A parity-based architecture for decentralized discrete-event control

Laurie Ricker, Hervé Marchand

A parity-based architecture for decentralized discrete-event control

S.L. Ricker
Dept. of Mathematics & Computer Science,
Mount Allison University
Sackville, NB, Canada
Email: lricker@mta.ca

H. Marchand
INRIA Rennes-Bretagne Atlantique,
Campus de Beaulieu,
Rennes, France
Email:herve.marchand@inria.fr

***Abstract*—We introduce a new architecture that extends the class of problems that can be solved in decentralized DES control in the absence of communication. In this architecture, unlike previous architectures that use either conjunction($\wedge$) or disjunction ($\vee$) to fuse local control decisions, the fusion rule is *exclusive or* ($\oplus$). We also characterize the new architecture, where controllers take a single decision, with respect to the recently-proposed multi-decision framework of Chakib and Khoumsi. Unlike previous architectures, parity-based controllers cannot predetermine their local control decision based solely on their local observations. Instead, the local control decisions are calculated *a priori*.**

I. Introduction

We are interested in decentralized architectures for discrete-event systems (DES) that require multiple agents (in the absence of communication) to cooperatively solve control problems. Decentralized agents can issue unconditional decisions (e.g., [1], [2]) or conditional decisions (e.g., [3]–[5]). Each agent has only a partial view of the system but must make local decisions despite its uncertainty of the exact system behavior. A global decision function determines how to combine the local decisions and the outcome is a success if all the correct decisions can be taken. Recently, a multi-decision framework that allows agents to make unconditional and conditional decisions in a parallel fashion has been proposed [6]. The multi-decision framework extends the class of decentralized control problems that can be solved without communication between agents. This approach requires controllers to make a vector of control decisions based on a particular partition of the problem space. Here we introduce an algorithm to generate local decisions, leading to a new global decision function, where this new class of control problems can be solved within the classical single decision framework.

In decentralized control, unconditional decisions arise when controllers make a single definitive decision (e.g., disable) but the default control action in the presence of uncertainty is enablement[1]. Thus to synthesize a correct decentralized control policy, it must be the case that at least one controller (that has the ability to control the event) is certain when a disablement decision must be taken.

[1]We restrict our discussion here to the "enable by default" architecture of [1] and note that for the "disable by default" architecture of [2] one simply switches "enablement" for "disablement".

Conditional decisions can only be made for events that are controlled by at least two controllers [3]–[5], and for this family of decentralized languages, it must the case that when one controller is uncertain about a disablement decision, there is at least one other controller that is certain about the enablement decisions for each of the first controller's uncertainties. The first controller issues a conditional disablement decision that can be overruled by any of the other controllers that issue an enablement decision with certainty.

When neither an unconditional nor a conditional architecture can solve the decentralized problem, the last resort is to design a decentralized communication protocol (e.g., [7], [8]); however, an example from decentralized diagnosis (Example 8 in [9]), could not be diagnosed in either the unconditional or the conditional architecture, yet there was a way in which local decisions could be fused to arrive at the correct global decisions. More importantly, this solution was reached in the absence of communication. The characterization of this new architecture was left as an open problem. We define this new architecture in the context of control.

II. Background and Notation

The general behavior of the discrete-event systems (DES) examined here is described as a regular language L over a finite alphabet of events Σ. L is *prefix-closed* if $L = \overline{L}$, where the *prefix closure* of L is $\overline{L} := \{t \in \Sigma^* \mid \exists w \in \Sigma^*, s \in L \text{ such that } s = tw\}$.

L can also be represented a by finite-state automaton: $M_L = \{Q, \Sigma, T_L, q_0, F\}$ where Q is a finite set of states, $T_L \subseteq Q \times \Sigma \times Q$ is a transition relation, $q_0 \in Q$ is the initial state, and $F \subseteq Q$ is a set of final (or marked) states. The transition relation is easily extended to Σ^* and we say $L := \{s \in \Sigma^* \mid \exists q \in Q \text{ s.t. } (q_0, s, q) \in T_L\}$.

