FROM GLOBAL SPECIFICATIONS TO DISTRIBUTED IMPLEMENTATIONS

Madhavan Mukund
Chennai Mathematical Institute
92 G N Chetty Road, Chennai 600 017, India
madhavan@cmi.ac.in

Abstract We study the problem of synthesizing distributed implementations from global specifications. We work in the framework of transition systems. The main question we address is the following.

> Given a global transition system TS over a set of actions Σ together with a distribution of Σ into local components $\langle \Sigma_1, \ldots, \Sigma_k \rangle$, does there exist a distributed transition system over $\langle \Sigma_1, \ldots, \Sigma_k \rangle$ that is *"equivalent"* to TS?

We focus on two different types of distributed transition systems—loosely cooperating systems and synchronously communicating systems. For "equivalence" we consider three possibilities—state-space isomorphism, language equivalence and bisimulation.

We survey the current state of knowledge about the different versions of the problem that arise from choosing a concrete notion of equivalence and a specific model of distributed transition systems.

1. Introduction

Designing distributed systems has always been a challenging task. Interactions between individual processes can introduce subtle errors in the system's overall behaviour that 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 [5] introduced the concept of *regions* to describe how to associate places of nets with states of a transition system. In [15], Nielsen, Rozenberg and Thiagarajan use regions to characterize the class of transition systems that arise from elementary net systems. Subsequently, several authors have extended this characterization to larger classes of nets (for a sample of the literature, see [2, 12, 17]).

Here, we consider *distributed transition systems*—networks of transition systems that coordinate their activity by synchronizing on common actions. We focus on two models of distributed transition systems, which we refer to as *loosely cooperating systems* and *synchronously communicating systems*. The first class has also been called synchronized product systems and has been widely studied [1]. The second class corresponds to Zielonka's asynchronous automata [18]. The names we have chosen for these two classes are intended to reflect more accurately the underlying structure of the two system models.

Both these models come with a natural notion of component and induced notions of concurrency and causality. These models have a well-understood theory, at least in the linear-time setting [4, 16, 18]. Variants of these models are also the basis for system descriptions in a number of model-checking tools [8, 7].

The synthesis problem can broadly be stated as follows:

> Given a global transition system TS over a set of actions Σ together with a distribution of Σ into local components $\langle \Sigma_1, \ldots, \Sigma_k \rangle$, does there exist a distributed transition system over $\langle \Sigma_1, \ldots, \Sigma_k \rangle$ that is *"equivalent"* to TS?

We consider three possible interpretations of the term "equivalence": state-space isomorphism, language equivalence and bisimulation. For the first two interpretations, we provide complete characterizations. The synthesis problem with respect to bisimulation is still open in general and we provide some positive results in the restricted setting where we are in search of deterministic implementations.

The paper is organized as follows. We begin by defining the two classes of distributed transition systems we will be looking at. Next, in Section 3, we define the three types of synthesis problems that we are interested in. These problems are then addressed in turn in Sections 4–6. In Section 7 we consider the synthesis problem in a more abstract setting, where concurrency is presented implicitly using an independence relation. We conclude with a short discussion of related work.

2. Distributed transition systems

We begin by defining labelled transition systems, a general framework for modelling computing systems.

Labelled transition system. Let Σ be a finite nonempty set of actions. A labelled transition system over Σ is a structure $TS = (Q, \rightarrow, q_{\mathrm{in}})$, 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 $q_0 \xrightarrow{a_1} \cdots \xrightarrow{a_n} q_n$ as $q_0 \xrightarrow{a_1 \cdots a_n} q_n$. In every transition system $TS = (Q, \rightarrow, q_{\mathrm{in}})$ that 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 coordinated through common actions. To formalize this, we begin with the notion of a distributed alphabet.

Distributed alphabet. 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 $loc_{\widetilde{\Sigma}}(a)$.

Without loss of generality, we may assume that $\Sigma_i \neq \Sigma_j$ for any pair of distinct components of a distributed alphabet. 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 $\{1, 2, \ldots, k\}$.

The first model of distributed transition systems that we consider is loosely cooperating systems. Each *process* of a loosely cooperating system is a local transition system over one component of a distributed alphabet. The global transition relation forces all processes that share an action a to move jointly when a occurs. This synchronization, however, does not involve any communication of "local" information between processes. Each process participating in a loosely cooperating transition is free to choose for itself any move that is available in its local transition relation, independent of how the other processes choose to move.

