Università di Pisa

Dipartimento di Informatica

Technical Report : TR-96-28

Compact Transition Systems

Pierpaolo Degano, Corrado Priami

July 09, 1996

ADDR: Corso Italia 40,56125 Pisa,Italy. TEL: +39-50-887111. FAX: +39-50-887226

Compact Transition Systems *

Pierpaolo Degano and *Corrado Priami*
Dipartimento di Informatica, Università di Pisa
Corso Italia, 40, I-56125 Pisa, Italy
{degano,priami}@di.unipi.it

Abstract

A major drawback of representing concurrent systems as transition systems is the exponential size of their semantic representations. We present a way of obtaining compact transition systems that are mildly exponential in average. The idea is to have (at least) one computation to represent all the ones only differing for the temporal ordering in which concurrent transitions occur. We characterise through axioms the transition systems which may be reduced in size, still preserving non interleaving bisimulation equivalences. We also exemplify our reduction technique on a process calculus whose compact transition system is generated in *SOS* style.

1 Introduction

The behaviour of concurrent systems is conveniently given by an operational approach based on *transition systems*. A generally accepted assumption is that transitions represent occurrences of events and that the execution of a concurrent system is represented by interleaving transitions. The previous assumption leads to transition systems that have $\mathcal{O}(n^n)$ states and transitions, where n is the number of the events. This is the *state explosion* problem and it is a major impediment for a mechanical verification of processes, commonly performed on labelled transition systems [6].

In this paper we overcome the problem of the state explosion for a particular class of transition systems with independence [17] yielding a *compact* version of them. Like almost all the proposals presented in the literature

*Work partially supported by ESPRIT Basic Research Action n. 8130 - LOMAPS.

[7, 14, 15, 27, 22], we exploit a relation of concurrency among the transitions enabled in a given state. The basic idea is to give an ordering in which concurrent transitions are sequentially fired. A single computation is often sufficient to represent all the ones obtained by permutation of concurrent transitions, especially when non time dependent aspects of systems (e.g., causality, locality, enabling, etc.) are of interest. Hence, in order to get compact representations, the first step is to identify such computations. As an example, the *CCS*-like process $a.b \,|\, (c + d)$ originates two sets of maximal computations, one with c and the other with d, concurrent with a that in turn causes b. All the computations in a set are equivalent under any sensible non interleaving aspect.

The next step consists in choosing a representative for each of these equivalence classes. For instance, in the example above one may choose the paths in which a precedes b which in turn precedes c, and the one with a before b before d. In this way, only one transition is kept out of the three outgoing from the initial state. Again, out of the three transitions from the configuration reached, only one is kept. Finally, two more transitions are possible leading to the same final state. In this example the compact transition system has 4 transitions instead of 10.

We are mainly interested in checking non interleaving equivalences based on bisimulation. They essentially consider the semantics of concurrency as primitive, i.e. not semantically expressible by any combination of other relations. We characterize this intuition. For this purpose, we postulate some additional axioms. Basically we assume that firing an event does not prevent a later occurrence of the ones concurrent with it. Furthermore, we show how the complete transition system can be retrieved from its compact representative. Finally, we consider calculi for concurrency that intentionally express our transition systems, and we equip them with an *SOS* semantics that directly yields compact transition systems.

Compact transition systems have $\mathcal{O}(n^{\log \log n})$ transitions in average, where n is the number of events. Thus, state explosion is reduced to have a mild exponential complexity. (A polynomial of degree 9 dominates it when there are 10^{80} events (estimated number of atoms), and of degree 4 when there are 2^{16} events.) Actually, the complexity decreases as the number of concurrent events increases. For example, the compact representation of a system with n (pairwise) concurrent events has exactly n transitions. Formal evidence is then given that non interleaving representations may be more economic than the interleaving ones. In particular, non interleaving bisimulations can be checked in $O(n^{\log \log n})$ time in average. Needless to say, any simplification depending on the selected bisimulation, see [6, 13, 9], is still applicable to compact transition systems, with a further reduction of the number of their transitions. The same is true for any smart representation of transition systems like BDDs [5].

Compact transition systems have even greater benefits when starting from a linguistic level. In this case, to avoid waste of time we derive on the fly the compact transition systems of the processes under investigation. First we consider a subset of *CCS* [19] introduced by [20] whose only operators are prefix, choice and parallel composition without communication. Adding recursion creates no problem. Communication requires some additional axioms on transition systems with independence, that are satisfied by (higher-order, name-passing) calculi with no scope operator.

We define the operational semantics of our calculus in *SOS* style. We rely on proved transition systems [11, 12] whose transitions are labelled by encodings of their deduction trees. This additional information expresses also the concurrency relation, making the explicit definition of independence useless. The compact transition system generated in this way is equivalent to the one obtained by the reduction above. As a matter of fact, all the results about bisimulation and complexity sketched above still hold. A prototype to check bisimulation based on these ideas runs on Macintosh and is described in [2].

Some related work similar to ours is sketched below. In [14, 15] the problem is addressed of finding compact representation of processes for studying safety properties like absence of deadlock, but not bisimulation equivalences. The random selection of the run to be kept leads to an efficient solution. The solution by [7, 26] represents processes by exploiting the properties of independence and mutual exclusion of events, as in our case. These representatives preserve the failure *CSP* semantics [4]; however, nothing is said about bisimulations. Also [27] reduces labelled transition systems, and preserves an equivalence based on the notions of stable failure and divergence. The used semantic model conforms with the semantics of *CSP* up to the first divergence of the system. A predefined relation of independence of events is used in [22] for constructing a transition system that contains at least one representative path among those obtained by interleaving its independent events. A tool is implemented, yielding a state graph on which model checking may be performed efficiently. Some methods propose reductions on Petri nets that permit efficient checking of deadlocks and liveness properties. Among them, the method presented by [16] has been developed for a particular class of Petri nets. The simplification relies on a kind of reachability relation on computations. The approach of [18] is based on unfoldings of Petri nets that make it easier to establish reachability. In the solution by [23] the user must supply a pomset grammar which describes the behaviour of the considered net.

The paper is organized as follows. In the next section we recall transition systems with independence [17], and we impose further restrictions on them. Subsection 2.1 describes how to reduce the size of this kind of transition systems without loosing any information on their non-interleaving behaviour. In Subsection 2.2 we estimate the complexity of the size of compact transition systems and of the time needed to obtain them. Eventually, we characterize non interleaving bisimulations in Subsect. 2.3, and we prove that our reduction preserves them. Given a calculus for concurrency, Sect. 3 describes how to derive its compact transition system. Finally, Sect. 4 integrates the previous sections to cope with communications.

2 Compact transition systems

The main idea of our approach to limit state explosion is selecting a representative for all computations which differ only in the temporal ordering in which their concurrent transitions fire. First, we recall transition systems with independence.

A *transition system* is a quadruple $\langle \Gamma, A, \longrightarrow, P \rangle$, where Γ is a set of configurations ranged over be $P, \gamma, \gamma', \ldots$; A is a labelling alphabet with metavariables $a, b, \ldots$; $\longrightarrow \subseteq \Gamma \times A \times \Gamma$ is the transition relation; and P is the start state. A *computation* is a path in the transition system. Also, we write $Tr(\gamma)$ for the transitions with source γ.

If a transition system is equipped with a symmetric, irreflexive relation $\smile \subseteq \longrightarrow \times \longrightarrow$, often called *concurrency*, we get a *transition system with independence*

$$TS = \langle \Gamma, A, \longrightarrow, P, \smile \rangle.$$

These transition systems have been introduced by [28], following [1, 25, 24]. Hereafter, we restrict ourselves to finite state and finite branching *TS*s.

A formal characterization of these transition systems is given by the following axioms [17]:

$$\gamma_0 \xrightarrow{a} \gamma_0' \sim \gamma_0 \xrightarrow{a} \gamma_1 \;\Rightarrow\; \gamma_0' = \gamma_1 \tag{1}$$

$$\gamma_0 \xrightarrow{a} \gamma_0' \smile \gamma_0' \xrightarrow{b} \gamma_0'' \;\Rightarrow \tag{2}$$
$$\exists \gamma_1 . \gamma_0 \xrightarrow{a} \gamma_0' \smile \gamma_0 \xrightarrow{b} \gamma_1 \;\wedge\; \gamma_0 \xrightarrow{b} \gamma_1 \smile \gamma_1 \xrightarrow{a} \gamma_0''$$

$$\gamma_0 \xrightarrow{a} \gamma_0' \sim \gamma_1 \xrightarrow{a} \gamma_1' \smile \gamma_2 \xrightarrow{b} \gamma_2' \;\Rightarrow\; \gamma_0 \xrightarrow{a} \gamma_0' \smile \gamma_2 \xrightarrow{b} \gamma_2' \tag{3}$$

where $\sim$ is the least equivalence relation including $<$ defined by

$$\gamma_0 \xrightarrow{a} \gamma_0' < \gamma_1 \xrightarrow{a} \gamma_1' \;\Leftrightarrow$$
$$\exists b . \gamma_0 \xrightarrow{a} \gamma_0' \smile \gamma_0 \xrightarrow{b} \gamma_1 \;\wedge \gamma_0 \xrightarrow{a} \gamma_0' \smile \gamma_0' \xrightarrow{b} \gamma_1' \;\wedge$$
$$\gamma_0 \xrightarrow{b} \gamma_1 \smile \gamma_1 \xrightarrow{a} \gamma_1'$$

The relation $\sim$ says when two transitions are occurrences of the same event. [1] Axiom (1) states that the occurrence of an event in a given state drives to a unique state. Axiom (2) (see Fig. 1) shows the intuitive property that two concurrent transitions which occur consecutively can also occur in the reverse order. Axiom (3) extends $\smile$ to events.

