Two-phase distributed observation problems[*]

Stavros Tripakis
Verimag Laboratory
Centre Equation, 2, avenue de Vignate, 38610 Gières, France
Stavros.Tripakis@imag.fr

Abstract

We introduce and study problems of distributed observation with bounded or unbounded memory. We are given a system modeled as a finite-word language L over some finite alphabet Σ and subalphabets $\Sigma_1, ..., \Sigma_n$ of Σ modeling n distinct observation points. We want to build (when there exist) n observers which collect projections of a behavior in L onto $\Sigma_1, ..., \Sigma_n$, then send them to a central decision point. The latter must determine whether the original behavior was in a given $K \subseteq L$. In the unbounded-memory case, observers record the entire sequence they observe. In the bounded-memory case, they are required to be finite-state automata.

We show that, when L is trace-closed with respect to the usual dependence relation induced by $\Sigma_1, ..., \Sigma_n$, unbounded-memory observability is equivalent to K being centrally observable and trace-closed, thus decidable. When L is not trace-closed, the problem is undecidable, even if K and L are regular. We also show that bounded-memory observability is equivalent to unbounded-memory observability (thus decidable) when L is trace-closed and Σ_i are pairwise disjoint. Otherwise, the problem remains open. In the decidable cases, observers and decision function can be automatically synthesized.

1 Introduction

In this paper we study problems of distributed observation. Such problems arise naturally in contexts of distributed control (e.g., see [9, 23, 24, 17, 16, 18, 26, 19, 2, 20, 14]). There, a number of agents control a single plant, each observing (and acting upon) only part of the plant. Distributed control always "hides" a distributed observation problem, since the agents must infer, based on a set of partial observations, facts about the original behavior of the plant [26]. This remains true independently of whether the agents are allowed to communicate or not [27].

Distributed observation problems are also interesting for their own sake. For instance, when monitoring a large, distributed system such as a network, a vehicle controller consisting of many components (ECUs), a manufacturing plant, etc., one usually relies on local monitors which collect information at different parts of the system. This information can then be gathered and analyzed off-line at a central point. Even for finite sets of observations, such problems are inherently difficult from a complexity point of view [28]. Notice that it is often impractical to endow the local monitors with advanced capabilities such as communication and clock synchronization, which would change the distributed nature of the problem in a significant way.

The general architecture of the problems we study in this paper is shown in Figure 1. There are n observers which are attached to different (distributed) points of the system under observation. The observation takes place in two phases. The first phase, the *collection phase*, is *on-line*: the system is left to execute for a finite amount of time and, meanwhile, each observer collects its own local observation. The second phase, the *decision phase*, is *off-line*: every observer sends its observation to a central decision point, which makes a decision. The decision may involve, for instance, detecting faults in the behavior of the system, measuring its performance, and so on.

In particular, we consider a very simple modeling setting. The system under observation (or *plant*) is modeled as a regular language L over some finite alphabet Σ. A letter in Σ can be seen as an event generated by the plant and a finite word in L can be seen as a behavior of the plant. Observer i can only observe a subset of events $\Sigma_i \subseteq \Sigma$. Thus, the observation that observer i collects from a behavior $\rho \in L$ is the *projection* of ρ onto Σ_i. Notice that Σ_i are not necessarily disjoint. A regular language $K \subseteq L$ models a set of *distinguished* behaviors of the plant. For example, behaviors in K may be those satisfying a given requirement while those in $L - K$ do not. The objective of the decision point is to determine whether the plant produced a behavior

[*]Work partially supported by CNRS STIC project "CORTOS" and IST Network of Excellence "ARTIST2".

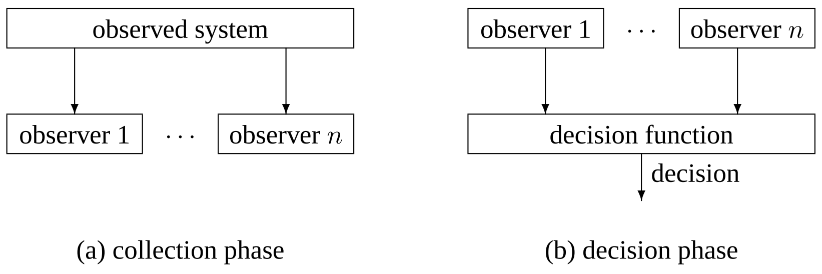

Figure 1. Distributed observation in two phases.

in K or in $L - K$.

Obviously, it is not always possible to make a correct decision, based only on the distributed observations. For example, if $\Sigma = \{a, b\}$, $\Sigma_1 = \{a\}$, $\Sigma_2 = \{b\}$, $L = \{ab, ba\}$ and $K = \{ab\}$, then it is impossible, based on the observations (a, b), to determine whether ab or ba happened. Thus, our first concern is to *check observability*. Our second concern is to *synthesize* the observers and the decision function automatically.

In this paper, we consider two versions of the problem, depending on the memory requirements of the observers. In the *unbounded memory* version, the observers are assumed to record the entire observed sequence (notice that, even in this case, observability is not guaranteed – c.f. example above). In the *finite memory* version, the observers are required to be finite state automata.

The main results we obtain are as follows. We show that, when L is trace-closed, unbounded-memory distributed observability is equivalent to K being trace-closed and centrally observable (i.e., observable by a single observer able to see all observable events). Checking observability in this case is decidable for regular languages, since central observability reduces to a simple emptiness question and trace-closure is also decidable. We show that checking observability in the general case (i.e., when L is not trace-closed) is undecidable, by a reduction of the intersection problem for rational relations. We also show that bounded-memory distributed observability is equivalent to unbounded-memory distributed observability (thus, also equivalent to trace-closure of K, thus, decidable) when L is trace-closed and the subalphabets Σ_i are pairwise disjoint. In all decidable cases, observers and decision functions can be automatically synthesized.

