Configuration Structures
(extended abstract)

R.J. van Glabbeek*
Computer Science Department
Stanford University
Stanford, CA 94305, USA
rvg@cs.stanford.edu

G.D. Plotkin*
Department of Computer Science
University of Edinburgh
Edinburgh EH9 3JZ, UK
gdp@dcs.ed.ac.uk

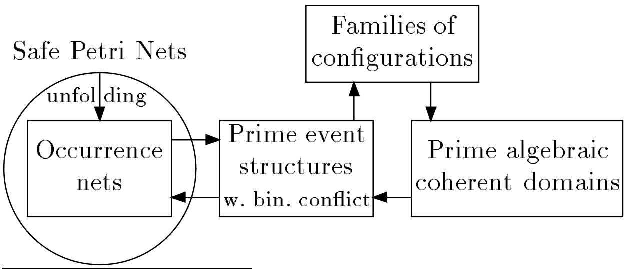
Abstract

Configuration structures provide a model of concurrency generalising the families of configurations of event structures. They can be considered logically, as classes of propositional models; then sub-classes can be axiomatised by formulae of simple prescribed forms. Several equivalence relations for event structures are generalised to configuration structures, and also to general Petri nets. Every configuration structure is shown to be ST-bisimulation equivalent to a prime event structure with binary conflict; this fails for the tighter history preserving bisimulation. Finally, Petri nets without self-loops under the collective token interpretation are shown behaviourally equivalent to configuration structures, in the sense that there are translations in both directions respecting history preserving bisimulation. This fails for nets with self-loops.

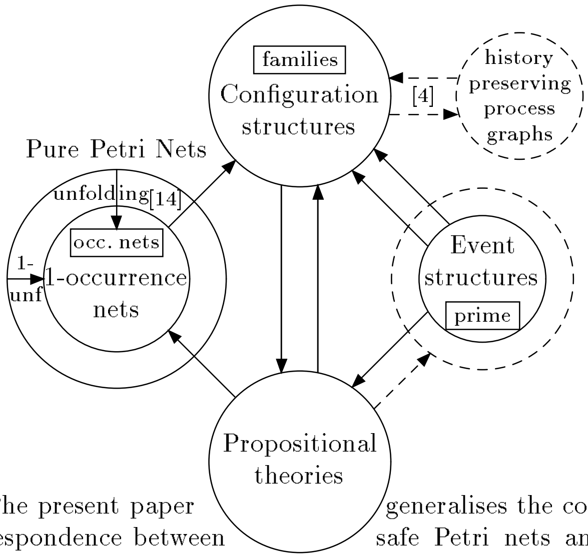
1 Introduction

The aim of this paper is to connect several models of concurrency, by providing translations between them and studying which notions of behavioural equivalence these translations preserve.

In Nielsen, Plotkin & Winskel [16] *event structures* were introduced as a stepping stone between *Petri nets* and *Scott domains*. It was established that every *safe* Petri net can be unfolded into an *occurrence net*; the occurrence nets are then in correspondence with event structures; and they in turn correspond bijectively with *prime algebraic coherent Scott domains*.



In Winskel [22] a more general notion of event structure was proposed, corresponding to a more general kind of Scott domain. The event structures from [16] are now called *prime event structures with binary conflict*. The translation from event structures to domains passes through a stage of *families of configurations of event structures*. Van Glabbeek & Goltz [6] found it convenient to use such families as a model of concurrency in their own right. In this context the families were called *configuration structures*.



The present paper generalises the correspondence between safe Petri nets and configuration structures to unsafe nets. For this purpose we use a more general kind of configuration structure, the *set systems*. These have an attractive alternative presentation as propositional theories. As event structures are insufficiently expressive, they are skipped as an intermediate step in the translation. In the full version of this paper they will be generalised to match all configuration structures. The connection between configuration structures and Scott domains is generalised in Van Glabbeek [4], where *history preserving process graphs* are considered as an alternative presentation of (labelled) domains.

*This work was supported by ONR under grant number N00014-92-J-1974.

Configuration structures A *set system* is given by a set E and a collection C of subsets of E. When a set system is used to represent a concurrent system, we call it a *configuration structure*. The elements of E are then *events* and the elements of C *configurations*. An event represents an occurrence of an action the system may perform; a configuration X represents a state of the system, namely the state in which the events in X have occurred. The configuration structures of [6] were required to satisfy several requirements, due to [22], ensuring that they could be obtained as the families of finite configurations of event structures, namely

- all configurations are finite (*finitariness*),
- C contains the empty configuration (*rootedness*),
- C is closed under nonempty bounded unions ($\overline{\bigcup_\bullet}$),
- and for any two distinct events occurring in a configuration there is a subconfiguration containing one but not the other (*coincidence-freeness*).

In the present paper a more expressive kind of configuration structure is considered, not bound by these requirements. Many of our results however concern finitary rooted configuration structures. A further generalisation of this model was previously proposed by PINNA & POIGNÉ [17]. Their *event automata* are rooted finitary configuration structures together with a transition relation between the configurations.

Our configuration structures are, up to isomorphism, the *extensional Chu spaces* of GUPTA & PRATT [11, 10, 20]. It was in their work that the idea arose of using the full generality of such structures in modelling concurrency. It should be noted however that the computational interpretation in [11, 10, 20] differs slightly from that in [16, 22, 6, 17].

Formulae In Section 3 we consider set systems from the point of view of (infinitary) propositional logic: E is now thought of as the set of *propositions* and C as the set of *models*. Following PRATT [20] we observe a bijective correspondence between configuration structures and infinitary propositional theories up to logical equivalence. We give a number of results equating the closure of C under certain operations with its axiomatisation by formulae of certain simple forms.

Event structures The behaviour of event structures is traditionally described in terms of their configurations: the consistent and secured sets of events. In Section 4 we relate this description with one in terms of consistent and left-closed set of events. The later corresponds with a *logical* view of event structures as propositional theories. A somewhat different logical view was presented in GUNAWARDENA [9]. The precise relations with ours are yet to be investigated.

In Section 4.1 we classify event structures along three lines and discuss the characterisation of the associated classes of configuration structures.

Equivalences A model of computation should have a notion of behaviour, and the possibility of comparing behaviours with respect to suitable equivalence relations. To this end, in Section 2, we equip configuration structures with functions $l : E \to Act$ *labelling* events with *actions*, and derive associated *asynchronous* labelled transition relations. We consider various *bisimulation* relations, adapted for concurrency, between the resulting labelled transition graphs. Section 4.2 considers several classes of event structures within this framework. Whether previously known classes of event structures are as general as arbitrary configuration structures depends on the notion of equivalence used for the comparison. Theorems 1 and 2 are noteworthy in this respect; they imply that the original event structures of [16] are *ST-bisimulation* universal.

