ELSEVIER Theoretical Computer Science 153 (1996) 211–244

Theoretical Computer Science

Petri nets and bisimulation

Mogens Nielsen*, Glynn Winskel

BRICS[1], Department of Computer Science, University of Aarhus, Ny Munkegade, Bldg. 540, DK-8000 Aarhus, Denmark

Abstract

Several categorical relationships (adjunctions) between models for concurrency have been established, allowing the translation of concepts and properties from one model to another. A central example is a coreflection between Petri nets and asynchronous transition systems. The purpose of the present paper is to illustrate the use of such relationships by transferring to Petri nets a general concept of bisimulation.

0. Introduction

Category theory has been used to structure the seemingly confusing world of models for concurrency – see [27] for a survey. The general idea is to formalize that one model is more expressive than another in terms of an "embedding", most often taking the form of a coreflection, i.e. an adjunction in which the unit is an isomorphism. The models are equipped with behaviour preserving morphisms, to be thought of as kinds of simulations. Besides providing an abstract language for expressing relationships between seemingly very different models, category theory also allows the translation of constructions and properties between models via adjunctions. For instance, most process algebra constructs, like parallel and nondeterministic composition, may be understood in terms of universal constructions, like product and coproduct. The preservation properties of adjoints are helpful in showing, and explaining why, semantics is respected in moving from one model to another. A coreflection central to this paper is that embedding asynchronous transition systems, in the sense of Bednarczyk [1] and Shields [22], in Petri nets.

The purpose of this paper is to illustrate the translation of concepts between models, focussing here on the transference of the concept of bisimulation to Petri nets

* Corresponding author.

[1] Centre of the Danish National Research Foundation.

0304-3975/96/$09.50
SSDI 0304-3975(95)00122-0

from other models. The notion of bisimulation was defined categorically in [8] in a form directly applicable to a wide range of models equipped with a notion of path. This general definition takes the form of an existence of a span of open maps. In [8] it was shown that in the special case of standard labelled transition systems with sequential paths, the definition agrees with the strong bisimulation of Milner [12], and in the case of event structures with nonsequential paths in the form of pomsets, the definition yielded an interesting strengthening of the the history-preserving bisimulation introduced by Rabinovich and Trakhtenbrot [20]. Here we show how the coreflection from other models to nets combined with abstract properties of the general definition of bisimulation from [8], provides a notion of bisimulation on nets which automatically inherits a number of important properties.

The main message of this paper is that the categorical view of models for concurrency, like Petri nets, provides guidelines for definitions of concepts like behavioural equivalences, consistent across a range of models. We illustrate how a notion of bisimulation can be read off for nets, and that this comes automatically equipped with a number of essential properties. The categorical approach here contrasts with the more common alternative of searching for a sensible candidate for bisimulation on nets and, having found one then checking it possesses these essential properties.

A word on our choice of morphisms, which might otherwise seem rather arbitrary. Objects of our categories will represent processes. Morphisms will represent a relationship between one process and another. Following [27], the morphisms we focus on here arise in relating the behaviours of processes and their components in languages like CCS. In CSS, communication is based on the synchronisation of atomic actions. Because of this we can restrict attention to morphisms which respect the granularity of actions, in the sense that an action may only be sent to at most one action, and not to a computation consisting of several actions. As is shown in [27], the resulting definitions of morphisms are sufficient to express via morphisms the relationship between a constructed process and its components built up using the operations of CCS. Conversely, the choice of morphisms also produces universal constructions which form the basis of a process description language. This language is a little richer than that of CCS and CSP in the sense that their operations are straightforwardly definable within it.

1. Models – a coreflection

In this section we introduce the models of Petri nets and asynchronous transition systems, and present a coreflection between them. The purpose is mainly to set the scene for the main results in the next section, and hence the presentation here focusses on central definitions and constructions. For further details and all missing proofs we refer to [27].

1.1. Transition systems

Transition systems are a frequently used model of parallel processes. They consist of a set of states, with an initial state, together with transitions between states which are labelled to specify the kind of events they represent.

Definition. A *transition system* is a structure

$$(S, i, L, tran)$$

where

- S is a set of *states* with *initial state* i,
- L is a set of *labels*,
- $tran \subseteq S \times L \times S$ is the *transition relation*. As usual, a transition (s, a, s') is drawn as $s \xrightarrow{a} s'$.

It is convenient to introduce *idle* transitions, associated with any state. This has to do with our representation of partial functions. We view a partial function from a set L to a set L' as a (total) function $\lambda: L \cup \{*\} \to L' \cup \{*\}$ such that $f(*) = *$, where $*$ is a distinguished element standing for "undefined". This representation is reflected in our notation $\lambda: L \to_* L'$ for a partial function λ from L to L'. It assumes that $*$ does not appear in the sets L and L', and more generally we shall assume that the reserved element $*$ does not occur in any of the sets of the structures we consider. The expected composition of partial functions is obtained by composing their representations. We shall identify total functions on a set L with partial functions never yielding $*$ on L.

Definition. Let $T = (S, i, L, tran)$ be a transition system. An *idle transition* of T typically consists of $(s, *, s)$, where $s \in S$. Define

$$tran_* = tran \cup \{(s, *, s) \mid s \in S\}.$$

Idle transitions help give a simple definition of morphism between transition systems.

Definition. Let

$$T_0 = (S_0, i_0, L_0, tran_0) \quad \text{and} \quad T_1 = (S_1, i_1, L_1, tran_1)$$

be transition systems. A *morphism* $f: T_0 \to T_1$ is a pair $f = (\sigma, \lambda)$, where

- $\sigma: S_0 \to S_1$, a function between sets of states,
- $\lambda: L_0 \to_* L_1$, a partial function between sets of labels, are such that $\sigma(i_0) = i_1$ and

$$(s, a, s') \in tran_0 \Rightarrow (\sigma(s), \lambda(a), \sigma(s')) \in tran_{1*}.$$

The intention behind the definition of morphism is that the effect of a transition with label a in T_0 leads to inaction in T_1 precisely when $\lambda(a)$ is undefined. In our definition of morphism, idle transitions represent this inaction, so we avoid the fuss of considering whether or not $\lambda(a)$ is defined. With the introduction of idle transitions, morphisms on transition systems can be described as preserving transitions and the initial state. It is stressed that an idle transition $(s, *, s)$ represents inaction, and is to be distinguished from the action expressed by a transition (s, a, s') for a label a.

Transition systems with morphisms form a category **T** in which the composition of two morphisms $f = (\sigma, \lambda): T_0 \to T_1$ and $g = (\sigma', \lambda'): T_1 \to T_2$ is $g \circ f = (\sigma' \circ \sigma, \lambda' \circ \lambda)$: $T_0 \to T_2$ and the identity morphism for a transition system T has the form $(1_S, 1_L)$ where 1_S is the identity function on states and 1_L is the identity function on the labelling set of T. (Here composition on the left of a pair is that of total functions while that on the right is of partial functions).

1.2. Petri nets

A Petri net may be seen as a transition system with an explicit representation of (global) states as sets of (local) states (usually called conditions). The specific version adopted here was introduced in [10].

Definition. A *Petri net* consists of $(B, M_0, E, pre, post)$, where
B is a set of *conditions*, with *initial marking* M_0 a nonempty subset of B,
E is a set of *events*, and
$pre: E \to \mathcal{P}ow(B)$ is the *precondition* map such that $pre(e)$ is nonempty for all $e \in E$,
$post: E \to \mathcal{P}ow(B)$ is the *postcondition* map such that $post(e)$ is nonempty for all $e \in E$.

A Petri net comes with an initial marking consisting of a subset of conditions which are imagined to hold initially. Generally, a *marking*, a subset of conditions, formalizes a notion of global state by specifying those conditions which hold. Markings can change as events occurs, precisely how being expressed by the transitions

$$M \xrightarrow{e} M'$$

events e determine between markings M, M'. In defining this notion it is convenient to extend events by an "idling event".

Definition. Let $N = (B, M_0, E, pre, post)$ be a Petri net with events E.
Define $E_* = E \cup \{*\}$.
We extend the pre- and postcondition maps to $*$ by taking

$$pre(*) = \emptyset, \qquad post(*) = \emptyset.$$

Notation. Whenever it does not cause confusion we write ${}^\bullet e$ for the preconditions $pre(e)$ and $e^\bullet$ for the postconditions, $post(e)$, of $e \in E_*$. We write ${}^\bullet e^\bullet$ for ${}^\bullet e \cup e^\bullet$.

Definition. Let $N = (B, M_0, E, pre, post)$ be a net. For $M, M' \subseteq B$ and $e \in E_*$, define

$$M \xrightarrow{e} M' \quad \text{iff} \quad {}^\bullet e \subseteq M \;\&\; e^\bullet \subseteq M' \;\&\; M \setminus {}^\bullet e = M' \setminus e^\bullet .$$

Say $e_0, e_1 \in E_*$ are *independent* iff ${}^\bullet e_0^\bullet \cap {}^\bullet e_1^\bullet = \emptyset$.

A marking M of N is said to be *reachable* when there is a sequence of events, possibly empty, $e_1, e_2 \ldots e_n$ such that

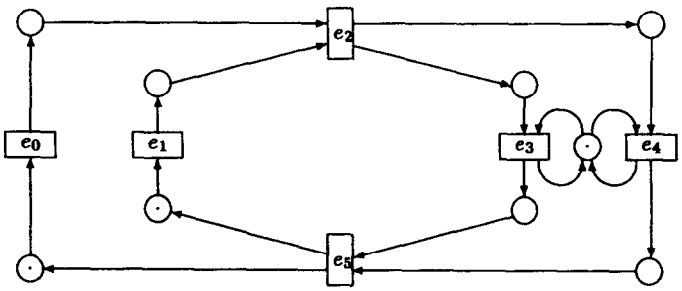
$$M_0 \xrightarrow{e_1} M_1 \xrightarrow{e_2} \cdots \xrightarrow{e_n} M_n = M$$

in N. There is *contact* at a marking M when for some event e, all its preconditions are marked at M and yet e cannot occur at M:

$${}^\bullet e \subseteq M \;\&\; e^\bullet \cap (M \setminus {}^\bullet e) \neq \emptyset.$$

A net is said to be *safe* when contact never occurs at any reachable marking.

Example. The following is an example of a standard graphical representation of a safe net with six events and nine conditions. Notice in particular that events e_0 and e_1 are independent, whereas e_3 and e_4 are not. One of the essential properties of nets is this possibility of specifying independence amongst events in terms of pre- and post-conditions.



