Foundations of Software Technology and Theoretical Computer Science
19th Conference, Proceedings, C. Pandu Rangan, V. Raman and R. Ramanujam (eds.)
Springer Lecture Notes in Computer Science 1739 (1999), 219–231.

Synthesizing distributed transition systems from global specifications*

Ilaria Castellani[1], Madhavan Mukund[2], and P S Thiagarajan[2]

[1] INRIA Sophia Antipolis, 2004 route des Lucioles, B.P. 93, 06092 Sophia Antipolis Cedex, France. E-mail: Ilaria.Castellani@sophia.inria.fr
[2] Chennai Mathematical Institute, 92 G.N. Chetty Road, Chennai 600 017, India. E-mail: {madhavan,pst}@smi.ernet.in

Abstract. We study the problem of synthesizing distributed implementations from global specifications. In particular, we characterize when a global transition system can be implemented as a synchronized product of local transition systems. Our work extends a number of previous studies in this area which have tended to make strong assumptions about the specification—either in terms of determinacy or in terms of information concerning concurrency.

We also examine the more difficult problem where the correctness of the implementation in relation to the specification is stated in terms of bisimulation rather than isomorphism. As an important first step, we show how the synthesis problem can be solved in this setting when the implementation is required to be deterministic.

1 Introduction

Designing distributed systems has always been a challenging task. Interactions between the processes can introduce subtle errors in the system's overall behaviour which may pass undetected even after rigorous testing. A fruitful approach in recent years has been to specify the behaviour of the overall system in a global manner and then *automatically synthesize* a distributed implementation from the specification.

The question of identifying when a sequential specification has an implementation in terms of a desired distributed architecture was first raised in the context of Petri nets. Ehrenfeucht and Rozenberg [ER90] introduced the concept of *regions* to describe how to associate places of nets with states of a transition system. In [NRT92], Nielsen, Rozenberg and Thiagarajan use regions to characterize the class of transition systems which arise from elementary net systems. Subsequently, several authors have extended this characterization to larger classes of nets (for a sample of the literature, see [BD98,Muk92,WN95]).

Here, we focus on *product transition systems*—networks of transition systems which coordinate by synchronizing on common actions [Arn94]. This model

* This work has been sponsored by IFCPAR Project 1502-1. This work has also been supported in part by BRICS (Basic Research in Computer Science, Centre of the Danish National Research Foundation), Aarhus University, Denmark.

comes with a natural notion of component and induced notions of concurrency and causality. It has a well-understood theory, at least in the linear-time setting [Thi95]. This model is also the basis for system descriptions in a number of model-checking tools [Kur94,Hol97].

We establish two main sets of results in this paper. First, we characterize when an *arbitrary* transition system is isomorphic to a product transition system with a specified distribution of actions. Our characterization is effective—for finite-state specifications, we can synthesize a finite-state implementation. We then show how to obtain implementations when concurrency is specified in terms of an abstract independence relation, in the sense of Mazurkiewicz trace theory [Maz89]. We also present realizability relationships between product transition systems in terms of a natural preorder over the distribution of actions across agents. Our result subsumes the work of Morin [Mor98] on synthesizing product systems from *deterministic* specifications.

Our second result deals with the situation when we have global specifications which are behaviourally equivalent to, but not necessarily isomorphic to, product systems. The notion of behavioural equivalence which we use is strong bisimulation [Mil89]. The synthesis problem here is to implement a global transition system TS as a product transition system $\widetilde{TS}$ such that TS and $\widetilde{TS}$ are bisimilar to each other. We show how to solve this problem when the implementation is deterministic. Notice that the specification itself may be nondeterministic. Since distributed systems implemented in hardware, such as digital controllers, are deterministic, the determinacy assumption is a natural one. Solving the synthesis problem modulo bisimulation in the general case where the implementation may be nondeterministic appears to be hard.

The problem of expressing a global transition system as a product of component transition systems modulo bisimilarity has also been investigated in the context of process algebras in [Mol89,MM93]. In [GM92], Groote and Moller examine the use of decomposition techniques for the verification of parallel systems. These results are established in the context of transition systems which are generated using process algebra expressions. Generalizing these results to arbitrary, unstructured, finite-state systems appears hard.

The paper is organized as follows. In the next section we formally introduce product transition systems and formulate the synthesis problem. In Section 3, we characterize the class of transition systems which are isomorphic to product transition systems. The subsequent section extends these results to the context where the distributed implementation is described using an independence relation. Next, we show that deterministic systems admit canonical minimal implementations. In Section 6, we present our second main result, characterizing the class of transition systems which are bisimilar to deterministic product systems.

2 The synthesis problem for product transition systems

Labelled transition systems provide a general framework for modelling computing systems. A labelled transition system is defined as follows.

Definition 2.1. *Let Σ be a finite nonempty set of actions. A labelled transition system over Σ is a structure $TS = (Q, \rightarrow, q_{\text{in}})$, where Q is a set of* states*, $q_{in} \in Q$ is the* initial state *and $\rightarrow \subseteq Q \times \Sigma \times Q$ is the* transition relation.

We abbreviate a transition sequence of the form $q_0 \xrightarrow{a_1} q_1 \cdots \xrightarrow{a_n} q_n$ as $q_0 \xrightarrow{a_1 \cdots a_n} q_n$. In every transition system $TS = (Q, \rightarrow, q_{\text{in}})$ which we encounter, we assume that each state in Q is reachable from the initial state—that is, for each $q \in Q$ there exists a transition sequence $q_{in} = q_0 \xrightarrow{a_1 \cdots a_n} q_n = q$.

