf{A}$, is an isomorphism then the adjunction is called a *coreflection*.

The role of the following lemma will be to determine morphisms on event structures from morphisms on domains, where events are exhibited as prime intervals.

2.5.4 Lemma. *Let $f : D_0 \to D_1$ be a morphism in* **D**. *Then*

$$([c, c'] \sim [d, d']\ \ \&\ \ f(c) \prec f(c')) \Rightarrow (f(d) \prec f(d')\ \ \&\ \ [f(c), f(c')] \sim [f(d), f(d')]).$$

Proof. Let $c \prec c'$ and $d \prec d'$ in D and suppose $[d, d'] \leq [c, c']$. Because f is additive and stable we get

$$f(d) = f(d' \sqcap c) = f(d') \sqcap f(c),$$
$$f(c') = f(d' \sqcup c) = f(d') \sqcup f(c).$$

Because f is $\preceq$-preserving too the above equations make $f(d) \prec f(d')$ iff $f(c) \prec f(c')$. It follows that if $[d,d'] \sim [c,c']$ and $f(d) \prec f(d')$ then $[f(d),f(d')] \sim [f(c),f(c')]$. ∎

The following theorem establishes the coreflections between the various categories.

2.5.5 Theorem.
The event structure $\mathcal{E}(F)$ with morphism $1_F : F \cong \mathcal{F}\mathcal{E}(F)$ is free over F with respect to $\mathcal{F}$, for each $F \in \mathbf{F}$.
The family $\mathcal{LP}r(D)$ with morphism $\phi : D \cong \mathcal{D}\mathcal{LP}r(D)$ is free over D with respect to $\mathcal{D}$, for each $D \in \mathbf{D}$, where

$$\phi(d) = \{p \sqsubseteq d \mid p \text{ is a complete prime}\}.$$

The tree T with morphism $c_T : T \cong \mathcal{T}(T)$ is free over T with respect to $\mathcal{T}$, for each $T \in \mathbf{T}$, where

$$c_T(d) = \langle d_0, d_1, \ldots, d_n, \ldots \rangle$$

where $\perp = d_0 \prec d_1 \prec \cdots \prec d_n \prec \cdots$ with $d = \bigsqcup_n d_n$ (i.e. the sequence is a branch up to d).
The resulting coreflections cut-down to coreflections between the associated categories with synchronous morphisms.

Proof. The first two isomorphisms presented above are known by earlier results (1.1.13 and 1.3.7). We only present the proof of the coreflection from **D** to **F**. The other two coreflections are easier to show, and left to the reader. The proofs go through virtually unchanged with synchronous morphisms instead giving the coreflections in the synchronous cases.

Let $\mathrm{D} \in \mathbf{D}$. Certainly, by previous results, $\mathcal{LP}r(\mathrm{D}) \in \mathbf{F}$ and $\phi : \mathrm{D} \cong \mathcal{D}\mathcal{LP}r(\mathrm{D})$ when defined as above. Suppose $\mathrm{F} \in \mathbf{F}$ and $f : \mathrm{D} \to (\mathrm{F}, \subseteq)$ is a morphism in **D**. We require a unique $\theta : \mathcal{LP}r(\mathrm{D}) \to \mathrm{F}$ in **F** so that $\mathcal{D}(\theta)\ \phi = f$.

Recall the 1–1 correspondence between complete primes and prime intervals of D under the equivalence relation $\sim$, shown in 1.3.8; an equivalence class $[d,d']_\sim$ corresponds to the unique prime p in $\phi(d') \setminus \phi(d)$. This makes it easy to define the required partial function $\theta : P \to_* E$ from complete primes P of D to events of F. It is easy to see that if $[z,z'] \sim [w,w']$ in $(\mathrm{F}, \subseteq)$ then $z' \setminus z = w' \setminus w$, both containing the same unique event. Thus, by the lemma above, the following definition of θ is well–defined:

For $p \in P$, take a prime interval $[d,d']$ whose equivalence class corresponds to p. If $f(d) \prec f(d')$ then take $\theta(p)$ to be the unique event in $f(d') \setminus f(d)$, and otherwise take $\theta(p)$ to be undefined.

Let d be a finite element of D. Take a covering chain up to d:

$$\perp = d_0 \prec d_1 \prec \cdots \prec d_n = d.$$

By induction along the chain we obtain $\theta\phi(d) = f(d)$. As both functions are additive this implies $\theta\phi(d) = f(d)$ for all $d \in D$, so the functions are equal. Hence, provided we can show θ is a morphism we do have the required commutativity $(\mathcal{D}\theta)\ \phi = f$.

Now we show θ is a morphism. Suppose $x \in \mathcal{LP}r(\mathrm{D})$. Then

$$\theta x = f\phi^{-1}(x) \in \mathrm{F}.$$

Suppose $p, p' \in x$ and $\theta(p) = \theta(p')$ being equal to e say. Assume $p \neq p'$, in order to obtain a contradiction. Take a covering chain up to $p \sqcup p'$ in D. Without loss of generality we may assume this yields

$$d \prec d' \sqsubseteq c \prec c'$$

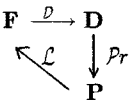
where the equivalence class of $[d, d']$ corresponds to p and that of $[c, c']$ corresponds to p'. The image under f yields

$$f(d) \subset f(d') \subseteq f(c) \subset f(c')$$

where $f(d') \setminus f(d) = f(c') \setminus f(c) = \{e\}$. But this is impossible. Hence $p = p'$.

Therefore θ is a morphism $\theta : \mathcal{L}\mathcal{P}r(\mathrm{D}) \to \mathrm{F}$ in $\mathbf{F}$ so that $(\mathcal{D}(\theta))\ \phi = f$. Any other morphism θ' satisfying $(\mathcal{D}(\theta'))\ \phi = f$ must satisfy $(\mathcal{D}(\theta')) = f\phi^{-1} = (\mathcal{D}(\theta))$ and so equal θ. So θ is unique too. ∎

There is also a triangle of functors:



Here $\mathcal{L}$ takes a prime event structure P to its family $\mathcal{L}(\mathrm{P})$ of consistent, left closed subsets, and acts on a morphism $\theta : \mathrm{P}_0 \to \mathrm{P}_1$ to give $\mathcal{L}(\theta) : x \mapsto \theta x$—this is a morphism in $\mathbf{F}$ by proposition 2.5.2, and so well-defined. The functor $\mathcal{P}r$ acts on a domain D to give the prime event structure $\mathcal{P}r(\mathrm{D})$. Its action on morphisms is more complicated to describe, and is best done using lemma 2.5.4. Let $f : \mathrm{D}_0 \to \mathrm{D}_1$ be a morphism in $\mathbf{D}$ between domains D_0 and D_1 with complete primes P_0 and P_1 respectively. Each complete prime $p \in \mathrm{P}_0$ corresponds to an equivalence class of prime intervals, as in 1.3.8, and because f respects this equivalence—the content of lemma 2.5.4—f determines a partial function $\theta : P_0 \to_* P_1$. More precisely, let $p \in P_0$ correspond to the equivalence class $[d, d']_\sim$ in D_0. Define $(\mathcal{P}r(f))(p) = p'$ if $f(d) \prec f(d')$ and p' corresponds to $[f(d), f(d')]$ in D_1, and undefined otherwise. By lemma 2.5.4, $\mathcal{P}r(f)$ is a well-defined partial function, and, as in the proof of 2.5.5 above, it can be checked that it is a morphism $\mathcal{P}r(\mathrm{D}_0) \to \mathcal{P}r(\mathrm{D}_1)$ of prime event structures (or see appendix B, lemma B9, of [W1]). Theorems 1.3.5 and 1.3.7 show how for any prime event structure P there is an isomorphism $\mathrm{P} \cong \mathcal{P}r\mathcal{D}\mathcal{L}(\mathrm{P})$ and for any domain $\mathrm{D} \in \mathbf{D}$ there is an isomorphism $\mathrm{D} \cong \mathcal{D}\mathcal{L}\mathcal{P}r(\mathrm{D})$. Further it is easy to show these are natural isomorphisms in the sense of [Mac], which is precisely what is required to establish an equivalence of the categories $\mathbf{P}$ and $\mathbf{D}$. In this way we have shown:

2.5.6 Theorem. *The functor $\mathcal{D}\mathcal{L} : \mathbf{P} \to \mathbf{D}$ is an equivalence of categories with adjoint $\mathcal{P}r$.*

We can interpret this theorem as expressing that, while on the surface the categories $\mathbf{P}$ and $\mathbf{D}$ look very different, they are essentially the same model of processes.

Thus the various categories are all related by coreflections—recall coreflections compose. Adjunctions satisfy a useful property: right adjoints preserve limits like products and left adjoints preserve colimits like coproducts [Mac p.114]. These facts, with the natural isomorphisms of the coreflections, enable us to construct products and sums in the various categories. On these can then be based constructions like parallel composition and sum in the same manner as in the last section.

2.5.7 Theorem.

(i) $F_0 \times_F F_1 \cong \mathcal{F}(\mathcal{E}(F_0) \times_E \mathcal{E}(F_1))$ *for* $F_0, F_1 \in \mathbf{F}$.

(ii) $D_0 \times_D D_1 \cong \mathcal{D}(\mathcal{L}\mathcal{P}r(D_0) \times_F \mathcal{L}\mathcal{P}r(D_1))$ *for* $D_0, D_1 \in \mathbf{D}$.

(iii) $T_0 \times_T T_1 \cong \mathcal{T}(T_0 \times_D T_1)$ *for* $T_0, T_1 \in \mathbf{T}$.

Proof. (i) For $\mathrm{F}_0, \mathrm{F}_1 \in \mathbf{F}$ we have $\mathcal{E}(\mathrm{F}_0) \times_E \mathcal{E}(\mathrm{F}_1)$ is a product in $\mathbf{E}$. It is preserved by the right adjoint $\mathcal{F}$ so

$$\mathcal{F}(\mathcal{E}(\mathrm{F}_0) \times_E \mathcal{E}(\mathrm{F}_1)) \cong (\mathcal{F}\mathcal{E}(\mathrm{F}_0)) \times_F (\mathcal{F}\mathcal{E}(\mathrm{F}_1)) \cong \mathrm{F}_0 \times_F \mathrm{F}_1.$$

The proofs of (ii) and (iii) are similar. ∎

Because left adjoints preserve coproducts we know how to construct *e.g.* the coproduct of two families F_0, F_1 provided it exists. If $F_0 +_F F_1$ exists then by the preservation of colimits,

$$\mathcal{E}(F_0 +_F F_1) \cong \mathcal{E}(F_0) +_E \mathcal{E}(F_1).$$

Hence

$$F_0 +_F F_1 \cong \mathcal{F}\mathcal{E}(F_0 +_F F_1) \cong \mathcal{F}(\mathcal{E}(F_0) +_E \mathcal{E}(F_1)).$$

But of course it must exist for this argument to apply. The following lemma provides a sufficient condition for existence.

2.5.8 Lemma. *Let $F : \mathbf{A} \to \mathbf{B}$ be a coreflection from $\mathbf{A}$ to $\mathbf{B}$ with right adjoint G. Suppose $\mathbf{B}$ has coproducts. Let $A_0, A_1 \in \mathbf{A}$. If $FG(F(A_0) +_B F(A_1)) \cong F(A_0) +_B F(A_1)$ then $G(F(A_0) +_B F(A_1))$ is a coproduct of A_0, A_1 in $\mathbf{A}$.*

Proof. Consider the image category ImF. It is a full subcategory of $\mathbf{B}$ and $F : \mathbf{A} \to ImF$ is an equivalence of categories with adjoint the restriction of G. Let $A_0, A_1 \in \mathbf{A}$. Then their images $F(A_0)$, $F(A_1)$ have a coproduct $F(A_0) +_B F(A_1)$ in $\mathbf{B}$. If $FG(F(A_0) +_B F(A_1)) \cong F(A_0) +_B F(A_1)$ then there is an object D in ImF isomorphic to $F(A_0) +_B F(A_1)$. The object D is a coproduct of $F(A_0)$, $F(A_1)$ in ImF. Hence $G(F(A_0) +_B F(A_1)) \cong GD$ is a coproduct of $GF(A_0)$, $GF(A_1)$, and thus of A_0, A_1, in $\mathbf{A}$. ∎

2.5.9 Theorem.

(i) $F_0 +_F F_1 \cong \mathcal{F}(\mathcal{E}(F_0) +_E \mathcal{E}(F_1))$ *for* $F_0, F_1 \in \mathbf{F}$.

(ii) $D_0 +_D D_1 \cong \mathcal{D}(\mathcal{LP}r(D_0) +_F \mathcal{LP}r(D_1))$ *for* $D_0, D_1 \in \mathbf{D}$.

(iii) $T_0 +_T T_1 \cong \mathcal{T}(T_0 +_D T_1)$ *for* $T_0, T_1 \in \mathbf{T}$.

Proof. It is easy to see:

(i) $\mathcal{E}\mathcal{F}(\mathcal{E}(\mathrm{F}_0) +_E \mathcal{E}(\mathrm{F}_1)) \cong \mathcal{E}(\mathrm{F}_0) +_E \mathcal{E}(\mathrm{F}_1)$ for $\mathrm{F}_0, \mathrm{F}_1 \in \mathbf{F}$,

(ii) $\mathcal{LP}r\mathcal{D}(\mathcal{LP}r(\mathrm{D}_0) +_F \mathcal{LP}r(\mathrm{D}_1)) \cong \mathcal{LP}r(\mathrm{D}_0) +_F \mathcal{LP}r(\mathrm{D}_1)$ for $\mathrm{D}_0, \mathrm{D}_1 \in \mathbf{D}$,

(iii) $\mathcal{T}(\mathrm{T}_0 +_D \mathrm{T}_1) \cong \mathrm{T}_0 +_D \mathrm{T}_1$ for $\mathrm{T}_0, \mathrm{T}_1 \in \mathbf{T}$,

from which the results follow by the lemma above. ∎

It is not hard to see that the sum of families has the effect of taking their union, once the events of the families are made disjoint, and that the sum of domains and trees essentially glues disjoint copies of them together at their bottom elements. Similarly we can define indexed sums so that they are coproducts too. For example we can define

$$\Sigma_{k \in K} \mathrm{F}_k = \mathcal{F}(\Sigma_{k \in K} \mathcal{E}(\mathrm{F}_k)),$$

and we shall encounter a use for indexed sums of trees soon.

Remark. Similar theorems hold in the synchronous case.

The theorems above construct products and sums in the different categories. But of course they are all based on constructions on event structures. There are generally more direct characterisations of the constructions in the different categories. By theorem 2.5.7(i) above and 2.3.3 characterising the configurations of a product we obtain immediately such a result for families.

2.5.10 Theorem. *Let F_0 and F_1 be stable families with events E_0 and E_1. Their product in $\mathbf{F}$ is the family consisting of those subsets $x \subseteq E_0 \times_* E_1$ which satisfy (a), (b), (c) and (d) of proposition 2.3.3.*

Proof. The product $\mathrm{F}_0 \times_F \mathrm{F}_1$ in $\mathbf{F}$ is isomorphic to $\mathcal{F}(\mathcal{E}(\mathrm{F}_0) \times_E \mathcal{E}(\mathrm{F}_1))$. But 2.3.3 characterises this family as those subsets of $E_0 \times_* E_1$ which satisfy (i), (ii), (iii) and (iv). ∎

The characterisations of the product on trees $\mathbf{T}$ are simplified through the use of another functor $\mathcal{T}_F : \mathbf{F} \to \mathbf{T}$ behaving like $\mathcal{T}$ but acting on families instead of domains. Let $\mathrm{F} \in \mathbf{F}$ be a family with events E. Define $\mathcal{T}_F(\mathrm{F})$ to be the set of finite and infinite sequences s of events E ordered by extension which have the form

$$s = \langle e_1, e_2, \ldots, e_n, \ldots \rangle$$

where $\{e_1, e_2, \ldots, e_i\} \in \mathrm{F}$ for all i at which e_i is defined. Let $\theta : \mathrm{F}_0 \to \mathrm{F}_1$ be a morphism in $\mathbf{F}$. Then $\theta : E_0 \to_* E_1$. It can be extended to sequences by insisting

$$\theta^*(\langle\rangle) = \langle\rangle,$$

and

$$\theta^*(es) = \begin{cases} \theta(e)\theta^*(s) & \text{if } \theta(e) \text{ is defined,} \\ \theta^*(s) & \text{if } \theta(e) \text{ is undefined.} \end{cases}$$

We use es where e is an element and s is a sequence to denote the result of prefixing e to the beginning of s.

2.5.11 Lemma. *The operation $\mathcal{T}_F$ is a functor $\mathcal{T}_F : \mathbf{F} \to \mathbf{T}$. We have*

$$\mathcal{T}_F(F) \cong \mathcal{T}\,\mathcal{D}(F)$$

for $F \in \mathbf{F}$. Moreover this isomorphism is natural in F.

Proof. Clearly covering chains

$$\emptyset \prec x_1 \prec x_2 \prec \cdots \prec x_n \prec \cdots$$

in $(\mathrm{F}, \subseteq)$ are in 1–1 correspondence with sequences

$$\langle e_1, e_2, \ldots, e_n, \ldots \rangle$$

where $x_n = \{e_1, e_2, \ldots, e_n\}$ for each n. This gives an isomorphism $\mathcal{T}_F(\mathrm{F}) \cong \mathcal{T}((\mathrm{F}, \subseteq)) = \mathcal{T}\,\mathcal{D}(\mathrm{F})$. It is easily checked to be natural. ∎

Thus the two functors $\mathcal{T}_F, \mathcal{T}\,\mathcal{D} : \mathbf{F} \to \mathbf{T}$ are naturally isomorphic, so $\mathcal{T}_F$ is a right adjoint too. As such it preserves products yielding the following characterisation of the product of trees.

2.5.12 Lemma. *Let S and T be trees. Then*

$$S \times_T T \cong \mathcal{T}_F(\mathcal{L}\mathcal{P}r(S) \times_F \mathcal{L}\mathcal{P}r(T)).$$

Proof. As a right adjoint $\mathcal{T}_F$ preserves products. Thus $\mathcal{T}_F(\mathcal{L}\mathcal{P}r(\mathrm{S}) \times_F \mathcal{L}\mathcal{P}r(\mathrm{T}))$ is the product in $\mathbf{T}$ of $\mathcal{T}_F\mathcal{L}\mathcal{P}r(\mathrm{S})$ and $\mathcal{L}\mathcal{P}r(\mathrm{T})$. However

$$\mathcal{T}_F\mathcal{L}\mathcal{P}r(\mathrm{S}) \cong \mathcal{T}\,\mathcal{D}\mathcal{L}\mathcal{P}r(\mathrm{S}) \cong \mathcal{T}(\mathrm{S}) \cong \mathrm{S},$$

using the natural isomorphism between $\mathcal{T}_F$ and $\mathcal{T}\,\mathcal{D}$, the eqivalence of categories $\mathbf{D}$ and $\mathbf{P}$ and the coreflection from $\mathbf{T}$ to $\mathbf{D}$. ∎

This explains the product of trees in terms of the image under $\mathcal{T}_F$ of the product of families. Such an image can be understood very simply.

