Fundamenta Informaticae XI (1988) 357-404
North-Holland

BEHAVIOR STRUCTURES AND NETS

A. RABINOVICH and B.A. TRAKHTENBROT

School of Mathematical Sciences
Raymond and Beverly Sackler Faculty of Exact Sciences
Tel-Aviv University, Tel-Aviv, Israel 69978

ABSTRACT

Behavior Structures integrate causality and branching. Nets of Behavior Structures provide a unifying approach to different net models of Concurrency. The Theory is illustrated wrt Nets over automata in particular wrt Petri Nets.

0. Introduction

In the theory of Concurrency there is a proliferation of models for describing concurrent processes. An evident source of this phenomenon lies in the controversies:

Interleaving Semantics vs Partial Order (causality)
Linear Time vs Branching Time

Yet another source of diversity is connected with the way a complex system is assembled from elementary blocks. In Algebras of Processes the assembling proceeds in an explicit compositional way through the chosen repertoire of operations upon processes. At the other hand in Networks concurrently executing components are assumed to communicate through wired channels. In these models (like Petri Nets, Data Flow Networks), no compositionality seems to be assumed explicitly. Indeed, for a long time in the theory of Petri Nets the question of modularity did not even arise. In its simplest form this question deals with a subnet N' of a given net N. Assume that N' is replaced in N by a net N'' which behaves like N'; is it the case that the new net produced as result of the replacing behaves like the original net N? Note originally [Milner, Hoare] Algebras (and compositionality) were oriented on interleaving models, whereas Nets were deemed to be more appropriate for the revealing of causality.

Our main concern in this paper is about Nets of Processes, which are expected to bridge between these original views on algebras and networks. The underlying idea may be traced to Mazurkiewicz [Maz84] and Pratt [Pr]; it acknowledges that synchronization of processes is a sufficient tool for composing complex nets from appropriate "blocks". The construct "Net of Processes" may be parametrized wrt a favorite model of processes. Mazurkiewicz and Pratt considered Nets of Pomset Processes, whereas in this paper we deal with Nets of Behavior Structures. Unlike Pomset Processes, which reflect causality but ignore branching, Behavior Structures (BS) integrate causality with branching. Actually, Behavior Structures are one more version in the series of models, beginning with Event Structures [NPW], which includes Configuration Structures [NPW], Prefix Structures [Maz84], and Behavioral Systems [Shi]. However we find Behavior Structures more appropriate for our purpose and we say about this more in this sequel.

Through Nets of Behavior Structures we aim at a unifying approach to different Net models of Concurrency. In this way we hope to explain phenomena which may be blurred by more common ad hoc approaches. We illustrate the situation for Nets N over automata (multiautomata), which cover as particular cases both Petri Nets and Data Flow Networks. Usually, there is a strong intuition (and a general consensus) that N behaves globally as an *automaton (multiautomaton);* yet the inherent causal aspects of this interleaving - branching behavior are still to be discerned. For example, in a Petri Net the multiautomaton behavior is convincingly exhibited by the token game. But starting with Petri's seminal work much effort went (and is still going on) into defining the "genuine" causal behavior of such nets. We believe that our results from Appendix explain why for the case of Petri's original model (C/E Nets), all the known and apparently different approaches result in essentially the same causal semantics. This is in full contrast with the generalized models (P/T Nets) for which different treatments of causality are possible and it is up to personal responsibility to choose the appropriate one.

In this sequel our exposition is organized as follows:

Section 1 is dedicated to the Behavior Structure (BS) machinery, including as a conceptual contribution carefully elaborated definitions of BS, BS-bisimulation, embedding into BS, and synchronization of BS. The main technical result about synchronization is that it is fully compositional and also inherits bisimulation and embeddings.

Section 2. The notions and results of Section 1 may be used to formulate and investigate in a very general setting Nets of Processes. More specifically we concentrate on Nets over Automata (Multiautomata) and introduce the crucial notion of robust BS-semantics for such nets. Then we show that robust BS-semantics actually coincides with or refines semantics provided by other existing non compositional approaches (Occurrence Semantics, Traces, etc.). We clarify also conditions under which the robust BS-semantics may be recovered from partial information.

Finally, the Appendix illustrates the theory wrt Petri Nets. Here we follow and improve in some respects the modular approach Mazurkiewicz elaborated for C/E Petri Nets. Relying on the techniques of robust BS-semantics we characterize the deep difference between Petri's original model (C/E nets) and its generalizations known as P/T nets. At the same time we claim that the modular approach is consistent with earlier established operational semantics. This material is presented without proofs. We use also Nets of Processes in the investigation of Data Flow Networks. These results will appear in [RT].

Our work was strongly influenced by Mazurkiewicz's approach [Maz84], where one can already find the main ideas about compositional semantics for nets. Mazurkiewicz's elegant theory covers fully the case of finite C/E Nets. Roughly speaking our aim was to examine how far the methodology can be advanced into the realm of more general nets.

As usually for texts in semantics there is the painful dilemma between the need to be very careful and the desire to keep the paper at an acceptable size and level of intelligibility. We try to avoid slipping down into cumbersome details; in particular, often we provide only sketches of proofs, counting on the collaboration of the reader.

1. Processes

Preliminary Remarks

Figure 1 schematically represents the "refining" order between six models of processes, which differ from each other by the way they do or do not reflect branching and causality.

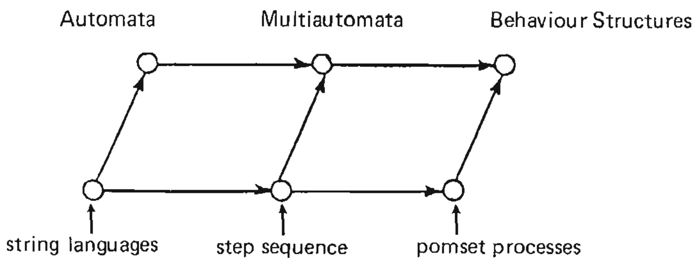


Fig. 1

The lower level contains the linear time models whereas the upper level contains the corresponding branching versions. Behavior Structures (defined in 1.3) are the most discriminating model we know and the main object of interest in this section (1.3 - 1.6). We include also in 1.1 - 1.2 a rather cursory survey of other models. In each model π one distinguishes in a standard way the *concrete* and *abstract* approaches to the objects under consideration. For concrete objects one defines the most discriminating equivalence relation, called *strict isomorphism* and designated as $\equiv_{isom}$; one should always have in mind that strict isomorphisms differ from (standard) isomorphism through the additional requirement that the compared objects have the same declared alphabet. Equivalence classes wrt $\equiv_{isom}$ are *abstract* objects, each element P of the class being a representative of the abstract object [P]. For each of the three branching models, of great importance is a weaker equivalence than $\equiv_{isom}$ called bisimulation: $\equiv_M$ is the Milner-Park bisimulation for automata, $\equiv_{MM}$, $\equiv_{BS}$ the respective bisimulations for Multiautomata and Behavior Structures. When referring to the basic equivalence $\equiv\pi$ in the model π we have in mind strict isomorphism in the case of linear models and bisimulation in the case of branching models. Finally, for each pair of comparable models $\pi<\rho$ there is also an embedding relation "process P_1 from model π is embeddable into process P_2 from model ρ". Through appropriate combination of embeddings and basic equivalences many other equivalences may be defined.

We can now characterize the format of two tasks (parametrized wrt different models) we consider in this section.

Task 1. For a given model compare different equivalences between processes in this model.

Task 2. For two comparable models π, ρ prove that each process in π is embeddable in some process in ρ. Find out when the embedding is unique (up to some favorite equivalence).

The next task has to do with Synchronization, which is the crucial operation on processes. Synchronization is also parametrized with respect to the model π; the most relevant for us is synchronization of BS (Section 1.6).

Let R be one of the relations considered above: it may be an equivalence in some model π, or an embedding from π into ρ. We say that synchronization (in model π) inherits the relation R if:

$$P_1RP'_1,\ P_2RP'_2,\ldots$$

implies

$$Synch(P_1,\ P_2,\ldots)RSynch(P'_1,\ P'_2,\ldots)$$

Task 3. For a given relation prove that it is inherited by synchronization.

In all models Synchronization has the following basic properties:

(i) Synchronization inherits isomorphism. Hence, synchronization of abstract objects is correctly defined through arbitrary representatives.

(ii) Synchronization is **fully compositional.** In more detail:

a) Up to isomorphism the result of $Synch(P_1,\ P_2,\ldots)$ does not depend on the order of the operands.

b) Given any disjoint partition of the set $\{P_i\}$ into sequences $seq_1, seq_2, \cdots$,

$$Synch(P_1,P_2,\ldots)\equiv_{isom}Synch(Synch(seq_1),Synch(seq_2),\ldots)$$

Our main results for Behavior Structures are in 1.5 (Task 1), 1.4 (Task 2) and 1.6 (Task 3).

We want to finish the short survey of this section with some explanation why we prefer to deal with Behavior Structure despite the already existing notions of Event Structures and Configuration Structures, which essentially capture the same entity. (Unfortunately we were not aware about Events Structures, when we started this work. We are thankful to R. Milner and U. Montanari who called our attention to this omission.) It seems enough clear that ultimately the objects to be considered in Process Theory are Configuration Structures (CS) or Behavior Structures (BS), whereas Event Structures (ES) are only a compact way to represent CS or BS. The comparison with matrices as a tool to represent linear functionals suggests itself. And indeed it is difficult to reflect the crucial equivalence, namely BS-bisimulation, in terms of ES. Actually the only difference between labeled CS and BS is that in an LCS all configurations refer to a global set of events, whereas in a BS each configuration uses its local set of events; that is why a BS should be equipped with a family of appropriate embeddings between different configurations. Though BS's form a broader class of concrete objects than LCS's it is the case that every BS is strictly isomorphic to an LCS. Hence, when synchronizing abstract BS's one could manage with LCS-representatives. But we prefer to deal directly with Concrete BS's to avoid transformations to LCS's.

1.1. Linear Time Processes

We consider in this section the three linear models mentioned above (Fig. 1). We deal mainly with the most discriminating Pomset model and give only some hints about the other two simpler models. In doing so we heavily rely on the careful exposition of the subject in [Maz88]. The only significant deviation is in the definition of Synchronization, where we prefer a broader treatment of *Concrete Pomsets* (Labeled Posets - in [Maz88]) as representatives of *Abstract Pomsets* (Qualified Pomsets - in [Maz88]). This deviation is not necessary as far as the Pomset model is considered; its usefulness will become evident later wrt the more discriminating model of Behavior Structures (1.3-1.6).

Pomsets (Partial ordered multisets)

A Concrete Pomset (or a Labeled Poset) P is a quadruple

$$<V, \leq, l, \Sigma>$$

where V is a set (of events), $\leq$ is a partial order on V, Σ is an alphabet (of actions), and l is a labeling function: $V \rightarrow \Sigma$.

In this sequel we consider only **finite** concrete pomsets, i.e. V is finite or empty. But note that the alphabet Σ is never empty and may be even infinite; it is not required that each action from Σ should appear as a label of an event in V. The relevance of the alphabet and its impact on synchronization will be illustrated later.

For a Concrete Pomset P its components are parametrized as V_P, $\leq_P$, l_P, alph(P). P is said to be *autoconcurrent* if for some incomparable events v_1, v_2 in V_P it is the case that $l_P(v_1)=l_P(v_2)$. Otherwise P is without autoconcurrency (or according to the terminology in [Maz88], P has the self-dependence property).

Assume that U is an arbitrary downward closed subset of V_P, i.e., $u\leq_P v\in U$ implies $u\in U$. Then, the concrete Pomset PU with $V_{PU}=U$ and the other components inherited from P is said to be a prefix of P. We write Q≤P for "Q is a prefix of P".

Concrete Pomsets P, R are *isomorphic* iff there exists a bijection $f: V_P\rightarrow V_R$ which respects ordering and labeling. P, R are said to be *strictly isomorphic* (notation $P\equiv_{Pom}R$) iff they are isomorphic and alph(P)=alph(R). Clearly, $\equiv_{Pom}$ is an equivalence relation; each $\equiv_{Pom}$ equivalence class is said to be an *Abstract Pomset,* whose representatives are the concrete Pomsets belonging to this class. [P] is the notation for the Abstract Pomset represented by the Concrete Pomset P. For Abstract Pomsets the property of being autoconcurrent is correctly defined through arbitrary representatives. For Abstract Pomsets P and Q we say that P is a prefix of Q if there exists a representative of P which is a prefix of some representative of Q. These definitions are easily seen to be independent of the choice of representatives.

Assume that $[P]\leq[Q]$. If Q is without autoconcurrency then it has a **unique** prefix R such that $P\equiv_{Pom}R$. However if Q is autoconcurrent uniqueness cannot be guaranteed. This situation is not relevant for the Pomset Model Theory as developed in [Maz88], but must be taken into account when synchronization for more elaborate models (say for Behavior Structures) is considered.

Synchronization

Before we proceed to the definition let us emphasize two points. First, we consider Synchronization as an operation upon finite or infinite sequence of operands. In [Maz88] the infinite case is not mentioned explicitly though it emerges naturally from the context. Second, we start with Synchronization of Concrete Pomsets which unlike [Maz88] produces as a result a well-defined set of Concrete Pomsets. (In [Maz88], Synchronization is defined directly on Abstract Pomsets.)