The product of two finite-state automata is defined as follows. Let $M_1 = \{Q_1, \Sigma, T_1, q_{0,1}, F_1\}$ and $M_2 = \{Q_2, \Sigma, T_2, q_{0,2}, F_2\}$. Then $M_1 \times M_2 = M = \{Q_1 \times Q_2, \Sigma, T_1 \times T_2, (q_{0,1}, q_{0,2}), F_1 \times F_2\}$.

The marked language of L is defined as $L_m := \{s \in L \mid \exists q \in F \text{ s.t. } (q_0, s, q) \in T_L\}$. The specification for the system is $K \subseteq L$ and is represented by $M_K = (Q, \Sigma, T_K, q_0, F_K)$. Furthermore, without loss of generality,

we assume that M_K is a subautomaton of M_L[2]. The language $K \subseteq L$ is m-*closed* if $K = \overline{K} \cap L_m$. In the sequel, we assume that K is m-closed.

We also define a special product that we call *synchronized composition*, denoted by $\times_{\mathcal{S}}$, which is defined as follows. Assume that we have n finite-state automata $M_1, \ldots, M_n$, where $M_j = (Q_j, \Sigma_j, T_j, q_{0,j}, F_j)$, for $j = 1, 2, \ldots, n$. Then $M = M_1 \times_{\mathcal{S}} M_2 \times_{\mathcal{S}} \ldots \times_{\mathcal{S}} M_n = (Q_{\mathcal{S}}, \Sigma_{\mathcal{S}}, T_{\mathcal{S}}, \langle q_{0,1}, q_{0,2}, \ldots, q_{0,n} \rangle, F_{\mathcal{S}})$, where $Q_{\mathcal{S}} \subseteq Q_1 \times Q_2 \times \ldots \times Q_n$; $\Sigma_{\mathcal{S}} \subseteq \Sigma_1 \times \Sigma_2 \times \ldots \times \Sigma_n$; $T_{\mathcal{S}} \subseteq Q_{\mathcal{S}} \times \Sigma_{\mathcal{S}} \times Q_{\mathcal{S}}$; and $F_{\mathcal{S}} \subseteq F_1 \times F_2 \times \ldots \times F_n$. The state set $Q_{\mathcal{S}}$ is a set of state vectors of the form $q_{\mathcal{S}} = (q_1, \ldots, q_n)$ and we will occasionally refer to the j^{th} component of $q_{\mathcal{S}}$ as $q_{\mathcal{S}}(j)$, where $j \in \{1, \ldots n\}$. We can think of these automata as running concurrently, and, thus, there is some synchronization of events in each of the alphabets. We do not yet formally define the transition relation, but, rather, define it in a specific context in the sequel (see Section V-A).

For DES control problems, a set of n controllers, denoted by $I = \{1, \ldots, n\}$, is responsible for ensuring that their collective decisions, made in response to observations of L, can distinguish between behaviour in $\overline{K}$ and in $L \setminus \overline{K}$.

To formalize partial observation problems for DES, Σ is partitioned into observable events Σ_o and unobservable events Σ_{uo}. The observable events for each agent $i \in I$ are denoted by $\Sigma_{o,i}$, where $\Sigma_o = \cup_{i \in I} \Sigma_{o,i}$. The natural projection $\pi_i : \Sigma^* \to \Sigma_{o,i}^*$ defines the observations of an agent $i \in I$ by removing all occurrences of events in $\Sigma \backslash \Sigma_{o,i}$ from a sequence in Σ^*. The inverse projection $\pi_i^{-1} : \Sigma_{o,i}^* \to 2^{\Sigma^*}$ captures all the sequences s that produce the same natural projection for agent i. Formally, $\pi_i^{-1}(t) = \{s \in \Sigma^* \mid \pi_i(s) = t\}$, for $i \in I$, and can easily be extended to languages. We will use the notation $[\![t]\!]_i$ whenever we refer to $\pi_i^{-1}\pi_i(t)$.