Petri nets In MESEGUER, MONTANARI & SASSONE [14], arbitrary (non-safe) Petri nets are unfolded into occurrence nets. Their unfolding generalises the one of [16] and preserves the behaviour of nets under a particular interpretation due to GOLTZ & REISIG [8]. We call this interpretation the *individual token interpretation*. An alternative way of understanding the behaviour of nets is the *collective token interpretation*. In the latter view there are nets which cannot be represented by an event structure, let alone by a prime one with binary conflict, or an occurrence net.

In Section 5 we establish a connection between *pure nets* and configuration structures. Pure nets are nets without *self-loops*. We *1-unfold* pure nets into *pure 1-occurrence nets*, which generalise the occurrence nets of [16]. These pure 1-occurrence nets are shown to correspond with rooted finitary configuration structures.

Through the translation into configuration structures the equivalence notions on such structures are inherited by pure 1-occurrence nets; we generalise them to all nets. The correspondence between pure 1-occurrence nets and configuration structures preserves the identity of events and configurations (*configuration equivalence*). The 1-unfolding preserves any reasonable form of *history preserving bisimulation equivalence* [5] on nets, thus making any pure net history preserving equivalent to a configuration structure. Moreover, *any* net is ST-bisimulation equivalent to such a structure (and hence to a prime event structure with binary conflict). In contrast, we find a Petri net *not* history preserving equivalent (in an appropriate sense) to any configuration structure.

In future work we would like to connect our models with appropriate versions of *higher dimensional automata* [19]. One could also consider other equivalence relations appropriate for the study of concurrency, such as those based on notions of multiple observers [18]. In work on event structures WINSKEL and others have employed various notions of morphism [22]; yet another notion occurs in the category of Chu spaces. It seems likely to be of importance to understand the relation with the present equivalence-based approach.

Acknowledgment We thank Vladimiro Sassone for helpful comments.

2 The computational interpretation of configuration structures

In order to interpret a configuration structure as a concurrent system, it is necessary to know, not only what are the admissible states, but also how the system can evolve from one state to the other.

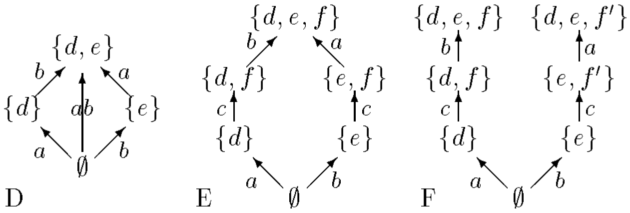
Definition 1 Let (E, C) be a configuration structure. For X, Y in C write $X \longrightarrow Y$ if $X \subset Y$, $Y - X$ is finite, and $\forall Z(X \subset Z \subset Y \Rightarrow Z \in C)$. The relation $\longrightarrow$ is called the *step transition relation*.

Here $X \longrightarrow Y$ indicates that the represented system can go from state X to state Y by concurrently performing a number of events (namely the ones in $Y - X$). The first requirement is unavoidable. The second one represents our assumption that in a finite time only finitely many events can happen. The last requirement says that a number of events can be performed concurrently, or simultaneously, only if they can be performed in any order. This requirement represents our postulate that different events do not synchronise in any way; they can happen in one step only if they are causally independent. Hence our transition relation $\longrightarrow$ and the corresponding computational interpretation of configuration structures could be called *asynchronous*. It should be noted that other computational interpretations of configuration structures are possible. The one of GUPTA & PRATT [11, 10, 20] is obtained by dropping the last two requirements in Definition 1. By labelling the events, we may observe transitions:

Definition 2 A *labelled* configuration structure (over an alphabet Act) is a triple $\mathrm{C} = (E, C, l)$ with (E, C) a configuration structure and $l : E \rightarrow Act$.

The components of such a structure C are denoted E_{C}, C_{C} and l_{C} respectively (a convention that also applies to other structures given as tuples). The *labelled transition relation* $X \xrightarrow{L} Y$, with L a finite multiset of labels, holds iff $X \longrightarrow Y$ and $L = \sum\{l(e) | e \in (Y - X)\}$.

Example 1 These are the labelled transition relations for $\mathrm{D} = (\{d, e\}, \{\emptyset, \{d\}, \{e\}, \{d, e\}\}, l)$ and $\mathrm{E} = (\{d, e, f\}, \{\emptyset, \{d\}, \{e\}, \{d, f\}, \{e, f\}, \{d, e, f\}\}, l)$, both with $l(d) = a$, $l(e) = b$ and $l(f) = c$, and for a structure F.



Such pictures of configuration structures are somewhat misleading representations, as they suggest a notion of global time, under which at any time the represented system is in one of its states, moving from one state to another by following the transitions. Although this certainly constitutes a valid interpretation, we favour a more truly concurrent view, in which all events can be performed independently, unless the absence of certain configurations indicates otherwise. Under this interpretation, the configurations can be thought of as *possible* states the system can be in, *from the point of view of a possible observer*. They are introduced only to indicate (by their absence) the dependencies between events in the represented system.

In particular, in the structure D above, the events d and e are completely independent, and there is no need to assume that they are performed either simultaneously or in a particular order. The "diagonal" in the picture serves merely to remind us of the independence of these events. In terms of higher dimensional automata [19] it indicates that "the square is filled in".

On the other hand, the absence of any "diagonals" in E indicates two distinct linearly ordered computations. In one the event f can only happen after event d, and e in turn has to wait for f; the other has a causal ordering $e < f < d$. There is no way to view d and e as independent; if there were, there should be a transition $\emptyset \xrightarrow{ab} \{d, e\}$.

If we do not care about the order of events d and e in configuration structure D, we can say that the configuration $\{d, e\}$ has only one *(concurrent) history*. The configuration $\{d, e, f\}$ of structure E has two histories. In order to count these histories formally, we introduce a concept of (monoidal) *homotopy* between the paths in the graph representations of configuration structures.

Definition 3 A *path* in a configuration structure C is a non-empty sequence of configurations $X_0X_1\cdots X_n$, such that there is a transition $X_{i-1} \longrightarrow X_i$ for $i = 1,...,n$. *Homotopy* is the smallest equivalence on the paths of C such that if a path π can be obtained by deleting a configuration from another path ρ (this not being the first or the last one in ρ) then π is equivalent to ρ. A configuration X is *reachable* if there is a path $\emptyset X_1\cdots X_n$ with $X_n = X$. A *history* of a configuration is a homotopy class of such paths.

Let $\mathcal{R}$ be the operation that deletes all unreachable configurations from configuration structures. The *reachable part* $\mathcal{R}(\mathrm{C})$ of structure C is always finitary and coincidence-free. Under the computational interpretation of Definition 1 this is the only part that matters. But, when possible, we will also take unreachable configurations into account in connecting models of concurrency, in order to accommodate computational interpretations such as the one of Gupta and Pratt.