As morphisms on nets we take:

Definition. Let $N = (B, M_0, E, pre, post)$ and $N' = (B', M'_0, E', pre', post')$ be nets. A *morphism* $(\beta, \eta): N \to N'$ consists of a relation $\beta \subseteq B \times B'$, such that its opposite relation $\beta^{op} \subseteq B' \times B$ is a partial function from B' to B, and a partial function $\eta: E \to_* E'$ such that

$$\beta M_0 = M'_0,$$

$$\beta {}^\bullet e = {}^\bullet \eta(e)$$

and

$$\beta e^\bullet = \eta(e)^\bullet .$$

Thus morphisms on nets preserve initial markings and events when defined. A morphism $(\beta, \eta): N \to N'$ expresses how occurrences of events and conditions in N induce occurrences in N'. Morphisms on nets preserve behaviour:

Proposition 1. *Let $N = (B, M_0, E, pre, post)$, $N' = (B', M'_0, E', pre, post')$ be nets. Suppose $(\beta, \eta): N \to N'$ is a morphism of net.*

- *If $M \xrightarrow{e} M'$ in N then $\beta M \xrightarrow{\eta(e)} \beta M'$ in N'.*
- *If ${}^\bullet e_1^\bullet \cap {}^\bullet e_2^\bullet = \emptyset$ in N then ${}^\bullet \eta(e_1)^\bullet \cap {}^\bullet \eta(e_2)^\bullet = \emptyset$ in N'.*

Proof. By definition,

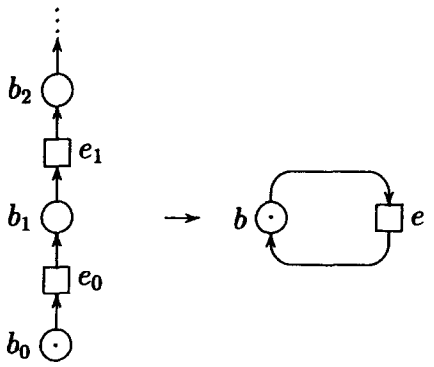
$${}^\bullet \eta(e) = \beta {}^\bullet e \quad \text{and} \quad \eta(e)^\bullet = \beta e^\bullet$$

for e an event of N. Observe too that because β^{op} is a partial function, β in addition preserves intersections and set differences. These observations mean that $\beta M \xrightarrow{\eta(e)} \beta M'$ in N' follows from the assumption that $M \xrightarrow{e} M'$ in N, and that independence is preserved. □

Proposition 2. *Nets and their morphisms form a category in which the composition of two morphisms $(\beta_0, \eta_0): N_0 \to N_1$ and $(\beta_1, \eta_1): N_1 \to N_2$ is $(\beta_1 \circ \beta_0, \eta_1 \circ \eta_0): N_0 \to N_2$ (composition in the left component being that of relations and in the right that of partial functions).*

Definition. Let **N** be the category of nets described above.

Remark. The rich structure of conditions on nets leaves room for variation, and another definition of morphism gives sensible results on the subclass of "safe" nets. A limitation with the above definition of morphism on nets is that it does not permit all "folding" morphisms of the kind illustrated in the example below.



The folding sends each event $e_0, e_1, \ldots$ to the common event e, and each condition $b_0, b_1, \ldots$ to the condition b. By restricting attention to safe nets we can relax the

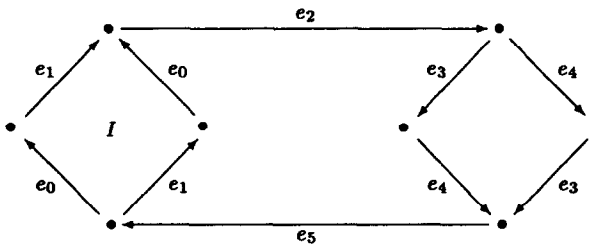
definition of morphisms on nets to include foldings, as in [25, 26], and still parallel the results of this paper – see [27] and the remark following Corollary 21.

1.3. Asynchronous transition systems

Following tradition, the behaviour of a net may be described via its *reachable case graph*, i.e. a transition system in which the states are the reachable markings and the transitions are triples

$$M \xrightarrow{e} M'$$

as defined above. The case graph of our previous net example will be as follows:



Notice how the event pairs (e_0, e_1) and (e_3, e_4) give rise to the same kind of diamonds in the underlying transition system. Hence, in order to get a representation of the important distinction between the pairs in terms of independence, we need to add some structure to the notion of case graph, here indicated by the I in the independent diamond. This is exactly the motivation behind *asynchronous transition systems*, as introduced independently by Bednarczyk [1] and Shields [22]. The idea on which they are based is simple enough: extend transition systems by, in addition, specifying which transitions are independent of each other. More accurately, transitions are to be thought of as occurrences of events which bear a relation of independence.

Definition. An *asynchronous transition system* consists of $(S, i, E, I, tran)$, where $(S, i, E, tran)$ is a transition system, $I \subseteq E^2$, the *independence relation* is an irreflexive, symmetric relation on the set E of *events* such that

(1) $e \in E \Rightarrow \exists s, s' \in S.\ (s, e, s') \in tran$,

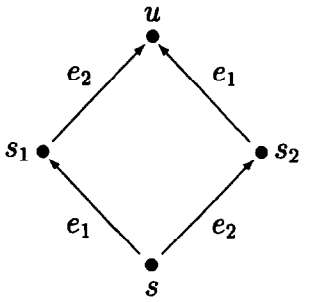
(2) $(s, e, s') \in tran \,\&\, (s, e, s'') \in tran \Rightarrow s' = s''$,

(3) $e_1 I e_2 \,\&\, (s, e_1, s_1) \in tran \,\&\, (s_1, e_2, u) \in tran$
$\Rightarrow \exists s_2.\ (s, e_2, s_2) \in tran \,\&\, (s_2, e_1, u) \in tran.$

Say an asynchronous transition system is *coherent* if it also satisfies

(4) $e_1 I e_2 \,\&\, (s, e_1, s_1) \in tran \,\&\, (s, e_2, s_2) \in tran$
$\Rightarrow \exists u.\ (s_1, e_2, u) \in tran \,\&\, (s_2, e_1, u) \in tran.$

Axiom (1) says every event appears as a transition, and axiom (2) that the occurrence of an event at a state leads to a unique state. Axioms (3) and (4) express properties of independence: if two independent events can occur one immediately after the other then they should be able to occur with their order interchanged (3); if two events can occur independently from a common state then they can occur together and in so doing reach a common state (4). Both situations lead to an "independence square" associated with the independence $e_1 I e_2$:



Morphisms between asynchronous transition systems are morphisms between their underlying transition systems which preserve the additional relations of independence.

Definition. Let $T = (S, i, E, I, tran)$ and $T' = (S', i', E', I', tran')$ be asynchronous transition systems. A *morphism* $T \to T'$ is a morphism of transition systems

$$(\sigma, \eta):(S, i, E, tran) \to (S', i', E', tran')$$

such that

$$e_1 I e_2 \,\&\, \eta(e_1), \eta(e_2) \text{ both defined} \Rightarrow \eta(e_1) I' \eta(e_2).$$

Morphisms of asynchronous transition systems compose as morphisms between their underlying transition systems, and are readily seen to form a category.

Definition. Let **A** be the category of asynchronous transition systems.

1.4. Asynchronous transition systems and nets

1.4.1. An adjunction

There is an adjunction between the categories **A** and **N**.[2] First, we note there is an obvious functor from nets to asynchronous transition systems, that associated with the case graph of a net.

[2]The adjunction between *coherent* asynchronous transition systems and nets is shown in detail in [27], to which the reader can refer for missing details (including missing proofs) in this section – the argument is virtually unaffected when working with the broader category of all asynchronous transition systems.

Definition. Let $N = (B, M_0, E, {}^\bullet(), ()^\bullet)$ be a net. Define $na(N) = (S, i, E, I, tran)$, where

$S = \mathcal{P}ow(B)$ with $i = M_0$,

$e_1 I e_2 \Leftrightarrow {}^\bullet e_1 \cap {}^\bullet e_2 = \emptyset$,

$(M, e, M') \in tran \Leftrightarrow M \xrightarrow{e} M'$ in N, for $M, M' \in \mathcal{P}ow(B)$.

Let $(\beta, \eta): N \to N'$ be a morphism of nets. Define

$$na(\beta, \eta) = (\sigma, \eta)$$

where $\sigma(M) = \beta M$, for any $M \in \mathcal{P}ow(B)$.

It may be shown [27] that *na* is indeed a functor, and that the construction $na(N)$, for a net N, yields a coherent asynchronous transition system.

As a preparation for the definition of a functor from asynchronous transition systems to nets we examine how a condition of a net N can be viewed as a subset of states and transitions of the asynchronous transition system $na(N)$. Intuitively, the *extent* $|b|$ of a condition b of a net is to consist of those markings and transitions at which b holds uninterruptedly. In fact, for simplicity, the extent $|b|$ of a condition b is taken to be a subset of $tran_*$, the transitions (M, e, M') and idle transitions $(M, *, M)$ of $na(N)$; the idle transitions $(M, *, M)$ play the role of markings M.

Definition. Let b be a condition of a net N. Let *tran* be the transition relation of $na(N)$. Define the *extent* of b to be

$$|b| = \{(M, e, M') \in tran_* \mid b \in M \;\&\; b \in M' \;\&\; b \notin {}^\bullet e^\bullet\}.$$

Not all subsets of transitions $tran_*$ of a net N are extents of conditions of N. For example, if $(M, e, M') \notin |b|$ and $(M', *, M') \in |b|$ for a transition $M \xrightarrow{e} M'$ in N this means the transition starts the holding of b. But then $b \in e^\bullet$ so any other transition $P \xrightarrow{e} P'$ must also start the holding of b. Of course, a condition cannot be started or ended by two independent events because, by definition, they can have no pre- or postcondition in common. These considerations motivate the following definition of condition of a general asynchronous transition system. The definition is a generalization of the notion of regions for transition systems introduced by Ehrenfeucht and Rozenberg [17].

Definition. Let $T = (S, i, E, I, tran)$ be an asynchronous transition system. Its *conditions* are nonempty subsets $b \subseteq tran_*$ such that

(1) $(s, e, s') \in b \Rightarrow (s, *, s) \in b \;\&\; (s', *, s') \in b$

(2) (i) $(s, e, s') \in {}^\bullet b \;\&\; (u, e, u') \in tran \Rightarrow (u, e, u') \in {}^\bullet b$

(ii) $(s, e, s') \in b^\bullet \;\&\; (u, e, u') \in tran \Rightarrow (u, e, u) \in b^\bullet$

where for $(s, e, s') \in tran$ we define

$$(s, e, s') \in {}^\bullet b \Leftrightarrow (s, e, s') \notin b \;\&\; (s', *, s') \in b,$$

$$(s, e, s') \in b^\bullet \Leftrightarrow (s, *, s) \in b \,\&\, (s, *, s') \notin b,$$

$${}^\bullet b^\bullet = {}^\bullet b \cup b^\bullet.$$

(3) $(s, e_1, s') \in {}^\bullet b^\bullet \,\&\, (u, e_2, u') \in {}^\bullet b^\bullet \Rightarrow \neg\, e_1 I e_2$.

Let B be the set of conditions of T. For $e \in E_*$, define

$$e^\bullet = \{b \in B \mid \exists s, s'.(s, e, s') \in {}^\bullet b\},$$

$${}^\bullet e = \{b \in B \mid \exists s, s'.(s, e, s') \in b^\bullet\},$$

$${}^\bullet e^\bullet = {}^\bullet e \cup e^\bullet.$$

(Note that ${}^\bullet *^\bullet = \emptyset$.)

Further, for $s \in S$, define $M(s) = \{b \in B \mid (s, *, s) \in b\}$.

As an illustrative exercise, we check that the extent of a condition of a net is indeed a condition of its asynchronous transition system.

