Theoretical Computer Science 71 (1990) 133-153
North-Holland

SOME BEHAVIOURAL ASPECTS OF NET THEORY

P.S. THIAGARAJAN
The Institute of Mathematical Sciences, Madras 600 113, India

0. Introduction

Net theory was initiated by C.A. Petri in the early 60s [29]. The subject matter of the theory is distributed systems and processes. The key aspect of net theory is that the three fundamental relationships that can exist between the occurrences of a pair of actions at a state are clearly separated from each other at all levels of the theory. These three relationships are

(i) at the state s, the action a_2 can occur only after the action a_1 has occurred (causality);

(ii) a_1 can occur or a_2 can occur at s but not both (conflict, choice, indeterminacy);

(iii) at the state s both a_1 and a_2 can occur but with *no order* over their occurrences (concurrency).

Another important feature of net theory is that states and changes-of-states (called transitions) are viewed as two intertwined but distinct entities; they are treated on an "equal" footing by the theory.

Over the years net theory has evolved along many directions. It is difficult to give an overview of the whole theory in one place. Hence we shall attempt to do something more modest here. We shall first convey the basic concerns of net theory by presenting a simple system model called elementary net systems. Then we shall give a brief sketch of some of the tools that have been proposed to describe the *behaviour* of elementary net systems. We shall concentrate on those tools that have either directly come out of net theory or which have been prodded into existence by the insistence of net theory that causality, conflict and concurrency should be clearly separated from each other in behavioural descriptions of distributed systems.

In our presentation we will concentrate on motivations and basic definitions at the expense of stating theorems. The few results that we present are stated without proofs. The proofs can be found in [26]. We shall however leave a trail of pointers to the literature, using which the interested reader can get a reasonable overview of net theory and related topics.

In the next section the elementary net system model is presented. Using this model we then define the basic concepts of net theory. This sets the stage for developing the behavioural tools that can capture the essential features of distributed systems

as defined by the elementary net system model. Section 2 develops some notation and introduces a purely sequential mode of behavioural description called firing sequences. In the subsequent section the theory of traces which have an independent existence is used to recover information concerning concurrency from the firing sequences. In Section 4 the notion of nonsequential processes is introduced. Nonsequential processes are a behavioural tool developed within net theory to describe the nonsequential stretches of behaviour of an elementary net system.

Both trace theory and the theory of nonsequential processes represent concurrency directly but handle information concerning conflict in an indirect fashion. One must work with the whole set of traces or nonsequential processes in order to talk about conflicts. This disadvantage can be overcome with the help of behavioural tools called unfoldings and labelled event structures that are presented in Section 5. The unfolding of an elementary net system is a single object in which all the basic behavioural features of the system are represented in a transparent fashion. Labelled event structures are direct descendants of unfoldings and they are more pleasing mathematical objects.

1. Elementary net systems

Elementary net systems, as the name suggests, are meant to be the simplest system model of net theory. They may be viewed as transition systems obeying a particular principle of change. This view of elementary net systems is explained in more detail in [36]. Here, for the sake of brevity, we shall make a direct presentation.

Definition 1.1. A *net* is a triple $N=(S, T, F)$ where S and T are sets and $F \subseteq (S \times T) \cup (T \times S)$ is such that

(i) $S \cap T = \emptyset$

(ii) $\text{domain}(F) \cup \text{range}(F) = S \cup T$ where

$$\text{domain}(F) = \{x \mid \exists y.(x, y) \in F\} \quad \text{and}$$

$$\text{range}(F) = \{y \mid \exists x.(x, y) \in F\}.$$

Thus a net may be viewed as a directed bipartite graph with no isolated elements. Note that we admit the *empty* net $N_0 = (\emptyset, \emptyset, \emptyset)$.

S is the set of *S-elements*, T is the set of *T-elements* and F is the *flow relation* of the net $N=(S, T, F)$. In diagrams the S-elements will be drawn as circles, the T-elements as boxes and the elements of the flow relation as directed arcs. An example of a net is shown in Fig. 1.

In this paper, unless otherwise stated, the S-elements will be used to denote the local atomic states called *conditions* and the T-elements will be used to denote local atomic changes-of-states called *events*. The flow relation will model a *fixed* neighbourhood relation between the conditions and events of a system. Following usual

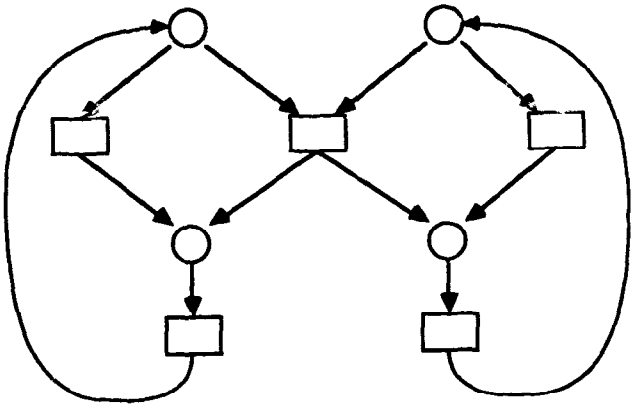

Fig. 1.

practice, we shall represent such nets of conditions and events by triples of the form $N = (B, E, F)$.

Let $N = (B, E, F)$ be a net. Then $X_N = B \cup E$ is the set of *elements* of N. Let $x \in X_N$. Then

$$^{\bullet}x = \{y \mid (y, x) \in F\} \quad \text{(the set of } \textit{pre-elements} \text{ of } x\text{)},$$

$$x^{\bullet} = \{y \mid (x, y) \in F\} \quad \text{(the set of } \textit{post-elements} \text{ of } x\text{)}.$$

This "dot" notation is extended to subsets of X_N in the obvious way. For $e \in E$ we shall call $^{\bullet}e$ the set of *pre-conditions* of e and we shall call $e^{\bullet}$ the set of *post-conditions* of e.

Definition 1.2. An *elementary net system* is a quadruple $\mathcal{N} = (B, E, F, c_{in})$ where

(i) $N_{\mathcal{N}} = (B, E, F)$ is a net called the *underlying net* of $\mathcal{N}$.

(ii) $c_{in} \subseteq B$ is the *initial case* of $\mathcal{N}$.

In diagrams the initial case will be shown by "marking" the members of c_{in}. In Fig. 2 is an example of an elementary net system. Through the rest of the paper we shall refer to this net system as $\mathcal{N}_2$.

In most of what follows, we will only deal with elementary net systems. Hence we will refer to them as net systems. The dynamics of a net system are simple. A state (usually called a *case*) of the system consists of a set of conditions holding concurrently. An event can occur at a case iff all its pre-conditions and none of its post-conditions hold at the case. When an event occurs each of its pre-conditions ceases to hold and each of its post-conditions begins to hold. This simple and restrictive notion of states and changes-of-states leads to a surprisingly rich and sophisticated class of objects. Indeed one of our aims here is to convince the reader that the essential features of distributed systems can be isolated and studied using net systems. First however we must formalize the dynamics of net systems.

Let $N = (B, E, F)$ be a net. Then $\rightarrow_N \subseteq 2^B \times E \times 2^B$ is the (*elementary*) *transition relation generated by* N and is given by

$$\rightarrow_N = \{(k, e, k') \mid k - k' = {}^{\bullet}e \wedge k' - k = e^{\bullet}\}$$

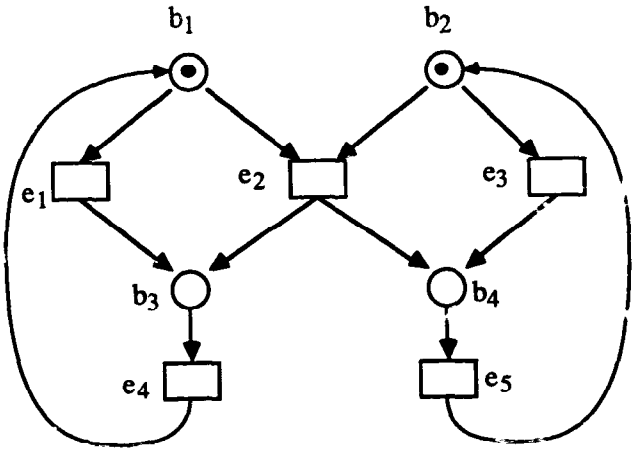

Fig. 2.

Definition 1.3. Let $\mathcal{N} = (B, E, F, c_{in})$ be a net system.