Equivalence relations In order to abstract from certain details in the representation of concurrent systems we consider a range of equivalence relations on configuration structures. VAN GLABBEEK & GOLTZ [5] suggest *history preserving bisimulation equivalence*, defined on event structures in terms of their configurations, as the coarsest equivalence completely capturing the interplay of causality and branching. The most straightforward generalisation of this notion to general configuration structures is the following.

Definition 4 Two rooted labelled conf. structures C and D are *configuration preserving bisimulation equivalent* if there is a relation $B \subseteq C_{\mathrm{C}} \times C_{\mathrm{D}} \times 2^{E_{\mathrm{C}} \times E_{\mathrm{D}}}$ such that $(\emptyset,\emptyset,\emptyset) \in B$ and whenever $(X,Y,f) \in B$ then

- f is a bijection between X and Y preserving labelling and subconfigurations (an *isomorphism*),
- $X \xrightarrow{L} X' \in C_{\mathrm{C}} \Rightarrow \exists Y', f'. \begin{cases} Y \xrightarrow{L} Y' \wedge f' \upharpoonright X = f \\ \wedge (X',Y',f') \in B, \end{cases}$
- $Y \xrightarrow{L} Y' \in C_{\mathrm{C}} \Rightarrow \exists X', f'. \begin{cases} X \xrightarrow{L} X' \wedge f' \upharpoonright X = f \\ \wedge (X',Y',f') \in B. \end{cases}$

However, this equivalence makes more distinctions than necessary to capture causality and branching: it distinguishes structures E and F above. Therefore, in the full version, we define *history preserving bisimulation equivalence* to be a coarser equivalence, also generalising the relation from [5], which identifies systems like E and F. Its definition is similar, but with histories playing the rôle of configurations. Even coarser equivalences are *step-bisimulation* (as above, but without the isomorphisms f) and *interleaving bisimulation equivalence* (similar, but using only *single-action* transitions, in which L is a singleton). Between step and history preserving bisimulation we have *ST-bisimulation equivalence* [7], based on a notion of state in which events may have been partly executed.

Definition 5 An *ST-configuration* is a pair (S,T) of configurations with $T \longrightarrow S$. The elements of S are the events that have *started*, whereas T contains the ones that have *terminated*. The transition relation between configurations extends to one between ST-configurations by putting $(S,T) \xrightarrow{L,K} (S',T')$ iff $S \xrightarrow{L} S'$, $T \xrightarrow{K} T'$ and $T \longrightarrow S'$. An *ST-map* between (S,T) and (U,V) is a bijection $f: S \rightarrow U$ with $f(T) = V$. ST-bisimulation equivalence is now defined as in Definition 4, but using ST-configurations and transitions instead of ordinary ones, and letting the f's be ST-maps.

Process graphs In [4] rooted finitary configuration structures are represented as labelled transition systems, or *process graphs*, by taking the configurations as states, the empty configuration as initial state, and the single-action transitions as transitions. In the graph representation of structure D from Example 1 the two a-transitions can be recognised as stemming from the same event (d) because they are opposites in a square. This cannot be said for structure E and certainly not for F. Let $\sim$ be the least equivalence on single-action transitions such that if $X \xrightarrow{a} U \xrightarrow{b} Y$, $X \xrightarrow{b} V \xrightarrow{a} Y$ and $U \neq V$, then $(X \xrightarrow{a} U) \sim (V \xrightarrow{a} Y)$. We say that a configuration structure has *recognisable* events if the evident map from equivalence classes to events is a bijection. If this is the case, the configuration structure, up to isomorphism, is completely determined by its graph.

3 Axiomatisation of set systems

In this section we consider set systems $\mathrm{C} = (E,C)$ from a logical point of view: E is thought of as a collection of *propositions* and C as the collection of *models*. Connecting with the computational point of view, we associate with an event the proposition that it has happened. There is an associated theory Φ_{C} of all valid sentences, those holding in all models; these are the laws of C.

To make this precise, we choose a language: infinitary propositional logic with E as the set of propositional variables, and closed under negation and all conjunctions of sets of formulae. A formula ϕ is *valid* in (E,C) iff it is true in all elements of C; Φ_{C} denotes the class of formulae valid in C. Equally, given a class

Φ of formulae over a set E, we can define a configuration structure $\mathrm{C}_\Phi = (E, \mathcal{M}(\Phi))$, where $\mathcal{M}(\Phi)$ is the set of *models* of Φ, those interpretations making every formula in Φ true. We say that Φ *axiomatises* C_Φ. In particular Φ_{C} axiomatises C for any set system C.

This point of view is due to Pratt [11, 20]. He noted a natural "conjunctive normal form." For any two subsets X,Y of E, let $X \Rightarrow Y$ abbreviate the "propositional sequent" $\bigwedge X \Rightarrow \bigvee Y$. Then for any (E, C), if Φ is the collection of sequents valid in (E, C) then $(E, C) = \mathrm{C}_\Phi$. Thus any configuration structure can be axiomatised by a set of propositional sequents.

We now consider correspondences between axiomatisations by classes of formulae and closure conditions on set systems. First, there is logical interest in *Scott sequents* where both antecedent and consequent are finite, and also in *Tarski sequents* where, further, the consequent is a singleton [3, 21].

Proposition 1 A set system (E, C) is Scott sequent axiomatisable iff C is closed in the product topology on 2^E. It is Tarski sequent axiomatisable iff C is closed under intersections and directed unions.

Next we consider closure conditions that arise naturally when considering configuration structures. We denote closure under non-empty intersections by $\bigcap_\bullet$, under bounded non-empty intersections by $\overline{\bigcap}_\bullet$, under bounded non-empty unions by $\overline{\bigcup}_\bullet$ and under directed unions by $\bigcup_\uparrow$. The results appear below. The

stable	$\overline{\bigcap}_\bullet$	$\equiv$	$(\mathrm{any}, \mathrm{ddc})$
	$\overline{\bigcap}_\bullet$, $\bigcup_\uparrow$	$\equiv$	$(\mathrm{finite}, \mathrm{ddc})$
	$\overline{\bigcap}_\bullet$, $\overline{\bigcup}_\bullet$	$\equiv$	$(1, \mathrm{ddc}), (\mathrm{any}, 0)$
	$\overline{\bigcap}_\bullet$, $\bigcup_\uparrow$, $\overline{\bigcup}_\bullet$	$\equiv$	$(1, \mathrm{ddc}), (\mathrm{finite}, 0)$
prime	$\bigcap_\bullet$	$\equiv$	$(\mathrm{any}, \leq 1)$
	$\bigcap_\bullet$, $\bigcup_\uparrow$	$\equiv$	$(\mathrm{finite}, \leq 1)$
	$\bigcap_\bullet$, $\overline{\bigcup}_\bullet$	$\equiv$	$(1, 1), (\mathrm{any}, 0)$
	$\bigcap_\bullet$, $\bigcup_\uparrow$, $\overline{\bigcup}_\bullet$	$\equiv$	$(1, 1), (\mathrm{finite}, 0)$
union	$\overline{\bigcup}_\bullet$	$\equiv$	$(1, \mathrm{any}), (\mathrm{any}, 0)$
	$\overline{\bigcup}_\bullet, \bigcup_\uparrow$	$\Leftarrow$	$(1, \mathrm{any}), (\mathrm{finite}, 0)$
	$\bigcup_\uparrow$	$\Leftarrow$	$(\mathrm{finite}, \mathrm{any})$