A large class of distributed systems can be fruitfully modelled as networks of local transition systems whose moves are globally synchronized through common actions. To formalize this, we begin with the notion of a distributed alphabet.

Definition 2.2. *A distributed alphabet over Σ, or a* distribution *of Σ, is a tuple of nonempty sets $\widetilde{\Sigma} = \langle \Sigma_1, \ldots, \Sigma_k \rangle$ such that $\bigcup_{1 \leq i \leq k} \Sigma_i = \Sigma$. For each action $a \in \Sigma$, the* locations *of a are given by the set $loc_{\widetilde{\Sigma}}(a) = \{i \mid a \in \Sigma_i\}$. If $\widetilde{\Sigma}$ is clear from the context, we write just $loc(a)$ to denote $loc_{\widetilde{\Sigma}}(a)$.*

We consider two distributions to be the same if they differ only in the order of their components.

Henceforth, for any natural number k, $[1..k]$ denotes the set $\{1, 2, \ldots, k\}$.

Definition 2.3. *Let $\langle \Sigma_1, \ldots, \Sigma_k \rangle$ be a distribution of Σ. For each $i \in [1..k]$, let $TS_i = (Q_i, \rightarrow_i, q^i_{\text{in}})$ be a transition system over Σ_i. The product $(TS_1 \parallel \cdots \parallel TS_k)$ is the transition system $TS = (Q, \rightarrow, q_{\text{in}})$ over $\Sigma = \bigcup_{1 \leq i \leq k} \Sigma_i$, where:*

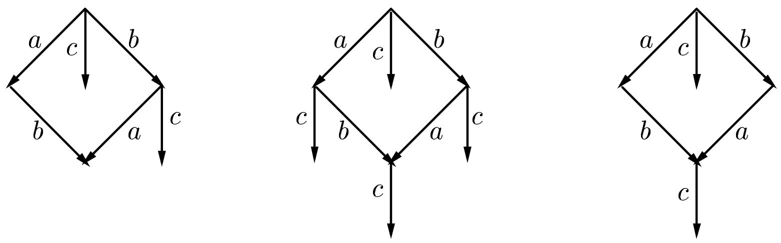
- $q_{\text{in}} = (q^1_{\text{in}}, \ldots, q^k_{\text{in}})$.
- $Q \subseteq (Q_1 \times \cdots \times Q_k)$ *and* $\rightarrow \subseteq Q \times \Sigma \times Q$ *are defined inductively by:*
  - $q_{\text{in}} \in Q$.
  - *Let $q \in Q$ and $a \in \Sigma$. For $i \in [1..k]$, let $q[i]$ denote the i^{th} component of q. If for each $i \in loc(a)$, TS_i has a transition $q[i] \xrightarrow{a}_i q'_i$, then $q \xrightarrow{a} q'$ and $q' \in Q$ where $q'[i] = q'_i$ for $i \in loc(a)$ and $q'[j] = q[j]$ for $j \notin loc(a)$.*

We often abbreviate the product $(TS_1 \parallel \cdots \parallel TS_k)$ by $\parallel_{i \in [1..k]} TS_i$.

The synthesis problem

The synthesis problem can now be formulated as follows. If $TS = (Q, \rightarrow, q_{\text{in}})$ is a transition system over Σ, and $\widetilde{\Sigma} = \langle \Sigma_1, \ldots, \Sigma_k \rangle$ is a distribution of Σ, does there exist a $\widetilde{\Sigma}$-implementation of TS—that is, a tuple of transition systems $\langle TS_1, \ldots, TS_k \rangle$ such that TS_i is a transition system over Σ_i and the product $\parallel_{i \in [1..k]} TS_i$ is isomorphic to TS?

Example 2.4. Let $\widetilde{\Sigma} = \langle \{a, c\}, \{b, c\} \rangle$ be a distribution of $\{a, b, c\}$. The first transition system below is $\widetilde{\Sigma}$-implementable—using expressions in the style of process algebra, we can write the product as $(a + c) \parallel (bc + c)$. Similarly, the second transition system may be implemented as $(ac + c) \parallel (bc + c)$.



On the other hand, the system on the right is not $\widetilde{\Sigma}$*-implementable. Intuitively, the argument is as follows. If it were implementable in two components with alphabets* $\{a, c\}$ *and* $\{b, c\}$*,* c *would be enabled at the initial state in both components. But* c *can also occur after both actions* a *and* b *have occurred. So,* c *is possible after* a *in the first component and after* b *in the second component. Thus, there are two* c *transitions in both components and the product should exhibit all their combinations, giving rise to the system in the centre. We can formalize this argument once we have proved the results in the next section.*

3 A characterization of implementable systems

We now characterize $\widetilde{\Sigma}$-implementable systems. In this section, unless otherwise specified, we assume that $\widetilde{\Sigma} = \langle \Sigma_1, \ldots, \Sigma_k \rangle$, with $\Sigma = \bigcup_{i \in [1..k]} \Sigma_i$.

The basic idea is to label each state of the given system by a k-tuple of local states (corresponding to a global state of a product system) such that the labelling function satisfies some consistency conditions. We formulate this labelling function in terms of local equivalence relations on the states of the original system—for each $i \in [1..k]$, if two states q_1 and q_2 of the original system are i-equivalent, the interpretation is that the global states assigned to q_1 and q_2 by the labelling function agree on the i^{th} component. Our technique is similar to the one developed independently by Morin for the more restrictive class of *deterministic* transition system specifications [Mor98].