Control problems are characterized by the partition of Σ into a set of controllable events Σ_c (which can be prevented from happening) and a set of uncontrollable events Σ_{uc} (which must always be allowed to occur). A language K is *controllable* wrt L if $\overline{K}\Sigma_{uc} \cap L \subseteq \overline{K}$. Each controller $i \in I$ has a set of events that it controls $\Sigma_{c,i}$, where $\Sigma_c = \cup_{i \in I} \Sigma_{c,i}$. We overload our notation for I and define the set of supervisors for which σ is controllable: $I_c(\sigma) = \{i \in I \mid \sigma \in \Sigma_{c,i}\}$.

For the decentralized architectures that we are interested in, each agent makes its own local control decisions about the system behavior based on its partial observations. We assume a finite set of local decisions $\mathsf{LD} = \{0, 1\}$. Let the local decision function be $h_i : \Sigma_{o,i}^* \to \mathsf{LD}^{\Sigma}$, for $i \in I$.

To correctly solve the decentralized DES problem of interest, a global decision $\mathsf{GD} = \{0, 1\}$ is made by combining or fusing the local decisions of each agent in such a way that the correct decision is taken overall. Let the global decision function be $H : \mathsf{LD}^{\Sigma^n} \to \mathsf{GD}^{\Sigma}$. The decentralized architecture that we use is illustrated in Fig. 1.

[2]If this is not the case, then it can be easily obtained by taking the product $M_K \times M_L$ as the new automaton, where marked states in this new automaton are the states which are marked in the first component.

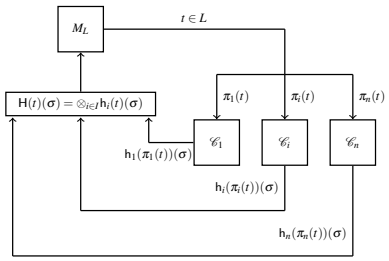

Fig. 1. Decentralized DES architecture, where decentralized controllers $\mathcal{C}_i$ (for $i \in I$) make control decisions $\mathsf{h}_i(t)$ that are combined by a fusion rule (generically denoted here by $\otimes$) to produce a global control decision $\mathsf{H}(t)$ to either enable or disable events after observing sequence t generated by M_L.

In the work that follows, we have reversed the classical DES assignment of enable = 1 and disable = 0, which has led to the use of conjunction as the fusion rule in unconditional architecture [1] and disjunction in the conditional architecture [3]. These assignments do not lend themselves easily to a straightforward explanation of a parity-based architecture. In the epistemic logic formulation of the decentralized control problem for the unconditional architecture [10], a disablement control decision was issued locally if an agent "knew" or was certain of the truth about disabling. Thus, we will assume that 0 corresponds to enable, 1 corresponds to disable and, therefore, the fusion rule for the unconditional architecture is disjunction, while conjunction is now used for the conditional architecture. Note that we still assume that the unconditional architecture is "enable by default" and the conditional architecture is "disable by default".

In the unconditional architecture, for all controllable events $\sigma \in \Sigma_{c,i}$, we want to find local decision functions $h_1, \ldots, h_n$ such that the global decision function H satisfies:

$$\begin{aligned} &(\forall u \in \overline{K})(\forall \sigma \in \Sigma) u\sigma \in \overline{K} \Rightarrow \\ &\quad H(\pi(u))(\sigma) = \bigvee_{i \in I} h_i(\pi_i(u))(\sigma) = 0 \wedge \\ &(\forall u \in \overline{K})(\forall \sigma \in \Sigma_c) u\sigma \in L \setminus \overline{K} \Rightarrow \\ &\quad H(\pi(u))(\sigma) = \bigvee_{i \in I} h_i(\pi_i(u))(\sigma) = 1. \end{aligned} \tag{1}$$

