ELSEVIER

Theoretical Computer Science 139 (1995) 163–186

Theoretical Computer Science

Sharing out control in distributed processes☆

Anne Bergeron*

LACIM, Université du Québec à Montréal, C.P. 8888 Succ. Centre-Ville, Montréal, Canada, H3C 3P8

Received November 1993; revised May 1994
Communicated by M. Nivat

Abstract

In this paper, we discuss the problem of distributed control of discrete processes. Given n sites from which only partial information is available about a process, we describe how to share out controllable events in order that the process meets a given global specification. As with many problems involving partial observation, the solution relies on inefficient algorithms. The second part of the paper treats the question of identifying problems that can be solved efficiently.

0. Introduction

A *discrete event process* is a process that changes state according to the discrete occurrence of events. It is modeled by an automaton on the set Σ of possible events. The central problem of discrete control theory is to describe how to restrict, observe, or report the behavior of such processes [4]. These problems are usually solved by constructing automata, called *controllers*, that will simultaneously function with the original process, and take the necessary control actions according to their state.

In this context, a *control action* is to prevent the occurrence of a particular event. Events that can be prevented are called *controllable*. For example, in a traffic control problem, the event 'a car crosses the intersection' can be prevented by a red signal, or a gate.

Suppose that we are given n sites, from which we can get only partial information about a process **P**. We want to construct n controllers, one at each site i, that together will prevent **P** from entering 'unwanted' states. The fact that only partial information is available at each site will impose constraints on these controllers. If the assignment of the controllable events to the various sites is also fixed, solutions to this problem can be found in [3, 5].

* Corresponding author E-mail: anne@lacim.uqam.ca.

☆ This work was partially supported by grants from BNR Ltd., FCAR of Québec and NSERC of Canada.

SSDI 0304-3975(94)00132-3

In this paper, we give a third solution which characterize, in a simple way, *all* possible assignments of controllable events such that a solution exists. As with many problems involving partial observation, the algorithms rely on (severely) inefficient algorithms. In order to tackle this problem, we introduce the concept of *linear observability*. We then show that, for a certain class of processes, linear observability is equivalent to observability, thus ensuring efficient computations.

Section 1 covers basic definitions. In Sect. 2–5, we present the problem of distributed control in terms of automata, and give algorithms to find all possible assignments of controllable events to various sites such that a solution is *effective*. Linear observability and its consequences are discussed in Sections 6–8.

1. Automata and discrete process

Let Σ be a finite set whose elements are called *events*, and Σ^* be the set of all finite sequences of elements of Σ. An *automaton* $\mathbf{A}$ on the set Σ of events is given by an arbitrary *partial* function – multiplicatively denoted by a dot $\cdot$ – called a *transition function*:

$$\cdot : \mathbf{S_A} \times \Sigma \to \mathbf{S_A}$$

where $\mathbf{S_A}$ is an arbitrary set, called the *states* of $\mathbf{A}$.

Every transition function can be naturally extended to any sequence x in Σ^* in the following way:

(i) $\mathbf{s} \cdot \lambda = \mathbf{s}$ where λ is the empty sequence,

(ii) $\mathbf{s} \cdot (x\sigma) = (\mathbf{s} \cdot x) \cdot \sigma$ whenever the right-hand side is defined.

Among the states $\mathbf{S_A}$, we distinguish an *initial state* $\mathbf{i_A}$ and a subset $\mathbf{F_A} \subseteq \mathbf{S_A}$ of *final* or *marked states*. We will denote by $\bar{\mathbf{A}}$ the automaton obtained by marking all the states of an automaton $\mathbf{A}$. The language *recognized* by the automaton $\mathbf{A}$ is the set $L(\mathbf{A}) = \{x \mid \mathbf{i} \cdot x \in \mathbf{F_A}\}$. A state $\mathbf{s}$ is *accessible* if there is at least one sequence x such that $\mathbf{i_A} \cdot x = \mathbf{s}$. We will always assume that $\mathbf{S_A}$ contains only accessible states.

Definition 1.1 (*Partial ordering of automata*). Let $\mathbf{A}$ and $\mathbf{B}$ be two automata. The relation $\mathbf{A} \leqslant \mathbf{B}$ holds whenever

$$L(\mathbf{A}) \subseteq L(\mathbf{B}) \quad \text{and} \quad L(\bar{\mathbf{A}}) \subseteq L(\bar{\mathbf{B}}).$$

When both $\mathbf{A} \leqslant \mathbf{B}$ and $\mathbf{B} \leqslant \mathbf{A}$, we write $\mathbf{A} \approx \mathbf{B}$.

Definition 1.2 (*Product of automata*). Given two automata $\mathbf{A}$ and $\mathbf{B}$, the *product* $\mathbf{A} \times \mathbf{B}$ has states $\mathbf{S} = \mathbf{S_A} \times \mathbf{S_B}$ and transition function:

$$\cdot : \mathbf{S} \times \Sigma^* \to \mathbf{S}$$

where $(\mathbf{s}, \mathbf{t}) \cdot x = (\mathbf{s} \cdot x, \mathbf{t} \cdot x)$ whenever the right hand side is defined.

The product has initial state $\mathbf{i} = (\mathbf{i_A}, \mathbf{i_B})$ and marked states $\mathbf{F} = \{(\mathbf{s},\mathbf{t}) \mid \mathbf{s} \in \mathbf{F_A} \text{ and } \mathbf{t} \in \mathbf{F_B}\}$.

Elementary properties of these definitions will be used throughout this paper.

Proposition 1.3. *Let* $\mathbf{A}, \mathbf{B}, \mathbf{C}$ and $\mathbf{D}$ *be any automata*:

(i) $\mathbf{A} \times \mathbf{B} \leqslant \mathbf{A}$ *and* $\mathbf{A} \times \mathbf{B} \leqslant \mathbf{B}$.

(ii) $\mathbf{A} \leqslant \mathbf{B}$ *if and only if* $\mathbf{A} \approx \mathbf{B} \times \mathbf{A}$.

(iii) $\mathbf{C} \leqslant \mathbf{A} \times \mathbf{B}$ *if and only if* $\mathbf{C} \leqslant \mathbf{A}$ *and* $\mathbf{C} \leqslant \mathbf{B}$.

Example 1.4. Consider the following process (Fig. 1) where tokens can be exchanged between buffers W, A, B and C. Suppose that buffer W has capacity 3, and all others have capacity 1. In this process, the various events can be represented by pairs of letters, AB meaning, for example, that a token is transformed from buffer A to buffer B. The automaton $\mathbf{P}$ modeling the process has 9 states, of which 8 are *legal* (not exceeding any buffer capacity) and one is an *alarm* state, meaning an overflow of one of the buffers. In Fig. 2, each legal state is depicted by the contents of buffers A, B and C. For clarity, we labeled only one of the arrows when an event is reversible.

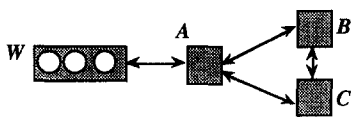

Fig. 1. Exchange of tokens between 4 buffers.

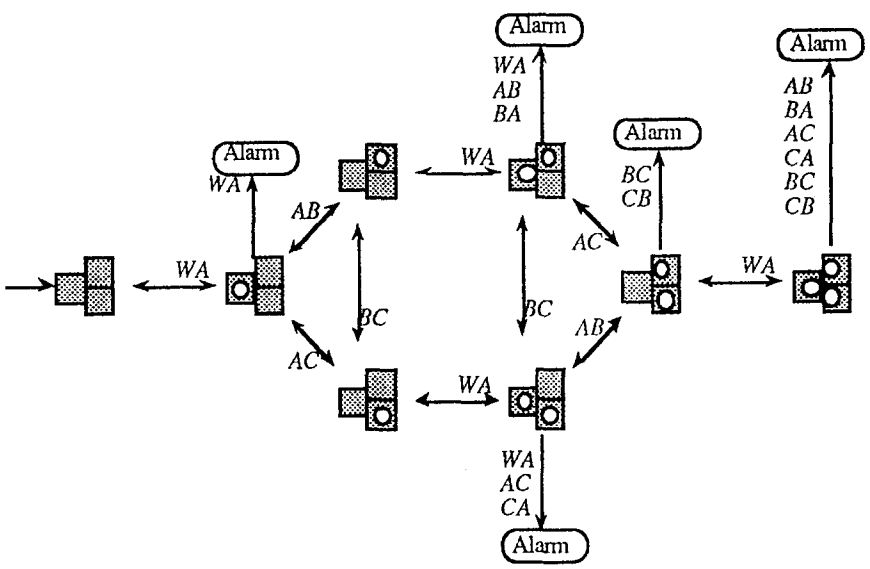

Fig. 2. The automaton $\mathbf{P}$.

Fig. 3.

The automaton **Z** of legal (shaded) states, with possible moves between them, can be thought of as the *specification* of the legal behavior of **P**. If we assume that all states are final, it is easy to see that $\mathbf{Z} \leqslant \mathbf{P}$. Fig. 2 displays clearly what should be the control actions such that only legal states are possible. For example, in the state shown in Fig. 3, the three events WA, AB and BA must be prevented, since they lead to the alarm state.