Theorem 3.1. *A transition system* $TS = (Q, \rightarrow, q_{\mathrm{in}})$ *is* $\widetilde{\Sigma}$*-implementable with respect to a distribution* $\widetilde{\Sigma} = \langle \Sigma_1, \ldots, \Sigma_k \rangle$ *if and only if for each* $i \in [1..k]$ *there exists an equivalence relation* $\equiv_i \subseteq (Q \times Q)$ *such that the following conditions are satisfied:*

(i) *If* $q \xrightarrow{a} q'$ *and* $a \notin \Sigma_i$*, then* $q \equiv_i q'$*.*
(ii) *If* $q \equiv_i q'$ *for every* i*, then* $q = q'$*.*
(iii) *Let* $q \in Q$ *and* $a \in \Sigma$*. If for each* $i \in loc(a)$*, there exist* $s_i, s_i' \in Q$ *such that* $s_i \equiv_i q$*, and* $s_i \xrightarrow{a} s_i'$*, then for each choice of such* s_i*'s and* s_i'*'s there exists* $q' \in Q$ *such that* $q \xrightarrow{a} q'$ *and for each* $i \in loc(a)$*,* $q' \equiv_i s_i'$*.*

Proof. $(\Rightarrow)$: Suppose $\|_{i \in [1..k]} TS_i$ is a $\widetilde{\Sigma}$-implementation of TS. We must exhibit k equivalence relations $\{\equiv_i\}_{i \in [1..k]}$, such that conditions *(i)*—*(iii)* are satisfied. Assume, without loss of generality, that TS is not just isomorphic to $\|_{i \in [1..k]} TS_i$ but is in fact equal to $\|_{i \in [1..k]} TS_i$.

For $i \in [1..k]$, let $TS_i = (Q_i, \rightarrow_i, q^i_{\text{in}})$. We then have $Q \subseteq (Q_1 \times \cdots \times Q_k)$ and $q_{\text{in}} = (q^1_{\text{in}}, \ldots, q^k_{\text{in}})$. Define $\equiv_i \subseteq (Q \times Q)$ as follows: $q \equiv_i q'$ iff $q[i] = q'[i]$.

Since TS is a product transition system, it is clear that conditions *(i)* and *(ii)* are satisfied. To establish condition *(iii)*, fix $q \in Q$ and $a \in \Sigma$. Suppose that for each $i \in loc(a)$ there is a transition $s_i \xrightarrow{a} s'_i$ such that $s_i \equiv_i q$. Clearly, for each $i \in loc(a)$, $s_i \xrightarrow{a} s'_i$ implies $s_i[i] \xrightarrow{a}_i s'_i[i]$. Moreover $s_i[i] = q[i]$ by the definition of $\equiv_i$. Since TS is a product transition system, this implies $q \xrightarrow{a} q'$, where $q'[i] = s'_i[i]$ for $i \in loc(a)$ and $q'[i] = q[i]$ otherwise.

($\Leftarrow$) : Suppose we are given equivalence relations $\{\equiv_i \subseteq (Q \times Q)\}_{i \in [1..k]}$ which satisfy conditions *(i)—(iii)*. For each $q \in Q$ and $i \in [1..k]$, let $[q]_i \stackrel{\text{def}}{=} \{s \mid s \equiv_i q\}$. For $i \in [1..k]$, define the transition system $TS_i = (Q_i, \rightarrow_i, q^i_{\text{in}})$ over Σ_i as follows:

- $Q_i = \{[q]_i \mid q \in Q\}$, with $q^i_{\text{in}} = [q_{\text{in}}]_i$.
- $[q]_i \xrightarrow{a}_i [q']_i$ iff $a \in \Sigma_i$ and there exists $s \xrightarrow{a} s'$ with $s \equiv_i q$ and $s' \equiv_i q'$.

We wish to show that TS is isomorphic to $\|_{i \in [1..k]} TS_i$. Let $\|_{i \in [1..k]} TS_i = (\widehat{Q}, \rightsquigarrow, \hat{q}_{\text{in}})$. We claim that the required isomorphism is given by the function $f : Q \rightarrow \widehat{Q}$, where $f(q) = ([q]_1, \ldots, [q]_k)$.

- We can show that f is well-defined—that is $f(q) \in \widehat{Q}$ for each q—by induction on the length of the shortest path from q_{in} to q. We omit the details.
- We next establish that f is a bijection. Clearly condition *(ii)* implies that f is injective. To argue that f is onto, let $([s_1]_1, \ldots, [s_k]_k) \in \widehat{Q}$ be reachable from $\hat{q}_{\text{in}}$ in n steps. We proceed by induction on n.
  - *Basis:* If $n = 0$, $([s_1]_1, \ldots, [s_k]_k) = \hat{q}_{\text{in}} = f(q_{in})$.
  - *Induction step:*
    Let $([r_1]_1, \ldots, [r_k]_k)$ be reachable from $\hat{q}_{\text{in}}$ in $n-1$ steps. Consider a move $([r_1]_1, \ldots, [r_k]_k) \stackrel{a}{\rightsquigarrow} ([s_1]_1, \ldots, [s_k]_k)$. By the induction hypothesis there exists $q \in Q$ such that $f(q) = ([q]_1, \ldots, [q]_k) = ([r_1]_1, \ldots, [r_k]_k)$. Now, $([r_1]_1, \ldots, [r_k]_k) \stackrel{a}{\rightsquigarrow} ([s_1]_1, \ldots, [s_k]_k)$ implies that $[r_i]_i \xrightarrow{a}_i [s_i]_i$ for each $i \in loc(a)$. Hence, for each $i \in loc(a)$, there exist r'_i, s'_i such that $r'_i \equiv_i r_i$, $s'_i \equiv_i s_i$ and $r'_i \xrightarrow{a} s'_i$. By condition *(iii)*, since $q \equiv_i r_i$, for any choice of such r'_i's and s'_i's there exists q' such that $q \xrightarrow{a} q'$ and $q' \equiv_i s'_i$ for each $i \in loc(a)$. We want to show that $f(q') = ([s_1]_1, \ldots, [s_k]_k)$. For $i \in loc(a)$ we already know that $q' \equiv_i s'_i \equiv_i s_i$. So suppose $i \notin loc(a)$. In this case $q \xrightarrow{a} q'$ implies $[q']_i = [q]_i$, by condition *(i)*. From $[q]_i = [r_i]_i$ and $[r_i]_i = [s_i]_i$ it follows that $[q']_i = [s_i]_i$.
- It is now easy to argue that f is an isomorphism—we omit the details.

