JOURNAL OF COMPUTER AND SYSTEM SCIENCES 34, 422–461 (1987)

Concurrent Histories: A Basis for Observing Distributed Systems

PIERPAOLO DEGANO AND UGO MONTANARI

Dipartimento di Informatica, Università di Pisa, Pisa, Italy

Received December 31, 1985; revised April 30, 1986

A new notion of transition systems, called *distributed transition systems*, is introduced, where states are sets of processes and transitions specify which processes stay idle. A notion of observations based on partial orderings, called *concurrent histories*, is defined on computations. Several observational equivalences, e.g., bisimulation, are given on observations. As case studies, Petri C/E systems and P/T nets, and Milner's CCS are translated to distributed transition systems.

1. INTRODUCTION

Many models of concurrent and of distributed systems have been proposed in the literature. However, some problems are still under discussion.

One of the issues concerns *interleaving* versus *true concurrency*, i.e., the way in which the temporal/causal ordering of events is described. In the interleaving approach [1, 2, 3, 17, 18], the fact that a set of events may occur concurrently/independently is described by saying that they may occur in any order. Models based on true concurrency [4, 5, 8, 9, 11–16, 21, 22, 24, 25, 27] use instead partial orderings to explicitly describe the temporal/causal relations among events.

When dealing with distributed systems, seen as a collection of spatially distributed processes, we prefer the latter approach mainly because the notion of temporal/causal dependency plays a crucial rôle in defining important properties. It is quite difficult, in fact, to recover causal dependencies in the interleaving models, when needed. As a consequence, the treatment of properties such as fairness or starvation may be awkward.

We think that the above issue cannot be satisfactorily settled unless a precise definition is available of *what and how to observe* out of the computations of a model. Unfortunately, most of the proposed approaches based on true concurrency are inadequate in this respect.

This paper aims at defining a simple operational model, which is basically a transition system, and a flexible notion of observation through which an abstract semantics can be given. A transition system should also be definable through a rewriting system. Moreover, we want to be able to distinguish between *temporal* and *causal* dependencies. In other words, we wish to observe, in a *single* com-

0022-0000/87 $3.00

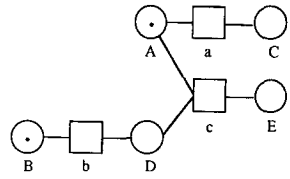

FIG. 1.1. A place/transition (P/T) net showing confusion.

putation, the causal dependencies among the actions performed by the various processes, regardless of the possibility that these actions may have to occur in other orders in different computations.

A classical problem showing the need for the above distinction regards the so-called confusion problem in Petri nets. Let us have the place/transition net depicted in Fig. 1.1., and assume that we observe, for every event, not only the occurred transition, but also the refused transitions, i.e., those in conflict with it. In Fig. 1.2. we have the possible observations as defined in this paper. In both Figs. 1.2a, b, the two events are causally independent, but temporally dependent. For instance, in Fig. 1.2a, the event on the left can only occur *after* the event on the right, since otherwise the refused transition (c) would not be enabled. A similar situation of causally independent, but temporally dependent events may also arise when observations of non-contact-free condition/event system are considered.

Causal and temporal dependencies coincide in a class of distributed systems that we call *completely concurrent*. Here, if a causal partial ordering of events is observed from a computation, not only its events are generated in a total temporal ordering which is *sound*, i.e., compatible with the causal one (which is always the case), but they can also be generated (by other computations) in *all* temporal orderings which are compatible with the causal one, i.e., the transition system is *completely concurrent*. Both contact-free condition/event (C/E) systems and place/transition (P/T) nets are completely concurrent; in this case, our observations essentially coincide with Petri non-sequential processes. However, we have a richer notion of observational equivalence, which provides a flexible tool for defining abstract semantics.

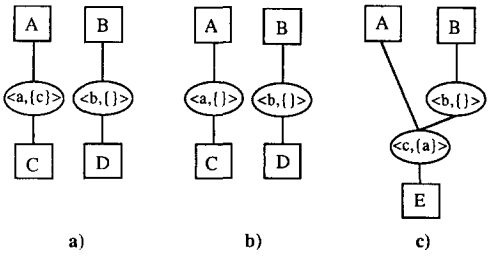

FIG. 1.2. The observations for the three computations of the P/T net in Fig. 1.1. The drawing conventions are: partial orderings are represented through their Hasse diagrams growing downwards; the minimal and maximal sets of boxes represent the initial and final distributed states; and the circles represent events.

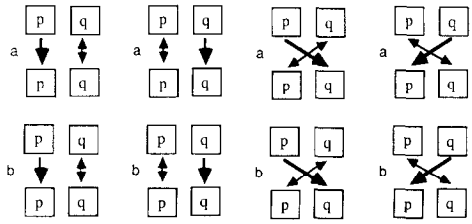

FIG. 1.3. The transitions of a distributed transition system with a single state consisting of two processes, p and q. Four of the transitions are labelled by a and four by b; the processes are represented as boxes; pairs of processes related by bijections are connected by a double thin arrow; and a single thick arrow connects processes transformed by the transition.

In our transition systems, called *distributed transition systems*, states are *non-intersecting set* of process states (processes for short). A transition specifies, through a partial bijection, those processes which stay *idle* in the transition itself; the remaining processes are transformed by it. A computation is a finite or infinite sequence of both states and transitions. From a computation we define both an *interleaving observation*, which simply consists of the initial and final (if any) states and of the sequence of transition labels; and a *partial ordering observation*, called *concurrent history*. The partial ordering is defined on the processes in the initial and final (if any) states and on the transition occurrences: two transitions are related if the second uses, directly or transitively, some of the processes generated by the first. In Fig. 1.3 we see the eight transitions of a transition system with a single state consisting of two processes, p and q. Four of the transitions are labelled by a and four by b.

In Fig. 1.4 three infinite computations and their partial ordering observations are depicted. The computation in (a) is observed as the concurrent history in (b); the computations in (c) and (e) are both observed as the concurrent history in (d). When considering interleaving observations instead, both computations in (a) and (c) are observed as $\{p, q\}\ (ab)^{\infty}$, while the computation in (e) is observed as

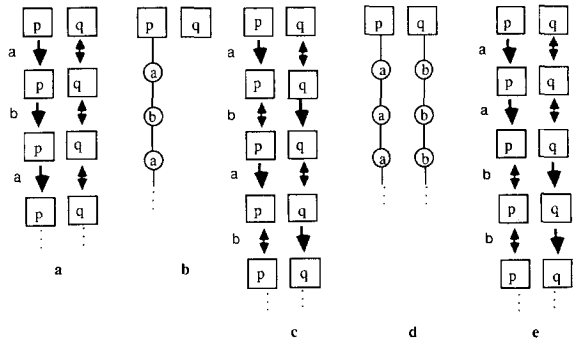

FIG. 1.4. Three infinite computations, in (a), (c), and (e); and two concurrent histories, in (b) and (d).

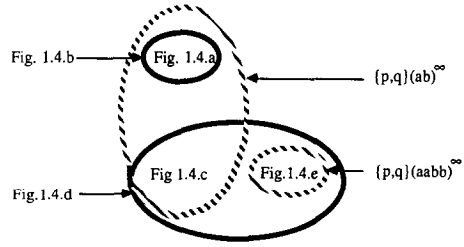

FIG. 1.5. Interleaving and partial ordering observations are incomparable. The computations included in dotted (continuous) lines have the same interleaving (partial ordering) observations.

$\{p, q\}(aabb)^{\infty}$ (see Fig. 1.5). Note that the two observational approaches are incomparable, i.e., neither is finer than the other: the computation itself provides the initial observation. The notion of process identifying transition, as opposed to identifications obtained by allowing intersecting states, is essential to define such a notion of computation.

Our notion of computation has no couterpart in P/T net theory, where firing sequences and Petri non-sequential processes play the rôle of our interleaving and partial ordering observations. Note also that the states of our transition systems are essentially markings, and thus do not directly embody any notion of individual token. The P/T net corresponding to the transition system of Fig. 1.3 is reported in Fig. 5.3.

Also Milner's Calculus for Communicating Systems (CCS) can be translated in our formalism and given a partial ordering semantics. The resulting transition given is completely concurrent, as well. For instance, Fig. 1.6a shows the observation of a computation of the CCS term $(E_0|E_1|E_2)\backslash\alpha$, where

$$E_0 = \alpha E_0 + \beta E_0;$$
$$E_1 = \alpha E_1 + \gamma E_1;$$
$$E_2 = \alpha^{-} E_2.$$

In Fig. 1.6b the same computation is observed, but without the events corresponding to invisible actions. We prove that the interleaving semantics of two

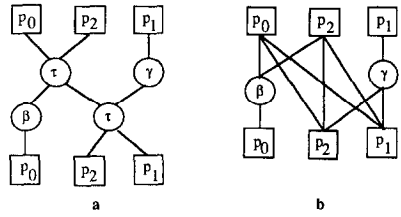

FIG. 1.6. Two observations of a computation of the CCS term $(E_0|E_1|E_2)\backslash\alpha$, where $E_0 = \alpha E_0 + \beta E_0$; $E_1 = \alpha E_1 + \gamma E_1$; $E_2 = \alpha^{-} E_2$.

distributed transition systems derived from two CCS terms is the same up to bisimulation, according to our definition, if and only if the two terms are observationally equivalent, according to Milner.

In this paper we explictly address fairness issues. It is easy to define many notions of fairness for the infinite computations of our transition systems. Besides the classical notions of weak and strong fairness [20], which we call *local* weak and *local* strong since they involve a *single* process, we also have the notions of *global* weak and *global* strong fairness, following [8]. In a globally (weakly/strongly) unfair computation, the same set of processes, which can proceed (almost always/infinitely often) all together, is discriminated forever. In other words, global fairness assures that a processor is made available to any set of processes requiring it. We think that global fairness is an important property of distributed systems; it should be firmly requested. Instead, having local fairness is a matter of the particular application.

In this paper we show that none of the above fairness properties is observable in the interleaving approach. This is true in the sense that both fair and unfair computations may be given the same observation, as shown by the computations in Figs. 1.4a (unfair) and 1.4c (fair) which have the same observation $\{p, q\}(ab)^{\infty}$.

Also in the partial ordering approach local fairness, both weak and strong, is not observable. On the other hand, global fairness of distributed transition systems derived from rewriting systems is observable and in this case global weak and global strong fairness coincide. This result strengthens our believe that global fairness is a basic notion.

The paper is organized as follows. Section 2 gives some simple notions on transition systems and defines our observation devices, called nondeterministic measurement systems (NMSs for short). A NMS is the tree of the nondeterministic computations (ordered by prefix) of a given transition system, the nodes of which are labelled using an *observation function*. The label of a node reports what is observed on the system up to that point. Several equivalence notions are defined on NMSs, among which is bisimulation [19].

Section 3 defines the model of *distributed transition systems*. Abstracting out from computations we get our basic partial ordering observations, namely *concurrent histories*. In the same section we also define our notion of complete concurrency and prove sufficient a sort of commutativity condition. Complete concurrency is characterized by the fact that the interleavings of the partial ordering observations are exactly the interleaving observations.

In Section 4 rewriting systems are introduced; the commutativity property above is proved to hold; and global fairness is shown to be observable through concurrent histories.

As a first case study, Section 5 translates C/E systems to distributed transition systems and P/T nets into *finite* rewriting systems and shows that our partial ordering observations essentially coincide with Petri non-sequential processes, whenever these are defined.

Section 6 examines Milner's CCS and gives a partial ordering semantics for it.

This paper was motivated by the need for formalizing the ideas contained in [10]. The original results of the paper concern Sections 3, 4, and 5; a first version of concurrent histories is in [8, 9]; most of the results reported in Sections 2 and 6 are taken from [6, 7].

2. Transition Systems and Equivalences

We first give some simple notions about transition systems.

Definition 2.1. A **transition system** is a quadruple $\langle Q, T, c, q_0 \rangle$, where

- Q is a countable set of **states**;
- T is a countable set of **transitions**;
- $c: T \to Q \times Q$ is a function giving for every transition the **initial** and the **final** states;
- $q_0 \in Q$ is the **initial** state.

A **computation** ξ is a finite or infinite path (understood as a sequence of *occurrences* of both states and transitions) starting form q_0.

A transition system is a purely operational, intensional model. To make transition systems more extensional, we may simply define an equivalence relation on them. To this purpose, the information present in a state is often insufficient, typically for concurrent systems, where also the interactions with other systems must be considered. Thus it is convenient to derive from a transition system a different device, called Nondeterministic Measurement System (NMS).

Definition 2.2. Given a transition system $\mathbf{T} = \langle Q, T, c, q_0 \rangle$, an **observation function** o is a partial function from the computations of $\mathbf{T}$ to a set D, called **observations**.

Definition 2.3. A **NMS** is a tree (possibly with limit points on its infinite branches) whose nodes are labelled by observations in D.

Given a NMS, its **opened** NMS is obtained by erasing all the nodes at infinite depth, i.e., its limit points, if any.

A **subtree** of a NMS t is a NMS t' consisting of a node of t, together with all its descendents. Sometimes we identify a subtree with its root.