The topic of this paper is distributed observation. Although we use results from trace theory as well as from the theory of rational relations, our objective is not to make contributions in these domains, but to study the distributed observation. Thus, the contributions of this paper are, first, the definition of two off-line distributed observation problems, second, (un)decidability results for these problems, and third, links of such problems to trace and rational relation theory. The third point is, to the best of our knowledge, rarely found in works on decentralized controller synthesis. Some of these links may prove helpful in answering the remaining open questions.

Related work: The unbounded-memory distributed observation problem was introduced in [26], where it was shown to be undecidable using a direct reduction from Post's Correspondence Problem. In this paper, we provide a different proof, using a reduction from the intersection problem of rational relations, known to be undecidable [6]. The new proof is interesting not only as an alternative proof but also because it relates the problem to the theory of rational relations.

The synthesis of bounded-memory observers (and the decision function) is related to the synthesis problems considered in [4, 21, 8, 22, 1, 25]. Generally put, the problem is, given a specification S, synthesize a distributed system I which satisfies S. In [21, 8] S is a finite labelled transition system (LTS), I is a product of LTSs synchronizing on common labels and I is required to be isomorphic or bisimilar to S. LTSs do not have accepting states, whereas in our observation setting, the decision function plays a role of "global" acceptance.

In [25], S is a regular language, I is a *(safe) asynchronous automaton* [29] and the language of I must be contained in S, plus some conditions to avoid trivial solutions. In [4, 3] I is a Petri net. Both Petri nets and (safe) asynchronous automata are more powerful than our distributed observers in terms of communication between the concurrent agents. Indeed, in our model, the observation of a letter leads an observer to a unique next state *independently* of the states of the other observers (thus, the observers do not communicate at all during the collection phase). Whereas in Petri nets and (safe) asynchronous automata a transition generally depends on the global state of the system. Another difference is that [8, 25] allow nondeterminism, whereas our observers are required to be deterministic.

In [1] S is a message-sequence chart (MSC) graph, I

is a set of automata asynchronously communicating over FIFO buffers and it is required that I exhibits exactly the behaviors of S. MSC specifications are also considered in [22]. Also slightly related to this paper are works on distributed system estimation or fault diagnosis based on Petri nets [15, 5]. The problem in [15] is to estimate the current marking of a Petri net based on the local observations. [5] show how to construct, given a Petri net, an unfolding which "explains" in some sense the local observations of the set of distributed sensors.

[13, 29] have independently considered so-called *mixed* and *weakly-mixed* trace languages. These are languages accepted by products of finite automata synchronizing on common letters (i.e., products of LTSs with accepting states). In mixed languages, the automata have *local* initial and accepting sets of states and the initial/accepting set of the product is the cartesian product of the local initial/accepting sets. In weakly-mixed languages, the initial and accepting sets are arbitrary, i.e., *global*. Weakly-mixed languages correspond to a special case of bounded-memory distributed observation, where $L = \Sigma^*$. In [13] it is shown that every weakly-mixed language is a finite union of mixed languages and that every regular trace language is the homomorphic image of a weakly-mixed language. Partial decidability results on the problem, given a regular trace language, is it mixed (weakly-mixed), are provided in [7].

Finally, as mentioned in the introduction, distributed observation is related to distributed control. In fact, unbounded-memory distributed observation can be reduced to distributed control without communication [27]. This may seem surprising, since observation is done in two phases and the second phase is centralized, whereas the distributed control problem used in [27] is on-line, without *a priori* communication between the controllers. The paradox is resolved by the fact that there exist plants which allow the controllers to communicate *indirectly*, through enabling and disabling plant transitions. Thus, two-phase observation can be simulated by on-line control.

2 Preliminaries

In this section we give a brief overview of standard concepts, to be used in the rest of the paper. For more details, the reader is referred, for instance, to [6, 11].

Recognizable and rational subsets of monoids: A monoid is a set X equipped with a binary product operator $\cdot$ which is associative (i.e., for all $x_1, x_2, x_3 \in X$, $(x_1 \cdot x_2) \cdot x_3 = x_1 \cdot (x_2 \cdot x_3)$) and a neutral element 1 (i.e., for all $x \in X$, $x \cdot 1 = 1 \cdot x = x$). The product naturally extends to subsets of X: for $X_1, X_2 \subseteq X$, $X_1 \cdot X_2 = \{x_1 \cdot x_2 \mid x_1 \in X_1, x_2 \in X_2\}$. A subset Y of X is called a generator of X iff $X = Y^*$, where $Y^* = \bigcup_{i \geq 0} Y^i$ and $Y^0 = \{1\}$, $Y^{i+1} = Y \cdot Y^i$.

A machine over X is a tuple (S, s_0, t), where S is a set of states, $s_0 \in S$ is the initial state, and $t : S \times X \to S$ is the (deterministic) transition function, satisfying the following condition: $\forall x, x' \in X, \forall s \in S, t(s, x \cdot x') = t(t(s, x), x')$. Given $x \in X$, $t(x)$ is a shorthand notation for $t(s_0, x)$. An automaton over X is a tuple $A = (S, s_0, t, F)$, where (S, s_0, t) is a machine over X and $F \subseteq S$ is the set of final states. The subset of X recognized by A, denoted $L(A)$, is the set $\{x \in X \mid t(x) \in F\}$. The class of recognizable subsets of X, denoted $Rec(X)$, is defined as the class of all subsets of X which are recognized by finite-state automata over X.

The class of rational subsets of X, denoted $Rat(X)$, is the class of all subsets of X which can be defined by rational expressions on X. The latter are $\emptyset$ (denoting the empty set), x, for $x \in X$ (denoting the set $\{x\}$), $e \cdot e'$, $e + e'$, e^*, where e, e' are rational expressions, denoting, respectively, $Y \cdot Y'$, $Y \cup Y'$ and Y^*, where Y, Y' are the sets denoted by e, e'.

For any finitely generated monoid X, $Rec(X) \subseteq Rat(X)$. The converse inclusion does not generally hold. It holds in the monoid Σ^*, where Σ is a finite set (or *alphabet*). Σ^* is the set of all finite words over Σ, concatenation of words plays the role of the product operator and the empty word, denoted ϵ, is the neutral element. Kleene's theorem states that $Rec(\Sigma^*) = Rat(\Sigma^*)$.