last column indicates the form of the allowed formulae which are all implications. For example: $(\mathrm{any}, \leq 1)$ indicates a sequent with no restriction on the antecedent and whose consequent has at most one element; and $(\mathrm{any}, \mathrm{ddc})$ indicates an implication whose antecedent can be any conjunction of propositional letters, and whose consequent has the form $\dot{\bigvee}_{j \in J} \bigwedge Z_j$ where the Z_j are sets of letters, and we write $\dot{\bigvee}\, \Phi$ for $(\bigvee \Phi) \wedge \bigwedge\{\neg(\phi \wedge \phi') \mid \phi, \phi' \in \Phi, \phi \neq \phi'\}$, the *disjoint disjunction* of Φ. So, for example the sixth entry states that a configuration structure is axiomatisable by sequents of the form $(\mathrm{finite}, \leq 1)$ iff it is closed under non-empty bounded intersections and directed unions; this is —essentially— due to Larsen and Winskel [13] as axiomatisations of the form $(\mathrm{finite}, \leq 1)$ correspond to Scott information systems. The entries with $\Leftarrow$'s indicate that only the implication from right to left holds; a counterexample is provided by the collection of all co-finite proper subsets of the natural numbers. A minor, but useful, variation, is to restrict to rooted configuration structures (i.e. containing $\emptyset$); then in the last column one changes "any" to "non-empty," and "finite" to "finite and non-empty" on the left of the implications (but not the right).

4 Event structures

Event structures were introduced by Winskel [22]:

Definition 6 A *(general) event structure* is a triple $\mathrm{E} = (E, Con, \vdash)$ where

- E is a set of *events*,
- $Con \subseteq \mathcal{P}_f(E)$ is a nonempty *consistency predicate* such that: $Y \subseteq X \in Con \Rightarrow Y \in Con$,
- and $\vdash \subseteq Con \times E$ is the *enabling relation*, which satisfies $X \vdash e \wedge X \subseteq Y \in Con \Rightarrow Y \vdash e$.

The idea behind the enabling relation is that an event e can happen only if a set X of events enabling e occurred previously; Con consists of the finite sets of events that, potentially, can occur during a single run of the system. Winskel explained the behaviour of event structures by associating with each E a family of configurations $\mathcal{S}(\mathrm{E})$, the possible states or runs of the system:

Definition 7 The set $\mathcal{S}(\mathrm{E})$[1] of *(secured) configurations* of an event structure $\mathrm{E} = (E, Con, \vdash)$ consists of those $X \subseteq E$ which are

- *consistent*: every finite subset of Y of X is in Con,
- and *secured*: $\forall e \in X.\ \exists e_0, \ldots, e_n \in X.\ e_n = e \wedge \forall i \leq n.\ \{e_0, \ldots, e_{i-1}\} \vdash e_i$.

Winskel [22] gave an *intrinsic* characterisation of the configuration structures of event structures. A family of configurations C is of the form $\mathcal{S}(\mathrm{E})$ iff C is non-empty and has the properties of *finite-completeness*, *finiteness* and *coincidence-freeness*. Here a family of configurations is finitely-complete iff it is closed under directed unions and bounded unions (or equivalently bounded finite unions); finiteness means that if

[1] We also write $\mathcal{S}(\mathrm{E})$ for the structure $(E, \mathcal{S}(\mathrm{E}))$.

an event occurs in a configuration, then it occurs in a finite subconfiguration. The properties of finiteness and closure under directed unions together say that the infinite configurations are precisely the directed unions of the finite ones; let us call this property $\bigcup_\uparrow$-*finitary.* Furthermore, the properties of non-emptiness and closure under bounded unions are equivalent to rootedness and closure under nonempty bounded unions ($\overline{\bigcup}_\bullet$). It follows that a set of configurations is the $\mathcal{S}$-image of an event structure iff it is $\bigcup_\uparrow$-finitary, rooted, $\overline{\bigcup}_\bullet$-closed and coincidence-free.

Instead of using all configurations to describe the behaviour of an event structure E, one can just as well restrict attention to the finite ones, $\mathcal{S}_f(\mathrm{E})$ (for $\mathcal{S}(\mathrm{E})$ can be recovered from $\mathcal{S}_f(\mathrm{E})$ by means of closure under directed unions). The corresponding characterisation has already been mentioned in the introduction. Note that finitariness and coincidence-freeness come for free when considering reachable parts of configuration structures.

We consider yet another interpretation of event structures as configuration structures, namely by using the *left-closed* configurations. These are the consistent sets of events X that satisfy $\forall e \in X.\ \exists Y \subseteq X.\ Y \vdash e$.

We show that the left-closed interpretation of event structures is almost the same as the secured interpretation. In order to rule out a pathological case where this does not hold, we define the *irreflexive* event structures to be the ones satisfying $X \vdash e \Rightarrow X - \{e\} \vdash e$. Note that every event structure E can be transformed into an irreflexive one with the same secured configurations by deleting unwanted enablings $X \vdash e$ with $e \in X$. Such a transformation preserving the left-closed configurations $\mathcal{L}(\mathrm{E})$ is obtained by adding the missing enablings $X - \{e\} \vdash e$ (when $X \vdash e$). Thus under both interpretations it is no essential limitation to consider irreflexive event structures only. The following proposition says that for such structures the left-closed interpretation contains more information than the secured interpretation.

Proposition 2 Let E be an irreflexive event structure. Then $\mathcal{S}_f(\mathrm{E}) = \mathcal{R}(\mathcal{L}(\mathrm{E}))$.

Under the computational interpretation of Definition 1 any such extra information is irrelevant and $\mathcal{L}$ and $\mathcal{S}$ can be regarded as equivalent.

With any event structure $\mathrm{E} = (E, Con, \vdash)$ we also associate the propositional theory

$$\begin{aligned}\Phi(\mathrm{E}) = \{&\textstyle\bigwedge X \Rightarrow \bot \mid X \in \mathcal{P}_f(E) - Con\} \\ &\cup \{e \Rightarrow \textstyle\bigvee\{\bigwedge Y \mid Y \vdash e\}\}.\end{aligned}$$

This logical view of event structures corresponds exactly with the left-closed interpretation:

Proposition 3 $\mathrm{C}_{\Phi(\mathrm{E})} = \mathcal{L}(\mathrm{E})$ for any E.