Given a transition system, a corresponding NMS can be obtained by deciding what to observe in any computation.

Definition 2.4. (*from transition systems to NMSs*). Given a transition system $\mathbf{T} = \langle Q, T, c, q_0 \rangle$ and an observation function o from the computations of $\mathbf{T}$ to a set D, the **derived** NMS t is obtained by unfolding $\mathbf{T}$ starting from q_0 and labelling its nodes using o.

Note that NMSs derived from transition systems may have an uncountable number of nodes.

Here, the purpose of the observation function is to abstract out the relevant information from the whole computation, and not only from the last state.

One of the uses of function o might be to define a two-step operational semantics, filtering out only *some* (e.g., unfair) computations.

EXAMPLE 2.1. (*fairness*). Let $\mathbf{T} = \langle \{q\}, \{a, b\}, c, q \rangle$, $c(a) = c(b) = \langle q, q \rangle$, and let $o(\xi)$ be "undefined" if ξ is infinite and one transition is used only a finite number of times (i.e., ξ is unfair), $o(\xi) = \xi$ otherwise.

DEFINITION 2.5. Given a transition system $\mathbf{T} = \langle Q, T, c, q_0 \rangle$, the **free** derived NMS is the derived NMS of $\mathbf{T}$ with $o(\xi) = q_0 \omega q$, where ω is the sequence of the transitions occurring in ξ and q is the final state of ξ, if any.

DEFINITION 2.6. Given a transition system $\mathbf{T} = \langle Q, T, c, q_0 \rangle$, two countable sets A and R, and an **abstraction** partial function abstr

$$\text{abstr}: (T \multimap A) \cup (Q \multimap R)$$

the **abstract** derived NMS is the free derived NMS of $\mathbf{T}$, where abstr(ω) is substituted for every observation ω (understanding abstr as homomorphically extended to sequences).

EXAMPLE 2.2 (*hiding*). Let us consider a transition system and a total abstraction function abstr, from T to an alphabet A containing the symbol τ. Define the partial function abstr′ as abstr, but forgetting all the τ's. In this way, a node at infinite depth may get a finite label.

Note in the example above that, if in the given transition system there is a finite number of transitions from every node, and no terminal state, the derived opened NMS is a generating tree in the sense of Smyth ([23], where D is a domain).

Different observation functions permit the observation of different properties of the behavior of transition systems.

DEFINITION 2.7 (*observable property*). Given a transition system $\mathbf{T} = \langle Q, T, c, q_0 \rangle$ and an abstraction function abstr, let t be the abstract derived NMS and let o be the corresponding observation function. A property of the computations of $\mathbf{T}$, defined by a predicate P, is **observable** (through t), if for every pair of computations ξ_1 and ξ_2

$$o(\xi_1) = o(\xi_2) \qquad \text{implies} \qquad P(\xi_1) = P(\xi_2).$$

DEFINITION 2.8 (*comparing two NMSs*). Given a transition system $\mathbf{T} = \langle Q, T, c, q_0 \rangle$ and two observation functions o and o', let t and t' be the derived

NMSs. We say that t is **finer** than t' and that t' is **coarser** than t iff for every pair of computations ξ_1 and ξ_2

$$o(\xi_1) = o(\xi_2) \qquad \text{implies} \qquad o'(\xi_1) = o'(\xi_2).$$

It t is neither finer nor coarser than t', we call them **incomparable**; if t is both finer and coarser than t', we call them **similar**.

The next step addresses the definition of equivalence relations on NMSs having the same set of observations. We examine only three of them.

DEFINITION 2.9. Two NMSs are (**finite-**) **structurally** equivalent iff their (opened) NMSs are isomorphic.

DEFINITION 2.10 (*bisimulation equivalence*). 1. Let Φ be a function from relations to relations on opened NMSs defined as follows:

$\Phi(\text{Rel}) = \{\langle t, u\rangle |$ (i) the labels of the roots of t and u are the same;

(ii) for every subtree t' of t there exists a subtree u' of u such that $\langle t', u'\rangle \in \text{Rel}$;

(iii) for every subtree u' of u there exists a subtree t' of t such that $\langle t', u'\rangle \in \text{Rel}\}$;

2. $\approx = \cup \{\text{Rel} | \text{Rel} \subseteq \Phi(\text{Rel})\}$;

3. two NMSs are **bisimulation** equivalent iff the pair of their opened NMSs is in $\approx$.

PROPOSITION 2.1. *We have that*

- *Φ is monotonic on the lattice of binary relations under inclusion.*
- *$\approx$ is an equivalence relation.*

DEFINITION 2.11. Two NMSs are (**finite-**) **result** equivalent iff the sets of the labels of their leaves (at finite depth) coincide.

PROPOSITION 2.2. *Finite-structural equivalence implies bisimulation equivalence implies finite-result equivalence.*

DEFINITION 2.12. Given two observation functions o and o', two transition systems are (**finite-**) **structurally/bisimulation/(finite-) result equivalent** iff their derived NMSs (w.r.t. o and o') are (finite-) structurally/bisimulation/(finite-) result equivalent.

Of course, many other equivalence relations can be defined. We have considered the above three because they have a direct counterpart in the literature. For instance, if we consider Example 2.2, two transition systems are finite-structurally

equivalent iff they have the same synchronization tree [26]; they are bisimulation equivalent if they are observationally equivalent [17]; and they are finite-result equivalent if they recognize the same language (here we consider the terminal states of the transition system as accepting states).

DEFINITION 2.13 (*observational semantics*). Given a transition system $\mathbf{T} = \langle Q, T, c, q_0 \rangle$, an abstraction function abstr, and an equivalence relation $\equiv$ on abstract derived NMSs, the **semantics** of **T** is its abstract derived NMS defined up to $\equiv$.

3. DISTRIBUTED TRANSITION SYSTEMS

In this section we define a class of transition systems and several NMSs suitable for modelling concurrent distributed systems.

DEFINITION 3.1. A **distributed transition system** is a quintuple $\langle Q, T, c, \mathrm{Q}_0, 1 \rangle$, where

- $\langle Q, T, c, \mathrm{Q}_0 \rangle$ is a transition system;
- the states in Q are finite, pairwise disjoint, sets of **processes**;
- function 1 labels a transition $u \in T$ with $c(u) = \langle \mathrm{P}, \mathrm{Q} \rangle$ by a partial injective function

$$1(u) = i\colon \mathrm{P} \multimap \mathrm{Q}.$$

The elements of P and Q related by i are called **idles** of u, the non-idle elements of P and Q are called **heads** and **tails** of u, respectively.

We use the word *process* to express in short what could be better called *process state.*

Figure 3.1 depicts the transition u labelled by $i(p_k) = p'_k$, $k = 1, 2$, with

$$c(u) = \langle \mathrm{P}, \mathrm{Q} \rangle = \langle \{p_1, p_2, p_3\}, \{p'_1, p'_2, p_4, p_5\} \rangle.$$

The heads and the tails of u are $\{p_3\}$ and $\{p_4, p_5\}$; the idles of u in P and in Q are $\{p_1, p_2\}$ and $\{p'_1, p'_2\}$. Processes are represented as boxes; pairs related by i are connected by double thin arrows; a single thick arrow relates heads to tails.

Operationally speaking, the occurrence of a transition instance from P to Q consists first of splitting P in two disjoint sets, the heads and the idles. State Q is then

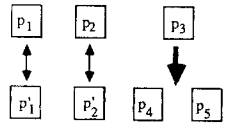

FIG. 3.1. A transition.

obtained by joining the tails and the related idles. The relation between heads and tails can be seen as a *causal dependency* relation, in the sense that the heads are the input of a step producing as output the tails. Idles are then those processes not involved in the step. Therefore, we will treat partial function i as *identifying* in a computation, but *not* in the transition system itself, the related subsets of idles in P and Q.

We now carry over the distributed transition systems the semantic notions defined for transition systems.

DEFINITION 3.2. Given a distributed transition system $\mathbf{T} = \langle Q, T, c, \mathrm{Q}_0, 1 \rangle$,

- the **free interleaving** NMS of **T** is the free NMS of
- given an abstraction function abstr, the **abstract interleaving** NMS of **T** is the abstract NMS of
- given an abstraction function abstr and an equivalence relation on abstract derived NMSs, the **interleaving semantics** of **T** is the semantics of

the transition system $\langle Q, T, c, \mathrm{Q}_0 \rangle$.

We now examine some notions which apply to computations of distributed transition systems.

DEFINITION 3.3. An infinite computation ξ of a given distributed transition system $\langle Q, T, c, \mathrm{Q}_0 \rangle$ is called

- **locally weakly (strongly) fair** iff no process (up to identifications) occurs, from some point onwards, in every state of ξ and is almost always (infinitely often) head of a transition in T;
- **globally weakly (strongly) fair** iff no set of processes (up to identifications) occurs, from some point onwards, in every state of ξ and is almost always (infinitely often) the set of the heads of a transition in T.

The standard notions of weak and strong fairness make reference to a single process and thus coincide with out *local* notions.

Unfortunately, none of the notions of fairness defined above is observable when the interleaving approach is used. This is shown by the following property, where the abstraction function reports the fact that a transition occurred. This is all that is needed to talk about fairness.

PROPERTY 3.1 (fairness is not observable in the interleaving approach). *There exists a distributed transition system* $\mathbf{T} = \langle Q, T, c, \mathrm{Q}_0, 1 \rangle$, *such that no fairness property above is observable* (*see Definition* 2.7.) *through the abstract interleaving NMS with respect to the abstraction function abstr defined as*

$$\mathrm{abstr}(u) = \text{“}\textit{occurred}\text{,”} \quad u \in T; \qquad \mathrm{abstr}(\mathrm{Q}) = \mathrm{Q}, \textit{ for all } \mathrm{Q} \in Q.$$

Proof. Let $\mathbf{T} = \langle Q, T, c, \mathrm{Q}_0, 1 \rangle$, where

- $Q = \{\{p_1, p_2\}\}$;
- $T = \{u_1, u_2\}$;
- $c(u_1) = c(u_2) = \langle \{p_1, p_2\}, \{p_1, p_2\} \rangle$;
- $\mathrm{Q}_0 = \{p_1, p_2\}$;
- $1(u_1)(p_1) = p_1$, $1(u_2)(p_2) = p_2$.

Here both p_1 and p_2 can proceed independently. It is easy to see that the computation where only p_1 moves forever has the same observation (i.e., Q_0 followed by the infinite sequence of "*occurred*") as the computation where both processes move alternately. ▮

Furhter results on fairness will be given in Section 4. We now introduce our notion of partial ordering observations, which we call **concurrent histories**.

DEFINITION 3.4 (*causal independence of adjacent transitions*). Given a distributed transition system,

- a transition u_1 is **adjacent** to a transition u_2 iff the final state of u_1 is the initial state of u_2;
- given two adjacent transitions u_1 and u_2, u_2 is **independent** of u_1 iff the idles of u_1 contain, via identifications, the heads of u_2.

PROPERTY 3.2. *Given a distributed transition system and a pair of adjacent transitions u_1 and u_2, u_2 independent from u_1, it is possible to partition, by composing the identifications i_1 and i_2, the initial state of u_1 (the final state of u_2) in*

- *the heads (tails) of u_1,*
- *the heads (tails) of u_2, and*
- *the idles of both transitions.*

DEFINITION 3.5 (*the partial ordering of causal dependency relation* $\leqslant$). Given a computation $\xi = \langle \mathrm{Q}_0 u_0 \mathrm{Q}_1 u_1 \cdots \}$ consider as identified all the processes which are related by the partial functions $i_0, i_1, \ldots$.

The **leaves** of ξ are either the processes in Q_n, $n \geqslant 0$, if u_{n-1} is the last transition of the computation, or the idle processes which stay idle from some transition onwards, if the computation is infinite. We partition the processes of Q_0 and the leaves of ξ in the following three classes, and call

- **heads** of ξ the processes in Q_0 which are not leaves;
- **tails** of ξ the leaves which are not in Q_0;
- **idles** of ξ the leaves which are in Q_0.

Then we define the following relation R on transition occurrences, heads, tails, and idles of ξ:

- pRu, if head p occurs as head of u;
- $u_i R u_j$, if $i<j$ and there exists a process being both tail of u_i and head of u_j;
- uRp, if tail p occurs as tail of u.

Finally, let $\leqslant$ of ξ be the partial ordering defined as the transitive reflexive closure of R.

Intuitively speaking, through the partial ordering $\leqslant$ we intend to represent the **causal** dependencies among different transition occurrences: if $u_1 \leqslant u_2$, transition u_2 uses, directly or transitively, some processes generated by u_1.

We now define our domain of observations as consisting of concurrent histories.

DEFINITION 3.6. The partially ordered set $\langle I, \leqslant \rangle$ is **finitely preceded** iff $\forall i \in I$, the set of all its predecessors is finite.

DEFINITION 3.7. Let A be a countable set of **event labels** and E be a countable set of **process labels**. Sets A and E are disjoint.

DEFINITION 3.8. A **concurrent history**, or **history** for short, $h \in H$ is a triple $\langle S, 1, \leqslant \rangle$, where

- S is a set of **subsystems**;
- $1: S \to A \cup E$ is a **labelling function**;
- $\leqslant$ is a partial ordering relation on S, called **causal relation**.