Depending on the various states of **Z**, the events WA, AB, BA, AC, CA, BC, and CB must eventually be disabled or authorized by 'controllers' (only AW can never lead to an alarm state).

2. Distributed control and control automata

Let **P** be a discrete process, our objective is to construct automata $\mathbf{C}_1, \ldots, \mathbf{C}_n$, such that when they function together with **P**, the whole process will have a prescribed behavior. This new process is described by the product:

$$\mathbf{P} \times \mathbf{C}_1 \times \cdots \times \mathbf{C}_n \text{ or shortly } \mathbf{P} \times \prod \mathbf{C}_i.$$

Each of these automata will have to perform control actions (preventing events). This aspect is, surprisingly, settled by the following definition.

Definition 2.1 (*Control automata*). A *control automaton*, or *controller*, is an automaton with a specified subset of events Σ_C, its *controllable events*.

That this definition captures the usual notion of a 'controller' needs some explanations. Its apparent inadequacy comes from the fact that we need to formalize only some aspects of the notion of control action. Indeed, in order to be able to control a discrete process, we must first have the *capacity* to take a control action, and *decide* which control action to take.

The *capacity* to take action is a design decision: we grant this capacity to a controller by prescribing a subset of controllable events, Σ_C. The intended meaning is that these events can be prevented by the controller.

Since we are working with automata, the *decision* to take action can depend only on the state of the controller. In a given state **s**, the controller must decide if it will prevent or authorize a given event σ. If the decision is to prevent it, then the event σ must never occur in state **s**, so $\mathbf{s} \cdot \sigma$ can, for all practical purposes, be undefined. If the decision is to authorize it, then the controller must know what to do in case of the occurrence of σ, thus $\mathbf{s} \cdot \sigma$ must be defined. The simplest strategy for a controller is thus to prevent an event σ in Σ_C iff $\mathbf{s} \cdot \sigma$ is undefined.

When several controllers are working together, we are interested in the net result of their controlling actions. In general, we will want that any event undefined in the product be prevented. This is captured by the following definition.

Definition 2.2 (*Effective products of controllers*). Let $\mathbf{C}_1, \ldots, \mathbf{C}_n$ be n controllers, each having a set Σ_{C_i} of controllable events, the product $\prod \mathbf{C}_i$, is *effective* if for each accessible state $\mathbf{S} = (\mathbf{s}_1, \ldots, \mathbf{s}_n)$ of $\prod \mathbf{C}_i$, and for each $\sigma \in \Sigma$ such that $\mathbf{S} \cdot \sigma$ is undefined we have

$$\exists i \text{ such that } \mathbf{s}_i \cdot \sigma \text{ is undefined and } \sigma \in \Sigma_{C_i}.$$

Definition 2.2 says that a group of controller is effective if, for each 'global' action that has to be prevented, at least one of the controllers was *able* to prevent it and *did* prevent it. When the product has only one factor, we have the notion of *effective control automaton*. Such an automaton has the property that if $\mathbf{s} \cdot \sigma$ is undefined, then σ must be a controllable event.[1]

Remark 2.3. Assume that an automaton $\mathbf{P}$ faithfully models an 'existing' process, that is all (and only) possible sequences are defined in $\mathbf{P}$. The automaton $\mathbf{P}$ can be viewed as an effective controller by setting

$$\Sigma_C = \Sigma.$$

In this case, impossible events are interpreted as prevented events.

We will also consider controllers with no controllable events (i.e. $\Sigma_C = \emptyset$). Such controllers are clearly 'ineffective' in the sense that, if $\mathbf{A} \times \mathbf{B}$ is effective and $\mathbf{A}$ has no controllable events, then $\mathbf{B}$ must be effective.

Algorithm 2.4 (*Constraints characterizing all effective products*). Given n automata $\mathbf{C}_1, \ldots, \mathbf{C}_n$ it is possible to characterize easily all distributions Σ_{C_i} such that the product $\prod \mathbf{C}_i$, is effective. Indeed, for each state $\mathbf{S} = (\mathbf{s}_1, \ldots, \mathbf{s}_n)$ of $\prod \mathbf{C}_i$, and for each $\sigma \in \Sigma$ such that $\mathbf{S} \cdot \sigma$ is undefined we must have, by definition

$$\sigma \in \bigcup_{i \in \mathbf{J}_S} \Sigma_{C_i},$$

where $\mathbf{J}_S = \{i \mid \mathbf{s}_i \cdot \sigma \text{ is undefined}\}$.

On the other hand, any distribution Σ_{C_i} which respects all these constraints, for all possible $\mathbf{S}$ and σ, will yield an effective product.

Verifying that a given distribution Σ_{C_i} yields an effective product $\prod \mathbf{C}_i$ amounts to the traversal of the graph of the automata $\prod \mathbf{C}_i$. If, on the other hand, the problem is to

[1] Effective control automata are the control automata of [3].

distribute a set Σ_C of controllable events among various sites, Algorithm 2.4 provides a list of constraints of the form

σ must be assigned to at least one controller in the set $\mathbf{J}_S$.

which has at least one solution if and only if $\mathbf{S} \cdot \sigma$ is undefined implies that $\sigma \in \Sigma_C$. This solution is obtained by trivially setting $\Sigma_{C_i} = \Sigma_C$. Other considerations can then be applied in order to propose a particular architecture: minimality (σ is assigned to the least number of controllers), convenience (σ is assigned to the nearest controller), etc. An example of this kind of computation is given in Section 5.

3. Partial observation and observation automata

Suppose that we are given n sites, each having a partial view of a process **P**. In Example 1.4, we could have, for instance, the two sites (see Fig. 4) where Site 1 cannot observe exchanges between buffers B and C, and Site 2 cannot observe exchanges between buffers W and A.

The fact that each site have only partial information about the process will impose restrictions on the kind of automata that can function as controllers on these various sites. By 'partial information' we will mean that, for each site i, we are given a subset Σ_{U_i} of Σ, its *unobservable events*. Clearly, any automata $\mathbf{C}_i$ 'placed' in Site i cannot change state upon the occurrence of events in Σ_{U_i}, but the obvious solution to restrict these automata to the complement of Σ_{U_i} does not work. Indeed, since our objective is ultimately to turn these automata into controllers, we must allow the possibility of control over unobservable events. For example, one can prevent or authorize calls dialed from a telephone without actually monitoring those calls. We will model this by allowing the automata to 'loop' on unobservable events.

Definition 3.1 (*Observation automata*). An automaton **O** is an *observation automaton* with respect to Σ_U, its *unobservable events*, if whenever $\mathbf{s} \cdot \sigma$ is defined and $\sigma \in \Sigma_U$ we have $\mathbf{s} \cdot \sigma = \mathbf{s}$.

Given any automaton **A**, and a set of unobservable events Σ_U, it is always possible to construct an observation automaton greater than **A**, and which is minimal among

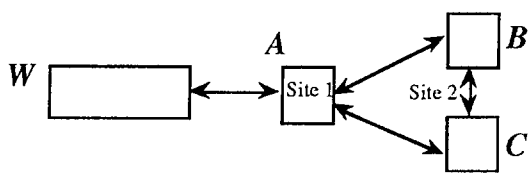

Fig. 4. Sites of observation of the process **P**.

all observation automata greater than **A**. This construction is basic in partial observation problems and is a variant of the well-known determinization algorithm in automata theory:

Algorithm 3.2 (*The minimal observer construction*). Let $\mathbf{S}_\mathbf{A}$ be the set of states of an automaton **A**, with initial state **i** and marked states $\mathbf{F}_\mathbf{A}$. Let Σ_U be a set of unobservable events. The *minimal observer* $\mathbb{O}(\mathbf{A})$ with respect to Σ_U is defined in the following way. The states of the automaton $\mathbb{O}(\mathbf{A})$ are nonempty subsets $\mathscr{P}^+(\mathbf{S}_\mathbf{A})$ of $\mathbf{S}_\mathbf{A}$. The initial state is:

$$\mathbf{I} = \{\mathbf{i}\cdot u \mid \mathbf{i}\cdot u \text{ is defined and } u \in \Sigma_U^*\}$$

and the marked states of $\mathbb{O}(\mathbf{A})$ are all subsets that contain at least one marked state of **A**. The transition function

$$\circ : \mathscr{P}^+(\mathbf{S}_\mathbf{A}) \times \Sigma \to \mathscr{P}^+(\mathbf{S}_\mathbf{A})$$

is defined with the following rule.

Let $\mathbf{S} \subseteq \mathbf{S}_\mathbf{A}$. If $\mathbf{s}\cdot\sigma$ is defined in **A** for at least one $\mathbf{s} \in \mathbf{S}$ then $\mathbf{S}\circ\sigma$ is defined and

(i) if $\sigma \notin \Sigma_U$, $\mathbf{S}\circ\sigma = \{\mathbf{s}\cdot\sigma u \mid \mathbf{s} \in \mathbf{S}, \mathbf{s}\cdot\sigma u \text{ is defined and } u \in \Sigma_U^*\}$

(ii) if $\sigma \in \Sigma_U$, $\mathbf{S}\circ\sigma = \mathbf{S}$

otherwise $\mathbf{S}\circ\sigma$ is undefined.