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

- $q_{\mathrm{in}} = (q^1_{\mathrm{in}}, \ldots, q^k_{\mathrm{in}})$.

- $Q \subseteq (Q_1 \times \cdots \times Q_k)$ 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 $j \notin loc(a)$.

A richer model is that of a synchronously communicating system. Here, as before, we have one process for each component of the distributed alphabet. However, the moves for each action are specified *jointly* for all processes that synchronize on that action. Thus, the local behaviour of each process is conditional on the behaviour of the other processes that it synchronizes with. Abstractly, this corresponds to the processes pooling together the information available to them locally and deciding on a combined move based on this shared information.

Synchronously communicating systems. Let $\langle \Sigma_1, \ldots, \Sigma_k \rangle$ be a distribution of Σ. For each $i \in [1..k]$, let P_i be a process with a finite set of local states Q_i that includes a distinguished initial state q^i_{in}. We associate with each action $a \in \Sigma$ a transition relation $\rightarrow_a \subseteq \prod_{i \in loc(a)} Q_i \times \prod_{i \in loc(a)} Q_i$.

The *synchronously communicating system* $TS_1 \parallel\!\mid \cdots \parallel\!\mid TS_k$ is the transition system $TS = (Q, \rightarrow, q_{\text{in}})$ over $\Sigma = \bigcup_{i \in [1..k]} \Sigma_i$ 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 as follows:
  - $q_{\text{in}} \in Q$.
  - Let $q \in Q$ and $a \in \Sigma$ such that $loc(a) = \{i_1, \ldots, i_m\}$. For $i \in [1..k]$, let $q[i]$ denote the i^{th} component of q. Then $q \xrightarrow{a} q'$ provided $\langle q[i_1], \ldots, q[i_m] \rangle \rightarrow_a \langle q'[i_1], \ldots, q'[i_m] \rangle$ and for $j \notin loc(a)$, $q[j] = q'[j]$.

Expressiveness of the two models

It is not difficult to see that every loosely cooperating system admits a description as a synchronously communicating system by setting each transition relation $\rightarrow_a$ to be the direct product of the local a-transitions $\rightarrow_i \cap (Q_i \times \{a\} \times Q_i)$ over all $i \in loc(a)$.

On the other hand, the following example (from [18]) shows that synchronously communicating systems are strictly more expressive than loosely cooperating systems.

Example 1 Let $\Sigma = \{a_1, b_1, a_2, b_2, c\}$ be distributed as $\langle\{a_1, b_1, c\}, \{a_2, b_2, c\}\rangle$. We can construct a synchronously communicating system that exhibits the following behaviour: Process P_i alternately performs action c followed by either a_i or b_i such that between any two c's (which are jointly performed by both processes) P_1 performs a_1 if and only if P_2 performs a_2. We can represent this behaviour as (the prefix closure of) an extended regular expression $[c \cdot (a_1 \parallel a_2 + b_1 \parallel b_2)]^*$, where $\parallel$ represents the shuffle operation.

To implement this behaviour as a synchronously communicating system, for $i \in \{1,2\}$ we set $Q_i = \{q_c^i, q_a^i, q_b^i\}$ and $q_{\text{in}}^i = q_c^i$. The transition relations are then defined as follows:

- $\langle q_c^1, q_c^2\rangle \rightarrow_c \langle q_a^1, q_a^2\rangle$, $\langle q_c^1, q_c^2\rangle \rightarrow_c \langle q_b^1, q_b^2\rangle$
- $q_a^1 \rightarrow_{a_1} q_c^1$, $q_b^1 \rightarrow_{b_1} q_c^1$
- $q_a^2 \rightarrow_{a_2} q_c^2$, $q_b^2 \rightarrow_{b_2} q_c^2$

In other words, when P_1 and P_2 perform c, the joint move coordinates their behaviour so that either they are constrained to next perform either a_1 and a_2 or b_1 and b_2, respectively.

Informally, the reason that this behaviour cannot be captured by a loosely cooperating system is that we would have to split up the $\rightarrow_c$ move into local components and then take the direct product of these local c-moves. This would introduce undesirable global moves on c actions, such as a move in which P_1 moves to q_a^1 where a_1 is enabled while P_2 simultaneously moves to q_b^2 where b_2 is enabled. A more formal argument will be presented later. □

3. The synthesis problem

Broadly speaking, the synthesis problem for distributed transition systems is the following. Let $TS = (Q, \rightarrow, q_{\text{in}})$ be a transition system over Σ and $\widetilde{\Sigma} = \langle\Sigma_1, \ldots, \Sigma_k\rangle$ be a distribution of Σ. Does there exist a distributed transition system over $\widetilde{\Sigma}$ that is *"equivalent"* to TS?