We have no exact characterisation of the structures of the form $\mathcal{L}(\mathrm{E})$; however, a structure is of the form $\mathcal{L}_f(\mathrm{E})$ (the *finite* left-closed configurations of an event structure) iff it is finitary, rooted and $\overline{\bigcup}_\bullet$-closed.

4.1 Other brands of event structures

One can classify general event structures along three dimensions: stability, having recognisable events and degree of conflict. A further subclass of *synchronisation trees* [15] will also be considered. Yet other classes have been considered in the literature, for example *bundle* event structures (Langerak [12]) and *flow* event structures (Boudol [2] and Castellani).

Definition 8 E is *stable* iff $X \vdash_\mathrm{E} e$, $Y \vdash_\mathrm{E} e$ and $X \cup Y \cup \{e\} \in Con_\mathrm{E}$ imply $X \cap Y \vdash_\mathrm{E} e$.

Winskel defined the stable event structures in [22]. The causal dependencies between the events in a configuration of such a structure can be given by a partial order. The intrinsic characterisation of the configuration structures of stable event structures consists of the same requirements as above, together with closure under nonempty bounded intersections (*stability*). Stability is equivalent to an interesting *local completion principle*: if $X \subseteq Y \in C$ then there is a least Y' in C with $X \subseteq Y' \subseteq Y$.

Say that an event structure E has recognisable events iff the associated configuration structure $\mathcal{S}(\mathrm{E})$ does. We do not possess a particularly pleasing description of those event structures with recognisable events, or of their associated configuration structures. However, we do in the case where they are stable.

Definition 9 E is *prime* iff for all e in E, e occurs in an $\mathcal{S}$-configuration, and there is a least X with $X \vdash e$.

This is (configuration) equivalent to the original definition in [22]. Prime event structures possess a transitive *global causal dependency* relation, where $e \leq e'$ iff $e \in X$ where X is the least configuration containing e'.

Proposition 4 An event structure has the same configurations as a prime event structure iff it has the same configurations as a stable one with recognisable events.

The configuration structures associated to prime event structures can be characterised by strengthening stability to the requirement that (E, C) be *prime*, meaning that C is closed under non-empty intersections. Primality is equivalent to a *global* completion principle: if $X \subseteq Y \in C$ then there is a least Y' in C with $X \subseteq Y'$.

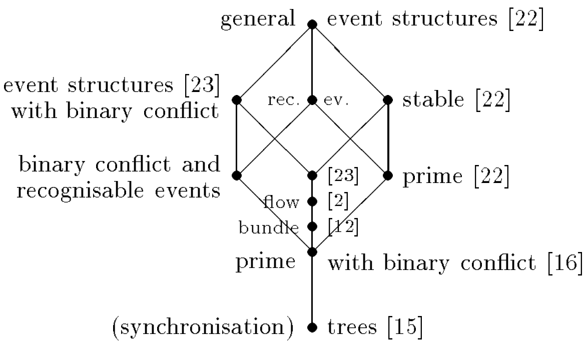
The other dimension of variation is the degree of consistency. We consider one possibility:

Definition 10 E has *binary conflict* if every X not in Con has a subset not in Con with two elements.

See [23] for a direct definition, and also for the stable case. The prime event structures with binary conflict are (equivalent to) the original event structures from Nielsen, Plotkin and Winskel [16]. They are characterised by the requirements for general event structures plus primality and *coherence*, that for all X,Y,Z in C if each of $X \cup Y, Y \cup Z, Z \cup X$ is in C, then $X \cup Y \cup Z$ is too. It was noted by Arend Rensink that this does not extend to stable event structures.

Finally we consider *synchronisation trees.* These are the prime event structures with binary conflict such that for any two consistent events e,e' either $e \leq e'$ or $e' \leq e$. This is equivalent to the definition in [15]. Their configuration structures can be characterised by the extra requirement of linearity: if X and Y are a bounded pair of configurations, then $X \subseteq Y$ or $Y \subseteq X$.

Up to configuration equivalence, the classes of structures of this section are related by inclusion as indicated below. For the sake of completeness, the flow and bundle event structures are indicated as well.



4.2 The difference between the brands of structures up to equivalence notions

One defines *labelled* event structures, and their associated labelled configuration structures, in the evident way. From here onwards we assume all structures to be labelled. The equivalence relations from Section 2 are inherited by (labelled) irreflexive event structures via $\mathcal{L}$. We now turn to comparing classes of event structures and configuration structures with respect to our "test range" of equivalences. We only state our results; the proofs will appear in the full version of this paper.

Proposition 5 Every rooted configuration structure is interleaving bisimulation equivalent to a synchronisation tree.

Since interleaving bisimulation and synchronisation trees were introduced when considering concurrency as interleaving, this proposition may not be surprising. More of a surprise may be that, as regards ST-bisimulation, the prime event structures are universal.

Theorem 1 Every rooted configuration structure is ST-bisimulation equivalent to a prime event structure.

We can reduce yet further, to *binary* conflict:

Theorem 2 Every (prime) event structure is history preserving bisimulation equivalent to an event structure (resp., prime event structure) with binary conflict.

Putting these together, we see that every configuration structure is ST-bisimulation equivalent to a prime event structure with binary conflict. On the other hand, there is a configuration structure not history-preserving bisimulation equivalent to any event structure. In the terminology of the Section 3, it is the one with three events a,b,c axiomatised by $a \wedge b \Rightarrow c$.

5 Petri nets

Definition 11
A (labelled) *Petri net* is a tuple (S, T, F, K, M_0, l) with

- S and T two disjoint sets of *places* and *transitions*,
- $F : (S \times T \cup T \times S) \rightarrow \mathbb{N}$, the *flow relation*,
- $K : S \rightarrow \mathbb{N} \cup \{\infty\}$, the *capacity allocation*,
- $M_0 : S \rightarrow \mathbb{N}$, the *initial marking*, satisfying $M_0(s) \leq K(s)$ for $s \in S$.
- and $l : T \rightarrow Act$, the *labelling function*.

Petri nets are pictured by drawing the places as circles (subscripted by '$\kappa = K(s)$' if they have finite capacity) and the transitions as boxes, containing their label. Between s in S and t in T there are $F(s,t)$ *arcs*.

When a Petri net represents a concurrent system, a global state of such a system is given as a *marking*, a function $M : S \rightarrow \mathbb{N}$ satisfying $M(s) \leq K(s)$ for $s \in S$. Such a state is depicted by placing $M(s)$ dots (*tokens*) in each place s. The initial state is given by the marking M_0. In order to describe the behaviour of a net, we describe the the step transition relation between markings.