2.5.13 Lemma. *Let F_0 and F_1 be stable families with events E_0 and E_1 respectively. Then*

$$\mathcal{T}_F(F_0 \times_F F_1)$$

is the product of trees $\mathcal{T}_F(F_0)$, $\mathcal{T}_F(F_1)$. Moreover, for s a sequence of events in $E_0 \times_* E_1$,

$$s \in \mathcal{T}_F(F_0 \times_F F_1) \Leftrightarrow \pi_0^*(s) \in \mathcal{T}_F(F_0) \ \ \& \ \ \pi_1^*(s) \in \mathcal{T}_F(F_1),$$

where π_k are the projections $E_0 \times_* E_1 \to_* E_k$ for $k = 0, 1$.

Proof. The tree $\mathcal{T}_F(\mathrm{F}_0 \times_F \mathrm{F}_1)$ is the product stated as $\mathcal{T}_F$ preserves products. By the definition of how $\mathcal{T}_F$ acts on objects, if $s \in \mathcal{T}_F(\mathrm{F}_0 \times_F \mathrm{F}_1)$ then $\pi_0^* s \in \mathcal{T}_F(\mathrm{F}_0)$ and $\pi_1^* s \in \mathcal{T}_F(\mathrm{F}_1)$. The converse follows by 2.3.3 characterising configurations of the product. ∎

For a family F, the construction consists of the sequences of events allowed by F. The result above expresses the intuitive fact that the sequences of events allowed by the product of families are precisely those sequences whose projections are allowed by the components.

Another characterisation of the product of trees yields Milner's expansion theorem.

2.5.14 Definition. Define prefixing on domains to be induced by the operation on event structures. Let $\mathrm{D} \in \mathbf{D}$ and e be any element (not generally in D). Define $e\mathrm{D}$ to be the domain $\mathcal{DF}(e\mathcal{EL}\mathcal{P}r(\mathrm{D}))$.

2.5.15 Theorem. *Let S and T be trees. Then*

$$S \cong \sum_{a \in A} aS_a \quad \text{and} \quad T \cong \sum_{b \in B} bT_b$$

for some sets of events A and B and trees S_a and T_b indexed by $a \in A$ and $b \in B$ respectively. We have the following characterisation of the product of S and T in **Tr** *:*

$$S \times T \cong \sum_{a \in A} (a,*)S_a \times T +_T \sum_{a \in A, b \in B} (a,b)S_a \times T_b +_T \sum_{b \in B} (*,b)S \times T_b.$$

Proof. In proving the characterisation it is smoothest to represent trees as sequences of events allowed by families. We introduce some notation. Let S be a set of sequences and a an element. Define

$$a^\frown S = \{\langle\rangle\} \cup \{as \mid s \in S\}.$$

Let F be a family of configurations with a configuration $\{a\}$. Define the family of sets

$$\mathrm{F}/a = \{x \setminus a \mid a \in x \ \ \& \ \ x \in \mathrm{F}\}.$$

Clearly if F is stable then so is F/a.

Let F be a stable family. Then directly from the definition we obtain this recursive characterisation of $\mathcal{T}_F$

$$\mathcal{T}_F(\mathrm{F}) = \bigcup_{\{a\} \in F} a^\frown \mathcal{T}_F(\mathrm{F}/a).$$

Let F_0 and F_1 be stable families. By lemma 2.5.13,

$$u \in \mathcal{T}_F(\mathrm{F}_0 \times_F \mathrm{F}_0) \Leftrightarrow \pi_0^*(u) \in \mathcal{T}_F(\mathrm{F}_0) \ \ \& \ \ \pi_1^*(u) \in \mathcal{T}_F(\mathrm{F}_1),$$

where π_0, π_1 are the projections of the product in $\mathbf{F}$. Write $A = \{a \mid \{a\} \in \mathrm{F}_0\}$ and $B = \{b \mid \{b\} \in \mathrm{F}_1\}$. Considering the different forms an initial event of a non–null sequence u can take, we see $u \in \mathcal{T}_F(\mathrm{F}_0 \times_F \mathrm{F}_1)$ iff

$$u = \begin{cases} (a,*)u' & \text{for } a \in A \ \& \ {\pi_0}^*(u') \in (\mathrm{F}_0/a) \ \& \ {\pi_1}^*(u') \in \mathrm{F}_1 \text{ or} \\ (a,b)u' & \text{for } a \in A \ \& \ b \in B \ \& \ {\pi_0}^*(u') \in (\mathrm{F}_0/a) \ \& \ {\pi_1}^*(u') \in (\mathrm{F}_1/b) \text{ or} \\ (*,b)u' & \text{for } b \in B \ \& \ {\pi_0}^*(u') \in \mathrm{F}_0 \ \& \ {\pi_1}^*(u') \in (\mathrm{F}_1/b). \end{cases}$$

Hence $\mathcal{T}_F(\mathrm{F}_0 \times_F \mathrm{F}_1) =$

$$\begin{aligned} &\textstyle\bigcup_{a \in A}(a,*)^\frown \mathcal{T}_F(\mathrm{F}_0/a \times_F \mathrm{F}_1) \cup \\ &\textstyle\bigcup_{(a,b) \in A \times B}(a,b)^\frown \mathcal{T}_F(\mathrm{F}_0/a \times_F \mathrm{F}_1/b) \cup \bigcup_{b \in B}(*,b)^\frown \mathcal{T}_F(\mathrm{F}_0 \times_F \mathrm{F}_1/b) \\ &\cong \\ &\textstyle\Sigma_{a \in A}(a,*)^\frown \mathcal{T}_F(\mathrm{F}_0/a) \times_T \mathcal{T}_F(\mathrm{F}_1) + \\ &\textstyle\Sigma_{(a,b) \in A \times B}(a,b)^\frown \mathcal{T}_F(\mathrm{F}_0/a) \times_T \mathcal{T}_F(\mathrm{F}_1/b) + \Sigma_{b \in B}(*,b)^\frown \mathcal{T}_F(\mathrm{F}_0) \times_T \mathcal{T}_F(\mathrm{F}_1/b). \end{aligned}$$

This is the product of $\mathcal{T}_F(\mathrm{F}_0)$ and $\mathcal{T}_F(\mathrm{F}_1)$ in $\mathbf{T}$.

Assume now that S and T are trees. Take $\mathrm{F}_0 = \mathcal{LP}r(\mathrm{S})$ and $\mathrm{F}_1 = \mathcal{LP}r(\mathrm{T})$. Then $\mathrm{S} \cong \mathcal{T}_F(\mathrm{F}_0)$ and $\mathrm{T} \cong \mathcal{T}_F(\mathrm{F}_1)$. Writing $S_a = \mathbf{Tr}\ (\mathrm{F}_0/a)$ and $T_b = \mathbf{Tr}\ (\mathrm{F}_1/b)$ we obtain the result. ∎

Restricting the events of the product in accord with *e.g.* the synchronisation algebra for CCS we obtain the recursive characterisation of the parallel composition of synchronisation trees that Milner uses in [M1]. In this way we obtain a formal translation between the noninterleaving semantics using event structures, their families, prime event structures and the equivalent domains and the interleaving semantics in [M1] all of which factor through a semantics in terms of synchronisation trees. (See [W3] for more on trees, synchronisation trees and semantics using them.)

3. PETRI NETS AND EVENT STRUCTURES.

It is shown how Petri nets also possess morphisms "extending" those on event structures. The morphisms preserve net behaviour (unlike those in [Br]) and can be viewed as special kinds of homomorphisms on nets viewed as algebras. The definition of morphism generalises the process morphisms in [GR]. Safe Petri nets are related to a full subcategory of prime event structures via a coreflection, and it is in this sense that their morphisms extend those on event structures. The coreflection uses the idea of unfolding a net to a net of occurrences. The categorical constructions of product and coproduct of safe nets are closely related to constructions in common use in net theory for modelling parallel compositions and nondeterministic sums.

3.1. Morphisms on Petri nets.

In [W5,6] it is proposed that we view Petri nets as kinds of two–sorted algebras over multisets, and that a useful definition of morphism on nets, appropriate to synchronised communication results by taking a restricted kind of homomorphism between nets viewed as algebras. This notion of morphism is markedly different from the standard kind defined in [Br] which do not respect the behaviour of nets. In this section we give a brief introduction to the new kind of morphisms, and refer the reader to [W4,5,6] for more details. (I do not hold with all the axioms generally placed on Petri nets, which I regard as too restrictive, so the reader is warned to expect some differences in some definitions. For example, I shall allow a Petri net to consist of a single marked condition, disallowed according to the standard definition.)

3.1.1 Definition. A *Petri net* is a 4–tuple (B, E, F, M_0) where
B is a non–null set of *conditions*,
E is a disjoint set of *events*,
F is a multiset of $(B \times E) \cup (E \times B)$, called the *causal dependency* relation,
M_0 is a non–null multiset of conditions, called the *initial marking*,
which satisfies the restrictions:

(i) $\forall e \in E \exists b \in B.\ F_{b,e} > 0$ and $\forall e \in E \exists b \in B.\ F_{e,b} > 0$ and

(ii) $\forall b \in B.\ [M_{0b} \neq 0$ or $(\exists e \in E.\ F_{e,b} \neq 0)$ or $(\exists e \in E.\ F_{b,e} \neq 0)]$.

Thus we insist that each event causally depends on at least one condition and has at least one condition which is causally dependent on it. It is insisted that nets have no *isolated* conditions (*i.e.* that a condition is either marked initially or the pre or post condition of some event) in order to give a simple treatment of morphisms, in which the multirelations never have infinite multiplicities. This restriction is no handicap because, according to the dynamic behaviour of nets, an isolated condition can never hold.

Remark. In more recent work I have found it useful to impose even less restrictions on the definition of a Petri net so as to allow the emptynet, useful for defining nets recursively, and perhaps even the net consisting of a single isolated event, for the purpose of labelling events of a net within the category of nets. In this set–up the only axiom is (ii) above, and initial markings may be empty. This work is still a little experimental, and so here we shall assume the more restrictive definition of a Petri net given above.

Now we make precise the sense in which Petri nets can be identified with special algebras.

3.1.2 Proposition. *A Petri net (B, E, F, M_0) determines a 2–sorted algebra over multisets: It has sorts multisets of conditions μB and multisets of events μE, with operations a constant multiset M_0 over B and two unary operations ${}^\bullet(\ )$ a multirelation from E to B with matrix $(F_{b,e})_{b \in B, e \in E}$, and $(\ )^\bullet$ a multirelation from E to B, with matrix $(F_{e,b})_{b \in B, e \in E}$.*

We describe the "token game" on Petri nets—it differs from some others in that we do not play the token game by firing only one event at a time but allow instead transitions in which finite multisets of events fire.

Let $\mathrm{N} = (B, E, F, M_0)$ be a Petri net.

A *marking* M is a multiset of conditions, *i.e.* $M \in \mu B$.

Let M, M' be markings. Let A be a finite multiset of events. Define

$$M \xrightarrow{A} M' \Leftrightarrow {}^\bullet A \leq M \ \ \& \ \ M' = M - {}^\bullet A + A^\bullet.$$

This gives the *transition relation* between markings.
(This definition has used multiset sum $+$, difference $-$, and multiset subset so $X \leq Y$ iff each multiplicity in X is less than the corresponding multiplicity in Y.)
The transition $M \xrightarrow{A} M'$ means that the finite multiset of events A can occur concurrently from the marking M to yield the marking M'. When we wish to stress the net N in which the transition $M \xrightarrow{A} M'$ occurs we write

$$\mathrm{N} : M \xrightarrow{A} M'.$$

A *reachable marking* of N is a marking M for which

$$M_0 \xrightarrow{A_0} M_1 \xrightarrow{A_1} \cdots \xrightarrow{A_{n-1}} M_n = M$$

for some markings and finite multisets of events.

The reason for only allowing finite multisets of events to occur as transitions is in order that the occurrence of an event only depends on a finite set of event occurrences, and so to tie–in nicely with the finiteness properties of configurations event structures.

Our morphisms on Petri nets are a restricted kind of homomorphism between algebras in which the multirelation between events is assumed to be a partial function.

3.1.3 Definition. Let $\mathrm{N} = (B, E, F, M)$ and $\mathrm{N}' = (B', E', F', M')$ be nets. A *morphism* from N to N′ is a pair of (η, β) consisting of a partial function η from E to E' and a multirelation β from B to B' such that

$$\beta \cdot M = M' \;\;\&\;\; \forall A \in \mu E.\; {}^\bullet(\eta \cdot A) = \beta \cdot ({}^\bullet A) \;\;\&\;\; (\eta \cdot A)^\bullet = \beta \cdot (A^\bullet).$$

(We use *e.g.* $\beta \cdot A$ to stand for the application of the multirelation β to the multiset A.)

Morphisms of nets preserve initial markings and the environments of events. As a consequence they respect the behaviour of nets in the sense of the two following results. The first lemma invokes the proviso that a particular application of a multirelation to a multiset should converge. Remember multirelations and multisets are not necessarily finite so, in general, such an application can lead to infinite sums. (A more complete treatment of multisets can be found in [W6].)

3.1.4 Lemma. *Let $(\eta, \beta) : N \to N'$ be a morphism of Petri nets. Then, provided $\beta \cdot M$ is convergent,*

$$N : M \xrightarrow{A} M' \Rightarrow N' : \beta \cdot M \xrightarrow{\eta \cdot A} \beta \cdot M'.$$

Proof. Directly from the definition we see that β preserves the initial marking. The remaining fact is proved using simple facts about multisets and multirelations. Assume $\mathrm{N} : M \xrightarrow{A} M'$. Then ${}^\bullet A \leq M$, so ${}^\bullet\eta \cdot A = \beta \cdot ({}^\bullet A) \leq \beta M$, and

$$M' = M - {}^\bullet A + A^\bullet.$$

Now applying β, assuming the application $\beta \cdot M$ converges,

$$\begin{aligned} \beta \cdot M' &= \beta \cdot (M - {}^\bullet A + A^\bullet) \\ &= \beta \cdot M - \beta \cdot ({}^\bullet A) + \beta \cdot (A^\bullet) && \text{by linearity} \\ &= \beta \cdot M - {}^\bullet(\eta \cdot A) + (\eta \cdot A)^\bullet && \text{by the defn. of morphism.} \end{aligned}$$

But these facts express that $\mathrm{N}' : \beta \cdot M \xrightarrow{\eta \cdot A} \beta \cdot M'$. ∎

3.1.5 Theorem.
Let $(\eta, \beta) : N \to N'$ be a morphism of Petri nets. Then β preserves the initial marking and preserves reachable markings i.e. if M is a reachable marking of N then $\beta \cdot M$ is a reachable marking of N'. Further, if $M \xrightarrow{A} M'$ and M is reachable in N then $\beta \cdot M \xrightarrow{\eta \cdot A} \beta \cdot M'$ in N'.

Proof. By repeated application of the lemma above. ∎

For safe nets a condition only holds or fails to hold and an event either occurs or does not occur; they do not happen with multiplicities.

3.1.6 Definition. Say a Petri net $\mathrm{N} = (B, E, F, M_0)$ is *safe* iff each multiplicity is at most 1 in F and M, for any reachable markings M. For safe nets we write xFy instead of $F_{x,y} = 1$.

3.1.7 Proposition. *Let $N = (B, E, F, M_0)$ be a safe net.*

Let M be a reachable marking. Let M' be a marking of N and A a finite multiset of its events. If $M \xrightarrow{A} M'$ then M, M' and $A, {}^\bullet A, A^\bullet$ are sets. Further, $M \xrightarrow{A} M'$ iff M, A, M' are sets and

$$(\forall e \in A.\, {}^\bullet e \subseteq M) \;\;\&\;\; (\forall e, e' \in A.\; e \neq e' \Rightarrow {}^\bullet e \cap {}^\bullet e' = \emptyset) \;\;\&\;\; M' = (M \setminus {}^\bullet A) \cup A^\bullet.$$

For a safe net N, an event e is said to have *concession* at a reachable marking M if ${}^\bullet e \subseteq M$. If two events e and e' have concession at a reachable marking M and share a common precondition, so ${}^\bullet e \cap {}^\bullet e' \neq \emptyset$, the events e, e' are said to be in *conflict* at M because if one occurs at M then the other does not. On the other hand, if $M \xrightarrow{A} M'$ the events in A are said to occur *concurrently.*

When nets are safe, just as their behaviour can be described using sets and relations instead of multisets and multirelations, so can morphisms be characterised in a more elementary, if less brief, manner in that, for example, in the proposition below $\beta^\bullet e$ is now the image, as a set, of the set ${}^\bullet e$ under the relation β. We use β^{op} to stand for the opposite relation to β, *i.e.* $x\beta^{op}y$ iff $y\beta x$.

3.1.8 Proposition. *Let $N_0 = (B_0, E_0, F_0, M_0)$ and $N_1 = (B_1, E_1, F_1, M_1)$ be safe nets. A pair (η, β) is a morphism $N_0 \to N_1$ iff η is a partial function from E_0 to E_1, and β is a relation between B_0 and B_1 such that:*

(i) $\beta M_0 = M_1$ *and* β^{op} *restricts to a total function* $M_1 \to M_0$,

(ii) *If* $\eta(e_0) = e_1$ *then*

$$\beta^\bullet e_0 = {}^\bullet e_1 \text{ and } \beta^{op} \text{ restricts to a total function } {}^\bullet e_1 \to {}^\bullet e_0 \quad \text{and}$$
$$\beta e_0{}^\bullet = e_1{}^\bullet \text{ and } \beta^{op} \text{ restricts to a total function } e_1{}^\bullet \to e_0{}^\bullet,$$

(iii) *If* $\eta(e_0)$ *is undefined then* $\beta^\bullet e_0 = \emptyset$ and $\beta e_0{}^\bullet = \emptyset$.

Proof. See [W6] for details. The proposition makes use of the following observation relating set application to multiset application: If $\beta : X \to Y$ is a relation such that the opposite relation $\beta^{op} : Y \to X$ is a partial function then the multirelation application $\beta(X)$ of β, regarded as a multirelation, to X, regarded as a multiset, is equal to the set image βX. The argument also uses the fact that we insist there are no isolated conditions; it is only because of this that multirelations on safe nets can be represented as relations. ∎

In proofs we sometimes find it easier to reason about morphisms between safe nets using the following proposition. It characterises such morphisms in terms of how they behave between initial markings and in the neighbourhood of events and conditions.