Informally, the initial state of $\mathbb{O}(\mathbf{A})$ is the set of states reachable from the initial state $\mathbf{i}_\mathbf{A}$ with unobservable sequences: the observer cannot distinguish between these states. The construction then proceeds recursively from the initial state. Suppose the observer 'thinks' it could be in any of the states of a subset **S**, if an observable event σ occurs, we have to compute all the states reachable with unobservable sequences from states of the form $\mathbf{s}\cdot\sigma$, where **s** is in **S** (see Fig. 5).

The following result summarizes the basic properties of the minimal observer construction. Its proof can be found in [3].

Theorem 3.3. *Let* **A** *be any automata, and* Σ_U *be a set of events, then*

(i) $\mathbb{O}(\mathbf{A})$ *is an observation automata with respect to* Σ_U.

(ii) $\mathbf{A} \leqslant \mathbb{O}(\mathbf{A})$.

(iii) *If* **X** *is an observation automata wrt* Σ_U *such that* $\mathbf{A} \leqslant \mathbf{X}$, *then* $\mathbb{O}(\mathbf{A}) \leqslant \mathbf{X}$.

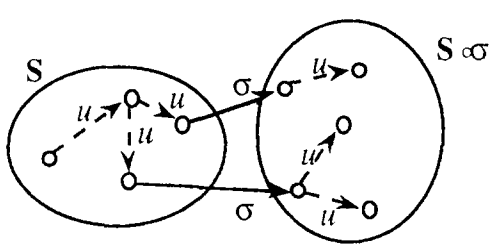

Fig. 5.

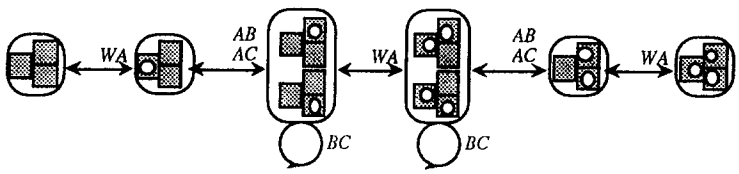

Fig. 6. Observer for Site 1 (BC and CB unobservable).

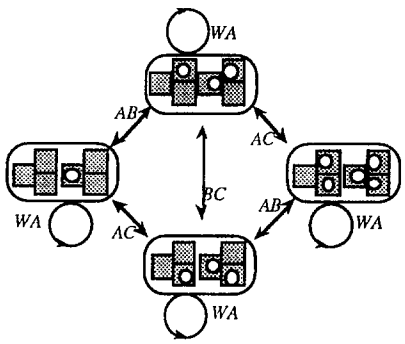

Fig. 7. Observer for Site 2 (AW and WA unobservable).

Example 3.4 (*Example* 1.4 *continued*). In Example 1.4, we were given two sites with the following unobservable events:

Site 1: BC and CB,

Site 2: AW and WA.

Applying the observation construction, with these sets, to the specification $\mathbf{Z}$ (that is, all the legal states), we get the observers of Figs. 6 and 7.

The automata of Figs. 6 and 7 are indeed observation automata, since they loop on all their respective unobservable events. Observe that this condition does not imply that there is a loop on each state. The 'no-loop' states are of two kinds: the event is *impossible* in the process $\mathbf{P}$ (like the event BC in the two first states of Fig. 6); or the event must be *prevented* so that the process does not enter an alarm state (like the event BC in the last two states of Fig 6). In the latter case, we can see that, despite the fact BC is unobservable from Site 1, the observer can decide effectively whether to prevent or authorize it.

4. Control and partial observation

Let $\mathbf{P}$ be a process, a *specification* for the process $\mathbf{P}$ will be modeled by an automaton $\mathbf{Z} \leqslant \mathbf{P}$. This assumption is natural since any legal and impossible sequence of events can be removed from the specification.

Given n sites the problem of control under partial observation is to construct n automata, one at each site, such that when these automata function simultaneously with the process $\mathbf{P}$, the global process 'meets' all specified behaviors, and all illegal behaviors are 'prevented'. This somewhat vague statement can be formalized within automata theory in the following way.

Suppose that $\mathbf{C}_1, \ldots, \mathbf{C}_n$ are n automata functioning with the process $\mathbf{P}$. Consider the product

$$\mathbf{P} \times \prod \mathbf{C}_i.$$

If $\mathbf{C}_i$ must function with only the information available at site i, it must be an observation automaton with respect to the set Σ_{U_i} of unobservable events at site i. Moreover, if the global process is to meet all specified behaviors, we must have

$$\mathbf{Z} \leqslant \mathbf{P} \times \prod \mathbf{C}_i.$$

That is, any sequence defined or marked by $\mathbf{Z}$, must also be defined and marked by $\mathbf{P} \times \prod \mathbf{C}_i$.

On the other hand, the purpose of the automata $\prod \mathbf{C}_i$ is to restrict the behavior of $\mathbf{P}$, such that only sequences defined or marked by $\mathbf{Z}$ are defined. Thus we want also that

$$\mathbf{P} \times \prod \mathbf{C}_i \leqslant \mathbf{Z}.$$

The next definition sums up these conditions.

Definition 4.1 (*Observable specification*). Given n sites with unobservable events Σ_{U_i}, a specification $\mathbf{Z}$ of a process $\mathbf{P}$ is *observable* if there exists automata $\mathbf{C}_1, \ldots, \mathbf{C}_n$ such that

(i) $\mathbf{C}_i$ is an observation automaton with respect to Σ_{U_i},
(ii) $\mathbf{Z} \approx \mathbf{P} \times \prod \mathbf{C}_i$.

The next theorem gives a general criterion for observability. It says that, in order to establish observability, it suffices to consider the minimal observers of $\mathbf{Z}$.

Theorem 4.2. *Given n sites with unobservable events Σ_{U_i}. Let $\mathbf{Z}$ be a specification of a process $\mathbf{P}$, and let $\mathbb{O}_i(\mathbf{Z})$ be the minimal observer with respect to Σ_{U_i}, then*

$\mathbf{Z}$ *is observable if and only if* $\mathbf{Z} \approx \mathbf{P} \times \prod \mathbb{O}_i(\mathbf{Z})$.

Proof. The 'if' part is obvious. On the other hand, we always have that

$$\mathbf{Z} \leqslant \mathbf{P} \times \prod \mathbb{O}_i(\mathbf{Z})$$

since $\mathbf{Z} \leqslant \mathbf{P}$ and $\mathbf{Z} \leqslant \mathbb{O}_i(\mathbf{Z})$ for each i. Suppose now that $\mathbf{Z}$ is observable by $\prod \mathbf{C}_i$, then $\mathbf{Z} \leqslant \mathbf{C}_i$ and since each $\mathbf{C}_i$ is an observation automata, we have, by Theorem 3.3 (iii), $\mathbb{O}_i(\mathbf{Z}) \leqslant \mathbf{C}_i$. Thus

$$\mathbf{P} \times \prod \mathbb{O}_i(\mathbf{Z}) \leqslant \mathbf{P} \times \prod \mathbf{C}_i$$

yielding

$$\mathbf{P} \times \prod \mathbb{O}_i(\mathbf{Z}) \leqslant \mathbf{Z}. \quad \square$$

Once it is established that a specification is observable, we still have the problem of determining the distribution Σ_{C_i} of controllable events among the sites in order that the product $\mathbf{P} \times \prod \mathbf{C}_i$ behaves 'operationally' like the formal specification $\mathbf{Z}$, meaning that every possible sequence of events in $\mathbf{P}$, that is undefined in $\prod \mathbf{C}_i$, is prevented by at least one of the controllers. Assuming that $\mathbf{P}$ is a controller with controllable events Σ (see Remark 2.3), we can express this condition simply by stating that the product $\mathbf{P} \times \prod \mathbf{C}_i$ must be effective with respect to Σ_{C_i}.

The following theorem states that applying Algorithms 2.4 to the set of observers $\mathbb{O}_i(\mathbf{Z})$ yields all possible solutions to the distributed control problem.