An effective synthesis procedure

Observe that Theorem 3.1 yields an effective synthesis procedure for finite-state specifications which is exponential in the size of the original transition system and the number of components in the distributed alphabet. The number of ways of partitioning a finite-state space using equivalence relations is bounded and we can exhaustively check each choice to see if it meets criteria *(i)–(iii)* in the statement of the theorem.

4 Synthesis and independence relations

An abstract way of enriching a labelled transition system with information about concurrency is to equip the underlying alphabet with an independence relation—intuitively, this relation specifies which pairs of actions in the system can be executed independent of each other.

Definition 4.1. *An* independence relation *over Σ is a symmetric, irreflexive relation $I \subseteq \Sigma \times \Sigma$.*

Each distribution $\widetilde{\Sigma} = \langle \Sigma_1, \ldots, \Sigma_k \rangle$ induces a natural independence relation $I_{\widetilde{\Sigma}}$ over Σ—two actions are independent if they are performed at nonoverlapping sets of locations across the system. Formally, for $a, b \in \Sigma$, $a \; I_{\widetilde{\Sigma}} \; b \Leftrightarrow loc(a) \cap loc(b) = \emptyset$. The following example shows that different distributions may yield the same independence relation.

Example 4.2. If $\Sigma = \{a, b, c, d\}$ and $I = \{(a, b), (b, a)\}$, then the distributions $\widetilde{\Sigma} = \langle \{a, c, d\}, \{b, c, d\} \rangle$, $\widetilde{\Sigma}' = \langle \{a, c, d\}, \{b, c\}, \{b, d\} \rangle$ and $\widetilde{\Sigma}'' = \langle \{a, c\}, \{a, d\}, \{b, c\}, \{b, d\}, \{c, d\} \rangle$ all give rise to the independence relation I.

However for each independence relation I there is a *standard distribution* inducing I, whose components are the maximal cliques of the dependency relation $D = (\Sigma \times \Sigma) - I$. In the example above, the standard distribution is the one denoted $\widetilde{\Sigma}$. Henceforth, we will denote the standard distribution for I by $\widetilde{\Sigma}_I$.

The synthesis problem with respect to an independence relation

We can phrase the synthesis problem in terms of independence relations as follows. Given a transition system $TS = (Q, \rightarrow, q_{\text{in}})$ and a nonempty independence relation I over Σ, does there exist an *I-implementation* of TS, that is a $\widetilde{\Sigma}$-implementation of TS such that $\widetilde{\Sigma}$ induces I?

We show that if a transition system admits a $\widetilde{\Sigma}$-implementation then it also admits a $\widetilde{\Sigma}_{I_{\widetilde{\Sigma}}}$-implementation. Thus the synthesis problem with respect to independence relations reduces to the synthesis problem for standard distributions.

We begin by showing that a system that has an implementation over a distribution $\widetilde{\Sigma}$ also has an implementation over any coarser distribution $\widetilde{\Gamma}$, obtained by merging some components of $\widetilde{\Sigma}$ and possibly adding some new ones.

Definition 4.3. *Let $\widetilde{\Sigma} = \langle \Sigma_1, \ldots, \Sigma_k \rangle$ and $\widetilde{\Gamma} = \langle \Gamma_1, \ldots, \Gamma_\ell \rangle$ be distributions of Σ. Then $\widetilde{\Sigma} \lesssim \widetilde{\Gamma}$ if for each $i \in [1..k]$, there exists $j \in [1..\ell]$ such that $\Sigma_i \subseteq \Gamma_j$. If $\widetilde{\Sigma} \lesssim \widetilde{\Gamma}$ we say that $\widetilde{\Sigma}$ is* finer *than $\widetilde{\Gamma}$, or $\widetilde{\Gamma}$ is* coarser *than $\widetilde{\Sigma}$.*

We then have the following simple observation.

Proposition 4.4. *If $\widetilde{\Sigma} \lesssim \widetilde{\Gamma}$ then $I_{\widetilde{\Gamma}} \subseteq I_{\widetilde{\Sigma}}$.*

Note that $\lesssim$ is not a preorder in general. In fact $\widetilde{\Sigma} \lesssim \widetilde{\Gamma}$ means that the maximal elements of $\widetilde{\Sigma}$ are included in those of $\widetilde{\Gamma}$. Let us denote by $\simeq$ the relation $\lesssim \cap \gtrsim$. Then, $\widetilde{\Sigma} \simeq \widetilde{\Gamma}$ just means that $\widetilde{\Sigma}$ and $\widetilde{\Gamma}$ have the same maximal elements—in general, it does not guarantee that they are identical. However, when restricted to distributions "without redundancies", $\lesssim$ becomes a preorder.