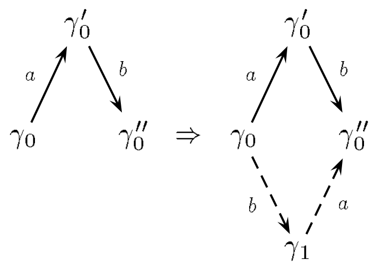


Figure 1: Consecutive concurrent transitions occur in any order.

We impose two sensible restrictions on asynchronous transition systems. We assume that given a finite set of (pairwise) concurrent transitions, there is a computation in which they occur consecutively. This assumption is made precise by the axiom

$$\forall i \neq j,\, 0 \leq i, j < n \tag{4}$$
$$(\gamma_i \xrightarrow{a_i} \gamma_i' \smile \gamma_j \xrightarrow{a_j} \gamma_j' \;\Rightarrow \quad \exists\, \tilde{\gamma}_i \xrightarrow{a_i} \tilde{\gamma}_{i+1} \sim \gamma_i \xrightarrow{a_i} \gamma_i')$$

[1] An event may be seen as the firing of a prefix of a language or as the firing of a condition in a C/E net. Thus many "interleaving" transitions can be associated to the same event.

An instance of Axiom (4) with $n = 2$ and $\gamma_i' = \gamma_j$ shows that the firing of a transition does not prevent the execution of the transitions concurrent with it. This instance is called *forward stability* in [1]. An immediate consequence of its statement is depicted in Fig. 2. Note that uniqueness of γ_2 below follows from Axiom (1).

Proposition 2.1 (forward stability)

$$\gamma_0 \xrightarrow{a} \gamma_0' \smile \gamma_0 \xrightarrow{b} \gamma_1' \;\Rightarrow\; \exists! \gamma_2 \,.\, \gamma_0' \xrightarrow{b} \gamma_2 \sim \gamma_0 \xrightarrow{b} \gamma_1'$$

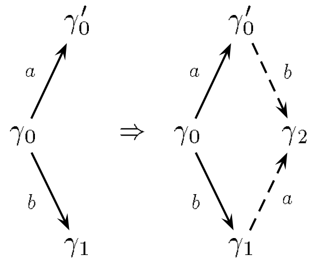


Figure 2: Forward stability.

Next, we assume that $\not\prec$ restricted to a computation represents an enabling relation in the sense of Petri net theory. Formally, given three transitions $\gamma_i \xrightarrow{a_i} \gamma_i'$ of a computation

$$\gamma_0 \xrightarrow{a_0} \gamma_0' \not\prec \gamma_1 \xrightarrow{a_1} \gamma_1' \not\prec \gamma_2 \xrightarrow{a_2} \gamma_2' \;\Rightarrow\; \gamma_0 \xrightarrow{a_0} \gamma_0' \not\prec \gamma_2 \xrightarrow{a_2} \gamma_2' \tag{5}$$

The following theorem will help later. It states that given two concurrent transitions in a computation, there exists another computation with the same source and target in which the two transitions occur in reverse order.

Theorem 2.2 (permutation of transitions)
Let $\gamma_0 \xrightarrow{a_0} \gamma_1 \xrightarrow{a_1} \ldots \gamma_n \xrightarrow{a_n} \gamma_{n+1}$ *be a computation with* $\gamma_0 \xrightarrow{a_0} \gamma_1 \smile \gamma_n \xrightarrow{a_n} \gamma_{n+1}$. *Then, there exists a permutation of indexes* $\sigma : [0..n] \to [0..n]$ *and a computation*

$$\gamma_0 \xrightarrow{a_{\sigma(h)}} \gamma_1' \xrightarrow{a_{\sigma(l)}} \ldots \gamma_n' \xrightarrow{a_{\sigma(k)}} \gamma_{n+1}$$

such that $\exists i \in [0..n]$ *and*

- $\sigma(0) = i+1,\ \sigma(n) = i$; *and*
- $\sigma(j) = j-1$ *for* $0 < j \leq i$; *and*
- $\sigma(m) = m+1$ *for* $n > m \geq i+1$

PROOF. The proof is by induction on n. If $n = 1$, the statement is trivially true by Axiom 2. Assume as inductive hypothesis that the proposition holds $\forall\, k \leq n$. Now, let $k+1$ transitions occur between $\gamma_0 \xrightarrow{a_0} \gamma_1$ and $\gamma_n \xrightarrow{a_n} \gamma_{n+1}$, and let h be the minimum index such that $\gamma_0 \xrightarrow{a_0} \gamma_1 \smile \gamma_h \xrightarrow{a_h} \gamma_{h+1}$. Assume $h < n$; if $h = n$ we only need a trivial adjustment to indexes. For any $l < h$ it is $\gamma_l \xrightarrow{a_l} \gamma_{l+1} \smile \gamma_h \xrightarrow{a_h} \gamma_{h+1}$. In fact, if $\gamma_l \xrightarrow{a_l} \gamma_{l+1} \not\smile \gamma_h \xrightarrow{a_h} \gamma_{h+1}$, by Axiom (5), it would be $\gamma_0 \xrightarrow{a} \gamma_1 \not\smile \gamma_h \xrightarrow{a_h} \gamma_{h+1}$ against h being minimum. Then, by Axiom (2) there exists the computation

$$\gamma_0 \xrightarrow{a_0} \gamma_1 \xrightarrow{a_1} \gamma_2 \cdots \gamma_{h-1} \xrightarrow{a_h} \gamma_h^1 \xrightarrow{a_{h-1}} \gamma_{h+1} \xrightarrow{a_{h+1}} \gamma_{h+2} \cdots \gamma_{n-1} \xrightarrow{a_n} \gamma_n$$

Repeat the above h times to obtain the computation

$$\gamma_0 \xrightarrow{a_h} \gamma_1^h \xrightarrow{a_0} \gamma_2^h \cdots \gamma_{n-1} \xrightarrow{a_n} \gamma_n.$$

Since there are k transitions between $\gamma_1^h \xrightarrow{a_0} \gamma_2^h$ and $\gamma_{n-1} \xrightarrow{a_n} \gamma_n$, the inductive hypothesis suffices. $\square$

We further restrict the class of transition systems by stipulating the axiom below (see Fig. 3). It says that concurrency does not operate choices. In other words, when two concurrent transitions can occur, the set of transitions enabled by firing either of them does not change (apart from the fired one). When some transitions are designated to represent the interaction of more elementary actions, say communications, the interplay between choice and interaction requires more attention. We will come on this issue in Section 4.

$$\gamma \xrightarrow{a} \gamma_0 \smile \gamma_0 \xrightarrow{b} \gamma_1 \not\smile \gamma_0 \xrightarrow{c} \gamma_0' \Rightarrow \gamma \xrightarrow{a} \gamma_0 \smile \gamma_0 \xrightarrow{c} \gamma_0' \qquad (6)$$

2.1 Reduction

We show how to extract a canonical compact representative from a given whole transition system in such a way that non interleaving equivalences are preserved and checkable efficiently. The basic idea is to give an ordering in

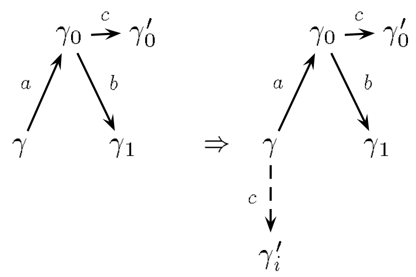


Figure 3: Concurrency does not operate choices.

which concurrent transitions are sequentially fired. Indeed all the computations that differ only in the ordering in which concurrent transitions occur convey the same information on the non interleaving aspects of a system. To preserve bisimulation equivalences, we must choose *deterministically* the the *leader* transitions to be kept out of those concurrent and outgoing from the same configuration γ.

We assume below a lexicographic ordering $\triangleleft_l$ on labels and we define the induced pre-ordering on the transitions outgoing from the same configuration as

$$\gamma_0 \xrightarrow{a} \gamma_0' \triangleleft \gamma_0 \xrightarrow{b} \gamma_1 \;\Leftrightarrow\; a \triangleleft_l b \,\vee\, \gamma_0 \xrightarrow{b} \gamma_1 \longrightarrow^* \gamma_0 \tag{7}$$

Note that $\triangleleft$ is a pre-ordering because there can be many transitions exiting from the same configuration that share the same label (*autoconcurrent transitions*) or that originate a loop. Consider for instance the transition system in Fig. 4 where the initial state is the source of two a-transitions. As it will be clear later, taking as leader either one or the other a-transition originates two different compact transition systems (the one in full color and the one with dashed transitions). Actually, it is necessary to keep them all if we are interested in preserving bisimulation equivalences.

The second condition in the definition above states that transitions originating a loop are greater than all the others. This is for postponing their selection, and ensuring fairness. For instance, without the condition above, in the *CCS*-like process $(rec\, x\; a.x) \,|\, b$, the transition labelled b could never be generated. Note that $\triangleleft$ only extends $\triangleleft_l$ with the condition on loops and therefore is total, because our transition systems are finite state.

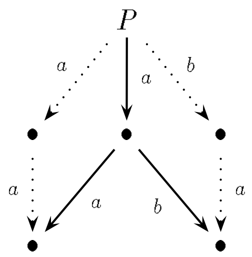


Figure 4: A transition system with autoconcurrency.