The subsystems with labels in A are called **events**, those with labels in E are called process occurrences, or, once more, simply **processes**.

We require that the processes always be minimal or maximal in $\leqslant$, and that the pair $\langle S, \leqslant \rangle$ be finitely preceded.

The processes which are minimal, but not maximal are called **heads**, those which are maximal, but not minimal are called **tails**, and those which are both minimal and maximal are called **idles**. Thus we partition processes into heads, tails, and idles.

Two subsystems s_1 and s_2 are **concurrent** if neither $s_1 \leqslant s_2$ nor $s_2 \leqslant s_1$.

Two histories will be identified if isomorphic, i.e., if there is a label- and order-preserving bijection between their subsystem.

In Fig. 3.2 we see two histories h_1 and h_2 (in parts (a) and (b)), with $A = \{a, b, c\}$ and $E = \{A, B, C\}$. The causal relations are depicted through their Hasses diagrams, growing downwards. Processes (resp. events) are represented as boxes (circles). History h_1 has a single head (labelled by A), three tails (one labelled by A and two by B), and no idle. Note that the tail labelled by A is concurrent with the other tails and with the greatest event, which in turn causes both tails labelled by B.

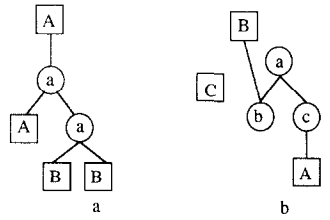

FIG. 3.2. Two concurrent histories.

History h_2 has one head, one tail, and one idle (labelled by B, A, and C, respectively).

DEFINITION 3.9 (*from a distributed transition system to its free NMS*) Given a distributed transition system $\langle Q, T, c, Q_0, 1 \rangle$, the **free (partial ordering)** derived NMS is the derived NMS of the transition system $\mathbf{T} = \langle Q, T, c, Q_0 \rangle$ where:

- the observations are histories, with the transitions in T as event labels and the processes in the states of Q as process labels;
- function $o(\xi)$ yields the history having as heads, tails, idles, events and $\leqslant$ the heads, tails, idles, transition occurrences and $\leqslant$ of ξ; the label of a process occurrence (event) is the process (transition) in the original transition system. ▮

PROPERTY 3.3 (constructing histories from computations). *More operationally, history $o(\xi)$ can be obtained by the following procedure*:

1. *for every occurrence (with heads* P *and tails* Q*) of a transition u in computation ξ, generate an event e labelled by u and let* $P \leqslant e \leqslant Q$ *(understanding e larger/smaller than all processes in* P/Q*);*
2. *in computation ξ, identify all the processes mapped by functions i in the transitions*;
3. *close reflexively and transitively $\leqslant$*;
4. *erase all the processes which are neither minimal nor maximal in $\leqslant$.*

PROPERTY 3.4 (soundness). *Given a computation ξ, the total ordering on its transition occurrences is sound, i.e., it is compatible with (i.e., larger than or equal to, in the set-theoretical sense) the partial ordering $\leqslant$ on the events of the concurrent history $o(\xi)$ (seen as transition occurrences).*

Proof. According to Definition 3.5., $u_i \leqslant u_j$ implies $i \leqslant j$. ▮

In Figure 3.3a we see a computation ξ with three transitions u_0, u_1, u_2; in (b) the result of applying steps 1–3 above; and in (c) the history observed from ξ.

The free derived NMS corresponds to the most complete observation of the *causal structure* which, in our view, is possible for a distributed transition system. In fact, we abstracted only out of the total temporal ordering on the sequence of trans-

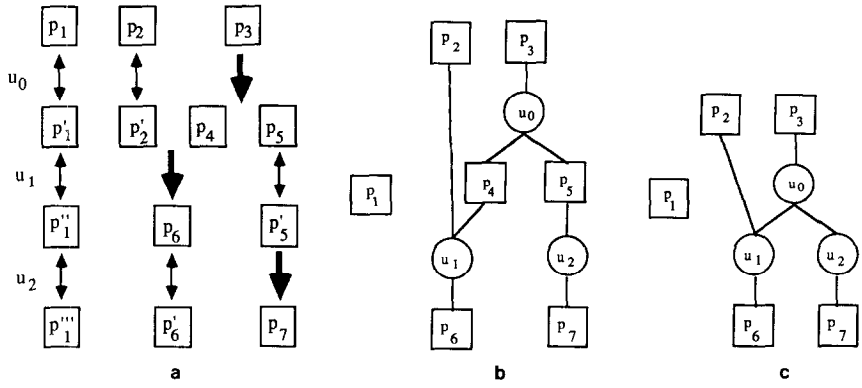

FIG. 3.3a. A computation. (b) An intermediate step in getting the observation. (c) The history observed.

itions, but we kept all the information about which transitions have occurred and about the causal dependencies among them. More abstract NMSs can be defined.

DEFINITION 3.10 (*abstract history*). Given a concurrent history $h = \langle S, 1, \leqslant \rangle$ with labels in $A \cup E$, two alphabets A' and E', and an **abstraction** partial function

$$\text{abstr}: (A \multimap A') \cup (E \multimap E'),$$

the **abstract** history (w.r.t. abstr) is $h' = \langle S', 1', \leqslant' \rangle$, where

- $S' = \{s | \text{abstr}(1(s)) \text{ is defined}\}$;
- $\leqslant'$ is the restriction of $\leqslant$ to S';
- $1'(s) = \text{abstr}(1(s))$.

Whenever not explicitly stated, we assume abstr to be total and, when restricted on E, to be the identity.

DEFINITION 3.11 (*abstract NMS*). Given a distributed transition system, having t as its free derived NMS, and an abstraction partial function abstr, the **abstract (partial ordering)** derived NMS (with respect to abstr) is obtained by replacing every history h labelling a node of t with the abstract history h' w.r.t. abstr.

DEFINITION 3.12 (*semantics*). Given a distributed system **T**, an abstraction function abstr, and an equivalence relation $\equiv$ on abstract derived NMSs, the **semantics** of **T** is its abstract derived NMS defined up to $\equiv$.

So far, we have studied properties of single computations. We now consider all the computations of a distributed transition system.

DEFINITION 3.13 (*complete concurrency*). Given a distributed transition system

$\mathbf{T} = \langle Q, T, c, Q_0, 1 \rangle$, an abstraction partial function abstr and a concurrent history $h = \langle S, 1, \leqslant \rangle$ generated by a computation, let Ξ be the set of all the computations of $\mathbf{T}$ having h as an observation in the abstract derived NMS with respect to abstr.

A distributed transition system $\mathbf{T}$ is called **completely concurrent** with respect to abstr iff, for all h, partial ordering $\leqslant$ is the intersection (in set-theoretical sense) of all the finitely preceded total orderings imposed by the computations in Ξ on the events of h (seen as transition occurrences).

Note that the total orderings imposed by the computations in Ξ on the events of S are *always* sound, i.e., finitely preceded total orderings compatible with $\leqslant$, by Property 3.4. The inverse property is imposed by our intuition, in order to give full meaning to observations of *asynchronous concurrent* systems, where the causal dependencies coincide with the *mandatory* temporal dependencies, i.e., the intersection of the temporal dependencies in all affine computations. Thus, according to our definition, only the *completely* concurrent systems are *truly* asynchronously concurrent.

An important property of completely concurrent distributed systems follows.

THEOREM 3.1 (complete concurrency implies that the interleavings of the partial ordering observations are the interleaving observations). *Given a completely concurrent distributed transition system w.r.t. an abstraction function abstr, let t and t' be its abstract interleaving and abstract partial ordering NMSs w.r.t. abstr:*

(i) *Given an observation* $Q_0 \omega Q$ *in t, there exists a history* $h = \langle S, 1, \leqslant \rangle$ *in t' where*

- Q_0 *is the set of the heads and the idles;*
- Q, *if any, is the set of the tails and the idles;*
- ω *is a total ordering compatible with* $\leqslant$ *restricted on the events.*

(ii) *Given a history* $h = \langle S, 1, \leqslant \rangle$ *in t', for every finitely preceded total ordering* ω *on the events compatible with* $\leqslant$, *there exists an observation* $Q_0 \omega Q$ *in t such that*

- Q_0 *is the set of the heads and the idles;*
- Q, *if h is finite, is the set of the tails and the idles.*

Proof. Immediate, from Definition 3.13. ∎

Complete concurrency is guaranteed by a sort of commutativity condition defined below, the sufficiency of which is proved by Lemmata 3.1 and 3.2 and Theorem 3.3.

DEFINITION 3.14 (*commutativity*). Given a distributed transition system $\mathbf{T} = \langle Q, T, c, Q_0, 1 \rangle$ we say that $\mathbf{T}$ is commutative with respect to an abstraction partial function if for every pair of adjacent transitions u_1 and u_2 from, say, P to Q and from Q to R, u_2 independent of u_1 (see Definition 3.4.), there exists a pair of different transitions u_2' and u_1' from P to Q′ to R, u_1' independent from u_2' and

- the heads and the tails of u_i and u'_i, $i = 1, 2$, are the same (up to identifications);
- $\text{abstr}(u_1) = \text{abstr}(u'_1)$ and $\text{abstr}(u_2) = \text{abstr}(u'_2)$.

THEOREM 3.2 (commutativity implies global weak = global strong fairness) *Given a distributed transition system* $\mathbf{T} = \langle Q, T, c, Q_0, 1 \rangle$ *commutative with respect to the abstraction function mapping every transition in* "occurred," *a computation is globally weakly fair (see Definition* 3.3) *iff it is globally strongly fair.*

Proof. If a set of processes appears in two succesive states of a computation and is the set of the heads of a transition in the second, so it is in the first, due to commutativity. ▮

Since the two notions of weak and strong global fairness coincide, when considering a commutative transition system, in the sequel we will use simply the term "global fairness."

LEMMA 3.1 (getting the same history by switching two consecutive independent transitions when the abstraction function is total). *Given:*

- *a distributed transition system* $\mathbf{T} = \langle Q, T, c, Q_0, 1 \rangle$ *commutative with respect to an abstraction* total *function* abstr;
- *a computation* $\xi = \{Q_0 u_0 Q_1 u_1 \cdots Q_k u_k Q_{k+1} u_{k+1} Q_{k+2} \cdots\}$;
- *the history* $h = \langle S, 1, \leqslant \rangle$ *labelling the node corresponding to* ξ *in the abstract NMS with respect to abstr; and*
- *two consecutive transition occurrences* u_k *and* u_{k+1} *of* ξ *corresponding to two concurrent events* e' *and* e'' *of* h, *respectively,*

there exist two transitions u'_k *and* u'_{k+1} *such that* $\xi' = \{Q_0 u_0 Q_1 u_1 \cdots Q_k u'_k Q'_{k+1} u'_{k+1} Q_{k+2} \cdots\}$ *is a computation of* $\mathbf{T}$ *generating the same* h *(i.e., a history isomorphic to* h*), where* u'_k *and* u'_{k+1} *originate* e'' *and* e', *respectively.*

Proof. Transition u_{k+1} is independent from u_k, because e' and e'' are concurrent and thus the tails of u_k and the heads of u_{k+1} in Q_{k+1} are disjoint. By hypothesis, $\mathbf{T}$ is commutative, therefore there exists a pair of transitions u'_k and u'_{k+1} (u'_{k+1} independent from u'_k) such that $\xi = \{Q_0 u_0 Q_1 u_1 \cdots Q_k u'_k Q'_{k+1} u'_{k+1} Q_{k+2} \cdots\}$ is a computation of T. Finally, the history h' obtained from ξ' through abstr is (isomorphic to) h, because we have $\text{abstr}(u_k) = \text{abstr}(u'_{k+1})$, $\text{abstr}(u_{k+1}) = \text{abstr}(u'_k)$, and the causal relation of u_k (u_{k+1}) with the other transition instances is the same as of u'_{k+1} (u'_k), since the heads and the tails of u_k (u_{k+1}) and u'_{k+1} (u'_k) are the same. ▮

LEMMA 3.2 (commutative implies completely concurrent when abstr is total). *A distributed transition system* $\mathbf{T} = \langle Q, T, c, Q_0, 1 \rangle$ *commutative with respect to an abstraction* total *function abstr is* completely concurrent *with respect to abstr.*

Proof. Let Ξ be the set of all its computations having the same concurrent history $\langle S, 1, \leqslant \rangle$ as observation in the abstract derived NMS. We have to prove that the total orderings imposed by the computations in Ξ on the events of S (seen as transition occurrences) are exactly the finitely preceded total orderings compatible with $\leqslant$:

(i) *The total orderings imposed by computations in Ξ are all sound, i.e., finitely preceded total orderings compatible with $\leqslant$ (see Property* 3.4).