To make the problem precise, we have to specify what we mean by "equivalent". In this paper, we will look at three versions of "equivalence".

- *State space equivalence*: The state space of the distributed transition system should be *isomorphic* to the global transition system TS.
- *Language equivalence*: The distributed transition system should admit the same sequences of actions as the global transition system TS.
- *Bisimulation equivalence*: The state space of the distributed transition system should be *bisimilar* to the global transition system TS.

These three interpretations of the word "equivalence" yield six synthesis problems in all, since we are considering two different types of distributed transition systems.

Broadly speaking, the results known for these problems can be classified as follows:

- Synthesis modulo isomorphism can be solved precisely for both classes of distributed transition systems using the theory of *regions* [2, 3, 5].
- Synthesis modulo language equivalence can be solved for loosely cooperating systems in terms of a precise characterization of the languages generated by such systems, based on the projections onto the components of the distributed alphabet.

  For synchronously communicating systems, however, the synthesis problem modulo language equivalence is much harder and is a celebrated result in the theory of concurrent systems [18].
- Synthesis modulo bisimulation is still unsolved in the general case. A characterization is known for the relatively simple case where the synthesized distributed transition system is required to be deterministic [3].

4. Synthesis modulo isomorphism

Regions were introduced in [5] to characterize global state spaces that arise from local presentations such as Petri nets. Broadly speaking, a region of an edge-labelled graph is a subset of vertices that satisfies a suitable consistency criterion on the labels of edges that enter and leave the subset.

In our setting, the basic idea is to label in a consistent manner each state of the input transition system by a k-tuple of local states (corresponding to a global state of a distributed transition system). We formulate this labelling 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 agree on the i^{th} component. Our presentation is taken from [3]. A restricted form of this characterization was obtained in [11], for deterministic transition system specifications.

4.1 Loosely Cooperating systems

Theorem 2 *Let $TS = (Q, \rightarrow, q_{\text{in}})$ be a transition system over Σ and let $\widetilde{\Sigma} = \langle \Sigma_1, \ldots, \Sigma_k \rangle$ be a distribution of Σ. Then, there exists a loosely cooperating system $TS_1 \parallel \cdots \parallel TS_k$ over $\widetilde{\Sigma}$ whose state space is isomorphic to TS 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 Sketch: $(\Rightarrow)$: Suppose $TS_1 \parallel \cdots \parallel TS_k$ is isomorphic to 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 in fact equal to $TS_1 \parallel \cdots \parallel TS_k$.

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

Since TS is a loosely cooperating 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 loosely cooperating 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 $TS_i = (Q_i, \rightarrow_i, q_{\text{in}}^i)$ 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 $TS_1 \parallel \cdots \parallel TS_k$. Let $TS_1 \parallel \cdots \parallel TS_k = (\widehat{Q}, \rightsquigarrow, \hat{q}_{\text{in}})$. We claim that the required isomorphism is given by the function $f : Q \to \widehat{Q}$, where $f(q) = ([q]_1, \ldots, [q]_k)$. To complete the proof, we have to argue that f is well-defined and that it constitutes a bijection. This can be done by induction on the length of the shortest path from q_{in} to q for each state q in TS. The details can be found in [3].

□

Intuitively, condition (i) in the preceding characterization guarantees that the actions are distributed correctly among the processes. Condition (ii) is an extensionality condition—the global state of the system is no more than the product of the local states of the individual processes. Finally, condition (iii) captures the essence of loose cooperation—any combination of local choices for an a-move can be combined into a global a-move.

4.2 Synchronously communicating systems

For synchronously communicating systems, the characterization is similar. The only difference is in condition (iii), which has to be appropriately modified to take into account that joint moves in such systems can depend on contextual information about other processes involved in the move. However, this contextual information is independent of the states of processes *not* involved in the move. In other words, for any action a, the a-moves enabled at a global state depend only on the local states of the processes in $loc(a)$. Two global states that agree on their projections onto the processes in $loc(a)$ must have the same a-transitions enabled.