Lemma 3. *Let N be a net with a condition b. Its extent $|b|$ is a condition of $na(N)$. Moreover,*

(I) $(M, e, M') \in {}^\bullet|b| \Leftrightarrow b \in e^\bullet$

(II) $(M, e, M') \in |b|^\bullet \Leftrightarrow b \in {}^\bullet e$,

whenever $M \xrightarrow{e} M'$ in N.

Proof. We prove (I) (the proof of (II) is similar):

$$\begin{aligned}(M, e, M') \in {}^\bullet|b| &\Leftrightarrow (M, e, M') \notin |b| \,\&\, (M', *, M') \in |b| \\ &\Leftrightarrow \neg(b \in M \,\&\, b \in M' \,\&\, b \notin {}^\bullet e^\bullet) \,\&\, b \in M' \\ &\Leftrightarrow (b \notin M \,\&\, b \in M') \text{ or } (b \in {}^\bullet e^\bullet \,\&\, b \in M') \\ &\Leftrightarrow b \in e^\bullet, \text{ as } M \xrightarrow{e} M'.\end{aligned}$$

Using (I) and (II), it is easy to check that $|b|$ is a condition of $na(N)$ – note that $|b|$ is nonempty because it contains, for instance, $(\{b\}, *, \{b\})$. □

Definition. Let $(\sigma, \eta): T \to T'$ be a morphism between asynchronous transition systems $T = (S, i, E, I, tran)$ and $T' = (S', i', E', I', tran')$. For $b \subseteq tran'_*$, define

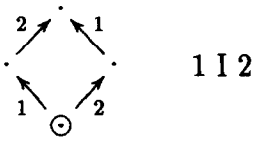
$$(\sigma, \eta)^{-1} b = \{(s, e, s') \in tran_* \mid (\sigma(s), \eta(e), \sigma(s')) \in b\}.$$

Definition. Let $T = (S, i, E, I, tran)$ be an asynchronous transition system. Define $an(T) = (B, M_0, E, pre, post)$ by taking B to be the set of conditions of T, $M_0 = M(i)$, with pre- and postcondition maps given by the corresponding operations in T, i.e. $pre(e) = {}^\bullet e$ and $post(e) = e^\bullet$ in T. Let $(\sigma, \eta): T \to T'$ be a morphism of asynchronous transition systems. Define $an(\sigma, \eta) = (\beta, \eta)$ where for conditions b of T and b' of T' we take

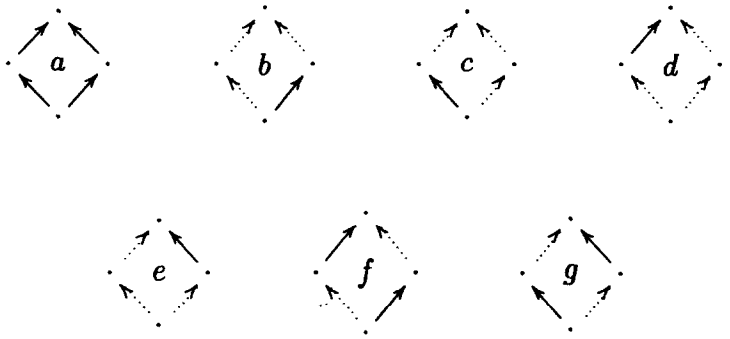
$$b \beta b' \text{ iff } b = (\sigma, \eta)^{-1} b'.$$

It may be shown that *an* as defined is indeed a functor, [27]. Let us illustrate here how a net is produced from an asynchronous transition system.

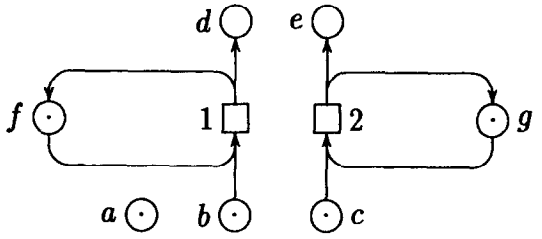
Example. Consider the following asynchronous transition system T with two independent events, 1 and 2:



It has these conditions, where those transitions in the condition are represented by solid arrows:



Consequently, the asynchronous transition system T yields this net $an(T)$:



Theorem 4. *The functors* $an: \mathbf{A} \to \mathbf{N}$ *and* $na: \mathbf{N} \to \mathbf{A}$ *form an adjunction with an left adjoint to na.*

1.4.2. A coreflection

Neither **A** nor **N** embeds fully and faithfully in the other category via the functors of the adjunction. This accompanies the facts that neither unit nor counit is an isomorphism (see [9, p. 88]); in passing from a net N to $an \circ na(N)$ extra conditions are most

often introduced; the net $an \circ na(N)$ is always safe even though N is not, as we will see. While passing from an asynchronous transition system T to $na \circ an(T)$ can, not only blow-up the number of states, but also collapse states which cannot be separated by conditions; in addition, the asynchronous transition system $na \circ an(T)$ is always coherent even though T is not.

A (full) coreflection between asynchronous transition systems and nets can be obtained at the cost of adding three axioms. Let $\mathbf{A}^0$ be the full subcategory of asynchronous transition systems $T = (S, i, E, I, tran)$ satisfying the following.

Axiom 1. Every state is reachable from the initial state, i.e. for every $s \in S$ there is a chain of events $e_1, \ldots, e_n$, possibly empty, for which $i \xrightarrow{e_1 \cdots e_n} s$, where i is the initial state.

Axiom 2. $M(u) = M(s) \Rightarrow u = s$, for all $s, u \in S$.

Axiom 3. ${}^\bullet e \subseteq M(s) \Rightarrow \exists s'.(s, e, s') \in tran$, for all $s \in S, e \in E$.

There is a close similarity to the regional axioms characterizing the case graphs of elementary net systems in terms of the regional axioms of Ehrenfeucht and Rozenberg, as presented in [17]. Axioms 2 and 3 enforce two separation properties. The contraposition of Axiom 2 says

$$u \neq s \;\Rightarrow\; M(u) \neq M(s),$$

i.e. that if two states are distinct then there is a condition of T holding at one and not the other.

Asynchronous transition systems satisfying Axiom 3 are necessarily coherent.

Proposition 5. *If an asynchronous transition system T satisfies Axiom 3 then T is coherent.*

Proof. Suppose $e_1 I e_2$ and $(s, e_1, s_1), (s, e_2, s_2)$ are transitions in T. Let b be a condition of T which e_2 exits, so, in particular, $(s, *, s) \in b$ and $(s, e_2, s_2) \notin b$. As $e_1 I e_2$, the condition b must contain (s, e_1, s_1) and so $(s_1, *, s_1)$. Thus ${}^\bullet e_2 \subseteq M(s_1)$. Axiom 3 now provides a transition (s_1, e_2, u). Property (3) in the definition of asynchronous transition systems together with property (1) (determinacy) now ensure coherence. □

Because the conditions of an asynchronous transition system support an operation of complementation (explained in [27]), Axioms 2 and 3 hold for any asynchronous transition system $na(N)$ got from a net N, but obviously Axiom 1 does not – we need further to make all states reachable. But here we note that the subcategory of asynchronous transition systems in which all states are reachable is coreflective in $\mathbf{A}$. The right adjoint to the inclusion functor, $\mathscr{R}$, defined below, restricts to reachable

states. Its composition with *na* yields the right adjoint of the coreflection between $\mathbf{A}^0$ and $\mathbf{N}$.

Definition. Let $\mathbf{A}^R$ be the full subcategory of $\mathbf{A}$ consisting of asynchronous transition systems $(S, i, E, I, tran)$ satisfyinfg Axiom 1, i.e. so that all states s are reachable.

Let $\mathscr{R}$ act on an asynchronous transition system $T = (S, i, E, I, tran)$ as follows:

$$\mathscr{R}(T) = (S', i', E', I', tran'),$$

where

S' consists of all reachable states of T,

$$E' = \{e \in E \mid \exists s, s' \in S'.\ (s, e, s') \in tran\},$$

$$I' = I \cap \{E' \times E'),$$

$$tran' = tran \cap (S' \times E' \times S').$$

For a morphism $(\sigma, \eta): T \to T'$ of asynchronous transition systems, define $\mathscr{R}(\sigma, \eta) = (\sigma', \eta')$ where σ' and η' are the restrictions of σ and η to the states, respectively events, of $\mathscr{R}(T)$.

We need the notion of *reachable extent* of a condition. This consists essentially of the reachable markings and transitions at which b holds uninterruptedly.

Definition. Let N be a net. Let $tran_*$ be the transitions and idle transitions of $\mathscr{R} \circ na(N)$. Define

$$|b|^R = |b| \cap tran_*.$$

And finally we can state the main result of this section, quoted from [27].

Theorem 6. *Defining* $na_0 = \mathscr{R} \circ na$, *the composition of functors, yields a functor* $na_0: \mathbf{N} \to \mathbf{A}^0$ *which is right adjoint to* $an_0: \mathbf{A}^0 \to \mathbf{N}$, *the restriction of an to* $\mathbf{A}^0$.

The unit at $T = (S, i, E, I, tran) \in \mathbf{A}^0$ *is an isomorphism*

$$(\sigma, 1_E): T \to na_0 \circ an_0(T),$$

where $\sigma(s) = M(s)$ *for* $s \in S$, *making the adjunction a coreflection.*

The counit at a net N is

$$(\beta, 1_E): an_0 \circ na_0 \to N,$$

where

$$c \beta b \text{ iff } \emptyset \neq c = |b|^R$$

between conditions c of $na_0(N)$ *and b of N.*

One consequence of the coreflection is that any net N can be converted to a safe net $an_0 \circ na_0(N)$ with the same behaviour, in the sense that there is an isomorphism

between the reachable asynchronous transition systems the two nets induce under na_0, for details see [27]. Another is that $\mathbf{A}^0$ has products and coproducts given by the same constructions as those of $\mathbf{A}$.

The coreflection $\mathbf{A}^0 \to \mathbf{N}$ cuts down to an equivalence of categories by restricting to the appropriate full subcategory of nets.

Definition. Let $\mathbf{N}^0$ be the full subcategory of *saturated nets*, i.e. nets such that

$$b \mapsto |b|^R$$

is a bijection between conditions of N and those of $na_0(N)$.

The nets in $\mathbf{N}^0$ are saturated with conditions in the sense that they have as many conditions as is allowed by their reachable behaviour and independence (regarded as an asynchronous transition system), see [27].

Theorem 7. *The functor an restricts to a functor* $an_0 : \mathbf{A}^0 \to \mathbf{N}^0$. *The functor* $\mathscr{R} \circ na$ *restricts to a functor* $na_0 : \mathbf{N}^0 \to \mathbf{A}^0$. *The functors* an_0, na_0 *form an equivalence of categories.*

1.5. Unfolding

There is a well-known operation of unfolding a transition system to a tree whose branches consist of sequences of occurrence of transitions that can be performed starting from the initial state. The operation in fact arises automatically as a right adjoint, part of a coreflection, between categories of synchronisation trees and transition systems. In more detail, define $\mathbf{S}$, the category of synchronisation trees, to be the full subcategory of transition systems whose objects satisfy:

- every state is reachable.
- the transitive closure of the transition relation is acyclic, and
- $s' \xrightarrow{a} s \,\&\, s'' \xrightarrow{b} s \Rightarrow a = b \,\&\, s' = s''$.