We can find such local decision functions (e.g., synthesize decentralized controllers) for L in the unconditional architecture iff K is controllable, *m*-closed, and *unconditionally co-observable*:

$$\begin{aligned} &(\forall t \in \overline{K})(\forall \sigma \in \Sigma_c) t\sigma \in L \setminus \overline{K} \Rightarrow \\ &\quad (\exists i \in I_c(\sigma))[\![t]\!]_i \sigma \cap \overline{K} = \emptyset. \end{aligned} \tag{2}$$

In such a circumstance, controller i takes the local decision $h_i(\pi_i(s))(\sigma) = 1$ when $\pi_i^{-1}\pi_i(s)\sigma \cap \overline{K} = \emptyset$ and $\sigma \in \Sigma_{c,i}$, i.e., it is certain of the disablement decision for event σ that it is authorized to take. Otherwise, controller i takes the local decision $h_i(\pi_i(s))(\sigma) = 0$. When $I = \{1\}$, K is said to be *observable* [11] if it satisfies (2).

In the conditional architecture, we want to find local decision functions $h_1, \ldots, h_n$ such that the global decision

function H satisfies:

$$\begin{aligned}&(\forall u \in \overline{K})(\forall \sigma \in \Sigma) u\sigma \in \overline{K} \Rightarrow \\ &\quad H(\pi(u))(\sigma) = \bigwedge_{i \in I} h_i(\pi_i(u))(\sigma) = \mathsf{0} \wedge \\ &(\forall u \in \overline{K})(\forall \sigma \in \Sigma_c) u\sigma \in L \setminus \overline{K} \Rightarrow \\ &\quad H(\pi(u))(\sigma) = \bigwedge_{i \in I} h_i(\pi_i(u))(\sigma) = \mathsf{1}.\end{aligned} \tag{3}$$

We can find such local decision functions for L in the conditional architecture iff K is controllable, *m*-closed, and *conditionally co-observable*:

$$\begin{aligned}&(\forall t \in \overline{K})(\forall \sigma \in \Sigma_c) t\sigma \in L \setminus \overline{K} \Rightarrow \\ &(\exists i \in I_c(\sigma))(\forall t'\sigma \in [\![t]\!]_i\sigma \cap \overline{K})(\exists j \in I_c(\sigma))[\![t']\!]_j\sigma \cap L \subseteq \overline{K}.\end{aligned}$$

When K is conditionally co-observable, controller i takes a local decision $h_i(\pi_i(s))(\sigma) = \mathsf{0}$ when $[\![s]\!]_i\sigma \cap L \subseteq \overline{K}$ and $\sigma \in \Sigma_{c,i}$, i.e., controller i is certain of the enablement decision for event σ for which it has authority to take. In addition, $h_i(\pi_i(s))(\sigma) = \mathsf{0}$ when $\sigma \in \Sigma_c \setminus \Sigma_{c,i}$. A local decision $h_i(\pi_i(s))(\sigma) = \mathsf{1}$ in taken all other circumstances. This local decision for controller i includes circumstances when there is uncertainty that $s\sigma \in L \setminus \overline{K}$ and $\sigma \in \Sigma_{c,i}$. This is interpreted as follows: although controller i is not certain about the disablement decision for σ, controller i can infer that there is another controller j that is certain about the enablement decision for σ for all $s'\sigma \in [\![s]\!]_i\sigma$ such that $s\sigma \in \overline{K}$. In this case, controller j can correctly override controller i's decision with $h_j(\pi_j(s))(\sigma) = \mathsf{0}$ decision if $s\sigma \in \overline{K}$. In the context of conditional control, the local decisions are interpreted as enable = 0 and conditional disable = 1.