3.1.9 Proposition. *Let $N = (B_i, E_i, F_i, M_i)$ be safe nets for $i = 0, 1$. Then $(\eta, \beta) : N_0 \to N_1$ is a morphism of nets iff η is a partial function from E_0 to E_1 and β is a relation from B_0 to B_1 which satisfy*

(i) $\beta M_0 \subseteq M_1$ *and* $\forall b_1 \in M_1 \exists! b_0 \in M_0.\; b_0 \beta b_1$, *and*

(ii) *if* $\eta(e_0) = e_1$ *then for* $b_1 \in B_1$

$$b_1 F_1 e_1 \Rightarrow \exists! b_0.\; (b_0 F_0 e_0 \;\;\&\;\; b_0 \beta b_1) \text{ and}$$
$$e_1 F_1 b_1 \Rightarrow \exists! b_0.\; (e_0 F_0 b_0 \;\;\&\;\; b_0 \beta b_1), \text{ and}$$

(iii) *if* $b_0 \beta b_1$ *then for* $e_0 \in E_0$

$$e_0 F_0 b_0 \Rightarrow \exists e_1.\ (e_1 F_1 b_1 \ \&\ \eta(e_0) = e_1) \text{ and}$$
$$b_0 F_0 e_0 \Rightarrow \exists e_1.\ (b_1 F_1 e_1 \ \&\ \eta(e_0) = e_1).$$

Proof. We use the characterisation of morphisms between safe nets given in the proposition above.

Suppose (η, β) consists of a partial function η on events and β is a relation between conditions which satisfy (i), (ii) and (iii) above. The condition (i) says $\beta M_0 \subseteq M_1$ and that β^{op} restricts to a total function $M_1 \to M_0$. Hence $\beta M_0 = M_1$. Assume $\eta(e_0) = e_1$. By (iii), $\beta^\bullet e_0 \subseteq {}^\bullet e_1$ and $\beta e_0{}^\bullet \subseteq e_1{}^\bullet$. By (ii), β^{op} restricts to a total function ${}^\bullet e_1 \to {}^\bullet e_0$ and $e_1{}^\bullet \to e_0{}^\bullet$. Hence $\beta^\bullet e_0 = {}^\bullet e_1$ and $\beta e_0{}^\bullet = e_1{}^\bullet$. By (iii), if $\eta(e_0)$ is undefined then $\beta^\bullet e_0 = \beta e_0{}^\bullet = \emptyset$. Thus together conditions (i), (ii) and (iii) imply that $(\eta, \beta) : \mathrm{N}_0 \to \mathrm{N}_1$ is a morphism.

Conversely, suppose $(\eta, \beta) : \mathrm{N}_0 \to \mathrm{N}_1$ is a morphism. Then (i) above follows as $\beta M_0 = M_1$ as multisets. Condition (ii) expresses the fact that β^{op} restricts to total functions ${}^\bullet e_1 \to {}^\bullet e_0$ and $e_1{}^\bullet \to e_0{}^\bullet$. Finally, (iii) follows because $\beta({}^\bullet e_0) = {}^\bullet(\eta(e_0))$ and $\beta(e_0{}^\bullet) = (\eta(e_0))^\bullet$, and so if $b_0 \beta b_1$ and $e_0 F_0 b_0$ (or $b_0 F_0 e_0$) we must have $\eta(e_0)$ is defined with $\eta(e_0) F_1 b_1$ (or $b_1 F_1 \eta(e_0)$). ∎

The main category which will concern us is that of safe nets.

3.1.10 Definition. Define **Net** to be the category of safe nets with net morphisms composed coordinatewise as functions and relations.

A remark on notation: sometimes it is easier to write partial functions as relations, writing $e_0 \eta e_1$ for instance instead of $\eta(e_0) = e_1$, and in this section we shall sometimes follow this practice when working with morphisms.

In future we shall have cause to use some rather special morphisms, subnets and foldings.

3.1.11 Definition. Let $(\eta, \beta) : \mathrm{N}_0 \to \mathrm{N}_1$ be a morphism of nets. When (η, β) consists of relations η and β which are the restrictions of the inclusion relations, *i.e.* $e_0 \eta e_1 \Leftrightarrow e_0 = e_1$ and $b_0 \beta b_1 \Leftrightarrow b_0 = b_1$, we say the net N_0 is a *subnet* of N_1. When η and β are total functions we say the morphism (η, β) is a *folding*.

Notice the subnet relation is a partial order. Our definition of subnet is a little different form some others because it involves initial markings, while our definition of folding is more restrictive than that generally proposed.

3.2. Categorical constructions on nets.

Morphisms on nets give rise to intuitive categorical constructions and the story is very similar in theme to that of event structures. The category of all nets has products but not necessarily coproducts though the latter exist in the smaller category of safe nets. For simplicity here we only present the both constructions for safe nets and refer the interested reader to [W6] for more details.

3.2.1 Definition. Let $\mathrm{N}_0 = (B_0, E_0, F_0, M_0)$ and $\mathrm{N}_0 = (B_1, E_1, F_1, M_1)$ be safe nets. Define their *product* to be the net consisting of events

$$E = E_0 \times_{\cdot} E_1,$$

associated with projections $\pi_0 : E \to_* E_0$ and $\pi_1 E \to_* E_1$, conditions

$$B = \{(0,b) \mid b \in B_0\} \cup \{(1,b) \mid b \in B_1\},$$

a disjoint union associated with relations $\rho_0 \subseteq B \times B_0$ and $\rho_1 \subseteq B \times B_1$ given by $b' \rho_k b \Leftrightarrow b' = (k,b)$, for $k = 0,1$, initial marking

$$M = \{(0,b) \mid b \in M_0\} \cup \{(1,b) \mid b \in M_1\},$$

and relation F given by

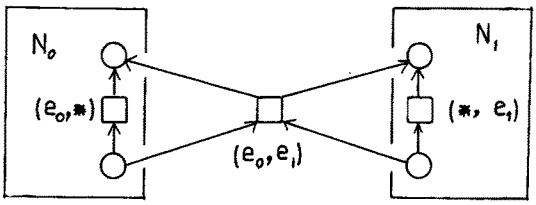
$$\begin{aligned} eFb \Leftrightarrow\ & (\exists e_0 \in E_0, b_0 \in B_0.\ e\pi_0 e_0 \ \&\ b\rho_0 b_0 \ \&\ e_0 F_0 b_0) \\ \text{or}\ & (\exists e_1 \in E_1, b_1 \in B_1.\ e\pi_1 e_1 \ \&\ b\rho_1 b_1 \ \&\ e_1 F_1 b_1) \end{aligned}$$

$$\begin{aligned} bFe \Leftrightarrow\ & (\exists e_0 \in E_0, b_0 \in B_0.\ e\pi_0 e_0 \ \&\ b\rho_0 b_0 \ \&\ b_0 F_0 e_0) \\ \text{or}\ & (\exists e_1 \in E_1, b_1 \in B_1.\ e\pi_1 e_1 \ \&\ b\rho_1 b_1 \ \&\ b_1 F_1 e_1). \end{aligned}$$

Define *projection* morphisms of nets:

$$\begin{aligned} \Pi_0 &= (\pi_0, \rho_0) : \mathrm{N}_0 \times \mathrm{N}_1 \to \mathrm{N}_0 \\ \Pi_1 &= (\pi_1, \rho_1) : \mathrm{N}_0 \times \mathrm{N}_1 \to \mathrm{N}_1. \end{aligned}$$

The product construction can be summarised in a simple picture. Disjoint copies of the two nets N_0 and N_1 are juxtaposed and extra events of synchronisation of the form (e_0, e_1) are adjoined, for e_0 an event of N_0 and e_1 an event of N_1; an extra event (e_0, e_1) has as preconditions those of its components ${}^\bullet e_0 \cup {}^\bullet e_1$ and similarly postconditions $e_0{}^\bullet \cup e_1{}^\bullet$.



The product on nets is closely related to various forms of parallel composition which have been defined on nets to model synchronised communication—for an early example see [LC].

3.2.2 Theorem. *The above construction* $N_0 \times N_1$, Π_0, Π_1 *is a product in* **Net**, *the category of nets.*

Proof. See [W6]. ∎

3.2.3 Definition. Let $\mathrm{N}_0 = (B_0, E_0, F_0, M_0)$ and $\mathrm{N}_1 = (B_1, E_1, F_1, M_1)$ be safe nets. Their *sum* is the net (B, E, F, M) with events $E = \{(0,e) \mid e \in E_0\} \cup \{(1,e) \mid e \in E_1\}$, a disjoint union associated with the obvious injections $\iota_0 : E_0 \to E_0 + E_1$ and $\iota_1 : E_1 \to E_0 + E_1$, conditions

$$B = \{(b_0, *) \mid b_0 \in B_0 \setminus M_0\} \cup \{(*, b_1) \mid b_1 \in B_1 \setminus M_1\} \cup \{(b_0, b_1) \mid b_0 \in M_0 \ \&\ b_1 \in M_1\},$$

associated with injections

$$\begin{aligned} b_0 in_0 b &\Leftrightarrow \exists b_1 \in B_1 \cup \{*\}.\ b = (b_0, b_1), \\ b_1 in_1 b &\Leftrightarrow \exists b_0 \in B_0 \cup \{*\}.\ b = (b_0, b_1), \end{aligned}$$

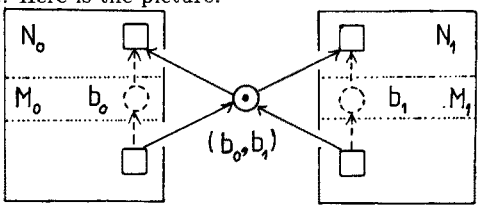
initial marking $M = M_0 \times M_1$, and relation F given by

$$\begin{aligned} eFb \Leftrightarrow\ & (\exists e_0 \in E_0, b_0 \in B_0.\ e_0\iota_0 e\ \ \&\ \ b_0 in_0 b\ \ \&\ \ e_0 F_0 b_0) \\ & \text{or}\ \ (\exists e_1 \in E_1, b_1 \in B_1.\ e_0\iota_1 e\ \ \&\ \ b_1 in_1 b\ \ \&\ \ e_1 F_1 b_1) \\ bFe \Leftrightarrow\ & (\exists e_0 \in E_0, b_0 \in B_0.\ e_0\iota_0 e\ \ \&\ \ b_0 in_0 b\ \ \&\ \ b_0 F_0 e_0) \\ & \text{or}\ \ (\exists e_1 \in E_1, b_1 \in B_1.\ e_0\iota_1 e\ \ \&\ \ b_1 in_1 b\ \ \&\ \ b_1 F_1 e_1). \end{aligned}$$

Define *injection* morphisms of nets:

$$\begin{aligned} I_0 &= (\iota_0, in_0) : \mathrm{N}_0 \to \mathrm{N}_0 + \mathrm{N}_1 \\ I_1 &= (\iota_1, in_1) : \mathrm{N}_1 \to \mathrm{N}_0 + \mathrm{N}_1. \end{aligned}$$

The coproduct construction can be summarised in a simple picture. The two nets N_0 and N_1 are laid side by side and then a little surgery is performed on their initial markings. For each pair of conditions b_0 in the initial marking of N_0 and b_1 in the initial marking of N_1 a new condition (b_0, b_1) is created and made to have the same pre and post events as b_0 and b_1 together. The conditions in the original initial markings are removed and replaced by a new initial marking consisting of these newly created conditions. Here is the picture:



3.2.4 Theorem. *The above construction* $N_0 + N_1$, I_0, I_1 *is a coproduct in the category* **Net**.

Proof. See [W6]. ∎

3.3. Occurrence nets and unfolding.

Nets are rather complex objects with an intricate behaviour which so far has been expressed in a dynamic way. We would like to know when two nets have essentially the same behaviour. In [NPW] and [W] we proposed a more "static" representation of their behaviour as a certain kind of net, a net of condition and event occurrences. This gave a generalisation of the familiar unfolding of a state–transition system to a tree [W2]. The nets of occurrences we called occurrence nets—a name I will stick with here. (Note here occurrence nets may contain forwards conflict.) The ideas here can be viewed as extending those in [Pe], where Petri proposes that the behaviour of a net be identified with the those causal nets which represent its "processes". Instead of a set of causal nets we represent the behaviour of a safe net by a single net of occurrences.

3.3.1 Definition. An *occurrence net* is a safe net (B, E, F, M) for which the following restrictions are satisfied:

(i) $b \in M \Leftrightarrow {}^\bullet b = \emptyset$, so the initial marking is identified with the set of conditions which are not preceded by any events in the F–relation,

(ii) $\forall b \in B.|{}^\bullet b| \leq 1$, so a condition can be caused to hold through the occurrence of at most one event,

(iii) F^+ is irreflexive and $\forall e \in E.\ \{e' \mid e' F^* e\}$ is finite, so we ban repetitions of the same event and insist the occurrence of an event can only depend on the occurrence of a finite number of events,

(iv) $\#$ is irreflexive where

$$e\#_m e' \Leftrightarrow_{def} e \in E \ \& \ e' \in E \ \& \ {}^\bullet e \cap {}^\bullet e' \neq \emptyset \text{ and}$$
$$x\#x' \Leftrightarrow_{def} \exists e, e' \in E. e\#_m e' \ \& \ eF^* x \ \& \ e'F^* x'.$$

In this way we eliminate those events which cannot possibly occur because they depend on the previous occurrence of conflicting events.

In an occurrence net we call the relation $\#_m$ defined above the *immediate conflict relation* and $\#$ the *conflict relation*. We define the *concurrency relation*, *co*, between pairs $x, y \in B \cup E$ by:

$$x \ co \ y \Leftrightarrow_{def} \neg(xF^+y \ \text{ or } \ yF^+x \ \text{ or } \ x\#y).$$

It is useful to generalise the *co*–relation to subsets, and not just pairs, of conditions. Intuitively we say a subset S of conditions of an occurrence net is *concurrent* if it possible for all the conditions in S to hold at some reachable marking. Similarly we say a finite subset of events is *concurrent* if they can occur concurrently from some reachable marking. For an occurrence net (B, E, F, M) these notions can be expressed simply.

3.3.2 Definition. For $S \subseteq B \cup E$ define

$$Co\,S \Leftrightarrow (\forall s, t \in S.\ s \ co \ t) \ \& \ \{e \in E \mid \exists s \in S.\ eF^* s\} \text{ is finite.}$$

Clearly $s \ co \ t$ iff $Co\{s, t\}$, for conditions and events s, t. The extra restriction simply ensures that together the elements in S only depend on a finite number of event occurrences. Obviously if T is a subset of events and $Co\,T$ then T must itself be a finite set.

3.3.3 Proposition. *Let $N = (B, E, F, M)$ be an occurrence net.*

Every event of N has concession at some reachable marking and every condition of N holds at some reachable marking.

Let e, e' be two events of N. Let b, b' be two conditions of N.

The relations $\#_m \subseteq E^2$ and $\# \subseteq (B \cup E)^2$ are binary, symmetric, irreflexive relations. The relation of conflict $x \# x'$ holds iff there is a reachable marking M at which events eF^*x and $e'F^*x'$ have concession and are in immediate conflict $e\#_m e'$.

The relation co is a binary, symmetric, reflexive relation between conditions and events of N. We have $b \ co \ b'$ iff there is a reachable marking of N at which b and b' both hold. We have $e \ co \ e'$ iff there is a reachable marking at which e and e' can occur concurrently.

Let S be a subset of conditions and T a subset of events. We have $Co\,S$ iff there is some reachable marking M for which $S \subseteq M$. We have $Co\,T$ iff there are reachable markings M, M' for which $M \xrightarrow{T} M'$.

3.3.4 Definition. Write **Occ** for the category of occurrence nets with net morphisms.

We observe two properties of morphisms between occurrence nets which we shall use later.

3.3.5 Lemma. *Let $N_0 = (B_0, E_0, F_0, M_0)$ and $N_1 = (B_1, E_1, F_1, M_1)$ be occurrence nets. Let $(\eta, \beta) : N_0 \to N_1$ be a morphism. Then*

(i) $b_0 \beta b_1 \ \& \ b_1 \in M_1 \Rightarrow b_0 \in M_0$ *and*

(ii) $b_0 \beta b_1 \ \& \ e_1 F_1 b_1 \Rightarrow \exists! e_0.\ \eta(e_0) = e_1 \ \& \ e_0 F_0 b_0.$

Proof.

(i) Assume $b_0 \beta b_1$ & $b_1 \in M_1$. If $b_0 \notin M_0$ then $e_0 F_0 b_0$ for some event e_0, which by the properties of morphisms implies there is some event $e_1 F_1 b_1$—impossible as $b_1 \in M_1$.

(ii) Assume $b_0 \beta b_1$ & $e_1 F_1 b_1$. As (η, β) is a morphism $b_0 \notin M_0$ so there is a unique event e_0 such that $e_0 F_0 b_0$. By the properties of morphisms $\eta(e_0) F_1 b_1$ and so $\eta(e_0) = e_1$, the unique event in ${}^\bullet b_1$. ∎

As a corollary we see that morphisms between occurrence nets reflect F–chains and conflict in the following sense.

3.3.6 Lemma. *Let $N_0 = (B_0, E_0, F_0, M_0)$ and $N_1 = (B_1, E_1, F_1, M_1)$ be occurrence nets. Let $(\eta, \beta) : N_0 \to N_1$ be a morphism.*
(i)If

$$b_1^{(n)} F_1 e_1^{(n)} b_1^{(n-1)} \cdots b_1^{(0)} F_1 e_1^{(0)}$$

is a chain in N_1 and $\eta(e_0^{(0)}) = e_1^{(0)}$, with $e_0^{(0)} \in E_0$, then there is a unique chain

$$b_0^{(n)} F_0 e_0^{(n)} b_0^{(n-1)} \cdots b_0^{(0)} F_0 e_0^{(0)}$$

in N_0 such that $b_0^{(i)} \beta b_1^{(i)}$ and $\eta(e_0^{(i)}) = e_1^{(i)}$ for $0 \leq i \leq n$.
(ii) For events e_0, e_0' of N_0 and e_1, e_1' of N_1 and conditions b_0, b_0' of N_0 and b_1, b_1' of N_1:

$$\eta(e_0) = e_1 \;\&\; \eta(e_0') = e_1' \;\&\; e_1 \# e_1' \Rightarrow e_0 \# e_0',$$
$$b_0 \beta b_1 \;\&\; b_0' \beta b_1 \;\&\; b_1 \# b_1' \Rightarrow b_0 \# b_0'.$$

(iii) For events e_0, e_0' of N_0 and e_1 of N_1 and conditions b_0, b_0' of N_0 and b_1 of N_1:

$$\eta(e_0) = e_1 \;\&\; \eta(e_0') = e_1 \Rightarrow e_0 = e_0' \;\text{ or }\; e_0 \# e_0',$$
$$b_0 \beta b_1 \;\&\; b_0' \beta b_1 \Rightarrow b_0 = b_0' \;\text{ or }\; b_0 \# b_0'.$$

Proof.

(i) The proof proceeds by induction down the chain $b_1^{(n)} F_1 e_1^{(n)} b_1^{(n-1)} \cdots b_1^{(0)} F_1 e_1^{(0)}$ We are given $\eta(e_0^{(0)}) = e_1^{(0)}$. By the properties of morphisms there is a unique condition $b_0^{(0)}$ such that $b_0^{(0)} \beta b_1^{(0)}$. Now by the above lemma there is a unique $e_0^{(1)}$ such that $\eta(e_0^{(1)}) = e_1^{(1)}$. Continuing we obtain the result. Notice, by the lemma, if $b_1^{(n)}$ is marked initially then so is $b_0^{(n)}$.