The inclusion functor $st : \mathbf{S} \hookrightarrow \mathbf{T}$ has as right adjoint the functor $ts : \mathbf{T} \to \mathbf{S}$ which on objects $T = (S, i, L, tran)$, a transition system, yields the synchronisation tree $ts(T) = (S', i', L, tran')$ where:

- The set S' consists of all finite, possibly empty, sequences of transitions

  $$(t_1, \ldots, t_j, t_{j+1}, \ldots, t_{n-1})$$

  such that $t_j = (s_{j-1}, a_j, s_j)$ and $t_{j+1} = (s_j, a_{j+1}, s_{j+1})$ whenever $1 < j < n$. The element $i' = ( )$, the empty sequence.
- The set $tran'$ consists of all triples (u, a, v) where $u, v \in S'$ and $u = (u_1, \ldots, u_k)$, $v = (u_1, \ldots, u_k, (s, a, s'))$, obtained by appending an a transition to u.

The transition system T unfolds to a synchronisation tree whose states and arcs represent occurrences of states and transitions.

What is the analogue of unfolding for models like Petri nets and asynchronous transition systems? This time the notion of occurrence should take account of the independence present in these more detailed models. Several answers have been proposed, Mazurkiewicz trace languages [10], occurrence nets [16] and event structures [16], though they are all closely related. Here we focus on one, event structures.

The events of an event structure are to be thought of as representing individual occurrences of actions of a system. The structural parts of an event structure are intended to capture the causal and nondeterministic aspects of such computations:

Definition. Define an *event structure* to be a structure $(E, \leqslant, Con)$ consisting of a set E, of *events* which are partially ordered by $\leqslant$, the *causal dependency relation*, and a *consistency relation* Con consisting of finite subsets of events, which satisfy

$\{e' \mid e' \leqslant e\}$ is finite,

$\{e\} \in Con$,

$Y \subseteq X \in Con \Rightarrow Y \in Con$,

$X \in Con \,\&\, e \leqslant e' \in X \Rightarrow X \cup \{e\} \in Con$,

for all events e, e' and their subsets X, Y.

We say two events $e, e' \in E$ are *concurrent*, and write $e \; co \; e'$, iff

$(e \not\leqslant e' \,\&\, e' \not\leqslant e \,\&\, \{e, e'\} \in Con)$.

The finiteness assumption restricts attention to discrete processes where an event occurrence depends only on finitely many previous occurrences. The axioms on the consistency relation express that all singletons of events are consistent, and that the relation is closed under subsets and downwards with respect to the causal dependency relation.

Say an event structure $E = (E, \leqslant, Con)$ is *coherent* if the consistency relation Con is determined by consistency on pairs of events, or alternatively if there is a, necessarily unique, binary conflict relation $\#$ on events such that

$$X \in Con \;\Leftrightarrow\; \forall e_1, e_2 \in X. \neg e_1 \# e_2.$$

We can describe coherent event structures by a triple $(E, \leqslant, \#)$ where, as before, E is a set of *events* partially ordered by a causal dependency relation $\leqslant$, and $\#$, the *conflict relation*, is a binary, symmetric, irreflexive relation on events, which satisfy

$\{e' \mid e' \leqslant e\}$ is finite,

$e \# e' \leqslant e'' \Rightarrow e \# e''$

for all $e, e', e'' \in E$. The property of $\#$, that two events causally dependent on conflicting events are themselves in conflict, follows from those of Con. We shall take the liberty of identifying $(E, \leqslant, \#)$, presenting a coherent event structure, with the

associated event structure $(E, \leqslant, Con)$; in other words, $(E, \leqslant, \#)$ should be understood as referring to the event structure $(E, \leqslant, Con)$ it determines.

To understand the "dynamics" of an event structure $(E, \leqslant, Con)$ we show how an event structure determines an asynchronous transition system $(S, i, E, I, tran)$. Guided by our interpretation we can formulate a notion of computation state of an event structure $(E, \leqslant, Con)$. Taking a computation state of a process to be represented by the set x of events which have occurred in the computation, we expect that

$$e' \in x \,\&\, e \leqslant e' \Rightarrow e \in x$$

– if an event has occurred then all events on which it causally depends have occurred too – and also that

$$\forall X \subseteq^{\text{fin}} x. X \in Con$$

– all finite subsets of events in the same computation are consistent. Let $C(E, \leqslant, \#)$ denote the subsets of events satisfying these two conditions, traditionally called the *configurations* of the event structure. We let S be the set of finite configurations and i the empty configuration.

Events manifest themselves as atomic jumps from one configuration to another. For configurations x, x' and event e, define

$$(x, e, x') \in tran \;\Leftrightarrow\; e \notin x \,\&\, x' = x \cup \{e\}.$$

We take two events to be independent in the asynchronous transition system iff they are concurrent in the event structure, i.e.

$$e_1 I e_2 \;\Leftrightarrow\; e_1 \,co\, e_2.$$

It is easy to see that this indeed defines an asynchronous transition system, $T = (S, i, E, I, tran)$ from the event structure $E = (E, \leqslant, Con)$. Furthermore, a coherent event structure gives rise to a coherent asynchronous transition system. The construction, which we call *ea*, identifying an event structure with an asynchronous transition system, extends to a functor with the following definition of morphisms for event structures:

Definition. Let $E = (E, \leqslant, Con)$ and $E' = (E', \leqslant', Con')$ be event structures. A *morphism* from E to E' consists of a total function $\eta: E \to E'$ on events which satisfies

$$\text{if } x \in C(E) \text{ then } \eta x \in C(E') \,\&\, \forall e_0, e_1 \in x. \eta(e_0) = \eta(e_1) \Rightarrow e_0 = e_1.$$

Write $\mathbf{E}$ for the category of event structures; composition is the usual composition of partial functions. Write $\mathbf{E}^0$ for the subcategory of *coherent* event structures.

The construction *ea* extends to a full and faithful functor:

Let $\eta: E \to E'$ be a morphism of event structures; it determines a morphism

$$ea(\eta) = (\sigma, \eta): ea(E) \to ea(E')$$

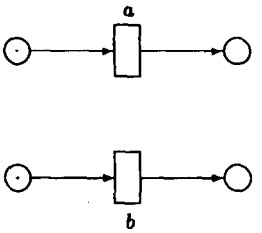
in which $\sigma(x) = \eta x$, simply the direct image of a configuration x under η. The "inclusion" functor $ea: \mathbf{E} \to \mathbf{A}$ has a right adjoint $ae: \mathbf{A} \to \mathbf{E}$ unfolding an asynchronous transition system to an event structure, forming a coreflection. We shall not go into the details of the construction of a right adjoint here, referring the reader to [27]; there it is shown how an asynchronous transition system determines a Mazurkiewicz trace language (easy) from which an event structure is obtained (harder).[3] The coreflection cuts down to one between the subcategory of coherent event structures and the subcategory of coherent asynchronous transition systems. In fact, the coreflection also cuts down to one, $ea_0: \mathbf{E}^0 \to \mathbf{A}^0$, $ae_0: \mathbf{A}^0 \to \mathbf{E}^0$. This is because it is easy to construct a net from a coherent event structure so that both induce the same asynchronous transition system (see [16, 27]); hence, images of $\mathbf{E}^0$ under ea lie in $\mathbf{A}^0$.

2. Labelled models and bisimulation

The coreflections of the previous sections enable us to place Petri nets within a broader picture of models for concurrency – [27] gives a fuller view. They allow us to apply to nets a general notion of bisimulation, obtained from a span of open maps, proposed in [8].

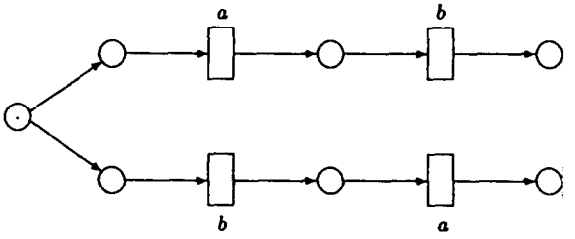
2.1. Labelled models and their relationship

Like most models for concurrency, nets [18] and asynchronous transition systems [14], or more precisely their labelled versions, have been used as models for process languages like CCS, [12]. As an illustration, following [18], the CCS expression $a.nil|b.nil$ is represented by the labelled net:



[3] In truth, this is only shown in detail for coherent structures in [27], though the slight generalisation, when coherence is not assumed, is also indicated there.

In contrast the (strongly bisimilar) expression $a.b.nil + b.a.nil$ is represented by



There is a general way of introducing labels to models in such a way that one may carry over adjunctions between unlabelled models to their labelled counterparts. Here we sketch the idea, applicable to the categories of nets, asynchronous transition systems and event structures. We assume a category $\mathbf{X}$ of structures each of which possesses a distinguished set of events and where morphisms have as a component a partial function between sets of events.

(i) Add to structures X an extra component of a (total) labelling function $l: E \to L$ from the structure's set of events E to a set of labels L; we obtain labelled structures as pairs (X, l).

(ii) We assume morphisms $f: X \to X'$ of unlabelled structures include a component η between sets of events. A morphism of labelled structures $(X, l) \to (X', l')$ is a pair (f, λ) where $f: X \to X'$ is a morphism on the underlying unlabelled structures and $\lambda: L \to_* L'$ is a partial function on the label sets such that $\lambda \circ l = l' \circ \eta$. Composition of morphisms is done coordinatewise.

Morphisms between labelled structures are of this generality in order to obtain operations of process calculi as universal constructions. However, for our purpose of studying bisimulation, it suffices to work with subcategories of structures having a common set of labels L, and restrict to morphisms as above, but with the extra condition that the component λ is the identity on L – this implies that the event component η is total. We call the resulting category $\mathbf{X}_L$; this subcategory is the fibre over L with respect to the obvious functor projecting labelled structures to their label sets. For emphasis:

- The objects of $\mathbf{X}_L$ consist of structures (X, l) where X is an object of $\mathbf{X}$, and $l: E \to L$ is a (total) labelling function from E the events of X to the labelling set L
- The morphisms of $\mathbf{X}_L$ from (X, l) to (X', l') correspond to morphisms $f: X \to X'$ of $\mathbf{X}$ of which the event component η preserves labels, i.e. $l' \circ \eta = l$.

Correspondingly, for a set of labels L, we denote the fibres over L in the labelled versions of our categories of nets, asynchronous transition systems and event structures by $\mathbf{N}_L$, $\mathbf{A}_L$, $\mathbf{A}_L^0$ and $\mathbf{E}_L^0$ respectively. Similarly the category of transition systems over label set L, with morphisms having the identity as label component, will be denoted $\mathbf{T}_L$, and its full subcategory of synchronisation trees $\mathbf{S}_L$. We remark that *synchronisation trees* can be identified with those event structures having empty *co*-relation.