We denote the language generated by M_L under the control of the global decision function H by $\mathcal{L}(H/M_L)$ and $\mathcal{L}_m(H/M_L)$, where H satisfies either (1) or (3) depending on the architecture:

$$\begin{cases} \varepsilon \in \mathcal{L}(H/M_L); \\ (\forall t \in \mathcal{L}(H/M_L))(\forall \sigma \in \Sigma) \\ \qquad t\sigma \in L \wedge H(\pi(t))(\sigma) = 0 \Leftrightarrow t\sigma \in \mathcal{L}(H/M_L). \end{cases}$$

and $\mathcal{L}_m(H/M_L) = \mathcal{L}(H/M_L) \cap L_m$.

III. Classifying decentralized languages

In [3], the classification of decentralized languages (again, just referring to the "enable by default" unconditional architecture) was as follows:

$$L_{ucoobs}(L,K) \subseteq L_{ccoobs}(L,K) \subseteq L_{obs}(L,K),$$

where $L_{ucoobs}(L,K) = \{K' \subseteq K \mid K'$ is unconditionally coobservable wrt $L, \Sigma_{o,1}, \ldots, \Sigma_{o,n}, \Sigma_c\}$, $L_{ccoobs}(L,K) = \{K' \subseteq K \mid K'$ is conditionally coobservable wrt $L, \Sigma_{o,1}, \ldots, \Sigma_{o,n}, \Sigma_c\}$ and $L_{obs}(L,K) = \{K' \subseteq K \mid K'$ is observable wrt $L, \Sigma_o, \Sigma_c\}$.

There is one additional language family of K that is worth considering.

Definition 1: A language $K \subseteq L$ is distributed observable *w.r.t. L, $\Sigma_{o,1}, \ldots, \Sigma_{o,n}$ and $\Sigma_{c,1}, \ldots, \Sigma_{c,n}$ if*

$$\begin{aligned}&(\forall t \in \overline{K})(\forall \sigma \in \Sigma_c) t\sigma \in L \setminus \overline{K} \Rightarrow \\ &\quad \textstyle\bigcap_{i \in I_c(\sigma)} [\![t]\!]_i\sigma \cap \overline{K} = \emptyset.\end{aligned}$$

This class of languages was first examined in [10]. When a language is distributed observable, if the controllers combined or pooled their individual estimates then they could all definitively discern that σ must be disabled.

Thus, we now have the family of distributed-observable languages to consider: $L_{dobs}(L,K) = \{K' \subseteq K \mid K'$ is distributed observable wrt $L, \Sigma_o, \Sigma_c\}$.

Lemma 1: $L_{ccoobs}(L,K) \subseteq L_{dobs}(L,K)$

Lemma 2: $L_{dobs}(L,K) \subseteq L_{obs}(L,K)$

We update the classification as follows:

$$L_{ucoobs}(L,K) \subseteq L_{ccoobs}(L,K) \subseteq L_{dobs}(L,K) \subseteq L_{obs}(L,K).$$

We conclude this section with an example of a distributed-observable language that is neither unconditionally nor conditionally co-observable.

Example 1: Let M_L and M_K be given as in Fig. 2, which is Example 8 from [9] adapted for control. Here $\Sigma_{o,1} = \{\mathsf{a}_1, \mathsf{a}_2, \sigma\}$ and $\Sigma_{o,2} = \{\mathsf{b}_1, \mathsf{b}_2, \sigma\}$ while $I_c(\sigma) = \{1,2\}$.

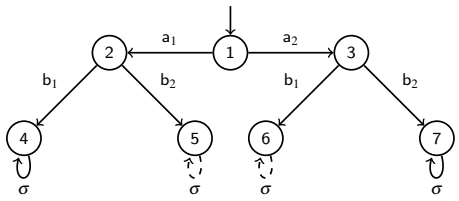

Fig. 2. Joint M_L (all transitions) and M_K (only solid transitions) that generates a language that is distributed observable but not unconditionally or conditionally co-observable.