Definition. A sequence Emb

$$\text{Emb}=V,\ V_1,\ f_1,\ V_2,\ f_2,\ldots \quad (*)$$

is an embedding (of the sets V_i into the set V via the embedding functions f_i) if:

a) For each i, f_i is an injection of V_i into V

b) $V=\bigcup_i f_i(V_i)$

We describe a construction, called concretization, which transforms an arbitrary embedding (*) into an isomorphic embedding:

$$\text{ConcretEmb}=V^c,\ V_1{}^c,\ f_1{}^c,\ V_2{}^c,\ f_2{}^c,\dots$$

(i) V^c consists of the following "concrete" codes for the elements of V:

$code(v)=_{def} v_1, v_2, v_3,\dots$ where

$$\begin{cases} v_i=nil & \text{if} \quad v\notin f_i(V_i) \\ v_i\in V_i & \text{and} \quad f_i(v_i)=v, \quad \textit{otherwise} \end{cases}$$

(ii) $f_i{}^c(v_i)=_{def} code(f_i(v_i))$

Finally, an embedding Emb is said to be *concrete* if Emb=ConcretEmb.

We are going to define the notion of Concrete Synchronized Embedding; this is a sequence

$$P, P_1, f_1, P_2, f_2, \dots \quad (**)$$

where P and P_i are finite concrete pomsets $<V, \leq, l, \Sigma>$, $<V_i, \leq_i, l_i, \Sigma_i>$. and the f_i are embedding functions which fulfill the following conditions:

a) The sequence V, V_1, f_1, V_2, $f_2,\dots$ is a concrete embedding

b) The f_i preserve labels: $l_i(x)=l(f_i(x))$

c) The partial order $\leq$ is inherited from the partial orders $\leq_i$, i.e., $p\leq q$ iff for some sequence $p_1, \dots, p_n$ of elements of V,

$p=p_1$, $p_n=q$ and for each j<n there is k such that $f_k{}^{-1}(p_j)\leq_k f_k{}^{-1}(p_{j+1})$

d) $\Sigma=\bigcup_i \Sigma_i$

Now, with a sequence of concrete pomsets $P_1, P_2,\dots$ associate the set (which may be empty) of concrete pomsets Synch($P_1,P_2,\dots$), where $P\in$Synch($P_1,P_2,\dots$), if for some appropriate functions $f_1, f_2,\dots$ $<P, P_1, f_1, P_2, f_2,\dots>$ is a concrete synchronized embedding. The sequence $P_1, P_2,\dots$ is said to be synchronizable if Synch($P_1,P_2,\dots$) is not empty.

Remarks

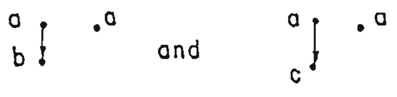
(i) Let us illustrate the impact of the declared alphabet on synchronization. A concrete pomset P is said to be empty if $V_P=\emptyset$. Empty concrete pomsets may still behave differently depending on their alphabets. To make this point clear consider two concrete pomsets P_1, P_2 with alphabets Σ_1, Σ_2, where P_1 is empty. Assume that P_2 is also empty. Then P_1, P_2 are synchronizable and produce a unique concrete pomset which is also empty; its alphabet is $\Sigma_1\bigcup \Sigma_2$. On the other hand assuming that P_2 is not empty, there are two possibilities:

a) In P_2 there is an event with label from Σ_1. Then P_1, P_2 are not synchronizable.

b) Otherwise P_1, P_2 are synchronizable and produce a unique pomset; it differs from P_2 only through its alphabet which is $\Sigma_1\bigcup \Sigma_2$.

(ii) If we restrict ourselves with concrete pomsets P_1 and P_2 without autoconcurrency, then if they are synchronizable there may be only one concrete pomset in their synchronization. However in the general case many pomsets are possible; hence the synchronization of two pomsets may result in a set of pomsets. Synchronization becomes nondeterministic.

Example: Synchronizing the pomsets



we obtain two pomsets:

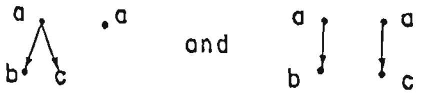


Fig. 2

(iii) It is easy to check that given a Concrete Synchronized Embedding (**) and an arbitrary prefix P' of P there exists a Concrete Synchronized Embedding

$$P',\ P'_1,\ f'_1,\ P'_2,\ f'_2 \tag{***}$$

such that for all i:

a) $P'_i \leq P_i$

b) f'_i is the restriction of f_i to P'_i

For any alphabet Σ each set of Concrete Pomsets over the alphabet Σ is a Concrete Pomset Language over Σ. The Concrete Pomset Languages L_1, L_2 are strictly isomorphic (notation $L_1 \equiv_{Pom} L_2$) iff for each concrete pomset in one of them there is a strictly isomorphic concrete pomset in the other. Clearly $\equiv_{Pom}$ is an equivalence relation among Concrete Pomset Languages; each equivalence class is said to be an Abstract Pomset Language, whose representatives are the Concrete Pomset Languages. [L] is the notation for the Abstract Pomset Languages represented by the Concrete Pomset Language L. An Abstract Pomset *Language* is said to be an Abstract Pomset *Process* if it is prefix closed. Correspondingly a Concrete Pomset Language L is said to be a Concrete Pomset Process iff [L] is an Abstract Pomset Process. These notions of processes work quite well in the Pomset Model. However it is easy to see that they do not capture a peculiarity which may happen to be relevant in more discriminating models. Namely, assume that the concrete pomset $P \in L$ has two prefixes R, Q such that $R \equiv_{Pom} Q$. We might be interested in a version of prefix closure, which assures that both R and Q have their own counterpart in L. Considerations of this sort become relevant in the Behavior Structures Model.

Finally, given a sequence of Concrete Pomset Languages $\Pi_1, \Pi_2, \ldots$

$$Synch(\Pi_1, \Pi_2, \ldots) =_{def} \Pi =_{def} \{P : P \text{ is finite and } P \in Synch(P_1, P_2, \ldots) \text{ where } P_i \in \Pi_i\}$$
$$\text{with } \mathrm{alph}(\Pi) = \bigcup \mathrm{alph}(\Pi_i)$$

Important properties of synchronization are formulated and fully justified in [Maz88] wrt abstract pomset languages. Let us emphasize the following facts which generalize some of these properties for infinite sequences of Concrete Pomset Languages.

Claim 1. Synchronization of Concrete Pomset Languages has the following properties:

(i) It inherits the strict isomorphism relation $\equiv_{Pom}$.

(ii) It inherits the property of being a process (i.e. of being a prefix closed language).

(iii) It is fully compositional.

In particular it follows that synchronization of Abstract Pomset Languages and Processes is correctly defined via synchronization of Concrete Pomset Languages which represent them. Note also, that the synchronization of Processes (unlike the synchronization of languages in general) is never empty because it contains at least the empty Pomset.

String Languages (Processes) and Step-Sequence Languages (Processes).

Let us briefly consider the other two linear time models (see Fig. 1) To this end in addition to the Pomset version one has to deal with two more special versions which arise when the underlying posets are in fact *tosets* (total ordered sets) or *stosets* (step oriented set). A *stoset* P may be characterized as follows: P is the disjoint sum of $P_1, \ldots, P_n$ (called steps) such that in each step P_i the elements are pairwise incomparable, and for $j<i$ there holds the implication: $x \in P_i$ and $y \in P_j$ imply $y<x$. Accordingly, we have two more series of notions:

(i) *Concerning Tosets:* Concrete Tomset Languages (and Processes), Abstract Tomset Languages (these are essentially string languages), Abstract Tomset Processes.

(ii) *Concerning Stosets:* Concrete Stomset Languages (and Processes), Abstract Stomset Languages (these are step-sequence languages), Abstract Stomset Processes.

Hence there are three variants of linear time processes we have to deal with. For each of them the basic equivalence between languages (respectively $\equiv_{pom}$, $\equiv_{tom}$, $\equiv_{stom}$) is nothing but the strict isomorphism as defined above. However for synchronization in the Toset and Stoset models things look different. The point is that when applying the general definition of synchronization for pomset processes to tomset processes, the result will not necessarily be a tomset process. The same remark holds for stomset processes. The direct definition for tomset processes is well known and quite simple; for stomset processes the direct definition is a bit more complicated. Another (indirect) way to get the same result is to apply the general definition for pomset processes and then to *coarsen* the resulting pomset process P, namely:

a) Tomset process: take all the linearizations of pomsets in P.

b)Stomset process: take all the step-linearizations of pomsets in P.

In this sequel we deal mainly with concrete pomsets, so we omit the adjective "concrete".

1.2. Branching Time Processes: Automata and Multiautomata

A **Milner Process** over the action alphabet Σ is a rooted transition diagram whose edges are labeled by actions from Σ. In a Milner process a path from the root to a node represents a sequential scenario - the string of actions labeling the edges leading to the node. Thus the notion of a Milner process is a refinement of the notion of a Tomset Process, since it represents a set of strings as well as a description of their branching behavior. We use "Automaton" as synonym for "Milner Process"; at this stage we consider automata as a particular case of multiautomata when only single transitions are allowed.

A **Concrete Multiautomaton** M over Σ is given by:

i) A set Q of states.

ii) An initial state $q_0 \in Q$.

iii) Multitransitions of the form $q \xrightarrow{A} q'$, where A is a finite multiset over Σ.

A multiautomaton must satisfy the requirement:

iv) If $q \xrightarrow{A} q'$ and if $B \subset A$ is a partial multiset then there is some q'' in Q such that $q \xrightarrow{B} q'' \xrightarrow{A-B} q'$.

In a multiautomaton a path from the root generates a stomset. Thus multiautomata are related to stomset processes as automata are related to tomset processes.

A multiautomaton is called **deterministic** if for any state q and any multiset A there is at most one q' such that $q \xrightarrow{A} q'$

The notion of multiautomaton *without autoconcurrency* is defined in the same way as the notion of *multiautomaton* except that in iii) "multiset" is replaced by "set".

Isomorphisms of automata (multiautomata) are defined in an obvious way. What we are interested in are *strict isomorphisms* which assume that the automata (multiautomata) under consideration are isomorphic and have the same (declared) action-alphabet.

A relation R between the states Q of a multiautomaton M and the states Q' of a multiautomaton M' is an *MM - bisimulation* between M and M' if:

i) $q_0 R q_0'$

ii) If qRq' and $q \xrightarrow{A} p$ then for some p' in Q' there hold : $q' \xrightarrow{A} p'$ and pRp'.

iii) If qRq' and $q' \xrightarrow{A} p'$ then for some p in Q there hold: $q \xrightarrow{A} p$ and pRp'.

Multiautomata include automata as a special case when all the multitransitions are simple -- i.e. labeled by a singleton multiset. In this sense the well known notion of bisimulation for automata [Mil] is included as a particular case of bisimulation for multiautomata. Multiautomata (in particular - automata) M and M' are MM-bisimular - notation $M \equiv_{MM} M'$ - if there exists an MM-bisimulation between them.

The notion of *M -bisimulation* is defined by restricting a multiset A to a single action in the definition of MM - bisimulation. It is clear that $\equiv_{MM}$ equivalence implies $\equiv_M$.

Synchronization of Automata and Multiautomata

Again we consider a sequence (perhaps infinite)

$$A_1, A_2, \dots \quad (*)$$

If A_i are concrete automata then there is a simple way to define an automaton $A = Synch(A_1, A_2, \dots)$ with $\text{alph}(A) = \bigcup \text{alph}(A_i)$. This is done taking the Cartesian product of their state sets with the following transition rules:

$<q_1, q_2, \dots, q_k, \dots> \xrightarrow{a} <q'_1, q'_2, \dots, q'_k, \dots>$ iff $q_k \xrightarrow{a} q'_k$ in each A_k whose alphabet contains a. Here we assume that for each k q_k is a state in A_k and that the "global" initial state is the "vector" of the component initial states.

If the A_i are multiautomata, the alphabet and the states of $Synch(A_1, A_2, \dots)$ are defined as above and the multitransition rules are as follows:

For a multiset S of actions let S_i be the submultiset of S which contains all those actions which are in $\text{alph}(A_i)$. Then the transition

$<q_1, q_2, \dots, q_k, \dots> \xrightarrow{S} <q'_1, q'_2, \dots, q'_k, \dots>$ is possible iff for each i with $S_i \neq \emptyset$ the transition $q_i \xrightarrow{S_i} q'_i$ is in A_i.

Remarks.

(i) Note that $Synch_M$ and $Synch_{MM}$ have different meaning when applied to a sequence of automata

$$A_1, A_2, \dots$$

Using programming language jargon one would say that $Synch_M$ operates on genuine automata, whereas $Synch_{MM}$ operates on their conversions to multiautomata.

(ii) Formally we did not impose any restriction on neither the cardinality of the set of states of an automaton (multiautomaton) nor the branching degree (the cardinality of arrows which exit from a state). Note, however, that up to bisimularity one can consider only those states which are reachable from the initial state. Therefore if the branching degree is finite (or even countable) one can deal with a countable set of states. However, when synchronizing an infinite sequence of automata (multiautomata) there may appear a uncountable set of reachable states even when for each of the components the set of states is countable and the degree is finite. A simple way to avoid non-desirable high cardinalities for the set of states and for the branching degree amounts to the following restriction:

Say that the sequence of processes

$$A_1, A_2, \ldots \quad (*)$$

has *finite degree* iff for each action a in alph$=\bigcup$ alph(A_i) there is only a finite number of processes whose alphabet contains a.

It is an easy exercise to show that if (*) has *finite degree* and the A_i are countable and with *finite or countable branching degree* then $Synch(A_1, A_2, \ldots)$ is bisimular to an automaton(multiautomaton) with a countable set of states and with no more than countable branching degree.

Claim 2 . Synchronization of Automata (Multiautomata) is fully compositional; it inherits strict isomorphism and M-bisimulation (MM-bisimulation).

1.3. Configuration Structures, Behavior Structures, Event Structures

We are going to define the general notion of Concrete Behavior Structure (BS). First we consider the particular case of BS-without autoconcurrency, and then we give the general definition.