In the monoid $\Sigma_1^* \times \Sigma_2^*$, where Σ_1, Σ_2 are finite disjoint alphabets, $Rec(\Sigma_1^* \times \Sigma_2^*)$ is a strict subset of $Rat(\Sigma_1^* \times \Sigma_2^*)$ [6]. $\Sigma_1^* \times \Sigma_2^*$ is the set of all pairs of finite words over Σ_1 and Σ_2. The neutral element is (ϵ, ϵ) and the product is defined as piecewise concatenation: $(\rho_1, \rho_2) \cdot (\sigma_1, \sigma_2) = (\rho_1\sigma_1, \rho_2\sigma_2)$. The rational subsets of $\Sigma_1^* \times \Sigma_2^*$ are also called *rational relations* over Σ_1 and Σ_2.

A number of problems on rational sets are undecidable. For instance, it is undecidable to check, given $R \in Rat(\Sigma_1^* \times \Sigma_2^*)$, whether $R \in Rec(\Sigma_1^* \times \Sigma_2^*)$. It is also undecidable to check, given $R_1, R_2 \in Rat(\Sigma_1^* \times \Sigma_2^*)$, whether $R_1 \cap R_2 = \emptyset$.

Projections of words: Given $\rho \in \Sigma^*$ and $\Sigma_1 \subseteq \Sigma$, the projection of ρ onto Σ_1, denoted $P_{\Sigma_1}(\rho)$, is the word $\pi \in \Sigma_1^*$ obtained from ρ by erasing all letters not in Σ_1. For example, if $\Sigma = \{a, b, c\}$ and $\Sigma_1 = \{a, c\}$, then $P_{\Sigma_1}(abbcbab) = aca$. When no confusion arises, we abbreviate $P_{\Sigma_1}(\rho)$ by $P_1(\rho)$. The following properties of projections can be easily derived from the definitions.

Lemma 1 *Let $\Sigma, \Sigma_1, \Sigma_2$ be finite alphabets such that $\Sigma_2 \subseteq \Sigma_1 \subseteq \Sigma$. Then, $\forall \rho, \rho' \in \Sigma^*$. $P_{\Sigma_1}(\rho) = P_{\Sigma_1}(\rho') \Rightarrow P_{\Sigma_2}(\rho) = P_{\Sigma_2}(\rho')$.*

Projection extends naturally to sets of words, thus, $P_{\Sigma_1}(L) = \{P_{\Sigma_1}(\rho) \mid \rho \in L\}$. Given $K \subseteq \Sigma_1^*$, the inverse projection is defined as $P_{\Sigma_1}^{-1}(K) = \{\rho \in \Sigma^* \mid P_{\Sigma_1}(\rho) \in K\}$ (notice that $\Sigma \supseteq \Sigma_1$ is implicit in this case).

Trace equivalence: A concurrent alphabet is a pair (Σ, D) where Σ is a finite alphabet and $D \subseteq \Sigma \times \Sigma$ is a reflexive and symmetric relation, called the dependence relation. D induces the irreflexive and symmetric relation $I = (\Sigma \times \Sigma) - D$, called the independence relation. D also induces the equivalence $\equiv$, called the trace equivalence, defined as the reflexive and transitive closure of the relation $\equiv^1 \subseteq \Sigma^* \times \Sigma^*$, where $\pi \equiv^1 \rho$ iff there exist words $\sigma_1, \sigma_2 \in \Sigma^*$ and $(a, b) \in I$, such that $\pi = \sigma_1 ab \sigma_2$ and $\rho = \sigma_1 ba \sigma_2$. That is, $\pi \equiv \rho$ iff π can be obtained from ρ by repeatedly swapping adjacent independent letters. $L \subseteq \Sigma^*$ is said to be a trace language over (Σ, D) if it is closed under $\equiv$, that is, $\forall \rho, \rho' \in \Sigma^*$, $\rho \equiv \rho' \Rightarrow (\rho \in L \Leftrightarrow \rho' \in L)$. Alternatively, a trace language L can be viewed as a subset of the *quotient* monoid of Σ^* with respect to $\equiv$, denoted $M(\Sigma^*, D)$. Thus, an element of $M(\Sigma^*, D)$ is an equivalence class of words related by $\equiv$. For $\rho \in \Sigma^*$, $[\rho] = \{\rho' \in \Sigma^* \mid \rho \equiv \rho'\}$ is the equivalence class of ρ, $[\rho] \in M(\Sigma^*, D)$.

Consider a finite alphabet Σ and subalphabets $\Sigma_1, ..., \Sigma_n \subseteq \Sigma$. Let $\Sigma_o = \bigcup_{i=1,...,n} \Sigma_i$ and $\Sigma_u = \Sigma - \Sigma_o$. We define the following dependence relation:

$$D_{\Sigma_1,...,\Sigma_n} = (\bigcup_{i=1,...,n} \Sigma_i \times \Sigma_i) \cup \{(u, u) \mid u \in \Sigma_u\}. \quad (1)$$

In other words, two distinct letters are dependent iff there is a subalphabet Σ_i containing both. Notice that the Σ_i's are not necessarily disjoint neither do they cover Σ.

The following lemma states an important property relating projections and trace equivalence. Similar results can be found in [10, 13].

Lemma 2 *Let Σ be a finite alphabet and $\Sigma_i \subseteq \Sigma$, for $i = 1, ..., n$. Let $\equiv$ be the trace equivalence induced by $D_{\Sigma_1,...,\Sigma_n}$ defined as above. Then, for any $\rho, \rho' \in \Sigma^*$, $\rho \equiv \rho'$ iff $\forall i \in \{1, ..., n\} \,.\, P_i(\rho) = P_i(\rho')$ and $P_{\Sigma_u}(\rho) \equiv P_{\Sigma_u}(\rho')$.*

3 Distributed Observation Problems

Unless otherwise stated, in the rest of the paper, we assume that Σ is a finite alphabet and consider subalphabets $\Sigma_i \subseteq \Sigma$, for $i = 1, ..., n$. $\Sigma_o = \bigcup_{i=1,...,n} \Sigma_i$ is the set of all *observable* events. Notice that Σ_i need not be disjoint and Σ_o need not be equal to Σ.