(ii) *Every finitely preceded total ordering compatible with $\leqslant$ is induced by a computation.*

The history $h = \langle S, 1, \leqslant \rangle$ and a finitely preceded total ordering $O = \{e_0, e_1, \ldots\}$ on its events compatible with $\leqslant$ are given, and we must find a computation ξ with observation h and inducing O. Let ξ^0 be any computation in Ξ. We construct a sequence of computations $\{\xi^0, \xi^1, \ldots\}$ all with observation h, as follows. Assume that $\xi^{j_n} \in \Xi$ induces an ordering O^{j_n}. If $O^{j_n} = O$, the required computation is found. Otherwise, assume inductively that O^{j_n} has the same n first elements O has and that e_n occurs as the $(m+1)$th element, i.e., $O^{j_n} = \{e_0, e_1, \ldots, e_{n-1}, e'_n, \ldots, e'_{m-1}, e_n, \ldots\}$. Using Lemma 3.1. it is easy to construct a computation $\xi^{j_{n+1}}$ with observation h and inducing the generation ordering $O^{j_{n+1}} = \{e_0, e_1, \ldots, e_{n-1}, e'_n, \ldots, e_n, e'_{m-1}, \ldots\}$. In fact, e_n and e'_{m-1} are concurrent in h, for they appear in reverse order in O^{j_n} and O, which are both compatible with $\leqslant$. Performing a total of m–n exchanges we obtain $O^{j_n+m-n} = O^{j_{n+1}}$. This proves the inductive step.

We now define ξ as the computation having $Q_i = Q^{j_i}{}_i$ and $u_i = u^{j_i}{}_i$, which is indeed a computation since $u^{j_i}{}_i = u^{(j_{i+1})}{}_i$ and $u^{(j_{i+1})}{}_{i+1}$ are consecutive in computation $\xi^{j_{i+1}}$. Finally, the total ordering induced by ξ is exactly O. ∎

THEOREM 3.3 (commutative implies completely concurrent). *A distributed transition system $\mathbf{T} = \langle Q, T, c, Q_0, 1 \rangle$ commutative with respect to an abstraction partial function abstr is* completely concurrent *with respect to abstr.*

Proof. We extend abstr by defining a total function abstr′, which evaluates to "*undefined*" whenever abstr was undefined. It is easy to see that **T** is commutative with respect to abstr′, too.

Given a computation ξ, let histories $h = \langle S, 1, \leqslant \rangle$ and $h' = \langle S', 1', \leqslant' \rangle$ be its observations using abstr and abstr′, respectively; and let O and O' be the finitely preceded total orderings on their events induced by ξ. Recall that h and O are obained from h' and O' by forgetting the events labelled by "*undefined*":

(i) Since O' is compatible with $\leqslant'$ (Lemma 3.2), it is immediate that O is compatible with $\leqslant$.

(ii) Conversely, a finitely preceded total ordering O on the events of h compatible with $\leqslant$ is given, and we must find a computation ξ with observation h and including O. A history h' must exist, from which h was abstracted. In order to prove

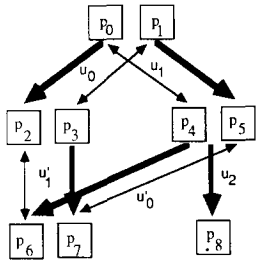

FIG. 3.4. A distributed transition system.

the theorem, is suffices to find an O' compatible with $\leqslant'$, from which O can be abstracted, since Lemma 3.2 can then be applied.

Such an O' does exist, since the relation $R = O \cup \leqslant'$ is a partial ordering (only the events labelled by "*undefined*" may be unrelated). In fact, a cycle in R would imply the existence of a cycle either in O or in $\leqslant'$, because O is compatible with $\leqslant$. ∎

EXAMPLE 3.1 (*confusion*). Consider the distributed transition system in Fig. 3.4, which is commutative with respect to the abstraction function

$$\text{abstr}(u_0) = \text{abstr}(u_0') = a; \qquad \text{abstr}(u_1) = \text{abstr}(u_1') = b; \qquad \text{abstr}(u_2) = c.$$

As stated by Theorem 3.3, the transition system is completely concurrent. In fact, both computations $\{\{p_0, p_1\}\, u_1\, \{p_4, p_5\}\; u_0'\, \{p_6, p_7\}\}$ and $\{\{p_0, p_1\}\, u_0\, \{p_2, p_3\}\; u_1'\, \{p_6, p_7\}\}$ (where p_0, p_4; p_5, p_7; p_1, p_3; p_2, p_6 have been identified) have as their observation the history depicted in Fig. 3.5, and the events labelled by (a) and by (b) are generated in both orders.

Let us now consider a different abstraction function abstr′ which associates to every transition a pair ⟨action, refusals⟩. The action is the same as above, and the refusals contain the actions labelling those transitions that have the same initial state of, and intersecting heads with, the selected transition:

$$\text{abstr}'(u_0) = \langle a, \{\}\rangle; \qquad \text{abstr}'(u_0') = \langle a, \{c\}\rangle;$$
$$\text{abstr}'(u_1) = \text{abstr}'(u_1') = \langle b, \{\}\rangle;$$
$$\text{abstr}'(u_2) = \langle c, \{a\}\rangle.$$

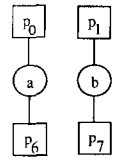

FIG. 3.5. An abstract history (w.r.t abstr) for the system in Fig. 3.4.

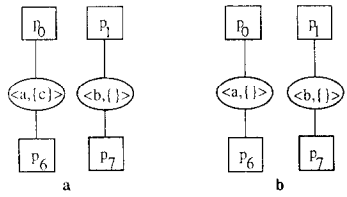

FIG. 3.6. Two abstract histories (w.r.t. abstr') for the system in Fig. 3.4.

It is immediately seen that the transition system is not completely concurrent (and thus not commutative) with respect to abstr'. In fact, Fig. 3.6. shows in (a) the abstract history which can be generated by the computation $\{\{p_0, p_1\}\, u_1\, \{p_4, p_5\}\, u_0'\, \{p_6, p_7\}\}$; in (b) the abstract history which can be generated by the computation $\{\{p_0, p_1\}\, u_0\, \{p_2, p_3\}\, u_1'\, \{p_6, p_7\}\}$. Notice that the history in Fig. 3.6a cannot be generated by any computation where the event labelled by $\langle b, \{\,\}\rangle$ is generated after the event labelled by $\langle a, \{c\}\rangle$; it is analogous for the history in Fig. 3.6b.

The above example shows that, even in the case of completely concurrent distributed systems, the nondeterministic choice among mutually exclusive actions may depend on the selected computation. We can formally state this fact as follows.

PROPERTY 3.5 (choice may be not observable). *There exists a distributed transition system* $\mathbf{T} = \langle Q, T, c, Q_0, 1\rangle$ *commutative with respect to an abstraction partial function abstr, which, if we consider*

- *an abstraction function abstr' defined as*

$$\text{abstr}'(u) = \langle \text{abstr}(u), \text{abstr}(I_u)\rangle,$$

where

$$c(u) = \langle \text{P}, \text{Q}\rangle$$

and

$$I_u = \{u' \in T \mid c(u') = \langle \text{P}, \text{R}\rangle, \quad \text{Q} \neq \text{R} \text{ and } u, u' \text{ have intersecting heads}\};$$

- *the abstract derived NMSs t and t' of* **T** *with respect to abstr and abstr',*

is such that

(i) **T** *is not completely concurrent with respect to abstr';*

(ii) *t' is finer than and not similar to t.*

The fact that the nondeterministic choice among mutual exclusive actions may be *non-objective*, here stated in the context of a formal notion of observation, was well-known at an informal level (e.g., in the Petri-net framework, where the situation of the above example is called confusion [11]).

4. Rewriting Systems

A standard way of defining a transition system is starting from a set of rewriting rules. The states of the system, to which the rewriting rules apply, are represented as sets of processes labelled by elements in a countable set E of **process types**. More precisely, rather than sets of labelled processes we consider hereto *equivalence classes* induced by label-preserving isomorphisms. *For every class* we choose *a standard representative*, with the only condition that *all representatives are pairwise disjoint.*

Definition 4.1. A **rewriting system** is a countable set Z of **rewriting rules**, labelled by pairs $lhs \to rhs$ of finite sets of labelled processes.

A rewriting rule specifies how a set of processes can evolve, regardless of the other processes present in a state. Its left- and right-hand sides can therefore be seen as describing the heads and tails of many transitions. Each of these transitions can be obtained by embedding the left- (right-) handside in a finite context, i.e., a finite set of idles, to obtain its initial (final) state.

Definition 4.2 (*from rewriting systems to distributed transition systems*). Given a set E of process types, a rewriting system Z, and a finite set Q_0 of E-labelled processes, the **derived distributed transition system of Z** is the distributed transition system $\langle Q, T, c, Q_0, 1 \rangle$ where:

- the states in Q are the finite sets of processes with labels in E (they are the standard representatives with respect to label-preserving isomorphism we discussed above);
- given two states P and R, for every rewrite rule z of Z, labelled by $lhs \to rhs$ with $lhs \subseteq P$ and $rhs \subseteq R$, and for every partial label-preserving bijection i: $P \multimap R$ with domain $P \backslash lhs$ and range $R \backslash rhs$, if any, a transition u appears in T. The element z is called **associated** to the transition u;
- $c(u) = \langle P, R \rangle$;
- $1(u) = i$.

Let us call **use** the function mapping a transition to its associated rewriting rule and a process to its process type.

The intuitive notion of asynchrony requires that a set of processes able to make a transition must maintain this capability also when merged with any other set of processes, which is actually expressed by the above definition.

Definition 4.3 (*from rewriting systems to initial NMSs*). Given a set of process types E, a rewriting system Z, and a finite set of E-labelled processes Q_0, the **initial** derived NMS of Z is the abstract derived NMS of the transition system derived from E, Z, and Q_0, with respect to the abstraction function use.

EXAMPLE 4.1. Let us have: $\{A\}$ as set E of process types (therefore, all processes are labelled by A); the rewriting system Z with rules (a) and (b) both labelled by $\{A\} \to \{A\}$; and a set with two elements as initial state. The derived distributed system of Z is $\mathbf{T} = \langle Q, T, c, Q_0, 1 \rangle$, where:

- $Q = \{\{\ \}, \{p_1^1\}, \{p_1^2, p_2^2\}, \ldots, \{p_1^i, p_2^i, \ldots, p_i^i\}, \ldots\}$;
- $\mathbf{T} = \{u_{i;j,\pi}^z \mid z = \text{a, b}; i = 0, 1, \ldots; j = 1, \ldots, i;$ and π is a permutation of $\{1, \ldots, i\}\}$;
- $c(u_{i;j,\pi}^z) = \langle \{p_1^i, p_2^i, \ldots, p_i^i\}, \{p_1^i, p_2^i, \ldots, p_i^i\} \rangle$;
- $Q_0 = \{p_1^2, p_2^2\}$;
- $1(u_{i;j,\pi}^z)(p_k^i) = p_{\pi(k)}^i$, provided that $k \neq j$.

Note that the sets of the heads and of the tails of $u_{i;j,\pi}^z$ are the singletons $\{p_j^i\}$ and $\{p_{\pi(j)}^i\}$. Finally, we have

- $\text{use}(u_{i;j,\pi}^z) = z$;
- $\text{use}(p_j^i) = A$.

Figure 1.3 shows the transitions concerning $\{p_1^2, p_2^2\}$. Figures 4.1a, c, depict two computations of $\mathbf{T}$, from which the initial histories in Figs. 4.1b, d, are observed. Note that in the abstract interleaving NMS (w.r.t. use) both computations are observed as $\{\{p_1^2, p_2^2\}\ a\ a\ a \ldots\}$.

THEOREM 4.1 (rewriting systems are commutative). *Given a set of process types E, a rewriting system Z, and a finite set of E-labelled processes Q_0, the derived distributed transition system of Z is commutative with respect to the abstraction function use.*

Proof. Let u_1 and u_2 be two transitions such that u_2 is independent from u_1, and let z_1 and z_2, respectively, be the rewriting rules associated to them. Furthermore, let I be the set of the processes which are idle in both u_1 and u_2 and recall that I and the heads (tails) of u_1 and u_2 are three disjoint sets of processes by Property 3.1. The required pair of transitions u_2', u_1' is immediately constructed tak-

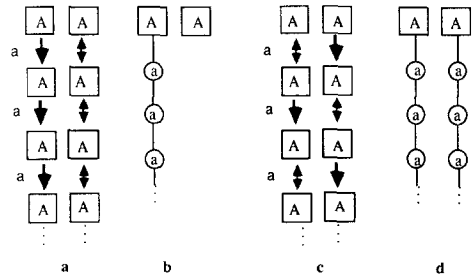

FIG. 4.1(a) Two computations of the distributed transition system T of Example 4.1., from which the histories in (b) and (d) are observed. All the transitions used are obtained from the same rewriting rule a, labelled by $A \to A$.

ing as final state of u_2' (and as initial state of u_1') the union of the tails of u_2 (i.e., the *rhs* of z_2), the heads of u_1 (i.e., the *lhs* of z_1), and the idles I. Clearly, we have

$$\text{use}(u_i) = \text{use}(u_i') = z_i, \qquad i = 1, 2. \quad \blacksquare$$

COROLLARY 4.1 (rewriting systems are completely concurrent). *Given a set of process types E, a rewriting system Z, and a finite set of E-labelled processes Q_0, the derived transition system of Z is completely concurrent with respect to use.*

Proof. Follows from Theorems 4.1 and 3.3. ∎

DEFINITION 4.4 (*abstract NMSs and semantics*). Given a set of process types E, a rewriting system Z, a finite set of E-labelled processes Q_0, and an abstraction partial function

$$\text{abstr}: (Z \multimap A) \cup (E \multimap E'),$$

let t be the initial derived NMS of Z.