It follows for general reasons [27] (and is easy to see) that the adjunction and coreflection between nets and asynchronous transition systems lifts to a coreflection between the labelled versions. The modified adjoints are essentially the adjoints presented in the previous sections, simply carrying the label parts across from one model to the other. Furthermore, this coreflection is part of a collection of coreflections as in the diagram below.

$$\begin{array}{ccccc} \mathbf{S}_L & \xrightarrow{st} & \mathbf{T}_L & & \\ se\downarrow & & & & \\ \mathbf{E}_L^0 & \xrightarrow[ea_0]{} & \mathbf{A}_L^0 & \xrightarrow[an_0]{} & \mathbf{N}_L \end{array}$$

These are accompanied by the coreflection $ea: \mathbf{E}_L \to \mathbf{A}_L$ between labelled event structures and asynchronous transition systems in general. When specifying a functor of one of the coreflections above, we adopt a convention; for example, the left adjoint from $\mathbf{S}_L$ to $\mathbf{T}_L$ is denoted *st* while its right adjoint is *ts*. The left adjoints, drawn above, embed one model in another. We have deliberately overloaded notation, and, for instance, used an_0 also for the labelled version of the embedding of $\mathbf{A}_L^0$ into $\mathbf{N}_L$. For details of the other coreflections we refer to [27]. The composition of right adjoints $ne = ae_0 \circ na_0$ yields the unfolding of nets into event structures, familiar from [16] (though the functor adds an extra marked isolated condition). Coreflections compose so the composition of left adjoints $en = an_0 \circ ea_0$ forms a coreflection with right adjoint *ne*. For readers familiar with net theory, we mention that for a net N, $en \circ ne(N)$ is simply the saturated version of the net unfolding of N as defined in [16]. Irritatingly, there are not coreflections from transition systems $\mathbf{T}_L$ to the categories of labelled nets $\mathbf{N}_L$ or asynchronous transition systems $\mathbf{A}_L$ or $\mathbf{A}_L^0$. This is simply because, unlike transition systems, both labelled nets and labelled asynchronous transition systems allow more than one transition with the same label between two states. This stops the natural bijection required for the "inclusion" of transition systems to be a left adjoint.

2.2. *Path-lifting morphisms*

In this section we briefly present some of the main ideas, definitions and results from [8], providing a general notion of bisimulation applicable to a wide range of models. For the missing proofs we refer to [8].

Informally, a computation path should represent a particular run or history of a process. For transition systems, a computation path is reasonably taken to be a sequence of transitions. Let us suppose the sequence is finite. For a labelling set L, define the category of branches $\mathbf{Bran}_L$ to be the full subcategory of transition systems, with labelling set L, with objects those finite synchronisation trees with one maximal branch; so the objects of $\mathbf{Bran}_L$ are essentially strings over alphabet L. A computation path in a transition system T, with labelling set L, can then be represented by a morphism

$$p: P \to T$$

in $\mathbf{T}_L$ from an object P of $\mathbf{Bran}_L$. How should we represent a computation path of a net or an event structure? To take into account the explicit concurrency exhibited by an event structure, it is reasonable to represent a computation path as a morphism from a partial order of labelled events, that is from a *pomset*. Note that Pratt's *pomsets*, with labels in L, can be identified with special kinds of labelled event structures in $\mathbf{E}_L$, those with consistency relation consisting of all finite subsets of events. Define the category of pomsets $\mathbf{Pom}_L$, with respect to a labelling set L, to be the full subcategory of $\mathbf{E}_L$ whose objects consist exclusively of finite pomsets. A computation path in an event structure E, with labelling set L, is a morphism

$$p : P \to E$$

in $\mathbf{E}_L$ from an object P of $\mathbf{Pom}_L$. What about computation paths in nets? The left adjoint $an_0 \circ ea_0$ of the coreflection $\mathbf{E}_L \to \mathbf{N}_L$ embeds labelled event structures, and so pomsets, in labelled nets. This enables us to identify pomsets P in $\mathbf{Pom}_L$ with their images $an_0 \circ ea_0(P)$ as labelled saturated nets in $\mathbf{N}_L$. Now, we can take a computation path in a net N, with labelling set L, to be a morphism

$$p : P \to N$$

in $\mathbf{N}_L$ from a pomset P, with labelling set $\mathbf{L}$ – where the pomset P is understood as the corresponding labelled saturated net in $\mathbf{N}_L$. In future, when discussing nets, we will deliberately confuse pomsets with their image in $\mathbf{N}_L$ under the embedding.

Generally, assume a category of models $\mathbf{M}$ (this can be any of the categories of labelled structures we are considering) and a choice of path category, a subcategory $\mathbf{P} \hookrightarrow \mathbf{M}$ consisting of path objects (these could be branches, or pomsets) together with morphisms expressing how they can be extended. Define a *computation path* in an object X of $\mathbf{M}$ to be a morphism

$$p : P \to X,$$

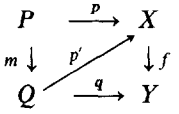
in $\mathbf{M}$, where P is an object in $\mathbf{P}$. A morphism $f : X \to Y$ in $\mathbf{M}$ takes such a path p in X to the path $f \circ p : P \to Y$ in Y. The morphism f expresses the sense in which Y simulates X; any computation path in X is matched by the computation path $f \circ p$ in Y.

We might demand a stronger condition of a morphism $f : X \to Y$ expressed succinctly in the following *path-lifting condition*:

Whenever, for $m : P \to Q$ a morphism in $\mathbf{P}$, in a "square"

$$\begin{array}{ccc} P & \xrightarrow{p} & X \\ {\scriptstyle m}\downarrow & & \downarrow{\scriptstyle f} \\ Q & \xrightarrow{q} & Y \end{array}$$

in **M** commutes, i.e. $q \circ m = f \circ p$, meaning the path $f \circ p$ in Y can be extended via m to a path q in Y, then there is a morphism p' such that in the diagram



the two "triangles" commute, i.e. $p' \circ m = p$ and $f \circ p' = q$, meaning the path p can be extended via m to a path p' in X which matches q. When the morphism f satisfies this condition we shall say it is **P**-*open*.

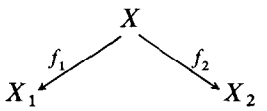
It is easily checked that **P**-open morphisms include all the identity morphisms (in fact, all isomorphisms) of **M** and are closed under composition there; in other words they form a subcategory of **M**.

For the well-known model of transition systems open morphisms are already familiar:

Proposition 8. *With respect to a labelling set L, the* $\mathbf{Bran}_L$*-open morphisms of* $\mathbf{T}_L$ *are the "zig-zag morphisms" of* [23], *the "p-morphism" of* [21], *the "abstraction homomorphisms" of* [3], *and the "pure morphisms" of* [2], *i.e. those label-preserving morphisms* $(\sigma, 1_L): T \to T'$ *on transition systems over labelling set L with the property that for all reachable states s of T*

$$\text{if } \sigma(s) \xrightarrow{a} s' \text{ in } T' \text{ then } s \xrightarrow{a} u \text{ in } T \text{ and } \sigma(u) = s', \text{ for some state } u \text{ of } T.$$

Let us return to the general set-up, assuming a path category **P** in a category of models **M**. Say two objects X_1, X_2 of **M** are **P**-*bisimlar* iff there is a *span* of **P**-open morphisms f_1, f_2:



For the interleaving models of transition systems and synchronisation trees with path category **P** taken to be branches, **P**-bisimulation coincides with Milner's strong bisimulation:

Theorem 9. *Two transition systems (and so synchronisation trees), over the same labelling set L, are* $\mathbf{Bran}_L$*-bisimilar iff they are strongly bisimilar in the sense of* [12].

Clearly, in general, the relation of **P**-bisimilarity between objects is reflexive (identities are **P**-open) and symmetric (in the nature of spans). It is also transitive provided **M** has pullbacks, and so an equivalence relation on objects, by virtue of the following fact.

Proposition 10. *Pullbacks of* **P**-*open morphisms are* **P**-*open.*

Transitivity of **P**-bisimilarity is clear for **M** with pullbacks; two spans of open morphisms combine to form a span by pulling back from their vertices, as we can do for all the models we consider:

Proposition 11. *The categories* $\mathbf{T}_L$, $\mathbf{S}_L$, $\mathbf{N}_L$, $\mathbf{A}_L^0$, $\mathbf{A}_L$, *and* $\mathbf{E}_L$ *have pullbacks.*

Proof. We show that $\mathbf{N}_L$ has pullbacks. There are coreflections from all categories $\mathbf{S}_L$, $\mathbf{E}_L$, $\mathbf{A}_L^0$ into $\mathbf{N}_L$. Using the fact that right adjoints preserve limits, and pullbacks in particular, we obtain pullbacks in any of $\mathbf{S}_L$,. $\mathbf{E}_L$, $\mathbf{A}_L^0$ as images under the right adjoints of the pullback in $\mathbf{N}_L$ of diagrams transported into $\mathbf{N}_L$ by the left adjoints. Because there are not coreflections from the categories $\mathbf{T}_L$ and $\mathbf{A}_L$ into nets, they require separate (though simple) treatments (or see [8]).

We construct pullbacks in $\mathbf{N}_L$ explicitly in the following way. Suppose $f_1 = (\sigma_1, \eta_1): N_1 \to N_0$ and $f_2 = (\sigma_2, \eta_2): N_2 \to N_0$ are morphisms in $\mathbf{N}_L$, where

$$N_i = (B_i, M_i, E_i, pre_i, post_i, l_i), \quad i = 0, 1, 2.$$

We want to construct a pullback $N = (B, M, E, pre, post, l)$, π_1, π_2:

$$\begin{array}{ccc} N & \xrightarrow{\pi_2} & N_2 \\ {\scriptstyle \pi_1}\downarrow & & \downarrow{\scriptstyle f_2} \\ N_1 & \xrightarrow[f_1]{} & N_0 \end{array}$$

The construction of the events of N, E, is based on pullbacks in the category of sets:

$$E = \{(e_1, e_2) \in E_1 \times E_2 \mid \eta_1(e_1) = \eta_2(e_2)\}.$$

The construction of the conditions of N, B is based on pushouts in the category of sets with partial functions. Let R denote the equivalence relation on $B_1 \cup B_2$ generated by R_0, where

$$b_1 R_0 b_2 \text{ iff there exists } b_0 \text{ in } B_0 \text{ such that } \beta_1(b_0) = b_1 \text{ and } \beta_2(b_0) = b_2.$$

We define

$$B = \text{the equivalence classes, } c, \text{ of } R, \text{ satisfying } \beta_1^{\mathrm{op}}(c) = \beta_2^{\mathrm{op}}(c).$$

And with these events and conditions of N we let

$$M = \{c \in B \mid c \subseteq M_1 \cup M_2\},$$

$$pre((e_1, e_2)) = \{c \in B \mid c \subseteq pre_1(e_1) \cup pre_2(e_2)\},$$

$$post((e_1, e_2) = \{c \in B \mid c \subseteq post_1(e_1) \cup post_2(e_2)\},$$

$$l((e_1, e_2)) = l_1(e_1)\ (= l_2(e_2)).$$

And finally we define the components $\pi_1 = (\bar{\beta}_1, \bar{\eta}_1)$ and $\pi_2 = (\bar{\beta}_2, \bar{\eta}_2)$ of the pullback as follows:

$\bar{\eta}_i((e_1, e_2)) = e_i$

$\bar{\beta}_i(b_i)$ = the R-equivalence class of b_i if this belongs to B, undefined otherwise.