Definition 1 (Distr. observation with unbounded memory) *Let $K \subseteq L \subseteq \Sigma^*$. We say that K* is observable *with respect to L and $(\Sigma_1, ..., \Sigma_n)$ iff there exists a function $f : \Sigma_1^* \times \cdots \times \Sigma_n^* \rightarrow \{0, 1\}$, such that*

$$\forall \rho \in L \,.\, \Big(\rho \in K \Leftrightarrow f\big(P_{\Sigma_1}(\rho), \cdots, P_{\Sigma_n}(\rho)\big) = 1\Big).$$

According to Definition 1, K is observable iff function f can decide whether a behavior ρ of L was in K based solely on the n observations, that is, the projections of ρ onto $\Sigma_1, ..., \Sigma_n$. [1] The observers here are trivial. They merely record the entire sequence they observe. Since this sequence is a-priori unbounded in length, the observers need unbounded memory. It is interesting to consider also the case where the observers are required to have finite memory.

Definition 2 (Distr. observation with bounded memory) *Let $K \subseteq L \subseteq \Sigma^*$. We say that K* is finitely observable *with respect to L and $(\Sigma_1, ..., \Sigma_n)$ iff there exist finite-state machines A_i over Σ_i, $A_i = (S_i, s_i^0, t_i)$, $i = 1, ..., n$, and a function $g : S_1 \times \cdots \times S_n \rightarrow \{0, 1\}$, such that*

$$\forall \rho \in L \,.\, \Big(\rho \in K \Leftrightarrow g\big(t_1(P_{\Sigma_1}(\rho)), \cdots, t_n(P_{\Sigma_n}(\rho))\big) = 1\Big).$$

According to Definition 2, in the finite-memory case, observer i does not record the entire observation $P_{\Sigma_i}(\rho)$, but only a state $s_i = t_i(P_{\Sigma_i}(\rho))$ among a finite set of states S_i. The decision function has access to the state of each observer in order to make its decision.

It should be noted that K is finitely observable in a trivial way when $K = \emptyset$ or $K = L$. Also, if an observer cannot observe anything, i.e., $\Sigma_i = \emptyset$, then this observer can be ignored, since it does not bring any information. On the other hand, if there exists an observer which can observe all events that all other observers can observe, i.e., $\exists i, \forall j, \Sigma_j \subseteq \Sigma_i$, then the problem degenerates to a centralized observation problem, since all observers except the i-th are redundant.

Obviously, finite observability implies observability. The converse is not generally true. This is illustrated below, in Example 1.

The following lemma states some closure properties of observability with respect to union, intersection and relative complement.

Lemma 3 *If K_1 and K_2 are observable (resp. finitely observable) w.r.t. L and $(\Sigma_1, ..., \Sigma_n)$ then $K_1 \cup K_2$, $K_1 \cap K_2$ and $L - K_1$ are also observable (resp. finitely observable) w.r.t. L and $(\Sigma_1, ..., \Sigma_n)$.*

[1] More generally, we could associate a *mask* function $M_i : \Sigma \rightarrow O_i \cup \{\tau\}$ to each observer [9], with the meaning that event $a \in \Sigma$ is either totally unobservable to observer i (when $M_i(a) = \tau$) or is perceived as $o \in O_i$ (when $M_i(a) = o$). This would not affect the results of this paper in an essential way.

4 Necessary and sufficient conditions

We begin by a necessary and sufficient condition for unbounded-memory observability. Intuitively, the condition states that K is observable w.r.t. L iff there exist no two behaviors in L which yield the same observations, yet one is in K and the other is not.

Lemma 4 *K is observable with respect to L and $(\Sigma_1, ..., \Sigma_n)$ iff*

$$\forall \rho \in K \ . \ \forall \rho' \in L{-}K \ . \ \exists i \in \{1, ..., n\} \ . \ P_{\Sigma_i}(\rho) \neq P_{\Sigma_i}(\rho'). \tag{2}$$

Although the above condition characterizes observability, it cannot be verified algorithmically. Indeed, as we shall see in Section 6, checking observability is undecidable for $n \geq 2$. Thus, we provide some conditions which are decidable.

Lemma 5 *If K is observable w.r.t. L and $(\Sigma_1, ..., \Sigma_n)$ then*

$$P_{\Sigma_o}(K) \cap P_{\Sigma_o}(L - K) = \emptyset. \tag{3}$$

Condition (3) essentially states that K is observable in a centralized manner. This is clearly a necessary, but not sufficient, condition for distributed observability. Notice that the above condition is necessary also for finite observability, since the latter implies observability.

Lemma 6 *If*

$$\exists i \in \{1, ..., n\} \ . \ P_{\Sigma_i}(K) \cap P_{\Sigma_i}(L - K) = \emptyset. \tag{4}$$

then K is finitely observable w.r.t. L and $(\Sigma_1, ..., \Sigma_n)$.

Condition (4) essentially states that all observers except the i-th one are redundant. Thus, this is a degenerate case where the problem is reduced to a centralized observation problem. It is well-known that in the centralized case observability and finite observability are equivalent: the finite observer is simply obtained by a determinization procedure (subset construction).