- The **abstract** derived NMS is obtained by replacing any history h labelling a node of t with the abstract history with respect to abstr.
- Given an equivalence relation $\equiv$ on NMSs, the **semantics** is the abstract derived NMS, defined up to $\equiv$.

We can now continue the discussion on fairness properties started at the beginning of Section 3.

THEOREM 4.2 (detecting global fairness from observations). *Given a set of process types E, a rewriting system Z, and a finite set of E-labelled processes Q_0, let $\mathbf{T} = \langle Q, T, c, Q_0, 1 \rangle$ be the derived distributed transition system of Z, and let Ξ be the set of all infinite computations of $\mathbf{T}$ having the same concurrent history h as observed in the initial derived NMS. If the union of the tails and the idles of h contains the set lhs of a rule in Z, then all the computations in Ξ are globally unfair. Otherwise, they are all globally fair.*

Proof. Let ξ be a globally unfair computation of Ξ, if any, and let *lhs* be the left hand-side of a rewriting rule included in the union of the tails and the idles of h. The same set *lhs* must appear in every state of all the computations of Ξ, from some point onwards. ∎

COROLLARY 4.2 (global weak = global strong fairness). *Given a set of process types E, a rewriting system Z, and a finite set of E-labelled processes Q_0, let $\mathbf{T} = \langle Q, T, c, Q_0, 1 \rangle$ be the derived distributed transition system of Z. A computation of $\mathbf{T}$ is globally strongly fair iff it is globally weakly fair.*

Proof. Immediate from Theorems 3.2 and 4.1. ∎

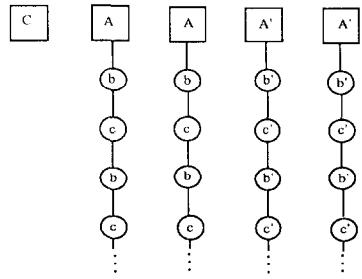

FIG. 4.2. An infinite history.

COROLLARY 4.3 (global fairness is observable). *Given a set of process types E, a rewriting system Z, and a finite set of E-labelled processes* Q_0, *let*

- $\mathbf{T} = \langle Q, T, c, Q_0, 1 \rangle$ *be the derived distributed transition system of Z;*
- *t be the abstract derived NMS with respect to the abstraction function abstr defined as* abstr(z) = "applied."

Global fairness of the computations of **T** *is observable through t (see Definition* 2.7).

Proof. Immediate from Theorem 4.2. ∎

EXAMPLE 4.2 (*fairness*). Let us have the set of process types $\{C, A, A, A', A'\}$ and the rewriting system with rewriting rules a, a', b, b', c, c', labelled

$$a \text{ by } \{C, A, A\} \to \{D\}; \quad b \text{ by } \{A\} \to \{B\}; \quad c \text{ by } \{B\} \to \{A\};$$
$$a' \text{ by } \{C, A', A'\} \to \{D\}; \quad b' \text{ by } \{A'\} \to \{B'\}; \quad c' \text{ by } \{B'\} \to \{A'\}.$$

Furthermore, let **T** be the derived distributed transition system having as initial state $Q_0 = \{C, A, A, A', A'\}$ (we simply write the label for the process).

According to Theorem 4.2, the set Ξ of the computations which generate the infinite initial history depicted in Fig. 4.2 contain globally fair computations only. Furthermore, these computations can be either.

(i) locally weakly unfair;

(ii) locally strongly unfair and locally weakly fair; or

(iii) locally strongly fair.

Examples of (i)–(iii) are given below, where "C can proceed" means that the transitions with associated rewriting rules a or a' are also possible:

(i) **repeat forever**

$\{C, A, A, A', A'\}$ apply b C can proceed
$\{C, B, A, A', A'\}$ apply c C can proceed
$\{C, A, A, A', A'\}$ apply b C can proceed

$\{C, A, B, A', A'\}$ apply c C can proceed
$\{C, A, A, A', A'\}$ apply b' C can proceed
$\{C, A, A, B', A'\}$ apply c' C can proceed
$\{C, A, A, A', A'\}$ apply b' C can proceed
$\{C, A, A, A', B'\}$ apply c' C can proceed

end repeat

(ii) **repeat forever**

$\{C, A, A, A', A'\}$ apply b C can proceed
$\{C, B, A, A', A'\}$ apply b C can proceed
$\{C, B, B, A', A'\}$ apply b' C can proceed
$\{C, B, B, B', A'\}$ apply b'
$\{C, B, B, B', B'\}$ apply c
$\{C, A, B, B', B'\}$ apply c
$\{C, A, A, B', B'\}$ apply c' C can proceed
$\{C, A, A, A', B'\}$ apply c' C can proceed

end repeat

(iii) $\{C, A, A, A', A'\}$ apply b C can proceed
$\{C, B, A, A', A'\}$ apply b' C can proceed

repeat forever

$\{C, B, A, B', A'\}$ apply b
$\{C, B, B, B', A'\}$ apply c
$\{C, A, B, B', A'\}$ apply b
$\{C, B, B, B', A'\}$ apply c
$\{C, B, A, B', A'\}$ apply b'
$\{C, B, A, B', B'\}$ apply c'
$\{C, B, A, A', B'\}$ apply b'
$\{C, B, A, B', B'\}$ apply c'

end repeat

COROLLARY 4.4 (local fairness is not observable). *There exist a set of process types E, a rewriting system Z, and a finite set of E-labelled processes* Q_0, *having*

- **T** *as derived distributed transition system of Z;*
- *t as the abstract derived NMS with respect to the abstraction function abstr defined as* abstr(z) = "applied."

Local fairness, both weak and strong, of the computations of **T** *is not observable through t (see Definition* 2.7).

Proof. Immediate, by considering Example 4.2 as a counterexample. ∎

5. PETRI NETS

We now consider Petri nets and compare them with our distributed transition systems. A number of models go under the name of Petri nets. Here, we consider the two basic ones: condition/event (C/E) systems and marked place/transition (P/T) nets. We translate the former to distributed transitions systems and the latter to rewriting systems. We briefly introduce the relevant definitions [11, 13, 21].

A **net** is a triple $\langle S, T; F\rangle$, where

- $S \cap T = \emptyset$;
- $F \subseteq (S \times T) \cup (T \times S)$.

Given a net $N = \langle S, T; F\rangle$, let $x, y \in S \cup T$,

- $\dot{}x$ denotes $\{y \mid yFx\}$ and $x\dot{}$ denotes $\{y \mid xFy\}$;
- x is **isolated** if $\dot{}x \cup x\dot{} = \emptyset$;
- N is **simple** if, whenever $\dot{}x = \dot{}y$ and $x\dot{} = y\dot{}$ then $x = y$;
- a subset $c \subseteq S$ is called **case**;
- $t \in T$ is c-**enabled** iff c is a case and $\dot{}t \subseteq c$ and $t\dot{} \subseteq S \backslash c$.

Furthermore, given $c_1, c_2 \subseteq S$ and $G \subseteq T$, the **step** $c_1[G > c_2$, is defined if

- $\forall t \in G$, t is c-**enabled**;
- $\forall t_1, t_2 \in G$, $t_1 \neq t_2$, $\dot{}t_1 \cap \dot{}t_2 = t_1\dot{} \cap t_2\dot{} = \emptyset$;
- $c_2 = (c_1 \backslash \dot{}G) \cup G\dot{}$ (understanding $\cdot$ extended on sets).

A **condition/event system** (C/E system) is a quadruple $\Sigma = \langle B, E; F, C\rangle$, where

- $\langle B, E; F\rangle$ is a simple net with no isolated element and $B \cup E \neq \emptyset$;
- $C \subseteq 2^B$ is an equivalence class of the reachability relation $R = (r \cup r^{-1})^*$ being $r \subseteq 2^B \times 2^B$, where $c_1 r c_2$ iff $\exists G \subseteq E$ such that $c_1[G > c_2$;
- $\forall e \in E$, $\exists c \in C$ such that e is c-enabled.

DEFINITION 5.1. A C/E system $\Sigma = \langle B, E; F, C \rangle$ is **finitely forking** if $\forall e \in E$, ${}^{\cdot}e$ finite implies $e^{\cdot}$ finite.

DEFINITION 5.2 (*from C/E systems to distributed transition systems*). Given a finitely forking C/E system $\Sigma = \langle B, E; F, C \rangle$ and a finite initial case $c_0 \in C$, the **associated** distributed transition system $\mathbf{T}_\Sigma$ is $\langle Q, T, c, Q_0, 1 \rangle$, where

- $Q = C$, considering the cases as disjoint. Thus, an element of B is **associated** to each process in a state of Q by a function called **P_abstr**;
- for every pair of states $P, R \in Q$ and for every $e \in E$, such that

$$P\_\text{abstr}(P)[e > P\_\text{abstr}(R),$$

a transition u is in T.

The element e in E is called **associated** by **P_abstr** to transition u:

- $c(u) = \langle P, R \rangle$;
- $Q_0 = c_0$;
- function $1(u) = i$: $P \backslash {}^{\cdot}e \to R \backslash e^{\cdot}$ identifies processes with the same associated element of B.

In our model it is always possible to observe a computation through a history. This is not the case for C/E systems, where constact-freeness is required to obtain a suitable partial ordering observation.

A **contact-free** C/E system is a C/E system $\langle B, E; F, C \rangle$ in which $\forall e \in E$, $\forall c \in C$,

- ${}^{\cdot}e \subseteq c$ implies $e^{\cdot} \subseteq B \backslash c$;
- $e^{\cdot} \subseteq c$ implies ${}^{\cdot}e \subseteq B \backslash c$.

The standard notion of partial ordering observation in Petri nets is called a C/E process and is defined as follows:

An **occurrence net** is a net $K = \langle S, T; F \rangle$ such that

- the transitive closure of F, denoted by F^+, is irreflexive;
- $\forall s \in S$, $|{}^{\cdot}s| \leqslant 1$ and $|s^{\cdot}| \leqslant 1$.

Furthermore,

- $S \cup T$ is considered as **ordered** by $<$, defined as F^+;
- the **slices** of K are the maximal subsets of S which do not contain elements related by $<$;
- K is **bounded** if every $A \subseteq S \cup T$ satisfying both conditions below is finite:

  (i) A is totally ordered;

  (ii) $\forall x \in (S \cup T) \backslash A$, $\exists y \in A$ such that $x \not< y$ and $y \not< x$.

Given a contact-free C/E system $\Sigma = \langle B, E; F, C \rangle$ and a bounded occurrence net $K = \langle S', T'; F' \rangle$, a **C/E process** of Σ is a function

$$p: K \to \Sigma$$

such that

- $p(S') \subseteq B$, $p(T') \subseteq E$; and, for all slice D of K
- p restricted on D is injective and $p(D) \in C$;
- $p(\dot{}t) = \dot{}p(t)$ and $p(t\dot{}) = p(t)\dot{}$ $\forall t \in T'$.

DEFINITION 5.3. Given a contact-free C/E system $\Sigma = \langle B, E; F, C \rangle$ and a case $c \in C$, a C/E process $p: K \to \Sigma$ is **based** on c if

$$c = \{p(s) | s \in S' \text{ and } s \text{ is a minimum of } <'\}, \text{ where } K = \langle S', T'; F' \rangle.$$

DEFINITION 5.4 (*processes are almost histories*). Given a C/E process p of a finitely forking and contact-free C/E system $\Sigma = \langle B, E; F, C \rangle$,

$$p: K \to \Sigma,$$

where $K = \langle S', T'; F' \rangle$, the **corresponding** history $h = \langle S, 1, \leqslant \rangle$ is defined as follows:

- the set of process (event) types is B (E);
- $S = (S' \cup T') \setminus \{s \in S' | \exists t_1, t_2 \in T' \text{ such that } t_1 <' s <' t_2\}$;
- $1(s) = p(s)$;
- $\leqslant$ is the reflexive closure of $<'$, restricted on S.

THEOREM 5.1 (contact-free C/E systems are commutative). *Given a finitely forking and contact-free C/E system $\Sigma = \langle B, E; F, C \rangle$ and a finite initial case $c_0 \in C$, the associated distributed transition system $\mathbf{T}_\Sigma = \langle Q, T, c, Q_0, 1 \rangle$ is* commutative *with respect to P_abstr.*

Proof. The proof will be immediate later, so we postpone it. ∎

COROLLARY 5.1 (contact-free C/E systems are completely concurrent). *Given a finitely forking and contact-free C/E system $\Sigma = \langle B, E; F, C \rangle$ and a finite initial case $c_0 \in C$, the associated distributed transition system $\mathbf{T}_\Sigma = \langle Q, T, c, Q_0, 1 \rangle$ is* completely concurrent *with respect to P_abstr.*

Proof. Follows from Theorems 3.3 and 5.1.