We leave it to the reader to check that these constructions indeed define a pullback in $\mathbf{N}_L$ as required. All the required properties follow by simple calculations. □

Corollary 12. *For all the model categories mentioned in previous proposition, and for all path categories,* $\mathbf{P}_L$, *the relation of* $\mathbf{P}_L$*-bisimilarity is an equivalence.*

Finally, we present a few general facts from [8] about how open morphisms and bisimilarity are preserved and reflected by functors, especially when part of a coreflection. For notational simplicity we shall assume the left adjoints of the coreflections are inclusions. It follows that for the coreflections of Section 2.1 in which the two categories of models share the same choice of path category, open morphisms and bisimilarity are preserved in both directions of the adjunction.

Proposition 13. *Let* **M** *be a full subcategory of* **N**, *and* **P** *a subcategory of* **M**. *A morphism* f *of* **M** *is* **P***-open in* **M** *iff* f *is* **P***-open in* **N**.

Lemma 14. *Let* **M** *be a coreflective subcategory of* **N** *with* R *right adjoint to the inclusion function* $\mathbf{M} \hookrightarrow \mathbf{N}$ *and* **P** *a subcategory of* **M**. *Then*:

(i) *A morphism* f *of* **M** *is* **P***-open in* **M** *iff* f *is* **P***-open in* **N**.

(ii) *The components of the counit of the adjunction* $\varepsilon_X : R(X) \to X$ *are* **P***-open in* **M**.

(iii) *A morphism* f *is* **P***-open in* **N** *iff* $R(f)$ *is* **P***-open in* **M**.

Corollary 15. *Let* **M** *be a coreflective subcategory of* **N** *with* R *right adjoint to the inclusion function* $\mathbf{M} \hookrightarrow \mathbf{N}$ *and* **P** *a subcategory of* **M**. *Then*:

(i) M_1, M_2 *are* **P***-bisimilar in* **M** *iff* M_1, M_2 *are* **P***-bisimilar in* **N**.

(ii) N_1, N_2 *are* **P***-bisimilar in* **N** *iff* $R(N_1), R(N_2)$ *are* **P***-bisimilar in* **M**.

Proof. (i) Directly from (i) of Lemma 14.

(ii) "*only if*": By Lemma 14(iii), a span of open morphisms in **N** has, as image under R, a span of open morphisms in **M**. Thus **P**-bisimilarity of N_1, N_2 in **N** implies **P**-bisimilarity of $R(N_1), R(N_2)$ in **M**.

"*if*": Suppose $R(N_1)$, $R(N_2)$ in **M** are **P**-bisimilar in **M** via a span of open morphisms $f_1 : M \to R(N_1)$, $f_2 : M \to R(N_2)$ in **M**. By Lemma 14(i), f_1, f_2 form a span of open morphisms in **N**. The components of the counits of the coreflection $\varepsilon_1 : R(N_1) \to N_1$ and $\varepsilon_2 : R(N_2) \to N_2$ are open by Lemma 14(ii). Hence the compositions $\varepsilon_1 \circ f_1, \varepsilon_2 \circ f_2$ form a span of open morphisms in **N** showing the **P**-bisimilarity of N_1, N_2 in **N**. □

2.3. Pom_L-bisimulation for nets

We have already seen (Lemma 8, Theorem 9) that for the well-known model of transition systems, the general definition of **P**-open morphism and **P**-bisimilarity coincide with familiar notions; in particular, we recover the equivalence of strong bisimilarity central to Milner's work. Here we explore how the general definitions specialise to the models of event structures and nets, with nonsequential observations in the form of pomsets.

We start by characterising $\mathbf{Pom}_L$-open morphisms on labelled asynchronous transition systems. Following our convention, we shall identify pomsets with their image under the embedding $\mathbf{E}_L \to \mathbf{A}_L$.

Proposition 16. *The $\mathbf{Pom}_L$-open morphisms of $\mathbf{A}_L$ are precisely those which satisfy the "zig-zag" condition of Proposition 8 and which, in addition, reflect consecutive independence, i.e. morphisms satisfying:*

η is total and label preserving

whenever $(\sigma(s), e', u') \in tran_2$ then there exists $(s, e, u) \in tran_1$ such that $\eta(e) = e'$ and $\sigma(u) = u'$

whenever $(s, e, u), (u, e', v) \in tran_1$, with s reachable, and $\eta(e) I_2 \eta(e')$ in T_2, then $e I_1 e'$ in T_1.

Proof. The proof of this proposition is a straightforward modification of the proof of the corresponding result from [8]. We are going to refer to parts of this proof later, and so present a part in some detail.

Let $f = (\sigma, \eta): T \to T'$ be an open morphism in $\mathbf{A}_L$. The function η is total and label preserving from definition of morphisms in $\mathbf{A}_L$, and by considering linear pomsets, where causal dependency is a total order, it is clear as in Proposition 8, that f satisfies the "zig-zag" condition. The only nontrivial part is the reflection of consecutive independence.

Suppose

$$s \xrightarrow{e} u \quad \text{and} \quad u \xrightarrow{e'} v$$

with s reachable, are two consecutive transitions in T for which

$$\sigma(s) \xrightarrow{\eta(e)} \sigma(u) \quad \text{and} \quad \sigma(u) \xrightarrow{\eta(e')} \sigma(v)$$

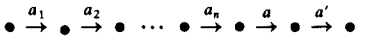
and assume $\eta(e)$ and $\eta(e')$ are independent in T'. Assume further $l(e) = l(\eta(e)) = a$ and $l(e') = l(\eta(e')) = a'$.

Because s is reachable there is a chain of transitions

$$i = s_0 \xrightarrow{e_1} s_1 \xrightarrow{e_2} \cdots \xrightarrow{e_n} s_n = s$$

in T from its initial state i. Assume $l(e_i) = a_i$. Let P be the linear pomset with $n + 2$ elements, ordered and labelled as indicated in the following associated labelled

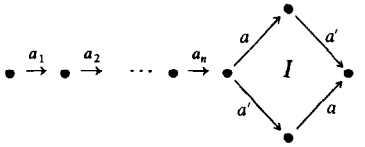
asynchronous transition system (only labels indicated for the transitions):



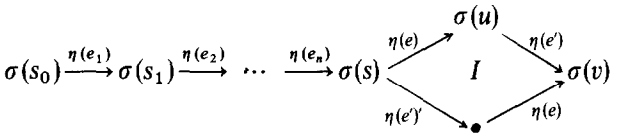
Let $p: P \to T$ be that morphism in $\mathbf{A}_L$ which maps this chain of transitions to

$$s_0 \xrightarrow{e_1} s_1 \xrightarrow{e_2} \cdots \xrightarrow{e_n} s \xrightarrow{e} u \xrightarrow{e'} v$$

in T. Let Q be the pomset differing from P only in that the a and a' labelled elements are unordered, i.e. the pomset associated with the following labelled asynchronous transition system:



Let $q: Q \to T'$ be that morphism in $\mathbf{A}_L$ mapping these transitions to



in T'. Letting $m: P \to Q$ be the obvious morphism of pomsets, we observe the commuting diagram:

$$\begin{array}{ccc} P & \xrightarrow{p} & T \\ {\scriptstyle m}\downarrow & & \downarrow{\scriptstyle f} \\ Q & \xrightarrow[q]{} & T' \end{array}$$

But f is open, so we obtain a morphism $p': Q \to T$ such that the two "triangles" commute in:

$$\begin{array}{ccc} P & \xrightarrow{p} & T \\ {\scriptstyle m}\downarrow & {\scriptstyle p'}\nearrow & \downarrow{\scriptstyle f} \\ Q & \xrightarrow[q]{} & T' \end{array}$$

Because p' preserves independence, we see that e and e' are independent in T. So because f is open it satisfies the "zig-zag" condition and reflects consecutive independence.

For the proof in the other direction we refer to [8]. □

And now to the question of bisimulations. In [8] it was shown that in the case of event structures taking the path category $\mathbf{P}$ to be pomsets one gets a reasonable strengthening of a previously studied equivalence, that of *history-preserving bisimulation*. Its definition depends on the simple but important remark, that a configuration

of an event structure can be regarded as a pomset, with causal dependency relation and labelling got by restricting that of the event structure.

Definition (Rabinovich–Trakhtenbrot [20], van Glabeek–Goltz [6]). A *history-preserving bisimulation* between two event structures E_1, E_2 consists of a set H of triples (x_1, f, x_2), where x_1 is a configuration of E_1, x_2 a configuration of E_2 and f is a isomorphism between them (regarded as pomsets), such that $(\emptyset, \emptyset, \emptyset) \in H$ and, whenever $(x_1, f, x_2) \in H$

(i) if $x_1 \stackrel{a}{\rightarrow} x_1'$ in E_1 then $x_2 \stackrel{a}{\rightarrow} x_2'$ in E_2 and $(x_1', f', x_2) \in H$ with $f \subseteq f'$, for some x_2' and f'.

(ii) if $x_2 \stackrel{a}{\rightarrow} x_2'$ in E_2 then $x_1 \stackrel{a}{\rightarrow} x_1'$ in E_1 and $(x_1', f', x_2') \in H$ with $f \subseteq f'$, for some x_1' and f'

We say a history-preserving bisimulation H is *strong* when it further satisfies

(I) $(x, f, y) \in H \,\&\, x' \subseteq x$, for a configuration x' of E_1 implies $(x', f', y') \in H$, for some $f' \subseteq f$ and $y' \subseteq y$.

(II) $(x, f, y) \in H \,\&\, y' \subseteq y$, for a configuration y' of E_2, implies $(x', f', y') \in H$, for some $f' \subseteq f$ and $x' \subseteq x$.

In [8] it is shown that $\mathbf{Pom}_L$-bisimilarity of event structures in $\mathbf{E}_L$ coincides with their being strong history-preserving bisimilar. However, this in itself does not show that $\mathbf{Pom}_L$-bisimilarity of event structures in the smaller category $\mathbf{E}_L^0$ of coherent event structures also coincides with strong history-preserving bisimilarity. There might conceivably be a span of open morphisms, $f_1 : E \rightarrow E_1$, $f_2 : E \rightarrow E_2$, from a noncoherent event structure E relating two coherent event structures E_1, E_2 which could never be replaced by a span of open morphisms from a coherent event structure. In fact, such is not the case, because for any event structure E in $\mathbf{E}_L$ there is an open morphism $f : E' \rightarrow E$ from a *coherent* event structure E' (Lemma 18). Hence a span of open morphisms $f_1 : E \rightarrow E_1$, $f_2 : E \rightarrow E_2$ in $\mathbf{E}_L$, with E_1, E_2 coherent, can always be converted to a span of open morphisms $f_1 \circ f : E' \rightarrow E_1$, $f_2 \circ f : E' \rightarrow E_2$ in $\mathbf{E}_L^0$. Consequently, $\mathbf{Pom}_L$-bisimilarity in the subcategory of coherent event structures $\mathbf{E}_L^0$ coincides with strong history-preserving bisimilarity. This result will also have implications for $\mathbf{Pom}_L$-bisimilarity between Petri nets, because of the coreflection from coherent event structures to nets.