Definition 4.5. *A distribution $\widetilde{\Sigma} = \langle \Sigma_1, \ldots, \Sigma_k \rangle$ of Σ is said to be* simple *if for each $i, j \in [1..k]$, $i \neq j$ implies that $\Sigma_i \not\subseteq \Sigma_j$.*

Proposition 4.6. *Let $\widetilde{\Sigma}$ and $\widetilde{\Gamma}$ be simple distributions of Σ. If $\widetilde{\Sigma} \simeq \widetilde{\Gamma}$ then $\widetilde{\Sigma} = \widetilde{\Gamma}$.*

For any independence relation I over Σ, the associated standard distribution $\widetilde{\Sigma}_I$ is a simple distribution, and is the coarsest distribution inducing I. At the other end of the spectrum, we can define the *finest* distribution inducing I as follows:

Definition 4.7. *Let I be an independence relation over Σ, and $D = (\Sigma \times \Sigma) - I$. The distribution $\widetilde{\Delta}_I$ over Σ is defined by:*

$$\widetilde{\Delta}_I = \{\{x, y\} \mid (x, y) \in D, x \neq y\} \cup \{\{x\} \mid x \ I \ y \text{ for each } y \neq x\}$$

Proposition 4.8. *Let I be an independence relation over Σ. Then the distribution $\widetilde{\Delta}_I$ is the finest simple distribution over Σ that induces I.*

A finer distribution can be faithfully implemented by a coarser distribution.

Lemma 4.9. *Let $\widetilde{\Sigma} = \langle \Sigma_1, \ldots, \Sigma_k \rangle$ and $\widetilde{\Gamma} = \langle \Gamma_1, \ldots, \Gamma_\ell \rangle$ be distributions of Σ such that $\widetilde{\Sigma} \lesssim \widetilde{\Gamma}$. Then, for each product transition system $\|_{i \in [1..k]} TS_i$ over $\widetilde{\Sigma}$, there exists an isomorphic product transition system $\|_{i \in [1..\ell]} \widehat{TS}_i$ over $\widetilde{\Gamma}$.*

Proof. For each $i \in [1..k]$ let $f(i)$ denote the least index j in $[1..\ell]$ such that $\Sigma_i \subseteq \Gamma_j$. For each $j \in [1..\ell]$, define $\widehat{TS}_j = (\widehat{Q}_j, \rightsquigarrow_j, \hat{q}^j_{\text{in}})$ as follows.

– If j is not in the range of f, then $\widehat{Q}_j = \{\hat{q}^j_{\text{in}}\}$, and $\hat{q}^j_{\text{in}} \overset{a}{\rightsquigarrow}_j \hat{q}^j_{\text{in}}$ for each $a \in \Sigma_j$.
– If j is in the range of f, let $f^{-1}(j) = \{i_1, i_2, \ldots, i_m\}$. Set $\widehat{TS}_j = (TS_{i_1} \| \cdots \| TS_{i_m})$.

It is then straightforward to verify that $\|_{i \in [1..k]} TS_i$ is isomorphic to $\|_{i \in [1..\ell]} \widehat{TS}_i$. We omit the details.

Corollary 4.10. *Let $TS = (Q, \rightarrow, q_{\text{in}})$ be a transition system over Σ, and $\widetilde{\Sigma}$ and $\widetilde{\Gamma}$ be two distributions of Σ such that $\widetilde{\Sigma} \lesssim \widetilde{\Gamma}$. If TS is $\widetilde{\Sigma}$-implementable it is also $\widetilde{\Gamma}$-implementable.*

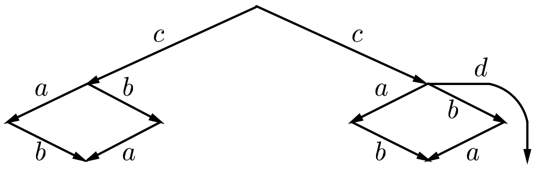
Let I be an independence relation over Σ. We have already observed that $\widetilde{\Sigma}_I$ is the coarsest distribution of Σ whose induced independence relation is I. Coupling this remark with the preceding corollary, we can now settle the synthesis problem with respect to independence relations.

Corollary 4.11. *Let $TS = (Q, \rightarrow, q_{\text{in}})$ be a transition system over Σ, and $\widetilde{\Sigma}$ be a distribution of Σ inducing the independence relation I. Then if TS is $\widetilde{\Sigma}$-implementable it is also $\widetilde{\Sigma}_I$-implementable. Moreover TS is I-implementable if and only if it is $\widetilde{\Sigma}_I$-implementable.*

We remark that the converse of Lemma 4.9 is not true—if $\widetilde{\Sigma} \lesssim \widetilde{\Gamma}$, it may be the case that TS is $\widetilde{\Gamma}$-implementable but not $\widetilde{\Sigma}$-implementable. Details can be found in the full paper [CMT99].

5 Canonical implementations and determinacy

A system may have more than one $\widetilde{\Sigma}$-implementation for a fixed distribution $\widetilde{\Sigma}$. For instance, the system



has two implementations with respect to the distributed alphabet $\widetilde{\Sigma} = \langle\{a,c,d\}, \{b,c,d\}\rangle$, namely $ca + c(a+d) \parallel c(b+d)$ and $c(a+d) \parallel cb + c(b+d)$.

One question that naturally arises is whether there exists a unique minimal or maximal family of equivalence relations $\{\equiv_1, \ldots, \equiv_k\}$ on states that makes Theorem 3.1 go through. We say that a family $\{\equiv_1, \ldots, \equiv_k\}$ is *minimal* (respectively, *maximal*) if there is no other family $\{\equiv'_1, \ldots, \equiv'_k\}$ with respect to which TS is $\widetilde{\Sigma}$-implementable with $\equiv'_i \subseteq \equiv_i$ (respectively, $\equiv_i \subseteq \equiv'_i$) for each $i \in [1..k]$.