THEOREM 5.2 (the processes of a contact-free C/E system coincide with the histories of the corresponding distributed transition system). *Given a finitely forking and contact-free C/E system $\Sigma = \langle B, E; F, C \rangle$, a finite initial case $c_0 \in C$, and the associated distributed transition system $\mathbf{T}_\Sigma = \langle Q, T, c, Q_0, 1 \rangle$, let H be the set of the*

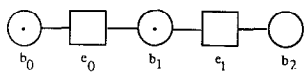

FIG. 5.1. A non-contact-free C/E system Σ.

histories corresponding to the finite C/E processes of Σ based on c_0. H coincides with the set of the finite observations of the computations of $\mathbf{T}_\Sigma$ in the abstract derived NMS with respect to P_abstr.

Proof. It is easy to see that there is a one-to-one correspondence between computations of $\mathbf{T}_\Sigma$ and the paths $\{c_0 e_0 c_1 e_1 \cdots\}$, where $c_i[e_i > c_{i+1}$. This is the case since process identifications in the computations of $\mathbf{T}_\Sigma$ are represented in the corresponding paths by the intersections of adjacent cases.

Given a process p, a path ω is immediately constructed by taking any total ordering on its events compatible with $<$. It is easy to prove by induction that the history corresponding to p and the history observed from the computation corresponding to ω are isomorphic.

The other direction is analogously proved, by noting that from a path it is immediate to obtain one process. ∎

We noted above that, when the condition of contact-freeness is relaxed, C/E processes are not defined, while computations can still be observed through histories. In this case, however, complete concurrency may be not satisfied.

EXAMPLE 5.1 (*Non-contact-free C/E systems may correspond to non-completely concurrent distributed transition systems*). Let us consider the non-contact-free C/E system Σ depicted in Fig. 5.1, where the initial case is represented through the marking. Figure 5.2a shows the corresponding transition system $\mathbf{T}_\Sigma$, where

$$P\_\text{abstr}(u_i) = e_i$$

and

$$P\_\text{abstr}(p_0) = P\_\text{abstr}(p_2) = b_0;$$
$$P\_\text{abstr}(p_1) = P\_\text{abstr}(p_4) = b_1;$$
$$P\_\text{abstr}(p_3) = P\_\text{abstr}(p_5) = b_2.$$

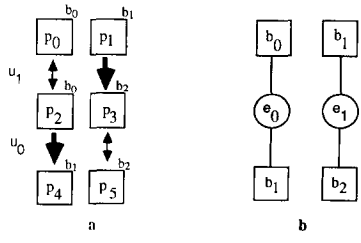

FIG. 5.2(a) The transition system $\mathbf{T}_\Sigma$; (b) A history of $\mathbf{T}_\Sigma$.

$\mathbf{T}_\Sigma$ is not completely concurrent with respect to P_abstr, as shown by history h in Fig. 5.2b. Actually, h can be generated only by the computation $\{\{p_0, p_1\}\, u_1\, \{p_2, p_3\}\, u_0\, \{p_4, p_5\}\}$, where p_0, p_2 and p_3, p_5 are identified.

We can now compare the interleaving and the partial ordering semantics of the distributed transition systems derived from C/E systems.

THEOREM 5.3 (C/E histories are recoverable from interleavings). *Given a finitely forking C/E system, let us consider its distributed transition system, its abstract interleaving NMS t w.r.t. P_abstr, and its initial partial ordering NMS t'. Then t is finer than t' (see Definition* 2.8).

Proof. From an interleaving observation it is immediate to recover the single computation generating it. ∎

We consider now P/T nets and show that they can be described in terms of rewriting systems and derived distributed transition systems.

Given a set S, a marking M of S is a function

$$M: S \to \mathbf{N},$$

where $\mathbf{N}$ is the set of the natural numbers.

PROPERTY 5.1 (markings are labelled sets of processes). *There is a one-to-one correspondence between the markings of S and the equivalence classes of S-labelled processes, under label-preserving isomorphisms.*

A **marked place/transition net** (P/T *net*) *is a quintuple* $N = \langle S, T; F, W, M \rangle$, *where*

- $\langle S, T; F \rangle$ *is a net, with S and T finite*;
- $: F \to \mathbf{N}$ *assigns a positive* **weight** *to each arc*;
- $M: S \to \mathbf{N}$ *is the* **initial marking** *of N.*

DEFINITION 5.5 (*from P/T nets to rewriting systems*). Given a P/T net $N = \langle S, T; F, W, M \rangle$, the **rewriting system** $\mathbf{Z}_\mathbf{N}$ has S as the set of process types and contains a rewriting rule z, labelled by $lhs \to rhs$, for every element $t \in T$. The (label-preserving equivalence class of the) set of labelled processes lhs contains, $\forall s \in {}^\cdot t$, a number $W(\langle s, t \rangle)$ of processes labelled by s. The set of labelled processes rhs contains, $\forall s \in t^\cdot$ a number $W(\langle t, s \rangle)$ of processes labelled by s.

DEFINITION 5.6 (*from P/T nets to distributed transition systems*). Given a P/T net $N = \langle S, T; F, W, M \rangle$, the **distributed transition system** $\mathbf{D}_N$ is the distributed transition system derived from S, Z_N and Q_0, being Q_0 the finite set of S-labelled processes corresponding to M.

PROPERTY 5.2 (P/T nets are finite rewriting systems). *The correspondence defined in the above Definitions* 5.5 *and* 5.6 *between P/T nets and distributed trans-*

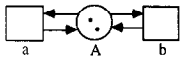

FIG. 5.3. The P/T net of Example 5.2.

ition systems derived from rewriting systems with a finite *number of rules is one-to-one.*

Proof. A triple $\langle S, Z_N, Q_0 \rangle$, Z_N finite, corresponds to both one distributed transition system and one P/T net. ∎

EXAMPLE 5.2. According to Property 5.2, the distributed transition system of Example 4.1 corresponds to a P/T net N, being generated from a set of two rewriting rules Z. More precisely, the net $N = \langle S, T; F, W, M \rangle$, depicted in Fig. 5.3, has

- $S = \{A\}$;
- $T = \{a, b\}$;
- AFa, aFA, AFb, bFA;
- $W(\langle A, a \rangle) = W(\langle a, A \rangle) = W(\langle A, b \rangle) = W(\langle b, A \rangle) = 1$;
- $M(A) = 2$. ∎

COROLLARY 5.2 (P/T nets are completely concurrent). *Given a P/T net* $N = \langle S, T; F, W, M \rangle$, *the distributed transition system* $\mathbf{D}_N$ *is* completely concurrent *with respect to function use* (*see Definition* 4.2).

Proof. Immediate from Corollary 4.1. ∎

Proof of Theorem 5.1. A contact-free C/E system Σ can be seen as a P/T net N, which has W constantly 1 and the initial marking M as the characteristic function of an initial case. Therefore, the distributed transition system $\mathbf{T}_\Sigma$ is essentially isomorphic to the distributed transition system $\mathbf{D}_N$ and thus $\mathbf{T}_\Sigma$ is commutative with respect to $P\_\text{abstr} = \text{use}$. The only difference is that in $\mathbf{T}_\Sigma$ the states are conveniently the cases, i.e., the reachable subsets of B, while in $\mathbf{D}_N$ it is necessary to have as states all the finite sets of processes with labels in B. Notice that states having more than one process with the same label are certainly not reachable, due to the contact-freeness condition. ∎

Given a P/T net $N = \langle S, T; F, W, M \rangle$, a **firing sequence** of N is $\{M_0 t_0 M_1 t_1 M_2 \cdots\}$, where

- M_i are markings of S and $M_0 = M$; $t_i \in T$;
- $M_i[t_i > M_{i+1}$, where $M[t > M'$ implies that $\forall s \in S$, $M(s) \geqslant W(s, t)$ and $M'(s) = M(s) - W(s, t) + W(t, s)$.

One may think that firing sequences in P/T nets play the rôle of computations in

distributed transition systems. This is not the case: firing sequences are more abstract, since they correspond to observations in abstract interleaving NMSs and many computations can generate the same observation.

THEOREM 5.3 (firing sequences coincide with interleaving observations and abstract out from computations). *Given a P/T net* $N = \langle S, T; F, W, M \rangle$ *and its distributed transition system* $\mathbf{D}_N$, *let*

- *FS be the set of the firing sequences of* N;
- *AFS be obtained from FS, by dropping the intermediate markings*;
- *IS be the set of the observations of the computations of* $\mathbf{D}_N$ *in the abstract interleaving NMS of* $\mathbf{D}_N$ *with respect to use.*

We have that

(i) *FS and AFS are isomorphic*;

(ii) $AFS = IS$;

(iii) *more than one computation may have the same observation.*

Proof. (i) The intermediate markings can be recovered by firing the transitions in the order, starting from M.

(ii) Given an observation ω, let ξ be one of the computations of $\mathbf{D}_N$ generating it. From ξ, an element σ of AFS can be obtained, by listing the initial marking, the transition labels, and the last marking, if any. It is easy to show, by induction, that $\omega = \sigma$. The converse is symmetric: from σ get one of the computations ξ, and from this the observation ω.

(iii) See Examples 4.1 and 5.2, and Fig. 1.4, too. ∎

Given a P/T net $N = \langle S, T; F, W, M \rangle$ and an occurrence net $K = \langle S', T'; F' \rangle$, a **P/T process** of N is a function

$$p: K \to N$$

such that

- $p(S') \subseteq S$, $p(T') \subseteq T$;
- $\langle S' \cup T', <' \rangle$ is finitely preceded. Let ${}^0K$ be the set of its minima;
- $\forall s \in S,\ M(s) = |p^{-1}(s) \cap {}^0K|$;
- $\forall t' \in T',\ \forall s \in S,$
  - (i) $W(s, p(t')) = |p^{-1}(s) \cap {}^\cdot t'|$
  - (ii) $W(p(t'), s) = |p^{-1}(s) \cap t'^\cdot|$.

DEFINITION 5.7 (*processes are almost histories*). Given a P/T process of a P/T net $N = \langle S, T; F, W, M \rangle$,

$$p: K \to \Sigma,$$

where $K = \langle S', T'; F' \rangle$, the **corresponding** history $h = \langle S, 1, \leqslant \rangle$ is defined as follows:

- the set of process (event) types is S (T);
- $S = (S' \cup T') \setminus \{s \in S' | \exists t_1, t_2 \in T' \text{ such that } t_1 <' s <' t_2\}$;
- $1(s) = p(s)$;
- $\leqslant$ is the reflexive closure of $<'$, restricted on S.

THEOREM 5.4 (the processes of a P/T net coincide with the histories of the corresponding distributed transition system). *Given a P/T net N and its distributed transition system* $\mathbf{D}_N$, *let H be the set of the histories corresponding to the finite P/T processes of N. H coincides with the set of the finite observations of the computations of* $\mathbf{D}_N$ *in the initial derived NMS.*

Proof. Given a finite process $p: K \to N$ and one of the total orderings $\{e_0, e_1, \ldots, e_n\}$ on the events of p compatible with $<$, let ω be $\{D_0 e_0 D_1 e_1 \cdots e_n D_{n+1}\}$, where D_i are the slices of p such that $D_0 = {}^0K$, $D_i [e_i > D_{i+1}, D_{n+1} = K^0$. It is easy to construct from ω a computation ξ of $\mathbf{D}_N$ generating a history h. By induction, history h is proved to correspond to p. Conversely, given a history h, let ξ be one of the computations generating it. By applying the Steps 1–3 defined in Property 3.3 we get, with the obvious modifications, the required process. ∎

The notion of firing sequence or, equivalently, the notion of observation in the interleaving NMS (Theorem 5.3), is too abstract: it is not possible to recover from it a single process or, equivalently, a single history .

PROPERTY 5.3 (processes are not recoverable from firing sequences). *There exists a P/T net N such that, if we consider its distributed transition system* $\mathbf{D}_N$, *its abstract interleaving NMS t w.r.t. use, and its initial partial ordering NMS t', then t is incomparable with t'* (*see Definition* 2.8).

Proof. Examples 4.1 and 5.2 show that t is not finer than t'. By Theorem 3.1, the same completely concurrent distributed transition system proves that t' is not finer than t. See also the examples in Figs. 1.3 and 1.4. ∎

DEFINITION 5.8 (*observational semantics of P/T nets and C/E systems*). Given an equivalence relation $\equiv$ on NMSs and an abstraction function abstr from net elements to an abstraction domain, the **observational semantics** of a P/T net (a finitely forking C/E system and an initial case) is the abstract derived NMS of the distributed transition system $\mathbf{D}_N$ (of the distributed transition system $\mathbf{T}_\Sigma$) with respect to abstr, defined up to $\equiv$.

6. The Distributed Transition System for CCS

In this section we translate Milner's CCS to distributed transition systems. Recall that the concrete syntax of pure CCS is

$$E::= x|\mathrm{NIL}|\mu E|E\backslash\alpha|E[\phi]|E+E|E|E|\mathrm{rec}\ x.E,$$

where x is a variable; Δ is a set of unary operators ranged over by $\alpha, \beta, \gamma, \ldots$; $\Delta^- = \{\alpha^- | \alpha \in \Delta\}$; $\Lambda = \Delta \cup \Delta^- \cup \{\tau\}$ is the set of **basic actions**, ranged over by μ, and $\tau \notin \Delta \cup \Delta^-$, which in turn is ranged over by λ; and ϕ is a permutation of Λ which preserves τ and the operation $^-$ of complementation.