Although we have not insisted on it, a reasonable requirement on event structures (and the other objects we consider here) is that they be countable. One might view with suspicion any result which depended crucially on allowing event structures to be uncountable. For this reason, some care has been taken to give countable constructions, at the cost of a little extra argumentation.

In preparation for the key lemma, Lemma 18, we first show how any consistency relation on events can be "simulated" by a conflict relation, ignoring for the moment causal dependency and labelling. A conflict relation consists of $(E, \#)$, where $\#$ is a binary irreflexive relation on E. In accord with the terminology for event structures,

we say a set $X \subseteq E$ is *consistent* iff

$$\forall e_1, e_2 \in X. \neg e_1 \# e_2.$$

A consistency relation consists of (E, Con), where Con is a family of finite subsets of E satisfying the following property familiar from event structures:

$$\{e\} \in Con,$$

$$Y \subseteq X \in Con \Rightarrow Y \in Con$$

for all elements e, e' and subsets X, Y of E. Of course, a conflict relation $(E, \#)$ determines a consistency relation (E, Con) in which Con consists of the finite consistent subsets of $(E, \#)$.[4]

Lemma 17. *Let Con be a consistency relation on a set A. There is $\#$, a conflict relation on a set B and a function $f: B \to A$ such that:*

(i) *If X is a finite consistent subset of $(B, \#)$, then $fX \in Con$, and*

$$\forall b_1, b_2 \in X. f(b_1) = f(b_2) \Rightarrow b_1 = b_2.$$

(ii) *If X is a finite consistent subset of $(B, \#)$ and*

$$fX \subseteq Y, \quad \text{for } Y \in Con,$$

then there is a finite consistent subset Z of $(B, \#)$ such that

$$X \subseteq Z \quad \text{and} \quad fZ = Y.$$

Moreover, if A is countable/finite then B can also be chosen to be countable/finite, respectively.

Proof. Assuming a consistency relation (A, Con) we first define M to consist of the minimal inconsistent subsets of A, i.e.

$$M = \{X \subseteq A \mid X \notin Con \,\&\, \forall Y \subsetneq X. Y \in Con\}.$$

Note sets in M are always finite. For $a \in A$, define

$$M(a) = \{X \in M \mid a \in X\}.$$

Now, define the set B to consist of elements of A with a twist. Formally, define the elements of B to be pairs

$$(a, t),$$

where $a \in A$ and t, a *twist*, is a tuple $\langle t_X \rangle_{X \in M(a)}$ of integers t_X, indexed by $X \in M(a)$, such that $0 < t_X < |X|$ – here $|X|$ denotes the size of X. (In particular, for $a \in A$, if

[4] Conflict and consistency relations play a role in models of constructive logic, where they correspond to Girard's coherent and qualitative domains [5].

$M(a) = \emptyset$, then a's only twist is the empty tuple, whereas if $M(a)$ consists solely of (unordered) pairs then all entries of its twists will be 1.) Define a conflict relation on B, by setting

$$(a, t) \# (a', t') \text{ iff } a = a' \,\&\, t \neq t' \text{ or}$$

$$a \neq a' \,\&\, \exists X \in M(a) \cap M(a').t_X = t'_X.$$

Define $f: B \to A$ to be the projection $(a, t) \mapsto a$. We should show (i) and (ii) above.

We first observe the following "counting property" of consistent sets of $(B, \#)$:

If X is a consistent set of $(B, \#)$ and $Y \in M$, then

$$|\{t_Y \mid \exists a \in Y.(a, t) \in X\}| = |Y \cap (fX)|.$$

To justify the counting property let

$$S = \{(a, t) \in X \mid a \in Y\}.$$

Then S, being a subset of X, is consistent in $(B, \#)$. The first clause in the definition of the conflict relation $\#$ on B ensures that $|S| = |fS|$. Clearly, $fS = Y \cap (fX)$, so

$$|S| = |Y \cap (fX)|.$$

The second clause in the definition of $\#$, ensures that

$$|S| = |\{t_Y \mid \exists a \in Y.(a, t) \in X\}|.$$

This establishes the counting property.

We now prove (i) and (ii).

(i) If (i) were to fail, there would be a consistent set X of $(B, \#)$ and $Y \in M$ such that $Y \subseteq fX$. But then by the counting property

$$|\{t_Y \mid \exists a \in Y.(a, t) \in X\}| = |Y|.$$

However this is impossible as each t_Y is bounded within the interval $\{k \mid 0 < k < |Y|\}$, of size $|Y| - 1$.

(ii) It suffices to show the following claim.

Claim. *Suppose X is a consistent set of $(B, \#)$ and $(fX) \cup \{a'\}$ is a consistent set of (A, Con). Then there is a twist u such that $(a', u) \in B$ and $X \cup \{(a', u)\}$ is a consistent set of $(B, \#)$.*

[*The notation $x = y \cup z$ means $x = y \cup z \,\&\, y \cap z = \emptyset$.*]

To construct a suitable twist u we need to find an assignment u_Y, for each $Y \in M(a')$, such that

$$u_Y \notin \{t_Y \mid \exists a \in Y.(a, t) \in X\}.$$

This is impossible only if

$$|\{t_Y \mid \exists a \in Y.(a, t) \in X\}| = |Y| - 1.$$

By the counting property, if this were so, then

$$|Y \cap (fX)| = |Y| - 1.$$

But then

$$Y \subseteq (fX) \dot{\cup} \{a'\},$$

contradicting the consistency of $(fX) \dot{\cup} \{a'\}$. Thus we can find a twist $u = \langle u_Y \rangle_{Y \in M(a')}$ such that $X \cup \{(a', u)\}$ is consistent.

The construction of $(B, \#)$ from (A, Con) can yield an uncountable set B even though A is countable. Suppose, for instance, that there is $a \in A$ for which $M(a)$ contains infinitely may sets of size greater than 2. Then B will include uncountably many-elements of the form (a, t). However, there will be a countable $(B, \#)$ fulfilling the conditions of the lemma when A is countable. The argument involves a little model theory. We can express the conditions on $f: B \to A$ as a countable theory in a predicate calculus. Our construction shows the theory to be consistent. It thus has a countable model, from which we can extract the required countable $(B, \#)$ and f.

In more detail, we take a predicate calculus with equality, over two sorts α and β, a single unary operation F from β to α, binary relation $\#$ on β, and for each $n \geqslant 0$ predicates Con_n^α and Con_n^β on α and β, respectively. If A is finite the construction above clearly yields a finite B, so we can restrict attention to countably infinite A enumerated as

$$a_0, a_1, \ldots, a_m, \ldots$$

We extend our language by constants $a_0, a_1, \ldots, a_m, \ldots$ of sort α. The theory T is to consist of:

- those atomic assertions and their negations which hold of A, i.e. those assertions $a_i = a_j$, $\neg(a_i = a_j)$, $Con_n^\alpha(a_{i_1}, \ldots, a_{i_n})$, $\neg\, Con_n^\alpha(a_{i_1}, \ldots, a_{i_n})$, which are true interpreted as assertions of consistency.
- the properties required of $\#$, Con_n^α, Con_n^β, e.g. assertions such as

$$\forall x_1, \ldots, x_n : \beta . Con_n^\beta(x_1, \ldots, x_n) \leftrightarrow \bigwedge_{i,j} \neg\, x_i \# x_j$$

  – here $\bigwedge_{i,j} \neg\, x_i \# x_j$ abbreviates a finite conjunction

  $$\cdots \wedge \neg\, x_i \# x_j \wedge \cdots \text{where } i, j \leqslant n,$$

  saying consistency in β is equivalent to conflict freeness, and other saying that consistency predicates are invariant under permutation, that consistency is closed under inclusion and contains all singletons.
- the conditions required on F, of the form

$$\forall x_1, \ldots, x_n : \beta . Con_n^\beta(x_1, \ldots, x_n) \rightarrow$$

$$\left(Con_n^{\alpha}(F(x_1), \ldots, F(x_n)) \,\&\, \left(\bigwedge_{i,j} F(x_i) = F(x_j) \to x_i = x_j\right)\right)$$

$$\forall x_1, \ldots, x_n : \beta,\ y : \alpha.\, Con_n^{\beta}(x_1, \ldots, x_n) \wedge Con_{n+1}^{\alpha}(F(x_1), \ldots, F(x_n), y)$$

$$\to (\exists x : \alpha.\, F(x) = y \wedge Con_{n+1}^{\beta}(x_1, \ldots, x_n, x)).$$

The theory T is countable, and is satisfied by our construction, so consistent. Every countable consistent first-order theory has a countable model (see e.g. [11, p. 65, Proposition 2.12]). In particular, the theory T has a countable model in which F is interpreted as a function f' from a countable set B' to a countable set A' which includes A. Restricting f' to the inverse image $B = f'^{-1}A$ we obtain a function $f : B \to A$ fulfilling the conditions required above, but now with respect to a countable $(B, \#)$ and (A, Con). □

Lemma 18. *Let $A = (A, \leqslant, Con, l)$ be a labelled event structure. Then, there is a labelled coherent event structure $E = (E, \leqslant', \#', l')$ and an open morphism $g : E \to A$.*

Moreover, if A is countable/finite then so can E be taken to be countable/finite, respectively.

Proof. There is a set B with binary conflict $\#$ and a function $f : B \to A$ satisfying the conditions of Lemma 17 with respect to the consistency relation Con on A. We first construct a labelled coherent asynchronous transition system T. Its states are finite consistent subsets x of $(B, \#)$ for which the direct image fx is a configuration of A. Its set of events is B with independence relation I, where

$$b_1 I b_2 \text{ iff } \neg b_1 \# b_2 \,\&\, f(b_1) co f(b_2).$$

Its labelling function is $l \circ f$. Its transitions are all (x, b, x') where $x' = x \cup \{b\}$ for x, x' and $b \in B$.

It can be verified that T is a labelled coherent asynchronous transition system. For example, to show property (4), assume x is a state of T with transitions (x, b_1, x_1) and (x, b_2, x_2) where $b_1 I b_2$. Then $x \cup \{b_1, b_2\}$ is a consistent set of $(B, \#)$. Hence because f preserves consistency, $fx \cup \{f(b_1), f(b_2)\} \in Con$. The two sets $fx_1 = fx \cup \{f(b_1)\}$ and $fx_2 = fx \cup \{f(b_2)\}$ are configurations of A and so $\leqslant$-downwards closed. Hence their union $fx \cup \{f(b_1), f(b_2)\}$ is consistent and $\leqslant$-downwards closed, and so a configuration of A. This ensures that $u = x \cup \{b_1, b_2\}$ is a state of T with transitions (x_1, b_2, u), (x_2, b_1, u), as required by (4).

There is an open morphism $(\sigma, f) : T \to ea(A)$ in $\mathbf{A}_L$, where $\sigma(x) = fx$. Here we have recourse to Lemma 16 characterising open morphisms in $\mathbf{A}_L$ and make essential use of the properties of f, expressed in Lemma 17. The right adjoint ae of the coreflection between $\mathbf{E}_L$ and $\mathbf{A}_L$ preserves open morphisms, by Lemma 14(iii), and there is an isomorphism $h : ae \circ ea(A) \cong A$, by the coreflection. Hence $g = h \circ ae(\sigma, f) : ae(T) \to A$, being the composition of an open morphism with an isomorphism, is itself an open

morphism in $\mathbf{E}_L$. Because T is *coherent* it unfolds under ae to an event structure of the form $ae(T) = (E, \leqslant', \#', l')$.