(i) $C_{\mathcal{N}}$, *the state space of* $\mathcal{N}$ is the least subset of 2^B containing c_{in} such that if $c \in C_{\mathcal{N}}$ and $(c, e, c') \in \rightarrow_{N_{\mathcal{N}}}$ then $c' \in C_{\mathcal{N}}$.

(ii) $TS_{\mathcal{N}} = (C_{\mathcal{N}}, E, \rightarrow_{\mathcal{N}})$ is *the transition system associated with* $\mathcal{N}$ where $\rightarrow_{\mathcal{N}}$ is $\rightarrow_{N_{\mathcal{N}}}$ restricted to $C_{\mathcal{N}} \times E \times C_{\mathcal{N}}$.

For the system $\mathcal{N}_2$ shown in Fig. 2, $\{\{b_1, b_2\}, \{b_1, b_4\}, \{b_2, b_3\}, \{b_3, b_4\}\}$ is its state space. We recall that a transition system is a triple $TS = (S, A, \rightarrow)$ where S is a set of states, A is a set of actions and $\rightarrow \subseteq S \times A \times S$ is the (labelled) transition relation. According to the above definition there is a natural way of explaining the dynamics of a net system with the help of a transition system. We are now in a position to bring out the particular and restricted notion of change adopted in net theory. Before doing so it will be convenient to adopt some notations.

Let $\mathcal{N} = (B, E, F, c_{in})$ be a net system, $c \in C_{\mathcal{N}}$ and $e \in E$. Then e is said to be *enabled* at c—denoted $c[e\rangle$—iff there exists $c' \in C_{\mathcal{N}}$ such that $(c, e, c') \in \rightarrow_{\mathcal{N}}$. We shall often write $c \xrightarrow{e} c'$ in place of $(c, e, c') \in \rightarrow_{\mathcal{N}}$.

Proposition 1.4. *Let* $\mathcal{N} = (B, E, F, c_{in})$ *be a net system,* $e \in E$ *and* c, c', c_1, *etc. members of* $C_{\mathcal{N}}$. *Then the following statements hold*:

(i) $c_1 \xrightarrow{e} c_2 \wedge c_3 \xrightarrow{e} c_4 \Rightarrow c_1 - c_2 = c_3 - c_4 \wedge c_2 - c_1 = c_4 - c_3$,

(ii) $c[e\rangle \Leftrightarrow {}^\bullet e \subseteq c \wedge e^\bullet \cap c = \emptyset$,

(iii) $c \xrightarrow{e} c' \wedge c \xrightarrow{e} c'' \Rightarrow c' = c''$.

(i) says that an event causes the same change in the system state whenever it occurs; its pre-conditions cease to hold and its post-conditions begin to hold.

(ii) says that an event is enabled at a case *if* and only if the fixed change associated with its occurrence is possible at the case. Thus no "side-conditions" are involved

in the enabling of an event. Net systems are in this sense clean flow models with the result that they are amenable to analysis using the basic techniques of linear algebra [22].

(iii) says that the transition systems associated with net systems are *deterministic*. Hence in order to connect up with other approaches to the theory of distributed systems such as CCS [25] or CSP [16] one must go over to *labelled* net systems. When one does so, it is possible to give an operational semantics for CCS-like processes in terms of labelled net systems. The interested reader can consult [8, 28].

Here we wish to emphasize that in net theory, the act of *labelling* is considered to be a step towards abstraction. Stated differently, the theory provides for and indeed starts at a primitive level of system modelling where the "bare skeleton" of a distributed system is described and studied. The advantage of starting this way is that the basic concepts concerning the behaviour of distributed systems can be captured—and separated from each other—in a clean way as we shall now see.

Through the rest of this section we fix a net system $\mathcal{N} = (B, E, F, c_{in})$. We let e, e', e_1, e_2 range over E, c, c', c'', c_1, c_2 and c_3 range over $C_{\mathcal{N}}$.

Let $e_1 \neq e_2$. We say that e_1 and e_2 can occur *concurrently* at c—denoted $c[\{e_1, e_2\}\rangle$—iff $c[e_1\rangle$ and $c[e_2\rangle$ and $({}^\bullet e_1 \cup e_1^\bullet) \cap ({}^\bullet e_2 \cup e_2^\bullet) = \emptyset$.

Thus e_1 and e_2 can occur concurrently at a case iff they can occur individually and their "neighbourhoods" are disjoint. When we say that e_1 and e_2 can occur concurrently, what we mean is that they can occur with *no order* over their occurrences. Hence net systems can in general display nonsequential patterns of behaviour. For the system $\mathcal{N}_2$, at the initial case e_1 and e_3 can occur concurrently. This notion of concurrency between a pair of event occurrences can be extended to a set of events in an obvious way. One then obtains the notion of *a step* and indeed one can define a transition relation between cases based on the notion of steps (see [36]).

Concurrency as defined above at once gives rise to the notion of conflict.

Let $e_1 \neq e_2$. e_1 and e_2 are said to be *in conflict at* c iff $c[e_1\rangle$ and $c[e_2\rangle$ but *not* $(c[\{e_1, e_2\}\rangle)$.

For the system $\mathcal{N}_2$, at the initial case e_1 and e_2 (as well as e_2 and e_3) are in conflict. If two events are in conflict at a case then either one of them may occur but not both. Thus net systems can display indeterminate behaviours. Conflict situations can be used to model the flow of information between a system and its environment. Wherein conflict and concurrency "overlap" there can be uncertainty regarding information flow. This situation is known as confusion. Before formalizing the notion of confusion, let us consider two examples.

For the system $\mathcal{N}_2$, let $c = \{b_1, b_2\}$, $c' = \{b_3, b_4\}$. It is clear that e_1 and e_3 can occur concurrently at c to lead the system from c to c'. Two sequential observers reporting on this transformation could claim the following.

Observer O_1: The conflict between e_1 and e_2 at c was resolved in favour of e_1 which then occurred to lead the system to the state $c_1 = \{b_2, b_3\}$. At c_1, the event e_2 occurred without being in conflict with any event and this led the system to the state c'.

Observer O_2: The conflict between e_2 and e_3 at c was resolved in favour of e_3 which then occurred to lead the system to the state $c_3 = \{b_1, b_4\}$. At c_3, the event e_1 occurred without being in conflict with any event and this led the system to the state c'.

Thus the confusion here is over *which* conflict was resolved in going from c to c'. This type of confusion is often referred to as *symmetric* confusion. In Fig. 3 a different kind of confusion is shown often referred to as *asymmetric confusion.*

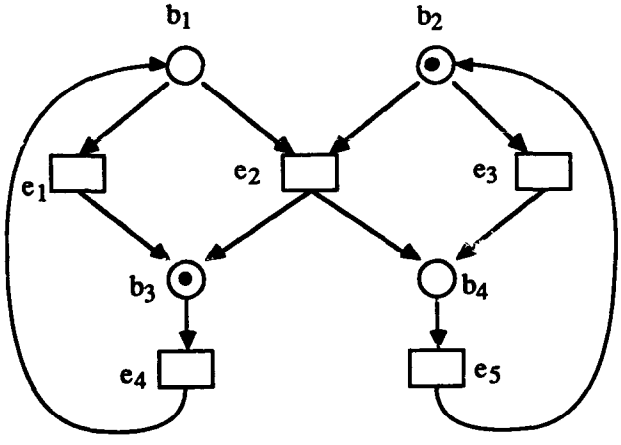

Fig. 3.

Let $c = \{b_2, b_3\}$ and $c' = \{b_1, b_4\}$. Clearly e_3 and e_4 can occur concurrently at c to lead the system from c to c'. The confusion here is regarding *whether or not* a conflict (between e_2 and e_3) was resolved in going from c to c'. The observer who records the occurrence of e_4 first will claim that a conflict was resolved whereas the observer who records the occurrence of e_3 first will claim that no conflict was resolved. In general, confusion can be a mixture of both types of confusion outlined above and the general definition is as follows.

Let $c[e\rangle$. Then

$$cfl(e, c) = \{e' \mid e \text{ and } e' \text{ are in conflict at } c\}$$

We say that (c, e_1, e_2) is a *confusion* iff

(i) $c[\{e_1, e_2\}\rangle$,

(ii) $cfl(e_1, c) \neq cfl(e_1, c_2)$ where $c \xrightarrow{e_2} c_2$.