TABLE I
Truth table for XOR ($\oplus$).

p	q	p $\oplus$ q
0	0	0
0	1	1
1	0	1
1	1	0

It is straightforward to assert that K is neither unconditionally nor conditionally co-observable; however, K is distributed-observable:

$$\begin{aligned}[\![\mathsf{a}_1\mathsf{b}_2]\!]_1\sigma \cap [\![\mathsf{a}_1\mathsf{b}_2]\!]_2\sigma \cap \overline{K} = \emptyset, \\ [\![\mathsf{a}_2\mathsf{b}_1]\!]_1\sigma \cap [\![\mathsf{a}_2\mathsf{b}_2]\!]_2\sigma \cap \overline{K} = \emptyset.\end{aligned}$$

A global decision function that would correctly control the example in Fig. 2, where 1 corresponds to disable and 0 corresponds to enable, follows.

$$\begin{aligned}H(\pi(\mathsf{a}_1\mathsf{b}_2))(\sigma) = H(\pi(\mathsf{a}_2\mathsf{b}_1))(\sigma) = \mathsf{1} \\ H(\pi(\mathsf{a}_1\mathsf{b}_1))(\sigma) = H(\pi(\mathsf{a}_2\mathsf{b}_2))(\sigma) = \mathsf{0}\end{aligned}$$

In [9], the following local decision functions (updated for a control problem) are proposed as a possible solution:

$$\begin{aligned} &h_1(\pi_1(\mathsf{a_1b_2}))(\sigma) = \mathsf{1} \text{ and } h_2(\pi_2(\mathsf{a_1b_2}))(\sigma) = \mathsf{0}; \\ &h_1(\pi_1(\mathsf{a_2b_1}))(\sigma) = \mathsf{0} \text{ and } h_2(\pi_2(\mathsf{a_2b_1}))(\sigma) = \mathsf{1}; \\ &h_1(\pi_1(\mathsf{a_1b_1}))(\sigma) = \mathsf{1} \text{ and } h_2(\pi_2(\mathsf{a_1b_1}))(\sigma) = \mathsf{1}; \\ &h_1(\pi_1(\mathsf{a_2b_2}))(\sigma) = \mathsf{0} \text{ and } h_2(\pi_2(\mathsf{a_2b_2}))(\sigma) = \mathsf{0}; \end{aligned} \tag{4}$$

Since K is neither unconditionally nor conditionally co-observable, we cannot use binary operations $\vee$ or $\wedge$ to combine the local decision functions; however, if we combine the local decision functions using exclusive or (denoted by $\oplus$), with reference to Table I, then we do achieve the anticipated values for the global function noted above. In the next section, we characterize when $\oplus$ can be used as the fusion rule to achieve correct control decisions. $\diamond$

IV. DECENTRALIZED CONTROL WITH A PARITY-BASED ARCHITECTURE

Motivated by the possibility of a new class of decentralized problems that are neither unconditionally nor conditionally co-observable but can be solved without introducing communication between controllers, as first suggested in [9], we now define new local and global decision functions for this architecture. In particular, this new architecture gives rise to a new fusion rule.

Definition 2: Given n local decisions functions $h_i : \Sigma_{o,i}^* \to \mathsf{LD}^{\Sigma}$ and the global decision function $H : \Sigma_o \to \mathsf{LD}^{\Sigma}$ such that $H(\mu)(\sigma) = \oplus_{i\in I} h_i(\pi_i(\mu))$ for $\mu \in \Sigma_o^*$ and $\sigma \in \Sigma$. Then H forms a valid *global decision function whenever*

$$\begin{aligned} &(\forall t \in \overline{K})(\forall \sigma \in \Sigma) t\sigma \in K \Rightarrow H(\pi(t))(\sigma) = \mathsf{0} \text{ and} \\ &(\forall t \in \overline{K})(\forall \sigma \in \Sigma_{uc}) H(\pi(t))(\sigma) = \mathsf{0}, \\ &(\forall t \in \overline{K})(\forall \sigma \in \Sigma_c) t\sigma \in L \setminus \overline{K} \Rightarrow H(\pi(t))(\sigma) = \mathsf{1}. \end{aligned}$$

We want to solve the following problem.