Condition (4) is not necessary, as we now show with an example. Let $\Sigma = \{a, b, c\}$, $\Sigma_1 = \{a, b\}$, $\Sigma_2 = \{b, c\}$. Let $\rho = abc$, $\rho_1 = ab$, $\rho_2 = bc$. Let $L = \{\rho, \rho_1, \rho_2\}$, $K = \{\rho_1, \rho_2\}$. We have $P_{\Sigma_1}(\rho) = P_{\Sigma_1}(\rho_1) = ab$, $P_{\Sigma_2}(\rho) = P_{\Sigma_2}(\rho_2) = bc$ and $P_{\Sigma_2}(\rho_1) = P_{\Sigma_1}(\rho_2) = b$. Then, $P_{\Sigma_1}(K) \cap P_{\Sigma_1}(L - K) = \{ab, b\} \cap \{ab\} = \{ab\}$ and $P_{\Sigma_2}(K) \cap P_{\Sigma_2}(L - K) = \{b, bc\} \cap \{bc\} = \{bc\}$. Thus, Condition (4) does not hold. However, Condition (2) holds. Indeed, observer 2 can distinguish between ρ and ρ_1, because $P_{\Sigma_2}(\rho) = bc \neq b = P_{\Sigma_2}(\rho_1)$. Also, observer 1 can distinguish between ρ and ρ_2, because $P_{\Sigma_1}(\rho) = ab \neq b = P_{\Sigma_1}(\rho_2)$.

When L and K are regular languages, conditions (3) and (4) can be checked algorithmically, since regular languages are closed under intersection, complementation and projection, and checking regular language emptiness is decidable. For $n = 1$, Conditions (2-4) are equivalent, which implies that checking centralized observability is decidable.

The following lemma provides another necessary condition for observability.

Lemma 7 *If K is observable (resp. finitely observable) w.r.t. L and $(\Sigma_1, ..., \Sigma_n)$ then $P_{\Sigma_o}(K)$ is observable (resp. finitely observable) w.r.t. $P_{\Sigma_o}(L)$ and $(\Sigma_1, ..., \Sigma_n)$.*

Proof Suppose K is observable and let f be the decision function. We claim that f is a valid decision function for observability of $P_{\Sigma_o}(K)$ w.r.t. $P_{\Sigma_o}(L)$ and $(\Sigma_1, ..., \Sigma_n)$. Let $\pi \in P_{\Sigma_o}(L)$. There exists $\rho \in L$ such that $\pi = P_{\Sigma_o}(\rho)$. For all $i = 1, ..., n$, since $\Sigma_i \subseteq \Sigma_o$, $P_{\Sigma_i}(\rho) = P_{\Sigma_i}(\pi) = \pi_i$ (Lemma 1). Suppose $f(\pi_1, ..., \pi_n) = 1$. Then, $\rho \in K$, thus, $\pi \in P_{\Sigma_o}(K)$. Suppose $f(\pi_1, ..., \pi_n) = 0$. Then, $\rho \in L - K$. We claim that $\pi \notin P_{\Sigma_o}(K)$. Otherwise, $\pi \in P_{\Sigma_o}(K) \cap P_{\Sigma_o}(L - K)$, thus, by Lemma 5, K is not observable, which contradicts the hypothesis. The proof is similar in the finite observability case. ∎

5 A special case: L trace language over $(\Sigma, D_{\Sigma_1, ..., \Sigma_n})$

In this section we examine a special case of the observation problems, namely, when L is a trace language over $(\Sigma, D_{\Sigma_1, ..., \Sigma_n})$. This is an important case in practice. For instance, when L is obtained from the asynchronous product of a set of automata over $\Sigma_1, ..., \Sigma_n$ then it is guaranteed to be trace-closed. This is also the case when $L = \Sigma^*$, that is, when we have no model of the system under observation (although we do have a model of the requirements, K). When L is trace-closed, we call the observation problems *simple*.

The main results of this section are two. First, we show that unbounded-memory simple observability is "almost" equivalent to trace-closure and decidable for regular languages. Second, we show finite-memory simple observability is equivalent to unbounded-memory simple observability, in the case where the subalphabets Σ_i are disjoint. In a sense this is the best we can do, since there is an example of a system which is observable but not finitely observable, where Σ_i are not disjoint (Example 1).

Theorem 1 *Let Σ be a finite alphabet and $\Sigma_i \subseteq \Sigma$, for $i = 1, ..., n$. Let $K \subseteq L \subseteq \Sigma^*$. Assume L is a trace language over $(\Sigma, D_{\Sigma_1, ..., \Sigma_n})$. Then, K is observable w.r.t. L and $(\Sigma_1, ..., \Sigma_n)$ iff K is a trace language over $(\Sigma, D_{\Sigma_1, ..., \Sigma_n})$ and Condition (3) holds.*

Proof Only if: Suppose K is observable. Let $\rho \in K$ and consider $\rho' \in \Sigma^*$ such that $\rho \equiv \rho'$, where $\equiv$ is the trace equivalence induced by $D_{\Sigma_1,...,\Sigma_n}$. Since $\rho \in L$ and L is trace-closed, $\rho' \in L$. By the "only if" part of Lemma 2, $\forall i \in \{1,...,n\}$, $P_i(\rho) = P_i(\rho')$. By Lemma 4, ρ' must be in K, otherwise K would not be observable w.r.t. L and $(\Sigma_1,...,\Sigma_n)$. Thus, K is a trace language. By Lemma 5, Condition (3) holds.

If: Suppose K is not observable. By Lemma 4, there exist $\rho \in K$, $\rho' \in L - K$, such that $\forall i \in \{1,...,n\}$, $P_i(\rho) = P_i(\rho')$. Without loss of generality, we can assume that $\rho = \sigma \cdot u$ and $\rho' = \sigma' \cdot u'$, where $\sigma = P_{\Sigma_o}(\rho)$, $\sigma' = P_{\Sigma_o}(\rho')$ and $u, u' \in (\Sigma - \Sigma_o)^*$. This is because we can "push" unobservable letters to the end of ρ and ρ', obtaining trace-equivalent words (recall that unobservable letters are independent from observable letters). Transforming ρ into a new, trace-equivalent word does not affect the assumption $\rho \in K$ (because K is trace-closed) neither the assumption $\forall i \in \{1,...,n\}$, $P_i(\rho) = P_i(\rho')$ (by the "only if" part of Lemma 2). Similarly for ρ'. Thus, we still have ρ and ρ' contradicting the observability of K. We will next show that $\sigma \equiv \sigma'$. This implies $\sigma \cdot u \equiv \sigma' \cdot u$, thus, by trace-closure of K, $\sigma' \cdot u \in K$. Since $\sigma' = P_{\Sigma_o}(\sigma' \cdot u)$ and $\sigma' = P_{\Sigma_o}(\rho')$, we have $P_{\Sigma_o}(K) \cap P_{\Sigma_o}(L - K) \neq \emptyset$, which contradicts our hypothesis. Thus, K must be observable.