It seems safe to assert that distributed systems are difficult to implement and analyze mainly because of the problem of confusion. Net theory provides some strong positive evidence in support of this claim. It turns out that systems that are confusion-free admit a nice theory. More precisely one can identify a large subclass of confusion-free net systems by placing a simple restriction on the underlying nets. And this subclass has a nice theory. In fact we can identify subclasses of *sequential* (concurrency-free), *determinate* (conflict-free) and *confusion-free* net systems by

requiring the underlying nets to be S-graphs, T-graphs and Free-choice nets respectively. Actually, in the case of a sequential system one must require the underlying S-graph to be connected and one must also require that exactly one condition holds at the initial case. Here are the definitions of the three net classes.

(1) An *S-graph* is a net $N=(B, E, F)$ such that $\forall e \in E.\ |{}^\bullet e| = 1 = |e^\bullet|$.

(2) A *T-graph* is a net $N=(B, E, F)$ such that $\forall b \in B.\ |{}^\bullet b| = 1 = |b^\bullet|$.

(3) A *Free-choice net* is a net $N=(B, E, F)$ such that $\forall b \in B.\ \forall e \in E.\ (b, e) \in F \Rightarrow b^\bullet = \{e\} \vee \{b\} = {}^\bullet e$.

It is easy to check that every S-graph as well as every T-graph is a Free-choice net but the converse is not true in general. Clearly, not every S-graph (T-graph) is a T-graph (S-graph). In Fig. 4 examples are shown of net systems based on the three kinds of nets. The interested reader can verify that the system shown in Fig. 4(a) (Fig. 4(b), (c)) exhibits no concurrency (no conflict, no confusion) within its state space.

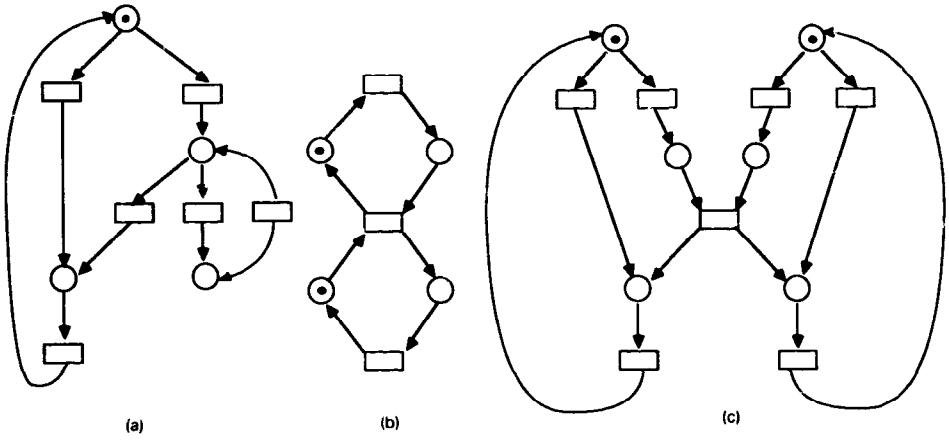

Fig. 4.

Net systems based on S-graphs essentially correspond to sequential state machines. Net systems based on T-graphs are known—in a larger context—as *marked graphs* and their theory is very well understood [7, 12, 20]. What is surprising is that net-systems-based Free-choice nets also admit a beautiful theory [15, 38]. Thus net theory suggests that it is not the combination of concurrency and conflicts as such that causes problems. It is only when these two phenomena combine to produce confusion that life becomes difficult.

Before concluding this section we wish to point out that the elementary net system model can be generalized in a variety of ways. One obvious and popular generalization leads to a model known as *Petri nets* but which we prefer to called *marked nets*.

Let $N=(S, T, F)$ be a net. Then a *marking* of N is a function $M: S \to \mathbf{N}_0$ ($=\{0, 1, 2, \ldots\}$). The transition $t \in T$ is *enabled* to occur at the marking M—denoted $M[t\rangle$—iff $\forall s \in {}^\bullet t.\ M(s) > 0$. When the enabled transition t occurs at the marking

M, a new marking M' is obtained which is given by

$$\forall s \in S.\quad M'(s) = \begin{cases} M(s)-1 & \text{if } s \in {}^\bullet t - t^\bullet, \\ M(s)+1 & \text{if } s \in t^\bullet - {}^\bullet t, \\ M(s) & \text{otherwise.} \end{cases}$$

The transformation of M into M' by the occurrence of t at M is denoted as $M[t\rangle M'$. A *marked net* is then defined to be a quadruple $MN = (S, T, F, M_{in})$ where $N_{MN} = (S, T, F)$ is a net called the *underlying net* of MN and M_{in} is a marking of N_{MN} called the *initial marking* of MN. The *state space of* MN—denoted $[M_{in}\rangle$—also referred to as the set of *reachable markings of* MN is the least set of markings of N_{MN} containing M_{in} such that if $M \in [M_{in}\rangle$, $t \in T$ and M' is a marking of N_{MN} such that $M[t\rangle M'$, then $M' \in [M_{in}\rangle$.

A slight generalization of marked nets were independently discovered as *vector addition systems* in [21]. Over the years a number of interesting and difficult decision problems concerning marked nets have been studied and solved (see [17] for a limited overview of this topic). Marked nets also have some interesting connections to formal language theory [18].

A second generalization of elementary net systems which is more vital from a practical standpoint was first achieved by Genrich and Lautenbach [13]. The idea is quite simple. Let $\mathcal{N} = (B, E, F, c_{in})$ be an elementary net system. Then B can be viewed as a set of atomic propositions and each $c \in C_{\mathcal{N}}$ can be viewed as a boolean valuation of B. An event then transforms one boolean valuation in $C_{\mathcal{N}}$ into another subject to certain restrictions determined by F, the flow relation. We can now generalize by replacing B by a set of *predicate symbols* P. Instead of $C_{\mathcal{N}}$, we identify the state space to be a set μ of (set-theoretic) *structures* for P with respect to a chosen domain D of individuals. An event then transforms one structure in μ into another subject to certain restrictions imposed by the flow relation F. What one then obtains is a first-order net system which is very rich in expressive power. The model can be made more useful by exploiting the standard notions of first-order logic such as function symbols, constants and individual variables.

The notion of an event however is kept the "same" so that a first-order version of Proposition 1.4 goes through smoothly. As a result, we once again obtain a clean flow model and the tools of linear algebra become available for analysis. First-order net systems come in different forms. The two most well-known versions are known as *Predicate/Transition nets* [11] and *coloured Petri nets* [19]. These models play a crucial role in the applications of net theory [5].

2. The behaviour of net systems: preliminaries

We now wish to survey the concepts and techniques that have been proposed in and around net theory to study the behaviour of distributed systems. We shall do so by providing various answers to the question: What is the behaviour of a net system?

The most primitive behavioural representation is called firing sequences. Here the net system is viewed as generating a set of strings over the events of the system. All information concerning condition-holdings is thrown away. This is a desirable feature in that the states are after all abstract entities whose only role is to "implement" the intended pattern of event occurrences. However, we will show that treating the condition-holdings on par with event occurrences can lead to a number of useful intermediate behavioural representations that are of independent interest. Returning to firing sequences, what they convey is the mere causal ordering over the event occurrences; all information concerning concurrency and conflict(-resolution) is "lost". The various other behavioural tools we shall present can be seen as an attempt to recover this information either partially or completely. Now for some preliminaries.

We fix a net system $N_0 = (B_0, E_0, F_0, c_0)$ for the rest of this section and through the next three sections (up to Section 5). We let b, b', b'' with or without subscripts range over B_0. We let e, e', e'' with or without subscripts range over E_0. We let c, c', c'' with or without subscripts range over C_{N_0} which we shall write, for convenience, as C_0.

In dealing with sequences we shall adopt the following conventions. Given a set of symbols Σ, we let Σ^* denote the free monoid generated by Σ. The null sequence will be represented as Λ. If ρ is a sequence of symbols and x is a symbol, then $\#_x(\rho)$ is the number of times x appears in ρ.

We will also have to deal with labelled posets. Let Σ be a nonempty alphabet set. Then a *Σ-labelled* poset is a triple $\pi = (X, \leq, \varphi)$ where $(X, \leq)$ is a poset and $\varphi : X \to \Sigma$ is a *labelling function.*

Let $\pi = (X, \leq, \varphi)$ be a finite Σ-labelled poset. In other words, π is such that X is a finite set. Then $lo(\pi)$ (the set of *unlabelled linear orders* of π) is the subset of X^* is defined as follows: $\rho \in lo(\pi)$ iff the following conditions are satisfied:

(i) $\forall x \in X.\ \#_x(\rho) = 1$;

(ii) $\forall x, y \in X.\ \forall \rho' \in \text{Prefix}(\rho)[x \leq y \Rightarrow \#_x(\rho') \leq' \#_x(\rho)]$.