Below we use the following notations and terminology.

$P \leq Q$ - P is a prefix of Q.

$P \leq^A Q$ - $P \leq Q$ and A is the multiset of labels in the "suffix" $P - Q$.

$P << Q$ - P is an immediate prefix of Q, i.e. $P - Q$ consists of one labeled event.

$P <<^a Q$ - $P << Q$ and a is the label in $P - Q$.

In all cases above when $\leq$ (respectively $<<$) is replaced by $\leq_{isom}$ (respectively $<<_{isom}$) the intent is that P is *isomorphic* to a prefix (to an immediate prefix) in Q.

We have to consider also a special class of automata (call them *standard* automata) which slightly generalize the notion of a tree-like automata, namely:

a) All states (nodes) are reachable from the initial state (root).

b) The diagram contains no oriented cycles.

c) The diagram contains no parallel edges.

Behavior Structures without autoconcurrency

A BS B without autoconcurrency over the alphabet Σ is given by:

1) A standard automaton over Σ, designated by M(B) and called the Milnering of B.

2) A labeling of each node n in M(B) by a concrete pomset $\overline{n}$ (designated in this sequel as $\overline{n}$ or P_n) without autoconcurrency which obeys the requirements:

   (i) $n \overset{a}{\rightarrow} m$ in M(B) implies $\overline{n} <<^a{}_{isom} \overline{m}$.

   (ii) If Q is an immediate prefix of $\overline{m}$, then there is a unique n such that $Q \equiv_{isom} \overline{n}$ and $n \overset{a}{\rightarrow} m$ for some n. (This implies that the root (the initial state) is labeled by the empty pomset.)

From the definition it follows that a BS without autoconcurrency has the following important property.

The *Unique embedding* property. For any $n << m$ there is an unique isomorphism between $\overline{n}$ and an immediate prefix of $\overline{m}$. Moreover (due to the lack of autoconcurrency) for each $n < m$ there is induced an unique "embedding" $\phi_{n,m}: \overline{n} \rightarrow \overline{m}$ of $\overline{n}$ onto a prefix of $\overline{m}$. For these embeddings the following two nice properties hold. First, for all $m < n < k$ the diagrams (Fig. 3a) commute. Second, assuming that $\phi_{n,k}(\overline{n}) \subset \phi_{m,k}(\overline{m})$, then $n < m$ and therefore there is an embedding $\phi_{n,m}$.

Relying on the family of embeddings $\phi_{m,n}$ one can associate in a natural way with each BS without autoconcurrency also a multiautomaton MM(B) which we call the Multimilnering of B. Namely:

(i) The nodes of MM(B) are the same as the nodes of M(B).

(ii) Consider nodes $n < m$; since $\phi_{n,m}(\overline{n})$ is a prefix in $\overline{m}$ for some multiset A, then $\phi_{n,m}(\overline{n}) \leq^A \overline{m}$. (Recall: A is the multiset of labels which appear in $\overline{m} - \phi_{n,m}(\overline{n})$.) A transition $n \overset{A}{\rightarrow} m$ appears in MM(B) iff all the events in $\overline{m} - \phi_{n,m}(\overline{n})$ are maximal in $\overline{m}$ and hence incomparable with each other. (Intuitively, this means that the actions from A may be performed simultaneously.)

We come to the general notion of a BS by removing the restriction that the pomsets are not autoconcurrent. But then the unique embedding property would not be valid any more, because if $\overline{m}$ is an autoconcurrent pomset there may be many embeddings of $\overline{n}$ onto immediate prefixes in $\overline{m}$. Hence, we cannot refer to the family of embedding functions which are implicit in the case without autoconcurrency and which have the nice properties mentioned above. The remedy is to require them by definition.

Definition. A Behavior Structure B is given by:

a) A standard automaton M(B) - the Milnering of B.

(b) A labeling that assigns to each node n a pomset P_n (sometimes designated also as $\overline{n}$) over Σ. P_0 is empty (sometimes denoted by ϵ).

(c) A set of embeddings $\phi_{n,m} : P_n \rightarrow P_m$ for each pair $n \leq m$ such that

(1) $\phi_{n,m}$ is a isomorphic injection of P_n onto a prefix of P_m. ($\phi_{n,n}$ is the identity).

(2) Every prefix Q of P_n is obtained as $\phi_{k,n}(P_k)$ for some $k \leq n$.

(3) All diagrams on Fig. 3a commute for m<n<k.

$$\begin{tikzcd}
P_m \arrow[r, "f_m"] \arrow[d, "\phi_{m,m_1}"'] & P'_{\sigma m} \arrow[d, "\phi_{\sigma m,\sigma m_1}"] \\
{} \arrow[r, "f_{m_1}"'] & P'_{\sigma m_1}
\end{tikzcd}$$

a)

$$\begin{tikzcd}
P_m \arrow[r, "\phi_{m,n}"] \arrow[dr, "\phi_{m,k}"'] & P_n \arrow[d, "\phi_{n,k}"] \\
& P_k
\end{tikzcd}$$

b)

Fig. 3

(4) If $\phi_{n,k}(P_n) \subset \phi_{m,k}(P_m)$ then $n \leq m$.

The Multimilnering is defined as before (for BS without autoconcurrency).

We say that two Behavior Structures B and B' are *isomorphic* if there exists the following family of isomorphisms:

a) An isomorphism σ between the diagrams of B and B' (i.e. essentially isomorphism of automata).

b) For each node $m \in B$ an isomorphism f_m between the concrete pomsets P_m and $P'_{\sigma(m)}$. Moreover the commutation of the diagram 3b is required.

Two isomorphic behavior structures are strictly isomorphic iff they have the same action-alphabet. Note that the concrete pomsets which occur in a Behavior Structures B inherit the alphabet of B.

Important warning: In order to deal appropriately with equivalences among BS and with operations on BS (especially with synchronization) we need to point explicitly on the respective alphabets. In other words, when referring to a Behavior Structure B we have in mind also alph(B).

Behavior Structures correlate with Pomset Processes as Milner Processes with Tomset Processes. A Behavior Structure exhibits a set of Pomsets (at the nodes), and it describes how they fit together.

Configuration Structures and the transformation CB

Definition: A (unlabeled) **Configuration Structure** (CS) is a collection B of partially ordered sets *(Posets)* $\{X, <_X\}$ with the following properties:

i) Every $X \in B$ is finite

ii) If $X \in B$ then every prefix of X is in B

iii) If $X, Y \in B$ and $b \in X \cap Y$ then b determines the same prefix in X and Y (set and order).

In particular - if $Y, X \in B$ and $Y \subset X$ then Y is a prefix of X and $<_Y$ is the restriction of $<_X$. Thus B itself is partially ordered by the relation "$Y \leq X$ iff Y is a prefix of X"; hence it may be visualized as an acyclic directed graph whose nodes are identified with the configurations $X \in B$.

Each $X \in B$ is called a configuration of B.
Let E be $\bigcup \{X \mid X \in B\}$. E is called *the set of events in B*.

A *labeled CS* is a CS together with a set (of actions) Σ and a labeling function

$$l : E \to \Sigma$$

For every $X \in B$ we denote by P_X the corresponding pomset whereas alph$(P_X) = \Sigma$ (in other words P_X is over Σ). Each node X in the labeled CS is labeled by a pomset P_X.

Clearly, the global labeling function induces two specific labelings in the acyclic graph of the nodes. First, each node X is now labeled by a pomset P_X where $V_{P_X} = X$ and alph$(P_X) = \Sigma$.

Second, if X is an immediate prefix of Y, then the arrow from X to Y is labeled by the action a from Σ which labels the only element in $Y - X$.

Summarizing we see that each LCS over Σ may be visualized as a standard automaton M over Σ, whose states are labeled by pomsets over Σ, where the following conditions hold:

a) If in M there is a transition $n \overset{a}{\to} m$ then P_n is an immediate prefix of P_m and $P_m - P_n$ consists of a single event labeled by a.

b) Assume Q is an immediate prefix of P_n and the event in $P_n - Q$ is labeled by a; then there is a unique m such that $m \overset{a}{\to} n$ in M and $P_m = Q$.

The transformation CB: LCS$\to$BS. Given an LCS C the transformation CB yields a BS B, with the embedding $\phi_{n,m}$ *uniquely* defined as follows: Let l(P_n) and l(P_m) be the pomsets assigned to nodes n, m in C; then $\phi_{n,m}$ identifies P_n unambiguously as a prefix of P_m.

Event Structures and the transformation EC

A somewhat more compact description of an LCS (and indirectly also of a BS) is through the set of events E and the order induced on it [W],[NPW].

Definition: An *event structure* (ES) is a triple $<E, \leq, \#>$ where $\leq$ is a finitary partial order (every element has a finite prefix) and $\#(A)$ is the conflict predicate over P(E) satisfying:

(1) If $\#(A)$ and for every $a \in A$ there is $b \in B$ such that $b \geq a$, then $\#(B)$. (Monotonicity).

(2) No singleton is in conflict.

Note that by (1) we have $\#(A)$ iff $\#(\overline{A})$ where $\overline{A}$ is the prefix determined by A.

A Labeled Event Structure (LES) is an ES with some labeling function $l : E \to \Sigma$.

Note : In [W80] different versions of the notion Event Structure are considered. In this paper we refer only to the version above.

The transformation ES: ES→CS. Given an Event Structure E, we obtain a Configuration Structure EC(E) as follows: take as configurations of EC(E) all conflict free prefixes of E. In the labeled case the labels of the events are preserved.

Definition:

(i) A conflict predicate $\#$ is **binary** if there is some binary relation $\#_2$ such that $\#(A)$ iff $a \#_2 b$ for some $a, b \in A$.

(ii) A CS B is **coherent** [NPW], if for $X, Y \in B$ either $X \bigcup Y \in B$ or $\overline{\{a,b\}} \notin B$ for some $a \in X$ and $b \in Y$.

It is easy to check the following:

Fact. $\#$ is binary on E iff EC(E) is coherent.

In [NPW] only binary conflict is discussed.

Mutual Retrievability of BS, LCS, LES

One can consider more transformations between BS, LCS, LES.

Transformation CE: CS→ES. Given a Configuration Structure B, we obtain an Event Structure CE(B) as follows: Let E be the set of events $(E = \bigcup \{X | X \in B\})$.

Define:

(i) e<f if $e <_X f$ for some $X \in B$.

(ii) $\#(A)$ if $\bar{A} \notin B$,where $\bar{A}$ is the prefix determined by A in E.

(iii) Finally -- CE(B)=$< E, <, \# >$

Transformation BE: BS$\rightarrow$LES. Given a Behavior Structure B, transformation BE yields a Labeled Event Structure E as follows:

First let us call a node of a BS prime if it has a unique immediate predecessor. From the definition of BS it follows that a node n is prime iff it is labeled by a pomset P_n which has a maximal element.
Take as events of E all prime nodes of B and label an event n by the label of the maximal element in P_n. The order of E is inherited from the order in B. A set of nodes A is in conflict in E if it has no upper bound in B.

The following claim slightly extends a result in [NPW]. It shows that BS, LES and LCS are retrievable from each other.

Claim 3: The following diagram commutes (see Fig. 4).

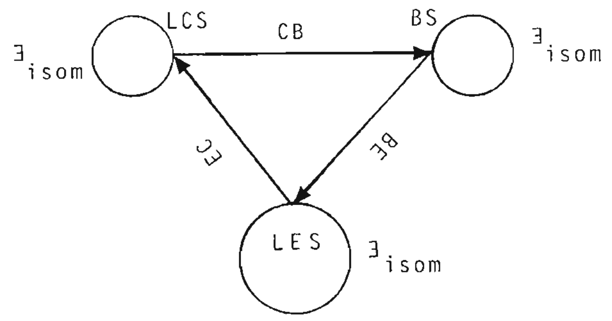


Fig. 4

Proof: Follows directly from the definitions.

1.4. Coarsening and Refining Processes

It is quite evident what coarsening should mean for those kind of processes we have considered. Let us look in more detail to the coarsening of the most discriminating processes - Behavior Structures.

Given a Behavior Structure B, it can be coarsened to a process P of lower level in one of the following ways (where the alphabet of B is *inherited* by P).

(a) **Pom(B)** is the collection of Pomsets in B (ignore the diagram).

(b) **The string language Lin(B)** is obtained by taking the strings of actions that lead to nodes in B (note that such a string is always a linearization of the Pomset at the node).

(c) **The Milnering M(B)** is the automaton whose states are the nodes of B (ignore the Pomsets). Together with the actions along the edges this turns the BS into a transition diagram.

(d) **The MultiMilnering MM(B)** is the multiautomaton whose states are the nodes of B and such that $n \xrightarrow{A} m$ if the events in $P_m - \phi_{n,m}(P_n)$ are maximal in P_m and A is the multiset of labels which decorate these events.

Example (see Fig 5). Consider the Behavior Structure B:

Its string language: Lin(B)= {a, ab, aa, ac, aab, aba, aac, aca, abac, acab, aabc, aacb, aaca, acaa}.

Pom(B) and MM(B) are shown in Fig. 5.