Let $TS = \langle \Gamma, A, \longrightarrow, P, \smile \rangle$ be a transition system with independence. Then, let the auxiliary function $\mathcal{C}$ be defined as

$$\begin{aligned} \mathcal{C}(Tr(\gamma)) = & \, min_{\triangleleft}(Tr(\gamma)) \cup \\ & \mathcal{C}(\{\gamma \xrightarrow{a} \gamma' \not\smile \gamma \xrightarrow{b} \gamma'' \in min_{\triangleleft}(Tr(\gamma))\}) \end{aligned} \tag{8}$$

Function $\mathcal{C}$ is well-defined because it is applied to finite state and finite branching transition systems. It (recursively) discards the transitions which are concurrent with the (current) leaders and keeps instead those which are not. Note that a transition $\hat{\gamma}_0 \xrightarrow{b} \hat{\gamma}_1$ equivalent to a discarded one $\gamma_0 \xrightarrow{b} \gamma_1$ will occur after a leader $\gamma_0 \xrightarrow{a} \gamma_0'$, due to (iterated applications of) forward stability. Starting from the initial configuration, a transition $P \xrightarrow{b} \gamma \notin \mathcal{C}(Tr(P))$ is removed from the whole transition system, together with the subgraph G reached only through it. Then, the reduction is iteratively applied to the configurations which are targets of computations with source P and made up of the transitions kept so far.

The compact transition system of TS is $cTS = \langle \Gamma', A, \hookrightarrow, P, \smile /_{\hookrightarrow} \rangle$ where

$$\hookrightarrow = \bigcup_{\gamma \in \Gamma} \{\mathcal{C}(Tr(\gamma)) \mid P \hookrightarrow^* \gamma\} \quad \text{and} \quad \Gamma' = \{\gamma \in \Gamma \mid P \hookrightarrow^* \gamma\}.$$

The following property is immediate from the definition of $\mathcal{C}$.

Proposition 2.3 *Given a transition system* TS*, its compact version* cTS *is uniquely and finitely generated.*

Proof. Reduction is terminating as the size of TS decreases at each step. Indeed the relation $\not\smile$ is irreflexive and TS is finite state and finite branching thus the same configuration is processed only once. Moreover, cTS is unique because the computation of $\not\smile$ and of $min_{\lhd}(Tr(\gamma))$ is deterministic. $\square$

We claim that no information on the behaviour of a process is lost during the generation of its compact representation. Indeed, it is possible to rebuild the whole transition system from its compact one.

Theorem 2.4 *Let* $\mathrm{TS} = \langle \Gamma, A, \longrightarrow, P, \smile \rangle$, *and let* $\mathrm{cTS} = \langle \Gamma', A, \hookrightarrow, P, \smile' \rangle$ *be its compact transition system. Then,*

$$\forall\, \gamma_0 \xrightarrow{a} \gamma_0' \in \mathrm{TS}, \exists\, \tilde{\gamma}_0 \xrightarrow{a} \tilde{\gamma}_0' \in \mathrm{cTS} \,.\, \gamma_0 \xrightarrow{a} \gamma_0' \sim \tilde{\gamma}_0 \xrightarrow{a} \tilde{\gamma}_0'$$

Proof. Consider a transition $\gamma_i \xrightarrow{a} \gamma_i'$ of TS not occurring in cTS. We distinguish two cases according to γ_i. If $\gamma_i \in \Gamma'$, then $\exists\, \gamma_i \overset{c}{\hookrightarrow} \gamma_j \in cTS$ with $\gamma_i \xrightarrow{a} \gamma_i' \smile \gamma_i \xrightarrow{c} \gamma_j$ by definition of $\mathcal{C}$. Now, either $\gamma_j \overset{a}{\hookrightarrow} \gamma_j' \sim \gamma_i \xrightarrow{a} \gamma_i'$, or we are in the previous case. Since TS is finite state and finite branching, this case is proved.
The other case is $\gamma_i \notin \Gamma'$. Consider the minimal predecessor $\gamma \in cTS$ of γ_i such that the computation

$$P \xrightarrow{s}{}^{*} \gamma \xrightarrow{b} \gamma' \xrightarrow{s'}{}^{*} \gamma_i$$

is in TS. Since $\gamma \xrightarrow{b} \gamma' \notin cTS$, there must exist $\gamma \overset{c}{\hookrightarrow} \hat{\gamma}$ with $\gamma \xrightarrow{b} \gamma' \smile \gamma \xrightarrow{c} \hat{\gamma}$. We proceed by induction on the number of transitions between γ and γ_i.
Base case. There is a single transition between γ and γ_i ($\gamma' = \gamma_i$). If $\gamma_i \xrightarrow{a} \gamma_i' \smile \gamma \xrightarrow{b} \gamma'$, by Axiom (2) we have $\gamma \xrightarrow{a} \tilde{\gamma} \sim \gamma_i \xrightarrow{a} \gamma_i'$. Thus, we are in the previous case. For the case $\gamma_i \xrightarrow{a} \gamma_i' \not\smile \gamma \xrightarrow{b} \gamma'$, we examine the relations between the c and the a transition. By forward stability there exists $\gamma' \xrightarrow{c} \hat{\gamma}'$ in TS such that $\gamma' \xrightarrow{c} \hat{\gamma}' \sim \gamma \xrightarrow{c} \gamma'$. If $\gamma' \xrightarrow{a} \gamma_i' \smile \gamma' \xrightarrow{c} \hat{\gamma}'$, we are done because $\hat{\gamma} \xrightarrow{b} \hat{\gamma}' \sim \gamma \xrightarrow{b} \gamma'$ will eventually appear in cTS. Otherwise by Axiom (6) $\gamma \xrightarrow{a} \tilde{\gamma} \sim \gamma_i \xrightarrow{a} \gamma_i'$ and we are again in the previous case.
Assume that the theorem holds when there are n transitions between γ and γ_i labelled $b_1, \ldots, b_n$. Now apply the arguments above to the transition labelled b_i. This allows us to reach a state in which the distance between γ and γ_i decreases to $n-1$ and the inductive hypothesis applies. $\square$

Theorem 2.4 gives us the means to retrieve a whole transition system from its compact representation.

Theorem 2.5 *Let* $\mathrm{TS} = \langle \Gamma, A, \longrightarrow, P, \smile \rangle$*, and let* $\mathrm{cTS} = \langle \Gamma', A, \hookrightarrow, P, \smile' \rangle$ *be its compact transition system. Then, it is always possible to rebuild (the part reachable from* P*) of* TS *from* cTS.

Proof. The transition system TS is finitely obtained by the following fix-point construction. We define a function Ψ that only rebuilds the diamonds of concurrent transitions. We apply Ψ until its fix-point is reached.

We build a sequence of transition systems $TS_i = \langle \Gamma_i, A, \xrightarrow{a}_i, P, \smile_i \rangle$ with $\Gamma_i \subseteq \Gamma_{i+1}$, $\xrightarrow{a}_i \subseteq \xrightarrow{a}_{i+1}$ and $\smile_i \subseteq \smile_{i+1}$ (we abbreviate the above conditions by saying $TS_i \subseteq TS_{i+1}$) starting with $TS_0 = cTS$. These conditions introduce a family of equivalence relations $\sim_i$ defined according to $\smile_i$ such that, by Theorem 2.4, $\sim_i \subseteq \sim$ and $\smile_i \subseteq \smile$.
Function $\Psi : TS_i \rightarrow TS_{i+1}$ is such that

$$\begin{aligned}&\forall \gamma_0 \xrightarrow{a}_i \gamma_1 \smile_i \gamma_1 \xrightarrow{b}_i \gamma_2 \;\Rightarrow \\ &\quad (\gamma_0 \xrightarrow{b}_{i+1} \gamma_1' \sim_{i+1} \gamma_1 \xrightarrow{b}_i \gamma_2) \wedge (\gamma_1' \xrightarrow{a}_{i+1} \gamma_2 \sim_{i+1} \gamma_0 \xrightarrow{a}_i \gamma_1)\end{aligned}$$

according to Axiom (2), and

$$\smile_{i+1} = \smile_i \cup \langle \gamma_0 \xrightarrow{b}_{i+1} \gamma_1', \gamma_0 \xrightarrow{a}_i \gamma_1 \rangle, \langle \gamma_0 \xrightarrow{b}_{i+1} \gamma_1', \gamma_1' \xrightarrow{a}_{i+1} \gamma_2 \rangle$$

according to the definition of $\sim$. An application of Ψ only adds nodes and transitions already present in TS, because of Axiom (2). Since our transition systems are finite state and finite branching, Ψ is strictly monotonic (it increases the cardinality of Γ_i (if $\gamma_1' \notin \Gamma_i$), and strictly increases the cardinality of $\longrightarrow_i$). Also, $\forall n \geq 0$, $\Psi^n(cTS) \subseteq TS$, and Ψ has a unique fixed-point $\overline{TS} \subseteq TS$.
To prove that $\overline{TS} \supseteq TS$ follow the same pattern of the proof of Theorem 2.4.
$\square$

We report an example of application of the transformation Ψ introduced in the proof above. Consider the cTS in Fig. 5(a) with $\overline{a} \smile a \smile b$. Figure 5(b), shows $\Psi^1(cTS)$ where the diamond between a and b is restored. After a further step, we get the whole transition system depicted in part (c).

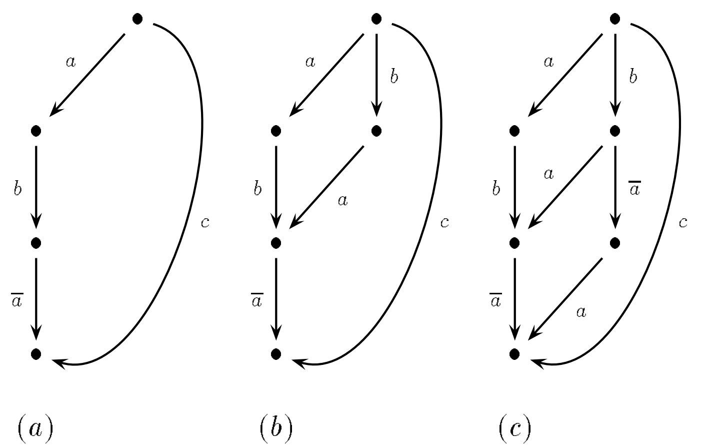


Figure 5: Reconstruction of the full transition system.

2.2 Complexity estimation

In this sub-section we estimate space and time complexity of compact transition systems. We cannot make any claim on the minimality of our reduction, as it will be shown at the end of Sect. 3.