Prefix(ρ) is the set of prefixes of ρ and $\leq'$ is the usual ordering over the integers. Consider the Σ-labelled poset $\pi_1 = (X_1, \leq_1, \varphi_1)$ (with $\Sigma = \{a, b\}$) whose Hasse diagram is shown in Fig. 5.

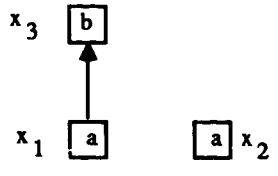

Fig. 5.

We have indicated φ_1 by writing $\varphi_1(x)$ inside the diagram representing x. We will follow this convention through the rest of the paper. It is easy to check that $lo(\pi_1) = \{x_1x_2x_3, x_1x_3x_2, x_2x_1x_3\}$.

For Σ labelled poset $\pi = (X, \leqslant, \varphi)$ we now define $LO(\pi)$ (the set of labelled linear orders of π) as:

$$LO(\pi) = \{\varphi(\rho) \mid \rho \in lo(\pi)\}.$$

Here we have denoted, by abuse of notation, the natural extension of φ to X^* also as φ. For the poset π_1, of Fig. 5, we then have $LO(\pi_1) = \{aab, aba\}$.

We can now introduce the first and the most primitive of our behavioural tools. Actually $TS_{\mathcal{N}}$, the transition system associated with $\mathcal{N}$ can also be viewed as a representation of the behaviour of $\mathcal{N}$. We can however afford to ignore this, given our present aims.

The *set of firing sequences* of $\mathcal{N}_0$—denoted FS_0—is the least subset of E_0^* (recall that $\mathcal{N}_0 = (B_0, E_0, F_0, c_0)$) given by

(i) $\Lambda \in FS_0$ and $c_0[\![\Lambda\rangle c_0$;

(ii) suppose $\rho \in FS_0$, $c_0[\![\rho\rangle c$ and $c \xrightarrow{e} c'$; then $\rho e \in FS_0$ and $c_0[\![\rho e\rangle c'$.

Thus $[\![\ \rangle$ is the obvious "extension" of $\rightarrow_{\mathcal{N}}$ to $\{c_0\} \times E_0^* \times C_0$.

For the system $\mathcal{N}_2$, $e_1e_4e_1$ and $e_3e_1e_5$ are firing sequences. As mentioned earlier, firing sequences "hide" information concerning concurrency and conflict-resolution. We will now see how the theory of traces can be applied to extract information concerning concurrency from the firing sequences.

3. Traces

The theory of traces was introduced by Mazurkiewicz [23] to model the non-sequential behaviour of distributed programs. The basic idea is to postulate an independence relation over the letters of an alphabet. The members of the alphabet represent the actions that can be executed by a program. Two actions that are in the independence relation are supposed to occur concurrently whenever they occur "adjacent" to each other. This induces an equivalence relation over the language which is a sequential description of the behaviour of the program.

Definition 3.1. (i) A *concurrent alphabet* is a pair $Z = (\Sigma, I)$ where Σ is a nonempty alphabet set and $I \subseteq \Sigma \times \Sigma$ is an *irreflexive* and *symmetric independence relation.*

(ii) Let $\rho, \rho' \in \Sigma^*$. Then $\rho \dot{\sim}_I \rho'$ iff there exist $\rho_1, \rho_2 \in \Sigma^*$ and $(a, b) \in I$ such that $\rho = \rho_1 ab \rho_2$ and $\rho' = \rho_1 ba \rho_2$.

(iii) $\sim_I =^{\text{def}} (\dot{\sim}_I)^*$.

It is easy to check that $\sim_I$ as defined above is an equivalence relation. (In fact it is a congruence.) For $\rho \in \Sigma^*$ we denote by $[\rho]_I$ the equivalence class of strings containing ρ; we call it a *trace.* In other words, $[\rho]_I = \{\rho' \mid \rho \sim_I \rho'\}$. Where I is clear from the context we will write $[\rho]$ instead of $[\rho]_I$. The *set of traces* over Σ^* generated by the concurrent alphabet $Z = (\Sigma, I)$ is given by

$$\Sigma^*/\sim_I = \{[\rho] \mid \rho \in \Sigma^*\}.$$

A *trace language* over the concurrent alphabet $Z = (\Sigma, I)$ is simply a subset of $\Sigma^*/\sim_I$.

A good deal of effort has gone into the study of trace languages. A survey of the major results in this area can be found in [1]. A nice application of trace theory to the theory of net systems is presented in [24]. In the recent past, trace languages have also been studied from the standpoint of formal languages. In such studies the term "partially commutative monoids" is used instead of "trace languages" [6]. Pomsets, which are basically labelled posets can be viewed as a generalization of traces. Pomsets form the basis of a theory of distributed systems which is under construction by Pratt [32].

Returning to our main concern, a simple but crucial observation concerning traces is the following.

Proposition 3.2. *Let $Z=(\Sigma, I)$ be a concurrent alphabet and $t \in \Sigma^*/\sim_I$. Then there is a* unique *(upto isomorphism) Σ-labelled poset $\pi=(X, \leq, \varphi)$ such that $LO(\pi)=t$ and $\forall x, y \in X.\ \varphi(x)=\varphi(y) \Rightarrow x \leq y \vee y \leq x$.*

Actually this result can be stated in a more precise form but we will not pause to do so here. The idea should be clear and we will proceed to consider an example.

Let $\Sigma=\{a, b, c\}$ and $I=\{(a, b), (b, a), (b, c), (c, b)\}$. Then $\{abc, bac, acb\}$ is a trace and it is represented by the Σ-labelled poset shown in Fig. 6.

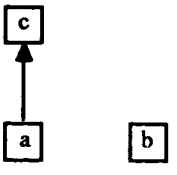

Fig. 6.

We shall introduce one more notion before we relate trace theory to net systems. Let $Z=(\Sigma, I)$ be a concurrent alphabet and let $L \subseteq \Sigma^*$ be a (sequential) language. Then L is *consistent* with I iff $\forall \rho \in L.\ [\rho] \subseteq L$.

Suppose $\Sigma=\{a, b\}$ and $I=\{(a, b), (b, a)\}$. Then clearly $L=\{ab\}$ is *not* consistent with I.

Definition 3.3. (i) $Z_0=(E_0, I_0)$ is *the concurrent alphabet of* $\mathcal{N}_0=(B_0, E_0, F_0, c_0)$ where

$$I_0=\{(e_1, e_2) \mid ({}^\bullet e_1 \cup e_1^\bullet) \cap ({}^\bullet e_2 \cup e_2^\bullet)=\emptyset\}.$$

(ii) The *trace language* of $\mathcal{N}_0$—denoted by T_0—is

$$T_0 \stackrel{\text{def}}{=} \{[\rho] \mid \rho \in FS_0\}.$$

Proposition 3.4. *FS_0 is consistent with I_0.*

It is easy to check that I_0 as specified in Definition 3.3 is irreflexive and symmetric so that Z_0 is indeed a concurrent alphabet. For the net system $\mathcal{N}_2$ its independence relation, denoted as I_2, is given by

$$I_2 = \{(e_1, e_3), (e_3, e_1), (e_4, e_3), (e_3, e_4), (e_1, e_5), (e_5, e_1), (e_4, e_5), (e_5, e_4)\}.$$

$\{e_2e_4e_1e_5, e_2e_4e_5e_1, e_2e_5e_4e_1\}$ is a member of $T_{\mathcal{N}_2}$. The labelled poset representation of this trace is shown in Fig. 7. As seen in this diagram, the trace theory formalism enables us to reconstruct information concerning concurrency from the firing sequences of $\mathcal{N}_0$ via the independence relation I_0. It is important to note that I_0 depends purely on the underlying net of $\mathcal{N}_0$.