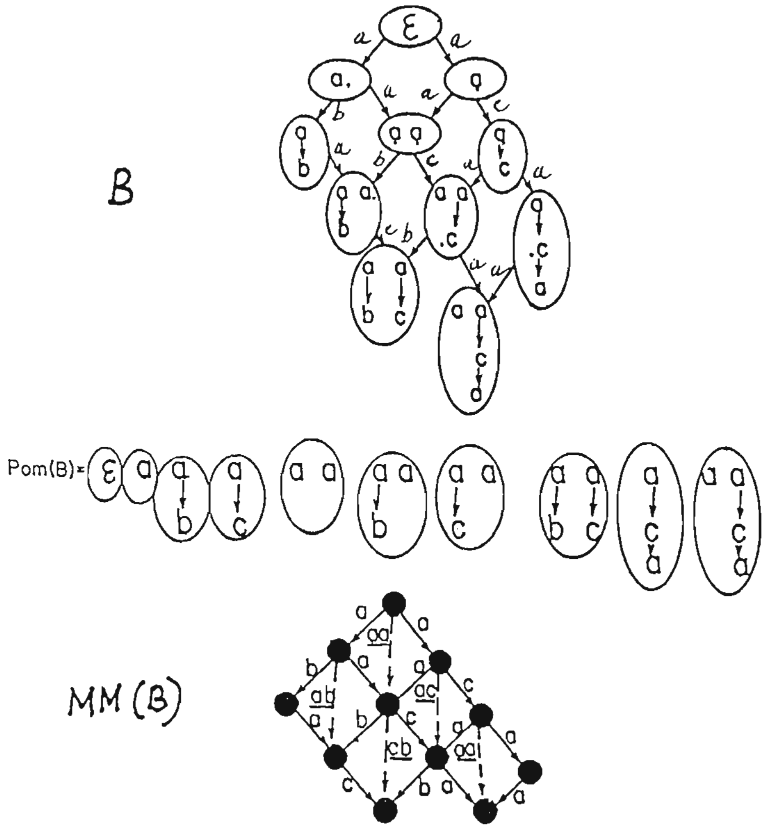


Fig. 5

For processes P_1, P_2 from models $l_1<l_2$ say that P_1 is embeddable in P_2 iff there exists a process P_1' of the same model l_1 as P_1 such that $P_1 \equiv_{l_1} P_1'$ and P_1' is the result of coarsening P_2. Accordingly, we consider the embedding relation $Emb_{l_1 l_2}$; $Emb_{l_1 l_2}(P_1, P_2) =_{def}$ P_1 from model l_1 is embeddable in P_2 from model l_2. Though coarsening is always possible it is not immediately evident that embedding is always possible. The following theorem shows that this is still the case.

Claim 4 (Embedding Theorem). For arbitrary $l_1<l_2$ and given process P_1 from model l_1 there exists a process P_2 from model l_2 such that $Emb_{l_1 l_2}(P_1, P_2)$ holds.

Proof: As previously we confine here with the case when l_2 refers to behavior structures and omit the indexes l_1, l_2 because they are clear from the context. We consider below the embeddings of Pomset Processes, Automata and Multiautomata.

a) For every Pomset Process P there is some Behavior Structure B such that $\mathrm{P} \equiv_{Pom} \mathrm{Pom(B)}$.

Assume that for each p, $q \in \mathrm{P}$ the sets of events V_p, V_q have empty intersection; otherwise we would first replace P by some $\mathrm{Q} \equiv_{Pom} \mathrm{P}$ for which this condition holds. Let us construct a labeled event structure E such that the corresponding BS will have the desirable property.

Events of E: For each pomset $p \in \mathrm{P}$ put in E the events of p ordered and labeled exactly as in p. For different pomsets p and q of P the events of p are in binary conflict with the events of q. Since P is prefix closed it follows that a pomset is in P iff it is isomorphic to a labeled configuration of E.

b) For every Automaton M there is some Behavior Structure B such that M is bisimular to M(B).

Construct a Behavior Structure B as follows:

Consider a tree like transition diagram T(M) which is bisimular to M; marking each node of T(M) by the string from the root to this node we get a Behavior Structure B. It is clear that M(B)=T(M); therefore M(B) is bisimular to M by the transitivity of Milner Park bisimulation.

c) For every Multiautomaton M there is some Behavior Structure B such that M is bisimular to MM(B).

First we show how to construct for a multiautomaton without autoconcurrency M a Behavior Structure without autoconcurrency B such that $\mathrm{MM(B)} \equiv_{MM} \mathrm{M}$.

An execution step-sequence σ of the multiautomaton M has the form

$$q_0 A_1 q_1 A_2 \cdots q_{n-1} A_n q_n$$

where q_i are states of M, q_0 is the initial state of M and $q_{i-1} \xrightarrow{A_i} q_i$.

Define B as follows:

As nodes of B take the execution sequences of M. As pomset which labels the node

$q_0 A_1 q_1 A_2 \cdots q_{n-1} A_n q_n$ take the corresponding stomset it generates: namely, the events labeled by A_i are considered to precede the events labeled by A_j for $i<j$.

Note that this is a pomset without autoconcurrency.

Now let us define the order between the nodes. Node σ immediately precedes node τ if one of the following conditions holds.

1) $\sigma = q_0 A_1 q_1 A_2 \cdots q_{n-1} A_n q_n$ and $\tau = q_0 A_1 q_1 A_2 \cdots q_{n-1} A_n q_n a q_{n+1}$ for some action a.

2) $\sigma = q_0 A_1 q_1 A_2 \cdots q_{n-1} A_n q_n$ and $\tau = q_0 A_1 q_1 A_2 \cdots q_{n-1} A_n \bigcup a \overline{q}_n$ for some action a such that $q_n \xrightarrow{a} \overline{q}_n$

It is clear that for $\sigma < \tau$ the pomset p_σ assigned to σ is a prefix of the pomset p_τ assigned to τ; moreover since this pomsets are without autoconcurrency there exists a unique embedding of p_σ as a prefix of p_τ. This completes the definition of B.

The construction is illustrated by Fig. 6.

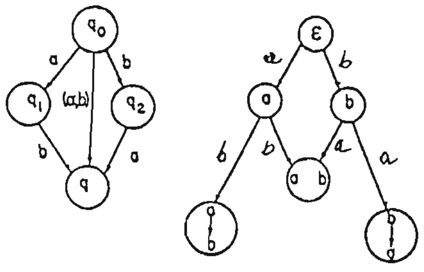


Fig. 6

We still have to check that there exists a MM bisimulation between MM(B) and M. It is easy to see that the relation R,

$$R(\sigma, q) \text{ iff } q \text{ is the last state of the step sequence } \sigma,$$

is such a bisimulation.

Finally, the validity of the claim for arbitrary multiautomata follows from the validity for multiautomata without autoconcurrency and from two facts that are connected to renaming of alphabets.

A renaming l of an action-alphabet Σ into an alphabet Σ' is simply a mapping of Σ into Σ'. Note that two different actions in Σ may be mapped into one in Σ'. If N is a process (automaton, BS, etc.) with alphabet Σ, the notation $l(N)$ is used for the process one gets when the actions are everywhere renamed according to l.

Fact 1: If $B_1 \equiv_{MM} B_2$ then for arbitrary renaming l, it is the case that $l(B_1) \equiv_{MM} l(B_2)$.

Fact 2: For every multiautomaton M there exists a multiautomaton without autoconcurrency N and a relabeled version $l(\mathrm{N})$ of N such that $\mathrm{M} \equiv_{MM} l(\mathrm{N})$.

Claim 4 does not tell the full story. There is usually more than one candidate BS. Sometimes the choice is very natural such as the choice above for automata, but in general for Pomset Processes and Multiautomata a natural construction is not so obvious. We shall return to this topic later.

1.5. Equivalences among Behavior Structures

Warning: equivalent Processes are always assumed to have the same alphabet.

Definition: Let B and C be two Behavior Structures.

a) B and C are **linear equivalent** ($B \equiv_{Lin} C$) if Lin(B)=Lin(C).

b) B and C are **Pomset equivalent** ($B \equiv_{Pom} C$) if Pom(B)$\equiv_{Pom}$Pom(C).

c) B and C are **M Bisimular** ($B \equiv_M C$) if M(B) and M(C) are M - bisimular.

d) B and C are **MM Bisimular** ($B \equiv_{MM} C$) if MM(B) and MM(C) are MM - bisimular.

e) A **BS-bisimulation** between Behavior Structures B and C is a ternary relation R(n,m,f) such that:

(1) n is a node of B , m is a node of C and f is an isomorphism between P_n and P_m.

(2) $R(\epsilon,\epsilon,\emptyset)$.

(3) If R(n,m,f) and $n \overset{a}{\rightarrow} n_1$ then also $R(n_1,m_1,f')$ for some m_1 with $m \overset{a}{\rightarrow} m_1$ and for some f' extending f.

(4) The dual requirement of (3) wrt B and C.

$$\begin{tikzcd}
P_m \arrow[r, "f"] \arrow[d, "\phi_{m,m_1}"'] & P_n \arrow[d, "\phi_{n,n_1}"] \\
P_{m_1} \arrow[r, "f'"] & P_{n_1}
\end{tikzcd}$$

Fig. 7

B and C are **BS Bisimular** ($B \equiv_{BS} C$) if there is a BS bisimulation between them.

Claim 5 (Non retrievability of BS).

(1) $\equiv_{BS}$ implies $\equiv_{MM}$, $\equiv_{M}$ and $\equiv_{pom}$.

(2) $\equiv_{BS}$ is a strong refinement of the equivalences which arise via coarsing Behavior Structures. It may happen that $MM(B_1)\equiv_{MM}MM(B_2)$ and $Pom(B_1)\equiv Pom(B_2)$ but B_1 is not $\equiv_{BS}$ equivalent to B_2.

Proof: (1) From $B_1\equiv_{BS}B_2$ it follows obviously that $B_1\equiv_{pom}B_2$.

Let us show that $B_1\equiv_{BS}B_2$ implies $B_1\equiv_{MM}B_2$. Given a BS bisimulation $R(m,n,f)$ between B_1 and B_2, consider the following relation $\overline{R}(n,m)$ between nodes of MM(B_1) and nodes of MM(B_2):

$$\overline{R}(m,n) \text{ iff there is } f \text{ such that } R(m,n,f).$$

Let us show that $\overline{R}$ is a MM - bisimulation between MM(B_1) and MM(B_2).

(i) Clearly, the initial nodes are related by $\overline{R}$.

(ii) Assume that $\overline{R}(m,n)$ holds and that $m\xrightarrow{A}\overline{m}$ for a multiset $A=\{a_1,...a_k\}$. Then, for some f $R(m,n,f)$ holds and therefore there exist $m_1, \cdots m_{k-1}$ such that

$$m\xrightarrow{a_1}m_1$$

$$m_i\xrightarrow{a_{i+1}}m_{i+1} \text{ for } i=1, \cdots k-2$$

$$m_{k-1}\xrightarrow{a_k}\overline{m}$$

Since R is a BS - bisimulation there exist $n_1, \cdots n_{k-1}$ and $f_1, \cdots f_{k-1}, \overline{f}$ such that

$$n\xrightarrow{a_1}n_1$$

$$n_i\xrightarrow{a_{i+1}}n_{i+1} \text{ for } i=1,k-2$$

$$n_{k-1}\xrightarrow{a_k}\overline{n}$$

$$R(m_i,n_i,f_i),$$

$$R(\overline{m},\overline{n},\overline{f})$$

Moreover, the following extensions hold:

$$f_{i+1} \text{ extends } f_i \text{ for } i=1,...k-2$$

$$f_1 \text{ extends } f$$

$$\overline{f} \text{ extends } f_{k-1}.$$

Therefore, by the definition of $\overline{R}$ it follows that $\overline{R}(\overline{m},\overline{n})$ holds. Since $\overline{f}$ is an isomorphism between the pomsets $\overline{m}$ and $\overline{n}$ and it maps a prefix m of $\overline{m}$ onto n it follows that $n\xrightarrow{A}\overline{n}$. This completes the proof of requirement (ii) from the definition of MM - bisimulation (see 1.1). The dual requirement is proved in a similar way.

(2) Here is an example of two Behavior Structures B_1 and B_2 which are $\equiv_{MM}$ and $\equiv_{pom}$ equivalent but there is no BS - bisimulation between them.

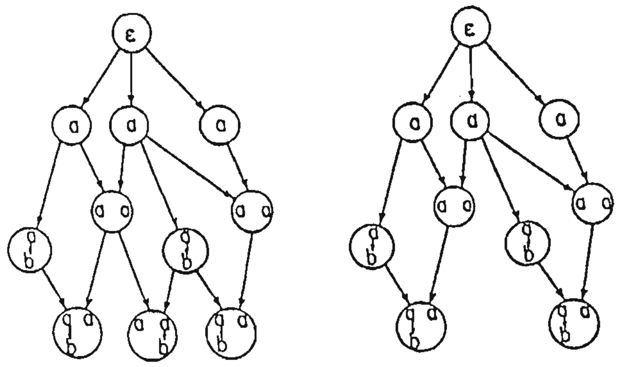


Fig. 8

1.6. Synchronization of Behavior Structures

This is the crucial operation and we aim at a careful and detailed definition.

We start with a sequence of Behavior Structures

$$B_1, B_2, \ldots \quad (*)$$

with typical notations m_i, n_i,... for the nodes of B_i and $\overline{m}_i$, $\overline{n}_i$,... for the pomset labels of these nodes. The result B of the synchronization of the sequence (*) is defined in the points (i)-(ii)-(iii) below.

(i) **Nodes** of B (typical notations: M, N, ...)