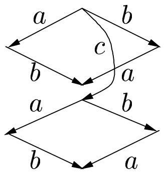
It turns out that for deterministic systems, we can find unique minimal implementations. A transition system $TS = (Q, \rightarrow, q_{\rm in})$ is *deterministic* if $q \stackrel{a}{\longrightarrow} q'$ and $q \stackrel{a}{\longrightarrow} q''$ imply that $q' = q''$.

Theorem 5.1. *Suppose $TS = (Q, \rightarrow, q_{\rm in})$ is deterministic and $\widetilde{\Sigma}$-implementable. Then there exists a unique minimal family $\{\equiv_1, \ldots, \equiv_k\}$ of equivalence relations on states with respect to which TS is $\widetilde{\Sigma}$-implementable.*

Proof. Suppose that $\{\equiv_1, \ldots, \equiv_k\}$ and $\{\equiv'_1, \ldots, \equiv'_k\}$ are two families which represent $\widetilde{\Sigma}$-implementations of TS. Let $\{\widehat{\equiv}_1, \ldots, \widehat{\equiv}_k\}$ be the intersection family, given by $q \mathrel{\widehat{\equiv}_i} q' \Leftrightarrow q \equiv_i q' \wedge q \equiv'_i q'$.

By definition, $\widehat{\equiv}_i \subseteq \equiv_i$ and $\widehat{\equiv}_i \subseteq \equiv'_i$. Thus, it suffices to show that $\{\widehat{\equiv}_1, \ldots, \widehat{\equiv}_k\}$ represents a $\widetilde{\Sigma}$-implementation of TS. From the definition of the relations $\{\widehat{\equiv}_1, \ldots, \widehat{\equiv}_k\}$, it is obvious that both conditions *(i)* and *(ii)* of Theorem 3.1 are satisfied. Now suppose that $q \in Q$ and $a \in \Sigma$. For every $i \in loc(a)$, let $s_i, s'_i \in Q$ be such that $s_i \mathrel{\widehat{\equiv}_i} q$ and $s_i \stackrel{a}{\longrightarrow} s'_i$. This means that for every $i \in loc(a)$, both $s_i \equiv_i q$ and $s_i \equiv'_i q$. Hence by condition *(iii)* there exists q' such that $q \stackrel{a}{\longrightarrow} q'$ and $q' \equiv_i s'_i$ for every $i \in loc(a)$. Similarly, there exists q'' such that $q \stackrel{a}{\longrightarrow} q''$ and $q'' \equiv_i s'_i$ for every $i \in loc(a)$. Since TS is deterministic, it must be $q' = q''$. Thus, $q' = q''$ is such that $q' \mathrel{\widehat{\equiv}_i} s'_i$ for each $i \in loc(a)$.

This result leads us to conjecture that the synthesis problem for deterministic systems is much less expensive computationally than the synthesis problem in the general case, since it suffices to look for the unique minimal family of equivalence relations which describe the implementation.



We conclude this section by observing that a deterministic system may have more than one maximal family $\{\equiv_1, \ldots, \equiv_k\}$ for which it is $\widetilde{\Sigma}$-implementable. For instance the system on the left has two distinct maximal implementations with respect to the distribution $\langle\{a, c\}, \{b, c\}\rangle$, namely (fix $X.\ a + cX) \parallel b + cb$ and $a + ca \parallel$ (fix $Y.\ b + cY$), whose components are not even language equivalent.

6 Synthesis modulo bisimulation

In the course of specifying a system, we may accidentally destroy its inherent product structure. This may happen, for example, if we optimize the design and eliminate redundant states. In such situations, we would like to be able to reconstruct a product transition system from the reduced specification. Since the synthesized system will not, in general, be isomorphic to the specification, we need a criterion for ensuring that the two systems are behaviourally equivalent. We use strong bisimulation [Mil89] for this purpose.

In general, synthesizing a behaviourally equivalent product implementation from a reduced specification appears to be a hard problem. In this section, we show how to solve the problem for reduced specifications which can be implemented as *deterministic* product transition systems—that is, the global transition system generated by the implementation is deterministic. Notice that the specification itself may be nondeterministic. Since many distributed systems implemented in hardware, such as digital controllers, are actually deterministic, our characterization yields a synthesis result for a large class of useful systems.

We begin by recalling the definition of bisimulation.

Definition 6.1. *A* bisimulation *between a pair of transition systems* $TS_1 = (Q_1, \rightarrow_1, q^1_{\rm in})$ *and* $TS_2 = (Q_2, \rightarrow_2, q^2_{\rm in})$ *is a relation* $R \subseteq (Q_1 \times Q_2)$ *such that:*

- $(q^1_{\rm in}, q^2_{\rm in}) \in R$.
- *If* $(q_1, q_2) \in R$ *and* $q_1 \xrightarrow{a}_1 q_1'$, *there exists* q_2', $q_2 \xrightarrow{a}_2 q_2'$ *and* $(q_1', q_2') \in R$.
- *If* $(q_1, q_2) \in R$ *and* $q_2 \xrightarrow{a}_2 q_2'$, *there exists* q_1', $q_1 \xrightarrow{a}_1 q_1'$ *and* $(q_1', q_2') \in R$.

The synthesis problem modulo bisimilarity

The synthesis problem modulo bisimilarity can now be formulated as follows. If $TS = (Q, \rightarrow, q_{\rm in})$ is a transition system over Σ, and $\widetilde{\Sigma} = \langle \Sigma_1, \ldots, \Sigma_k \rangle$ is a distribution of Σ, does there exist a product system $\|_{i\in[1..k]}\ TS_i$ over $\widetilde{\Sigma}$ such that $\|_{i\in[1..k]}\ TS_i$ is bisimilar to TS?