Theorem 3 *Let $TS = (Q, \to, q_{\text{in}})$ be a transition system over Σ and let $\widetilde{\Sigma} = \langle \Sigma_1, \ldots, \Sigma_k \rangle$ be a distribution of Σ. Then, there exists a synchronously communicating system $TS_1 \mathbin{|||} \cdots \mathbin{|||} TS_k$ over $\widetilde{\Sigma}$ whose state space is isomorphic to TS 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) *For $q, q' \in Q$ and $a \in \Sigma$, let $q \equiv_a q'$ denote that for each $i \in loc(a)$, $q \equiv_i q'$. Then, whenever $q \equiv_a q'$, if there is a move $q \xrightarrow{a} q_1$ then there is also a move $q' \xrightarrow{a} q'_1$ such that $q_1 \equiv_a q'_1$.*

An effective synthesis procedure. Observe that Theorems 2 and 3 yield effective synthesis procedures for finite-state specifications. 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 statements of the two theorems. In both cases, the number of states of each process in the resulting distributed transition system is exponential in the size of the original transition system and the number of components in the distributed alphabet.

5. Synthesis modulo language equivalence

Languages. Let $TS = (Q, \rightarrow, q_{\text{in}})$ 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, $L(TS) = \{w \mid q_{\text{in}} \xrightarrow{w} q, q \in Q\}$.

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 that we encounter are finite.

5.1 Loosely Cooperating systems

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 $L = \{w \mid w{\restriction}_{\Sigma_i} \in L_i, i \in [1..k]\}$.

We begin with the following basic connection between product languages and loosely cooperating systems [16].

Lemma 4 $L(TS_1 \parallel \cdots \parallel TS_i) = \{w \mid w{\restriction}_{\Sigma_i} \in L(TS_i), i \in [1..k]\}$.

It turns out that a product language is always the product of its projections [16].

Lemma 5 *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 if and only if $L = \{w \mid w{\restriction}_{\Sigma_i} \in L_i, i \in [1..k]\}$.*

The preceding result allows us to complete Example 1 showing that synchronously communicating systems are a richer class than loosely cooperating systems.

Example 1 (continued). Let $\Sigma = \{a_1, b_1, a_2, b_2, c\}$ be distributed as $(\{a_1, b_1, c\}, \{a_2, b_2, c\})$. Our goal was to establish that the behaviour described by the extended regular expression $[c \cdot (a_1 \parallel a_2 + b_1 \parallel b_2)]^*$, where $\parallel$ represents the shuffle operation, cannot be implemented as a loosely cooperating system.

To verify this, we argue that this is not a product language. It is easy to see that the projections of the desired behaviour onto Σ_1 and Σ_2 yield the languages $L_1 = [c \cdot (a_1 + b_1)]^*$ and $L_2 = [c \cdot (a_2 + b_2)]^*$. However, the projections of ca_1b_2 also lie in L_1 and L_2, although ca_1b_2 is not a string in the language defined by the expression $[c \cdot (a_1 \parallel a_2 + b_1 \parallel b_2)]^*$. $\square$

We can now state the synthesis result for loosely cooperating systems modulo language equivalence.

Theorem 6 *Let $TS = (Q, \rightarrow, q_{\text{in}})$ be a finite-state transition system over Σ and let $\widetilde{\Sigma} = \langle \Sigma_1, \ldots, \Sigma_k \rangle$ be a distribution of Σ. Then, we can effectively decide whether there exists a loosely cooperating system $TS_1 \parallel \cdots \parallel TS_k$ over $\widetilde{\Sigma}$ such that $L(TS_1 \parallel \cdots \parallel TS_k) = L(TS)$.*

Proof: Lemma 5 yields following decision procedure. For $i \in [1..k]$, construct the finite-state system TS_i such that $L(TS_i) = L{\restriction}_{\Sigma_i}$. This can be done by relabelling all moves not in Σ_i by ε and then performing an ε-closure on the resulting system. Checking whether $L(TS) = L(TS_1 \parallel \cdots \parallel TS_k)$ is then just an instance of the language equivalence problem for finite-state automata. $\square$

5.2 Synchronously communicating systems

Concurrent alphabet. A distributed alphabet $\langle \Sigma_1, \ldots, \Sigma_n \rangle$ gives rise to a natural *independence relation* I_{loc} between letters: $(a, b) \in I_{loc}$ if and only if $loc(a) \cap loc(b) = \emptyset$. Thus, a and b are independent when they are performed by disjoint sets of processes in the system. Clearly, the relation I_{loc} is irreflexive and symmetric. Such a relation is called an independence relation.

An alphabet equipped with an independence relation is also called a *concurrent alphabet*. This notion was introduced by Mazurkiewicz as a technique for studying concurrent systems from the viewpoint of formal language theory [9].