Theorem 2.6 *Let* TS *be a transition system with n equivalence classes originated by $\sim$. Then, the* average space complexity *of the* cTS *is* $\mathcal{O}(n^{\log\log n})$.

Proof. Since our transition systems are finite state and finite branching with initial state P, the maximal length of acyclic paths is at most n. Also, we can assume that

(A1) because of forward stability there exists an integer $0 < p < n$ such that

$$|N_i| < |N_{i+1}| \text{ for } i < p$$

and

$$|N_i| > |N_{i+1}| \text{ for } i \geq p$$

where N_i is the number of nodes reached by the acyclic computations from P of length i;

(A2) any of the k_γ transition with source γ has $k_\gamma/2$ transitions concurrent with it.

Note that transitions originating loops do not appreciably contribute to the number of transitions, because their choice is delayed as much as possible according to $\lhd$. As a consequence, hereafter we may think of our graphs as acyclic.
We estimate first the number p and then the order of magnitude of the transitions in cTS, assuming that our transition systems have the following structure. If the initial state P has out-degree $k = max_\Gamma\{k_\gamma\}$, the configurations reached from P after one transition will have about $\frac{3}{4}k$ as out-degree in average. This follows from assumption (A2) and from the fact that in average only half of the transitions non-concurrent with a leader appear immediately after it. Hence, the number of transitions in half graph is

$$\sum_{i=0}^{p}\prod_{j=0}^{i} k(\frac{3}{4})^j \tag{9}$$

By assumption (A1), we must stop computing the summation above when p is such that $\lceil k(\frac{3}{4})^p\rceil = 1$. Since $p = \lceil\frac{\log k}{\log 3/4}\rceil$ and k is $\mathcal{O}(n)$, p is $\mathcal{O}(\log n)$. Now, we estimate the number of transitions in the compact transition system. Let $\theta \in Tr(P)$ be a leader, that belongs to $\hookrightarrow$. Then, we proceed recursively with the $k/2$ transitions not concurrent with θ (assumption (A2)), until only 1 transition is left. Thus the number of transitions in $\hookrightarrow$ exiting from P is $\log k$. Hence, the only difference with equation (9) is that $\log k$ replaces k. An upper bound to the total number of transitions is

$$\sum_{i=0}^{p}\prod_{j=0}^{i} \log k,$$

which is of order $\mathcal{O}(\log k)^p$. Therefore, the order of the number of transitions in cTS is

$$\mathcal{O}((\log n)^{\log n}) = \mathcal{O}(n^{\log\log n}).$$

□

The following theorem establishes exactly how many transitions occur in cTS, assuming that $\smile$ is empty or there is no autoconcurrency (concurrent transitions have distinct labels) and $\smile$ holds for any pair of transitions exiting from the same configuration. (Note that the second condition amounts to say that the system is deterministic). Furthermore, we establish also the worst space complexity of our representations: it occurs when all concurrent transitions share the same label and it is the same of the complete transition system.

Theorem 2.7 *Let* TS *be a transition system with* n *equivalence classes originated by* $\sim$. *Then,* if

1. either $\smile = \emptyset$ or
   $\smile$ *is such that* $\forall \gamma \, . \, \gamma \xrightarrow{a} \gamma' \smile \gamma \xrightarrow{b} \gamma'' \Rightarrow a \neq b$ *and* $\not\frown /_{Tr(\gamma)} = \emptyset$,
   then *the* cTS *has exactly* n *transitions;*

2. $\forall \gamma \, . \, \gamma \xrightarrow{a} \gamma' \smile \gamma \xrightarrow{b} \gamma'' \Rightarrow a = b$, then cTS $=$ TS.

PROOF. We start with item *1.* If $\smile = \emptyset$, *TS* has exactly n transitions because the equivalences classes of $\sim$ are singletons.
In the other case, all transitions outgoing from a node are concurrent and have distinct labels. Thus only one is kept in *cTS*.

Item *2.* According to the preordering $\triangleleft$ we discard no transition. □

There are cases in which the hypotheses of Theorem 2.7, item *1*, do not apply, but the compact transition system has exactly n transitions as well. Consider the *TS* depicted in Fig. 6, where the a-transitions are not concurrent with the b-transitions, and all the other pairs are in $\smile$. (The labels of some transitions are omitted for the sake of readability, assuming that "parallel" arcs belong to the same $\sim$-equivalence class.) The transitions of *cTS* are the ones in full color in the figure. There exists also a (bisimilar) *TS*, with the same relations between transitions, in which the configurations target of the transitions labelled a and b are duplicated (see Fig. 7). Its corresponding *cTS* contains two occurrences (out of six) of the $c-$ and the d-transition (Fig. 7, full color).

We now estimate the generation time of *cTS*. We assume below that the comparison of two labels has a unitary cost, as well as checking whether two transitions are in the $\smile$ relation.

Theorem 2.8 *Let* TS *be a transition system with* n *equivalence classes originated by* $\sim$. *Then,*

1. *the average generation time of* cTS *is* $\mathcal{O}(n^{\log \log n})$*;*

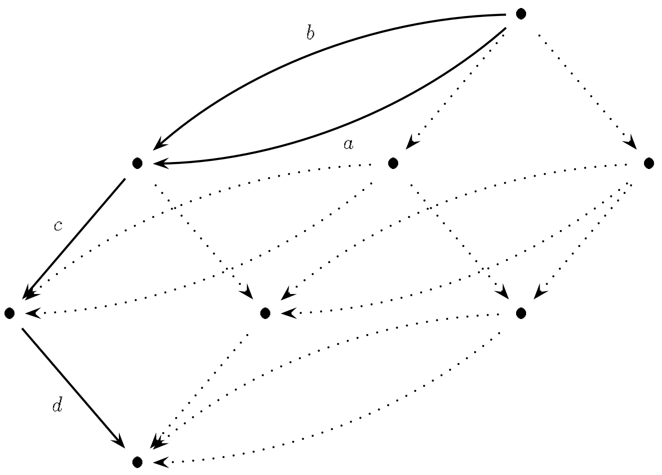


Figure 6: An optimum case of space complexity.

2. if either $\smile = \emptyset$ or

$\smile$ *is such that* $\forall \gamma \,.\, \gamma \xrightarrow{a} \gamma' \smile \gamma \xrightarrow{b} \gamma'' \;\Rightarrow a \neq b$ *and* $\not\smile /_{Tr(\gamma)} = \emptyset$,

then *the generation time of* cTS *is* $\log n$.

Proof. Item *1.* The selection of the leader θ out of k transitions requires $\log k$ time. Then, we must iterate this selection for the subset of the $k/2$ transitions which are not concurrent with θ. An upper bound to the iteration of such a selection is again $\log k$. As a consequence, the transition to be kept among those exiting from the same node are determined in $\log^2 k$ time. From Theorem 2.6, we know that the above procedure is applied $\log n$ times. Hence, an upper bound to the average generation time is given by

$$\sum_{i=0}^{\log n} \prod_{j=0}^{i} \log^2 k$$

The generation time is therefore of order $\mathcal{O}((\log^2 n)^{\log n}) = \mathcal{O}(n^{\log \log n})$.

Item *2.* Given the transitions exiting from the same configuration, we either discard no transitions or them all but one. By Theorem 2.7, item *1*, we have

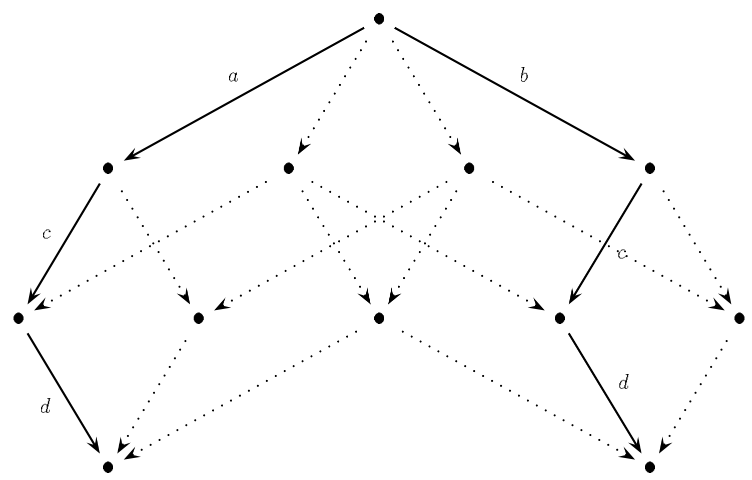


Figure 7: Non optimum space complexity.

that the generation time is $\log n$. $\square$

2.3 Bisimulation

The behaviour of distributed systems is usually compared through bisimulation equivalences. Intuitively, two transition systems are bisimilar if, starting from their initial state, whenever one can fire a transition with a label l, the other can as well, and the reached configurations are again bisimilar, and vice versa. More precisely, we have the following definition.

Definition 2.9 *Let $T = \langle \Gamma, A, \longrightarrow, P, \smile \rangle$ and $T' = \langle \Gamma', A, \longrightarrow', P', \smile' \rangle$ be two transition systems. Then a binary symmetric relation $\mathcal{R} \subseteq \Gamma \times \Gamma'$ is a bisimulation iff*

- $P \mathcal{R} P'$;
- *if* $\gamma_0 \mathcal{R} \gamma_0'$ *and* $\gamma_0 \xrightarrow{a} \gamma_1$, *then* $\gamma_0' \xrightarrow{a} \gamma_1'$ *and* $\gamma_1 \mathcal{R} \gamma_1'$;
- *if* $\gamma_0 \mathcal{R} \gamma_0'$ *and* $\gamma_0' \xrightarrow{a} \gamma_1'$, *then* $\gamma_0 \xrightarrow{a} \gamma_1$ *and* $\gamma_1 \mathcal{R} \gamma_1'$;

If there exists a bisimulation between T and T' we say that the two transition systems are bisimilar*, and we write $T \approx T'$.*

It is immediate to prove that bisimulations induce equivalence relations.