We complete the proof by showing $\sigma \equiv \sigma'$. To see this, observe that the "if" part of Lemma 2 applies to σ and σ' by taking Σ to be Σ_o. ■

We can therefore conclude that simple observability for regular languages is decidable:

Corollary 1 *When K and L are regular languages and L is trace-closed, checking observability is decidable.*

Proof By Lemma 5 and Theorem 1, Condition (3) and trace-closure of K is a necessary and sufficient condition for simple observability of K. Checking trace-closure of regular languages is decidable. So is checking Condition (3), since regular languages are closed under intersection, complementation and projection, and checking emptiness for regular languages is decidable. ■

Obviously, simple finite observability implies simple observability. The converse is not always true, as the following example shows. (Similar examples can be found in [21, 8].)

Example 1 (Observability $\not\Rightarrow$ finite observability) *Let $\Sigma_1 = \{a_1, b\}$, $\Sigma_2 = \{a_2, b\}$ and $\Sigma = \{a_1, a_2, b\}$. Let $K = ((a_1 + a_2)(a_1 + a_2 + b))^*$. It can be checked that K is observable w.r.t. Σ^* and (Σ_1, Σ_2). Intuitively, it suffices for the decision function to check that between every two consecutive b's (or from the beginning of the computation until the first b observed) the sum of a_1's and a_2's is odd. However, K is not finitely observable. Intuitively, this is because the number of b's can be arbitrary, and observer i needs to record the parity of the number of a_i's it observes between* every *two consecutive b's.* ■

Observe that, in the example above, $\Sigma_1 \cap \Sigma_2 \neq \emptyset$. It turns out that when Σ_i are pairwise disjoint, and K is regular, finite simple observability is equivalent to simple observability, as we show below. Disjointness of Σ_i means the observers do not share any observable event. This is not an unusual situation in practice: often in a distributed system the observers are placed separately so that one observer has no access to information transferred through the interfaces of the others.

Theorem 2 *Let Σ be a finite alphabet and $\Sigma_i \subseteq \Sigma$, for $i = 1,...,n$, so that $\forall 1 \leq i < j \leq n, \Sigma_i \cap \Sigma_j = \emptyset$ (i.e., Σ_i are pairwise disjoint). Let $K \subseteq L \subseteq \Sigma^*$. Assume that K is regular and that L is a trace language over $(\Sigma, D_{\Sigma_1,...,\Sigma_n})$. Then, K is finitely observable w.r.t. L and $(\Sigma_1,...,\Sigma_n)$ iff K is observable w.r.t. L and $(\Sigma_1,...,\Sigma_n)$.*

Proof Finite observability clearly implies observability, so we only need to prove the converse. By Part 1 of Theorem 1, K is a trace language over $(\Sigma, D_{\Sigma_1,...,\Sigma_n})$ and Condition (3) holds. Let us view K as a subset of the quotient monoid $M(\Sigma^*, D_{\Sigma_1,...,\Sigma_n})$. Let $\Sigma_u = \{u_1,...,u_m\}$. By the fact that Σ_i are pairwise disjoint and disjoint with Σ_u, it can be easily shown that $M(\Sigma^*, D_{\Sigma_1,...,\Sigma_n})$ is isomorphic to $\Sigma_1^* \times \cdots \Sigma_n^* \times \{u_1\}^* \times \cdots \times \{u_m\}^*$: it suffices to map $[\rho]$ to $(P_{\Sigma_1}(\rho),...,P_{\Sigma_n}(\rho), P_{\{u_1\}}(\rho),...,P_{\{u_m\}}(\rho)$ and vice-versa. By Mezei's theorem [6], there exists $q \geq 0$ and regular languages $K_{i,j} \subseteq \Sigma_i^*$ and $U_{l,j} \subseteq \{u_l\}^*$, for $i = 1,...,n$, $l = 1,...,m$, $j = 1,...,q$, such that

$$K = \bigcup_{j=1}^{q} K_{1,j} \times \cdots \times K_{n,j} \times U_{1,j} \times \cdots \times U_{m,j}.$$

We claim that we can assume, without loss of generality, that $U_{l,j} = \{u_l\}^*$, for any $l = 1,...,m$, $j = 1,...,q$. Indeed, suppose there is some l, j such that $U_{l,j} \neq \{u_l\}^*$, that is, there is some k such that $u_l^k \notin U_{l,j}$. Now, consider $\rho = (\rho_1,...,\rho_n,\sigma_1,...,\sigma_m) \in K_{1,j} \times \cdots \times K_{n,j} \times U_{1,j} \times \cdots \times U_{m,j} = K_j$ (notice that this j is the same as for $U_{l,j}$ fixed above). Also consider $\rho' = (\rho_1,...,\rho_n,\sigma_1,...,u_l^k,...,\sigma_m)$. Since K is observable and for all $i = 1,...,n$, $P_{\Sigma_i}(\rho) = \rho_i = P_{\Sigma_i}(\rho')$, it must be that $\rho' \in K$. This means that by adding u_l^k to $U_{l,j}$ we do not add any extra words in K. Proceeding like this for any word "missing" from $U_{l,j}$, we reach our claim. The result of the claim is that we can ignore all $U_{l,j}$ when we build the observers.