(ii) Suppose e_1, e_1' are two events in conflict with $\eta(e_0) = e_1$ and $\eta(e_0') = e_1'$. Then this can only arise through two events t_1, t_1' being in immediate conflict with $t_1 F_1^* e_1$ and $t_1' F_1^* e_1'$. So $t_1 \#_m t_1'$ *i.e.* $t_1 \neq t_1'$ with $b_1 F_1 t_1$ and $b_1 F_1 t_1'$ for some condition b_1. But then two simple applications of part (i) yield events t_0, t_0' in immediate conflict (with $\eta t_0 = t_1$ and $\eta t_0' = t_1'$) so that $t_0 F_0^* e_0$ and $t_0' F_0^* e_0'$. This makes $e_0 \# e_0'$. The same argument works to show: If $b_0 \beta b_1$ and $b_0' \beta b_1$ and $b_1 \# b_1'$ then $b_0 \# b_0'$.

(iii) Suppose $\eta(e_0) = \eta(e_0')$ are defined and equal e_1. There is a chain

$$c_n F_1 d_n \cdots d_1 F_1 c_0 F_1 d_0$$

in N_1 with $d_0 = e_1$ and $c_n \in M_1$. Applying part (i) we obtain chains

$$b_n F_0 e_n \cdots e_1 F_0 b_0 F_0 e_0 \text{ and}$$
$$b_n' F_0 e_n' \cdots e_1' F_0 b_0' F_0 e_0'.$$

such that $\eta(e_i) = \eta(e_i') = d_i$ and $b_i \beta c_i$ and $b_i' \beta c_i$ for $0 \leq i \leq n$ and ${}^\bullet b_n = {}^\bullet b_n{}' = {}^\bullet c_n = \emptyset$. As f is a morphism, β^{op} is a function when restricted to initial markings so $b_n = b_n'$. We can now show $e_0 (\# \cup 1) e_0'$. Suppose $e_0 \neq e_0'$. Then the chains leading up to these events must differ at some earliest point, giving rise to one of these two situations:

$$e_i = e_1' \ \ \& \ \ b_{i-1} \neq b_{i-1}' \tag{1}$$
$$b_i = b_1' \ \ \& \ \ e_i \neq e_i' \tag{2}$$

for some $i < n$. Case (1) is impossible as β^{op} should be a function restricted to $d_i{}^\bullet \to e_i{}^\bullet$. The remaining case, case (2), implies e_i and e_i' are in immediate conflict so $e_0 \# e_0'$. The same argument shows conditions with the same image must either be equal or in conflict. ∎

There is a natural idea of depth of an element of an occurrence net, useful to prove properties of occurrence nets by induction.

3.3.7 Definition. Let N $= (B, E, F, M)$ be an occurrence net. Inductively define the *depth* of an element $x \in B \cup E$ as follows:
For $b \in M$ take $depth(b) = 0$;
For $e \in E$ take $depth(e) = \max\{depth(b) \mid bFe\} + 1$;
For $b \in B \setminus M$ take $depth(b) = depth(e)$ for that unique e such that eFb.

3.3.8 Proposition. *An occurrence net $N = (B, E, F, M)$ is the least upper bound, with respect to the subnet order, of its subnets $N^{(n)}$ of depth n i.e. Define $N^{(n)} =_{def} (B^{(n)}, E^{(n)}, F^{(n)}, M)$ where*

$$B^{(n)} = \{b \in B \mid depth(b) \leq n\}$$
$$E^{(n)} = \{e \in e \mid depth(e) \leq n\}$$
$$xF^{(n)}y \Leftrightarrow x, y \in B^{(n)} \cup E^{(n)} \ \ \& \ \ xFy.$$

Then $N^{(n)}$ is a subnet of N and $N = \bigcup_{n \in \omega} N^{(n)}$—the coordinatewise union of the nets $N^{(n)}$.

Proof. Clear. ∎

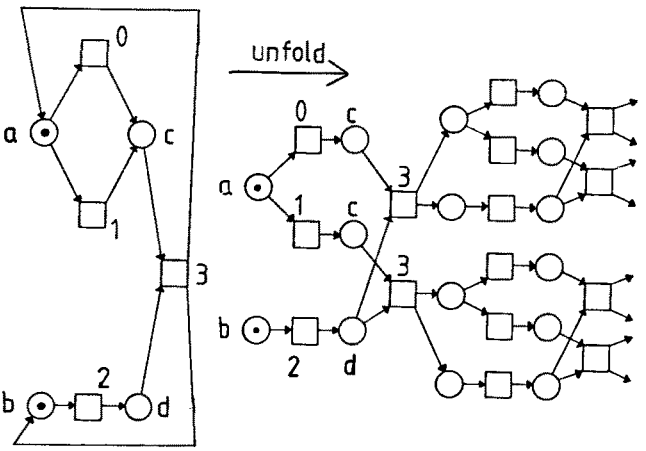
3.3.9 Theorem. *Let $N = (B, E, F, M)$ be a safe net. There is a unique occurrence net $\mathcal{U}N = (B_0, E_0, F_0, M_0)$ with a folding $f = (\eta, \beta) : \mathcal{U}N \to N$ which satisfies:*

$$B_0 = \{(\emptyset, b) \mid b \in M\} \cup \{(\{e_0\}, b) \mid e_0 \in E_0 \ \ \& \ \ b \in B \ \ \& \ \ \eta(e_0) F b\},$$
$$E_0 = \{(S, e) \mid S \subseteq B_0 \ \ \& \ \ Co(S) \ \ \& \ \ e \in E \ \ \& \ \ \beta S = {}^\bullet e\},$$
$$xF_0y \Leftrightarrow \exists w, z.\ y = (w, z) \ \ \& \ \ x \in z,$$
$$M_0 = \{(\emptyset, b) \mid b \in M\},$$
and
$$e_0 \eta e \Leftrightarrow \exists S \subseteq B_0.\ e_0 = (S, e),$$
$$b_0 \beta b \Leftrightarrow b \in M \ \ \& \ \ b_0 = (\emptyset, b) \ \ \text{or} \ \ \exists e_0 \in E_0.\ b_0 = (\{e_0\}, b).$$

Proof. The existence is shown by giving an inductive definition. It is routine, tedious and omitted. The uniqueness follows because every element of an occurrence net has finite depth. ∎

3.3.10 Definition. Let N be a safe net. Define its *occurrence net unfolding*, $\mathcal{U}$(N), to be the unique net and *folding* morphism that satisfy the requirements of theorem 3.3.9 above.

3.3.11 Example. This example illustrates a safe net together with its occurrence net unfolding. The associated folding morphism from the occurrence net unfolding to the original net is indicated by the inscriptions.



A characterising property of the occurrence net unfolding is expressed in the following proposition. Roughly it says every possible occurrence of an event in the original net is matched by a unique event in the unfolding.

3.3.12 Theorem. *Let N be a safe net. The occurrence net unfolding $\mathcal{U}(N)$ and folding $f = (\eta, \beta) : \mathcal{U}(N) \to N$ satisfy*

$$Co\,S \;\;\&\;\; \beta S = {}^\bullet e \Rightarrow \exists! e_0.\; S = {}^\bullet e_0 \;\;\&\;\; \eta(e_0) = e, \qquad (*)$$

where e is an event of N and e_0 is an event and S a subset of conditions of N_0. Further, $\mathcal{U}(N)$ and the folding f are determined to within isomorphism by $(*)$ i.e. if $f_1 : N_1 \to N$ is a folding from an occurrence net N_1 which also satisfies $(*)$ then there is an isomorphism $h : N_1 \cong \mathcal{U}(N)$ such that $f_1 = f\ h$.

Proof. Let N be a safe net. It is follows directly from their definitions that the unfolding $\mathcal{U}(\mathrm{N})$ of a net N and the folding $f : \mathcal{U}(\mathrm{N}) \to \mathrm{N}$ satisfy $(*)$.

To show uniqueness to within isomorphism, assume $f_0 : \mathrm{N}_0 \to \mathrm{N}$ and $f_1 : \mathrm{N}_1 \to \mathrm{N}$ are foldings from occurrence nets $\mathrm{N}_0 = (B_0, E_0, F_0, M_0)$ and $\mathrm{N}_1 = (B_1, E_1, F_1, M_1)$ onto N which both satisfy $(*)$. For an occurrence net $O = (B, E, F, M)$ let $O^{(n)} = (B^{(n)}, E^{(n)}, F^{(n)}, M^{(n)})$ be the subnet of conditions and events of depth $\leq n$, for $n \in \omega$, and write $f_0^{(n)}$ and $f_1^{(n)}$ for the restrictions of the foldings to the subnets $\mathrm{N}_0^{(n)}$ of N_0 and $\mathrm{N}_1^{(n)}$ of N_1, so $f_0^{(n)} : \mathrm{N}_0^{(n)} \to \mathrm{N}$ and $f_1^{(n)} : \mathrm{N}_1^{(n)} \to \mathrm{N}$.

We construct by induction on $n \in \omega$ isomorphisms $h^{(n)} = (\eta^{(n)}, \beta^{(n)}) : \mathrm{N}_0^{(n)} \to \mathrm{N}_1^{(n)}$ which satisfy

$$f_1^{(n)}\ h^{(n)} = f_0^{(n)}.$$

For the basis, define $h^{(0)} = (\emptyset, \beta^{(0)}) : \mathrm{N}_0^{(0)} \to \mathrm{N}_1^{(0)}$ where

$$b_0 \beta^{(0)} b_1 \Leftrightarrow b_0 \in M_0 \;\;\&\;\; b_1 \in M_1 \;\;\&\;\; \beta_0 b_0 = \beta_1 b_1.$$

Certainly, $f_1^{(0)}\ h^{(0)} = f_0^{(0)}$ because β_0, and β_1, restrict to 1–1 correspondences between the initial markings M_0 and M, and M_1 and M respectively, and hence $\beta^{(0)}$ is a 1–1 correspondence $M_0 \to M_1$.

Now for the step in the inductive definition, assume $h^{(n)}$ to be defined so that $h^{(n)}$ is an isomorphism and $f_1^{(n)}\ h^{(n)} = f_0^{(n)}$. Take $h^{(n+1)}$ to be a pair of relations $(\eta^{(n+1)}, \beta^{(n+1)})$ given by:

$$e_0 \eta^{(n+1)} e_1 \text{ if } \eta_0(e_0) = \eta_1(e_1) \ \ \& \ \ \beta^{(n)\bullet} e_0 = {}^\bullet e_1,$$
$$b_0 \beta^{(n+1)} b_1 \text{ if } \beta_0 b_0 = \beta_1 b_1 \ \ \& \ \ \eta^{(n+1)\bullet} b_0 = {}^\bullet b_1.$$

Of course, it needs to be shown that $h^{(n+1)}$ is an isomorphism with $f_1^{(n+1)}\ h^{(n+1)} = f_0^{(n+1)}$.

We first show $\eta^{(n+1)}$ is a 1–1 correspondence between the events of $\mathrm{N}_0^{(n+1)}$ and $\mathrm{N}_1^{(n+1)}$ such that $\eta_0(e_0) = \eta^{(n+1)}\eta_1(e_0)$ for all events e_0 of $\mathrm{N}_0^{(n+1)}$. Suppose e_0 is an event of $\mathrm{N}_0^{(n+1)}$. Then $\eta_0(e_0) = e$ say. As f_0 is a morphism $\beta_0{}^\bullet e_0 = {}^\bullet e$. Because e_0 has depth at most $n+1$, ${}^\bullet e_0$ consists of conditions in $\mathrm{N}_0^{(n)}$. Because $h^{(n)}$ is a morphism $Co(\beta^{(n)\bullet} e_0)$ in $\mathrm{N}_1^{(n)}$ and by commutativity $\beta_1\beta^{(n)}S = \beta_0 S = {}^\bullet e$. By the property $(*)$ of f_1 we see there is a unique event e_1 in N_1 such that ${}^\bullet e_1 = \beta^{(n)\bullet} e_0$ and $\eta_1(e_1) = e$. Thus $e_0 \eta^{(n+1)} e_1$ for some e_1. It is unique because if $e_0 \eta^{(n+1)} e_1, e_1'$ then from the inductive assumption $\eta_1(e_1) = \eta_1(e_1')$, forcing $e_1 = e_1'$ by $(*)$. Similarly, if e_1 is an event of $\mathrm{N}_1^{(n+1)}$ there is a unique event e_0 such that $e_0 \eta^{(n+1)} e_1$. This shows $\eta^{(n+1)}$ is a 1–1 correspondence. Clearly $\eta_0^{(n+1)} = \eta^{(n+1)}\eta_1^{(n+1)}$ from the definition of $\eta^{(n+1)}$.

A similar argument shows $\beta^{(n+1)}$ is a 1–1 correspondence too. Let b_0 be a condition of $\mathrm{N}_0^{(n+1)}$. Then $b_0 \beta b$ for some $b \in B$. The case when ${}^\bullet b_0 = \emptyset$ has been dealt with in the basis of the inductive construction, so we may assume $e_0 F_0 b_0$ for some unique event e_0 in $\mathrm{N}_0^{(n+1)}$. As f_0 is a morphism there is some e with eFb and $\eta_0(e_0) = e$. By the previous argument $\eta^{(n+1)}(e_0) = e_1$ for some event e_1 of $\mathrm{N}_1^{(n+1)}$. As f_1 is a morphism there is a unique condition b_1 of $\mathrm{N}_1^{(n+1)}$ so that $b_1 \beta_1 b$. Now we see from the inductive assumption that $\beta_0 \beta^{(n+1)} b_1$. Moreover such a b_1 is unique. For, suppose $b_0 \beta^{(n+1)} b_1, b_1'$. Then $e_1 F_1 b_1, b_1'$ and $b_1, b_1' \beta_1 b$. But then $b_1 = b_1'$ as f_1 is a morphism. Thus $\beta^{(n+1)} : B_0^{(n+1)} \to B_1^{(n+1)}$ is a total function. Similarly, if b_1 is a condition of $\mathrm{N}_1^{(n+1)}$ there is a unique condition b_0 such that $b_0 \beta^{(n+1)} b_1$. Hence $\beta^{(n+1)}$ is a 1–1 correspondence. Clearly from its definition $\beta_0^{(n+1)} = \beta^{(n+1)}\beta_1^{(n+1)}$.

Now we can define $h = (\bigcup_n \eta^{(n)}, \bigcup_n \beta^{(n)})$ to obtain the required isomorphism $\mathrm{N}_0 \cong \mathrm{N}_1$. ∎

3.3.13 Theorem. *Let N be a safe Petri net. Then the occurrence net unfolding $\mathcal{U}(N)$ and folding f are cofree over N i.e. for any morphism $g : N_1 \to N$ with N_1 an occurrence net there is a unique morphism $h : N_1 \to \mathcal{U}(N)$ such that the following diagram commutes:*

$$\begin{array}{ccc} \mathcal{U}(N) & \xrightarrow{f} & N \\ {\scriptstyle h}\uparrow & \nearrow_{g} & \\ N_1 & & \end{array}$$

*Unfolding extends to a functor $\mathcal{U}$: **Net** $\to$ **Occ** which is a right adjoint to the inclusion functor* **Occ** $\to$ **Net**. *Further, this adjunction is a coreflection: the folding $f : \mathcal{U}(O) \to O$ for each occurrence net O forms a natural isomorphism.*

Proof. Assume $\mathrm{N} = (B, E, F, M)$ is a safe net which has an occurrence net unfolding $\mathcal{U}(\mathrm{N}) = (B_0, E_0, F_0, M_0)$ and folding $f = (\eta_0, \beta_0) : \mathcal{U}(\mathrm{N}) \to \mathrm{N}$. Assume N_1 is an occurrence net of the form $\mathrm{N}_1 = (B_1, E_1, F_1, M_1)$ and that $g = (\eta_1, \beta_1) : \mathrm{N}_1 \to \mathrm{N}$ is a morphism.

It is convenient to first establish necessary and sufficient conditions for there to be a morphism making the above diagram commute, and then later to construct a pair of relations which is unique so the conditions are satisfied.

Let $h = (\eta, \beta)$ consist of a partial function η from E_1 to E_0 and a relation $\beta \subseteq B_1 \times B_0$. We show that $h : \mathrm{N}_1 \to \mathcal{U}(\mathrm{N})$ is a morphism and $g = f\ h$ iff the following conditions are satisfied:

(I) $\eta(e_1) = e_0 \Leftrightarrow \exists e \in E.e_0 = (\beta^\bullet e_1, e)\ \ \&\ \ \eta_1(e_1) = e$ for all e_0, e_1,

(II) $b_1 \beta b_0 \Leftrightarrow \exists b \in B.b_0 = (\eta^\bullet b_1, b)\ \ \&\ \ b_1 \beta_1 b$ for all b_0, b_1.

Firstly suppose h is a morphism such that $g = f\ h$. We show that the conditions (I) and (II) must then be satisfied.

(I)

"$\Rightarrow$." Let $\eta(e_1) = e_0$. Then because $g = fh$ we have $\eta_1(e_1) = e$ for some e and S such that $e_0 = (S, e)$. However because h is a morphism we must have $S = \beta^\bullet e_1$, as required.

"$\Leftarrow$." Suppose $e_0 = (\beta^\bullet e_1, e)$ and $\eta_1(e_1) = e$ for some $e \in E$. We first show $e_0 \in E_0$. Because h is a morphism $Co(\beta^\bullet e_1)$ and by the commutativity $\beta_0(\beta^\bullet e_1) = \beta_1{}^\bullet e_1 = {}^\bullet e$. Thus $e_0 = (\beta^\bullet e_1, e) \in E_0$, and $\eta_0(e_0) = e$. By commutativity $\eta(e_1) = e_0'$ and $\eta_0(e_0') = e$ for some $e_0' \in E_0$. As h is a morphism ${}^\bullet e_0' = \beta^\bullet e_1$. Thus $e_0' = (\beta^\bullet e_1, e) = e_0$. Hence $\eta(e_1) = e_0$.

(II)

"$\Rightarrow$." Suppose $b_1 \beta b_0$. Then by the commutativity, $b_1 \beta_1 b$ and $b_0 \beta_0 b$ for some $b \in B$. If ${}^\bullet b_1 = \emptyset$ then $b_1 \in M_1$ so $b_0 \in M_0$ and ${}^\bullet b_0 = \emptyset$. Otherwise ${}^\bullet b_1 = \{e_1\}$, for some event e_1, so as h is a morphism $\eta(e_1) = e_0$, for some e_0, and ${}^\bullet b_0 = \{e_0\}$. In either case $b_0 = (\eta^\bullet b_1, b)$.