Traces and trace languages. Given a concurrent alphabet (Σ, I), I induces a natural equivalence relation $\sim$ on Σ^*: two words w and w' are related by $\sim$ if and only if w' can be obtained from w by a sequence of permutations of adjacent independent letters. More formally, $w \sim w'$ if there is a sequence of words $v_1, \ldots, v_k$ such that $w = v_1$, $w' = v_k$ and for each $i \in [1..k-1]$, there exist words u_i, u_i' and letters a_i, b_i satisfying

$$v_i = u_i a_i b_i u_i', \;\; v_{i+1} = u_i b_i a_i u_i' \;\; \textit{and}\; (a_i, b_i) \in I.$$

Actually, $\sim$ defines a *congruence* on Σ^* with respect to concatenation: If $u \sim u'$ then for any words w_1 and w_2, $w_1 u w_2 \sim w_1 u' w_2$. Also, both right and left cancellation preserve $\sim$-equivalence: $wu \sim wu'$ implies $u \sim u'$ and $uw \sim u'w$ implies $u \sim u'$.

Equivalence classes of words of Σ^* under $\sim$ and are called *traces*. Let $[w]$ denote the $\sim$-equivalence class corresponding to the word w. Since the relation $\sim$ is a congruence, the composition operation on traces is given by

$$\forall u, v \in \Sigma^*.\; [u][v] = [uv].$$

Recognizable trace languages. Sets of traces are called *trace languages*—in other words, a trace language over (Σ, I) is a language of words over Σ that is closed with respect to the equivalence $\sim$. A *recognizable trace language* is a recognizable (or regular) language over Σ that is closed with respect to $\sim$.

Recognizable trace languages can be characterized in terms of the structure of the minimum DFAs that accept these languages. Let $L \subseteq \Sigma^*$ be a recognizable string language and let $A_L = (S, \Sigma, \delta, s_0, S_F)$ be the minimum DFA for L. As usual, for $w = a_1 \ldots a_m$, we let $\delta(q, w)$ denote the unique state reached by A_L on reading w at state q. Then, it is a well-known fact that for each pair of words w_1, w_2 over Σ, $\delta(s_{\text{in}}, w_1) = \delta(s_{\text{in}}, w_2)$ if and only if $w_1 \equiv_{R_L} w_2$, where $\equiv_{R_L}$ is the right-equivalence relation defined by L, given by

$$\forall u, u' \in \Sigma^*.\; u \equiv_{R_L} u' \stackrel{\text{def}}{=} \forall v \in \Sigma^*.\; uv \in L \text{ iff } u'v \in L$$

If L is a recognizable trace language and $w \sim w'$, then it is easy to see that $w \equiv_{R_L} w'$ as well. From this it follows that if L is a recognizable trace language and $A_L = (S, \Sigma, \delta, s_0, S_F)$ is the minimum DFA for L, then for each $w \in \Sigma^*$, for each $w' \in [w]$, $\delta(s_{\text{in}}, w) = \delta(s_{\text{in}}, w')$. In particular, this means that for each state $s \in S$ and each pair of independent letters $(a, b) \in I$, it must be the case that $\delta(s, ab) = \delta(s, ba)$. In fact, this turns out to be a characterization of recognizable trace languages.

Lemma 7 *Let L be recognizable subset of Σ^*. Let $A_L = (S, \Sigma, \delta, s_0, S_F)$ be the minimum DFA for L. Then, L is a recognizable trace language over (Σ, I) if and only if for each $s \in S$ and each pair of letters $(a, b) \in I$, $\delta(s, ab) = \delta(s, ba)$.*

A celebrated theorem of Zielonka [18] states that if we are given a distribution $\widetilde{\Sigma} = \langle \Sigma_1, \ldots, \Sigma_k \rangle$ of Σ and a recognizable trace language L over the communication alphabet (Σ, I_{loc}) induced by $\widetilde{\Sigma}$, we can *always* construct from A_L, the minimum DFA for L, a *deterministic* synchronously communicating system $TS_1 \mathbin{|||} \cdots \mathbin{|||} TS_k$ over $\widetilde{\Sigma}$ (that is, the global transition relation of the system is deterministic) such that language of $TS_1 \mathbin{|||} \cdots \mathbin{|||} TS_k$ is L. The construction is too intricate to present here, but we need to slightly qualify Zielonka's theorem to achieve the main result that we are after.

The complication is that Zielonka's theorem holds for synchronously communicating systems with (global) accepting states, whereas we are looking at prefix-closed behaviours of distributed transition systems *without* any accepting states.