Bisimulation equivalences are preserved by our reduction when the semantics considers concurrency as primitive, therefore not semantically expressible by any combination of other relations. In other words, concurrency is a basic ingredient of the bisimilarity check, in the style of [10, 8, 3, 17]. We call *non interleaving* the equivalences respecting the axiom below

$$\begin{aligned} &\gamma_0 \xrightarrow{a} \gamma_1 \smile \gamma_0 \xrightarrow{b} \gamma_2, \gamma_0 \approx \gamma_0' \;\Rightarrow \\ &\quad \exists \gamma_1', \gamma_2' \,.\, \gamma_0' \xrightarrow{a} \gamma_1' \smile \gamma_0' \xrightarrow{b} \gamma_2', \gamma_1 \approx \gamma_1', \gamma_2 \approx \gamma_2' \end{aligned} \tag{10}$$

We now prove the main theorem of this subsection.

Theorem 2.10 *Let $T = \langle \Gamma, A, \longrightarrow, P, \smile \rangle$ and $T' = \langle \Gamma', A, \longrightarrow', P', \smile' \rangle$ be two transition systems. Also, let* $\mathrm{cT} = \langle \Gamma_c, A, \hookrightarrow, \gamma \rangle$ *and* $\mathrm{cT}' = \langle \Gamma_c', A, \hookrightarrow', \gamma' \rangle$ *be their compact transition systems, respectively. If $\approx$ is a non interleaving bisimulation, then*

$$T \approx T' \;\Leftrightarrow\; \mathrm{cT} \approx \mathrm{cT}'.$$

Proof. The same concurrency relation holds between the transitions that are related by a bisimulation $\mathcal{R}$, by Axiom (10).

$\Rightarrow$) Let $\mathcal{R}$ be a non interleaving bisimulation between T and T'. Define

$$\mathcal{R}_c = \mathcal{R} - \{\langle \gamma_i, \gamma_j \rangle \mid \gamma_i \notin \Gamma_c \;\vee\; \gamma_j \notin \Gamma_c'\}.$$

We prove that $\mathcal{R}_c$ is a bisimulation between cT and cT'.
If $\langle \gamma_i, \gamma_j \rangle \in \mathcal{R}_c$, whenever $\gamma_i \overset{a}{\hookrightarrow} \gamma_{i+1}$ there exists γ_{j+1} such that $\gamma_j \xrightarrow{a} \gamma_{j+1}$ and $\langle \gamma_{i+1}, \gamma_{j+1} \rangle \in \mathcal{R}$. We are left to prove that $\gamma_j \overset{a}{\hookrightarrow} \gamma_{j+1}$, hence that $\gamma_{j+1} \in \Gamma_c'$. Let

$$S_h = \{a_k \mid \gamma_h \xrightarrow{a_k} \gamma_k\} \text{ and } S_h^c = \{a_k \mid \gamma_h \overset{a_k}{\hookrightarrow} \gamma_k\}.$$

Then,

$$\langle \gamma_i, \gamma_j \rangle \in \mathcal{R}_c \subseteq \mathcal{R} \;\Rightarrow\; S_i = S_j \;\Rightarrow\; min_{\triangleleft} S_i = min_{\triangleleft} S_j \;\Rightarrow\; S_i^c = S_j^c.$$

Therefore,

$$\gamma_i \overset{a}{\hookrightarrow} \gamma_{i+1}, \gamma_j \xrightarrow{a} \gamma_{j+1} \ \wedge\ \langle \gamma_{i+1}, \gamma_{j+1} \rangle \in \mathcal{R} \ \Rightarrow\ \langle \gamma_{i+1}, \gamma_{j+1} \rangle \in \mathcal{R}_c.$$

Finally, since $\langle P, P' \rangle \in \mathcal{R}_c$, we conclude that $\mathcal{R}_c$ is a bisimulation between cT and cT'. Also, since $\mathcal{R}$ is a non interleaving bisimulation, if $\gamma_i \overset{a}{\hookrightarrow} \gamma_{i+1}$, $\gamma_j \overset{a}{\hookrightarrow} \gamma_{j+1}$, and $\langle \gamma_{i+1}, \gamma_{j+1} \rangle \in \mathcal{R}$, then $\forall \gamma_i \overset{b}{\hookrightarrow} \gamma'_{i+1}, \gamma_j \overset{b}{\hookrightarrow} \gamma'_{j+1}, \langle \gamma'_{i+1}, \gamma'_{j+1} \rangle \in \mathcal{R}$ then $\gamma_i \overset{a}{\hookrightarrow} \gamma_{i+1} \smile \gamma_i \overset{b}{\hookrightarrow} \gamma'_{i+1}$ iff $\gamma_j \overset{a}{\hookrightarrow} \gamma_{j+1} \smile \gamma_j \overset{b}{\hookrightarrow} \gamma'_{j+1}$. Hence, also $\mathcal{R}_c$ is non interleaving because $\hookrightarrow \subseteq \longrightarrow$. Symmetrically, in the symmetric case.

$\Leftarrow$) Let $\mathcal{R}_c$ a non interleaving bisimulation between cT and cT' and let $\langle \gamma_i, \gamma_j \rangle \in \mathcal{R}_c$. Then, there exist $\gamma_{i-1} \in \Gamma_c$ and $\gamma_{j-1} \in \Gamma'_c$ such that

$$\langle \gamma_{i-1}, \gamma_{j-1} \rangle \in \mathcal{R}_c \ \wedge\ \gamma_{i-1} \overset{a}{\hookrightarrow} \gamma_i \ \wedge\ \gamma_{j-1} \overset{a}{\hookrightarrow} \gamma_j.$$

Also, $\langle \gamma_i, \gamma_j \rangle \in \mathcal{R}_c$ implies that $S_i^c = S_j^c$. Therefore, if the two sets above are non empty, there are $\gamma_k \in \Gamma_c$ and $\gamma'_k \in \Gamma'_c$ such that

$$\langle \gamma_k, \gamma'_k \rangle \in \mathcal{R}_c \ \wedge\ \gamma_i \overset{a_k}{\hookrightarrow} \gamma'_k \ \wedge\ \gamma_j \overset{a_k}{\hookrightarrow} \gamma'_k.$$

If $a \smile a_k$ apply to the sources and targets of these transition the transformation Ψ of Theorem 2.5. The application of Ψ adds to cT the node γ_{i_1} and the transitions

$$\gamma_i \xrightarrow{a_k} \gamma_{i_1} \xrightarrow{a} \gamma_k.$$

Similarly, cT' is extended with the node γ_{j_1} and the transitions

$$\gamma_j \xrightarrow{a_k} \gamma_{j_1} \xrightarrow{a} \gamma'_k.$$

Indeed γ_{i-1} and γ_{j-1} are still non interleaving bisimilar, because from both of them an a_k-transition exits, concurrent with $\gamma_{i-1} \xrightarrow{a_k} \gamma_{i1}$ and $\gamma_{j-1} \xrightarrow{a_k} \gamma_{j1}$, respectively. Similarly, γ_{i1} and γ_{j1} are non interleaving bisimilar.
It is then safe defining

$$\mathcal{R}_c \cup \{\langle \gamma_{i_1}, \gamma_{j_1} \rangle\}$$

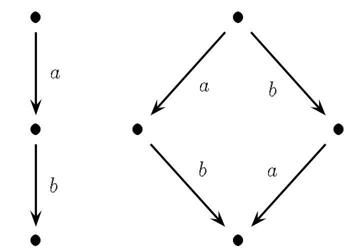
and more generally to obtain the i^{th} approximation starting with $\mathcal{R}^0 = \mathcal{R}_c$ we define

$$\mathcal{R}^i = \mathcal{R}^{i-1} \cup \{\langle \gamma, \gamma' \rangle\}$$

where γ and γ' are the added configurations. When the fixed point of Ψ is reached, the required non interleaving bisimulation is the fixed point of $\mathcal{R}^i$.

□

The following example shows that the reduction does not preserve instead the interleaving bisimulation, because it reduces concurrency to non-determinism and interleaving. The two processes $a \mid b$ and $a.b{+}b.a$ are clearly interleaving bisimilar. Instead, their compact transition systems depicted in Fig. 8(a and b), are not.



(a) (b)

Figure 8: Compact transition systems of (a) $a \mid b$ and of (b) $a.b + b.a$.

We end this section with a time complexity estimation of the non interleaving bisimilarity check. We use the well-known partition-refinement algorithm by Paige and Tarjan [21] whose complexity is $\mathcal{O}(N \log N)$, where N is the number of nodes that approximately are as many as the transitions. Since there are $\mathcal{O}(n^{\log \log n})$ transitions in a cTS, we have the following corollary.

Corollary 2.11 *The check for non interleaving bisimilarity of two* cTS *with* n *equivalences classes has the average time complexity* $\mathcal{O}(n^{\log \log n})$.

Similarly, Theorem 2.7 allows us to state the following corollary.

Corollary 2.12 *The check for non interleaving bisimilarity of two* cTS *satisfying item 1 of Theorem 2.7 has the average time complexity* $\mathcal{O}(n \log n)$.

3 An application

In this section we directly generate the compact transition system from a specification given in a process calculus. In this way, one avoids the generation of the (exponential) complete transition system and its subsequent reduction. We rely on proved transition systems [11, 12], that are defined in *SOS* style. Their transitions are labelled by their proofs, and this permits to retrieve the concurrency relation. The language used here as a test bed is a non-deterministic, concurrent calculus without communication [20]. We will show later how this tiny calculus can be enriched with other features similar to the ones of *CCS* or π-calculus.