Theorem 4.3. *If $\mathbf{Z}$ is observable by $\prod \mathbf{C}_i$, then the effectiveness of the product $\mathbf{P} \times \prod \mathbf{C}_i$ with respect to the distributions Σ_{C_i} implies the effectiveness of the product $\mathbf{P} \times \prod \mathbb{O}_i(\mathbf{Z})$ with the same distribution.*

Proof. Suppose that the product $\mathbf{P} \times \prod \mathbf{C}_i$ is effective with respect to the distribution Σ_{C_i}. Let $\mathbf{S} = (\mathbf{s}, \mathbf{s}_1, \ldots, \mathbf{s}_n)$ be an accessible state of $\mathbf{P} \times \prod \mathbb{O}_i(\mathbf{Z})$ with the sequence x. Suppose that $\mathbf{S} \cdot \sigma$ is undefined. Since $\mathbf{P} \times \prod \mathbb{O}_i(\mathbf{Z}) \approx \mathbf{P} \times \prod \mathbf{C}_i$, the sequence x is also defined in $\mathbf{P} \times \prod \mathbf{C}_i$ leading this automaton in state $\mathbf{T} = (\mathbf{s}, t_1, \ldots, \mathbf{t}_n)$ such that $\mathbf{T} \cdot \sigma$ is undefined. Since the product $\mathbf{P} \times \prod \mathbf{C}_i$ is effective, either $\mathbf{s} \cdot \sigma$ is undefined, or there exists j such that $\mathbf{t}_j \cdot \sigma$ is undefined and $\sigma \in \Sigma_{C_j}$. If $\mathbf{s} \cdot \sigma$ is undefined, the case is settled. In the other cases, since $\mathbb{O}_j(\mathbf{Z}) \leqslant \mathbf{C}_j$, $\mathbf{s}_j \cdot \sigma$ must also be undefined. $\square$

These results give a method to find all possible assignments of controllable events Σ_{C_i} among n sites. Given a specification $\mathbf{Z}$ of a process $\mathbf{P}$, and sets of unobservable events Σ_{U_i}, we must

(1) construct the minimal observers $\prod \mathbb{O}_i(\mathbf{Z})$ using Algorithm 3.2,
(2) test the equivalence $\mathbf{Z} \approx \mathbf{P} \times \prod \mathbb{O}_i(\mathbf{Z})$,
(3) and apply Algorithm 2.4 to the product $\mathbf{P} \times \prod \mathbb{O}_i(\mathbf{Z})$.

Note that, since the minimal observer construction is based on a subset construction, the various automata in step (1) may yield automata that have an exponential number of states in terms of the number of states of $\mathbf{Z}$. This is a serious drawback that will be discussed in detail in Section 6.

In certain situations, step (2) can be skipped altogether. Indeed, when all the states of the specification are final, we have the following theorem.

Theorem 4.4. *Suppose that $\mathbf{Z} = \bar{\mathbf{Z}}$ then the product $\mathbf{Z} \times \mathbf{P} \times \prod \mathbb{O}_i(\mathbf{Z})$ is effective with respect to $\emptyset$, Σ and Σ_{C_i}, if and only if*

(1) *$\mathbf{P} \times \prod \mathbb{O}_i(\mathbf{Z})$ is effective with respect to Σ and Σ_{C_i},*
(2) *$\mathbf{Z} \approx \mathbf{P} \times \prod \mathbb{O}_i(\mathbf{Z})$.*

Proof. Suppose first that $\mathbf{P} \times \prod \mathbb{O}_i(\mathbf{Z})$ is effective and $\mathbf{Z} \approx \mathbf{P} \times \prod \mathbb{O}_i(\mathbf{Z})$. Let $\mathbf{S} = (\mathbf{z}, \mathbf{s}, \mathbf{s}_1, \ldots, \mathbf{s}_n)$ be an accessible state of the product

$$\mathbf{Z} \times \mathbf{P} \times \prod \mathbb{O}_i(\mathbf{Z})$$

such that $\mathbf{S} \cdot \sigma$ is undefined. If $\mathbf{z} \cdot \sigma$ is defined, the effectiveness of $\mathbf{P} \times \prod \mathbb{O}_i(\mathbf{Z})$ is sufficient to show the effectiveness of the whole product. If $\mathbf{z} \cdot \sigma$ is undefined, since $\mathbf{Z} \approx \mathbf{P} \times \prod \mathbb{O}_i(\mathbf{Z})$, then

$$(\mathbf{s}, \mathbf{s}_1, \ldots, \mathbf{s}_n) \cdot \sigma$$

must also be undefined, and again we can apply the effectiveness of $\mathbf{P} \times \prod \mathbb{O}_i(\mathbf{Z})$.

On the other hand, suppose that $\mathbf{Z} \times \mathbf{P} \times \prod \mathbb{O}_i(\mathbf{Z})$ is effective. Since we assumed that $\mathbf{Z}$ was a controller with $\Sigma_C = \emptyset$, we can conclude immediately, by Remark 2.3, that $\mathbf{P} \times \prod \mathbb{O}_i(\mathbf{Z})$ is effective. In order to show that

$$\mathbf{Z} \approx \mathbf{P} \times \prod \mathbb{O}_i(\mathbf{Z}).$$

it is sufficient to show that

$$\mathbf{P} \times \prod \mathbb{O}_i(\mathbf{Z}) \leqslant \mathbf{Z}$$

since the reverse inequality is always true. Let x be defined in $\mathbf{P} \times \prod \mathbb{O}_i(\mathbf{Z})$ and x' be the longest prefix of x defined in $\mathbf{Z}$. If $x \neq x'$ then x can be written as $x'\sigma y$. When x' has been parsed by the product $\mathbf{Z} \times \mathbf{P} \times \prod \mathbb{O}_i(\mathbf{Z})$, this product is in state

$$\mathbf{S} = (\mathbf{z}, \mathbf{s}, \mathbf{s}_1, \ldots, \mathbf{s}_n)$$

and $\mathbf{S} \cdot \sigma$ is undefined because $\mathbf{Z} \cdot \sigma$ is undefined. Since the product is effective, there is at least one controller for which σ is undefined and σ is in Σ_C. This controller cannot be $\mathbf{Z}$, since $\mathbf{Z}$ has no controllable events, so it must be either $\mathbf{P}$ or one of the $\mathbb{O}_i(\mathbf{Z})$. Thus $x'\sigma$ is not defined in $\mathbf{P} \times \prod \mathbb{O}_i(\mathbf{Z})$ and we must have $x = x'$.

If x is marked by $\mathbf{P} \times \prod \mathbb{O}_i(\mathbf{Z})$, it is defined in $\mathbf{Z}$. Since we assumed that $\mathbf{Z} = \bar{\mathbf{Z}}$, then x is marked by $\mathbf{Z}$. □

5. An example of computation

In this section, we show a complete computation of all the distributions of controllable events for the process described in Example 1.4. We already obtained the two observers in Section 3 (Fig. 6 and 9). Since all states are final, we have only to test the effectiveness of

$$\mathbf{Z} \times \mathbf{P} \times \prod \mathbb{O}_i(\mathbf{Z}).$$

Fortunately, this product has a simple structure (that can be identified with the automaton $\mathbf{Z}$) (see Fig. 8).

There are 15 nontrivial control constraints associated with this problem, each of which corresponds to an arrow going to the alarm state in the process (Fig. 2). Other

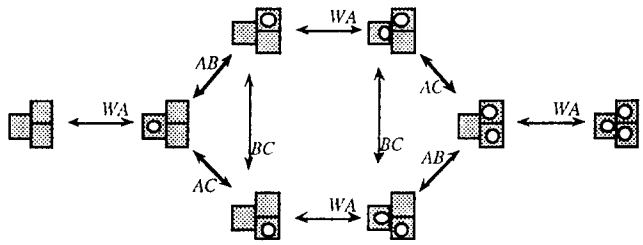

Fig. 8. The product $\mathbf{Z} \times \mathbf{P} \times \prod \mathbb{O}_i(\mathbf{Z})$.

S	σ	$J_S = \{i \mid s_i \cdot \sigma$ is undefined$\}$
	*WA*	{1}
	*WA*	{1}
	AB	{2}
	BA	{1}
	*WA*	{1}
	AC	{2}
	CA	{1}
	*BC*	{1, 2}
	CB	{1, 2}
	*AB*	{1, 2}
	BA	{1}
	AC	{1, 2}
	CA	{1}
	BC	{1, 2}
	CB	{1, 2}

Fig. 9.

constraints correspond to impossible events of the process **P**. Fig. 9 displays all the 15 constraints, and is obtained by noting, for each state **S** and each undefined and possible transition σ, the sites for which the transition is undefined. This set of constraints is easily solved and we obtain the following possible distributions:

WA, *BA* and *CA* must be assigned to the controller in Site 1,

AB and *AC* must be assigned to the controller in Site 2,

BC and *CB* can be assigned to any of two controllers.

It is interesting to note that *BC* and *CB* can be assigned to Site 1, although Site 1 cannot observe these events.

6. Linear observability

Although we obtained a computational characterization of all the solutions of a distributed control problem, it has, as noted, the major drawback to rely on an inefficient algorithm. Indeed, Algorithm 3.2 can – and does – lead to computational disaster since the number of states of an observer can be exponential in the number of states of the specification [6, 7].

In this section, we develop some tools to investigate problems that can be solved efficiently. These tools are of practical importance since various formalism used to describe distributed processes are known to lead to some kind of state explosion. Software tools have been developed to analyze those specification, and they can handle gracefully automata that have several thousands of states [2]. With specifications of this size, however, the prospect of constructing controllers using algorithms such as Algorithm 3.2 is completely unrealistic.

In order to develop a theory of efficiently observable processes, we first focus on a much stronger partial order relation between automata.