Consider, for instance, the language $\{a, b\}$ over the distributed alphabet $\langle \{a\}, \{b\} \rangle$. This language is a recognizable trace language in the framework of Zielonka's theorem. The language is recognized by a transition system with two processes P_a and P_b where the states of P_a are $\{q_0, q_0'\}$, the states of P_b are $\{q_1, q_1'\}$, the initial state is $\langle q_0, q_1 \rangle$, the transition relations are given by $q_0 \rightarrow_a q_0'$ and $q_1 \rightarrow_b q_1'$ and the final states are $\{\langle q_0', q_1 \rangle, \langle q_1', q_0 \rangle\}$. By examining the global states of the system at the end of the words a and b, we can effectively make a *global* choice between two independent actions across the system. This is not possible in our model—if P_a and P_b can locally perform a and b, respectively, we *cannot* rule out the global behaviours ab and ba.

fI-closed languages. Let L be a trace language over (Σ, I). We say that L is *forward-independence closed (fI-closed)* provided $wa \in L$, $wb \in L$ and $(a, b) \in I$ always implies $wab \in L$.

It is not difficult to see that the prefix-closed languages associated with synchronously communicating systems are necessarily fI-closed. We then have the following version of Zielonka's theorem

Theorem 8 *Let $TS = (Q, \rightarrow, q_{\text{in}})$ be a finite-state transition system over Σ and let $\widetilde{\Sigma} = \langle \Sigma_1, \ldots, \Sigma_k \rangle$ be a distribution of Σ. Then, there exists a synchronously communicating system $TS_1 \mathbin{|||} \cdots \mathbin{|||} TS_k$ over $\widetilde{\Sigma}$ such that $L(TS_1 \mathbin{|||} \cdots \mathbin{|||} TS_k) = L(TS)$ if and only if $L(TS)$ is* *an fI-closed recognizable trace language over the independence alphabet* (Σ, I_{loc}) *induced by* $\widetilde{\Sigma}$.

The complexity of Zielonka's construction is analyzed in [14], which has an alternative presentation of the proof of Zielonka's main result. Let k be the width of the distributed alphabet and let n be the number of states in the minimum DFA for $L(TS)$. Then, the number of states of each process in the resulting synchronously communicating system $TS_1 \parallel\!\mid \cdots \parallel\!\mid TS_k$ is $2^{O(2^k n \log n)}$.

6. Synthesis modulo bisimulation

In the course of specifying a system, we may accidentally destroy its inherent distributed 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 distributed 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 [10] for this purpose.

In general, synthesizing a behaviourally equivalent distributed 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* distributed 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 9 *A* bisimulation *between a pair of transition systems* $TS_1 = (Q_1, \rightarrow_1, q^1_{\mathrm{in}})$ *and* $TS_2 = (Q_2, \rightarrow_2, q^2_{\mathrm{in}})$ *is a relation* $R \subseteq (Q_1 \times Q_2)$ *such that:*

- $(q^1_{\mathrm{in}}, q^2_{\mathrm{in}}) \in R$.
- *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 can now be formulated as follows. If $TS = (Q, \rightarrow, q_{\mathrm{in}})$ 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 loosely cooperating system $TS_1 \parallel \cdots \parallel TS_k$ (respectively, a synchronously communicating system $TS_1 \parallel\!| \cdots \parallel\!| TS_k$) over $\widetilde{\Sigma}$ such that $TS_1 \parallel \cdots \parallel TS_k$ (respectively, $TS_1 \parallel\!| \cdots \parallel\!| TS_k$) is bisimilar to TS?

For deterministic transition systems, bisimilarity coincides with language equivalence. We can use this fact to get a simple characterization of transition systems which are bisimilar to deterministic distributed transition systems. We first recall a basic definition.

Bisimulation quotient. Let $TS = (Q, \rightarrow, q_{\text{in}})$ 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 $TS/_{\sim_{TS}} = (\widehat{Q}, \rightsquigarrow, [q_{\text{in}}])$ 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 following results then follow easily.

Theorem 10 *Let TS be a transition system over Σ and let $\widetilde{\Sigma}$ be a distribution of Σ.*

(i) The system TS is bisimilar to a deterministic loosely cooperating system over $\widetilde{\Sigma}$ if and only if the bisimulation quotient $TS/_{\sim_{TS}}$ is deterministic and the language $L(TS)$ is a product language over $\widetilde{\Sigma}$.