Indeed, let $A_{i,j}$ be the finite-state deterministic automaton recognizing $K_{i,j}$, for $i = 1,...,n$, $j = 1,...,q$. For

$i = 1, ..., n$, we build a finite-state deterministic machine A_i, defined as the *synchronous product* of $A_{i,1}, ..., A_{i,q}$: that is, the set of states of A_i is the cartesian product of the sets of $A_{i,1}, ..., A_{i,q}$, and a move of A_i corresponds to a move of each of $A_{i,1}, ..., A_{i,q}$ by the same letter. The machines $A_1, ..., A_n$ correspond to the n finite-state observers. The decision function g is defined as follows:

$$g((s_{1,1}, ..., s_{1,q}), ..., (s_{n,1}, ..., s_{n,q})) = \begin{cases} 1, & \text{if } \exists j \in \{1, ..., q\} \,.\, \forall i \in \{1, ..., n\} \,.\, s_{i,j} \in F_{i,j} \\ 0, & \text{otherwise} \end{cases}$$

where $F_{i,j}$ is the set of accepting states of $A_{i,j}$. It can be checked that $\rho \in K$ iff $g\big(t_1(P_{\Sigma_1}(\rho)), \cdots, t_n(P_{\Sigma_n}(\rho))\big) = 1$. Thus, K is finitely observable w.r.t. L and $(\Sigma_1, ..., \Sigma_n)$. It is interesting to note that the above construction does not depend on L.

6 The general case

We now consider the general case, i.e., where L is not trace-closed. The main result is that checking unbounded-memory observability is undecidable, even when K and L are regular languages. This was first shown in [26], using a direct reduction of Post's Correspondence Problem. In this paper, we provide an alternative proof using a reduction from the problem of checking whether the intersection of two rational relations is empty, which is known to be undecidable (see, for instance, [6]). In the process, we provide some links between the distributed observation problems and the theory of rational relations.

For this section, let us assume that $\Sigma_1, ..., \Sigma_n$ are pairwise disjoint alphabets and let $\Sigma = \cup_{i=1,...,n}\Sigma_i$. Given $L \subseteq \Sigma^*$, we define

$$P_{\Sigma_1,...,\Sigma_n}(L) = \{(P_{\Sigma_1}(\rho), ..., P_{\Sigma_n}(\rho)) \mid \rho \in L\}.$$

Thus, $P_{\Sigma_1,...,\Sigma_n}(L)$ is a subset of $\Sigma_1^* \times \cdots \times \Sigma_n^*$. If L is regular (i.e., a rational subset of Σ^*) then $P_{\Sigma_1,...,\Sigma_n}(L)$ is a rational subset of $\Sigma_1^* \times \cdots \times \Sigma_n^*$. On the other hand, given $R \in Rat(\Sigma_1^* \times \cdots \times \Sigma_n^*)$, there exists a regular language $L \subseteq \Sigma^*$ such that $R = P_{\Sigma_1,...,\Sigma_n}(L)$ [6].

Lemma 8 *For any $R \in Rat(\Sigma_1^* \times \cdots \times \Sigma_n^*)$ we can build $L \in Rec(\Sigma^*)$ such that $R = P_{\Sigma_1,...,\Sigma_n}(L)$.*

We can now restate Condition (2) in the context of rational relations.

Lemma 9 *K is observable w.r.t. L and $(\Sigma_1, ..., \Sigma_n)$ iff*

$$P_{\Sigma_1,...,\Sigma_n}(K) \cap P_{\Sigma_1,...,\Sigma_n}(L - K) = \emptyset. \quad (5)$$

Theorem 3 *Checking unbounded-memory distributed observability for regular languages is undecidable.*

Proof We reduce the intersection problem of rational relations to checking observability. Let Σ_1, Σ_2 be disjoint alphabets and let $\Sigma = \Sigma_1 \cup \Sigma_2$. Consider $R_1, R_2 \in Rat(\Sigma_1^* \times \Sigma_2^*)$. Checking $R_1 \cap R_2 = \emptyset$ is undecidable [6]. By Lemma 8, we can build regular languages $L_i \subseteq \Sigma^*$ such that $R_i = P_{\Sigma_1,\Sigma_2}(L_i)$, for $i = 1, 2$. If $L_1 \cap L_2 \neq \emptyset$ then $P_{\Sigma_1,\Sigma_2}(L_1) \cap P_{\Sigma_1,\Sigma_2}(L_2) \neq \emptyset$, i.e., $R_1 \cap R_2 \neq \emptyset$. If $L_1 \cap L_2 = \emptyset$ then $(L_1 \cup L_2) - L_1 = L_2$. Thus, $R_1 \cap R_2 = \emptyset$ iff $P_{\Sigma_1,\Sigma_2}(L_1) \cap P_{\Sigma_1,\Sigma_2}(L_2) = \emptyset$ iff $P_{\Sigma_1,\Sigma_2}(L_1) \cap P_{\Sigma_1,\Sigma_2}((L_2 \cup L_1) - L_1) = \emptyset$. By Lemma 9, the last equation holds iff L_1 is observable w.r.t. $L_1 \cup L_2$ and (Σ_1, Σ_2). ■

7 Synthesis

In the cases where checking observability or finite observability is decidable, we can synthesize observers and decision function that solve the problem. Let us outline how this can be done.

First, consider the unbounded-memory case and suppose L is trace-closed (otherwise, checking observability is undecidable). The first step is to ensure Condition (3) and trace-closure of K hold. Then, we just need to synthesize the decision function f, since the observers are trivial (they record the entire sequence they observe). Let $\pi_i \in \Sigma_i^*$, for $i = 1, ..., n$. We define f as follows:

$$f(\pi_1, ..., \pi_n) = \begin{cases} 1, & \text{if } K \cap \bigcap_{i=1,...,n} P_{\Sigma_i}^{-1}(\pi_i) \neq \emptyset \\ 0, & \text{otherwise} \end{cases} \quad (6)$$

That is, f should return 1 iff there is a word in K which produces the observations $(\pi_1, ..., \pi_n)$. Clearly, f is computable. It is also correct. Indeed, let $\rho \in L$ such that $P_i(\rho) = \pi_i$, for $i = 1, ..., n$. If $\rho \in K$ then $f(\pi_1, ..., \pi_n) = 1$ by definition of f. If $\rho \notin K$ then $f(\pi_1, ..., \pi_n)$ must be equal to 0. Otherwise, there exists $\rho' \in K \cap \bigcap_{i=1,...,n} P_{\Sigma_i}^{-1}(\pi_i)$, which implies $P_i(\rho') = \pi_i$, for $i = 1, ..., n$. This contradicts observability of K.