"$\Leftarrow$." Suppose $b_0 = (\eta^\bullet b_1, b)$ and $b_1 \beta_1 b$ for some $b \in B$. Either $b_1 \in M_1$ or ${}^\bullet b_1 \neq \emptyset$. Assume $b_1 \in M_1$. Then $b_0 = (\emptyset, b) \in M_0$. As h is a morphism there is some $b_1' \in M_1$ such that $b_1' \beta b_0$. As g is a morphism $b_1 = b_1'$ so $b_1 \beta b_0$ as required. Now assume the other case, that ${}^\bullet b_1 \neq \emptyset$ and let e_1 be the unique event such that $e_1 F_1 b_1$. As g is a morphism $\eta_1(e_1) = e$ and eFb, for some e. By the commutativity $\eta(e_1) = e_0$, for some e_0. Thus $b_0 = (\{e_0\}, b) \in B_0$ and $e_0 F_0 b_0$. As h is a morphism there is some b_1' so that $b_1' \beta b_0$ and $e_1 F_1 b_1'$. Therefore by the commutativity $b_1' \beta_1 b$. Thus both $e_1 F_1 b_1'$ with $b_1' \beta_1 b$ and $e_1 F_1 b_1$ with $b_1 \beta_1 b$. But, then as g is a morphism, $b_1 = b_1'$. Therefore $b_1 \beta b_0$ as required.

Thus we have shown that if $h : \mathrm{N}_1 \to \mathcal{U}(\mathrm{N})$ is a morphism such that $g = fh$ then the conditions (I) and (II) are satisfied. Now we show the converse, that the conditions (I) and (II) ensure that h is a morphism such that $g = fh$.

Suppose the conditions (I) and (II) are satisfied. First we show h is a morphism $h : \mathrm{N}_1 \to \mathcal{U}(\mathrm{N})$. We check that the conditions (i), (ii) and (iii) of proposition 3.1.9 hold:

(i) Clearly, by (II), if $b_1 \beta b_0\ \ \&\ \ b_1 \in M_1$ then $b_0 = (\emptyset, b) \in M_0$. Also, if we assume $b_1, b_1' \in M_1$ and $b_1 \beta b_0$ and $b_1' \beta b_0$ then, by (II), $b_1 \beta_1 b$ and $b_1' \beta_1 b$ for some b which implies $b_1 = b_1'$, as g is a morphism.

(ii) Suppose $\eta(e_1) = e_0$.

Assume $e_0 F_0 b_0$. Then, by (I), $e_0 = (\beta^\bullet e_1, e)\ \ \&\ \ \eta_1(e_1) = e$ for some $e \in E$. From the definition of the unfolding, $eFb\ \ \&\ \ b_0 = (\{e_0\}, b)$ for some $b \in B$. As g is a morphism $e_1 F_1 b_1$ and $b_1 \beta_1 b$ for some unique condition $b_1 \in B_1$. Therefore, by (II), b_1 is the unique condition such that $b_1 \beta b_0$ and $e_1 F_1 b_1$, as required.

Assume $b_0 F_0 e_0$. Then, by (I), $e_0 = (\beta^\bullet e_1, e)\ \ \&\ \ \eta_1(e_1) = e$ for some $e \in E$. By the properties of the unfolding, $b_0 \in \beta^\bullet e_1$. Thus $b_1 \beta b_0\ \ \&\ \ b_1 F_1 e_1$ for some $b_1 \in B_1$. We also need the uniqueness

of b_1. Let $\beta_0(b_0) = b$. Assume $b_1'\beta b_0$ & $b_1'F_1e_1$ for some $b_1' \in B_1$. Then by (ii) $b_1'\beta_1 b$, which combined with $b_1'F_1e_1$ implies $b_1' = b_1$ as g is a morphism. So, as required b_1 is unique so that $b_1\beta b_0$ & $b_1F_1e_1$.

(iii) Suppose $b_1\beta b_0$.

Assume $e_1F_1b_1$. As g is a morphism, $\eta_1(e_1) = e$ and eFb for some $e \in E$. By (II), $b_0 = (\eta^\bullet b_1, b)$ and $b_1\beta_1 b$ for some $b \in B$. By the definition of the unfolding, $e_0F_0b_0$ and $\eta_0(e_0) = e$ for some $e_0 \in E_0$. Thus $\eta^\bullet b_1 = \eta\{e_1\} = \{e_0\}$. Hence $\eta(e_1) = e_0$.

Assume $b_1F_1e_1$ for $e_1 \in E_1$. By (II), $b_0 = (\eta^\bullet b_1, b)$ & $b_1\beta_1 b$ for some $b \in B$. As g is a morphism bFe & $\eta_1(e_1) = e$ for some $e \in E$. Take $e_0 = (\beta^\bullet e_1, e)$. Then, by (I), $\eta(e_1) = e_0$, and clearly $b_0F_0e_0$.

Now by proposition 3.1.9 we can conclude h is a morphism $\mathrm{N}_1 \to \mathcal{U}(\mathrm{N})$. In addition, we require the commutativity $g = f\ h$ *i.e.* $(\eta_1, \beta_1) = (\eta_0, \beta_0)\ (\eta, \beta)$. These follow from (I) and (II) by the following arguments:

Suppose $\eta_0\eta(e_1) = e$. Then $\eta(e_1) = e_0$ and $\eta_0(e_0) = e$ for some $e_0 \in E_0$. By (I)"$\Rightarrow$", $\eta_1(e_1) = e$. Now suppose $\eta_1(e_1) = e$. Take $e_0 = (\beta^\bullet e_1, e)$. Then by (I)"$\Leftarrow$" $\eta(e_1) = e_0$. Therefore $(\eta_0\ \eta)e_1 = \eta_0(e_0) = e$. Combining these results we see $\eta_0\eta = \eta_1$.

Suppose $b_1(\beta_0\beta)b$. Then $b_1\beta b_0$ and $b_0\beta_0 b$ for some $b_0 \in B_0$. By (II)"$\Rightarrow$", this implies $b_1\beta_1 b$. Suppose $b_1\beta_1 b$. Take $b_0 = (\eta^\bullet b_1, b)$. Then by (II)"$\Leftarrow$" $b_0 \in B_0$ and so $b_0\beta_0 b$. Therefore $b_1(\beta_0\ \beta)b$. Combining these results we see $\beta_0\beta = \beta_1$. This completes the proof that $g = f\ h$.

We have completed that part of the proof showing that $h : \mathrm{N}_1 \to \mathcal{U}(\mathrm{N})$ is a morphism and $g = fh$ iff h satisfies (I) and (II). Now it remains to show that such a morphism h exists and moreover is unique.

We show the existence of such an h by giving an inductive definition—see [Acz]. Define $h = (\eta, \beta)$ to consist of the pair of smallest relations $\eta \subseteq E_1 \times E_0$ and $\beta \subseteq B_1 \times B_0$ which satisfy:

$$e_0 = (\beta^\bullet e_1, e) \ \ \& \ \ \eta_1(e_1) = e \Rightarrow \eta(e_1) = e_0 \text{ and}$$
$$b_0 = (\eta^\bullet b_1, b) \ \ \& \ \ b_1\beta_1 b \Rightarrow b_1\beta b_0.$$

This inductive definition provides a least $h = (\eta, \beta)$ which satisfies (I) and (II). (Note the inductive definition has closure ordinal ω because we assume an event depends on only a finite number events.) Thus by our previous work $h : \mathrm{N}_1 \to \mathcal{U}(\mathrm{N})$ is a morphism for which $g = fh$.

The ultimate step in the proof is to show that the h defined inductively above is the unique morphism $h : \mathrm{N}_1 \to \mathcal{U}(\mathrm{N})$ for which $g = fh$. Suppose $h' = (\epsilon, \alpha)$ were another morphism such that $g = fh'$. Then it too would satisfy (I) and (II). Consequently by the definition of h, $\eta \subseteq \epsilon$ and $\beta \subseteq \alpha$. The converse inclusions are established by induction on the depth of the conditions and events of N_1:

Zero Depth: Clearly if $b_1 \in M_1$ and $b_1\alpha b_0$ then, as α satisfies (II), $b_1\beta b_0$ too.

Nonzero Depth: Assume $\epsilon e_1 = e_0$ where $depth(e_1) = n + 1$. As ϵ satisfies (I) we have $e_0 = (\alpha^\bullet e_1, e)$ and $\eta_1(e_1) = e$ for some $e \in E$. Each condition in $\alpha^\bullet e_1$ has strictly less depth than $n + 1$. Thus $\alpha^\bullet e_1 = \beta^\bullet e_1$ so as η satisfies (I) we obtain $\eta(e_1) = e_0$. Assume $b_1\alpha b_0$ where $depth(b_1) = n + 1$. As α satisfies (II), $b_0 = (\epsilon^\bullet b_1, b)$ and $b_1\beta_1 b$. Here the unique event e_1 such that $e_1F_1b_1$ has depth $n + 1$. By the argument just given $e_1\epsilon e_0 \Leftrightarrow \eta(e_1) = e_0$. Because η satisfies (II) we obtain $b_1\beta b_0$.

This induction shows that $\epsilon \subseteq \eta$ and $\alpha \subseteq \beta$ which together with the previously shown converse inclusions yields $h = h'$. We have established the existence and uniqueness of a morphism $h : \mathrm{N}_1 \to \mathcal{U}(\mathrm{N})$ making $g = fh$.

Finally, we conclude that $\mathcal{U}(\mathrm{N}), f$ is cofree over N. The adjunction follows from the cofreeness—see [Mac, p.81]. Clearly each folding $f : \mathcal{U}(O) \to O$, for $O \in \mathbf{Occ}$, is an isomorphism, so the adjunction is a coreflection. ∎

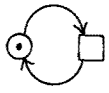
3.3.14 Corollary. *The unfolding operation on safe nets preserves limits; in particular it preserves products. Thus the unfolding of the product (in* **Net***) of two nets $\mathcal{U}(N_0 \times N_1)$ is isomorphic to the product (in* **Occ***) of the unfoldings $\mathcal{U}(N_0) \times_{occ} \mathcal{U}(N_1)$. To within isomorphism, the product of two occurrence nets $N_0 \times_{occ} N_1$ in* **Occ** *is the net $\mathcal{U}(N_0 \times N_1)$. The inclusion functor* **Occ** $\to$ **Net** *preserves colimits and in particular coproducts. The category* **Occ** *has coproducts which coincide with those in* **Net**.

Proof. As remarked (see [Mac]) right adjoints preserve limits and left adjoints preserve colimits. To prove the result characterising product in **Occ** note that the unfolding of an occurrence net yields an occurrence net isomorphic to the original. Because the inclusion functor **Occ** $\to$ **Net** preserves colimits, it follows directly that coproducts in **Occ** coincide with those in **Net**. ∎

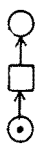
Now we consider coproducts further. The next example shows that the unfolding need not preserve coproducts however.

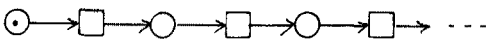
3.3.15 Example. This example is essentially the same as that given in [W3] for a category of transition systems where unfolding yields a tree. The unfolding of the net is of course itself.

The unfolding of the net

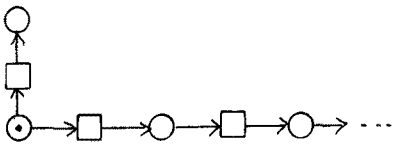


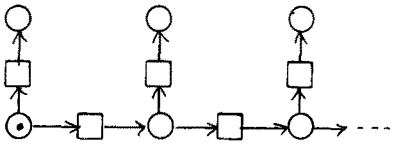
is





The coproduct of their unfoldings in **Occ** and the unfolding of their coproduct in **Net** are:





Of course we can restrict to subcategories of nets so that unfolding does preserve coproducts. A subcategory for which this is true is that for which nets satisfy: every condition in the initial marking has no pre–events.

3.4. Occurrence nets and prime event structures.

We show the relationship between the category of occurrence nets and a full subcategory of prime event structures. We show that constructions given in [NPW] determine a coreflection from

these event structures to nets. This pleasant categorical set–up extends that of the previous section and makes it easy to relate semantics given in terms of nets to those in terms of event structures, stable families, finitary prime algebraic domains and trees, and through them to other models like the *pomset* model of V.Pratt [Pr] and the *behaviour systems* of M.Shields [Sh].

Clearly an occurrence net determines an event structure [NPW]; just strip the conditions away but remember the more abstract causal dependency and conflict relation they induce.

3.4.1 Definition. Let $\mathrm{N} = (B, E, F, M)$ be an occurrence net. Define $\mathcal{E}_o(\mathrm{N}) = (E, \mathrm{Con}, F^* \restriction E)$ where $X \in \mathrm{Con}$ iff $X \subseteq_{fin} E \ \ \& \ \ \forall e, e' \in X.\ \neg(e \# e')$.

A morphism between occurrence nets N and N′ consists in part of a partial function $\eta : E \to E'$ between the associated sets of events. The partial function η is a morphism on the associated event structures.

3.4.2 Lemma. *Let $(\eta, \beta) : N_0 \to N_1$ be a morphism between occurrence nets. Then $\eta : \mathcal{E}_o(N_0) \to \mathcal{E}_o(N_1)$ is a morphism in* $\mathbf{P}$.

Proof. Assume $\mathrm{N}_0 = (B_0, E_0, F_0, M_0)$ and $\mathrm{N}_1 = (B_1, E_1, F_1, M_1)$. We are required to show:

$$\forall x \in \mathcal{L}\mathcal{E}_o(\mathrm{N}_0).\ (\eta x \in \mathcal{L}\mathcal{E}_o(\mathrm{N}_1) \ \ \& \ \ (\forall e, e' \in x.\ \eta(e) = \eta(e') \neq * \Rightarrow e = e')).$$

Let $x \in \mathcal{L}\mathcal{E}_o(\mathrm{N}_0)$.

Consider ηx. Suppose $e_1 F_1^\smile e_1' \in \eta x$. Then $\eta(e_0') = e_1'$ for some event $e_0' \in x$. By lemma 3.3.6 (i) there is some event $e_0 F_0^\smile e_0'$ such that $\eta(e_0) = e_1$. Hence ηx is left–closed. If two events e_1, e_1' in ηx are in conflict then by lemma 3.3.6(ii) this can only arise through two events events $e_0, e_0' \in x$ being in conflict, which is impossible as x is consistent. Thus x is consistent and left–closed so $\eta x \in \mathcal{L}\mathcal{E}_o(\mathrm{N}')$. By lemma 3.3.6 (iii), because x is consistent it follows that each event in ηx is the image under η of a unique event in x. ∎

3.4.3 Corollary. *The operation $\mathcal{E}_o$ extends to a functor* $\mathbf{Occ} \to \mathbf{P}$ *from occurrence nets to prime event structures by defining $\mathcal{E}_o$ on morphisms (η, β) by $\mathcal{E}_o(\eta, \beta) = \eta$.*

Proof. We have seen $\mathcal{E}_o(f) : \mathcal{E}_o(\mathrm{N}_0) \to \mathcal{E}_o(\mathrm{N}_1)$ is a morphism. Clearly $\mathcal{E}_o$ preserves identities and composition so it is a functor. ∎

Remark. Note that now we not only have a functor $\mathcal{E}_o : \mathbf{Occ} \to \mathbf{P}$ from occurrence nets to event structures but also the functor $\mathcal{E}_o\mathcal{U} : \mathbf{Net} \to \mathbf{P}$, translating arbitrary safe nets to event structures.

The prime event structures determined by occurrence nets have a special form; their consistency predicates are induced by a binary conflict relation and the consistent sets are precisely those finite sets which are conflict free. We focus on the corresponding subcategory of $\mathbf{P}$.

3.4.4 Definition. Let $(E, \mathrm{Con}, \leq)$ be a prime event structure. Define the *conflict relation* $\#$ between events e, e' by

$$e \# e' \Leftrightarrow \{e, e'\} \notin \mathrm{Con}.$$

3.4.5 Proposition. *Let E be an event structure. The conflict relation $\#$ is a binary, symmetric, irreflexive relation which satisfies*

$$e \# e' \leq e'' \Rightarrow e \# e''.$$

Proof. Clearly $\#$ is a binary, symmetric relation. Because $\{e\} \in \mathrm{Con}$ it is irreflexive. Suppose $e' \leq e''$. Then $\{e, e''\} \in Con \Rightarrow \{e, e'\} \in \mathrm{Con}$ and hence $e \# e' \Rightarrow e \# e''$. ∎

For any prime event structure, for a finite subset X of events, we have

$$X \in \mathrm{Con} \Rightarrow \forall e, e'.\ \neg e \# e'.$$

In the subcategory of interest the consistency predicate is determined by the conflict relation.

3.4.6 Definition. Define $\mathbf{P}^{\#}$ to consist of those prime event structures $\mathrm{E} = (E, \mathrm{Con}, \leq)$ for which

$$X \in \mathrm{Con} \Leftrightarrow X \subseteq_{fin} E \ \ \& \ \ \forall e, e' \in X.\ \neg(e \# e').$$

In this case we shall write E as $(E, \#, \leq)$.

We characterise morphisms in $\mathbf{P}^{\#}$ in terms of the conflict relations on event structures. They preserve enablings and reflexive closures of the conflict relation.

3.4.7 Proposition. *A morphism between prime event structures $(E_0, \#_0, \leq_0)$ and $(E_1, \#_1, \leq_1)$ is a partial function $\theta : E_0 \rightarrow_* E_1$ such that*

$$\forall e \in E_0.\ \theta(e) \text{ is defined} \Rightarrow [\theta(e)] \subseteq \theta[e] \ \ \&$$
$$\forall e, e' \in E_0.\ (\theta(e) \#_1 \theta(e') \ \text{ or } \ \theta(e) = \theta(e')) \Rightarrow (e \#_1 e' \ \text{ or } \ e = e').$$

Proof. Directly from earlier characterisations of morphisms on prime event structures specialised to this case. ∎

By the definition of $\mathbf{P}^{\#}$ we have a functor $\mathcal{E}_o : \mathbf{Occ} \rightarrow \mathbf{P}^{\#}$. It is natural to ask if, conversely, an event structure in $\mathbf{P}^{\#}$ can be identified with an occurrence net. Of course we would like every morphism between event structures to correspond to net morphism between the associated nets. We seek a functor $\mathcal{N} : \mathrm{P} \rightarrow \mathbf{Occ}$ which "embeds" the category of event structures in the category of occurrence nets, so $\mathcal{E}_o\mathcal{N}(E)$ is naturally isomorphic to the original event structure E. Ideally, we would hope that $\mathcal{E}_o$ would be a right adjoint to $\mathcal{N}$ making a coreflection. This is indeed the case and we have all the benefits explained earlier. We explain the construction of $\mathcal{N}$, a minor modification of that in [NPW].

An event structure in $\mathbf{P}^{\#}$ can be identified with a canonical occurrence net. The basic idea is to produce an occurrence net with as many conditions as are consistent with the causal dependency and conflict relations of the event structure. But we do not want more than one condition with the same beginning and ending events—we want an occurrence net which is "condition–extensional" in the terms of [Br]. Thus we can identify the conditions with pairs of the form (e, A) where e is an event and A is a subset of events causally dependant on e and with every distinct pair of events in A in conflict. But not quite, we also want initial conditions with no beginning events.