(ii) The system TS is bisimilar to a deterministic synchronously communicating system over $\widetilde{\Sigma}$ if and only if the bisimulation quotient $TS/_{\sim_{TS}}$ is deterministic and the language $L(TS)$ is a recognizable trace language over (Σ, I_{loc}).

Proof Sketch: Part (i) follows from Lemma 5, since we can construct a finite-state automaton for each projection L_i of L onto Σ_i and then determinize these automata individually to obtain a deterministic loosely cooperating system.

Part (ii) is a direct consequence of our adaptation of Zielonka's theorem, which guarantees that if $L(TS)$ is an fI-closed recognizable trace language then we can always synthesize a deterministic synchronously communicating system with the same language. $\square$

7. The synthesis problem for concurrent alphabets

We now examine the situation where the distributed nature of the system in specified abstractly in terms of a concurrent alphabet rather than explicitly in terms of a distributed alphabet. The synthesis problem for concurrent alphabets can be phrased as follows:

> Let (Σ, I) be a communication alphabet and let $TS = (Q, \rightarrow, q_{\text{in}})$ be a transition system over Σ. Does there exist a distribution $\widetilde{\Sigma} = \langle \Sigma_1, \ldots, \Sigma_k \rangle$ of Σ with $I_{loc} = I$ and a distributed transition system over $\widetilde{\Sigma}$ that is "equivalent" to TS? (As before, the term equivalence can mean state-space isomorphism, language equivalence or bisimilarity.)

Implementing a concurrent alphabet

Given a concurrent alphabet (Σ, I), there are several ways to construct a distribution $\widetilde{\Sigma} = \langle \Sigma_1, \ldots, \Sigma_k \rangle$ of Σ so that the independence relation I_{loc} induced by *loc* coincides with I.

We begin by building the dependence graph for (Σ, I). Let $D = (\Sigma \times \Sigma) \setminus I$ be the *dependence relation* generated by (Σ, I). Construct a undirected graph $G_D = (\Sigma, E)$ such that $E = \{(a, b) \mid (a, b) \in D\}$.

One way to distribute Σ is to create a process p_e for every edge e in G_D. For each letter a, we then set $loc(a)$ to be the set of processes corresponding to edges incident on the vertex labelled a. In this distribution, each component of $\widetilde{\Sigma}$ consists of precisely two letters. We call this the *finest* distribution corresponding to (Σ, I).

Alternatively, we can create a process p_C for each maximal clique C in G_D. Then, for each letter a and each clique C, $p_C \in loc(a)$ if and only if the vertex labelled a belongs to C. We call this the *coarsest* distribution for (Σ, I).

In both cases, it is easy to see that $I_{loc} = I$. There may also be other distributions between the coarsest and finest distributions that induce I. In general, we just have to ensure that there for every pair $(a, b) \in D$, there is a process P_i with $\{a, b\} \subseteq P_i$—the process P_i "witnesses" the dependency between a and b.

Example 11 If $\Sigma = \{a, b, c, d\}$ and $I = \{(a, b), (b, a)\}$, then all of the following distributions induce I.

- The finest distribution, $(\{a, c\}, \{a, d\}, \{b, c\}, \{b, d\}, \{c, d\})$.
- The coarsest distribution, $(\{a, c, d\}, \{b, c, d\})$.

- An intermediate distribution, $(\{a,c,d\},\{b,c\},\{b,d\})$.

$\square$

A preorder on distributions. 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 $\widetilde{\Sigma}$.

It is not difficult to establish the following result [3].

Lemma 12 *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 distributed transition system $TS_1 \parallel \cdots \parallel TS_k$ over $\widetilde{\Sigma}$, there exists an isomorphic distributed transition system $\widehat{TS}_1 \parallel \cdots \parallel \widehat{TS}_\ell$ over $\widetilde{\Gamma}$.*

It turns out that this result does not hold in the general case, for arbitrary pairs of distributions $\widetilde{\Sigma}$ and $\widetilde{\Gamma}$ that are not related by the preorder. From Lemma 12 we derive the following corollary regarding the problem of synthesis modulo isomorphism for concurrent alphabets.

Corollary 13 *Let (Σ, I) be a concurrent alphabet and $TS = (Q, \rightarrow, q_{\text{in}})$ be a transition system over Σ. Then, the synthesis problem modulo isomorphism for distributed transition systems has a solution for TS if and only if it has a solution with respect to the coarsest distribution that induces I.*

For synthesis modulo language equivalence, a similar result holds for loosely cooperating systems—the synthesis problem has a solution if and only if it has a solution corresponding to the coarsest distribution.