To settle this question for deterministic implementations, we need to consider product languages.

Languages Let $TS = (Q, \rightarrow, q_{\rm in})$ be a transition system over Σ. The *language* of TS is the set $L(TS) \subseteq \Sigma^*$ consisting of the labels along all runs of TS. In other words, $L(TS) = \{w \mid q_{\rm in} \xrightarrow{w} q, q \in Q\}$.

Notice that $L(TS)$ is always prefix-closed and always contains the empty word. Moreover, $L(TS)$ is regular whenever TS is finite. For the rest of this section, we assume all transition systems which we encounter are finite.

Product languages Let $L \subseteq \Sigma^*$ and let $\widetilde{\Sigma} = \langle \Sigma_1, \ldots, \Sigma_k \rangle$ be a distribution of Σ. For $w \in \Sigma^*$, let $w \restriction_{\Sigma_i}$ denote the projection of w onto Σ_i, obtained by erasing all letters in w which do not belong to Σ_i.

The language L is a *product language* over $\widetilde{\Sigma}$ if for each $i \in [1..k]$ there is a language $L_i \subseteq \Sigma_i^*$ such that $L = \{w \mid w \restriction_{\Sigma_i} \in L_i, i \in [1..k]\}$.

For deterministic transition systems, bisimilarity coincides with language equivalence. We next show that we can extend this result to get a simple characterization of transition systems which are bisimilar to deterministic product transition systems. We first recall a basic definition.

Bisimulation quotient Let $TS = (Q, \rightarrow, q_{\mathrm{in}})$ be a transition system and let $\sim_{TS}$ be the *largest* bisimulation relation between TS and itself. The relation $\sim_{TS}$ defines an equivalence relation over Q. For $q \in Q$, let $[q]$ denote the $\sim_{TS}$-equivalence class containing q. The *bisimulation quotient* of TS is the transition system $TS/_{\sim_{TS}} = (\widehat{Q}, \rightsquigarrow, [q_{\mathrm{in}}])$ where

- $\widehat{Q} = \{[q] \mid q \in Q\}$.
- $[q] \overset{a}{\rightsquigarrow} [q']$ if there exist $q_1 \in [q]$ and $q_1' \in [q']$ such that $q_1 \xrightarrow{a} q_1'$.

The main result of this section is the following.

Theorem 6.2. *Let TS be a transition system over Σ and let $\widetilde{\Sigma}$ be a distribution of Σ. The system TS is bisimilar to a deterministic product transition system over $\widetilde{\Sigma}$ iff TS satisfies the following two conditions.*

- *The bisimulation quotient $TS/_{\sim_{TS}}$ is deterministic.*
- *The language $L(TS)$ is a product language over $\widetilde{\Sigma}$.*

To prove this theorem, we first recall the following basic connection between product languages and product systems [Thi95].

Lemma 6.3. $L(\|_{i\in[1..i]} TS_i) = \{w \mid w\restriction_{\Sigma_i} \in L(TS_i), i \in [1..k]\}$.

We also need the useful fact that a product language is always the product of its projections [Thi95].

Lemma 6.4. *Let $L \subseteq \Sigma^*$ and let $\widetilde{\Sigma} = \langle \Sigma_1, \ldots, \Sigma_k \rangle$ be a distribution of Σ. For $i \in [1..k]$, let $L_i = \{w \restriction_{\Sigma_i} \mid w \in L\}$. Then, L is a product language iff $L = \{w \mid w \restriction_{\Sigma_i} \in L_i, i \in [1..k]\}$.*

Notice that Lemma 6.4 yields an effective procedure for checking if a finite-state transition system accepts a product language over a distribution $\langle \Sigma_1, \ldots, \Sigma_k \rangle$. For $i \in [1..k]$, construct the finite-state system TS_i such that $L(TS_i) = L\restriction_{\Sigma_i}$ and then verify that $L(TS) = L(\|_{i\in[1..k]} TS_i)$.

Next, we state without proof some elementary facts about bisimulations.

Lemma 6.5. *(i) Let TS be a deterministic transition system. Then, $TS/_{\sim_{TS}}$ is also deterministic.*

(ii) Let TS_1 and TS_2 be deterministic transition systems over Σ. If $L(TS_1) = L(TS_2)$ then TS_1 is bisimilar to TS_2.

(iii) Let TS_1 and TS_2 be bisimilar transition systems. Then $TS_1/_{\sim_{TS_1}}$ and $TS_2/_{\sim_{TS_2}}$ are isomorphic. Further $L(TS_1) = L(TS_2)$.

We now prove both parts of Theorem 6.2.

Lemma 6.6. *Suppose that a transition system TS is bisimilar to a deterministic product transition system. Then, $TS/_{\sim_{TS}}$ is deterministic and $L(TS)$ is a product language.*

Proof. Let $\widehat{TS} =\|_{i\in[1..i]} \widehat{TS}_i$ be a deterministic product transition system such that TS is bisimilar to $\widehat{TS}$.

By Lemma 6.5 (iii), $L(TS) = L(\widehat{TS})$. Since $L(\widehat{TS})$ is a product language, it follows that $L(TS)$ is a product language.

To check that $TS/_{\sim_{TS}}$ is deterministic, we first observe that $\widehat{TS}/_{\sim_{\widehat{TS}}}$ is deterministic, by Lemma 6.5 (i). By part (iii) of the same lemma, $TS/_{\sim_{TS}}$ must be isomorphic to $\widehat{TS}/_{\sim_{\widehat{TS}}}$. Hence, $TS/_{\sim_{TS}}$ is also deterministic.