Let A be a set of *actions* (ranged over by $\mu, \alpha, \beta, \ldots$). A *process* $t \in T$ is expressed by the following BNF-like grammar:

$$t ::= nil \mid \mu.t \mid t|t \mid t + t$$

In the sequel, we will often omit nil's when irrelevant.

We now define the set of labels of transitions. Let $\vartheta \in \{+_1, +_0, ||_1, ||_0\}^*$. Then, the set of *proof terms* is $\Theta = \{\vartheta\mu\} \ni \theta$.

Intuitively, the symbol $||_0$ ($||_1$) denotes that an action has been performed by the left (right) component of a parallel composition. Tag $+_0$ ($+_1$) means that a nondeterministic choice is resolved in favour of the left (right) summand.

The operational semantics is obtained much in the standard way apart from prefixing an encoding of the proof of the transition to the action. If the standard labelling by actions is of interest, one only needs to apply to proof terms the function ℓ defined as $\ell(\vartheta\mu) = \mu$. Given a term $t_0 \in T$, its *proved transition system* is $PTS = \langle T,\ \Theta,\ \longrightarrow, t_0\rangle$, where $\longrightarrow$ is defined by the axiom and rules of Tab. 1.

We report below an example of derivation of the a transition from the process $(a.b + c)|d$.

$$\dfrac{\dfrac{a.b \xrightarrow{a} b}{a.b + c \xrightarrow{+_0 a} b}}{(a.b + c)|d \xrightarrow{||_0 +_0 a} b|d}$$

Hereafter, when t is fixed, we simply write θ for $t \xrightarrow{\theta} t'$ because the proved transition system is deterministic.

As it is, a proved transition system lacks the independence relation. However, since the labels of the transitions record the proof of their derivations, it is easy to define $\smile$. Two transitions of $Tr(t)$ are concurrent if the

$$Act : \mu.t \xrightarrow{\mu} t$$

$$Sum_0 : \frac{t_1 \xrightarrow{\theta} t_2}{t_1 + t \xrightarrow{+_0\theta} t_2} \qquad Sum_1 : \frac{t_1 \xrightarrow{\theta} t_2}{t + t_1 \xrightarrow{+_1\theta} t_2}$$

$$Asyn_0 : \frac{t_1 \xrightarrow{\theta} t_2}{t_1|t \xrightarrow{||_0\theta} t_2|t} \qquad Asyn_1 : \frac{t_1 \xrightarrow{\theta} t_2}{t|t_1 \xrightarrow{||_1\theta} t|t_2}$$

Table 1: Proved transition system

left-most symbol for which they differ is a $||$. For instance, the a-transition derived above and the transition $(a.b+c)|d \xrightarrow{||_1 d} (a.b+c)|nil$ are concurrent. More precisely, given a process t, let $\theta_i, \theta_j \in Tr(t)$. Then, θ_i and θ_j are *concurrent*, denoted by $\theta_i \smile^0 \theta_j$, if and only if $\theta_i = \vartheta||_0\theta_0$ and $\theta_j = \vartheta||_1\theta_1$. Indeed, if t_i and t_j perform θ_i and θ_j respectively, and if $t = C[t_i|t_j]$ (C is a context), then ϑ represents $C[-]$. Note that if $t_i \xrightarrow{\theta_i} t'_i$ and $t_j \xrightarrow{\theta_j} t'_j$ then

$$C[t'_i|t_j] \xrightarrow{\theta_j} C[t'_i|t'_j] <_L t_j \xrightarrow{\theta_j} t'_j$$

and

$$C[t_i|t'_j] \xrightarrow{\theta_i} C[t'_i|t'_j] <_L t_i \xrightarrow{\theta_i} t'_i$$

where $<_L$ is the relation $<$ instantiated to our language. Then, let $\sim_L$ be the equivalence relation induced by $<_L$. Finally, the concurrency relation $\smile_L$ is the closure of $\smile^0$ with respect to $\sim_L$.

Note that both tags $+_i$ and $||_i$ are necessary for distinguishing concurrency from nondeterminism in transition systems. Consider the process $(\alpha|\beta)+(\gamma|\delta)$ from which four transitions are possible. If tags $+_i$ are dropped from proof terms, it is impossible to say, e.g., whether $||_0\alpha$ is concurrent or not with $||_1\beta$ and $||_1\delta$. Similarly, $||_i$ cannot be ignored. For instance, exchange in the above process $|$'s with $+$'s.

3.1 SOS generation

We first show that our calculus generates transition systems which satisfy the axioms introduced in the previous section.

Theorem 3.1
The transition system with independence $\mathrm{PTS}_I = \langle \mathcal{P}, \Theta, \longrightarrow, t_0, \smile_L \rangle$ *satisfies the Axioms 1-6.*

PROOF. All the axioms are satisfied because:

- PTS_I is deterministic (Axiom (1));
- rules $Asyn$ do not discard any context (Axioms (2,3,4,6));
- rules Sum discard the alternative context. Hence $\not\smile_L$ restricted to a computation only represents the enabling relation determined by nesting of prefixes μ (Axiom (5)).

$\square$

We now give the SOS rule that defines a compact proved transition system. We filter the deduction of a transition so that only some, hopefully a single interleaving of concurrent transitions can be generated. In particular, given a process t_0 with a set of leader transitions H, we prevent the deduction of any transition from t_0 concurrent with $\theta_h \in H$. Since rules $Asyn$ discard no context, their generation is only delayed. To determine the leaders of $Tr(t)$, we need an auxiliary definition first. The total preordering $\lhd$ on the proved transitions of a process is defined as follows. Let $\theta_1, \theta_2 \in Tr(t)$. Then $\theta_1 \lhd \theta_2$ if and only if $\ell(\theta_1) \lhd_l \ell(\theta_2)$, where $\lhd_l$ is a given lexicographic ordering on actions.

The *compact proved transition system* is $cPTS = \langle T, \Theta, \longrightarrow_c, t_0 \rangle$, where $\longrightarrow_c$ is defined by the following rule which has $\longrightarrow$ as auxiliary arrow,

$$Red : \frac{t \xrightarrow{\theta} t', t \stackrel{\theta'}{\not\longrightarrow}, \theta' \smile_L \theta, \theta' \lhd \theta}{t \xrightarrow{\theta}_c t'}$$

where $t \stackrel{\theta'}{\not\longrightarrow}$ stands for there is no transition from t labelled θ'. Note that checking this negative premise is effective because of the finite branching property of our transition systems.

As an example, consider the process

$$((b.f \,|\, (b + c)) \,|\, a) + (e \,|\, d)$$

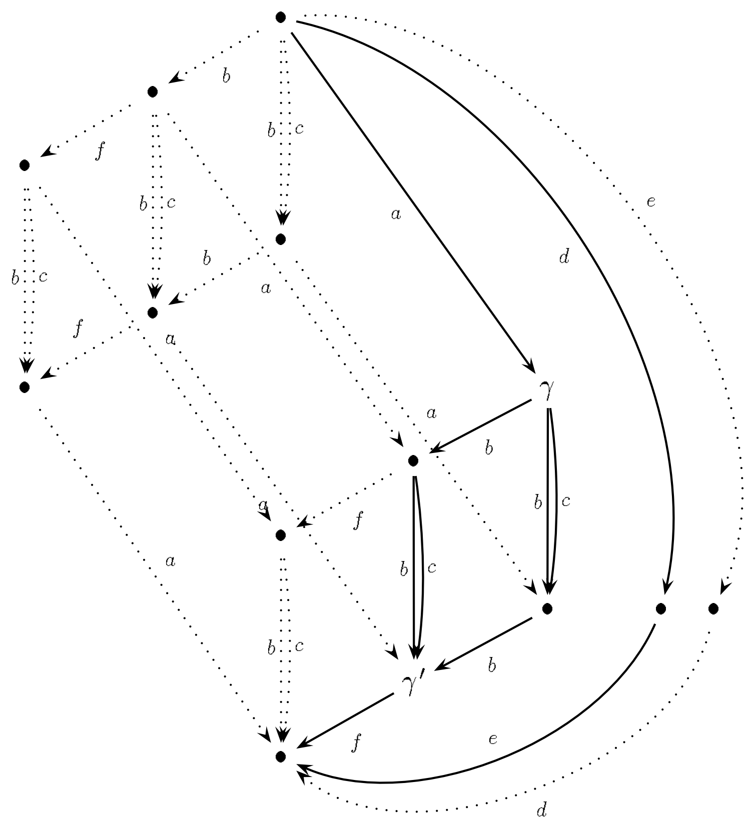


Figure 9: Compact (full color) and complete *PTS* of $((b.f \mid (b{+}c)) \mid a){+}(e \mid d)$. For readability, transitions are labelled by actions only.

Its *cPTS* is illustrated in Fig. 9 (full color). The compact representation has 7 states and 10 transitions against 14 states and 30 transitions of the complete *PTS*.
Note that without autoconcurrency between the b's in the left summand, only 3 (with labels $+_0||_0||_1{+}_0 b$, $+_0||_0||_1{+}_1 c$ and $+_0||_0||_0 b$) of the 6 transitions between γ and γ' (see the figure) will be kept.

It is immediate noticing that rule *Red* is just a rephrasing of function $\mathcal{C}$ in Subsect. 2.1. Indeed, the compact proved transition system of a term t obtained inductively coincides with the one defined in Subsect. 2.1.

Theorem 3.2 *For all terms t,* $\langle T, \Theta, \longrightarrow_c, t\rangle = \langle T, \Theta, \hookrightarrow, t\rangle$.

The following corollary reports some complexity estimations of the transition systems originated by our calculus. We assume that the operators of the language are equiprobable. This and forward stability ensure that our transition systems fit the assumptions of Theorem 2.6. Hence, the complexity estimation are still valid, once the number of equivalence classes is replaced by the number of actions that occur in a process.

Corollary 3.3 *Given a process t with n occurrences of actions, let all operators of the language have the same probability to occur. Then, the* average space *complexity of the compact transition system of t is $O(n^{\log\log n})$.*

The following theorem determines the generation time complexity.