It turns out that T_0 also contains information regarding conflict resolution. To extract this, we need an ordering relation over T_0. Let $t_1, t_2 \in T_0$. Then

$$t_1 \sqsubseteq_0 t_2 \overset{\text{def}}{\Leftrightarrow} \forall \rho \in t_1.\ \exists \rho' \in t_2.\ \rho \leqslant \rho'.$$

Here $\leqslant$ stands for the usual prefix ordering over E_0^*. It is straightforward to verify that $\sqsubseteq_0$ is a partial ordering relation. In Fig. 8 we show an initial fragment of the poset (of traces) for the system $\mathcal{N}_2$. For convenience, each trace has been specified by a representative member of the trace.

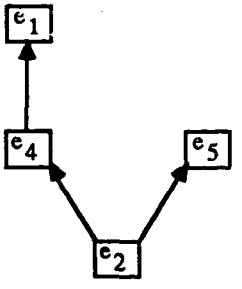

Fig. 7.

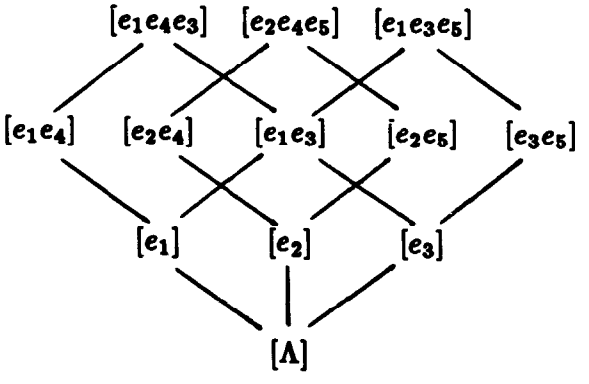

Fig. 8.

Let $t_1, t_2 \in T_0$. Then we say that t_1 and t_2 are *compatible*—and this is denoted $t_1 \uparrow t_2$—if there exists a $t \in T_0$ such that $t_1 \sqsubseteq_0 t$ and $t_2 \sqsubseteq_0 t$. We shall write $t_1 \not\uparrow t_2$ to denote the fact that t_1 and t_2 are *not* compatible.

We claim that the relation $\not\uparrow$ reflects all information concerning conflicts and their resolution. To substantiate this claim however, we must wait until event structures have been introduced. Here we shall only indicate that the relation $\not\uparrow$ carries *some* information concerning conflict.

Proposition 3.5. *Let $t_1, t_2 \in T_0$. Then $t_1 \not\uparrow t_2$ if there exist $\rho \in FS_0$ and $e_1, e_2 \in E_0$ such that the following conditions are fulfilled:*

(i) $\rho e_1, \rho e_2 \in FS_0$;

(ii) $[\rho e_1] \sqsubseteq_0 t_1$ *and* $[\rho e_2] \sqsubseteq_0 t_2$;

(iii) e_1 *and* e_2 *are in conflict at* c *where* $c_0 [\![\rho \rangle c$.

We propose that the poset $(T_0, \sqsubseteq_0)$ is a behavioural representation of $\mathcal{N}_0$ which captures all the features of causality, concurrency and conflict that arise during the history of $\mathcal{N}_0$. Our next task will be to obtain an alternative representation which is quite different in spirit but which will "agree" with the information provided by $(T_0, \sqsubseteq_0)$.

4. Nonsequential processes

Petri suggested that certain kinds of labelled nets called nonsequential processes should be used to describe the behaviour of net systems [30]. Before presenting this idea, we need to impose a restriction on net systems.

The net system $\mathcal{N} = (B, E, F, c_{in})$ is said to be *contact-free* iff

$$\forall c \in C_{\mathcal{N}}. \forall e \in E. \quad [{}^\bullet e \subseteq c \Rightarrow e^\bullet \cap c = \emptyset].$$

We will assume the generic net system $\mathcal{N}_0$ whose behaviour is under study to be contact-free. This does not involve any loss of generality. It turns out that every net system $\mathcal{N}$ can be converted into a contact-free net system $\mathcal{N}'$ such that $\mathcal{N}$ and $\mathcal{N}'$ are "behaviourally equivalent" in a strong sense. The interested reader is referred to [33] for details. Here we shall illustrate the principle with the help of an example. In Fig. 9 we show a net system (which is not contact-free) and its contact-free equivalent.

Note that in a contact-free net system an event is enabled at a case iff all its pre-conditions hold. Similarly, the definitions of concurrency, conflict and confusion become much simpler and more intuitively appealing in the absence of contact. Clearly, the system $\mathcal{N}_2$ is contact-free.

Next we need the notion of a labelled net.

A *Σ-labelled* net is a quadruple $N = (B, E, F, \varphi)$ where (B, E, F) is a net and $\varphi : B \cup E \to \Sigma$ is the *labelling function.*

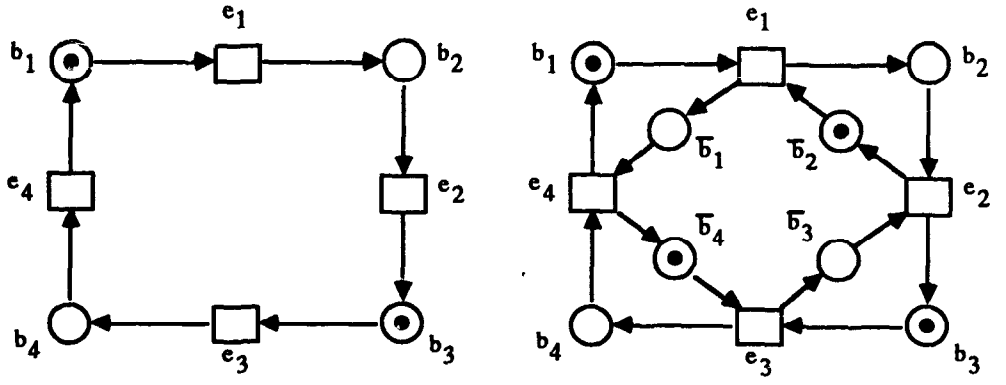

Fig. 9.

A nonsequential process of the net system $\mathcal{N}_0$ will be an X_0-labelled net $N = (B, E, F, \varphi)$ in which F and φ are required to satisfy a number of requirements. (Here $X_0 = B_0 \cup E_0$.) For instance, one requires F^* to be a p.o. relation and one demands $\varphi(B) \subseteq B_0$ and $\varphi(E) \subseteq E_0$. For our purposes it will be convenient to associate a nonsequential process with each firing sequence. This will enable us to build them up inductively. For a similar approach to the construction of processes, see [4]. More, our method of construction will directly lead to yet another behavioural representation called the unfolding. From now on we shall refer to nonsequential processes as processes. An example of a process of $\mathcal{N}_2$ is shown in Fig. 10.

As already mentioned, for each firing sequence ρ of $\mathcal{N}_0$ we will construct an X_0-labelled net $N_\rho = (B_\rho, E_\rho, F_\rho, \varphi_\rho)$ and call it a process of $\mathcal{N}_0$. Each member of $B_\rho \cup E_\rho$ will be of the form (y, Y) with $y \in X_0$ and $Y \subseteq B_\rho \cup E_\rho$. The labelling function will be the obvious projection operator; for each $(y, Y) \in B_\rho \cup E_\rho$ it will be the case that $\varphi_\rho((y, Y)) = y$. Hence in what follows we will suppress φ_ρ.

The idea is that for each $(b, X) \in B_\rho$ the set X will be a record of the unique history of $\mathcal{N}_0$ that led to this particular holding of b. Similarly, for $(e, X) \in E_\rho$ the set X will record the unique history that led to this particular enabling of e.

The construction of N_ρ is by induction on $|\rho|$. For convenience we will keep track of the conditions that hold in $\mathcal{N}_0$ after the run represented by the firing sequence ρ. This set of conditions will be encoded as c_ρ.

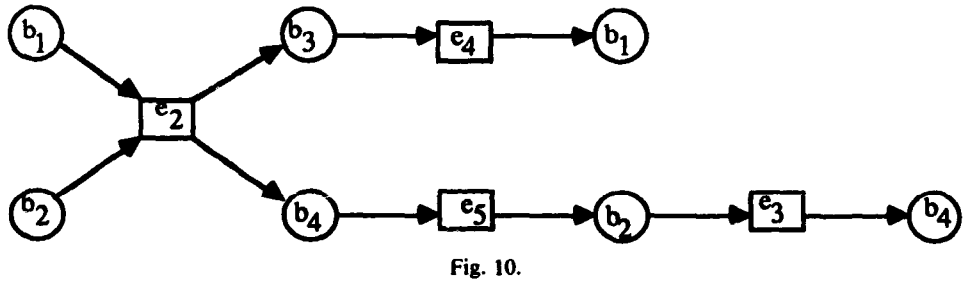

Fig. 10.