A node M is a concrete synchronized embedding (see 1.1):

$$\mathrm{M}=<P, \overline{m}_1, f_1, \overline{m}_2, f_2, \ldots>$$

Next we have to describe the condition under which for a pair of nodes

$$\mathrm{M}=<P, \overline{m}_1, f_1, \overline{m}_2, f_2, \ldots> \quad \mathrm{N}=<Q, \overline{n}_1, g_1, \overline{n}_2, g_2, \ldots>$$

it is the case that M immediately precedes N via a transition $\mathrm{M}\xrightarrow{a}\mathrm{N}$ and, when this holds what is the function ϕ_{MN} which embeds $\overline{M}$ into $\overline{N}$. Before we proceed to these important points of the definition let us choose more detailed notations for the pomsets $\overline{m}_i$, $\overline{n}_i$, P, Q, namely:

$$\overline{m}_i=< V_i, \leq_i, l_i, \Sigma_i >; \quad P=< V, \leq, l, \Sigma >$$
$$\overline{n}_i=< V'_i, \leq'_i, l'_i, \Sigma'_i >; \quad Q=< V', \leq', l', \Sigma' >$$

Actually we shall refer the elements of a pomset as labeled events, i.e. pairs <event, label>.

Let us go on with the definition.

(ii) **Immediate precedence in** B

$\mathrm{M}\xrightarrow{a}\mathrm{N}$ iff there is a set I of indices and an action a which satisfy two conditions:

Condition 1. If $j \notin I$ then $m_j = n_j$ and $a \notin \mathrm{alph}(B_j)$ otherwise, if $j \in I$ then $m_j \xrightarrow{a} n_j$ and $a \in \mathrm{alph}(B_j)$.

Before we formulate the other condition note that if $j \in I$ there exists an embedding of $\overline{m}_j$ into $\overline{n}_j$. We use the notation ϕ_j for this embedding and the notation $<\delta_j, \alpha_j>$ for the unique labeled event in $\overline{n}_j$ which is not in $\phi_j(\overline{m}_j)$.

Condition 2. For each $j \in I$ the label α_j is just a and the function g_j is an extension of the composition of ϕ_j and f_j; for $j \notin I$ the functions f_j and g_j coincide.

Now, we are going to describe the embedding function ϕ_{MN} (assumed that M immediately precedes N); the general case follows via transitivity. Recall that since M, N are identified with *concrete* synchronizing embeddings each element $d \in V$ is an appropriate sequence and so is each element $d' \in V'$ (see Section 1 for the definition of *concrete* embeddings).

(iii) **Definition of the embedding function ϕ_{MN}:**

Assume that M immediately precedes N then ϕ_{MN} is defined as follows. For $<t_1, t_2, \ldots> \in V$ define $\phi_{MN}(<t_1, t_2, \ldots>) = <\tau_1, \tau_2, \ldots> \in V'$ where

$$\tau_i = \begin{cases} nil & \text{if } t_i \text{ is } nil; \text{ otherwise} \\ t_i & \text{for } i \notin I \\ \phi_i(t_i) & \text{for } i \in I \end{cases}$$

(recall: ϕ_i embeds V_i into V'_i for $i \in I$)

Now if $M < N$ then there is a finite chain $M_1, M_2, \ldots M_k$ of nodes between M and N such that $M = M_1$, $M_i << M_{i+1}$ and $N = M_k$. Define ϕ_{MN} as the composition of the embeddings along this chain. It can be shown that ϕ_{MN} does not depend on the choice of the chain between M and N. Moreover, relying on the fact that B_i are Behavior Structures, it is a routine task to check that the definition is correct, i.e., that the embeddings behave properly.

The definition of Synchronization for Behavior Structure is completed.

Comment 1. Now, let us give some explanations which will help to see that the definition is reasonable. Assume that $M \xrightarrow{a} N$ and let $<\delta, \alpha>$ be the unique labeled event in N which is not covered by ϕ_{MN}. Then, for the δ_j ($j \in I$) defined earlier, the following holds:

(1) $\phi_j(\delta_j) = \delta$ and δ (as well as δ_j) is labeled with the action a.

On the other hand, for each $x \in V_i$,

(2) $f_i(x) \in V$, $\phi_{MN}(f_i(x)) \in V'$, $\phi_i(x) \in V'$, $g_i(\phi_i(x)) \in V'$ and $\phi_{MN}(f_i(x)) = g_i(\phi_i(x))$

Comment 2. As in the case of synchronization for automata (Section 1.2) it may happen that even starting with countable Behavior Structures, the construction described above produces an uncountable set of nodes. Note that only nodes reachable from the root may be produced; that is because if some pomsets are synchronizable so are their prefixes. Again as in the case for automata one can check that a countable set of nodes will be produced if the finite degree condition holds: for each action a there is only a finite set of B_i for which $a \in \text{alph}(B_i)$. In this case one can restrict oneself from the very beginning to "finitary" nodes, that is with concrete synchronizing embeddings $<P, m_1, f_1, m_2, f_2, \ldots>$ where all the m_i but finite numbers of them are roots of the B_i.

Now we are prepared to formulate the main theorems, which characterize the Synchronization of Behavior Structures.

Claim 6. Synchronization of Behavior Structures is fully compositional.

Proof: omitted.

Claim 7 (Equivalence Inheritance Theorem).

Synchronization inherits all the equivalences (considered above) among Behaviors Structures.

Proof: We show here that $\equiv_{pom}$, $\equiv_{MM}$, $\equiv_{M}$, and $\equiv_{BS}$ are congruences with respect to synchronization of Behavior Structures.

(1) For $\equiv_{pom}$ it is straightforward.

(2) We consider now the equivalence $\equiv_{MM}$; the case of $\equiv_{M}$ is essentially the same. For simplicity, we consider only synchronization of two BS's.

Assume that R_1 is a MM-bisimulation between B_1, B_1', and R_2 is a MM-bisimulation between B_2, B_2'.

We use the notations

m, m_1, for nodes of B_1 ,

m', m'_1,.. for nodes of B_1'

n, n_1, for nodes of B_2 ,

n', n'_1,.. for nodes of B_2'

$<P, m, f_1, n, f_2>$ is a typical node of $Synch(B_1, B_2)$ and of $MM(Synch(B_1, B_2))$

$<P', m', f_1', n, f_2'>$ is a typical node of $Synch(B_1', B_2')$ and of $MM(Synch(B_1', B_2'))$

We define an MM-bisimulation R between $Synch(B_1, B_2)$ and $Synch(B_1', B_2')$.

$R(<P, m, f_1, n, f_2>, <P', m', f_1', n, f_2'>)$ iff there following conditions hold:

(i) $R_1(n, n')$

(ii) $R_2(m, m')$

It is straightforward, but tedious to check that R is a MM - bisimulation and we omit this verification here.

(3) Assume that R_1 is a BS-bisimulation between B_1, B_1', and R_2 is a BS-bisimulation between B_2, B_2'.

In addition to the notations above we use

θ_1 for isomorphism between pomsets at nodes of B_1 and B_1'

θ_2 for isomorphism between pomsets at nodes of B_2 and B_2'

θ for isomorphism between pomsets at nodes of $Synch(B_1, B_2)$ and $Synch(B_1', B_2')$.

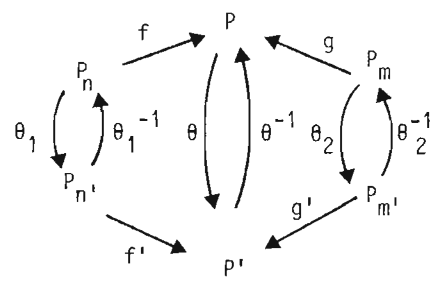
Now we define a BS-bisimulation R between $Synch(B_1, B_2)$ and $Synch(B_1', B_2')$.

$R(<P, m, f_1, n, f_2>, <P', m', f_1', n, f_2'>), \theta)$ iff there are θ_1 and θ_2 such that the following conditions hold:

(i) $R_1(n, n', \theta_1)$

(ii) $R_2(m, m', \theta_2)$

(iii) The following diagram commutes (see Fig. 9).



$$\begin{tikzcd}
 & P \arrow[dd, bend left, "\theta"] & \\
P_n \arrow[ur, "f"] \arrow[d, bend right, "\theta_1"'] & & P_m \arrow[ul, "g"'] \arrow[d, bend right, "\theta_2"'] \\
P_{n'} \arrow[u, bend right, "\theta_1^{-1}"'] \arrow[dr, "f'"'] & & P_{m'} \arrow[u, bend right, "\theta_2^{-1}"'] \arrow[dl, "g'"] \\
 & P' \arrow[uu, bend left, "\theta^{-1}"] &
\end{tikzcd}$$

Fig. 9

It is straightforward (but tedious) to check that R is BS - bisimulation and we omit this verification here.

Claim 8 (Embedding Inheritance). The embedding relations are inherited by synchronization.

Proof. Simplicity for we consider the synchronization of two operands. The embedding of pomset processes is simple and is left as an exercise. The embeddings of multiautomata is similar (though much more tedious) to the embedding of automata. So what we are going to check is the following (for automata M_i and behavior structures B_i). Assume

$$M_1 \equiv_M M(B_1), \quad M_2 \equiv_M M(B_2) \tag{1}$$

Then it should also hold

$$Synch(M_1, M_2) \equiv_M M(Synch(B_1, B_2)) \tag{2}$$

First we note, that since synchronization of automata inherits M-bisimulation (see

Claim 2) it follows from (1) that:

$$Synch(M_1, M_2) \equiv_M Synch(M(B_1), M(B_2)) \tag{3}$$

The comparison of (2) and (3) shows that what one actually has to check is

$$Synch(M(B_1), M(B_2)) \equiv_M M(Synch(B_1, B_2)) \tag{4}$$

To this end let us use the following notations for the nodes of the Behavior Structures above and hence also for the states of the respective automata:
$m_1, n_1, \dots$ nodes of B_1 (states in $M(B_1)$)
$m_2, n_2, \dots$ nodes of B_2 (states in $M(B_2)$)
$m, n, \dots$ nodes in $B =_{def} Synch(B_1, B_2)$ (states in $M(B)$)

Recall that actually $m, n, \dots$ have the format

$$<P, m_1, f_1, m_2, f_2>, \quad <Q, n_1, g_1, n_2, q_2>$$

for appropriate $P, Q, f_i, g_i, \dots$

The states of $Synch(M(B_1), M(B_2))$ are pairs $<m_1, m_2>, <n_1, n_2>$.

Recall that our goal is to prove (4). But the following relation R

$$<P, m_1, f_1, m_2, f_2> R <m_1, m_2>$$

between nodes of $A =_{def} Synch(M(B_1), M(B_2))$ and $N =_{def} M(Synch(B_1, B_2))$ is a M-bisimulation.

Indeed, the initial nodes are related by R. Note that for a tuple $<m_1, m_2>$ in A there corresponds some tuple $<P, m_1, f_1, m_2, f_2>$ in N only if P_{m_1} and P_{m_2} are synchronizable. Moreover in this case if $<m_1, m_2> \xrightarrow{a} <n_1, n_2>$ then P_{n_1} and P_{n_2} are synchronizable and there exists an extension Q of the pomset P and extensions g_i of f_i such that $<P, m_1, f_1, m_2, f_2> \xrightarrow{a} <Q, n_1, g_1, n_2, g_2>$ in $Synch(B_1, B_2)$. Therefore if there exists in A a transition $<m_1, m_2> \xrightarrow{a} <n_1, n_2>$ then in $M(Synch(B_1, B_2))$ there exists a transition $<P, m_1, f_1, m_2, f_2> \xrightarrow{a} <Q, n_1, g_1, n_2, g_2>$ to a state related by R with $<n_1, n_2>$. The dual condition holds obviously.

2. Nets of Processes

2.1. Provisos and Terminology

Following the terminology of Petri Nets we use the term *Net* for a bipartite (not necessary oriented) graph with nodes of two kinds, pictured respectively as circles and boxes. The difference between the two kinds is relevant for the notion of subnet. A subgraph N' of N is considered to be a subnet if the set of its nodes consists of *some* circles and of *all* boxes which are adjacent to these circles. In particular an *atomic subnet* contains a single circle and all its neighboring boxes. This is to be contrasted with an *atomic bunch* which contains a single box and all its neighboring circles. For technical reasons, that are not essential at this stage but may be convenient in applications to Data Flow and Petri Nets, the following restrictions upon nets are assumed from now on:

Lack of small loops: no atomic subnet may contain a loop.

Among the oriented nets, we first mention those which obey the additional restriction:

a) *There are no oriented loops in the net.* Hence, the set of all nodes is partially ordered.

For such nets, in addition to the partial order relation $\leq$ among nodes, we also consider two more binary relations:

(1) The binary conflict relation #

(2) The binary concurrency relation *co*

Since x *co* y is defined as the negation of "$x \leq y$ or $x \# y$" it remains to explain only what $x \# y$ means.

The definition of conflict:

(a) $box_1 \# box_2$ are in (direct!) conflict in the situation

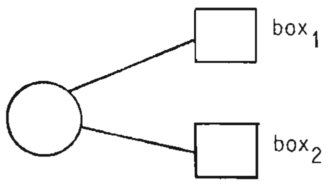


Fig. 10

(b) if $b_1 \# b_2$ and $b_1 \leq b'_1$ and $b_2 \leq b'_2$ then $b'_1 \# b'_2$.

Now, an occurrence Net is an Oriented Net to which, in addition to restriction a) the following restrictions also apply:

b) *Finitary partial order,* i.e. each node may have only a finite number of box predecessors. Hence there exist initial nodes (with no predecessors).

c) *All the initial nodes are circles*

d) Each circle has at most one input arrow (an oriented arc from a neighboring box, directed to this circle).

Given an occurrence net H one associates with it an event structure Ev(H) with binary conflict relation # as follows:

(1) *Events:* the boxes of H.

(2) *Partial order, conflict and concurrency - inherited from H.*

Appropriately labeled nets provide a useful pictorial representation for processes. Some of them, called also flow graphs, reflect the way complex processes are assembled from "elementary" ones. For example, an elementary process (agent) P over the action alphabet {a,b,c} is pictured as a circle labeled by P, from which there emerge three lines to boxes, each labeled with one of the alphabet actions (see Fig. 11).