Theorem 3.4 *Given a process t with n occurrences of actions, let all operators of the language have the same probability to occur. Also, assume that the generation of a transition has a cost equal to the number of rules applied in its derivation. Then, the* average time *complexity of the compact transition system of t is $O(n^{\log\log n})$.*

Proof. The generation of a transition whose source has out-degree k requires time proportional to $k-1$. Hence, the generation of the k transitions in $Tr(t)$ is of $\mathcal{O}(k^2)$. The selection of the leaders is performed in $\log k$ times and the check for the concurrency relation is of $\mathcal{O}(k)$ as we need to compare proof terms that have length k. According to Theorem 2.6 we prune the set $Tr(t)$ in $\log k$ iterations. Therefore the cost of the reduction, when applied to a configuration t, is of $\mathcal{O}(k\log k + \log^2 k)$. As a consequence, the generation of the transitions exiting from a node belonging to the cTS is of $\mathcal{O}(k^2)$. The above steps are iterated for any targets of the $\log k$ transitions kept so far. The out-degree of this new states will be in average $k/2$ because rules Sum discard the alternatives and $\mid$ and $+$ are equiprobable (we do not consider at all the number of prefixes). The iterations are $\log n$ (Theorem 2.6). Hence, the generation cost is

$$\sum_{i=0}^{\log n}(\frac{k}{2^j})^2\prod_{0=1}^{i}\log\frac{k}{2^j} \leq \sum_{i=0}^{\log n}(\frac{k}{2^j})^2\prod_{0=1}^{i}\log k \in \mathcal{O}((\log k)^{\log k})$$

Since $\mathcal{O}(k) = \mathcal{O}(n)$, we conclude the proof. □

Recursion can be added to our calculus still preserving all the results reported above. We only need to tag with a special symbol the first transitions performed by a recursive process and to let them greater than any other transitions in $\lhd$. This corresponds to delay as much as possible their selection in the reduction strategy. Adding a relabelling operator is absolutely straightforward.

Finally, no claim can be made on the minimality of the compact transition system. This is due to $\lhd$, that preorders transitions arbitrarily. For instance, if $a \lhd b \lhd c \lhd d$, the compact transition system of $a \,|\, (b.c + d)$ has 4 arcs, while the one for $b \,|\, (a.c + d)$ has 5 arcs for the duplication of an arc labelled b, although the two complete transition systems have the same shape. Note that the same problem will show up with *any* ordering.

4 Interaction

Particularly relevant to distributed systems is the case in which some transitions designate the interaction of more elementary activities. A typical case is communication. In this section we extend the results presented beforehand. We consider as given those transitions originated by cooperation of more than one action, that we call *compound*, and we assume its distinguished label be τ.

A problem may arise in *CCS*-like calculi, when actions that may communicate occur within the scope of a restriction operator $\backslash a$. In this case concurrency may implicitly make choices. Consider for example the *CCS*-like process

$$((a + c)|b.\overline{c})\backslash c.$$

The occurrence of a, concurrent with b and $\overline{c}$, prevents the latter to occur because it is restricted. Instead, if a is not chosen the computation with b followed by the communication τ between c and $\overline{c}$ is possible.

We will give below four axioms ensuring that concurrency makes no hidden choice, and that the occurrence of a compound transition prevents the one of its partners. We will also prove that *CCS* without restriction satisfies this new axioms.

Assume a family of $(n+1)$-ary ($n \geq 2$) relations $\mathcal{G}^n \subseteq (\longrightarrow_{/\sim})^n \times (\longrightarrow_{/\sim})$ over (equivalence classes of) transitions that identify the components of a compound transition. The first n components of an element of $\mathcal{G}^n$ represent the transitions of the partners originating the compound transition. This in turn is the $(n+1)^{th}$ component. We enrich our transition systems with such a family of relations, thus yielding

$$TS_g = \langle \Gamma, A, \longrightarrow, P, \smile, \mathcal{G}^n \rangle.$$

When the interaction mechanism is fixed (for instance based on complementary actions like in *CCS* or on same actions like in *TCSP*), the family of relations $\mathcal{G}^n$ can be safely omitted.

Assume in the following axioms that

$$\chi = ([\gamma_i \xrightarrow{d_i} \gamma_i']_{1 \leq i \leq n}, [\gamma_0 \xrightarrow{\tau} \gamma_0']) \in \mathcal{G}^n.$$

By abuse of notation, we write $\gamma \xrightarrow{d_i} \gamma' \in \chi$ if $\gamma \xrightarrow{d_i} \gamma'$ belongs to the ith equivalence class of χ.

The first requirement we put on $\mathcal{G}^n$ says that all the components of a compound transition are concurrent pairwise, but none is concurrent with the compound transition.

$$\forall \chi \in \mathcal{G}^n, 1 \leq i \neq j \leq n \,.\, (\gamma_i \xrightarrow{d_i} \gamma_i' \smile \gamma_j \xrightarrow{d_j} \gamma_j' \not\smile \gamma_0 \xrightarrow{\tau} \gamma_0') \tag{11}$$

Note that $\gamma_i \xrightarrow{d_i} \gamma_i' \smile \gamma_j \xrightarrow{d_j} \gamma_j'$ implies that any transition in $[\gamma_i \xrightarrow{d_i} \gamma_i']$ is concurrent with any transition belonging to the other class by Axiom (3).

For instance, the *CCS*-like communication in $a.P \,|\, \overline{a}.Q$ arises from the interaction of a and $\overline{a}$. Our ternary relation $\mathcal{G}^2$ contains

$$([a.P \,|\, \overline{a}.Q \xrightarrow{a} P \,|\, \overline{a}.Q], [a.P \,|\, \overline{a}.Q \xrightarrow{\overline{a}} a.P \,|\, Q], [a.P \,|\, \overline{a}.Q \xrightarrow{\tau} P \,|\, Q]).$$

Moreover, if a transition is concurrent with all the components of a compound transition, it is also concurrent with the compound transition itself.

$$\begin{aligned} &([\gamma_i \xrightarrow{d_i} \gamma_i']_{1 \leq i \leq n}, [\gamma_0 \xrightarrow{\tau} \gamma_0']) \in \mathcal{G}^n \wedge \\ &\quad \forall i \,.\, \gamma_i \xrightarrow{d_i} \gamma_i' \smile \gamma_n \xrightarrow{a} \gamma_n'', a \neq d_i \;\Rightarrow\; \gamma_0 \xrightarrow{\tau} \gamma_0' \smile \gamma_n \xrightarrow{a} \gamma_n'' \end{aligned} \tag{12}$$

Next, we postulate that there is a compound transition $\gamma_0 \xrightarrow{\tau} \gamma_0'$ if and only if each equivalence class of its components has a transition leaving γ_0 and a transition entering γ_0'.

$$\begin{aligned} &\gamma \xrightarrow{\tau} \gamma' \in [\gamma_0 \xrightarrow{\tau} \gamma_0'] \;\Leftrightarrow \\ &\quad \forall i, \exists \hat{\gamma}_i, \hat{\gamma}_i' \,.\, \{\gamma \xrightarrow{d_i} \hat{\gamma}_i, \hat{\gamma}_i' \xrightarrow{d_i} \gamma'\} \subseteq [\gamma_i \xrightarrow{d_i} \gamma_i'] \end{aligned} \tag{13}$$

To ensure that concurrency does not operate choices we also need the following axiom that extends Axiom (6), which still holds for τ.

$$\begin{aligned} &\gamma \xrightarrow{a} \gamma_0 \smile \gamma_0 \xrightarrow{b} \gamma_1 \not\smile \gamma_0 \xrightarrow{\tau} \gamma_0' \;\Rightarrow \\ &\quad \exists \gamma_i', d_i \,.\, \gamma_0 \xrightarrow{a} \gamma_0' \smile \gamma \xrightarrow{d_i} \gamma_i' \in \chi \end{aligned} \tag{14}$$

Note that some classical process calculi like *CCS*, *ACP*, *TCSP* or π-calculus do not verify the above axioms (in particular Axioms (13) and (14)). This is due to the interplay between concurrency and nondeterminism within the scope of a scope operator, as exemplified at the beginning of this section. Another example arises in *TCSP*, because the communication mechanism forces communications on a given set of labels. For example, the process

$$a.b \,||_{\{b\}}\, c.b$$

violates Axiom (13) because the two b-transitions cannot fire asynchronously. Finally, note instead that the sequential composition of processes (usually written ;) rises no problem. Indeed, in $P; Q$ no action of Q can occur concurrently with any action of P, nor they can be compound either.

As far as the reduction is concerned, the compact transition system will be the sextuple $cTS = \langle \Gamma, A, \hookrightarrow, P, \smile/_{\hookrightarrow}, \mathcal{G}^n/_{\hookrightarrow} \rangle$. Some relations $\mathcal{G}^n/_{\hookrightarrow}$ may have the equivalence class of compound transitions empty because of the definition of function $\mathcal{C}$. In this case we put a placeholder $\bullet$ in the corresponding class. For instance, consider process $P = a \,|\, b.\overline{a}$ and assume the *CCS* communication mechanism. The reduction selects $P \overset{a}{\hookrightarrow} \mathbf{0} \,|\, b.\overline{a}$ as unique transition from P. Then, we have the transition $\mathbf{0} \,|\, b.\overline{a} \overset{b}{\hookrightarrow} \mathbf{0} \,|\, \overline{a}$ and finally $\mathbf{0} \,|\, \overline{a} \overset{\overline{a}}{\hookrightarrow} \mathbf{0} \,|\, \mathbf{0}$. Since the reduction generates no state which enables both a transition with action a and one with action $\overline{a}$, the corresponding communication does not belong to the compact transition system of P. However, this is by no means a loss of information, because we have $([a \,|\, b.\overline{a} \overset{a}{\hookrightarrow} \mathbf{0} \,|\, b.\overline{a}], [\mathbf{0}|\, \overline{a} \overset{\overline{a}}{\hookrightarrow} \mathbf{0}|\mathbf{0}], [\bullet]) \in \mathcal{G}^2/_{\hookrightarrow}$ that is enough to reconstruct $[a|\overline{a} \xrightarrow{\tau} \mathbf{0}|\mathbf{0}]$ (see below).