Definition 4.1. Let $\rho \in FS_0$. Then $N_\rho = (B_\rho, E_\rho, F_\rho)$ is given by:

(i) $|\rho| = 0$: $N_\rho = (\emptyset, \emptyset, \emptyset)$ and $c_\rho = \{(b, \emptyset) \mid b \in c_0\}$ (recall that $\mathcal{N}_0 = (B_0, E_0, F_0, c_0)$).

(ii) $|\rho| > 0$: Let $\rho = \rho' e$ and assume that $N_{\rho'} = (B_{\rho'}, E_{\rho'}, F_{\rho'})$ and $c_{\rho'}$ are defined. Then $N_\rho = (B_\rho, E_\rho, F_\rho)$ is given by

(i) $E_\rho = E_{\rho'} \cup \{(e, X)\}$ where $X = \{(b, D) \mid b \in {}^\bullet e \wedge (b, D) \in c_{\rho'}\}$,

(ii) $B_\rho = B_{\rho'} \cup X \cup Y$ where $Y = \{(b, \{(e, X)\}) \mid b \in e^\bullet\}$,

(iii) $F_\rho = F_{\rho'} \cup (X \times \{(e, X)\}) \cup (\{(e, X)\} \times Y)$,

(iv) $c_\rho = (c_{\rho'} - X) \cup Y$.

N_ρ (with the obvious projection operator as the labelling function) is called a *process* of $\mathcal{N}_0$. We let P_0 denote the set of processes of $\mathcal{N}_0$ where

$$P_0 \overset{\text{def}}{=} \{N_\rho \mid \rho \in FS_0\}.$$

Actually P_0 just denotes the set of *finite processes* of $\mathcal{N}_0$ but for our current purposes they will do.

It is easy to see that there is a close relationship between the processes and traces of $\mathcal{N}_0$. In order to state this relationship in a strong way, we define an "inclusion" relation $\subseteq' \subseteq P_0 \times P_0$ as

$$N_\rho = (B_\rho, E_\rho, F_\rho) \subseteq' N_{\rho'} = (B_{\rho'}, E_{\rho'}, F_{\rho'})$$
$$\text{iff } B_\rho \subseteq B_{\rho'} \text{ and } E_\rho \subseteq E_{\rho'} \text{ and } F_\rho \subseteq F_{\rho'}.$$

Theorem 4.2. *$(P_0, \subseteq')$ and $(T_0, \sqsubseteq_0)$ are isomorphic posets. In fact, $f: P_0 \to T_0$ given by $f(N_\rho) = [\rho]$ is an isomorphism.*

The underlying nets of the processes of net systems are interesting objects in their own right. We shall call them *causal nets.* A causal net is a net $N = (B, E, F)$ such that

(i) $\forall b \in B. |{}^\bullet b|, |b^\bullet| \leq 1$,

(ii) F^* is a partial ordering relation over $X = B \cup E$.

Proposition 4.3. *The underlying net of each process of $\mathcal{N}_0$ is a causal net.*

An example of an infinite causal net is shown in Fig. 11.

Causal nets are interesting because they can be used to study concurrency in isolation from conflict. To see this let $N = (B, E, F)$ be a causal net and let $\leq = F^*$.

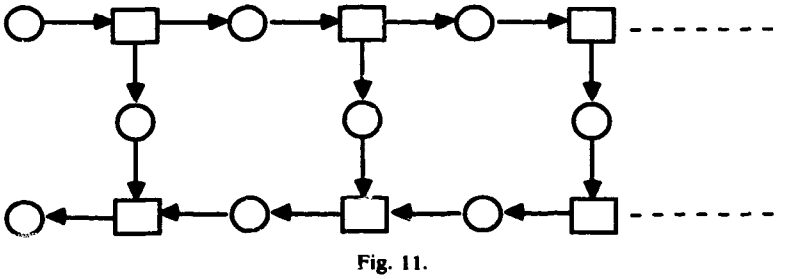

Fig. 11.

Then we can define the concurrency relation *co* as

$$\forall x, y \in B \cup E.\quad x \,co\, y \overset{\text{def}}{\Leftrightarrow} x \not< y \wedge y \not< x.$$

Thus in the setting of causal nets, concurrency just expresses the absence of causality and causality is simply a partial ordering relation. Hence the theory of posets can be applied to study the *co*-relation. This part of net theory was initiated by Petri [31]. A variety of density properties for causal nets have been proposed and their interrelationships have been investigated [3, 9, 10]. Returning to our main theme we are now ready to present unfoldings and labelled event structures.

5. Labelled event structures

Due to Theorem 4.2 the poset $(P_0, \subseteq')$ also contains information about conflicts and their resolutions. In a seminal paper, Nielsen, Plotkin and Winskel showed—among other things—how to "glue" together the elements of P_0 into a single object in which causality, concurrency and conflict are represented explicitly [27].

Definition 5.1. Let $N_\rho = (B_\rho, E_\rho, F_\rho)$ be the process associated with $\rho \in FS_0$. Then the *unfolding* of $\mathcal{N}_0$ is the triple $UF_0 = (\hat{B}_0, \hat{E}_0, \hat{F}_0)$ where

(i) $\hat{B}_0 = \bigcup \{B_\rho \mid \rho \in FS_0\}$,
(ii) $\hat{E}_0 = \bigcup \{E_\rho \mid \rho \in FS_0\}$,
(iii) $\hat{F}_0 = \bigcup \{F_\rho \mid \rho \in FS_0\}$.

As before, the labelling function is the obvious projection operator and we have suppressed it. An initial fragment of the unfolding of $\mathcal{N}_2$ is shown in Fig. 12. As this example shows, the unfolding of a net system will be in general an infinite object.

The unfolding of a net system presents a single record of all the runs of the system. In this record each occurrence of an element of the net system (condition-holding or event occurrence) is recorded separately so that the unique—in general—nonsequential history that led to this occurrence lies in its past. The underlying nets of the unfoldings of net systems are called *occurrence nets.*

Before we present the notion of occurrence nets it will be convenient to adopt some notations concerning posets. Let $\pi = (X, \leqslant)$ be a poset and $Y \subseteq X$. Then

$$\downarrow Y = \{x \mid \exists y \in Y.\, x \leqslant y\} \quad \text{and} \quad \uparrow Y = \{x \mid \exists y \in Y.\, y \leqslant x\}.$$

If $Y = \{y\}$ is a singleton, we will write $\uparrow y$ and $\downarrow y$ instead of $\uparrow\{y\}$ and $\downarrow\{y\}$ respectively. For $x, y \in X$, $x \uparrow y$ will denote the fact that there exists $z \in X$ such that $x \leqslant z$ and $y \leqslant z$. Finally $x \not\uparrow y$ will denote the negation of $x \uparrow y$.

An *occurrence net* is a net $N = (B, E, F)$ such that

(i) $\forall b \in B.\ |{}^\bullet b| \leqslant 1$;
(ii) $\leqslant_N \overset{\text{def}}{=} F^*$ is a partial ordering relation over X_N;
(iii) $\forall e_1, e_2 \in E.\ [e_1 \neq e_2 \wedge {}^\bullet e_1 \cap {}^\bullet e_2 \neq \emptyset \Rightarrow \uparrow e_1 \cap \uparrow e_2 = \emptyset]$ (where $\uparrow e$ is defined w.r.t. the ordering relation $\leqslant_N$).