Definition 12 Let $M : S \rightarrow \mathbb{N}$ be a marking of a net. A finite multiset $U : T \rightarrow \mathbb{N}$ of transitions is *enabled* under M if $\Sigma_{t \in T} U(t) \cdot F(s,t) \leq M(s)$ and $M'(s) = M(s) + \Sigma_{t \in T} U(t) \cdot (F(t,s) - F(s,t)) \leq K(s)$ for all $s \in S$. In that case U can *fire* under M, yielding the marking M', written $M \xrightarrow{U} M'$.

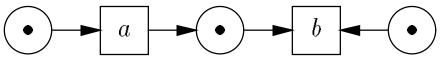
A chain $M_0 \xrightarrow{U_1} M_1 \xrightarrow{U_2} \cdots \xrightarrow{U_n} M_n$ starting from the initial marking M_0 is called a *firing sequence.* A marking M is *reachable* if there is such a sequence ending in $M = M_n$.

If a set U of transitions fires, for every transition t in U and every arc from a place s to t, a token moves along that arc from s to t. These tokens are consumed by the firing, but also new tokens are created, namely one for every outgoing arc of t. These end up in the places at the end of those arcs. If t occurs several times in U, all this happens several times (in parallel) as well.

The firing of U is only possible if there are sufficiently many tokens in the *preplaces* of U (the ones where the incoming arcs come from) and if by firing U, the capacities of the *postplaces* (where the outgoing arcs go to) are not exceeded. This is expressed by the two conditions in Definition 12.

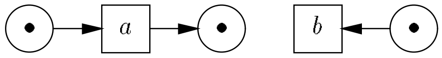
Definition 13 A net is said to be without capacities if the range of K is $\{\infty\}$ and without arcweights if the range of F is $\{0, 1\}$.

Equivalence relations on nets and the correspondence with set systems There are two different schools of thought in interpreting the causal behaviour of nets, which can be described as the *individual* and *collective token* philosophy.[2] The following example illustrates their difference.

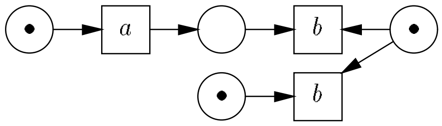
A: 

In this net, the transitions labelled a and b can fire once each. After a has fired, there are two tokens in the middle place. According to the individual token philosophy, it makes a difference which of these tokens is used in firing b. If the token that was there already is used (which must certainly be the case if b happens before the token from a arrives), the transitions a and b are causally independent. If the token that was produced by a is used, b is causally dependent on a. Thus, the net A above has two maximal computations, that can be characterised by the partial orders $\genfrac{}{}{0pt}{}{a}{b}$ and $a \twoheadrightarrow b$. According to the collective token philosophy on the other hand, all that is present in the middle place after the occurrence of a is the number 2. The preconditions for b to fire do not change, and consequently b is always causally independent of a.

The individual token approach has been formalised by the notion of a *process*, described in GOLTZ & REISIG [8]. A causality respecting bisimulation relation based on this approach was proposed by BEST, DEVILLERS, KIEHN & POMELLO [1] under the name *fully concurrent bisimulation*. It can be regarded as a form of history preserving bisimulation with these processes as histories. Below we contribute a form of history preserving bisimulation, and several related equivalences, based on the collective token philosophy. That both philosophies yield incomparable notions of equivalence follows from the following example.

B: 

In the collective token philosophy the precondition of b expressed by the place in the middle is redundant, and hence A must be equivalent to B. A and B are not fully concurrent bisimulation equivalent however, as B lacks the computation $a \twoheadrightarrow b$. On the other hand, A is fully concurrent bisimulation equivalent with C below.

C: 

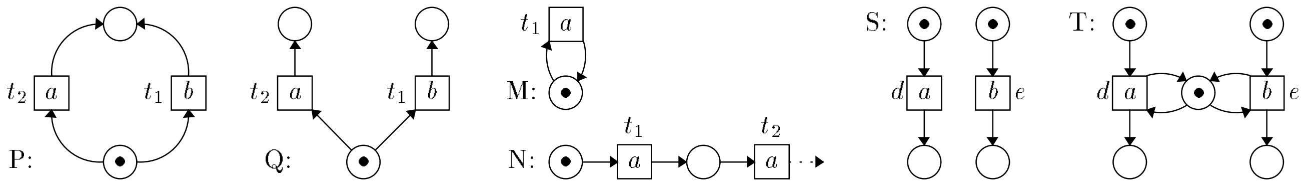
In fact, C is the occurrence net obtained from A by the unfolding of MESEGUER, MONTANARI & SASSONE [14], mentioned in the introduction. In the individual token philosophy, both nets have the computations $\genfrac{}{}{0pt}{}{a}{b}$ and $a \twoheadrightarrow b$. However, A does not have a run $a \twoheadrightarrow b$ in the collective token philosophy, and can therefore not be equivalent to C in any causality preserving way.

One often is interested in the behaviour of nets as far as it can be expressed in terms of transition firings. The places etc. are then seen as just a tool in specifying such behaviour. In this view, one of the most discriminating behavioural equivalences we can think of in the collective token framework is the following notion of *marking equivalence*:

Definition 14 Two nets N and N' are *marking equivalent* if $T_N = T_{N'}$ and there exists a bijection i between their reachable markings, such that the initial markings are related, $M \xrightarrow{U}_N M' \Leftrightarrow i(M) \xrightarrow{U}_{N'} i(M')$, and $l_N = l_{N'}$.

Note that marking equivalence preserves all causal information present in the net representation of a concurrent system. Whether two transitions are causally independent is a context-sensitive matter. It varies with the markings enabling them both. In such a marking two transitions are independent iff they can fire in one step. This kind of information is present in the step transition relation $\xrightarrow{U}$. The nets A and B are marking equivalent. However, for many purposes marking equivalence is too fine. It distinguishes for instance the nets P and Q below, as well as M and N.

[2]The individual token interpretation of ordinary nets has nothing to do with the concept of *Petri nets with individual tokens*; there the individuality is hardwired into the syntax of nets.



Definition 15 A *configuration* of a net is any finite multiset X of transitions with the property that the function $M_X : S \to \mathbb{Z}$ given by $M_X(s) = M_0(s) + \Sigma_{t \in T} X(t) \cdot (F(t,s) - F(s,t))$ is a marking (i.e. $0 \leq M_X(s) \leq K(s)$ for all $s \in S$). The firing relation between markings is inherited by the configurations of a net by $X \xrightarrow{U} X' \Leftrightarrow M_X \xrightarrow{U} M'_X$.

The *(labelled) configuration structure associated to net* N is the tuple $\mathcal{C}(N) = (T_N, C_N, \longrightarrow_N, l_N)$, in which C_N denotes the set of configurations of N.