Problem 1: Given M_L that recognizes L, K (such that $K \subseteq L$) and $\overline{L_m} = L$, a set of decentralized agents $I = \{1, \ldots, n\}$, n sets of observable events $\Sigma_{o,1}, \ldots, \Sigma_{o,n}$, n sets of controllable events $\Sigma_{c,1}, \ldots, \Sigma_{c,n}$, find n local decision functions $h_1, \ldots, h_n$, where $h_i : \Sigma_{o,i}^* \to \{\mathsf{0}, \mathsf{1}\}^{\Sigma}$, that form a valid global decision function H such that $\mathcal{L}_{\mathsf{m}}(H/M_L) = K$ and $\mathcal{L}(H/M_L) = \overline{K}$ where $\mathsf{0}$ corresponds to enable and $\mathsf{1}$ corresponds to disable. $\square$

We now show how one can can assign local control decisions in such a way that when they are fused using the XOR ($\oplus$) operator, the correct global control decision can be reached. To ensure that $\oplus$ produces the anticipated result for the global decision function, we must adhere to the parity rules for $\oplus$. In the general case, a global disable decision can only be reached if an odd number of controllers take a local decision of disable. This is a significant difference from the way in which local control decisions are assigned when K is unconditionally or conditionally co-observable. Thus, when assigning local control decisions, it may be the case that although controller i's local view of a sequence $t \in \overline{K}$ indicates σ should be disabled, resulting in $h_i(\pi_i(t))(\sigma) = 1$, to meet parity for the $\oplus$ fusion rule, it must issue $h_i(\pi_i(t))(\sigma) = 0$. Thus, in the parity-based architecture, the local control policies are all calculated off-line.

The decision as to which controller(s) will take a disablement decision to meet parity gives rise to a system of linear equations which can subsequently be associated with a particular class of constraint satisfaction problems. Because we want to use XOR as our fusion rule, we are interested in XOR-satisfiability (XORSAT) problems (summarized in [12]).

We can use standard techniques from linear algebra to determine whether a solution exists. An instance of an XORSAT problem consists of a set of v equations over w variables: $A\underline{y} = \underline{b}$, where A is an v by w matrix whose entries correspond to the local control decisions for each controller for a given sequence t wrt $\sigma \in \Sigma_c$, and $\underline{b}$ is a vector containing the anticipated global control decision for σ wrt t. According to well-known results from linear algebra, we have a solution for the XORSAT problem as follows:

1) if $rank(A) = v$ then $A\underline{y} = \underline{b}$ has a solution for any choice of $\underline{b}$;
2) if $rank(A) < v$, then we have a solution iff $\underline{b}$ is in the image of A.

We can decide whether or not the system has a solution in polynomial (in the number of variables) time.

Our XORSAT instance consists of equations corresponding to the local control decisions that are involved in the global control decisions that must be taken. The actual system of linear equations can be easily derived from the set of constraints that are described in Definition 2

Definition 3: Given two languages K and L with $K \subseteq L$, $\forall t \in \overline{K}$ and $\forall \sigma \in \Sigma$, we have to add the following constraints to the equation system:

1) if $\sigma \in \Sigma_{uc}$, then $\oplus_{i\in I} h_i(\pi_i(t))(\sigma) = 0$, where $h_i(\pi_i(t))(\sigma)$ is represented as a variable in $\underline{y}$;
2) if $t\sigma \in \overline{K}$, then $\oplus_{i\in I} h_i(\pi_i(t))(\sigma) = 0$;
3) if $t\sigma \in L \setminus \overline{K}$, then $\oplus_{i\in I} h_i(\pi_i(t))(\sigma) = 1$.

These equations together form a system of linear equations $A\underline{y} = \underline{b}$.

Note, though, that this system of linear equation may be infinite. For ease of exposition, in the sequel we assume it is finite and we refer to Section V to see how one can consider only a finite set of representative sequences to obtain a finite linear system.