The **Milner derivation relation** $E_1 — \mu \to E_2$ is defined as the least relation satisfying the following axiom and inference rules:

(Act)	$\mu E — \mu \to E$		
(Res)	$E_1 — \mu \to E_2$	**implies**	$E_1\backslash\alpha — \mu \to E_2\backslash\alpha, \quad \mu \notin \{\alpha, \alpha^-\}$
(Rel)	$E_1 — \mu \to E_2$	**implies**	$E_1[\phi] — \phi(\mu) \to E_2[\phi]$
(Sum)	$E_1 — \mu \to E_2$	**implies**	$E_1 + E — \mu \to E_2$ **and** $E + E_1 — \mu \to E_2$
(Com)	$E_1 — \mu \to E_2$	**implies**	$E_1 \mid E — \mu \to E_2 \mid E$ **and** $E \mid E_1 — \mu \to E \mid E_2$
	$E_1 — \lambda \to E_2$ **and** $E_1' — \lambda^- \to E_2'$	**implies**	$E_1 \mid E_1' — \tau \to E_2 \mid E_2'$
(Rec)	$E_1[\mathrm{rec}\ x.E_1/x] — \mu \to E_2$	**implies**	$\mathrm{rec}\ x.E_1 — \mu \to E_2$.

We will use the following conventions:

- $E_1 = \lambda \Rightarrow E_2$ stands for there exist E_1' and E_2' such that

$$E_1 — \tau \to^m E_1' — \lambda \to E_2' — \tau \to^n E_2, \qquad m, n \geqslant 0;$$

- $E = s \Rightarrow E'$, $s = \lambda_1 \cdots \lambda_n$, $n \geqslant 0$, stands for there exist E_i; $0 \leqslant i \leqslant n$, such that

$$E = E_0 = \lambda_1 \Rightarrow E_1 = \lambda_2 \Rightarrow \cdots = \lambda_n \Rightarrow E_n = E'.$$

Two terms E_1 and E_2 are **observationally equivalent** if they *bisimulate*, i.e., both conditions hold

(i) whenever $E_1 = s \Rightarrow E_1'$ then, for some E_2',

$$E_2 = s \Rightarrow E_2' \text{ and } E_1' \text{ is observationally equivalent to } E_2';$$

(ii) whenever $E_2 = s \Rightarrow E_2'$ then, for some E_1',

$$E_1 = s \Rightarrow E_1' \text{ is observationally equivalent to } E_1'.$$

We now define a distributed transition system for CCS.

DEFINITION 6.1 (*defining CCS processes*). A grape is a term defined by the following BNF-like grammar

$$G ::= E \mid \mathrm{id}|G \mid G|\mathrm{id} \mid G\backslash\alpha \mid G[\phi],$$

where E, $\backslash\alpha$, $[\phi]$ have the standard CCS meaning.

Intuitively speaking, a grape represents a subterm of a CCS term, together with its access path. A CCS term can be decomposed by function dec into a set of grapes.

DEFINITION 6.2 (*decomposing a CCS term into its processes*). Function dec decomposes a CCS term into a set of grapes and is defined by structural induction as follows:

$$\begin{aligned}
&\mathrm{dec}(x) = \{x\}\\
&\mathrm{dec}(\mathrm{NIL}) = \{\mathrm{NIL}\}\\
&\mathrm{dec}(\mu E) = \{\mu E\}\\
&\mathrm{dec}(E\backslash\alpha) = \mathrm{dec}(E)\backslash\alpha\\
&\mathrm{dec}(E[\phi]) = \mathrm{dec}(E)[\phi]\\
&\mathrm{dec}(E_1 + E_2) = \{E_1 + E_2\}\\
&\mathrm{dec}(E_1|E_2) = \mathrm{dec}(E_1)|\mathrm{id} \cup \mathrm{id}|\mathrm{dec}(E_2)\\
&\mathrm{dec}(\mathrm{rec}\ x.E) = \{\mathrm{rec}\ x.E\}.
\end{aligned}$$

We understand constructors as extended to operate on sets, e.g., $I\backslash\alpha = \{g\backslash\alpha \mid g \in I\}$. Note that the decomposition stops when an action, a sum, or a recursion is encountered, since these are considered as single processes.

EXAMPLE 6.1.

$$\mathrm{dec}((((\mathrm{rec}\ x.\alpha x + \beta x)|\mathrm{rec}\ x.\alpha x + \gamma x)|\mathrm{rec}\ x.\alpha^{-} x)\backslash\alpha) =$$
$$\{(((\mathrm{rec}\ x.\alpha x + \beta x)|\mathrm{id}|\mathrm{id})\backslash\alpha,\quad ((\mathrm{id}|\mathrm{rec}\ x.\alpha x + \gamma x)|\mathrm{id})\backslash\alpha,\quad (\mathrm{id}|\mathrm{rec}\ x.\alpha^{-} x)\backslash\alpha\}.$$

DEFINITION 6.3. A set of grapes I is **complete** if there exists a CCS term E such that $\mathrm{dec}(E) = I$.

PROPERTY 6.1. *Function dec is injective and thus defines a bijection between CCS terms and complete sets of grapes.*

Proof. Immediate by induction. ∎

DEFINITION 6.4 (*partial ordering derivation relation*). The **partial ordering derivation relation** $I_1 - [\mu, I_3] \rightarrow I_2$ is defined as the least relation satisfying the following axiom and inference rules:

(**act**) $\{\mu E\}$—$[\mu, \varnothing] \to \mathrm{dec}(E)$

(**res**) I_1—$[\mu, I_3] \to I_2$ **implies** $I_1 \backslash \alpha$—$[\mu, I_3 \backslash \alpha] \to I_2 \backslash \alpha$, $\mu \notin \{\alpha, \bar{\alpha}\}$

(**rel**) I_1—$[\mu, I_3] \to I_2$ **implies** $I_1[\phi]$—$[\phi(\mu), I_3[\phi]] \to I_2[\phi]$

(**sum**) $\mathrm{dec}(E_1)$—$[\mu, I_3] \to I_2$ **implies** $\{E_1 + E\}$—$[\mu, \varnothing] \to I_2$
and $\{E + E_1\}$—$[\mu, \varnothing] \to I_2$

(**com**) I_1—$[\mu, I_3] \to I_2$
implies $I_1 | \mathrm{id} \cup \mathrm{id} | \mathrm{dec}(E)$—$[\mu, I_3 | \mathrm{id} \cup \mathrm{id} | \mathrm{dec}(E)] \to I_2 | \mathrm{id} \cup \mathrm{id} | \mathrm{dec}(E)$
and $\mathrm{id} | I_1 \cup \mathrm{dec}(E) | \mathrm{id}$—$[\mu, \mathrm{id} | I_3 \cup \mathrm{dec}(E) | \mathrm{id}] \to \mathrm{id} | I_2 \cup \mathrm{dec}(E) | \mathrm{id}$
I_1—$[\lambda, I_3] \to I_2$ **and** I_1'—$[\bar{\lambda}, I_3'] \to I_2'$
implies $I_1 | \mathrm{id} \cup \mathrm{id} | I_1'$—$[\tau, I_3 | \mathrm{id} \cup \mathrm{id} | I_3'] \to I_2 | \mathrm{id} \cup \mathrm{id} | I_2'$

(**rec**) $\mathrm{dec}(E_1[\mathrm{rec}\ x.E_1/x])$—$[\mu, I_3] \to I_2$ **implies** $\{\mathrm{rec}\ x.E_1\}$—$[\mu, \varnothing] \to I_2$.

PROPERTY 6.2 *Given a quadruple* I_1—$[\mu, I_3] \to I_2$

(i) I_1 *and* I_2 *are complete sets of grapes*;

(ii) $I_3 \subset I_1$ *and* $I_3 \subset I_2$.

Proof. Immediate by induction. ∎

THEOREM 6.1 (correspondence between Milner's and partial ordering derivations). *We have a derivation* E_1—$\mu \to E_2$ *iff there exists a set of idles* I_3 *such that*

$$\mathrm{dec}(E_1)\text{—}[\mu, I_3] \to \mathrm{dec}(E_2).$$

Proof. Given a derivation, use the structure of its deduction to obtain the other derivation. ∎

THEOREM 6.2 (idle processes can always be added). *Given a derivable quadruple* $H \cup I_1$—$[\mu, I_3] \to T \cup I_2$, *and a set* I_3' *such that* $H \cup I_3'$ *is complete, also* $H \cup I_3'$—$[\mu, I_3'] \to T \cup I_3'$ is derivable.

Proof. The proof is by induction on the deduction structure of the quadruple. The base step is when the quadruple has been obtained by (**act**) and is trivial, since $I_3' = I_3 = \varnothing$. Also if the deduction step is an inference of type (**sum**) or (**rec**), the inductive step is trivial, since $I_3' = I_3 = \varnothing$. If the quadruple has been derived using (**res**), let I_3 by inductive hypothesis be the most general set of grapes completing both H and T. Then, the most general I_3' which completes both $H\backslash\alpha$ and $T\backslash\alpha$ is exactly $I_3\backslash\alpha$. The same argument holds in case of (**rel**). Let us consider now the first (**com**) rule. The most general I_3' completing $H|\mathrm{id}$ and $T|\mathrm{id}$ is the union of $I_3|\mathrm{id}$ (I_3 completes by hypothesis H and T) and of $\mathrm{id}|\mathrm{dec}(E)$. The proof of the remaining two (**com**) rules is similar. ∎

DEFINITION 6.5 (*from CCS terms no distributed transition systems*). Given a CCS term E, let $\mathbf{D}_{\mathrm{CCS}}(E) = \langle Q, T, c, \mathrm{dec}(E), 1 \rangle$ be its **associated** distributed transition system, where:

- the states in Q are the complete sets of grapes, considered as disjoint;
- the transitions in T are quadruples $u: I_1 — [\mu, I_3] \rightarrow I_2$;
- $c(u) = \langle I_1, I_2 \rangle$;
- the processes identified by $1(u)$ in I_1 and I_2 are the processes in I_3 (recall that $I_3 \subset I_1$ and $I_3 \subset I_2$ by Property 6.2).

EXAMPLE 6.2. Let us consider the CCS term of Example 6.1,

$$E = (((\mathrm{rec}\ x.\alpha x + \beta x) | \mathrm{rec}\ x.\alpha x + \gamma x) | \mathrm{rec}\ x.\alpha^{-} x) \backslash \alpha.$$

The distributed transition system $\mathbf{D}_{\mathrm{CCS}}(E) = \langle Q, T, c, \mathrm{dec}(E), 1 \rangle$ has as reachable state only the initial state $\mathrm{Q}_0 = \{p_0, p_1, p_2\}$, where

$p_0 = (((\mathrm{rec}\ x.\alpha x + \beta x) | \mathrm{id} | \mathrm{id}) \backslash \alpha$;
$p_1 = ((\mathrm{id} | \mathrm{rec}\ x.\alpha x + \gamma x) | \mathrm{id}) \backslash \alpha$;
$p_2 = (\mathrm{id} | \mathrm{rec}\ x.\alpha^{-} x) \backslash \alpha$;

and the following four transitions:

$u_0: \mathrm{Q}_0 — [\tau, \{p_1\}] \rightarrow \mathrm{Q}_0$;
$u_1: \mathrm{Q}_0 — [\beta, \{p_1, p_2\}] \rightarrow \mathrm{Q}_0$;
$u_2: \mathrm{Q}_0 — [\gamma, \{p_0, p_2\}] \rightarrow \mathrm{Q}_0$;
$u_3: \mathrm{Q}_0 — [\tau, \{p_0\}] \rightarrow \mathrm{Q}_0$.

DEFINITION 6.6 (*standard NMS and standard semantics*). Given a CCS term E, its distributed transition system $\mathbf{D}_{\mathrm{CCS}}(E)$, and the abstraction partial function **M_abstr** defined as

$$M\_\mathrm{abstr}(I_1 — [\lambda, I_3] \rightarrow I_2) = \lambda$$

(note that $\lambda \neq \tau$ and that abstr is not defined on processes/on states), the abstract interleaving (partial ordering) derived NMS with respect to M_abstr is called **standard** interleaving (partial ordering) NMS. The **standard** interleaving (partial ordering) semantics of $\mathbf{D}_{\mathrm{CCS}}(E)$ is its standard interleaving (partial ordering) NMS, defined up to bisimulation equivalence (see Definition 2.10).