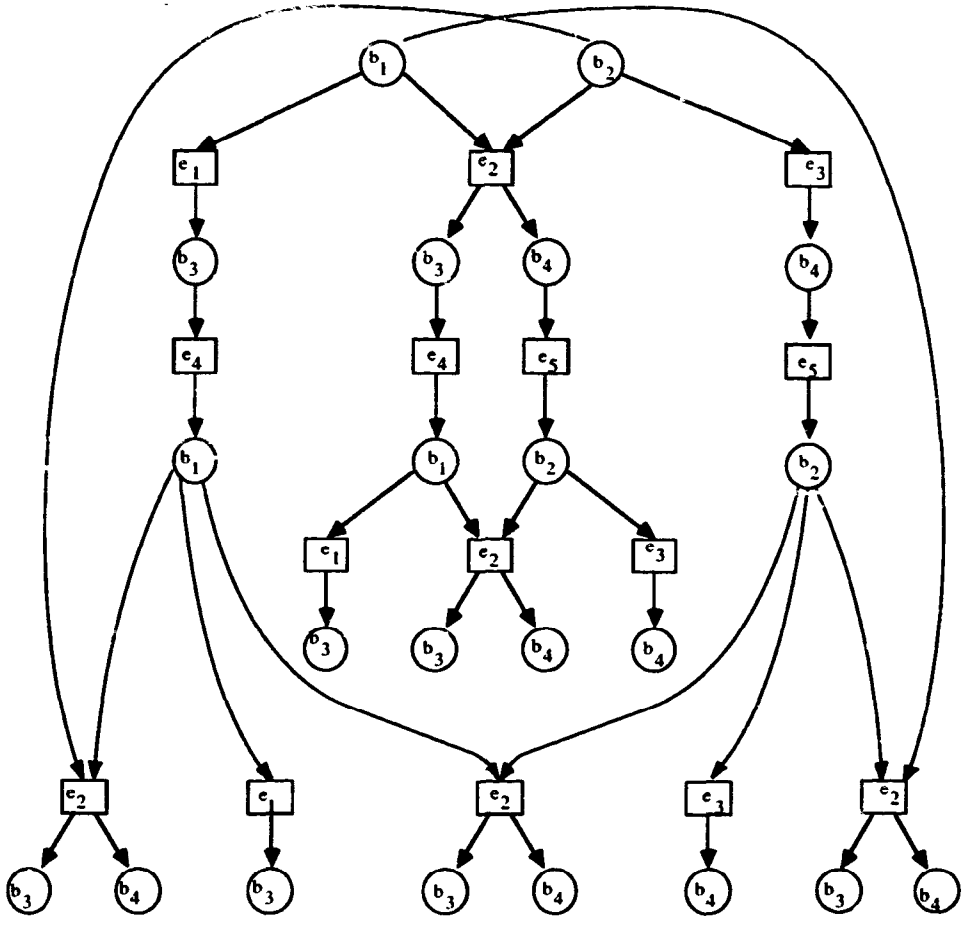

Fig. 12.

Proposition 5.2. *The unfolding of a net system is a labelled occurrence net.*

In an occurrence net $N = (B, E, F)$ causality is represented by the partial ordering relation $\leq_N = F^*$. The conflict relation $\#_N \subseteq X_N \times X_N$ is defined to be the least subset of $X_N \times X_N$ satisfying

(i) $\forall e_1, e_2 \in E.\ [e_1 \neq e_2 \wedge {}^\bullet e_1 \cap {}^\bullet e_2 \neq \emptyset \Rightarrow e_1 \#_N e_2]$;

(ii) $\forall x, y, z \in X_N.\ x \#_N y \leq_N z \Rightarrow x \#_N z$.

It is easy to check that $\#_N$ is irreflexive and symmetric. If two elements are in conflict then the idea is that in no stretch of behaviour can they both occur. On the other hand, for an element to occur, all the elements that lie in its "past" (as specified by $\leq_N$) must have occurred. These considerations will be made more precise when we come to deal with event structures. Going back to the occurrence net $N = (B, E, F)$ the concurrency relation co_N can now be defined as

$$\forall x, y \in X_N.\quad x\, co_N\, y \overset{\text{def}}{=} \mathbf{not}(x <_N y \vee y <_N x \vee x \#_N y).$$

Proposition 5.3. *Let N denote the underlying occurrence net of UF_0, the unfolding of the net system $\mathcal{N}_0$. Suppose that $x, y \in \hat{B}_0 \cup \hat{E}_0$. Then $x \#_N y$ iff there does* not *exist a process $N_\rho = (B_\rho, E_\rho, F_\rho)$ of $\mathcal{N}_0$ such that $x, y \in B_\rho \cup E_\rho$.*

Corresponding statements can be made about $\leq_N$ and co_N, the causality and concurrency relation respectively of the occurrence net underlying UF_0. In this sense UF_0 is a behavioural representation of $\mathcal{N}_0$ in which causality, concurrency and conflict are explicitly represented. We can now ask in what sense UF_0 and $(T_0, \sqsubseteq_0)$ are related to each other. To answer this question we must go over to labelled event structures.

In the present setting we note that a trace—via the labelled poset associated with it—can be seen as a more abstract representation of a process; it is a representation in which the conditions have been restricted away. Similarly an event structure is a more abstract representation of an occurrence net that is obtained by throwing away the conditions.

An *event structure* is a triple $ES = (E, \leq, \#)$ where

(i) E is a set of events.

(ii) $\leq \subseteq E \times E$ is a partial ordering relation called the *causality relation* of ES.

(iii) $\#$ is an irreflexive and symmetric relation called the *conflict relation* of ES.

(iv) $\#$ is "inherited" via $\leq$ in the sense that

$$\forall e_1, e_2, e_3 \in E. \quad e_1 \# e_2 \leq e_3 \Rightarrow e_1 \# e_3$$

Definition 5.4. $ES_0 = (\hat{E}_0, \leq_0, \#_0, \hat{\varphi}_0)$ is the *labelled event structure* of $\mathcal{N}_0$ given by (recall that $UF_0 = (\hat{B}_0, \hat{E}_0, \hat{F}_0)$)

(i) $\leq_0$ is $\leq_N$ restricted to $\hat{E}_0 \times \hat{E}_0$ where N is the underlying occurrence net of UF_0;

(ii) $\#_0$ is $\#_N$ restricted to $\hat{E}_0 \times \hat{E}_0$;

(iii) $\hat{\varphi}_0 : \hat{E}_0 \to E_0$ is the restriction of the labelling function of UF_0 to $\hat{E}_0$.

Proposition 5.5. *$(\hat{E}_0, \leq_0, \#_0)$ is an event structure.*

We now have yet another representation of the behaviour of $\mathcal{N}_0$ (apart from UF_0) in which causality, conflict and concurrency are explicitly represented. It turns out that $(T_0, \sqsubseteq_0)$ obtained via the theory of traces and ES_0 "agree" as to what the behaviour of $\mathcal{N}_0$ is. To bring this out we must represent ES_0 in terms of its states. The states of an event structure are called configurations.

Definition 5.6. Let $ES = (E, \leq, \#)$ be an event structure. Then $d \subseteq E$ is called a *configuration* iff it satisfies

(i) $(d \times d) \cap \# = \emptyset$ (conflict-free),

(ii) $d = \downarrow d$ (left-closed).

Theorem 5.7. *$(C_0^{\text{fin}}, \subseteq)$ and $(T_0, \sqsubseteq_0)$ are isomorphic posets. In fact, the map $g: T_0 \to C_0^{\text{fin}}$ given by $g([\rho]) = E_\rho$ (recall that $N_\rho = (B_\rho, E_\rho, F_\rho)$) is an isomorphism.*

Here C_0^{fin} is the set of *finite* configurations of ES_0. The alert but uninitiated reader might be puzzled by the fact that the agreement between ES_0 and $(T_0, \sqsubseteq_0)$ is stated in terms of an isomorphism between $(C_0^{\text{fin}}, \subseteq)$ and $(T_0, \sqsubseteq_0)$. We do so because it so happens that ES_0 and $(C_0^{\text{fin}}, \subseteq)$ are "equivalent" objects in a precise sense. This follows from the theory of event structures.

In [27] a basic representation theorem for event structures was established in terms of the posets of configurations. It turns out that for the event structure ES, the poset of configurations $(C_{ES}, \subseteq)$ is a *prime algebraic coherent* poset. Due to lack of space we will not go into details here. Moreover, given PO, a prime algebraic coherent poset, there is a *canonical* way of extracting an event structure ES from PO and it turns out that $(C_{ES}, \subseteq)$ and PO are isomorphic posets. Since a prime algebraic coherent poset is a special kind of an algebraic cpo, event structures can be identified with a restricted class of Scott domains.

If an event structure ES is *finitary* (i.e., $\downarrow e$ is finite for every e) then it turns out that the representation theorem cited above can be extended to establish a representation theorem linking ES and $(C_{ES}^{\text{fin}}, \subseteq)$ (see [26]). Clearly ES_0, the event structure associated with $\mathcal{N}_0$ is finitary. Hence we are justified in claiming—via Theorem 5.7—that trace theory and the theory of event structures agree as to what the behaviour of an elementary net system is.