To rebuild the complete transition system from its compact one, we need to rebuild also the compound transitions and the family of relations $\mathcal{G}^n$. Therefore, the fix-point construction of Theorem 2.5 must add a τ-transition from the source of any sequence of consecutive transitions that compose the τ-transition itself, according to relations $\mathcal{G}^n$. Hence, in the definition of Ψ we add the item

$$\begin{array}{l} \forall n > 1, \gamma_0 \xrightarrow{d_1}_i \gamma_1 \xrightarrow{d_2}_i \ldots \gamma_{n-1} \xrightarrow{d_n} \gamma_n, \\ \quad ([\gamma_j \xrightarrow{d_j}_i \gamma_{j+1}]_{0 \leq j < n}, [\gamma' \xrightarrow{\tau}_i \gamma'']) \in \mathcal{G}_i^n \;\; \Rightarrow \\ \quad \gamma_0 \xrightarrow{\tau}_{i+1} \gamma_n \sim_{i+1} \gamma' \xrightarrow{\tau}_i \gamma'' \end{array}$$

If $([\gamma_j \xrightarrow{d_j}_i \gamma_{j+1}]_{0 \leq j < n}, [\bullet]) \in \mathcal{G}_i^n$, then $([\gamma_j \xrightarrow{d_j}_i \gamma_{j+1}]_{0 \leq j < n}, [\gamma_0 \xrightarrow{\tau}_{i+1}$

$\gamma_n]) \in \mathcal{G}^n_{i+1}$. This item rebuilds compound transitions according to Axiom (13) and $\mathcal{G}^n$ according to $\sim$.

Also, according to Axiom (13), we join the following set to the items defining $\smile_{i+1}$

$$\{\langle \tilde{\gamma} \xrightarrow{a}_i \tilde{\gamma}', \gamma_0 \xrightarrow{\tau}_{i+1} \gamma_n\rangle \mid \forall j, 0 \leq j < n, \tilde{\gamma} \xrightarrow{a}_i \tilde{\gamma}' \smile_i \gamma_j \xrightarrow{d_j}_i \gamma_{j+1} \in \chi\}$$

The two sets are introduced to reconstruct the concurrency relation according to Axioms (11) and (13), respectively.

Here, we omit the proof that the reconstruction sketched above yields back the original complete transition system. It has the same shape of the proof of Theorem 2.5 and only requires some more technicalities.

We now show that *CCS* without restriction satisfies Axioms (1-6) and (11-14). First recall that communication is ruled by

$$Com_0 : \frac{t_0 \xrightarrow{\vartheta_0 a} t_0',\, t_1 \xrightarrow{\vartheta_1 \overline{a}} t_1'}{t_0 \,|\, t_1 \xrightarrow{\tau} t_0' \,|\, t_1'} \qquad Com_1 : \frac{t_0 \xrightarrow{\vartheta_0 \overline{a}} t_0',\, t_1 \xrightarrow{\vartheta_1 a} t_1'}{t_0 \,|\, t_1 \xrightarrow{\tau} t_0' \,|\, t_1'}$$

Theorem 4.1
The transition system with independence $\mathrm{PTS}_I = \langle \mathcal{P}, \Theta, \longrightarrow, t_0, \smile_L\rangle$ *satisfies the Axioms (1-6) and (11-14).*

PROOF. We only need to prove the statement for the second group of axioms. Axioms (11) and (13) are satisfied because rules *Asyn* and *Com* share the same premises.
To show that Axiom (12) is satisfied, consider a context from which a communication, and hence its partners a and $\overline{a}$, are enabled. Assuming that t_0 performs a and lays at the left of t_1,

$$C[t_0, t_1] \xrightarrow{\vartheta ||_0 \vartheta_0 a} C'[t_0', t_1]$$

$$C[t_0, t_1] \xrightarrow{\vartheta ||_1 \vartheta_1 \overline{a}} C''[t_0, t_1']$$

$$C[t_0, t_1] \xrightarrow{\vartheta \tau} C'''[t_0', t_1']$$

Assume now that a further transition b is enabled that is concurrent with the two partners. Three cases are possible: b is performed by a subterm of t_0, b is performed by a subterm of t_1, or b is performed by a subterm of the context $\mathcal{C}$. We work out only the third case because the other two are similar. We can write the b-transition as

$$\tilde{C}[t_0, t_1, t_2] \xrightarrow{\vartheta' b} \tilde{C}[t_0, t_1, t_2']$$

where t_2 is a subterm of context $\mathcal{C}$. By definition of $\smile_L$ its label must differ from the ones of the $\overline{a}$- and a-transitions for a $||_i$ and must share a (possibly empty) prefix of the proof term. Hence, $\vartheta' = \vartheta''||_0\hat{\vartheta}'$ and $\vartheta = \vartheta''||_1\hat{\vartheta}$. Therefrom the b-transition is concurrent with the communication as well. Axiom (14) is satisfied because the rules for $|$ discard no context. $\square$

Acknowledgements. The authors thank L. Pagli, A. Pietracaprina and G. Pucci for their help on the complexity issues.

References

[1] M.A. Bednarczyk. *Categories of Asynchronous Transition Systems.* PhD thesis, University of Sussex, 1988.

[2] A. Bianchi, S. Coluccini, P. Degano, and C. Priami. An efficient verifier of truly concurrent properties. In V. Malyshkin, editor, *Proceedings of PaCT'95, LNCS 964*, pages 36–50. Springer-Verlag, 1995.

[3] G. Boudol, I. Castellani, M. Hennessy, and A. Kiehn. A theory of processes with localities. *Theoretical Computer Science*, 114, 1993.

[4] S.D. Brookes, C.A.R. Hoare, and A.D. Roscoe. A theory of communicating sequential processes. *Journal of ACM*, 31(4):560–599, 1984.

[5] E. R. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, C-36(8):677–691, 1986.

[6] R. Cleaveland, J. Parrow, and B. Steffen. The concurrency workbench: A semantics-based tool for the verification of concurrent systems. *ACM Transaction on Programming Languages and Systems*, pages 36–72, 1993.

[7] M. Clegg and A. Valmari. Reduced labelled transition systems save verification effort. In *Proceedings of CONCUR'91, LNCS 527*. Springer-Verlag, 1991.

[8] Ph. Darondeau and P. Degano. Causal trees. In *Proceedings of ICALP'89, LNCS 372*, pages 234–248. Springer-Verlag, 1989.

[9] R. de Simone and D. Vergamini. Aboard AUTO. Technical Report 111, INRIA Sophia-Antipolis, 1989.

[10] P. Degano, R. De Nicola, and U. Montanari. A partial ordering semantics for CCS. *Theoretical Computer Science*, 75:223–262, 1990.

[11] P. Degano and C. Priami. Proved trees. In *Proceedings of ICALP'92, LNCS 623*, pages 629–640. Springer-Verlag, 1992.

[12] P. Degano and C. Priami. Causality for mobile processes. In *Proceedings of ICALP'95, LNCS 944*, pages 660–671. Springer-Verlag, 1995.

[13] J.C. Fernandez and L. Mounier. A tool set for deciding behavioural equivalences. In *Proceedings of CONCUR'91, LNCS 527*, 1991.

[14] P. Godefroid. Using partial orders to improve automatic verification methods. In *Proceedings of CAV'90*, 1990.

[15] P. Godefroid and P. Wolper. Using partial orders for the efficient verification of deadlock freedom and safety properties. In *Proceedings of CAV'91, LNCS 575*, pages 332–342. Springer-Verlag, 1991.

[16] R. Janicki and M. Koutny. Net implementation of optimal simulations. In *Proceedings of Application and Theory of Petri Nets '90*, pages 295–314, 1990.

[17] A. Joyal, M. Nielsen, and G. Winskel. Bisimulation from open maps. Technical report, BRICS RS-94-7, 1994.

[18] K.L. McMillan. Using unfoldings to avoid the state explosion problem in the verification of asynchronous circuits. In *Proceedings of CAV'92*, 1992.

[19] R. Milner. *Communication and Concurrency*. Prentice-Hall, London, 1989.

[20] R. Milner and F. Moller. Unique decomposition of processes. *Theoretical Computer Science*, pages 357 – 364, 1993.

[21] R. Paige and R. Tarjan. Three partition refinement algorithms. *SIAM Journal on Computing*, 16(6):973–989, 1987.

[22] D. Peled. All from one, one for all: On model checking using representatives. In *Proceedings of CAV'93, LNCS 697*, pages 409–423. Springer-Verlag, 1993.

[23] D.K. Probst and H.F. Li. Using partial order semantics to avoid the state explosion problem in asynchronous systems. In *Proceedings of CAV'90*, 1990.

[24] M.W. Shields. Concurrent machines. *Computer Journal*, pages 449–465, 1985.

[25] A. Stark. Concurrent transition systems. *Theoretical Computer Science*, pages 221–269, 1989.

[26] A. Valmari. A stubborn attack on state explosion. In *Proceedings of CAV'90*, 1990.

[27] A. Valmari and M. Tienari. An improved failure equivalence for finite-state systems with a reduction algorithm. In *Proceedings of IFIP WG6.1 Protocol Specification, Testing and Verification*, Stockholm, June 1991.

[28] G. Winskel and M. Nielsen. Models for concurrency. Technical Report DAIMI-PB-429, Computer Science Dept., Aarhus University, 1992. To appear in Handbook of Logic in Computer Science.