Second, consider the bounded-memory case where L is trace-closed and Σ_i are pairwise disjoint (otherwise, we do not know whether checking finite observability is decidable). Again, the first step is to check trace-closure of K. If K is not trace-closed, then it is not (finitely) observable. Otherwise, by Theorem 2, K is finitely observable. In fact, the proof of the theorem provides an algorithm to construct the observer machines and decision function g necessary for bounded-memory observation. The algorithm is based on the construction of a deterministic asynchronous automaton recognizing K, using existing algorithms (e.g. [29, 12]). Due to the assumption that Σ_i are pairwise disjoint, the asynchronous automaton is a set of non-communicating finite-state machines, as observed in

	Bounded-memory observation	Unbounded-memory observation
L trace	Equivalent to unbounded-memory observation when Σ_i pairwise disjoint. Otherwise ?	Equivalent to Condition (3) and K trace, decidable.
L not trace	?	Undecidable.

Table 1. Summary of results and open questions.

the proof of Theorem 2. The (global) accepting states of the automaton characterize the decision function g.

8 Summary of results and open questions

The main contributions of this paper are two. First, we introduced a set of simple, yet fundamental, problems of distributed observation. Second, we studied decidability of these problems and related them to existing concurrency theory, in particular, trace languages and rational relations. Open questions remain, and they appear quite challenging.

The results of the paper and the open questions are summarized in Table 1.

References

[1] R. Alur, K. Etessami, and M. Yannakakis. Realizability and verification of MSC graphs. In *ICALP'01*, volume 2076 of *LNCS*, 2001.

[2] A. Arnold, A. Vincent, and I. Walukiewicz. Games for synthesis of controllers with partial observation. *Theoretical Computer Science*, 303:7–34, 2003.

[3] E. Badouel, B. Caillaud, and P. Darondeau. Distributing finite automata through petri net synthesis. *Formal Aspects of Computing*, 13(6):447–470, Sept. 2002.

[4] E. Badouel and P. Darondeau. Theory of regions. In *Lectures on Petri Nets I: Basic Models*, volume 1491 of *LNCS*, pages 529–586. Springer-Verlag, 1998.

[5] A. Benveniste, E. Fabre, S. Haar, and C. Jard. Diagnosis of asynchronous discrete event systems, a net unfolding approach. *IEEE Trans. Aut. Control*, 48(5), 2003.

[6] J. Berstel. *Transductions and context-free languages*. Teubner, 1979.

[7] J. Berstel, L. Boasson, and M. Latteux. Mixed languages. Theoret. Comput. Sci. (to appear).

[8] I. Castellani, M. Mukund, and P. Thiagarajan. Synthesizing distributed transition systems from global specifications. In *FSTTCS'99*, volume 1738 of *LNCS*. Springer, 1999.

[9] R. Cieslak, C. Desclaux, A. Fawaz, and P. Varaiya. Supervisory control of discrete-event processes with partial observations. *IEEE Transactions on Automatic Control*, 33:249–260, 1988.

[10] P. Cori and D. Perrin. Automates et commutations partielles. *RAIRO - Informatique Théorique et Applications*, 19:21–32, 1985.

[11] V. Diekert and G. R. (eds.). *The Book of Traces*. World Scientific, 1995.

[12] V. Diekert and A. Muscholl. Construction of asynchronous automata. In [11].

[13] C. Duboc. Mixed product and asynchronous automata. *Theoretical Computer Science*, 48:183–199, 1986.

[14] P. Gastin, B. Lerman, and M. Zeitoun. Distributed games and distributed control for asynchronous systems. In *LATIN'04*, volume 2976 of *LNCS*. Springer, 2004.

[15] A. Giua. Pn state estimators based on event observation. In *CDC'97*, 1997.

[16] O. Kupferman and M. Vardi. Synthesizing distributed systems. In *Logic in Computer Science*, 2001.

[17] H. Lamouchi and J. Thistle. Effective control synthesis for DES under partial observations. In *IEEE Conference on Decision and Control*, 2000.

[18] P. Madhusudan and P. Thiagarajan. Distributed controller synthesis for local specifications. In *28th ICALP, Crete, Greece*, LNCS 2076, 2001.

[19] P. Madhusudan and P. Thiagarajan. A decidable class of asynchronous distributed controllers. In *CONCUR'02*, LNCS 2421, 2002.

[20] S. Mohalik and I. Walukiewicz. Distributed games. In *FSTTCS'03*, volume 2914 of *LNCS*. Springer, 2003.

[21] R. Morin. Decompositions of asynchronous systems. In *CONCUR'98*, volume 1466 of *LNCS*. Springer, 1998.

[22] M. Mukund, K. N. Kumar, and M. Sohoni. Synthesizing distributed finite-state systems from MSCs. In *CONCUR'00*, volume 1877 of *LNCS*. Springer, 2000.

[23] A. Pnueli and R. Rosner. Distributed reactive systems are hard to synthesize. In *Proc. of the 31th FOCS*, pages 746–757, 1990.

[24] K. Rudie and W. Wonham. Think globally, act locally: Decentralized supervisory control. *IEEE Transactions on Automatic Control*, 37, 1992.

[25] A. Stefanescu, J. Esparza, and A. Muscholl. Synthesis of distributed algorithms using asynchronous automata. In *CONCUR'03*, volume 2761 of *LNCS*. Springer, 2003.

[26] S. Tripakis. Undecidable problems of decentralized observation and control. In *IEEE Conference on Decision and Control (CDC'01)*, 2001.

[27] S. Tripakis. Decentralized control of discrete event systems with bounded or unbounded delay communication. *IEEE Transactions on Automatic Control*, 49(9), Sept. 2004.

[28] J. Tsitsiklis and M. Athans. On the complexity of decentralized decision making and detection problems. *IEEE Transactions on Automatic Control*, 30(5), 1985.

[29] W. Zielonka. Notes on finite asynchronous automata. *RAIRO - Informatique Théorique et Applications*, 21:99–135, 1987.