Occurrence nets have not been investigated as objects of independent interest in the way that causal nets have been studied. Event structures on the other hand have a substantial theory. Winskel has constructed a major part of this theory [39] and has demonstrated how event structures can be used to provide the "non-interleaved" denotational semantics of CCS-like languages [40]. Actually, what we have called event structures here are called *prime* event structures in the literature. It turns out that in semantic applications it is more convenient to use a generalization of prime event structures called stable event structures. For details, the reader is once again referred to [40].

6. Summary

Our aim here has been to give a general picture of the behavioural aspects of net theory. We have done so by presenting a number of behavioural notions which, regardless of their origins, reflect the basic concerns of net theory.

A number of other behavioural tools have not been presented (see, for example, [33, 35]). The relationship between trace theory and event structures can be established in a general setting [34]. Studies which relate a variety of behavioural notions to each other in a categorical framework can be found in [41] and also in [2].

In CCS and CSP, which are two other well-known approaches to the study of distributed systems, one is concerned in some sense only with behaviours. In these

approaches a great deal of the theory is devoted to the search for a suitable notion of behavioural equivalence for identifying process terms. In net theory the study of the interplay between the *structure* of a distributed system (as specified by a net) and its behaviour has traditionally been one of the main concerns.

In the recent past however, a number of bridges have been constructed with net theory on the one side and CCS and CSP on the other [14, 28, 37]. As a result one may expect that in the future the pursuit of net theory will reflect the concerns of CCS-like approaches in a more direct fashion.

Acknowledgment

I thank Carl Adam Petri for creating net theory and for teaching me his version of the theory. The line of presentation followed here has been strongly influenced by my joint work with Mogens Nielsen and Grzegorz Rozenberg. Many many thanks to Karen Møller for producing, as usual, a nice manuscript in record time. This paper was written during a very pleasant stay at the Computer Science Department of Aarhus University.

References

[1] IJ. Aalbersberg and G. Rozenberg, Theory of traces, Tech. Report 16, Computer Science Department, Univ. of Leiden, The Netherlands, 1986.

[2] M. Bednarczyk, Categories of asynchronous systems, Ph.D. Thesis, Computer Science Department, Univ. of Sussex, Great Britain, 1987.

[3] E. Best, A theorem on the characteristics of non-sequential processes, *Fund. Inform.* **III** (1) (1980) 77-94.

[4] E. Best and R. Devillers, Sequential and concurrent behaviour in Petri net theory, *Theoret. Comput. Sci.* **55** (1987) 87-136.

[5] W. Brauer, W. Reisig and G. Rozenberg, eds., *Petri Nets: Applications and Relationships to Other Models of Concurrency*, Lecture Notes in Computer Science **255** (Springer, Berlin, 1987).

[6] C. Choffrut, Free partially commutative monoids, Tech. Report 86-20, LITP, University of Paris 7, France, 1986.

[7] F. Commoner, A.N. Holt, S. Even and A. Pnueli, Marked directed graphs, *J. Comput. System Sci.* **5** (1971) 511-523.

[8] P. Degano, R. DeNicola and U. Montanari, A new operational semantics for CCS based on condition/event systems, Nota Interna B4-42, Department of Computer Science, Univ. of Pisa, Italy, 1986.

[9] C. Fernandez and P.S. Thiagarajan, *D*-continuous causal nets: a model of non-sequential processes, *Theoret. Comput. Sci.* **28** (1984) 171-196.

[10] C. Fernandez, M. Nielsen and P.S. Thiagarajan, Notions of realizable non-sequential processes, *Fund. Inform.* **IX** (1986) 421-454.

[11] H.J. Genrich, Predicate/transition nets, in: Lecture Notes in Computer Science **254** (Springer, Berlin, 1987) 207-247.

[12] H.J. Genrich and K. Lautenbach, Synchronisationsgraphen, *Acta Inform.* **2** (1973) 143-161.

[13] H.J. Genrich and K. Lautenbach, System modelling with high-level Petri nets, *Theoret. Comput. Sci.* **13** (1981) 109-136.

[14] U. Goltz, On representing CCS programs by finite Petri nets, in: Lecture Notes in Computer Science **324** (Springer, Berlin, 1988) 339-350.

[15] M. Hack, Analysis of production schemata by Petri nets, M.S. Thesis, TR-94, Project MAC, Department of Electrical Engineering, Massachusetts Institute of Technology, Cambridge, MA, USA, 1972.

[16] C.A.R. Hoare, *Communicating Sequential Processes* (Prentice Hall, London, 1985).

[17] M. Jantzen, Complexity of place/transition nets, in: Lecture Notes in Computer Science **254** (Springer, Berlin, 1987) 413-435.

[18] M. Jantzen, Language theory of Petri nets, in: Lecture Notes in Computer Science **254** (Springer, Berlin, 1987) 397-412.

[19] K. Jensen, Coloured Petri nets, in: Lecture Notes in Computer Science **254** (Springer, Berlin, 1987) 248-299.

[20] J.R. Jump and P.S. Thiagarajan, On the equivalence of asynchronous control structures, *SIAM J. Comput.* **2**(2) (1973) 67-87.

[21] R.M. Karp and R.E. Miller, Parallel program schemata, *J. Comput. System Sci.* **3**(2) (1969) 147-195.

[22] K. Lautenbach, Linear algebraic techniques for place/transition nets, in: Lecture Notes in Computer Science **254** (Springer, Berlin, 1987) 142-167.

[23] A. Mazurkiewicz, Concurrent program schemes and their interpretations, DAIMI Report PB-78, Computer Science Department, Aarhus Univ., Denmark, 1977.

[24] A. Mazurkiewicz, Semantics of concurrent systems: a modular fixed-point trace approach, in: Lecture Notes in Computer Science **188** (Springer, Berlin, 1985) 353-375.

[25] R. Milner, *A Calculus of Communicating Systems*, Lecture Notes in Computer Science **92** (Springer, Berlin, 1980).

[26] M. Nielsen, G. Rozenberg and P.S. Thiagarajan, Behavioural notions for elementary net systems, Internal Report, Computer Science Department, Aarhus Univ., Denmark.

[27] M. Nielsen, G. Plotkin and G. Winskel, Petri nets, event structures and domains: Part 1, *Theoret. Comput. Sci.* **13** (1980) 85-108.

[28] E.R. Olderog, Operational Petri net semantics for CCSP, in: Lecture Notes in Computer Science **266** (Springer, Berlin, 1987) 196-223.

[29] C.A. Petri, Kommunikation mit Automaten, Schrifften des IIM Nr. 2, Institut für Instrumentelle Mathematik, Bonn Univ., Fed. Rep. Germany, 1962.

[30] C.A. Petri, Non-sequential processes, Interner Bericht ISF-77-5, Gesellschaft für Mathematik und Datenverarbeitung, St. Augustin, Fed. Rep. Germany, 1977.

[31] C.A. Petri, Concurrency theory, in: Lecture Notes in Computer Science **254** (Springer, Berlin, 1987) 4-24.

[32] V.R. Pratt, Modelling concurrency with partial orders, *Internat. J. Parallel Programming* **15**(1) (1986) 33-71.

[33] G. Rozenberg and P.S. Thiagarajan, Petri nets: basic notions, structure and behaviour, in: Lecture Notes in Computer Science **224** (Springer, Berlin, 1986) 585-668.

[34] B. Rozoy and P.S. Thiagarajan, Event structures and trace monoids, Report 87-47, LITP, Univ. of Paris 7, France, 1987.

[35] P.H. Starke, Traces and semiwords, in: Lecture Notes in Computer Science **208** (Springer, Berlin, 1985) 332-349.

[36] P.S. Thiagarajan, Elementary net systems, in: Lecture Notes in Computer Science **254** (Springer, Berlin, 1987) 26-59.

[37] D. Taubner, The finite representation of abstract programs by automata and Petri nets, Ph.D. Dissertation, Technical Univ. of Munich, Fed. Rep. Germany, 1989.

[38] P.S. Thiagarajan and K. Voss, A fresh look at free choice nets, *Inform. and Control* **61**(2) (1984) 85-113.

[39] G. Winskel, Event structures, in: Lecture Notes in Computer Science **255** (Springer, Berlin, 1987) 325-392.

[40] G. Winskel, *Event structure semantics of CCS and related languages*, Lecture Notes in Computer Science **140** (Springer, Berlin, 1982).

[41] G. Winskel, Categories of models for concurrency, Tech. Report No. 58, Computer Laboratory, Cambridge University, U.K., 1986.