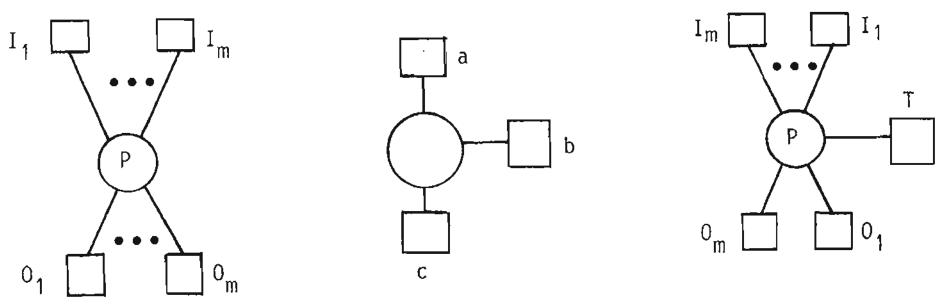


Fig. 11

Accordingly, when a set (which may be infinite) of agents are put together to evolve concurrently, the resulting system may be pictured as a net in which equally labeled boxes are identified. (Compare two nets from Fig. 11 with the net of Fig. 12.)

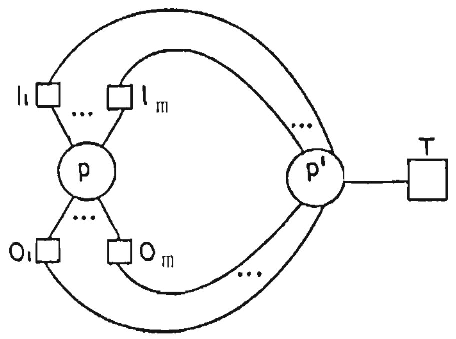


Fig. 12

On the other hand Labeled Occurrence Nets, through the interpretation Ev mentioned above, represent Labeled Event Structures and ultimately Behavior Structures as well. In contrast with Flow Graphs, which reflect the spacial structure of processes, Occurrence Nets record the temporal and causal structure of processes.

Nets of Processes

Fix some kind l of processes (Automata, BS, etc. ...). A Net of l-processes is a Net N with an appropriate labeling ϕ, i.e. formally a pair $<N,\phi>$, where

(1) ϕ assigns to each box b an action and different boxes get different labels. In this way each atomic subnet of N becomes qualified, i.e. associated with an action alphabet.

(2) ϕ assign to each circle c and hence to each atomic subnet a l-process $\phi(c)$ with the only restriction that the alphabet of $\phi(c)$ coincides with the alphabet of the atomic subnet "around" c.

Notes

a) By abuse of notation we shall not distinguish the boxes in $<N,\phi>$ and their labels. Due to (1) this is not dangerous.

b) The only situation when different circles c_1, c_2 may be equally labeled by ϕ is illustrated in the Fig. 13, where the atomic nets induced by c_1, c_2 have the same box neighbours.

c) In the literature about nets the circles are called places, whereas the boxes are called transitions or events (for Petri Nets); we use also - *ports*.

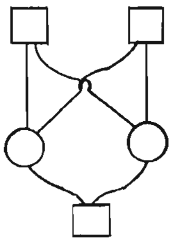


Fig. 13

Finally, for the semantics (behavior) of a net $<N, \phi>$ one declares the process

$$l(N,\ \phi) =_{def} Synch_l(\phi(p)\ :\ p \text{ is a place in } N)$$

where *Synch* is parametrized wrt the process model under consideration (Automata, Multiautomata, Pomset Processes, Behavior Structures).

2.2. Examples of Nets over Deterministic Automata

Recall that an automaton is called deterministic iff each node has at most one transition arrow with a given label a.

Data Flow Nets (DFN)

A DFN is a net over deterministic automata whose actions, called also *communications* are described by pairs $<c, v>$; here c is the name of a *channel* on which the communication take place and v is the value of a message which is passed through this channel. In principle communications may be dealt as with ordinary actions and used as labels of ports. But introducing the special notation is suggestive of a more compact and convenient representation of the flow graph. When drawing an agent the ports get labels only from the alphabet CH of channels.

Nets over C/E automata

C/E automata (C/E elementary agents) have the following simple form. (The relation of C/E automata to C/E Petri Nets is discussed in Appendix.)

(1) An elementary agent has only two states called *full* and *empty*.

(2) Actions are either input actions or output actions.

(3) An input action transforms the automaton from the empty state to the full state; an output action transforms the automaton from the full state to the empty state.

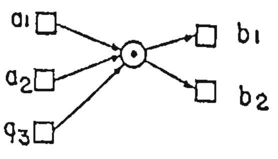


Fig. 14

Graphically a C/E automaton is represented by a circle and its ports by boxes. There are arcs from the input ports of the automaton to the circle and arcs from the circle to the output ports. The full state is represented by drawing a dot in the circle.

2.3. Robust BS-embeddings

Assume a given assortment $As=\{A_i\}$ of multiautomata agents and consider all possible nets over As, i.e. all nets $<N, \psi>$ in which $\psi(p)\in As$ or $\psi(p)$ is result of *1-1 renaming* of actions for a process $Q\in As$. According to the definition above each such net $<N, \psi>$ has a multiautomaton semantics MM(N, ψ). A BS - semantics Sem(N, ψ) for these nets is said to be **robust** if the following two conditions hold:

(a) (Consistency) MM(Sem(N, ψ))$\equiv_{MM}$MM(N, ψ).

(b) (Compositionality) Assume N is decomposed into disjoint subnets N_1,N_2, $\cdots$; then Sem(N)=Synch(Sem(N_1, ψ_1)), Sem(N_2,ψ_2),...), where the ψ_i are the respective restrictions of ψ .

If these conditions hold we also say that the multiautomaton semantics MM(N, ψ) is robustly embedded in the BS - semantics Sem(N, ψ).

Claim 1:

(i) A robust BS - embedding is always possible.

(ii) If the agents in As are automata then the robust BS - embedding is unique (up to BS - bisimulation).

Proof: As a direct consequence of the Embedding Inheritance Theorem, and the Full Compositionality Theorem, (Claims 7,8 Section 1) in order to establish a robust BS-semantics for nets over a given assortment $As=\{A_i\}$ of agents one should proceed as follows:

a) *Consistent embedding of agents:* Assign to each $A_i\in As$ a Behavioral Structure B_i such that $MM(B_i)\equiv_{MM}A_i$.

This is possible according to the Embedding Theorem (claim 3 of Section 1). Moreover if A is an automaton then there is a unique (up to BS-bisimulation) Behavior Structure B_A which MM-consistently embeds A. Namely, consider a tree-like transition diagram T(A) which is bisimular to A; marking each node of T(A) by the string from the root to this node we get the unique Behavior Structure B_A.

b) *Compositional extending on Nets:*

$$Sem(N, \psi)=Synch_{BS}(\psi'(p): `p\in N)$$

where $\psi'(p)$ is a BS in which the agent $\psi(p)$ was embedded consistently according to a).

Warning. In this sequel when we deal with nets $<N, \psi>$ over over automata under BS(N, ψ) we have in mind just the unique robust BS-semantics $Sem(N, \psi)$ as defined above. But note that in the case of Nets over multiautomata there may be many robust BS-semantics (see 3.2 below).

2.4. Trace Semantics

Now let us return to nets over arbitrary automata. A pomset behavior of such a net may be obtained from its linear behavior in the same way as Mazurkiewicz [Maz84] obtained it in the case of Condition Event nets.

To this end first we define a dependency relation on the set of ports of a net N as follows:

Given an automaton A, we introduce a dependency relation between its ports, namely all ports are considered to be pairwise dependent.

Further, the dependency relation for the net N is defined to be the union of the dependency relations of its components.

Now, for a given dependency relation $D\subset\Sigma\times\Sigma$ and string s over Σ a pomset, $P_D(s)$ is defined inductively as follows.

(1) $P_D(\epsilon)$ is the empty pomset over Σ .

(2) For $s\in\Sigma^*$, and $a\in\Sigma$ the pomset $P_D(s.a)$ is obtained from the pomset $P_D(s)=<E_s,<_s>$ by the following procedure:

  (a) Add to E_s a new event e_{new} labeled by a.

  (b) For each element e of E_s labeled by a port which depends on a let $e<e_{new}$; finally, let $<_{s.a}$ be the transitive closure of $<_s\bigcup <$.

Let $<N, \psi>$ be a net with dependency relation D and linear behavior L. Then, by definition, the trace semantics $Trace(N, \psi)$ of the net $<N, \psi>$ is $\bigcup\limits_{s\in L} P_D(s)$.

Let $BS(N, \psi)$ be the (unique) robust BS-semantics of the net under consideration. The following claim shows that despite the fact that $Trace(N, \psi)$ and $BS(N, \psi)$ are defined in quite different ways they are consistent with each other.

Claim 2: $Trace(N, \psi)=Pom(BS(N, \psi))$.

Proof. Recall that $BS(N, \psi)$ is obtained by embedding for each p the automaton A_p (assigned to the place p) into a unique (up to BS-bisimulation) Behavior Structure B_p which is MM-bisimular to A_p, and then taking the synchronization of all B_p:

$$B=Synch(B_p : p\in N)$$

In accordance with the Embedding Inheritance Theorem (Claim 8) it holds $Pom(B)=Synch_{Pom}(Pom(B_p) : p\in N)$. But since the A_p are automata, it follows that $Pom(B_p)$ consists of strings (tomsets). And the claim now follows from the following

Fact. Assume that $s_1, s_2, \cdots$ are strings over alphabets Σ_i respectively. Then

$$Synch_{Pom}(s_1, s_2,...)=\bigcup_{s\in L} P_D(s)$$

where $L=Synch_{tomset}(s_1, s_2,...)$ and the dependency relation D is $\bigcup\limits_i \Sigma_i\times\Sigma_i$

2.5. Occurrence Semantics

Here we propose an occurrence net semantics for nets over automata, via a straightforward generalization of the occurrence net semantics for Petri Nets [NPW]. We assign to a net $<N, \psi>$ over automata an occurrence net N^{ax} axiomatically and an occurrence net N^{op} in an operational way. We prove that $N^{ax}=N^{op}$. Moreover, it turns out that the Labeled Event Structure associated with them (as explained in Section 2.1) is nothing but a description of the (unique!) robust BS - semantics. We use the definitions of occurrence nets and of the relations $<$, $\#$, and *co* on as defined above in 2.1. We use also some Petri Net terminology according to which in a bunch of a box (event), a place is called precondition if it precedes the box and postcondition otherwise.

2.5.1. The Axiomatic Approach

Let $<N, \psi>$ be a net over automata . We are looking for an occurrence net N^{ax} with places labeled by states of these automata and ports labeled by labels of ports in $<N, \psi>$. The net N^{ax} should satisfy the axioms A1-A3.

(A1) Assume that a port $e \in N^{ax}$ is labeled by a where in $<N,\psi>$ the bunch of a is labeled by the automata $A_1, \ldots, A_k$; then for the bunch of e in N^{ax},

(*) e has k preconditions labeled $p_1, p_2, \ldots, p_k$ and k postconditions $p'_1, p'_2, \ldots, p'_k$. Here for each i p_i and p_i' are states of A_i and $p_i \xrightarrow{a} p'_i$ in A_i (see Fig. 15).

(A2) To each place $p \in <N, \psi>$ there corresponds in N^{ax} a place labeled with the initial state of $\psi(p)$; these are the initial places in N^{ax}.

(A3) Assume that some k places $pl_1, pl_2, \ldots pl_k$ are in *co* relation and that they are labeled by states $p_1, p_2, \ldots, p_k$ of $A_1, \ldots, A_k$. Assume also that a is a port in $<N, \psi>$ whose bunch uses just these labels $A_1, \ldots, A_k$. Then for each tuple of states $p'_1, p'_2, \ldots, p'_k$ of $A_1, \ldots, A_k$ such that a leads from p_j to p'_j in A_j there is in N^{ax} exactly one bunch with port labeled by a with the preconditions $pl_1, pl_2, \ldots pl_k$ and postconditions labeled $p'_1, p'_2, \ldots, p'_k$ (see Fig. 15).

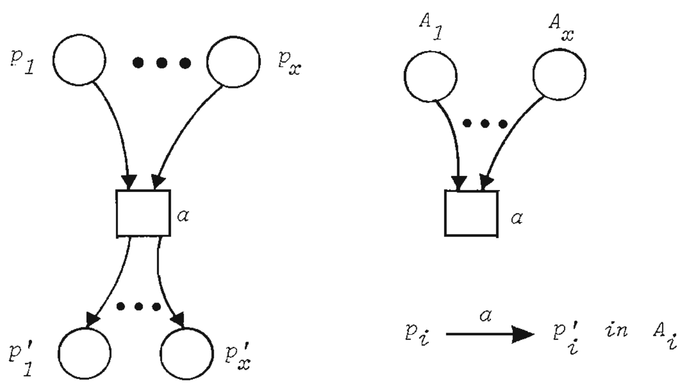


Fig. 15

2.5.2. The Operational Approach

Now we assign operationally to the net $<N, \psi>$ over automata an occurrence net N^{op}. First note that a linear behavior of $<N, \psi>$ is best given by a sequence of the form $\sigma = M_0 a_0 M_1 a_1 ... M_n a_n M_{n+1}$ where:

(1) M_i for $i \leq n+1$ is a global state of $<N, \psi>$, i.e. a function which assigns to each place pl in $<N, \psi>$ a state of the automaton $\psi(pl)$

(2) a_i are ports of N ;

(3) M_0 is the initial global state of N.