Definition 6.1 (*Morphisms between automta*). Let $\mathbf{A}$, $\mathbf{B}$ be automata with states $\mathbf{S}_A$, $\mathbf{S}_B$ initial states $\mathbf{i}_A$, $\mathbf{i}_B$, and final states $\mathbf{F}_A$, $\mathbf{F}_B$. A *morphism* from $\mathbf{A}$ to $\mathbf{B}$ is defined by a function $f: \mathbf{S}_A \to \mathbf{S}_B$ such that

(1) $f(\mathbf{i}_A) = \mathbf{i}_B$,

(2) $f(\mathbf{F}_A) \subseteq \mathbf{F}_B$,

(3) If $\mathbf{s} \cdot \sigma$ is defined then $f(\mathbf{s}) \cdot \sigma$ is also defined and equal to $f(\mathbf{s} \cdot \sigma)$, that is, the following diagram commutes:

$$\begin{array}{ccc} \mathbf{S}_A \times \Sigma & \xrightarrow{\langle f, 1_\Sigma \rangle} & \mathbf{S}_B \times \Sigma \\ \downarrow & & \downarrow \\ \mathbf{S}_A & \xrightarrow{f} & \mathbf{S}_B. \end{array}$$

When there exists a morphism from $\mathbf{A}$ to $\mathbf{B}$, we will write

$$\mathbf{A} \to \mathbf{B}$$

and we have immediately the following proposition.

Proposition 6.2. *If* $\mathbf{A} \to \mathbf{B}$ *then* $\mathbf{A} \leqslant \mathbf{B}$.

If $\mathbf{A} \to \mathbf{B}$ then the function $f: \mathbf{S}_A \to \mathbf{S}_B$ is unique. Occasionally, we will refer explicitly to this function with the notation $\mathbf{A} \xrightarrow{f} \mathbf{B}$. When f is injective, $\mathbf{A}$ is a *subautomaton* of $\mathbf{B}$, and we will use the notation $\mathbf{A} \hookrightarrow \mathbf{B}$. If both $\mathbf{A} \to \mathbf{B}$ and $\mathbf{B} \to \mathbf{A}$, then $\mathbf{A}$ and $\mathbf{B}$ are *isomorphic*, and we will write $\mathbf{A} \leftrightarrow \mathbf{B}$. Two isomorphic automata are essentially the same, up to a renaming of the states. Finally, if $\mathbf{A} \xrightarrow{f} \mathbf{B}$ and $\mathbf{B} \xrightarrow{g} \mathbf{C}$ then $\mathbf{A} \to \mathbf{C}$, by composition of g and f.

If $\mathbf{A} \xrightarrow{f} \mathbf{B}$, we will note $f(\mathbf{A})$ the *image* automaton of $\mathbf{A}$. That is, the states of $f(\mathbf{A})$ are

$$\mathbf{S}_{f(\mathbf{A})} = \{f(\mathbf{s}) \mid \mathbf{s} \in \mathbf{S}_{\mathbf{A}}\}$$

with initial state $f(\mathbf{i}_{\mathbf{A}})$, final states $f(\mathbf{F}_{\mathbf{A}})$, and transition function:

$$\cdot : \mathbf{S}_{f(\mathbf{A})} \times \Sigma \to \mathbf{S}_{f(\mathbf{A})}$$

where $\mathbf{t} \cdot \sigma$ is defined iff $\mathbf{s} \cdot \sigma$ is defined in $\mathbf{A}$ for at least one state $\mathbf{s}$ in $f^{-1}(\mathbf{t})$. Note that if $\mathbf{A}$ has k states, then $f(\mathbf{A})$ has at most k states. We have the decomposition given by the following proposition.

Proposition 6.3. If $\mathbf{A} \xrightarrow{f} \mathbf{B}$, *then* $\mathbf{A} \to f(\mathbf{A}) \hookrightarrow \mathbf{B}$.

The partial order relation $\mathbf{A} \to \mathbf{B}$ between automata enjoys properties similar to those in Proposition 1.3.

Proposition 6.4. *Let* $\mathbf{A}, \mathbf{B}$, *and* $\mathbf{C}$ *be any automata, then*

(i) $\mathbf{A} \times \mathbf{B} \to \mathbf{A}$ *and* $\mathbf{A} \times \mathbf{B} \to \mathbf{B}$.

(ii) $\mathbf{A} \to \mathbf{B}$ *if and only if* $\mathbf{A} \leftrightarrow \mathbf{A} \times \mathbf{B}$.

(iii) $\mathbf{C} \to \mathbf{A} \times \mathbf{B}$ *if and only if* $\mathbf{C} \to \mathbf{A}$ *and* $\mathbf{C} \to \mathbf{B}$.

Proof. (i) The morphisms are obtained through the projections $p_1(\mathbf{s}, \mathbf{t}) = \mathbf{s}$ and $p_2(\mathbf{s}, \mathbf{t}) = \mathbf{t}$. The three properties of Definition 6.1 are easy consequences of the definition of the product.

(ii) Suppose $\mathbf{A} \xrightarrow{f} \mathbf{B}$ and define the function $g : \mathbf{S}_{\mathbf{A}} \to \mathbf{S}_{\mathbf{A} \times \mathbf{B}}$ as $g(\mathbf{s}) = (\mathbf{s}, f(\mathbf{s}))$. We have that

(1) $g(\mathbf{i}_{\mathbf{A}}) = (\mathbf{i}_{\mathbf{A}}, f(\mathbf{i}_{\mathbf{A}})) = (\mathbf{i}_{\mathbf{A}}, \mathbf{i}_{\mathbf{B}}) = \mathbf{i}_{\mathbf{A} \times \mathbf{B}}$,

(2) if $\mathbf{s} \in \mathbf{F}_A$, then $g(\mathbf{s}) = (\mathbf{s}, f(\mathbf{s})) \in \mathbf{F}_{\mathbf{A} \times \mathbf{B}}$ since $f(\mathbf{s}) \in \mathbf{F}_{\mathbf{B}}$,

(3) if $\mathbf{s} \cdot \sigma$ is defined then $g(\mathbf{s}) \cdot \sigma$ is also defined and

$$g(\mathbf{s}) \cdot \sigma = (\mathbf{s}, f(\mathbf{s})) \cdot \sigma = (\mathbf{s} \cdot \sigma, f(\mathbf{s}) \cdot \sigma) = (\mathbf{s} \cdot \sigma, f(\mathbf{s} \cdot \sigma)) = g(\mathbf{s} \cdot \sigma)$$

Thus $\mathbf{A} \to \mathbf{A} \times \mathbf{B}$. Since we always have $\mathbf{A} \times \mathbf{B} \to \mathbf{A}$, we get $\mathbf{A} \leftrightarrow \mathbf{A} \times \mathbf{B}$. On the converse, if $\mathbf{A} \to \mathbf{A} \times \mathbf{B}$, we obtain $\mathbf{A} \to \mathbf{B}$ by composition through the projection $\mathbf{A} \times \mathbf{B} \to \mathbf{B}$.

(iii) If $\mathbf{C} \to \mathbf{A} \times \mathbf{B}$, we obtain by composition through the projections that $\mathbf{C} \to \mathbf{A}$ and $\mathbf{C} \to \mathbf{B}$. On the other hand, if both if $\mathbf{C} \xrightarrow{f} \mathbf{A}$ and if $\mathbf{C} \xrightarrow{g} \mathbf{B}$ we construct the function $h(\mathbf{s}) = (f(\mathbf{s}), g(\mathbf{s}))$ which verifies the three properties. □

The next definition parallels the definition of observable specifications of Section 4.

Definition 6.5 (*Linearly observable specifications*). Given n sites with unobservable events Σ_{U_i}, a specification $\mathbf{Z}$ of a process $\mathbf{P}$ is *linearly observable* if there exists automata $\mathbf{C}_1, \ldots, \mathbf{C}_n$ such that

(i) $\mathbf{C}_i$ is an observation automata with respect to Σ_{U_i},

(ii) $\mathbf{Z} \leftrightarrow \mathbf{P} \times \prod \mathbf{C}_i$.

Clearly, a linearly observable specification is observable. The terminology *linear* comes from the fact that we can bound the number of states of the various observers by the number of states of the specification.