LEMMA 6.1 (Milner's derivations are interleaving observations). *Given a CCS term E and its distributed transition system $\mathbf{D}_{\mathrm{CCS}}(E)$, we have that $E = s \Rightarrow E'$ iff there exists a computation of $\mathbf{D}_{\mathrm{CCS}}(E)$ from* dec(E) *to* dec(E') *with observation s in the standard interleaving NMS.*

Proof. Given a (multiple) derivation $E = E_0 — \mu_1 \rightarrow E_1 \cdots E_{n-1} — \mu_n \rightarrow E_n = E'$ such that $E = s \Rightarrow E'$, a corresponding computation of $\mathbf{D}_{\mathrm{CCS}}(E)$ can be found having the same (via dec) sequence of states and having as transition from states $\mathrm{dec}(E_i)$ and $\mathrm{dec}(E_{i+1})$ the partial ordering derivation $\mathrm{dec}(E_i) — [\mu_{i+1}, I_{i+1}] \rightarrow \mathrm{dec}(E_{i+1})$ having the same deduction structure as $E_i — \mu_{i+1} \rightarrow E_{i+1}$ has; and vice versa. ∎

THEOREM 6.3 (interleaving semantics is Milner's observational equivalence). *Given two CCS terms E and E' and their distributed transition systems $\mathbf{D}_{\mathrm{CCS}}(E)$ and $\mathbf{D}_{\mathrm{CCS}}(E')$, E is observationally equivalent to E', according to Milner's definition* [17], *iff $\mathbf{D}_{\mathrm{CCS}}(E)$ and $\mathbf{D}_{\mathrm{CCS}}(E')$ have the same standard* interleaving *semantics.*

Proof. Let t and t' be the standard interleaving NMSs of $\mathbf{D}_{\mathrm{CCS}}(E)$ and $\mathbf{D}_{\mathrm{CCS}}(E')$. We have to prove that t and t' can bisimulate each other according to our definition iff E and E' do according to Milner. In fact, whenever a step in one of the bisimulations takes place, a corresponding step can be found in the other as follows.

If a (multiple) derivation $E_0 — \mu_1 \rightarrow E_1 \cdots E_{n-1} — \mu_n \rightarrow E_n$ is given or selected in Milner's bisimulation, a corresponding computation can be found by following the same (via dec) sequence of states, and by using from states $\mathrm{dec}(E_i)$ and $\mathrm{dec}(E_{i+1})$ the transition corresponding to the partial ordering derivation having the same deduction structure as $E_i — \mu_{i+1} \rightarrow E_{i+1}$ has, and vice versa. ∎

THEOREM 6.4. *The distributed transition system $\mathbf{D}_{\mathrm{CCS}}(E)$ is commutative (with respect to M_abstr).*

Proof. A pair of adjacent transitions u_1 and u_2, with u_2 independent from u_1, are given. By Property 3.2, we can write them as

$$u_1 : H_1 \cup H_2 \cup I — [\mu_1, H_2 \cup I] \rightarrow T_1 \cup H_2 \cup I$$

and

$$u_2 : T_1 \cup H_2 \cup I — [\mu_2, T_1 \cup I] \rightarrow T_1 \cup T_2 \cup I.$$

The existence of the required pair of transitions

$$u_2' : H_1 \cup H_2 \cup I — [\mu_2, H_1 \cup I] \rightarrow H_1 \cup T_2 \cup I$$

and

$$u_1' : H_1 \cup T_2 \cup I — [\mu_1, T_2 \cup I] \rightarrow T_1 \cup T_2 \cup I$$

is guaranteed by Theorem 6.2 and Property 6.2. ∎

COROLLARY 6.1. *Distributed transition system* $\mathbf{D}_{CCS}(E)$ *is completely concurrent w.r.t.* $M_$abstr.

Proof. Follows from Theorems 6.4 and 3.3. ▮

COROLLARY 6.2. *Given a finite concurrent history* $h = \langle S, 1, \leqslant \rangle$, *let* H, I, T *be the sets of its heads, idles, and tails.*

Given the CCS term E, *let the distributed transition system* $\mathbf{D}_{CCS}(E)$ *have* t *as its standard partial ordering NMS.*

(i) *Given a CCS computation with* $E = s \Rightarrow E'$, *there exists a computation of* $\mathbf{D}_{CCS}(E)$ *having as observation the concurrent history* h, *where*

- $H \cup I = \mathrm{dec}(E)$ *and* $T \cup I = \mathrm{dec}(E')$;
- s *is a total ordering on the events of* h *compatible with* $\leqslant$.

(ii) *Given a computation of* $\mathbf{D}_{CCS}(E)$ *having as observation the finite history* h, *for every total ordering* s *on the events of* h, s *compatible with* $\leqslant$, *there exists* $E = s \Rightarrow E'$, *where* $\mathrm{dec}(E) = H \cup I$ *and* $\mathrm{dec}(E') = T \cup I$.

Proof. Follows from Theorem 3.1, since $\mathbf{D}_{CCS}(E)$ is completely concurrent with respect to $M_$abstr, and by Lemma 6.1. ▮

7. CONCLUSIONS

The paper defined distributed transition systems, a new simple model of distributed systems, and a way to observe them. Petri condition/event systems and place/transition nets, and Milner's Calculus of Communicating Systems have been translated to distributed transition systems, thus providing formal grounds for a precise comparison among different models.

Many issues have been given little attention here, since we wanted to concentrate on setting up our basic model. A first item concerns the relationships between the original theories and the new concepts introduced through the translation. An interesting point is studying the effects of the introduction of abstract observational equivalences in Petri nets, and the relationships between our partial ordering semantics for the Calculus of Communicating Systems and Milner's interleaving semantics. In the latter case, a result in [7] shows that partial ordering semantics is strictly finer than and reduces to Milner's when the parallel composition operator is not present. Another issue might be discussing synchrony versus asynchrony in our framework, e.g., comparing synchronous and asynchronous Calculi of Communicating Systems.

An important point regards the generation of distributed transition systems in a linguistic framework. Our observational equivalence must then be closed with respect to all contexts, i.e., a congruence must be defined.

A different kind of problem arises when the model is taken as the kernel of a specification language. A needed basic interpretation tool consists of growing our observations without generating computations first, using graph rewriting systems which directly manipulate concurrent histories; this was our starting point [4, 9, 27]. It is easy to prove that observing computations and growing observations give the same result, when observations do not hide any transition (Greibach rewriting rules). A method for defining limits of observations has been developed in [8], following the metric space approach.

This kernel may be extended in several directions, by adding to it orthogonal features which describe different aspects of distributed programs, as proposed in [10]. The first step consists of *generating* the rewriting rules by *synchronizing* several productions, where a production gives the specification of a stand-alone process. A second step adds information about the spatial structure of a distributed system by explicitly describing the connections among processes in a graph-like way [4, 9, 27]. A further step may introduce rule schemata parametrized on a Herbrand universe. In this way, a first notion of abstract data type can be introduced and a logic programming style can be given for specifying concurrent systems (see the last example of [10]).

ACKNOWLEDGMENTS

We wish to thank Rocco De Nicola, with whom large parts of the results in Sections 2 and 6 have been developed, and Ilaria Castellani for many interesting discussions. Ugo Montanari had the pleasure to learn about nets during two stays at GMD-ISF, with long discussions with C. A. Petri and his group. Special thanks are due to Hartmut Ehrig for his encouragement.

REFERENCES

1. A. ARNOLD AND M. NIVAT, Comportement de Processus, *in* "Les Mathématiques de l'Information, AFCET Colloq., Paris, 1982."
2. S. D. BROOKES, C. A. R. HOARE, AND A. W. ROSCOE, A theory of communicating sequential processes, *J. Assoc. Comput. Mach.* **31** (1984), 560–599.
3. M. BROY, Semantics of communicating processes, *Inform. and Control* **61** (1984), 202–246.
4. I. CASTELLANI AND U. MONTANARI, Graph grammars for distributed systems, *in* "Proceedings, 2nd Int. W. on Graph-Grammars and Their Applications to Computer Science" (H. Ehrig, M. A. Nagel, and G. Rozenberg, Eds.), Lecture Notes in Comput. Sci. Vol. 153, pp. 20–83, Springer–Verlag, New York/Berlin, 1983.
5. PH. DARONDEAU AND L. KOTT, On the observational semantics of fair parallelism *in* "Proceedings, 10th ICALP Barcelona, July 1983" (J. Diaz, Ed.), Lecture Notes in Comput. Sci. Vol. 154, pp. 147–151, Springer–Verlag, New York/Berlin, 1983.
6. P. DEGANO, R. DE NICOLA, AND U. MONTANARI, Partial ordering derivations for CCS, *in* "Proceedings, 5th Int. Conf. on Fundamentals of Computation Theory, Cottbus, September 1985" (L. Budach Ed.), Lecture Notes in Comput. Sci. Vol. 199, Springer–Verlag, New York/Berlin, pp. 520–533, 1985.
7. P. DEGANO, R. DE NICOLA, AND U. MONTANARI, Observational equivalences for concurrency models, *in* "Proceedings, IFIP Working Conference on the Formal Description of Programming Concepts, Ebberup, August 1986," (M. Wirsing, Ed.), North-Holland, Amsterdam.

8. P. DEGANO AND U. MONTANARI, Liveness properties as convergence in metric spaces, *in* "Proceedings, 16th Annual ACM SIGACT Symp. on Theory of Computing, Washington, DC, 1984," pp. 31–38.
9. P. DEGANO AND U. MONTANARI, Distributed systems, partial ordering of events, and event structures, *in* "Control Flow and Data Flow: Concepts of Distributed Programming" (M. Broy, Ed.), NATO ASI Series F., Vol. 14, pp. 7–106, Springer–Verlag, New York/Berlin, 1985.
10. P. DEGANO AND U. MONTANARI, Specification languages for distributed systems, *in* "Proceedings, TAPSOFT, Berlin, March 1985" (H. Ehrig, C. Floyd, M. Nivat, and J. Thatcher, Eds.), pp. 29–51, Lecture Notes in Comput. Sci. Vol. 185, Springer–Verlag, New York/Berlin, 1985.
11. H. J. GENRICH, K. LAUTENBACH, AND P. S. THIAGARAJAN, Elements of general net theory, *in* "Net Theory and Applications" (W. Brauer, Ed.), Lecture Notes in Comput. Sci. Vol. 84, Springer–Verlag, New York/Berlin, 1980.
12. U. GOLTZ AND A. MYCROFT, On the relationships of CCS and Petri nets, *in* "Proceedings, 11th ICALP, Antwerp, July 1984" (J. Paredaens, Ed.), Lecture Notes in Comput. Sci., Vol. 172, Springer–Verlag, New York/Berlin, pp. 196–208, 1984.
13. U. GOLTZ AND W. REISIG, The non-sequential behaviour of Petri nets, *Inform. and Control* **57** (1983), 125–147.
14. L. LAMPORT, Time, clocks and the ordering of events in a distributed system, *Comm. ACM* **12** (1978), 558–564.
15. P. E. LAUER, P. R. TORRIGIANI, AND M. W. SHIELDS, COSY: A specification language based on path expressions, *Acta Inform.* **12** (1979), 109–158.
16. A. MAZURKIEWICZ, Concurrent program schemas and their interpretation, *in* "Proceedings, Aarhus Workshop on Verification of Parallel Programs," 1977.
17. R. MILNER, Notes on a Calculus for Communicating Systems, *in* "Control Flow and Data Flow: Concepts of Distributed Programming" (M. Broy, Ed.), NATO ASI Series F, Vol. 14, pp. 205–228, New York/Berlin, Springer–Verlag, 1985.
18. M. NIVAT, Behaviors of processes and synchronized systems of processes, *in* "Theoretical Foundations of Programming Methodology" (M. Broy and G. Schmidt, Eds.), pp. 473–550, Reidel, Dordrecht, 1982.
19. D. PARK, Concurrency and automata on infinite sequences, Lecture Notes in Comput. Sci. Vol. 104, Springer–Verlag, New York/Berlin, 1981.
20. D. PARK, The "fairness" problem and nondeterministic computer networks, *in* "Foundations of Computer Science IV" (J. W. deBakker and J. vanLeeuwen, Eds.), Math. Centrum Tracts, Amsterdam, 1981.
21. W. REISIG, "Petri Nets: An Introduction," EACTS Monographs on Theoretical Computer Science, Springer–Verlag, New York/Berlin, 1985.
22. M. W. SHIELDS, "Non-sequential behaviour I," Internal Report CSR-120-82, Dept. of Computer Science, University of Edinburgh, 1982.
23. M. B. SMYTH, Power domains, *J. Comput. System. Sci.* **16** (1978), 23–36.
24. J. WINKOWSKI, Behaviors of concurrent systems, *Theoret. Comput. Sci.* **12** (1980), 39–60.
25. G. WINSKEL, Event structure semantics for CCS and related languages, *in* "Proceedings, 9th ICALP" (M. Nielsen and E. M. Schmidt, Eds.), Lecture Notes in Comput. Sci. Vol. 140, pp. 561–567, Springer–Verlag, New York/Berlin, 1982.
26. G. WINSKEL, Synchronization trees, *in* "Proceedings, 10th ICALP, Barcelona, July 1983" (J. Diaz, Ed.), Lecture Notes in Comput. Sci. Vol. 154, pp. 695–771, Springer–Verlag, New York/Berlin, 1983.
27. P. DEGANO AND U. MONTANARI, A model of distributed system based on graph rewriting, *J. Assoc. Comput. Mach.* **34** (1987).