Note that if $M_X \xrightarrow{U} M'$ then $X \oplus U$ is a configuration and $M' = M_{X \oplus U}$. It follows that the *reachable* configurations can equivalently be defined as those multisets of transitions X for which there is a firing sequence $M_0 \xrightarrow{U_1} M_1 \xrightarrow{U_2} \cdots \xrightarrow{U_n} M_n$ with $X = \bigoplus_{i=1}^n U_i$.

Note too that the configuration structure associated to a net is not a configuration structure in the sense of Definition 2. It is a structure in a class that enriches set systems in two ways: first by the use of multisets instead of sets, and second, by the addition of the transition relation. Still we define:

Definition 16 Two nets N and N' are *configuration equivalent* if $\mathcal{C}(N) = \mathcal{C}(N')$.

Reachable configuration equivalence is defined similarly and is strictly coarser than (reachable) marking equivalence: The nets P and Q are (reachable) configuration equivalent. However M and N below are not. The reason is that although in N all transitions have a different identity, even though they have the same label.

Next we will determine which class of nets can be described by means of set systems.

Definition 17 A *1-occurrence net* is a net in which every configuration is a set.

This implies that any transition can fire at most once. Were we interested only in the reachable configurations we could equivalently require that in every firing sequence $M_0 \xrightarrow{U_1} \cdots \xrightarrow{U_n} M_n$ the multisets $U_1, ..., U_n$ are sets and disjoint.

In general the firing relation $\longrightarrow_N$ of a net N is not determined by the set of configurations of N. The nets S and T have a very different behaviour: in S the actions a and b can be done in parallel, whereas in T there is mutual exclusion. Yet their configurations are the same: S corresponds to the configuration structure D of Example 1; T corresponds to a similar structure, but without the diagonal ab. Therefore it is not a good idea to equate a 1-occurrence net N with the configuration structure (T_N, C_N, l_N).

Definition 18 A net N is *pure* if there is no s in S_N and t in T_N with $F_N(s,t) > 0$ and $F_N(t,s) > 0$, i.e. if it is without *self-loops*.

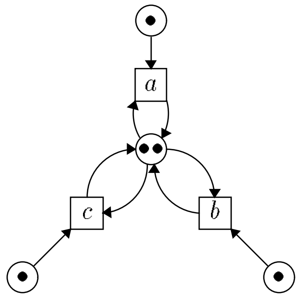
Proposition 6 In a pure net N we have $X \longrightarrow_N Y$ iff $\forall Z (X \subset Z \subset Y \Rightarrow Z \in C_N)$ for all X, Y in C_N.

It follows that for pure nets the transition relation is completely determined by the associated set of configurations. Thus for every pure 1-occurrence net N, $\mathcal{C}(N)$ is a set system.

As a consequence, all equivalence notions that are available for configuration structures are available for pure 1-occurrence nets as well. Two such nets are x-equivalent iff the associated configuration structures are. One also has these equivalences for comparing such nets with configuration or event structures.

Moreover, the equivalence notions of Section 2 generalise verbatim to configuration structures enriched with an explicit step transition relation, and hence apply to arbitrary 1-occurrence nets.

Now we can ask whether any 1-occurrence net, possibly with self-loops, is equivalent to a pure 1-occurrence net, or configuration structure. The net T is history preserving bisimulation equivalent to a pure 1-occurrence net. This does not hold in general:



Proposition 7 The 1-occurrence net above is not history preserving bisimulation equivalent to a configuration structure. However, every 1-occurrence net is ST-bisimulation equivalent to such a structure.

Below we show that the restriction to 1-occurrence nets is not very crucial; every net can be "unfolded" into a 1-occurrence net without changing its behaviour in any essential way. However, the unfolding cannot be configuration equivalent to the original, as the identity of transitions cannot be preserved.

Definition 19 Let $N = (S, T, F, K, M_0, l)$ be a Petri net. Its *1-unfolding* $N' = (S', T', F', K', M'_0, l')$ into a 1-occurrence net is given by (for $s \in S$, $t \in T$, $u \in T'$)

- $T' = T \times \mathbb{N}$ and $l'((t, n)) = l(t)$,
- $S' = S \cup (T' \times \{*\})$,
- $F'(s, (t, n)) = F(s, t)$ and $F'((t, n), s) = F(t, s)$,
- $F'(u, (u, *)) = F'((u, *), u') = F(u', (u, *)) = 0$ and $F'((u, *), u) = 1$ for $u, u' \in T'$ with $u \neq u'$,
- $K'(s) = K(s)$ and $K((u, *)) = \infty$,
- $M'_0(s) = M_0(s)$ and $M'_0((u, *)) = 1$.

Thus, every transition is replaced by countably many copies, each of which is connected with its environment (though the flow relation) in exactly the same way as the original. Furthermore, for every such copy u an extra place $(u, *)$ is created, containing one initial token, and having no incoming arcs and only one outgoing arc, going to u. This place guarantees that u can fire only once. In any reachable marking of the unfolded net, one can see exactly which transitions have fired, namely those transitions u for which the place $(u, *)$ is empty. Hence every such marking has only one configuration.

Proposition 8 Two nets are configuration equivalent iff their unfoldings are. Moreover, for each of our equivalences x, two 1-occurrence nets are x-equivalent iff their unfoldings are.

Proposition 8 shows we can define all our equivalences on general nets.

Definition 20 Two nets are said to be x-equivalent iff their unfoldings are.

Proposition 8 guarantees that this definition is consistent with the one we had already for 1-occurrence nets. Under this definition a net is configuration preserving bisimulation equivalent with its unfolding, although there is no *bijective* relation between the configurations. This is because a (trivial) element of choice is introduced by the construction.

Proposition 9 Every net is configuration preserving as well as fully concurrent bisimulation equivalent with its unfolding.

This tells us that the notion of unfolding preserves the behaviour of nets under both the collective and the individual token interpretation.

It is possible however to give a slightly different interpretation of nets, namely by excluding transitions from firing concurrently with themselves.[3] This amounts to simplifying Definition 12 by requiring U to be a set rather than a multiset. Under this interpretation our unfolding could introduce concurrency that was not present before. However, for this purpose Definition 19 can be adapted by removing the initial tokens from the places $((t, n), *)$ for $t \in T$ and $n > 0$ (but leaving the token in $((t, 0), *)$), and adding an arc from transition (t, n) to place $((t, n+1), *)$ for every $t \in T$ and $n \in \mathbb{N}$. If both Definition 12 and 19 are adapted as indicated above, Propositions 8 and 9 hold again.

Note that the construction in Definition 19 does not introduce self-loops. Thus unfoldings of pure nets remain pure.

Corollary 1 For every Petri net there exists an ST-bisimulation equivalent configuration structure. For every pure net there exists a configuration preserving bisimulation equivalent configuration structure. And for every pure 1-occurrence net N, $\mathcal{C}(N)$ *is* a configuration structure.