Proposition 6.6. *If* $\mathbf{Z}$ *is linearly observable, and if* $\mathbf{Z}$ *has* k *states, then it is linearly observable by automata that have at most* k *states.*

Proof. Suppose that $\mathbf{Z}$ is linearly observable by $\prod \mathbf{C}_i$, so that $\mathbf{Z} \to \mathbf{P} \times \prod \mathbf{C}_i$. By composing through projections, we obtain $\mathbf{Z} \to \mathbf{C}_i$. Let us note $f_i(\mathbf{Z})$ the image of $\mathbf{Z}$ by this morphism. Now, since both $\mathbf{Z} \to f_i(\mathbf{Z})$ and $\mathbf{Z} \to \mathbf{P}$ we have

$$\mathbf{Z} \to \mathbf{P} \times \prod f_i(\mathbf{Z}),$$

and from $f_i(\mathbf{Z}) \hookrightarrow \mathbf{C}_i$, we get

$$\mathbf{P} \times \prod f_i(\mathbf{Z}) \to \mathbf{P} \times \prod \mathbf{C}_i$$

and, since this last automaton is isomorphic to $\mathbf{Z}$, we finally have that

$$\mathbf{P} \times \prod f_i(\mathbf{Z}) \to \mathbf{Z}. \quad \square$$

Proposition 6.6 tell us that, in the presence of linear observability, there is no state explosion of the observers $\mathbf{C}_i$, and the number of states of the product $\mathbf{P} \times \prod \mathbf{C}_i$ is equal to the number of states of the specification $\mathbf{Z}$, thus ensuring efficient computations. In the sequel, we will want to identify conditions ensuring linear observability, or, even better, classes of processes where it can be proved that observability is equivalent to linear observability. Although very satisfying, such results cannot be expected in the general case, as shown by the following example.

Example 6.7. Consider the process $\mathbf{P}$ defined by the automaton of Fig. 10, with the specification $\mathbf{Z}$ consisting of all sequences that does not lead to the alarm state.

If the event u is unobservable, $\mathbf{Z}$ is observable by the observer $\mathbb{O}(\mathbf{Z})$ of Fig. 11. Suppose that $\mathbf{Z}$ is linearly observable by $\mathbf{C}$, then we must have $\mathbf{Z} \xrightarrow{f} \mathbf{C}$, and

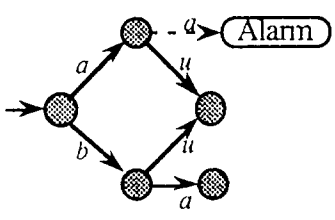

Fig. 10. The process $\mathbf{P}$.

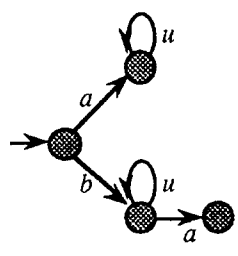

Fig. 11. The observer $\mathbb{O}(\mathbf{Z})$.

$f(\mathbf{i}\cdot a) = f(\mathbf{i}\cdot au) = f(\mathbf{i}\cdot bu) = f(\mathbf{i}\cdot b)$ since $\mathbf{C}$ is an observer. And since $f(\mathbf{i}\cdot ba)$ is defined, so is $f(\mathbf{i}\cdot a)\cdot a$, thus $\mathbf{Z} < \mathbf{P}\times\mathbf{C}$.

We have seen that if $\mathbf{Z}$ is linearly observable by $\prod C_i$, then $\mathbf{Z}\to\mathbf{C}_i$. When $\mathbf{Z}$ is a subautomaton of $\mathbf{P}$, and if $\mathbf{Z}$ is observable $\prod\mathbf{C}_i$, these morphism are sufficient to infer linear observability.

Theorem 6.8. *Let $\mathbf{Z}\hookrightarrow\mathbf{P}$ be a specification of a process $\mathbf{P}$, if $\mathbf{Z}$ is observable by $\prod\mathbf{C}_i$, and if, for each i, $\mathbf{Z}\to\mathbf{C}_i$, then $\mathbf{Z}$ is linearly observable.*

Proof. If $Z\hookrightarrow\mathbf{P}$ and $\mathbf{Z}\to\mathbf{C}_i$, we immediately get the relation $\mathbf{Z}\to\mathbf{P}\times\prod\mathbf{C}_i$. In order to establish that $\mathbf{P}\times\prod\mathbf{C}_i\to\mathbf{Z}$, consider the function f that associate to each accessible state $(\mathbf{s},\mathbf{s}_1,\ldots,\mathbf{s}_n)$ of $\mathbf{P}\times\prod\mathbf{C}_i$ the state $\mathbf{s}$. This state belongs to $\mathbf{Z}$ since if $(\mathbf{s},\mathbf{s}_1,\ldots,\mathbf{s}_n)$ is accessible by a sequence x, x is also defined in $\mathbf{Z}$ because $\mathbf{Z}\approx\mathbf{P}\times\prod\mathbf{C}_i$ by observability, and since $\mathbf{Z}$ is a subautomaton of $\mathbf{P}$, the sequence x will reach the state $\mathbf{s}$. □

7. Exchange networks

Exchanges networks are a straightforward generalization of Example 1.4, sharing features with Petri nets and vector addition systems [8]. Additional structure on the states of automata arising in this context will provide elegant and efficient ways of obtaining observers $\mathbf{C}_i$, and morphisms $\mathbf{Z}\to\mathbf{C}_i$, which are a necessary condition for linear observability.

Definition 7.1 (*Exchange networks*). An *exchange network* $\mathbf{E}$ is a graph whose nodes $\mathbb{P} = \{p_1,\ldots,p_k\}$ are called *places*, and vertices $\mathbb{C}$ are called *channels*. The source and target of each channel is given by functions:

$$s,t:\mathbb{C}\to\mathbb{P}.$$

A *configuration* is a function $C:\mathbb{P}\to\mathbb{N}$ which assign to each place p the number $C(p)$ of *tokens* in the place.

We will be interested in the various configurations obtained by moving tokens along the channels. An elementary *move* d will correspond to the transfer of a token from a place p_i to a place p_j, if

(1) there is a channel d such that $s(d) = p_i$ and $t(d) = p_j$;
(2) the place p_i is not empty, that is $C(p_i)\neq 0$.

A configuration C_1 is *accessible* from a configuration C_2 if there is a sequence of elementary moves transforming C_2 into C_1. Consider the set of all possible configurations with N tokens in a network with k places, we call this set the *simplex* $\Delta_{N,k}$.This is the set of points $(n_1,\ldots,n_k)$ of $\mathbb{N}^k$ defined by the equation $\sum n_i = N$.

We also associate to each channel $d \in \mathbb{C}$ connecting places p_i to p_j the vector

$$v_d = (x_1, \ldots, x_k)$$

whose coordinates are all 0 except, when $p_i \neq p_j$, $x_i = -1$ and $x_j = +1$.

Given an initial configuration $\mathscr{C}$ with N tokens, we can construct an associated automaton $\mathbf{A}_{\mathscr{C}}$ on the set of events $\mathbb{C}$, whose states are all accessible configurations from $\mathscr{C}$, and whose transition function is defined by $C \cdot d = C + v_d$. We say that this automaton 'lives' in the simplex $\Delta_{N,k}$. For example, if we take the network of Fig. 12, we have the representation for the associated automaton given in Fig. 13.

Note that if a channel is a loop, the move associated with the channel in a given configuration is defined if its source is nonempty (as the move h in Fig. 13), and is a loop of the automaton whenever it is defined. When dealing with exchange networks, we will always assume that all states are final.

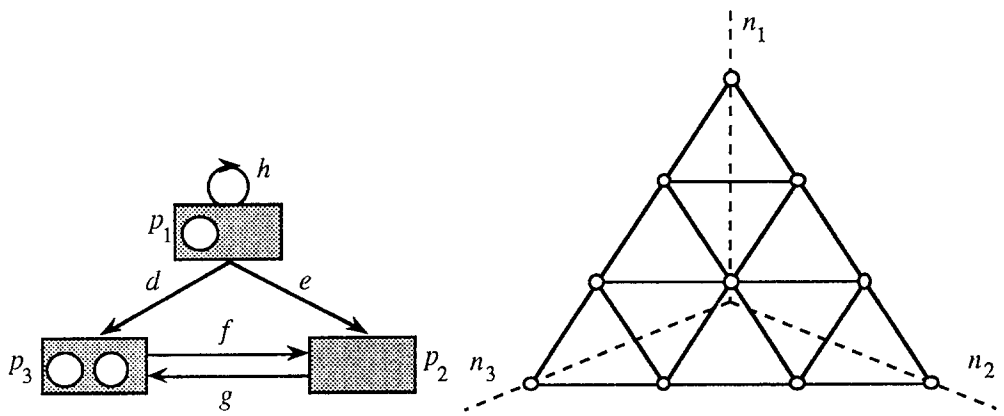

Fig. 12. Example of an exchange network and the corresponding simplex $\Delta_{3,3}$.

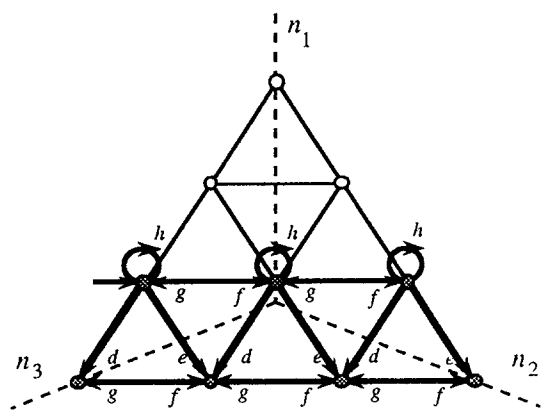

Fig. 13. An automaton living in the simplex $\Delta_{3,3}$.