(4) a_i transforms the global state M_i to the state M_{i+1}, i.e. for each place pl in the bunch of a_i, the transition $M_i(pl) \xrightarrow{a_i} M_{i+1}(pl)$ may be performed by the automaton $\psi(pl)$.

The construction of N^{op} heavily relies on an appropriate equivalence $=_3$ between linear behaviors of N, to be defined below. Namely, the ports of N^{op} are identified with the $=_3$ equivalence classes.

First we define on linear behaviors three relations $\underset{1}{=}, \underset{2}{=}, \underset{3}{=}$ as follows.

We say that $\sigma \underset{1}{=} \tau$ if:

(1) $\sigma = M_0 a_0 ... M_{i-1} a_{i-1} M_i a_i M_{i+1} ... M_n a_n M_{n+1}$ and $\tau = M_0 a_0 ... M_{i-1} a_i M'_i a_{i-1} M_{i+1} ... M_n a_n M_{n+1}$ and $i < n$

(2) there is no automaton which contains both ports a_{i-1} and a_i. (a_{i-1} and a_i do not belong to an atomic subnet of $<N, \psi>$).

We say that $\sigma \underset{2}{=} \tau$ if

(1) $\sigma = M_0 a_0 \ldots M_{n-1} a_{n-1} M_n a_n M_{n+1}$ and $\tau = M_0 a_0 \ldots M_{n-1} a_n M'_n$

(2) there is no automaton which contains both a_{n-1} and a_n

(3) M'_n and M_{n+1} coincide in the bunch of a_n

Finally the relation $\underset{3}{=}$ is defined to be the reflexive and transitive closure of $\underset{1}{=} \cup \underset{2}{=}$.

Remark: if $\sigma = M_0 a_0 M_1 \ldots M_n a_n M_{n+1}$ and $\tau = M_0 a'_0 M'_1 \ldots M'_k a'_k M'_{k+1}$ are $\underset{3}{=}$ equivalent then:

(i) $a_n = a'_k$

(ii) each pl in the bunch of a_n has the same local state in both M_{n+1} and M'_{k+1} ($M_{n+1}(pl) = M'_{k+1}(pl)$); it has also the same local state in both M_n and M'_k ($M_n(pl) = M'_k(pl)$).

Let us return to the construction of N^{op}.

Ports of N^{op}: For each $\sigma = M_0 a_0 M_1 \ldots M_n a_n M_{n+1}$ there is a port associated with $\sigma / \underset{3}{=}$. This port is labeled a_n. This is a correct definition due to *Remark* 1 above.

Places: With the equivalence class of $\sigma = M_0 a_0 M_1 \ldots M_n a_n M_{n+1}$, where in $<N, \psi>$ a_n is the common port of $A_{i_1}, A_{i_2}, \ldots, A_{i_k}$ we associate a set $Pl(\sigma)$ of k places labeled by the local states of $A_{i_1}, A_{i_2}, \ldots, A_{i_k}$ in the global state M_{n+1}. These places are the postconditions of the port assigned to $\sigma / \underset{3}{=}$.

To each place $p \in <N, \psi>$ there corresponds in N^{op} a place labeled with the initial state of $\psi(p)$; these are the initial places in N^{ax}.

To accomplish the definition of N^{op} we have still to explain when a place p in N^{op} is a precondition of a port e. To this end assume that $\sigma = M_0 a_0 M_1 \ldots a_{n-1} M_n a_n M_{n+1}$, $\tau = M_0 a_0 M_1 \ldots M_{n-1} a_{n-1} M_n$, and that a_{n-1}, a_n are ports of an automaton A_k; then the place of A_k associated with τ will be a precondition of the port assigned to σ. If $\sigma = M_0 a_0 M_1$ then the preconditions of the port assigned to σ are those initial places of the N^{op} which correspond to the automata containing a_0.

Example: Consider the following net over C/E automata.

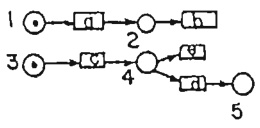


Fig. 16

the assigned occurrence net is in Fig 17.

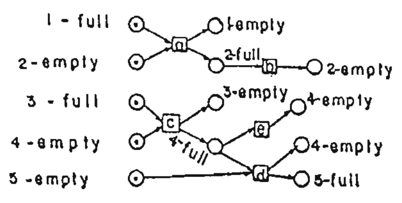


Fig. 17

Claim 3:

(1) The net N^{op} constructed in the above procedure is an occurrence net.

(2) N^{op} is the unique occurrence net which satisfies the axiomatic definition.

Proof. Omited.

Below, we use for the Labeled Event Structure induced by N^{op} and also for the corresponding Behavior Structure the notation $Occ(N, \psi)$

Claim 4 (Consistency).

For each net $<N, \psi>$ over automata: $BS(N, \psi) \equiv_{BS} Occ(N, \psi)$.

Proof. Postponed after Claim 5 below.

2.6. Retrievability

Let us consider in more detail nets over automata. Note, that the uniqueness of robust BS-semantics for such nets (in the sense of claim 1) does not mean that nets with the same MM-semantics have also the same (up to bisimulation) BS-semantics. Recall also the Non-Retrievability Theorem (Claim 5 Section 1.5). Nevertheless, in some cases the full BS-semantics of a net may be still recovered from some partial information as claimed in the following

Claim 5 :

Let $<N_1, \psi_1>$ and $<N_2, \psi_2>$ be nets over automata and B_1, B_2 their robust BS-semantics, and let B be an arbitrary BS. Then

(1) If $Pom(B_1) \equiv_{Pom} Pom(B_2)$ and $M(B_1) \equiv_M M(B_2)$ then $B_1 \equiv_{BS} B_2$.

(2) If N_1 and N_2 are nets over deterministic automata and $Pom(B_1) \equiv_{Pom} Pom(B_2)$ then $B_1 \equiv_{BS} B_2$.

(3) If $Pom(B) \equiv_{Pom} Pom(B_1)$ and $M(B) \equiv_M M(B_1)$ then $B \equiv_{BS} B_1$.

Proof. It is clear that (3) implies (1). Recall that the string language of a Behavior Structure is equal to the linearization of its pomset process, and note that two deterministic Behavior Structures with the same string languages are M-equivalent. Hence, (1) implies (2). Below we give for (3) a proof which only relies on the following facts.

Fact 1. Assume that Behavior Structures (multiautomata or automata) B_1, B_2 are M-bisimular. Then there exists also a special M-bisimulation R between them for which the following holds:

If pRq then there exists in B_1 a path from the root to p and there exists in B_2 a path from its root to q, such that the same strings appear along these paths.

The next two facts are about pomsets in $Pom(B_1)$.

Fact 2. If two pomsets in $Pom(B_1)$ have a common string in their linearization then they are isomorphic.

Fact 3. In $Pom(B_1)$ all the pomsets are not autoconcurrent.

We need two more facts about autoconcurrency:

Fact 4. For two pomsets without autoconcurrency there exist at most one isomorphism between them.

Fact 5. Assume that a M-bisimulation R between two Behavior Structures B_1, B_2 without autoconcurrency relates only nodes with isomorphic pomsets. Then $B_1 \equiv_{BS} B_2$.

Having in mind this facts the proof proceeds as follows: Let R be a M-bisimulation between B and $BS(N, \psi)$. By Fact 1 we may assume that if m, n are related by R then they have paths from their roots which are labeled by the same string. Therefore by Fact 2 and the assumption of the claim under consideration the pomsets at the related nodes are isomorphic. Note that $BS(N, \psi)$ is not autoconcurrent. Hence by assumption of the claim the Behavior Structure B is not autoconcurrent. Therefore the M-bisimulation R between B and $BS(N, \psi)$ satisfies the assumptions of Fact 5. And finally, by Fact 5, the Behavior Structures B and $BS(N, \psi)$ should be BS-bisimular.

Proof of Claim 4 (Sketch).

First we mention without proof the following facts.

Fact 1. The automaton $M(N, \psi)$ is M-bisimular to the Milnering of the Behavior Structure $Occ(N, \psi)$.

Fact 2. $Pom(Occ(N, \psi) \equiv_{pom} Trace(N, \psi)$.

On the other hand, according to the Embedding Inheritance Theorem (Claim 8, Section 1) the automata $M(N, \psi)$ is M-bisimular to the Milnering of $BS(N, \psi)$. Hence,

Fact 3. $Occ(N, \psi)$ and $BS(N, \psi)$ are M-bisimular.

Now, by Claim 2 (this section) $Pom(BS(N, \psi)) \equiv_{Pom} Trace(N, \psi)$. Hence,

Fact 4. $Occ(N, \psi)$ and $BS(N, \psi)$ are Pomset equivalent.

Finally, from Claim 5(3) and facts 3-4 above it follows that $Occ(N, \psi) \equiv_{BS} BS(N, \psi)$.

Comment. The semantics $BS(N, \psi)$ is defined in a compositional way, whereas the semantics $Occ(N, \psi)$ is defined in an operational way. Reviewing the proof of their consistency one could observe that the proof relies on pure operational arguments (for example Facts 1-2) or pure compositional arguments (for example Claim 5) and on the Claim 2, which is main bridge where both operational and compositional arguments are relevant.

3. Appendix (Petri Nets)

Petri Nets present the essence of nondeterminism, asynchrony and concurrency in an illuminating pictorial way. A Petri Net N has *places* (pictured as circles) in which tokens may be located, the current global state of N being just the Cartesian product of its current local states(i.e. of the numbers of tokens in the places). N has also *transitions* (pictured as boxes) which may fire according to specific rules. Though there exists a well elaborated and established system of notations and terminology for Petri Nets [BD], [GR], we mention below only the pure graph component N of the Petri Net. So, in fact referring to a Petri Net N, one has in mind also that to each place $p \in N$ there is assigned a quadruple, called the **kind** of p:

$m(p)$ - the number of input arrows (from boxes to this place)

$n(p)$ - the number of output arrows (from the place to boxes)

$k(p)$ - the initial number of tokens in the place.

$r(p)$ - the capacity of the place (may be infinite).

The firings change the global states via flowing of tokens among places and in this way one can associate with N an automaton M(N) (when only firing of single transitions is allowed) or a multiautomaton MM(N) (when simultaneous firing of multisets of transitions is allowed).

Whereas M(N) and MM(N) reflect the "interleaving" aspects of the behavior of N , much effort went (and is going) into describing causal semantics. According to Petri's view a run of a process should be a partial ordered multiset (pomset) of atomic actions, reflecting the causal relation between action occurrences.

In general each place in a Petri Net appears with a preassigned capacity - the maximal number of tokens it may contain. Actually, in Petri's original model only capacity 1 places were allowed; such nets are usually called C/E Petri nets.

Different definitions of causal semantics were considered in the theory of Petri Nets.

For C/E Petri Nets, in addition to Multiautomaton semantics, Pomset Semantics was also proposed: by Petri [Petri] in terms of causal nets and by Mazurkiewicz [Maz84] in terms of Trace processes. Event Structure semantics was provided in [NPW] and Labeled Configuration Semantics (Prefix Structure Semantics) in [Maz84], where "Prefix Structure" is used as synonym for "Labeled Configuration Structure".

Let N be a C/E Petri net. We use the notations: $PS(N)$ - Prefix Structure semantics; $NPW(N)$ - Event Structure semantics; $Mtrace(N)$ - the Mazurkiewicz Trace semantics; $P(N)$ - Petri's pomset semantics.

For nets N whose places have infinite capacity, a pomset semantics was developed in the Institute GMD: [BD], [GR]. We use for it the notation GMD(N).

Surprisingly, the following questions about Petri Nets were never explicitly considered.

Question 1: (About C/E Petri nets) Are the causal semantics listed above consistent with each other?

Question 2:(About all kinds of Petri nets) Are the causal semantics listed above modular? More precisely, is it the case that whenever a net N is transformed into N_1 through the replacement of a subnet N' of N by a net N'' the following holds:

$$N' =_{sem} N'' \text{ implies } N =_{sem} N_1?$$

3.1. Mazurkiewicz's modular approach to C/E Nets

This approach was originally elaborated in [Maz84] for C/E Petri Nets and amounts to a specific transformation of C/E Petri Nets into Nets over C/E automata (see Section 2.2). To this end with each possible kind $<m, n, k, 1>$ of a place Mazurkiewicz associates a C/E automaton $P(m, n, k, 1)$. The Mazurkiewicz Transform of a C/E Petri net N is the net $<N, \psi>$ over automata with the same bypartite graph as N; its labeling function ψ assigns to each place a C/E automaton which fits the kind of the place in the original Petri Net. The following fact (which is not mentioned explicitly in [Maz84]) holds:

Claim 1. The multiautomaton $MM(N)$ induced (operationally) by the token game for a C/E Petri Net N is MM-bisimular to the multiautomaton $MM(N, \psi)$ which is defined compositionally (via synchronization) for the Mazurkiewicz Transform. Moreover, Mazurkiewicz Transform is the unique assignment of the multiautomata (to kinds) for which each C/E Petri Net and the corresponding net over automata are MM-bisimular.

Note that Nets over C/E automata make sense for arbitrary topology, including such situations which usually are not regarded as legal for Petri Nets. For example an atomic net is deemed "illegal" because in the token game each box is assumed to have at least one precondition and one postcondition. In other words, $MM(N, \phi)$ gives a conservative extension of the operational token game for arbitrary topology.

A reasonable assumption is that whatever the causal semantics for a C/E Petri net N might be, it should be consistent with the multiautomaton semantics produced by the token game and, hence, also with $MM(N, \phi)$, where $<N, \phi>$ is the corresponding

Mazurkiewicz Transform. Finally, since in this case the agents in $<N, \phi>$ happen to be automata, consistency means that the unique robust BS-semantics should be considered.