However, for synchronously communicating systems, we can make a much stronger statement. Zielonka's theorem holds for *any* distribution that induces the same independence relation as the original concurrent alphabet. In other words, for synchronously communicating systems we have the following result.

Theorem 14 *Let (Σ, I) be a concurrent alphabet and $TS = (Q, \rightarrow, q_{\text{in}})$ be a transition system over Σ such that $L(TS)$ is an fI-closed recognizable trace language over (Σ, I). Then, for any distribution $\widetilde{\Sigma} = \langle \Sigma_1, \ldots, \Sigma_k \rangle$ such that $I_{loc} = I$ we can construct a (deterministic) synchronously communicating system $TS_1 \parallel\!\mid \cdots \parallel\!\mid TS_k$ over $\widetilde{\Sigma}$ such that $L(TS_1 \parallel\!\mid \cdots \parallel\!\mid TS_k) = L(TS)$.*

8. Discussion

The synthesis problem modulo isomorphism has been studied in a number of contexts, notably in the area of Petri nets. As we remarked in the Introduction, the theory of regions [5] was first developed to solve this problem in the context of nets. A synthesis result for elementary net systems was first proposed in [15]. This was extended to place-transition nets with a step-based semantics in [12]. There have been a number of related results in the area—see, for instance, the surveys [2, 17].

The problem modulo language equivalence has recently been lifted from the setting of synchronous communication to the setting of message-passing. Over the past few years, the graphical specification formalism known as Message Sequence Charts (MSCs) has become popular for specifying systems with asynchronous communication. In general, an MSC-based specification may not admit a finite-state implementation (where finite-state means that the set of reachable configurations is finite—that is, not only is each process locally finite-state, but there is also a uniform bound on the sizes of all the message buffers in the system). The class of MSC specifications that do admit finite-state implementations constitute the so-called regular MSC languages, which have been characterized in [6]. In [13], the synthesis problem modulo language equivalence is solved for regular MSC languages, where the distributed implementation is in terms of message-passing automata.

As we have seen here, very little is known about the synthesis problem modulo bisimulation. In general, one needs to expand on the input transition system to derive an implementation that is bisimilar to the global specification and yet has the structure of a distributed transition system. The chief difficulty is that it is difficult to quantify the collapse in the state space that occurs when one takes the bisimulation quotient of a distributed transition system. To put it another way, the difficulty lies in bounding the size of the synthesized system in terms of the size of the input system. In fact, the state of our knowledge is so poor in this area that even the following seemingly trivial problem is still open.

> Let $TS = (Q, \rightarrow, q_{\rm in})$ be a transition system over Σ and let $\widetilde{\Sigma} = \langle \Sigma_1, \ldots, \Sigma_k \rangle$ be a distribution of Σ. Suppose that TS is finite-state and is bisimilar to a (possibly infinite-state) loosely cooperating system $TS_1 \parallel \cdots \parallel TS_k$ over $\widetilde{\Sigma}$. Then, is it necessarily the case that TS is also bisimilar to a *finite-state* loosely cooperating system $TS'_1 \parallel \cdots \parallel TS'_k$ over $\widetilde{\Sigma}$?

References

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

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

[3] I. Castellani, M. Mukund and P.S. Thiagarajan: Synthesizing distributed transition systems from global specifications, *Proc. FSTTCS 19*, *LNCS* **1739** (1999) 219–231.

[4] W. Ebinger, A. Muscholl: Logical definability on infinite traces, *Theor. Comput. Sci.*, **154** (1996) 67–84.

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

[6] J.G. Henriksen, M. Mukund, K. Narayan Kumar and P.S. Thiagarajan: Regular Collections of Message Sequence Charts, *Proc. MFCS 2000*, *LNCS* **1893** (2000), 405–414.

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

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

[9] 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.

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

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

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

[13] M. Mukund, K. Narayan Kumar, M. Sohoni: Synthesizing distributed finite-state systems from MSCs, *Proc. CONCUR 2000*, *LNCS* **1877** (2000) 521–535.

[14] M. Mukund, M. Sohoni: Gossiping, asynchronous automata and Zielonka's theorem, *Report TCS-94-2*, Chennai Mathematical Institute (1994). Available via http://www.cmi.ac.in/techreps

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

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

[17] 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.

[18] W. Zielonka: Notes on finite asynchronous automata, *R.A.I.R.O.—Inform. Théor. Appl.*, **21** (1987) 99–135.