The configuration structure associated to a Petri net is always finitary and rooted. The following shows that Corollary 1 has an an inverse: every such configuration structure can be obtained as the image of a pure 1-occurrence net.

Theorem 3 For every finitary rooted configuration structure there exists a pure 1-occurrence net without capacities or arcweights with the same configurations.

Proof: As transitions of the net we take the events of the configuration structure. For every transition we add one place without incoming arcs, and with its only outgoing arc going to that transition. These places make sure that every transition fires only once. Let Φ be an axiomatisation of the configuration structure consisting of propositional sequents only. For every sequent $X \Rightarrow Y$ with X finite, we introduce a place in the net. This place has outgoing arcs to each of the transitions in X, and incoming arcs from each of the places in Y. Let n be the cardinality of X. As the configuration structure is rooted, $n \neq 0$. We finish the construction by putting $n - 1$ initial tokens in the created place.

[3] This distinction is independent of the individual–collective token dichotomy, thus yielding four computational interpretations of nets.

The place belonging to sequent $X \Rightarrow Y$ does not place any restrictions on the firing of the first $n-1$ transitions in X. However, the last one can only fire after an extra token arrives in the place. This can happen only if one of the transitions in Y fires first. The firing of more transitions in Y has no adverse effects, as each of the transitions in X can fire only once. Thus this place places the same restriction on the occurrence of events as the corresponding sequent. It follows that the constructed net has exactly the same reachable configurations as the original configuration structure. It even has the same unreachable ones. □

As a corollary we obtain that every pure net is configuration preserving bisimulation equivalent to a net without capacities or arcweights. Every pure 1-occurrence net is even configuration equivalent to a net without capacities or arcweights.

References

[1] E. Best, R. Devillers, A. Kiehn & L. Pomello (1991): *Concurrent bisimulations in Petri nets. Acta Informatica* 28, pp. 231–264.

[2] G. Boudol (1990): *Flow event structures and flow nets.* In I. Guessarian, editor: *Semantics of Systems of Concurrent Processes, Proceedings LITP Spring School on Theoretical Computer Science,* La Roche Posay, France, LNCS 469, Springer-Verlag, pp. 62–95.

[3] D.M. Gabbay (1981): *Semantic Investigations in Heyting's Intuitionistic Logic, Synthese Library* 148. D. Reidel.

[4] R.J. van Glabbeek (1995): *History preserving process graphs.* Report, Stanford University, Available at `ftp://boole.stanford.edu/pub/DVI/history.dvi.gz`.

[5] R.J. van Glabbeek & U. Goltz (1989): *Equivalence notions for concurrent systems and refinement of actions.* In A. Kreczmar & G. Mirkowska, editors: Proceedings 14^{th} Symposium on *Mathematical Foundations of Computer Science,* Porąbka-Kozubnik, Poland 1989, LNCS 379, Springer-Verlag, pp. 237–248.

[6] R.J. van Glabbeek & U. Goltz (1990): *Refinement of actions in causality based models.* In J.W. de Bakker, W.P. de Roever & G. Rozenberg, editors: *Proceedings REX Workshop on Stepwise Refinement of Distributed Systems: Models, Formalism, Correctness,* Mook, The Netherlands, May/June 1989, LNCS 430, Springer-Verlag, pp. 267–300.

[7] R.J. van Glabbeek & F.W. Vaandrager (1987): *Petri net models for algebraic theories of concurrency.* In J.W. de Bakker, A.J. Nijman & P.C. Treleaven, editors: *Proceedings PARLE conference,* Eindhoven, *Vol. II (Parallel Languages),* LNCS 259, Springer-Verlag, pp. 224–242.

[8] U. Goltz & W. Reisig (1983): *The non-sequential behaviour of Petri nets. Information and Computation* 57, pp. 125–147.

[9] J. Gunawardena (1992): *Causal automata. Theoretical Computer Science* 101, pp. 265–288.

[10] V. Gupta (1994): *Chu Spaces: A Model of Concurrency.* PhD thesis, Stanford University. Available at `ftp://boole.stanford.edu/pub/gupthes.ps.gz`.

[11] V. Gupta & V.R. Pratt (1993): *Gates accept concurrent behavior.* In *Proc. 34th Ann. IEEE Symp. on Foundations of Comp. Sci.*, pp. 62–71.

[12] R. Langerak (1992): *Transformations and Semantics for LOTOS.* PhD thesis, Department of Computer Science, University of Twente.

[13] K.G. Larsen & G. Winskel (1991): *Using information systems to solve recursive domain equations. Information and Computation* 91(2), pp. 232–258.

[14] J. Meseguer, U. Montanari & V. Sassone (1992): *On the semantics of Petri nets.* In W.R. Cleaveland, editor: *Proceedings CONCUR 92,* Stony Brook, NY, USA, LNCS 630, Springer-Verlag, pp. 286–301.

[15] R. Milner (1980): *A Calculus of Communicating Systems*, LNCS 92. Springer-Verlag.

[16] M. Nielsen, G.D. Plotkin & G. Winskel (1981): *Petri nets, event structures and domains, part I. Theoretical Computer Science* 13(1), pp. 85–108.

[17] G.M. Pinna & A. Poigné (1995): *On the nature of events: another perspective in concurrency. Theoretical Computer Science* 138(2), pp. 425–454.

[18] G.D. Plotkin & V.R. Pratt (1988): *Teams can see pomsets.* Manuscript available at `ftp://boole.stanford.edu/pub/DVI/pp2.dvi.gz`.

[19] V.R. Pratt (1991): *Modeling concurrency with geometry.* In *Proc. 18th Ann. ACM Symposium on Principles of Programming Languages*, pp. 311–322.

[20] V.R. Pratt (1994): *Chu spaces: complementarity and uncertainty in rational mechanics.* Tech. report, TEMPUS Summer School, Budapest. Available at `ftp://boole.stanford.edu/pub/DVI/bud.dvi.gz`.

[21] D.S. Scott (1974): *Completeness and axiomatizability in many-valued logic.* In L. Henkin et al., editors: *Proc. Tarski Symposium*, AMS, pp. 411–435.

[22] G. Winskel (1987): *Event structures.* In W. Brauer, W. Reisig & G. Rozenberg, editors: *Petri Nets: Applications and Relationships to Other Models of Concurrency, Advances in Petri Nets 1986, Part II; Proceedings of an Advanced Course,* Bad Honnef, September 1986, LNCS 255, Springer-Verlag, pp. 325–392.

[23] G. Winskel (1989): *An introduction to event structures.* In J.W. de Bakker, W.P. de Roever & G. Rozenberg, editors: *REX School/Workshop on Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency,* Noordwijkerhout, The Netherlands, May/June 1988, LNCS 354, Springer-Verlag, pp. 364–397.