On the other hand, for the net $<N, \phi>$ over C/E automata one can also consider the semantics $Trace(N, \phi)$ and $Occ(N, \phi)$ as defined in Sections 2.6. They may be easily shown to coincide with Mtrace(N) and NPW(N) for the original C/E Petri Nets. Here is our main result for C/E nets from which in particular there follow affirmative answers on Questions 1,2.

Claim 2. For C/E Nets all the semantics above are consistent with each other and retrievable from BS semantics.
Formally: Let N be a C/E Petri Net and $<N, \phi>$ the corresponding net of automata. Then

(1) $BS(N, \phi) \equiv_{BS} PS(N) \equiv_{BS} NPW(N)$

(2) $Pom(BS(N, \phi)) \equiv_{Pom} Mtrace(N) \equiv_{Pom} P(N)$.

Remarks: NPW(N) and P(N) were originally described in an operational way and were defined only for a restricted class of C/E nets. Two of the restrictions are: (1) the net has no external ports, that is each port is an input port for at least one place and an output port for at least one place; (2) the net should be "contact free". BS, PS and Mtrace semantics are given denotationally and do not impose restrictions on the topology and the initial state of the nets.

The fact that in the case under consideration robust semantics is unique, is a substantial message. Namely, there is not by chance that different definitions of causal semantics turned out to be equivalent (or consistent); as a matter of fact they reflect the genuine and unique causal semantics for C/E Petri nets. As we shall see below the situation is quite different for the more general case of Petri Nets, that are known as Place Transition (P/T) nets.

3.2. P/T Nets

In a P/T Petri net places with arbitrary capacities are allowed. Though the token game becomes a bit more complicated then in the particular case of C/E Petri nets, it still yields a well-defined multiautomaton. In order to generalize Mazurkiewicz's approach to P/T Petri nets one has to start with an appropriate assignment of multiautomata to kinds, along the lines it was done for C/E Petri nets. But the crucial difference is that at this time the P/T assortment consists of P/T Multiautomata, which are not necessarily equivalent to automata. (see the definition of $P(m, n, k, r)$ below). Further, Mazurkiewicz Transforms are defined as in the case of C/E Petri net and the analogue of Claim 1 holds:

Claim 1′. The multiautomaton $MM(N)$ induced (operationally) by the token game for P/T Petri Net N is MM-bisimular to the multiautomaton $MM(N, \psi)$ which is defined compositionally (via synchronization) through the corresponding P/T multiautomata $P(m, n, k, r)$. Moreover, there is a unique assignment of multiautomata to kinds, for which Petri Nets and the corresponding nets over automata are MM-bisimular.

Definition of the P/T multiautomata (agent) $P(m, n, k, r)$ which corresponds to the kind $<m, n, k, r>$ of a place.

(1) A P/T agent with capacity r has states 0,1,2...r.

(2) The set of actions Σ of the agent consists of input actions Σ^{in} and output actions Σ^{out}.

(3) A multiset S of actions leads from a state m to a state k if the number of output actions in S does not exceed m and if k is computed according to the *token game;* in the case of finite capacity r, the sum of m with number of input actions in S should not exceed the capacity r.
Formally: $|S \cap \mu(\Sigma^{out})| \leq m$ and $k = m - |S \cap \mu(\Sigma^{out})| + |S \cap \mu(\Sigma^{in})|$ and $m + |S \cap \mu(\Sigma^{in})| \leq r$.

An agent of capacity r with m input actions, n output actions and initial state k will be denoted by $P(m;n;k;r)$.

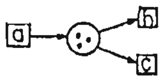


Fig. 18

Graphically the agent is represented by a circle and its ports by boxes. There are arcs from the input ports of the agent to its circle and arcs from the circle to its output ports. The state k is represented by drawing k dots(tokens) in the circle.

Let us address now the issue of causal behavior for P/T Petri nets; here the situation looks quite different from the C/E case. Consider for example a P/T multiautomaton $P(m;n;k;\infty)$. In contrast to the case C/E where an agent is a C/E automaton, there may be many BS which consistently embed a given $P(m;n;k;\infty)$; these BS's may even be not Pomset equivalent. And we are faced with the problem: what BS embedding should be provided for $P(m;n;k;\infty)$? No matter what embedding of $P(m;n;k;\infty)$ into BS will be chosen the BS semantics for nets will be robust. Hence, to choose BS semantics for P/T agents we must take into account considerations which are beyond the multiautomaton description of agent. In particular for a special class of P/T Petri nets (whose places have only infinite capacity) Best and Deviller [BD] proposed two ways how to define Pomset Semantics, an axiomatical one (via causal nets) and an operational one; they proved that both these ways provide the same Pomset Semantics, which we designated as *GMD*(*N*).

We provide a robust BS embedding for the P/T agents which will be consistent with the semantics proposed by Best and Deviller (GMD-semantics) and extends it to nets of arbitrary topology. Moreover this is the unique robust BS embedding which is consistent with GMD-semantics. The consistency between ours and GMD-semantics implies modularity for the latter.

Claim 3: There is a unique robust BS embedding of P/T agents with infinite capacity, which is consistent with GMD-semantics . (this embedding of $P(m;n;k;\infty)$ is denoted by B(m;n;k,∞)).

Below is depicted the Event Structure which corresponds to B(2,2,k,∞) with input ports a,b and output ports c,d.

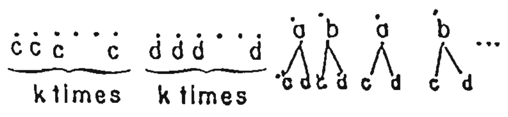


Fig. 19

Its events are of three kinds: (1) minimal but not maximal; there are infinitely many such events and they are labeled by a and b and there is no conflict between them; each such event is followed by two event of the second kind labeled by c and d. (2) maximal but not minimal, labeled by c and d ; for every such event e there is exactly one event less than e. Two such events are in conflict if there is an event of kind (1) less then both of them. (3) events which are maximal and minimal; they are labeled by c and d; there are k labeled by c and k labeled by d and each k+1 of these events are in conflict.

Capacity oriented semantics: Let N be a P/T Petri net and let p be one of its places. The used capacity of p is the maximal number t of tokens which may occur in p in any play of the token game. It may happen that the used capacity t of p is strictly less then the capacity of p. The P/T Petri net N' obtained from N by assigning to the place p any capacity $\geq t$ is easily shown to be MM equivalent to N. A robust BS-semantics is called **capacity oriented** if it is invariant under these transformations. Formally:

A robust BS semantics for nets over P/T agents is said to be capacity oriented iff the following holds: Assume the P/T Petri net N_2 is obtained from the P/T Petri net N_1 by only changing the original capacity assignment by capacities $\geq$ the used capacities. Then $BS(N_1) \equiv_{BS} BS(N_2)$.

Claim 4. There is no capacity oriented robust semantics for nets over P/T agents in which for agents $P(m;n;k;\infty)$ the behavior structure B(m;n;k,∞) is assigned.

The claim shows that GMD-semantics cannot be extended naturally a capacity oriented semantics. But there are many others capacity oriented semantics for P/T nets and for some of them we can provide a natural operational explanation of causality.

The $P(m;n;k;\infty)$ agents are deterministic multiautomata . The following question is still open: let N and N' be two P/T nets with the same GMD-semantics. Do they have the same BS-semantics under the above embedding?

4. History and Concluding Remarks

The idea of combining causality and branching in one entity appeared first in [NPW], where it was used to characterize semantics of C/E Petri Nets. In [NPW] the formalization of this idea is through Event Structures, Occurrence Nets and Prime Algebraic Coherent Posets (actually - Configuration Structures for Event Structures with binary conflict), which are shown to be retrievable from each other (see Claim 3, Section 1). The theory of Event Structures was extensively developed in [W80] and [W87].

A compositional way to define causal semantics for CCS, TCSP, etc. is through operations of the respective repertoire upon objects which take into account branching-causality. In [W], [LG] this was achieved through operations on Event Structures, among which Synchronization is the crucial operation.

Configuration Structures originated a series of notions which reflect in a more direct way the dynamism of processes than the "static" notion of Event Structure. Actually, Prefix Structures [Maz84], Behavior Systems [Shi] and Behavior Structures (this paper) are versions or slight extensions of Configuration Structures.

Our reasons to prefer Behavior Structures (BS) is that they are less restrictive. We have the feeling that synchronization of BS's is more natural and conceptually simpler than Synchronization of Event Structures. One more argument in favor of Behavior Structures has to do with formalization of "Bisimulation between branching-causal processes". Note that this relation was never defined for Event Structures, whereas for Behavior Structures it emerges naturally from the Milner-Park Bisimulation between Automata. Our definition of bisimulation between BS's comes close to the definitions elaborated independently in [GV] and [BC] but is more discriminating, and is shown to be a congruence. [GV] deals with equivalences between Petri Nets and contains also a result, which is similar to our theorem about nonretrievability of a BS (Claim 5, Section 1).

Some implicit hints about Nets of Processes are already in the early works of Milner and Hoare, but the first systematic treatment of the subject is in [Pr]. According to his view on causality, Pratt considers only Nets of Pomset Processes, but clearly the idea works for BS's as well, and we adapted it in the most general setting. In doing so we resigned from specific constructs and notions as utilization, fusion, etc. [Pr] which implicitly assume restrictions (like closedness under augmentation) upon the underlying Processes. Instead we use Synchronization and other standard notions.

Among Nets of Processes we distinguished as a specific area of investigation Nets over Automata and Multiautomata. The reason to do so was to pursue Mazurkiewicz's modular approach to C/E Petri Nets and to clarify how far it can be promoted to other models of concurrency based on net concepts. Incidently in this way we came also to the compositional semantics of P/T Petri Nets, as developed in [Maz88] and [Win]. There are, however, some points where our investigation goes beyond these papers. Namely, our concern is also about the consistency of the compositional approach with the existing operational definitions of interleaving and causal semantics; our results in Sections 2-3 show that consistency holds indeed. We investigate also the question whether compositional semantics is unique or recoverable from partial information. Our results are mainly about nets over automata and there is still much to do for the more general case of nets over multiautomata. Finally, let us note that we use Nets of Processes also in the study of Data flow Networks (see [RT]).

5. Acknowledgements

Both authors are glad to acknowledge the permanent participation of J. Hirshfeld in discussions on the subject. We hope to summarize some outcomes of the discussions about the semantics of Petri Nets in a future joint paper.

A late night conversation of B. Trakhtenbrot with A. Mazurkiewicz (In September 86 at the Conference on Mathematical Logic and its Applications - Druzba, Bulgaria) significantly influenced the choice of the topics investigated in this paper. B.T. benefited from illuminating discussions with C. A. Petri, E. Best, U. Goltz, W. Reisig on the interconnection between Petri Net Theory and other views on Concurrency, during a month stay in the GMD (Bonn).

A. Rabinovich benefited from stimulating discussions with R. Glabbeek and F. Vaandrager during his stay at CWI (Amsterdam).

6. References

[BC] G. Boudol, I. Castelani, On the Semantics of Concurrency: Partial Orders and Transition Systems, in TAPSOFT 87, LNCS 249.

[BD] E. Best, R. Deviller Sequential and Concurrent Behavior in Petri Net Theory. TCS Vol 50

[DDM] P. Degano, R. DeNicola, U. Montanari, A new operational semantics for CCS based on condition event systems. Nota Interna B4-42, Dept. of CS , Univ. Pisa 1986.

[GV] R. Glabbeek, F. Vaandrager, Petri Net Models for Algebraic Theories of Concurrency, Proc. PARLE Conference, Endhoven, LNCS 259, 1987.

[GR] U. Goltz, W. Reisig, The non Sequential Behavior of Petri Nets, Information and Control 57 Nos 2-3 1983.

[Ho] Hoare, C.A.R., Communicating sequential processes, Prentice Hall International, 1985.

[LG] R. Loogen, U. Goltz, A non-interleaing semantic model for nondeterministic Concurrent Processes Tech. Rep. 87-15 Aachen, 1987.

[Maz84] A. Mazurkiewicz, Semantics of Concurrent Systems: A modular fixed point Trace approach. In advances in Petri Nets 1984 LNCS 188.

[Maz88] A. Mazurkiewicz, Compositional Semantics of Pure Place/Transition Systems, in this Issue.

[Mil] Milner, R., A calculi for Synchrony and Asynchrony, TCS 25, 1986

[NPW] M. Nielsen, G. Plotkin, G. Winskel, Petri Nets, Event Structure and Domains, Part 1, TCS vol 13 No 1, 1980.

[Old] E. -R. Olderog, Operational Petri Net Semantics for CCSP. In advances in Petri Nets 1987 LNCS 266 .

[Pet] C. A. Petri, Non-sequential Processes, Tech. Report ISF-77-05, GMD, 1977.

[Pr] Pratt, V.R., The Pomset Model of Parallel Processes: Unifying the Temporal and Spatial, Proc. CMU/SERC workshop on Analysis of Concurrency, LNCS 197, Pittsburgh, 1984.

[RT] Rabinovich, A., Trakhtenbrot, B. A., Nets of Processes and Data Flow, to appear in Proceedings of ReX School, LNCS

[Shi] M. W. Shields, Non Sequential Behavior 1, Internal report CSR-120-82 University of Edinburgh.

[TRH] Trakhtenbrot, B. A., Rabinovich, A., Hirshfeld, J, Nets of Processes, Technical Report 97/88 Tel Aviv Univ. 1988.

[Win] Winkowski, J., Event Structure Representations of the Behavior of Place/Transition Systems, in this Issue.

[W80] G. Winskel, Events in Computations. Ph. D. thesis, CS dept., University of Edinburgh (1980).

[W87] G. Winskel, Event Structures, in Petri Nets: Application and Relationship to Other Models of Concurrency, LNCS 255. 1987.