Consider a network $\mathbf{E}$ with places $\mathbb{P} = \{p_1, \ldots, p_k\}$, channels $\mathbb{C}$, and source and target functions:

$$s, t : \mathbb{C} \to \mathbb{P}.$$

Let $\pi = \{\pi_1, \ldots, \pi_{k'}\}$ be a partition of $\mathbb{P}$. We can deduce from π a *derived network* $\mathbf{E}_\pi$ with places π, channels $\mathbb{C}$, and source and target functions:

$$s', t' : \mathbb{C} \to \pi$$

which assign to a channel d the class in π of $s(d)$ and of $t(d)$. The partition π also induces a linear transformation:

$$T_\pi : \mathbb{N}^k \to \mathbb{N}^{k'}$$

defined by

$$T_\pi(n_1, \ldots, n_k) = \left(\sum_{n_i \in \pi_1} n_i, \ldots, \sum_{n_i \in \pi_{k'}} n_i \right)$$

which maps the simplex $\Delta_{N,k}$ of $\mathbb{N}^k$ onto $\Delta_{N,k'}$ of $\mathbb{N}^{k'}$.

The transformations T_π define morphisms between automata living in simplex: configurations in $\mathbb{N}^k$ are mapped onto configurations in $\mathbb{N}^{k'}$ as if the places in each class π_j were glued together, and vectors associated to moves are mapped onto vectors associated to moves. We have the following:

Proposition 7.2. *Let* $\mathbf{E}$ *be a network with places* $\mathbb{P}$, *initial configuration* $\mathscr{C}$, *and associated automaton* $\mathbf{A}_{\mathscr{C}}$. *Let* π *be a partition of* $\mathbb{P}$, *and consider the derived network* $\mathbf{E}_\pi$ *with initial configuration* $T_\pi(\mathscr{C})$, *and associated automaton* $\mathbf{A}_{T_\pi(\mathscr{C})}$. *Then*

$$\mathbf{A}_{\mathscr{C}} \to \mathbf{A}_{T_\pi(\mathscr{C})}.$$

Proof. We will show that the function T_π defines a morphism. The two first properties are direct consequences of the definitions of associated automata: $\mathscr{C}$ and $T_\pi(\mathscr{C})$ are the initial states of the two automata, and all states are final.

Suppose now that d connects places p_i to p_j, and that $\pi_{i'}$ and $\pi_{j'}$ are the classes of p_i and p_j in the partition π. Let

$$\boldsymbol{v}_d = (x_1, \ldots, x_k) \quad \text{and} \quad \boldsymbol{w}_d = (y_1, \ldots, y_{k'})$$

be the vectors associated to channel d in $\mathbb{N}^k$ and $\mathbb{N}^{k'}$. We have easily that

$$T_\pi(\boldsymbol{v}_d) = \boldsymbol{w}_d.$$

If $C \cdot d$ is defined in $\mathbf{A}_{\mathscr{C}}$, then p_i is not empty in configuration C, thus $\pi_{i'}$ will not be empty in configuration $T_\pi(C)$, thus $T_\pi(C) \cdot d$ is defined. And

$$T_\pi(C) \cdot d = T_\pi(C) + \boldsymbol{w}_d = T_\pi(C) + T_\pi(\boldsymbol{v}_d) = T_\pi(C + \boldsymbol{v}_d) = T_\pi(C \cdot d). \quad \square$$

Let E be a network with places $\mathbb{P}$, initial configuration $\mathscr{C}$, and associated automaton $\mathbf{A}_{\mathscr{C}}$. Consider any subautomaton $\mathbf{Z} \hookrightarrow \mathbf{A}_{\mathscr{C}}$. Using Proposition 7.2, we can define the automaton $T_\pi(\mathbf{Z})$ which is the image of $\mathbf{Z}$ by the transformation T_π. And we have $\mathbf{Z} \to T_\pi(\mathbf{Z})$. Fig. 14 gives an example of such a morphism, with the partition $\{\{p_1, p_2\}, \{p_3\}\}$ of the network in Fig. 12.

Linear transformations based on partitions give an elegant way to define observers of specifications. The idea is that a set of unobservable events (channels) defines a natural partition of the places in a network, obtained by identifying places connected by unobservable channels (Fig. 15).

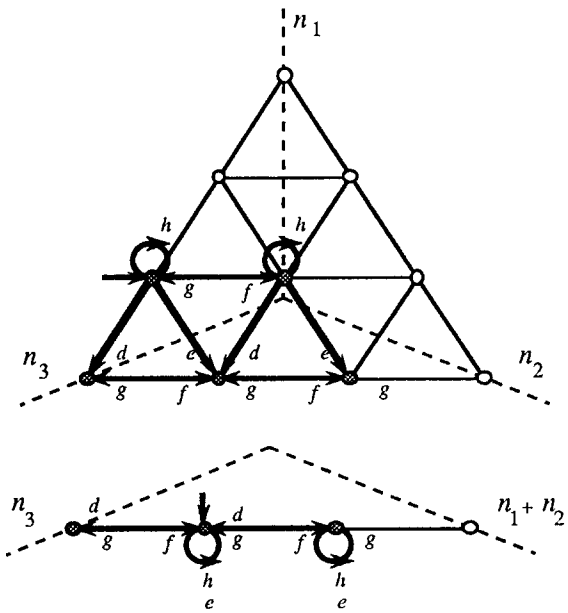

Fig. 14. $\mathbf{Z} \to T_\pi(\mathbf{Z})$.

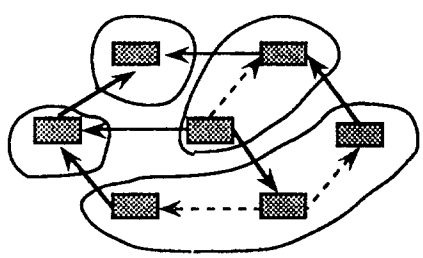

Fig. 15. Connected components of the restriction of a network to unobservable channels (dotted arrows are unobservable).

Proposition 7.3. *Let* $\mathbf{E}$ *be a network with places* $\mathbb{P}$, *channels* $\mathbb{C}$, *and* $\mathbf{Z} \hookrightarrow \mathbf{A}_{\mathscr{C}}$ *a sub-automaton of the automaton associated to* $\mathbf{E}$ *with initial configuration* $\mathscr{C}$. *For any subset of channels* $\mathbb{U} \subseteq \mathbb{C}$, *there exists a partition* π *of* $\mathbb{P}$ *such that*

(1) $T_\pi(\mathbf{Z})$ *is an observation automaton with respect to* $\mathbb{U}$,

(2) $T_\pi(C)$ *counts the number of tokens in each connected component of the restriction of* $\mathbf{E}$ *to* $\mathbb{U}$.

Proof. Consider the partition π obtained by identifying places that are in the same connected component of the restriction of the graph $\mathbf{E}$ to $\mathbb{U}$. Then, if $d \in \mathbb{U}$, $s(d)$ and $t(d)$ are in the same class of the partition, and $T_\pi(\mathbf{v}_d)$ is the null vector, establishing that $T_\pi(\mathbf{Z})$ is an observation automaton with respect to $\mathbb{U}$.

The fact that $T_\pi(C)$ counts the number of tokens in each connected component of the restriction of $\mathbf{E}$ to $\mathbb{U}$ is immediate by construction. □

8. Networks with bounded capacities

We now turn to particular class of specifications in exchange network, *networks with bounded capacities.* These specifications are described with inequalities of the form:

$$C(p_j) \leqslant \mathrm{Max}_j.$$

That is, there is a maximum (>0) number of tokens allowed in each place. A *legal* configuration is a configuration that satisfies all these inequalities, a *legal* move is a move that links two legal configuration.

In this section we prove that, if a network satisfies certain connectedness conditions, observability of this kind of specification is *equivalent* to linear observability, thus ensuring efficient computation of observers. Note that this was the case of the network in Example 1.4, and the fact that the two observers of Figs. 6 and 7 were simple was predictable.

We first establish the following lemma.

Lemma 8.1. *If a network with bounded capacities is strongly connected, and* C_1, C_2 *are two legal configurations, then* C_2 *is reachable from* C_1 *by a sequence of legal moves.*

Proof. The proof is based on the following observation. Given two places, p_1 and p_2 such that p_1 is not empty and p_2 is not full, it is possible transfer a token from p_1 to p_2 while keeping invariant the rest of the configuration. Indeed, since the network is strongly connected, there exists a path of channels connecting p_1 to p_2 (Fig. 16).

Working from the right-hand side, we find the first nonempty place p (which exists since p_1 is not empty) at the left of p_2 and transfer one chip from it to p_2. These moves are always possible and legal since the empty places between p and p_2 have capacity at

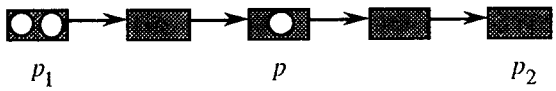

Fig. 16.

least 1. We then repeat this kind of move until one of the chip in p_1 has finally been moved to p_2.