3.4.8 Definition. Let $\mathrm{E} = (E, \#, \leq)$ be an event structure. Define $\mathcal{N}(\mathrm{E})$ to be (B, E, F, M) where

$$M = \{(\emptyset, A) \mid A \subseteq E \ \ \& \ \ (\forall a, a' \in A.\ a(\# \cup 1)a')\}$$
$$B = M \cup \{(e, A) \mid e \in E \ \ \& \ \ A \subseteq E \ \ \& \ \ (\forall a, a' \in A.\ a((\# \cup 1))a') \ \ \& \ \ (\forall a \in A.\ e < a)\}$$
$$F = \{(e, (e, A)) \mid (e, A) \in B\} \cup \{((c, A), e) \mid (c, A) \in B \ \ \& \ \ e \in A\}.$$

The proof of the coreflection between occurrence nets and event structures uses the following notation and lemma which expresses a property of the relation between conditions in a morphism between occurrence nets.

3.4.9 Notation. Let $(E, \#, \leq)$ be an event structure. Define

$$\lfloor\!\lfloor \emptyset \rfloor\!\rfloor = E \quad \text{and}$$
$$\lfloor\!\lfloor \{e\} \rfloor\!\rfloor = \{e' \in E \mid e < e'\}.$$

We also use this notation for occurrence nets with the understanding that it applies to the underlying event structure.

3.4.10 Lemma. *Let $h = (\eta, \beta) : N_0 \to N_1$ be a morphism between occurrence nets. If $b_0 \beta b_1$, for conditions b_0, b_1, then*

$$(\forall e.\ e F_0 b_0 \Leftrightarrow \eta(e) F_1 b_1) \ \ \& \ \ b_0{}^\bullet = (\eta^{-1} b_1{}^\bullet) \cap \lfloor\!\lfloor ({}^\bullet b_0) \rfloor\!\rfloor.$$

Proof. Suppose $b_0 \beta b_1$, for conditions b_0, b_1. Directly from proposition 3.1.9 we see $(\forall e.\ e F_0 b_0 \Leftrightarrow \eta(e) F_1 b_1)$ and $b_0{}^\bullet \subseteq (\eta^{-1} b_1{}^\bullet) \cap \lfloor\!\lfloor ({}^\bullet b_0) \rfloor\!\rfloor$. Take $e \in (\eta^{-1} b_1{}^\bullet) \cap \lfloor\!\lfloor ({}^\bullet b_0) \rfloor\!\rfloor$. We show $e \in b_0{}^\bullet$ and hence establish the converse inclusion. There are two cases: when ${}^\bullet b_0 = \emptyset$ and when ${}^\bullet b_0 \neq \emptyset$. Assume first ${}^\bullet b_0 = \emptyset$. In this case $b_0 \in M_0$ and $b_1 \in M_1$. Because $\eta(e) \in b_1{}^\bullet$ and as h is a morphism there is a b_0' with $b_0' F_0 e_0$ and $b_0' \beta b_1$. By the property of morphisms on initial markings we must have $b_0 = b_0'$. Hence in this case $e \in b_0{}^\bullet$. Now assume the other case, that ${}^\bullet b_0 \neq \emptyset$. Then ${}^\bullet b_0 = \{e_0\}$ and ${}^\bullet b_1 = \{\eta(e_0)\}$ for some e_0. Also $\eta(e) \in b_1{}^\bullet$. Because h is between occurrence nets, by lemma 3.3.6, there is a b_0' such that $e_0 F_0 b_0' F_0 e$ and $b_0' \beta b_1$. But now as h is a morphism $b_0 = b_0'$. Hence $e \in b_0{}^\bullet$. Thus in either case we have established the required converse inclusion and so shown $b_0{}^\bullet = (\eta^{-1} b_1{}^\bullet) \cap \lfloor\!\lfloor ({}^\bullet b_0) \rfloor\!\rfloor$. ∎

This time it is easier to establish the coreflection by showing the freeness of the occurrence net associated with an event structure.

3.4.11 Theorem. *Let E be an event structure in $\mathbf{P}^{\#}$.*

Then $\mathcal{N}(E)$ is an occurrence net. Moreover, $\mathcal{E}_o \mathcal{N}(E) = E$.

The net $\mathcal{N}(E)$ and identity function $1_E : E \to \mathcal{E}_o \mathcal{N}(E)$ is free over E with respect to $\mathcal{E}_o$ i.e. for any morphism $\eta : E \to \mathcal{E}_o(N)$ in $\mathbf{P}^{\#}$ there is a unique morphism $h : \mathcal{N}(E) \to N$ in **Occ** *such that $\mathcal{E}_o(h)\ 1_E = \eta$ (i.e. $\mathcal{E}_o(h) = \eta$).*

Proof. Let $(E, \#, \leq)$ be an event structure. It is easy to see $\mathcal{N}(E, \#, \leq)$ is an occurrence net and $\mathcal{E}_o\ \mathcal{N}(E, \#, \leq) = (E, \#, \leq)$. We prove freeness.

Let $\mathrm{N} \in$ **Occ** and $\eta : E \to \mathcal{E}_o(\mathrm{N})$ be a morphism in $\mathbf{P}^{\#}$. Define $h = (\eta, \beta)$ by taking

$$b_0 \beta b_1 \Leftrightarrow (\forall e.\ e F_0 b_0 \Leftrightarrow \eta(e) F_1 b_1) \ \ \& \ \ b_0{}^\bullet = (\eta^{-1} b_1{}^\bullet) \cap \lfloor\!\lfloor ({}^\bullet b_0) \rfloor\!\rfloor, \tag{1}$$

for b_0 a condition of N_0 and b_1 a condition of N. We require that $h : \mathcal{N}(E) \to \mathrm{N}$ is the unique morphism such that $\mathcal{E}_o(h) = \eta$.

To show h is a morphism we use the characterisation of proposition 3.1.9 and show h satisfies the conditions (i), (ii) and (iii) written there.

(i)If ${}^\bullet b_0 = \emptyset$ then ${}^\bullet b_1 = \emptyset$ so $\beta M_0 \subseteq M_1$. Let $b_1 \in M_1$. Take $b_0 = (\emptyset, \eta^{-1} b_1{}^\bullet)$. Then $b_0 \in M_0$ because $\eta^{-1} b_1{}^\bullet$ is pairwise $(\# \cup 1)$, and $b_0 \beta b_1$ by (1), the definition of β. Suppose $b_0' \beta b_1$ and

$b_0' \in M_0$. Then ${}^\bullet b_0' = {}^\bullet b_0 = \emptyset$ and $b_0'^\bullet = (\eta^{-1} b_1{}^\bullet) \cap \lfloor\!\lfloor ({}^\bullet b_0') \rfloor\!\rfloor = (\eta^{-1} b_1{}^\bullet) \cap \lfloor\!\lfloor ({}^\bullet b_0) \rfloor\!\rfloor = b_0{}^\bullet$. By the condition–extensionality of $\mathcal{N}(\mathrm{E})$ we obtain $b_0 = b_0'$.

(ii) Assume $\eta(e_0) = e_1$.

If $b_0 \in e_0{}^\bullet$ and $b_0 \beta b_1$ then ${}^\bullet b_1 = \eta^\bullet b_0 = \{e_1\}$. Hence $\beta e_0{}^\bullet \subseteq e_1{}^\bullet$. Let $b_1 \in e_1{}^\bullet$. Take $b_0 = (\{e_0\}, \eta^{-1} b_1{}^\bullet \cap \lfloor\!\lfloor \{e_0\} \rfloor\!\rfloor)$. Then $b_0 \in B_0$ by the properties of morphisms on event structures. Also $b_0 \in e_0{}^\bullet$. Suppose $b_0' \beta b_1$ with $b_0' \in e_0{}^\bullet$. Then ${}^\bullet b_0' = {}^\bullet b_0 = \{e_0\}$ and $b_0'^\bullet = (\eta^{-1} b_1{}^\bullet) \cap \lfloor\!\lfloor ({}^\bullet b_0') \rfloor\!\rfloor = (\eta^{-1} b_1{}^\bullet) \cap \lfloor\!\lfloor e_0 \rfloor\!\rfloor = b_0{}^\bullet$. By condition extensionality $b_0' = b_0$.

Suppose $b_0 \in {}^\bullet e_0$ and $b_0 \beta b_1$. Then $e_0 \in b_0{}^\bullet = (\eta^{-1} b_1{}^\bullet) \cap \lfloor\!\lfloor ({}^\bullet b_0') \rfloor\!\rfloor$ so $e_1 = \eta(e_0) \in b_1{}^\bullet$ which makes $b_1 \in {}^\bullet e_1$. Hence $\beta^\bullet e_0 \subseteq {}^\bullet e_1$. Let $b_1 \in {}^\bullet e_1$. Consider the two cases: ${}^\bullet b_1 = \emptyset$ and ${}^\bullet b_1 \neq \emptyset$. If ${}^\bullet b_1 = \emptyset$ take $b_0 = (\emptyset, \eta^{-1} b_1{}^\bullet)$. Otherwise ${}^\bullet b_1 = \{e'\}$, say. Because $\eta : E \to \mathcal{E}_o(\mathrm{N})$ is a morphism of event structures $\eta(e) = e'$ for some $e \in E$. In this case take $b_0 = (\{e\}, (\eta^{-1} b_1{}^\bullet) \cap \lfloor\!\lfloor e \rfloor\!\rfloor)$. In either case $b_0 \in B_0$ and by (1) we see $b_0 \beta b_1$. Assume $b_0' \beta b_1$. Then from (1), the definition of β, we see ${}^\bullet b_0' = {}^\bullet b_0$ and $b_0'^\bullet = b_0{}^\bullet$. By condition–extensionality $b_0' = b_0$.

(iii) Now suppose $b_0 \beta b_1$. If $e_0 F_0 b_0$ then by (1), $\eta(e_0)$ is defined and $\eta(e_0) F_1 b_1$. If $b_0 F_0 e_0$ then $e_0 \in b_0{}^\bullet \subseteq \eta^{-1} b_1{}^\bullet$ so $\eta(e_0)$ is defined and $b_1 F_1 \eta(e_0)$.

By proposition 3.1.9, we conclude $h = (\eta, \beta)$ is indeed a morphism $\mathcal{N}(\mathrm{E}) \to \mathrm{N}$.

It remains to show that $h : \mathcal{N}(\mathrm{E}) \to \mathrm{N}$ is the unique morphism such that $\mathcal{E}_o(h) = \eta$. We do this by showing that any such morphism $f = (\eta, \beta') : \mathcal{N}(\mathrm{E}) \to E$ must satisfy (1) *i.e.*

$$b_0 \beta' b_1 \Leftrightarrow (\forall e.\ e F_0 b_0 \Leftrightarrow \eta(e) F_1 b_1) \ \ \& \ \ b_0{}^\bullet = (\eta^{-1} b_1{}^\bullet) \cap \lfloor\!\lfloor ({}^\bullet b_0) \rfloor\!\rfloor.$$

which makes $\beta' = \beta$ so $f = h$. By the lemma above any such morphism f must satisfy

$$b_0 \beta' b_1 \Rightarrow (\forall e.\ e F_0 b_0 \Leftrightarrow \eta(e) F_1 b_1) \ \ \& \ \ b_0{}^\bullet = (\eta^{-1} b_1{}^\bullet) \cap \lfloor\!\lfloor ({}^\bullet b_0) \rfloor\!\rfloor.$$

To show the converse assume $(\forall e.\ e F_0 b_0 \Leftrightarrow \eta(e) F_1 b_1)$ and $b_0{}^\bullet = (\eta^{-1} b_1{}^\bullet) \cap \lfloor\!\lfloor ({}^\bullet b_0) \rfloor\!\rfloor$. Because f is a morphism, $b_0' \beta' b_1$ for some b_0' with ${}^\bullet b_0' = {}^\bullet b_0$ (consider the two cases ${}^\bullet b_0 = \emptyset$ and ${}^\bullet b_0 \neq \emptyset$). By the lemma above $b_0'^\bullet = (\eta^{-1} b_1{}^\bullet) \cap \lfloor\!\lfloor ({}^\bullet b_0') \rfloor\!\rfloor = (\eta^{-1} b_1{}^\bullet) \cap \lfloor\!\lfloor ({}^\bullet b_0) \rfloor\!\rfloor = b_0{}^\bullet$. By the condition–extensionality of $\mathcal{N}(\mathrm{E})$, $b_0 = b_0'$ so $b_0 \beta' b_1$. Thus β' satisfies (1). Hence $\beta' = \beta$ so $f = h$, establishing uniqueness. ∎

Thus there is a coreflection between event structures and occurrence nets with $\mathcal{E}_o$ as its right adjoint and $\mathcal{N}$ as its left adjoint. This composes with the coreflection between occurrence nets and safe nets to give a coreflection between event structures and safe nets.

Reasoning in the same way as we did for the coreflection between **Net** and **Occ**, we see, for instance,

$$\begin{aligned} \mathcal{E}_o(\mathrm{N}_0 \times_{Occ} \mathrm{N}_1) &\cong \mathcal{E}_o(\mathrm{N}_0) \times_P \mathcal{E}_o(\mathrm{N}_1) \\ E_0 \times_P E_1 &\cong \mathcal{E}_o \mathcal{U}(\mathcal{N}(E_0) \times_{Net} \mathcal{N}(E_1)) \\ E_0 +_P E_1 &\cong \mathcal{E}_o \mathcal{U}(\mathcal{N}(E_0) +_{Net} \mathcal{N}(E_1)), \end{aligned}$$

which translates constructions in one category to constructions in the other, giving the product and coproduct in $\mathbf{P}^\#$ in terms of the product and coproduct in **Net**. With extra labelling structure on nets one can carry out the construction for parallel composition and the other constructions needed for $\mathbf{Proc}_L$ pretty much as before, secure in the knowledge of how the different models and semantics are related.

We can summarise how the different categories are related in a diagram where all functors are left adjoints. The functor $\mathbf{P} \to \mathbf{P}^\#$ is a left–adjoint to the inclusion functor $\mathbf{P}^\# \to \mathbf{P}$; it takes an

event structure $(E, \mathrm{Con}, \leq)$ in $\mathbf{P}$ to the event structure $(E, \#, \leq)$ in $\mathbf{P}^{\#}$. This and the adjunctions involving transition systems **TS** and event structures in general $\mathbf{E}_g$ are not proved here. The inclusion functor identifying trees with a certain kind of transition system has the unfolding functor as a right adjoint—see [W3]. The inclusion functor $\mathbf{E} \to \mathbf{E}_g$ from stable event structures to general event structures has a right adjoint which essentially makes enough copies of the events to ensure the stability condition.

$$\begin{array}{cccccccc} \mathbf{T} & \to & \mathbf{D} & \leftarrow & \mathbf{F} & \to & \mathbf{E} & \to \mathbf{E}_g \\ \downarrow & & \uparrow\downarrow & \nearrow & & & & \\ \mathbf{TS} & & \mathbf{P} & & & & & \\ & & \downarrow & & & & & \\ & & \mathbf{P}^{\#} & \to & \mathbf{Occ} & \to & \mathbf{Net} & \end{array}$$

In particular the functor $\mathcal{DFE}_o\mathcal{U} : \mathbf{Net} \to \mathbf{D}$ translates safe nets into domains, and we can ask what properties of domains correspond to what properties of nets. There is a result relating confusion in nets to concreteness in domains. Say a net is *not confused* iff there no reachable marking at which either symmetric or asymmetric confusion occurs.

3.4.12 Theorem. *Let N be a safe net. Then*

$$\mathcal{DFE}_o\mathcal{U}(N) \textit{ is concrete iff } N \textit{ is not confused.}$$

Proof. See [NPW] for an account and [W] for the full details. ∎

Part 4. HIGHER TYPES.

As motivation the full–abstraction problem for typed λ–calculi is introduced. This motivates a more operational approach to domain theory. It is shown how event structures can be used to model datatypes of functions and functions on functions *etc.* . Using another definition of morphism event structures can be made into a cartesian closed category equivalent to one discovered by G.Berry. In this category functions are not ordered extensionally, by the pointwise order, as in Scott's category of domains but intensionally, by the stable order, which takes into account the manner in which they compute. It is indicated how a model of the λ–calculus can be constructed.

4.1. Background.

At first sight it is perhaps rather remarkable that event structures should provide models for programming languages with higher types such as the typed or untyped λ calculus. For one thing it is not immediately clear what an event at higher type is. More strikingly, the well–known models for such languages originating with D.Scott make essential use of a particular function space construction on domains, that formed by taking the set of continuous functions ordered pointwise. This construction quickly takes domains outside the finitary ones, and as we have seen all domains determined by event structures are finitary. Nevertheless there are forms of function space construction on event structures, yielding cartesian closed categories of event structures. The one we shall define gives rise to a different function space constructions on domains, and is associated with a more restricted class of functions than just those which are continuous, and the ordering on functions is different too.

In [P], Plotkin uncovered the full–abstraction problem for PCF, a programming language, built around a typed lambda calculus with fixed–point operators, whose terms at ground type—call them

programs—compute integers or truth values. We explain the problem briefly (refer to [B, C, W], especially [C], for more details). Plotkin defined a natural preorder on terms. In PCF only programs can yield definite results, and terms at higher type are of interest only in so far as they are parts of programs. It is natural to regard two terms (of the same type) as operationally equivalent iff they can be freely substituted for each other in any program without changing its output behaviour. Formally define the equivalence relation to hold between terms M and N of the same type by

> $M \equiv N$ iff for all program contexts $C[\ ]$ either the evaluations of both $C[M]$ and $C[N]$ diverge or they converge to the same value.

More generally, an operational preorder can be defined by taking

> $M \preceq N$ iff for all program contexts $C[\ ]$ if the evaluation of $C[M]$ converges to a value then so does that of $C[N]$ converge to the same value.

A denotational semantics also provides a preorder on terms. Write $M \sqsubseteq N$ iff the denotation of M is below that of N. Ideally one would hope that the two preorders, operational and denotational, are equal. In such a case it is said that the denotational semantics is *fully abstract*.

Unfortunately, as Plotkin showed, the obvious denotational semantics for PCF, interpreting higher types using the space of all continuous functions, does not lead to a fully abstract model. Plotkin produced two terms which were operationally equivalent but denotationally distinct through acting differently on "parallel or". "Parallel or" is a function which takes a pair of truth values including $\perp$ for "unknown" and gives value "true" if either argument is "true". "Parallel or" existed as a function in the domain but could never be defined in the language or supplied in a program context. Plotkin went on to show that by extending PCF to allow limited parallelism the obvious model became fully abstract. Milner filled out the picture by showing there is a fully abstract model for the original PCF but his method was essentially to construct a term model from the operational semantics. There remained—and still remains—the problem of providing a semantic construction of the fully–abstract model.

The full abstraction problem led G.Berry, and following him P.L.Curien, on the quest to find a semantic characterisation of the concept of sequential function at higher type. They hoped to eliminate problematic elements like "parallel or". Attacks on the problem led Berry to discover a range of new cartesian closed categories of domains. (Roughly, a category is cartesian closed iff it has products and function spaces—see [Mac].) Here it will be shown how the simplest of these, the category of finitary, distributive domains (which Berry called the dI–domains) with stable functions can be represented as a category of event structures. Other cartesian closed categories of event structures which are better approximations to the fully abstract model can be found in [W] and [BC, C]. I would especially like to highlight the work of Berry and Curien on CDS presented in [BC, C]. CDS, standing for "concrete data structures", is a programming language, which has been implemented, in which the data types are concrete data structures and the computations are "algorithms", in a technical sense, between them. CDS has an elegant mathematical theory; concrete data structures and algorithms form a cartesian closed category of objects intimately linked to event structures and so many of the concepts overlap those encountered in the study of Petri nets.