Because the constructions used in this proof preserve countability and finiteness, we see from Lemma 17, that in the proof B, and so E, may be made countable or finite according to whether A is countable or finite.

(We remark that an alternative proof is obtained by recognising that the states of T form the finite elements of a coherent stable family, and so of a coherent prime algebraic domain D. The event structure E is obtained, to within isomorphism, from the complete primes of D – see [25].) □

At long last we can show that restricting the category of event structures to those which are coherent does not effect the relation of bisimilarity.

Corollary 19. *Let E_1, E_2 be coherent event structures with labelling sets L. The following are equivalent:*

(i) *E_1, E_2 are $\mathbf{Pom}_L$-bisimilar in $\mathbf{E}_L^0$.*

(ii) *E_1, E_2 are $\mathbf{Pom}_L$-bisimilar in $\mathbf{E}_L$.*

(iii) *E_1, E_2 are strong history-preserving bisimilar.*

Proof. The equivalence between (i) and (ii) follows by Lemma 18. A span of open morphisms $f_1 : E \to E_1$, $f_2 : E \to E_2$ in $\mathbf{E}_L$, with E_1, E_2 coherent, can be converted to a span of open morphisms $f_1 \circ f : E' \to E_1$, $f_2 \circ f : E' \to E_2$ in $\mathbf{E}_L^0$, where $f : E' \to E$ is the open morphism provided by Lemma 18. The equivalence between (ii) and (iii) is shown in [8]. □

Via the coreflection between event structures and Petri nets, we can draw characterisations of $\mathbf{Pom}_L$-bisimilarity on nets.

Theorem 20. *Let N_1, N_2 be nets with labelling sets L. The following are equivalent:*

(i) *The nets N_1, N_2 are $\mathbf{Pom}_L$-bisimilar in $\mathbf{N}_L$.*

(ii) *The reachable case graphs $na_0(N_1)$, $na_0(N_2)$ are $\mathbf{Pom}_L$-bisimilar in $\mathbf{A}_L^0$.*

(iii) *The case graphs $na(N_1)$, $na(N_2)$ are $\mathbf{Pom}_L$-bisimilar in $\mathbf{A}_L$.*

(iv) *The unfoldings to event structures $ne(N_1)$, $ne(N_2)$ are strong history-preserving bisimilar.*

Proof. The equivalence between (i) and (ii) follows by Corollary 15 applied to the coreflection $\mathbf{A}_L^0 \to \mathbf{N}_L$. Because of the coreflection $\mathbf{E}_L^0 \to \mathbf{A}_L^0$, Corollary 15 yields the equivalence of (ii) with:

(iv)′ *The unfoldings to event structures $ne(N_1)$, $ne(N_2)$ are $\mathbf{Pom}_L$-bisimilar in $\mathbf{E}_L^0$.*

As we have seen (Corollary 19), we have that (iv)′ is equivalent to (iv). Finally, (ii) and (iii) are equivalent by Proposition 13 because $\mathbf{A}_L^0$ is a full subcategory of $\mathbf{A}_L$. □

So, for general reasons, the notion of bisimilarity for nets agrees with the notion of bisimilarity for the associated case graphs and unfoldings (where it amounts to strong history-preserving bisimilarity). Results expressing agreements of this kinds would probably be required of any notion of bisimilarity, and, without the help of some categorical machinery, would seem to require separate proofs. Of course, now we have characterised $\mathbf{Pom}_L$-bisimilarity on nets as strong history-preserving bisimilarity of their unfoldings to event structures, we may produce a characterisation in terms of nets and their "processes" along the lines of [24].

Many attempts have been made to define bisimilarity for noninterleaving models like Petri nets. The idea of parametrizing such definitions on a notion of observation is not new, see e.g. [4]. However, there are major differences with previous approaches. To point out one, we briefly address the question of robustness of our notion of bisimilarity. Of course, the results Corollary 19 and Theorem 20 show that the notion is robust across a range of models. But another issue is the sensitivity of our notion of $\mathbf{Pom}_L$-bisimilarity for nets to the particular choice of path category $\mathbf{Pom}_L$. The notion of $\mathbf{Pom}_L$-bisimilarity might seem questionable to those who view general pomsets as not observable.

However, let us define a pomset to be an *almost totally ordered multiset* iff it is of one of the two simple forms considered in the proof of Proposition 16, i.e. allowing at most two (maximal) elements to be unordered. Note that in the range of subclasses of *pomsets* considered in the literature, [19], this class is as close to $\mathbf{Bran}_L$ as one can get! Let us denote the full subcategory of $\mathbf{Pom}_L$ consisting of object of this simple form by $\mathbf{Atom}_L$.

Corollary 21. (i) *A morphism in* $\mathbf{N}_L$ *is* $\mathbf{Pom}_L$*-open iff it is* $\mathbf{Atom}_L$*-open.*

(ii) *Two nets are* $\mathbf{Pom}_L$*-bisimilar iff they are* $\mathbf{Atom}_L$*-bisimilar.*

Proof. Clearly (ii) follows from (i), so we concentrate on a proof of (i). The "only if" part of (i) follows immediately from definition of open maps. By inspecting the proof of Proposition 16, we observe that a morphism in $\mathbf{A}_L$ is $\mathbf{Pom}_L$-open if it is $\mathbf{Atom}_L$-open. By Proposition 13, a similar statement also holds of the category $\mathbf{A}_L^0$. Finally, a similar statement (the "if" part of (i)) holds also in $\mathbf{N}_L$ by Lemma 14. □

Remark. Similar results hold for the alternative category of Petri nets mentioned in Section 1.2. In particular, because there is also a coreflection between event structures and that category, $\mathbf{Pom}_L$-bisimilarity of nets in that framework will also amount to strong history-preserving bisimilarity of their event-structure unfoldings – another example of the robustness of the definitions.

3. Concluding remarks

We have illustrated how to introduce bisimilarity for Petri nets following a general pattern, a pattern which automatically guarantees consistency with bisimilarity on

a number of related models. This sets the scene, but many questions are left open, including a theory of our bisimulation for nets parallelling the well established theory of bisimulation for transition systems. Some initial ideas may be found in the game theoretic and logical characterizations for $\mathbf{Pom}_L$-bisimilation for transition systems with independence given in [15], which may be transferred immediately to nets, following the results of this paper. A particular unresolved issue is that of the decidability of our $\mathbf{Pom}_L$-bisimilarity on finite nets and asynchronous transition systems.

References

[1] M.A. Bednarczyk, Categories of asynchronous systems, Ph.D. thesis in Computer Science, Univ. of Sussex, Report No. 1/88, 1988.

[2] D.B. Benson and O. Ben-Shachar, Bisimulation of automata, *Inform. and Comput.* **79** (1988) 60–83.

[3] I. Castellani, Bisimulation and abstraction homomorphisms, *Proc. CAAP 85*, Lecture Notes in Computer Science, Vol. 185 (Springer, Berlin, 1985) 223–238.

[4] P. Degano, R. De Nicola and U. Montanari, Observational equivalences for concurrency models, in Wirsing, M. ed., *Formal Description of Programming Concepts – III*, IFIP (Elsevier, Amsterdam, 1987) 105–132.

[5] J.-Y. Girard, Linear logic, *Theoret. Comput. Sci.* **50** (1987) 1–102.

[6] R.J. van Glabbeek and U. Goltz, Equivalence notions for concurrent systems and refinement of actions, *Proc. MFCS*, Lecture Notes in Computer Science, Vol. 379, (1989) 237–248.

[7] A. Joyal and J. Moerdijk, A completeness theorem for open maps, *Ann. Pure Appl. Logic* **70** (1994) 51–86.

[8] A. Joyal, M. Nielsen and G. Winskel, Bisimulation from open maps, *LICS93* BRICS Report RS-94-7, Aarhus Univ., 1994.

[9] S. MacLane, *Categories for the Working Mathematician*, Graduate Texts in Mathematics (Springer, Berlin, 1971).

[10] A. Mazurkiewicz, Basic notions of trace theory, in: de Bakker, de Roever and Rozenberg, eds., *Linear Time, Branching Time and Partial Orders in Logics and Models for Concurrency*, Lecture Notes in Computer Science, Vol. 354 (Springer, Berlin, 1988) 285–263.

[11] E. Mendelson, *Introduction to Mathematical Logic* (Van Nostrand Reinhold, New York, 1964).

[12] A.R.G. Milner, *Communication and Concurrency* (Prentice-Hall, Englewood Cliffs, NJ, 1989).

[13] M. Mukund, Petri nets and step transition systems, *Int. J. Found. Comput. Sci.* **3** (1992) 443–478.

[14] M. Mukund, and M. Nielsen, CCS, locations and asynchronous transition systems, in: R. Shyamasundar, ed., *FST & TCS 92*, Lecture Notes in Computer Science, Vol. 652 (Springer, Berlin, 1992) 328–341.

[15] M. Nielsen and C. Clausen, Bisimulations, Games, and Logic, *Proc. CONCUR'94*, Lecture Notes in Computer Science, Vol. 836 (Springer, Berlin 1994) 385–400.

[16] M. Nielsen, G. Plotkin and G. Winskel, Petri nets, Event structures and Domains, Part 1, *Theoret. Comput. Sci.* **13** (1981) 85–108.

[17] M. Nielsen, G. Rozenberg, P.S. Thiagarajan, Elementary transition systems, *Theoret. Comput. Sci.* **96** (1992) 3–33.

[18] E.R. Olderog, *Nets, Terms and Formulas*, Cambridge Tracts in Theoretical Computer Science (1991).

[19] V.R. Pratt, Modelling concurrency with partial orders, *Int. J. Parallel Programming* **15** (1986) 33–71.

[20] A. Rabinovich and B. Trakhtenbrot, Behaviour structures and nets, *Fundam. Inform.* **11** (1988) 357–404.

[21] K. Segerberg, Decidbility of S4.1, *Theoria* **34** (1968) 7–20.

[22] M.W. Shields, Concurrent machines, *Comput. J.* **28** (1985) 449–465.

[23] J. Van Bentham, Correspondence theory, in: Gabbay and Guenther, eds., *Handbook of Philosophical Logic, Vol. II* (Reidel, Dordrecht, 1984) 167–247.

[24] W. Vogler, Deciding history preserving bisimilarity, *Proc. ICALP 91*, Lecture Notes in Computer Science, Vol. 510 (Springer, Berlin, 1991) 495–505.

[25] G. Winskel, Event structures, Lecture Notes in Computer Science, Vol. 255 (Springer, Berlin, 1987) 325–392.

[26] G. Winskel, Petri nets, algebras, morphisms and compositionality, *Inform. and Comput.* **72** (1987) 197–238.

[27] G. Winskel and M. Nielsen, Models for concurrency, in: Abramsky, Gabbay and Maibaum, eds., *Handbook of Logic in Computer Science, Vol. 4* (Oxford University Press, Oxford, 1995) 1–148.