The general argument is now easy. Assuming that two legal configurations have the same number of tokens, each 'extra' chip in a place of the first one will correspond to a 'hole' in the second one, and vice versa. □

Definition 8.2 (*Invertible sets of channels*). A sets $\mathbb{U} \subseteq \mathbb{C}$ of channels in a network is *invertible* if, in the restriction of the network to $\mathbb{U}$, every connected component is strongly connected.

The term invertible comes from the fact that if a token is moved along a channel $d \in \mathbb{U}$, there exists a sequence of channels in $\mathbb{U}$ such that the token can be moved back through them to its original position. The following lemma says that if unobservable channels are invertible for a given site, then any two configuration that are considered the same by an observer are linked by unobservable legal moves.

Lemma 8.3. *Let $\mathbb{U}$ be an invertible set of channels in a network* **E** *with bounded capacities, and π the partition that identifies places connected by channels in $\mathbb{U}$. If C_1, C_2 are two legal configurations such that*

$$T_\pi(C_1) = T_\pi(C_2)$$

then there exists a sequence of legal moves in $\mathbb{U}$ connecting C_1 to C_2.

Proof. By Proposition 7.3, $T_\pi(C)$ counts the number of tokens in each (strongly) connected component of the restriction of the network to the channels in $\mathbb{U}$. Thus if $T_\pi(C_1) = T_\pi(C_2)$, then both configurations have the same number of tokens in each component. Applying Lemma 8.1 to each of these component yield the desired sequence of legal moves in $\mathbb{U}$. □

We are now in position to prove the main theorem of this section. It states that when unobservable events are invertible, observability is equivalent to linear observability in networks with bounded capacities:

Theorem 8.4. *Let* **Z** *be a specification of a network* **E** *with bounded capacities, and n sites with invertible unobservable events $\mathbb{U}_i$. Let $T_i(\mathbf{Z})$ be the observer at site i obtained with the partition induced by $\mathbb{U}_i$. Then*
Z *is observable*
$\Leftrightarrow$ **Z** *is linearly observable*
$\Leftrightarrow$ **Z** *is linearly observable by* $\prod T_i(\mathbf{Z})$.

Proof. Clearly, if $\mathbf{Z}$ is linearly observable by $\prod T_i(\mathbf{Z})$, then it is observable. In order to show that observability implies linear observability, we will show that observability implies

$$T_i(\mathbf{Z}) \leftrightarrow \mathbb{O}_i(\mathbf{Z})$$

then, since $\mathbf{Z} \to T_i(\mathbf{Z})$, applying Theorem 6.8 with the morphisms $\mathbf{Z} \to \mathbb{O}_i(\mathbf{Z})$, we deduce that $\mathbf{Z}$ is linearly observable by $\prod \mathbb{O}_i(\mathbf{Z})$, thus by $\prod T_i(\mathbf{Z})$.

We first show that

$$\mathbb{O}_i(\mathbf{Z}) \to T_i(\mathbf{Z})$$

by defining, for any state $\mathbf{S}$ of $\mathbb{O}_i(\mathbf{Z})$, and any configuration $C \in \mathbf{S}$,

$$f(\mathbf{S}) = T_i(\mathbf{C}).$$

For this function to be well defined, we have to show that if $C_1, C_2 \in \mathbf{S}$ then $T_i(C_1) = T_i(C_2)$. This is true for the initial state $\mathbf{I}$ of $\mathbb{O}_i(\mathbf{Z})$ defined by

$$\mathbf{I} = \{\mathscr{C} \cdot u \mid \mathscr{C} \cdot u \text{ is defined and } u \in \mathbb{U}_i^*\}$$

since any configuration in this set is of the form $\mathscr{C} \cdot u$ and $T_i(\mathscr{C} \cdot u) = T_i(\mathscr{C})$. Suppose now that the statement is true for state $\mathbf{S}$. If d is any event such that $\mathbf{S} \circ d$ is defined, then $\mathbf{S} \circ d$ is either $\mathbf{S}$ or

$$\mathbf{S} \circ d = \{C \cdot du \mid C \in \mathbf{S},\ C \cdot du \text{ is defined and } u \in \mathbb{U}_i^*\}.$$

If $C_1, C_2 \in \mathbf{S} \circ d$, they can be written as

$$C_1 = C_1' \cdot du_1$$

$$C_2 = C_2' \cdot du_2.$$

where $C_1', C_2' \in \mathbf{S}$, and $T_i(C_1') = T_i(C_2')$. We then easily check that

$$T_i(C_1) = T_i(C_1' \cdot du_1) = T_i(C_1') + T_i(\mathbf{v}_d) = T_i(C_2') + T_i(\mathbf{v}_d) = T_i(C_2)$$

Thus, the function f is well defined. It induces a morphism since, if $\mathbf{S} \circ d$ is defined, then there exists a $C \in \mathbf{S}$ such that $C \cdot d$ is defined, thus $T_i(C) \cdot d$ is also defined. And we have

$$f(\mathbf{S} \circ d) = T_i(C \cdot d) = T_i(C) \cdot d = f(\mathbf{S}) \cdot d.$$

In order to show that

$$T_i(\mathbf{Z}) \to \mathbb{O}_i(\mathbf{Z})$$

we consider the function g defined as the inverse image of the transformation T_i:

$$g(T_i(C)) = T_i^{-1}(T_i(C)).$$

To prove that $T_i^{-1}(T_i(C))$ is always a state of $\mathbb{O}_i(\mathbf{Z})$, we first note that any state $\mathbf{S}$ of $\mathbb{O}_i(\mathbf{Z})$ is contained in such a set. Indeed, we know from the first part of the proof that

$C_1, C_2 \in \mathbf{S}$ implies $T_i(C_1) = T_i(C_2)$ thus

$$\mathbf{S} \subseteq T_i^{-1}(T_i(C)) \quad \text{for any } C \text{ in } \mathbf{S}.$$

Now, if $C_1 \in T_i^{-1}(T_i(C))$ and $C \in \mathbf{S}$, then $T_i(C_1) = T_i(C)$ and, by Lemma 8.3, there is a sequence of unobservable events that connects C to C_1, implying $C_1 \in \mathbf{S}$. So, for any C in $\mathbf{S}$,

$$\mathbf{S} = T_i^{-1}(T_i(C)).$$

The function g is thus well-defined. To prove that it induces a morphism we first remark that if $T_i(C) \cdot d$ is defined, then there is a configuration C' in $T_i^{-1}(T_i(C))$ such that $C' \cdot d$ is defined, implying that $T_i^{-1}(T_i(C)) \circ d$ is defined in $\mathbb{O}_i(\mathbf{Z})$. Furthermore, since $T_i(C) = T_i(C')$, we have

$$g(T_i(C)) \cdot d) = g(T_i(C') \cdot d) = g(T_i(C' \cdot d)) = T_i^{-1}(T_i(C' \cdot d))$$

and since $C' \cdot d \in T_i^{-1}(T_i(C)) \circ d$,

$$T_i^{-1}(T_i(C' \cdot d)) = T_i^{-1}(T_i(C)) \circ d = g(T_i(C)) \circ d. \quad \square$$

9. Final remarks

In this paper, we discussed several issues concerning distributed control of discrete event systems. The first sections illustrate that these problems are both easy and hard: if we skip the computational issues raised by the subset construction of Section 3, the synthesis and proof of the local controllers are simple procedures that can be readily implemented.

The second part of the paper addresses the hard part of the problem by restricting drastically the type of automata that can act as controllers. Investigations in that direction were triggered by the realization that most of the examples in the literature behaved 'well' with respect to partial observation. Linear observability seems to be a key concept in identifying classes of 'well behaved' problems. Although several problems will not be solved directly by linear constructions, it will be interesting to study problems that can be reformulated (with possibly a polynomial increase in the number of states) as linear problems. Results like Theorem 8.4 will play a major role in this study.

References

[1] A. Arnold, MEC: a system for constructing and analyzing transition systems, in: J. Sifakis, ed., *Automatic Verification of Finite State Systems*, Lecture Notes in Computer Science, Vol. 407 (Springer, Berlin, 1989) 117–132.

[2] A. Arnold, Systèmes de transitions finis et sémantique des processus communicants, *TSI* **9** (1990) 193–216.

[3] A. Bergeron, A unified approach to control problems in discrete event processes, *RAIRO Inform. Théor.* **27** (1993) 555–573.
[4] P. Ramadge and W. Wonham, The control of discrete event systems, *Proc. IEEE* **77** (1989) 81–98.
[5] K. Rudie and W. Wonham, Think globally, act locally: decentralized supervisory control, *IEEE Trans. Automat. Control* **37** (1992) 1692–1708.
[6] K. Rudie and J. Willems, The computational complexity of decentralized discrete-event control problems, IMA Preprint Series #1105, March 1993.
[7] J. Tsitsiklis, On the control of discrete-event dynamical systems. *Mat. Control. Signals Systems* **2** (1989) 95–107.
[8] L. Yong, and W. Wonham, Control of vector discrete-event systems I – the base model, *IEEE Trans. Automat. Control* **38** (1993) 1214–1227.