4.2. Higher–type events.

The simplest new cartesian closed category in [B] consists of the finitary, distributive domains with stable, continuous functions. As we have seen such domains are precisely those formed as the domains of configurations of stable event structures, so we can get an equivalent category by taking

the objects to be stable event structures and the morphisms between event structures to be stable, continuous functions on the associated domains of configurations. (It is easily confirmed that it is indeed a category under the usual function composition.)

4.2.1 Definition. Define $\mathbf{E}_{stab}$ to be the category with objects stable event structures and with morphisms from E_0 to E_1 the stable, continuous functions $f : (\mathcal{F}(\mathrm{E}_0), \subseteq) \to (\mathcal{F}(\mathrm{E}_1), \subseteq)$ on their configurations, *i.e.* f is continuous and

$$\forall X \subseteq \mathcal{F}(\mathrm{E}_0).\ X \neq \emptyset \ \ \& \ \ X\uparrow \Rightarrow f(\bigcap X) = \bigcap fX.$$

Composition is composition of functions and identities are the identity functions on configurations.

The product in the category is obtained very simply. The event structures are allowed to operate disjointly, completely in parallel, neither one having an effect on the other. It is easily defined for all event structures not just the stable ones.

4.2.2 Definition. Let $\mathrm{E}_0 = (E_0, \mathrm{Con}_0, \vdash_0)$ and $\mathrm{E}_1 = (E_1, \mathrm{Con}_1, \vdash_1)$ be stable event structures. Their *disjoint product*, $\mathrm{E}_0 \oplus \mathrm{E}_1$, is the structure $(E, \mathrm{Con}, \vdash)$ where the events are

$$E = \{0\} \times E_0 \cup \{1\} \times E_1,$$

a disjoint union, the consistency predicate is given by

$$X \in \mathrm{Con} \Leftrightarrow \{e \mid (0,e) \in X\} \in \mathrm{Con}_0 \ \ \& \ \ \{e \mid (1,e) \in X\} \in \mathrm{Con}_1,$$

and the enabling by

$$\begin{aligned} X \vdash e \Leftrightarrow & X \in \mathrm{Con} \ \ \& \ \ e \in E \ \ \& \\ & [(\exists e_0 \in E_0.\ e = (0,e_0) \ \ \& \ \ \{e' \mid (0,e') \in X\} \vdash_0 e_0) \ \text{ or} \\ & (\exists e_1 \in E_1.\ e = (1,e_1) \ \ \& \ \ \{e' \mid (1,e') \in X\} \vdash_1 e_1)]. \end{aligned}$$

Define the *projections* $p_k : \mathcal{F}(\mathrm{E}_0 \oplus \mathrm{E}_1) \to \mathcal{F}(\mathrm{E}_k)$ by taking $p_k(x) = \{e \mid (k,e) \in x\}$, for $k = 0, 1$.

4.2.3 Proposition. *Let E_0 and E_1 be event structures with events E_0, E_1 respectively. Then*

$$x \in \mathcal{F}(E_0 \oplus E_1) \Leftrightarrow x \subseteq E_0 \uplus E_1 \ \ \& \ \ p_0(x) \in \mathcal{F}(E_0) \ \ \& \ \ p_1(x) \in \mathcal{F}(E_1).$$

There is a 1–1 correspondence between $\mathcal{F}(E_0 \oplus E_1)$ and $\mathcal{F}(E_0) \times \mathcal{F}(E_1)$ given by

$$x \mapsto (p_0(x), p_1(x)).$$

The disjoint product is $\trianglelefteq$–continuous.

Proof. Obvious. Routine application of lemma 1.6.9 gives the continuity of the disjoint product. ∎

Thus we can identify x, a configuration of a disjoint product, with the pair $(p_0(x), p_1(x))$.

4.2.4 Theorem. *The disjoint product $E_0 \oplus E_1$ of stable event structures E_0 and E_1, with projections π_0, π_1, is a product in the category $\mathbf{E}_{stab}$.*

Proof. Obviously the disjoint product of stable event structures is stable. It is easy to see that the projections are stable functions. The disjoint product is easily seen to be a product now its configurations are recognised to be essentially pairs of configurations of the components. ∎

To be cartesian closed we must somehow represent the space of stable, continuous functions $f : \mathrm{E}_0 \to \mathrm{E}_1$ between two stable event structures E_0 and E_1 as an event structure itself. This is done by taking the events of a "function space" event structure to be basic parts of functions (x, e) standing for the event of outputting e, an event of E_1, at input x, a finite configuration of E_0. The function f will correspond to a configuration of events (x, e) in which x is a minimal input configuration at which e is output.

4.2.5 Definition. Let $\mathrm{E}_0 = (E_0, \mathrm{Con}_0, \vdash_0)$ and $\mathrm{E}_1 = (E_1, \mathrm{Con}_1, \vdash_1)$ be stable event structures. Their *stable function space*, $[\mathrm{E}_0 \to \mathrm{E}_1]$ is defined to be the event structure $(E, \mathrm{Con}, \vdash)$ with events E consisting of pairs (x, e) where x is a finite configuration in $\mathcal{F}(\mathrm{E}_0)$ and $e \in E_1$, a consistency predicate Con given by

$$\begin{aligned} &\{(x_0, e_0), \cdots, (x_{n-1}, e_{n-1})\} \in \mathrm{Con} \\ \text{iff } &\forall I \subseteq \{0, \cdots, n-1\}.\ \textstyle\bigcup_{i \in I} x_i \in \mathrm{Con}_0 \Rightarrow \{e_i \mid i \in I\} \in \mathrm{Con}_1 \ \ \& \\ &\forall i, j < n.\ x_i \uparrow x_j \ \ \& \ \ e_i = e_j \Rightarrow x_i = x_j, \end{aligned}$$

and an enabling relation given by

$$\{(x_0, e_0), \cdots, (x_{n-1}, e_{n-1})\} \vdash (x, e) \text{ iff } \{e_i \mid x_i \subseteq x\} \vdash_1 e.$$

4.2.6 Proposition. *The stable function space of two stable event structures is a stable event structure. The stable function space construction is $\trianglelefteq$-continuous.*

Proof. Let $\mathrm{E}_0 = (E_0, \mathrm{Con}_0, \vdash_0)$ and $\mathrm{E}_1 = (E_1, \mathrm{Con}_1, \vdash_1)$ be stable event structures. Obviously their function space forms an event structure $(E, \mathrm{Con}, \vdash)$. Suppose

$$\begin{aligned} &\{(w_i, a_i) \mid i \in I\} \vdash (x, e), \\ &\{(y_i, b_i) \mid j \in J\} \vdash (x, e) \text{ and} \\ &\{(w_i, a_i) \mid i \in I\} \cup \{(y_i, b_i) \mid j \in J\} \cup \{(x, e)\} \in \mathrm{Con}. \end{aligned}$$

Then, by the definition of enabling and consistency in the function space,

$$\begin{aligned} &\{a_i \mid w_i \subseteq x\} \vdash_1 e, \\ &\{b_i \mid y_j \subseteq x\} \vdash_1 e \text{ and} \\ &\{a_i \mid w_i \subseteq x\} \cup \{b_i \mid y_j \subseteq x\} \cup \{e\} \in \mathrm{Con}_1. \end{aligned}$$

As E_1 is stable $\{a_i \mid w_i \subseteq x\} \cap \{b_i \mid y_j \subseteq x\} \vdash_1 e$. By the definition of consistency on the function space $a_i = b_j \Rightarrow w_i = y_j$ so

$$\{(w_i, a_i) \mid i \in I\} \cap \{(y_i, b_i) \mid j \in J\} \vdash (x, e).$$

Thus the function space is a stable event structure.

It is easy to check $[\ \to\ ]$ is monotonic in each argument. The operation obtained by varying the right hand side argument is obviously continuous on events. Because events in the function space are built from only finite configurations so is that for the left hand argument. By lemma 1.6.9 the function space operation is $\trianglelefteq$-continuous. ∎

Remark. Given two event structures E_0 and E_1, not necessarily stable, a similar construction can be given to provide an event structure $[\mathrm{E}_0 \to_c \mathrm{E}_1]$, again not necessarily stable, whose configurations are in 1–1 correspondence with the continuous functions $(\mathcal{F}(\mathrm{E}_0), \subseteq) \to (\mathcal{F}(\mathrm{E}_0), \subseteq)$. (This is

remarked in [G] though for the more restrictive category of qualitative domains.) However, unlike the stable function space, this construction will not be the exponentiation in the category of domains of configurations with continuous functions. The category of event structures with continuous functions between their associated domains is not cartesian closed by Smyth's lemma 5 in [Smy], or Curien's theorem 2.4.13 and the remark that follows in [C] (p.158–160).

In fact the stable function space $[\mathrm{E}_0 \to \mathrm{E}_1]$ of stable event structures can be provided with a stable, continuous application function $\mathrm{ap} : [\mathrm{E}_0 \to \mathrm{E}_1] \oplus \mathrm{E}_0 \to \mathrm{E}_1$ given by

$$\mathrm{ap}(f, x) = \{e_1 \in E_1 \mid \exists x_0 \subseteq x.\ (x_0, e_1) \in f\}$$

for $f \in \mathcal{F}([\mathrm{E}_0 \to \mathrm{E}_1])$ and $x \in \mathcal{F}(\mathrm{E}_0)$. (We have identified (f, x) with the corresponding configuration of the disjoint product.) As we shall see this makes the function space an exponentiation in the category $\mathbf{E}_{stab}$. Firstly though it is helpful to show how the configurations of a stable function space $[\mathrm{E}_0 \to \mathrm{E}_1]$ correspond to stable, continuous functions $\mathcal{F}(\mathrm{E}_0) \to \mathcal{F}(\mathrm{E}_1)$.

4.2.7 Definition. Let E_0 and E_1 be stable event structures.

For $F \in \mathcal{F}([\mathrm{E}_0 \to \mathrm{E}_1])$ define

$$(\phi(F))(x) = \{e \in E_1 \mid \exists x' \subseteq x.\ (x', e) \in F\}$$

for $x \in \mathcal{F}(\mathrm{E}_0)$.

For $f : \mathcal{F}(\mathrm{E}_0) \to \mathcal{F}(\mathrm{E}_1)$ a stable, continuous function define $\mu(f)$ a subset of events of $[\mathrm{E}_0 \to \mathrm{E}_1]$ by

$$(x, e) \in \mu(f) \Leftrightarrow e \in f(x) \ \ \& \ \ (\forall x' \subseteq x.\ e \in f(x') \Rightarrow x' = x).$$

4.2.8 Theorem. *Let E_0 and E_1 be stable event structures.*
(i) For $F \in \mathcal{F}([E_0 \to E_1])$, the function $\phi(F) : \mathcal{F}(E_0) \to \mathcal{F}(E_1)$ is continuous and stable.
(i) For $f : \mathcal{F}(E_0) \to \mathcal{F}(E_1)$ a stable, continuous function, the subset $\mu(f) \in \mathcal{F}([E_0 \to E_1])$.
(iii) Further, ϕ and μ are mutual inverses giving a 1-1 correspondence between configurations $\mathcal{F}([E_0 \to E_1])$ and stable, continuous functions $\mathcal{F}(E_0) \to \mathcal{F}(E_1)$.

Proof. Assume $\mathrm{E}_0 = (E_0, \mathrm{Con}_0, \vdash_0)$, $\mathrm{E}_1 = (E_1, \mathrm{Con}_1, \vdash_1)$ and the function space $[\mathrm{E}_0 \to \mathrm{E}_1] = (E, \mathrm{Con}, \vdash)$.

(i) Firstly we check $\phi(F)(x) \in \mathcal{F}(\mathrm{E}_1)$ when $F \in \mathcal{F}([\mathrm{E}_0 \to \mathrm{E}_1])$ and $x \in \mathcal{F}(\mathrm{E}_0)$. So we require that $\phi(F)(x)$ is consistent and secured in E_1.

Suppose $Y \subseteq_{fin} \phi(F)(x)$. Write $Y = \{e_0, \ldots, e_{n-1}\}$. Then there are finite configurations $x_0, \ldots, x_{n-1}$ of E_0 such that $(x_0, e_0), \ldots, (x_{n-1}, e_{n-1}) \in F$ and $x_0, \ldots, x_{n-1} \subseteq x$. Thus $\bigcup_{i<n} x_i \in \mathrm{Con}_0$ and so as F is consistent in $[\mathrm{E}_0 \to \mathrm{E}_1]$ we obtain $Y = \{e_0, \ldots, e_{n-1}\} \in \mathrm{Con}_1$. Therefore $\phi(F)(x)$ is consistent.

Suppose $e \in \phi(F)(x)$. Then $(x', e) \in F$ for some finite configuration x' of E_0. As F is a configuration of $[\mathrm{E}_0 \to \mathrm{E}_1]$, so secured, there is a sequence $(x_0, e_0), \cdots, (x_n, e_n) = (x', e)$ in F such that $\{(x_0, e_0), \cdots, (x_{i-1}, e_{i-1})\} \vdash (x_i, e_i)$ for all $i \leq n$. Recall this means $\{e_j \mid j < i \ \ \& \ \ x_j \subseteq x_i\} \vdash_1 e_i$. Thus without loss of generality we may assume $x_i \subseteq x'$ for each i in the securing—any event (x_i, e_i) failing this can be removed to still leave a securing for (x', e). For such chains $e_0, \ldots, e_n$ is a securing for e in $\phi(F)(x)$.

Hence $\phi(F)(x)$ is consistent and secured, and so an element of $\mathcal{F}(\mathrm{E}_1)$. This shows $\phi(F)$ is a function $\mathcal{F}(\mathrm{E}_0) \to \mathcal{F}(\mathrm{E}_1)$. It is obviously monotonic. That it is continuous follows because F consists

of events of the form (x, e) with x finite. Suppose X is a nonempty, compatible subset of $\mathcal{F}(\mathrm{E}_0)$. By the monotonicity of $\phi(F)$ we see

$$\phi(F)(\bigcap X) \subseteq \bigcap_{x \in X} \phi(F)(x).$$

Suppose $e \in \bigcap_{x \in X} \phi(F)(x)$. Then for any $x \in X$ there is a finite configuration m_x of E_0 for which $m_x \subseteq x$ and $(m_x, e) \in F$. The set $\{m_x \mid x \in X\}$ is compatible and as F is consistent each $m_x = m$ say for $x \in X$. Now $m \subseteq \bigcap X$ making $e \in \phi(F)(\bigcap X)$ as, of course, $(m, e) \in F$. Therefore

$$\phi(F)(\bigcap X) = \bigcap_{x \in X} \phi(F)(x),$$

showing $\phi(F)$ is stable.

(ii) Let $f : \mathcal{F}(\mathrm{E}_0) \to \mathcal{F}(\mathrm{E}_1)$ be a stable, continuous function. We show $\mu(f) \in \mathcal{F}([\mathrm{E}_0 \to \mathrm{E}_1])$, *i.e.* that $\mu(f)$ is consistent and secured in $[\mathrm{E}_0 \to \mathrm{E}_1]$.

Suppose $\{(x_0, e_0), \cdots, (x_{n-1}, e_{n-1})\} \subseteq \mu(f)$. Assume $I \subseteq \{0, \ldots, n-1\}$ and $\bigcup_{i \in I} x_i \in \mathrm{Con}_0$. Write $x = \bigcup_{i \in I} x_i \in \mathrm{Con}_0$. Of course $x \in \mathcal{F}(\mathrm{E}_0)$. We have $\{e_i \mid i \in I\} \subseteq f(x)$. Consequently $\{e_i \mid i \in I\} \in \mathrm{Con}_1$. Assume $x_i \cup x_j \in \mathrm{Con}$ and $e_i = e_j = e$ say. Then as f is stable

$$f(x_i \cap x_j) = f(x_i) \cap f(x_j) \ni e.$$

As x_i and x_j are minimal inputs yielding e we must have $x_i = x_i \cap x_j = x_j$. Thus $\mu(f)$ is consistent.

We now require that

$$(IH) \qquad (x, e) \in \mu(f) \Rightarrow (x, e) \text{ is secured in } \mu(f).$$

for all finite configurations x of E_0. We show this by induction on the size $|x|$ of the finite configuration x. When $|x| = 0$ then $x = \emptyset$. If $(\emptyset, e) \in \mu(f)$ then $e \in f(\emptyset)$. As $f(\emptyset)$ is secured there is a securing $e_0, \ldots, e_n$ for e in $f(\emptyset)$. This makes $(\emptyset, e_0), \ldots, (\emptyset, e_n)$ a securing for $(\emptyset, e)$ in $\mu(f)$. Assume now that $|x| > 0$ and that (IH) holds for all strictly smaller configurations. Assume $(x, e) \in \mu(f)$. Then $e \in f(x)$ so there is a securing $e_0, \ldots, e_n = e$ for e in $f(x)$. There are $x_0, \ldots, x_n \subseteq x$ such that $(x_0, e_0), \ldots, (x_n, e_n) \in \mu(f)$ with $x_n = x$. Working along this sequence, we see that for each i, $0 \leq i \leq n$, either $x_i \subset x$ so (x_i, e_i) has a securing s_i in $\mu(f)$ by (IH), or $x_i = x$ so $\{(x_0, e_0), \ldots, (x_{i-1}, e_{i-1})\} \vdash (x_i, e_i)$ and $s_0 \frown \cdots \frown s_{i-1}(x_i, e_i)$ is a securing for (x_i, e_i). This shows (x, e) is secured in $\mu(f)$.

We conclude $\mu(f) \in \mathcal{F}([\mathrm{E}_0 \to \mathrm{E}_1])$.

(iii) Now we show ϕ and μ determine a 1–1 correspondence.

Let $F \in \mathcal{F}([\mathrm{E}_0 \to \mathrm{E}_1])$. We require $\mu\phi(F) = F$. Suppose $(x, e) \in F$. Then $e \in \phi(F)(x)$. If $e \in \phi(F)(x')$ for $x' \subset x$ there would have to be some $y \subset x$ with $(y, e) \in F$, impossible by the consistency requirement on F. Therefore $(x, e) \in \mu\phi(F)$. Suppose $(x, e) \in \mu\phi(F)$. Then $e \in \phi(F)(x)$ and $(x', e) \in F$ for some $x' \subseteq x$. But the minimality of x ensures $x = x'$, giving $(x, e) \in F$. We have shown $\mu\phi(F) = F$.