Lemma 6.7. *Let TS be a transition system over Σ and $\widetilde{\Sigma}$ be a distribution of Σ, such that $TS/_{\sim_{TS}}$ is deterministic and $L(TS)$ is a product language over $\widetilde{\Sigma}$. Then, TS is bisimilar to a deterministic product transition system over $\widetilde{\Sigma}$.*

Proof. Let $\widetilde{\Sigma} = \langle \Sigma_1, \ldots, \Sigma_k \rangle$. For $i \in [1..k]$, let $L_i = \{w{\restriction}_{\Sigma_i} \mid w \in L(TS)\}$. We know that each L_i is a regular prefix-closed language which contains the empty word. Thus, we can construct the minimal deterministic finite-state automaton $A_i = (Q_i, \to_i, q^i_{\text{in}}, F_i)$ recognizing L_i. Since L_i contains the empty word, $q^i_{\text{in}} \in F_i$. Consider the restricted transition relation $\to'_i = \to_i \cap (F_i \times \Sigma \times F_i)$. It is easy to verify that the transition system $TS_i = (F_i, \to'_i, q^i_{\text{in}})$ is a deterministic transition system such that $L(TS_i) = L_i$.

Consider the product $\widehat{TS} =\|_{i\in[1..k]} TS_i$. Lemma 6.3 tell us that $L(\widehat{TS}) = \{w \mid w{\restriction}_{\Sigma_i} \in L_i, i \in [1..k]\}$. From Lemma 6.4, it follows that $L(\widehat{TS}) = L(TS)$.

We claim that TS is bisimilar to $\widehat{TS}$. Consider the quotient $TS/_{\sim_{TS}}$. Both $TS/_{\sim_{TS}}$ and $\widehat{TS}$ are deterministic and $L(TS/_{\sim_{TS}}) = L(TS) = L(\widehat{TS})$. Thus, by Lemma 6.5 (ii), it must be the case that $TS/_{\sim_{TS}}$ is bisimilar to $\widehat{TS}$. By transitivity, TS is also bisimilar to $\widehat{TS}$.

Using standard automata theory, we can derive the following result from Theorem 6.2.

Corollary 6.8. *Given a finite-state transition system $TS = (Q, \to, q_{\text{in}})$ and a distributed alphabet $\widetilde{\Sigma}$, we can effectively decide whether TS is bisimilar to a deterministic product system over $\widetilde{\Sigma}$.*

The synthesis problem modulo bisimilarity appears to be quite a bit more difficult when the product implementation is permitted to be nondeterministic. We have a characterization of the class of systems which are bisimilar to nondeterministic product systems [CMT99]. Our characterization is phrased in terms of the structure of the execution tree obtained by unfolding the specification. Unfortunately, the execution tree may be infinite, so this characterization is not effective, even if the initial specification is finite-state. The difficulty lies in bounding the number of transitions in the product implementation which can collapse, via the bisimulation, to a single transition in the specification.

References

[Arn94] A. Arnold: *Finite transition systems and semantics of communicating sytems*, Prentice-Hall (1994).

[BD98] E. Badouel and Ph. Darondeau: Theory of Regions. *Lectures on Petri nets I (Basic Models), LNCS* **1491** (1998) 529–588.

[CMT99] I. Castellani, M. Mukund and P.S. Thiagarajan: Characterizing decomposable transition systems, *Internal Report, Chennai Mathematical Institute* (1999).

[ER90] A. Ehrenfeucht and G. Rozenberg: Partial 2-structures; Part II, State spaces of concurrent systems, *Acta Inf.* **27** (1990) 348–368.

[GM92] J.F. Groote and F. Moller: Verification of Parallel Systems via Decomposition, *Proc. CONCUR'92, LNCS* **630** (1992) 62–76.

[Hol97] G.J. Holzmann: The model checker SPIN, *IEEE Trans. on Software Engineering*, **23**, 5 (1997) 279–295.

[Kur94] R.P. Kurshan: *Computer-Aided Verification of Coordinating Processes: The Automata-Theoretic Approach*, Princeton University Press (1994).

[Maz89] A. Mazurkiewicz: Basic notions of trace theory, in: J.W. de Bakker, W.-P. de Roever, G. Rozenberg (eds.), *Linear time, branching time and partial order in logics and models for concurrency, LNCS* **354** (1989) 285–363.

[Mil89] R. Milner: *Communication and Concurrency*, Prentice-Hall, London (1989).

[Mol89] F. Moller: *Axioms for Concurrency*, Ph.D. Thesis, University of Edinburgh (1989).

[MM93] F. Moller and R. Milner: Unique decomposition of processes, *Theor. Comput. Sci.* **107** (1993) 357–363.

[Mor98] R. Morin: Decompositions of asynchronous systems, *Proc. CONCUR'98, LNCS* **1466** (1998) 549–564.

[Muk92] M. Mukund: Petri Nets and Step Transition Systems, *Int. J. Found. Comput. Sci.* **3**, 4 (1992) 443–478.

[NRT92] M. Nielsen, G. Rozenberg and P.S. Thiagarajan: Elementary transition systems, *Theor. Comput. Sci.* **96** (1992) 3–33.

[Thi95] P.S. Thiagarajan: A trace consistent subset of PTL, *Proc. CONCUR'95, LNCS* **962** (1995) 438-452.

[WN95] G. Winskel and M. Nielsen: Models for concurrency, in S. Abramsky, D. Gabbay and T.S.E. Maibaum, eds, *Handbook of Logic in Computer Science, Vol 4*, Oxford (1995) 1–148.