Let $f : \mathcal{F}(\mathrm{E}_0) \to \mathcal{F}(\mathrm{E}_1)$ be stable, continuous. We require $\phi\mu(f) = f$. Then, using the continuity of f,

$$e \in f(x) \Leftrightarrow \exists x' \subseteq x.\ (x', e) \in \mu(f) \Leftrightarrow e \in (\phi\mu(f))(x),$$

for any $e \in E_1$ and $x \in \mathcal{F}(\mathrm{E}_0)$. Therefore $\phi\mu(f) = f$, and we have established the required 1–1 correspondence. ∎

At this point we can quickly prove the cartesian closure of $\mathbf{E}_{stab}$, based on the observation that, for stable event structures $\mathrm{E}, \mathrm{E}_0, \mathrm{E}_1$ the two event structures

$$[\mathrm{E} \oplus \mathrm{E}_0 \to \mathrm{E}_1]$$

and

$$[\mathrm{E} \to [\mathrm{E}_0 \to \mathrm{E}_1]]$$

are the same up to a natural renaming of events.

4.2.9 Lemma. *Let E, E_0, E_1 be stable event structures. There is a 1–1 correspondence θ between the events of $[E \oplus E_0 \to E_1]$ and $[E \to [E_0 \to E_1]]$ given by*

$$\theta : ((w, x), e) \longleftrightarrow (w, (x, e)),$$

for w, x finite configurations of E, E_0 and event e of E_1, such that

$$X \in Con_P \Leftrightarrow \theta X \in Con_F$$

and

$$X \vdash_P e \Leftrightarrow \theta X \vdash_F \theta(e),$$

where $Con_P, \vdash_P$ are the consistency and entailment relations of $[E_0 \oplus E_1 \to E_2]$ and $Con_F, \vdash_F$ are the relations of $[E_0 \to [E_1 \to E_2]]$.

Proof. Let $\mathrm{E} = (E, \mathrm{Con}, \vdash)$, $\mathrm{E}_0 = (E_0, \mathrm{Con}_0, \vdash_0)$ and $\mathrm{E}_1 = (E_1, \mathrm{Con}_1, \vdash_1)$ be stable event structures. Assume

$$\begin{aligned} [\mathrm{E} \oplus \mathrm{E}_0 \to \mathrm{E}_1] &= (P, \mathrm{Con}_P, \vdash_P), \\ [\mathrm{E} \to [\mathrm{E}_0 \to \mathrm{E}_1]] &= (F, \mathrm{Con}_F, \vdash_F), \\ [\mathrm{E}_0 \to \mathrm{E}_1] &= (E_2, \mathrm{Con}_2, \vdash_2). \end{aligned}$$

Clearly $\theta : P \to F$ defined above is a 1–1 correspondence between sets of events.

Compare the notions of consistency.

Consistency in $[E \to [E_0 \to E_1]]$:

$\{(w_i, (x_i, e_i)) \mid i \in I\} \in \mathrm{Con}_F$ iff

$1(i)$ $\quad \forall J \subseteq I.\ \{w_i \mid i \in J\}\uparrow \Rightarrow \{(x_i, e_i) \mid i \in J\} \in \mathrm{Con}_2$ &

$1(ii)$ $\quad \forall i, j \in I.\ w_i \uparrow w_j$ & $(x_i, e_i) = (x_j, e_j) \Rightarrow w_i = w_j$.

Consistency in $[E \oplus E_0 \to E_1]$:

$\{((w_i, x_i), e_i)) \mid i \in I\} \in \mathrm{Con}_P$ iff

$2(i)$ $\quad \forall J \subseteq I.\ \{(w_i, x_i) \mid i \in J\}\uparrow \Rightarrow \{e_i \mid i \in J\} \in \mathrm{Con}_1$ &

$2(ii)$ $\quad \forall i, j \in I.\ w_i \uparrow w_j$ & $x_i \uparrow x_j$ & $e_i = e_j \Rightarrow w_i = w_j$ & $x_i = x_j$.

Assume $\{(w_i, (x_i, e_i)) \mid i \in I\} \in \mathrm{Con}_F$. We show it follows that $\{((w_i, x_i), e_i)) \mid i \in I\} \in \mathrm{Con}_P$. Because $1(i)$ holds $2(i)$ follows directly. To show $2(ii)$, assume $w_i \uparrow w_j$ and $x_i \uparrow x_j$ and $e_i = e_j$. By $1(i)$, $\{(x_i, e_i), (x_j, e_j)\} \in \mathrm{Con}$. Therefore $x_i = x_j$ by the property of consistency in a function space. Now $(x_i, e_i) = (x_j, e_j)$ so $w_i = w_j$ by $1(ii)$.

Assume $\{((w_i, x_i), e_i)) \mid i \in I\} \in \mathrm{Con}_P$. We show it follows that $\{(w_i, (x_i, e_i)) \mid i \in I\} \in \mathrm{Con}_F$. $1(ii)$ follows directly from $2(ii)$. We show $1(i)$. Let $J \subseteq I$ and suppose $\{w_i \mid i \in J\}\uparrow$. We need $\{(x_i, e_i) \mid i \in J\} \in \mathrm{Con}_2$. But this is proved as follows: Let $K \subseteq J$. If $\{x_i \mid i \in K\}\uparrow$ then $\{(w_i, x_i) \mid i \in K\}\uparrow$ so $\{e_i \mid i \in K\} \in \mathrm{Con}_1$ by $2(i)$. If $x_i \uparrow x_j$ and $e_i = e_j$, for $i, j \in J$, then, because $w_i \uparrow w_j$ too, by $2(ii)$ we obtain $x_i = x_j$.

Thus the correspondence preserves and reflects consistency. It also preserves and reflects entailment:

$$\begin{aligned}\{(w_i, (x_i, e_i)) \mid i \in I\} \vdash_f (w, (x, e)) &\Leftrightarrow \{(x_i, e_i) \mid w_i \subseteq w\} \vdash_2 (x, e)\\ &\Leftrightarrow \{e_i \mid w_i \subseteq w \ \ \& \ \ x_i \subseteq x\} \vdash_1 e\\ &\Leftrightarrow \{((w_i, x_i), e_i) \mid i \in I\} \vdash_p ((w, x), e).\end{aligned}$$

∎

Certainly $\mathbf{E}_{stab}$ has products including the null event structure as terminal object. The above results yield a natural 1–1 correspondence between morphisms $\mathrm{E}_0 \oplus \mathrm{E}_1 \to \mathrm{E}_2$ and $\mathrm{E}_0 \to [\mathrm{E}_1 \to \mathrm{E}_2]$ and so show that $\mathbf{E}_{stab}$ is cartesian closed [Mac. p.95–96]. We show the exponentiation more explicitly.

4.2.10 Theorem. *The category $\mathbf{E}_{stab}$ is cartesian closed. It has products as shown and an exponentiation of two stable event structures E_0 and E_1 has the form $[E_0 \to E_1]$, ap where ap : $[E_0 \to E_1] \oplus E_0 \to E_1$ is given by*

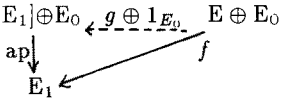
$$\mathrm{ap}(f, x) = (\phi f)(x)$$

for $f \in \mathcal{F}([E_0 \to E_1])$ and $x \in \mathcal{F}(E_0)$.
(We have identified (f, x) with the corresponding configuration of the disjoint product.)

Proof. By the preceding remarks the category $\mathbf{E}_{stab}$ is cartesian closed. Alternatively, this is shown by the following explicit demonstration of an exponentiation of two stable event structures. Let E_0 and E_1 be event structures. For ap as defined above we see

$$\mathrm{ap}(f, x) = \{e \in E_1 \mid \exists x' \subseteq x.\ (x', e) \in f\}$$

for $f \in \mathcal{F}([\mathrm{E}_0 \to \mathrm{E}_1])$ and $x \in \mathcal{F}(\mathrm{E}_0)$. The function ap is easily checked to be continuous and stable. In order for $[\mathrm{E}_0 \to \mathrm{E}_1]$, ap to be an exponentiation it is required that for any morphism $f : \mathrm{E} \oplus \mathrm{E}_0 \to \mathrm{E}_1$ there is a unique morphism $g : \mathrm{E} \to [\mathrm{E}_0 \to \mathrm{E}_1]$ such that the following diagram commutes:



Let $\mathrm{ab} : [\mathrm{E} \oplus \mathrm{E}_0 \to \mathrm{E}_1] \cong [\mathrm{E} \to [\mathrm{E}_0 \to \mathrm{E}_1]]$ be the isomorphism

$$\mathrm{ab} : F \mapsto \{(w, (x, e)) \mid ((w, x), e) \in F\}.$$

provided by the previous lemma. Take $g = \phi\ \mathrm{ab}\ \mu(f)$. This ensures g is a morphism. Then, recalling definitions,

$$g(w) = \{(x', e) \mid \exists w' \subseteq w.\ (w', (x', e)) \in \mathrm{ab}\mu(f)\}$$

for any configuration w of E. Hence

$$\begin{aligned}\mathrm{ap}(g(w), x) &= (\phi g(w))(x)\\ &= \{e \mid \exists w' \subseteq w, x' \subseteq x.\ ((w', x'), e) \in \mu(f)\}\\ &= (\phi\mu(f))(w, x)\\ &= f(w, x).\end{aligned}$$

This establishes the existence of g making the diagram commute. Uniqueness follows as if g' also makes the diagram commute then $(\phi g(w))x = (\phi g'(w))x$ for all w, x. But then $\phi g(w) = \phi g'(w)$ for all w. As ϕ is 1–1, $g(w) = g'(w)$ for all w. Hence $g = g'$. ∎

In the traditional function space used in denotational semantics the functions in the function space $[D \to E]$, where D and E are domains are ordered pointwise, *i.e.* two continuous functions f, g are ordered by

$$f \sqsubseteq g \Leftrightarrow \forall d \in D.\ f(d) \sqsubseteq g(d).$$

This ordering is called the *extensional* (or *Scott*) order. The inclusion order on the configurations of $[\mathrm{E}_0 \to \mathrm{E}_1]$ induces another order on stable, continuous functions $(\mathcal{F}(\mathrm{E}_0), \subseteq) \to (\mathcal{F}(\mathrm{E}_1), \subseteq)$ which we have seen can be expessed as

$$f \leq g \Leftrightarrow \mu f \subseteq \mu g.$$

This order is called the *stable* order (a name due to Berry). We give an example.

4.2.11 Example. The two point domain O consisting of $\perp \sqsubseteq T$ can be represented as the the configurations of the obvious event structure with a single event $\bullet$, so $\perp = \emptyset$ and $T = \{\bullet\}$. All the monotonic functions $\mathrm{O} \to \mathrm{O}$ are stable and continuous. Ordered extensionally they are

$$(\lambda x.\perp) \sqsubseteq (\lambda x.\ x = T \to T|\perp) \sqsubseteq (\lambda x.T)$$

while according to the stable ordering we only have

$$(\lambda x.\perp) \leq (\lambda x.\ x = T \to T|\perp) \quad \text{and} \quad (\lambda x.\perp) \leq (\lambda x.T),$$

because $(\lambda x.\ x = T \to T|\perp) \not\leq (\lambda x.T)$. For two functions to be in the stable order it is not only necessary that they are ordered extensionally but also that if they both output a value for common input then they do so for the same minimal value.

As an example we indicate how the category can be used to give a model for a λ–calculus with atoms.

4.2.12 Example. We can use the sum construction on event structures (it is a functor on $\mathbf{E}_{stab}$) and a constant event structure $\mathbf{A}$ of atomic events to define an operation

$$\mathrm{E} \mapsto \mathbf{A} + [\mathrm{E} \to \mathrm{E}].$$

This operation is $\leq$–continuous, being the composition of continuous things, and so has a least fixed point which can serve as a model for the λ–calculus with atoms following standard lines.

We return briefly to the problem of full abstraction. Even more simply than in the example above, we can give a denotational semantics to any typed λ–calculus including PCF; function types are interpreted as the stable function space. Because "parallel or" is not stable we have succeeded in eliminating it from the function spaces. However the model is not fully abstract—far from it. It now includes functions which are not monotonic with respect to the Scott order (in Berry's terms the model is not order extensional [B]) and these elements, like "parallel or" are not definable in PCF, and cause similar difficulties. Berry realised a fairly simple way to eliminate such non–order extensional functions. The basic idea was to work with *bidomains* which carry the extra structure of the Scott order which can then be used to cut down the functions allowed in the function space (see [W] for another approach based on event structures). Certainly this leads to a much more refined model of PCF than one based on the Scott function space but the fact remains that there are finite stable functions which are not definable in PCF. So at this point Berry and P.L.Curien, his student at the time, embarked on the study of sequentiality at higher type; they hoped to proceed

by analogy with Berry's work on stable functions and bidomains. Unfortunately, while this work did take exciting turns (the results are reported in [C]), it did not yield a fully abstract model. The full abstraction problem is still open.

Recent work of Girard has pointed the way to another appplication for the category of event structures with stable functions, or the equivalent category of dI–domains. In [G], Girard works with a proper subcategory of $\mathbf{E}_{stab}$ with objects called *qualitative domains* and shows how they give a model to his System F, the polymorphic λ-calculus. From the point of view of semantics qualitative domains are a little too restrictive because they are not closed under the useful operations of lifting (prefixing) or separated sum. However Girard's ideas can be extended to $\mathbf{E}_{stab}$ which is (see [CGW]).

Acknowledgements

I am grateful for discussions with Mogens Nielsen and Gordon Plotkin, and to the anonymous referee for suggested improvements.

References

[A] Aczel, P., A note on Scott's theory of domains. Unpublished note, Math. Dept., Univ. of Manchester, (1983).

[Ac] Aczel, P., An introduction to inductive definitions. In the handbook of Mathematical Logic, Ed. Barwise, J., North–Holland (1983).

[B] Berry, G., Modèles complètement adéquats et stables des lambda–calculs typés. Thèse de Doctorat d'Etat, Université de Paris VII (1979).

[Bk] Brookes, S.D., On the relationship of CCS and CSP. ICALP 1983, in Springer–Verlag Lecture Notes in Comp. Sc., vol.154 (1984).

[Br] Genrich, H.J., Lautenbach, K., and Thiagarajan, P.S., Elements of general net theory. In Net Theory and Applications, (Ed. Brauer, W.), Springer–Verlag Lecture Notes in Comp. Sci., vol.84 (1980).

[C] Curien, P.L., Categorical combinators, sequential algorithms and functional programming. Research notes in theoretical comp. sc., Pitman, London (1986).

[29] Coquand, T., Gunter, C., and Winskel, G., Polymorphism and domain equations. Submitted to Third Workshop on the Mathematical Foundations of Programming Language Semantics, New Orleans, LA 1987.

[F] Fogh, T., En semantik for synkroniserede parallelle processer. Master's thesis, Comp. Sc., Aarhus Univ., Denmark (1981).

[G] Girard, J.Y., The system F of variable types, fifteen years later. Manuscript, (1985).

[GR] Goltz, U. and Reisig, W., Processes of Place/Transition Nets. Icalp 83 and appears in Information and Control (1984).

[He] Hewitt, C., and Baker, H., Actors and continuous functionals. In "Formal description of programming concepts (ed. E.Neuhold), North Holland (1978).

[H] Hoare, C.A.R., Communicating sequential processes. Prentice Hall (1985)

[HBR] Hoare, C.A.R., Brookes, S.D., and Roscoe, A.W., A Theory of Communicating Processes, Technical Report PRG-16, Programming Research Group, University of Oxford (1981); in JACM (1984).

[KP] Kahn, G., and Plotkin, G., Domaines Concrètes. Rapport IRIA Laboria No. 336 (1978).

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

[LW] Larsen, K., and Winskel, G., Using information systems to solve recursive domain equations effectively. In the proceedings of the conference on Abstract Datatypes, Sophia-Antipolis, France in June 1984. Full version submitted to the journal "Information and Control" and appears as a report of the Computer Laboratory, University of Cambridge (1983).

[LC] Lauer, P. E. and Campbell, R. H., Formal semantics for a class of high–level primitives for coordinating concurrent processes. Acta Informatica 5 pp.297–332 (1974).

[Maz] Mazurkiewicz, A., Concurrent program schemes and their interpretations. Report PB–78 of the Computer Sc. Dept., University of Aarhus, Denmark (1977).

[Mac] Maclane, S., Categories for the Working Mathematician. Graduate Texts in Mathematics, Springer (1971).

[M] Milner, R., Fully abstract models of typed lambda–calculi. Theor. Comp. Sc., vol.4(1), 1–23 (1977).

[M1] Milner, R., A Calculus of Communicating Systems. Springer Lecture Notes in Comp. Sc. vol. 92 (1980).

[M2] Milner, R., Calculi for synchrony and asynchrony. Theoretical Computer Science 25, pp.267–310 (1983).

[MS] Montanari, U., and Simonelli, C., On distinguishing between concurrency and nondeterminism. Proc. Ecole de Printemps on Concurrency and Petri nets, Colleville (1980).

[NPW] Nielsen, M., Plotkin, G., Winskel, G., Petri nets, Event structures and Domains, part 1 . Theoretical Computer Science, vol. 13 (1981).

[P] Plotkin, G.D., LCF considered as a programming language. Theor. Comp. Sc., vol.5(3), 223–256 (1977).

[Pe] Petri, C.A., Nonsequential processes. GMD–ISF Report ISF–77–05 (1977).

[Pr] Pratt, V. R., On the composition of processes. Proc. of the 9th annual ACM symposium on Principles of Programming Languages, (1982).

[S] Scott, D. S., Domains for Denotational Semantics. ICALP '82. Springer–Verlag Lecture Notes in Comp. Sc. 140 (1982).

[S1] Scott, D. S., Lectures on a mathematical theory of computation. Oxford University Computing Laboratory Technical Monograph PRG–19 (1981).

[Sh1,2] Shields, M., Non–sequential behaviours: 1 and 2. Reports of the Comp. Sc. Dept., University of Edinburgh (part 1: 1982, part 2: 1983).

[Smy] Smyth, M.B., The largest cartesian closed category of domains. Theor. Comp. Sc., vol. 27 pp. 109–119 (1983).

[St] Stoy, J. Denotational semantics: The Scott–Strachey approach to programming language theory. MIT Press (1977).

[W] Winskel, G., Events in Computation. Ph.D. thesis, available as a technical report, Comp. Sc. Dept., University of Edinburgh (1980).

[W1] Winskel, G., Event structure semantics of CCS and related languages. Proc. ICALP '82. Springer–Verlag Lecture Notes in Comp. Sc. 140 and as a report of the Computer Sc. Dept., University of Aarhus, Denmark (1982).

[W2] Winskel, G., A representation of completely distributive algebraic lattices. Report of the Computer Science Dept., Carnegie-Mellon University (1983).

[W3] Winskel, G., Synchronisation trees. In Theoretical Computer Science, May 1985.

[W4] Winskel, G., A New Definition of Morphism on Petri Nets. Springer Lecture Notes in Comp Sc, vol. 166 (1984).

[W5] Winskel, G., Categories of Models for Concurrency. In the proceedings of the workshop on the semantics of concurrency, Carnegie-Mellon University, Pittsburgh, Springer Lecture Notes in Computer Science 197 (July 1984), and appears as a report of the Computer Laboratory, University of Cambridge (1984).

[W6] Winskel, G., Petri nets, algebras, morphisms and compositionality. Report 79 of the Computer Laboratory , University of Cambridge. To appear in Information and Control. An extended abstract appears in "Advances in Petri Nets", Springer–Verlag Lecture Notes in Comp. Sc. (1985).