Not surprisingly, we have the following result:

Proposition 1: Given non-empty languages K, L, L_{m} such that $K \subseteq L$ and $\overline{L_m} = L$, if the system of linear equations $A\underline{y} = \underline{b}$ defined as in Definition 3 has a solution, then Problem 1 has a solution.

For Example 1, we follow Definition 3 to arrive at the set of equations noted below. In particular, $H(\pi(\mathsf{a_1b_2}))(\sigma) = h_1(\pi_1(\mathsf{a_1b_2}))(\sigma) \oplus h_2(\pi_2(\mathsf{a_1b_2}))(\sigma)$. We know that the global decision $H(\pi(\mathsf{a_1b_2}))(\sigma)$ must be 1, but we need to find the value for the local decisions such that this equation is satisfied. We represent each distinct local control decision as a variable: y_1 represents $h_1(\mathsf{a}_1)(\sigma)$, y_2 represents $h_1(\mathsf{a}_2)(\sigma)$, y_3 represents $h_2(\mathsf{b}_1)(\sigma)$, y_4 represents $h_1(\mathsf{b}_2)(\sigma)$. We have

the following set of equations

$$\begin{aligned} y_1 \oplus y_4 &= 1 \quad [H(\mathsf{a}_1\mathsf{b}_2)(\sigma)], \\ y_1 \oplus y_3 &= 0 \quad [H(\mathsf{a}_1\mathsf{b}_1)(\sigma)], \\ y_2 \oplus y_4 &= 0 \quad [H(\mathsf{a}_2\mathsf{b}_2)(\sigma)], \\ y_2 \oplus y_3 &= 1 \quad [H(\mathsf{a}_2\mathsf{b}_1)(\sigma)]. \end{aligned} \tag{5}$$

Our system of equations is rewritten in the form $A\underline{y} = \underline{b}$:

$$\begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Matrix A has rank 3, which means that we must verify that $\underline{b}$ is in the image of A. The latter is straightforward to verify with any online linear algebra tool. One solution for this system is the one previously noted in (4) in Section III.

Next we introduce a necessary condition under which Problem 1 has a solution

Theorem 1: Given non-empty languages K, L, L_m such that $K \subseteq L_{\mathsf{m}}$ and $\overline{L_m} = L$. There exist n local decision functions $h_1, \ldots, h_n$ that form a valid global decision function H such that $\mathcal{L}_{\mathsf{m}}(H/M_L) = K$ and $\mathcal{L}(H/M_L) = \overline{K}$ if K is controllable, m-closed, distributed observable.

V. Verifying distributed observability and solving XORSAT

Prior to constructing a system of equations for XORSAT, we want to first verify that our system is distributed-observable. We reuse a variation of a finite-state structure $\mathcal{U}$ from [13] on which we verify distributed observability. Secondly, we want to avoid the construction of a possibly infinite system of equations $A\underline{y} = \underline{b}$ for XORSAT. Conveniently, we also use $\mathcal{U}$ to construct a finite system of equations.

A. The $\mathcal{U}$ structure

We build an automaton $\mathcal{U}$ using the synchronous composition defined earlier:

$$\mathcal{U} = (X, \mathbb{A}, T^{\mathcal{U}}, x_0) = M_{L_0} \times_{\mathcal{S}} M_{L_1} \times_{\mathcal{S}} \ldots \times_{\mathcal{S}} M_{L_n},$$

where M_{L_i} is simply a copy of M_L for each controller $i \in \{0\} \cup I$.

The alphabet of $\mathcal{U}$ is a set of vector labels [14]. We have two types of labels, corresponding to the occurrence and observation of an event in Σ_o (and thus $\Sigma_{o,i}$) and events that are not officially observed, i.e., events in $\Sigma_{uo,i}$ and $\Sigma \setminus \Sigma_o$. Let $I_o(\sigma) = \{i \in I \mid \sigma \in \Sigma_{o,i}\}$. We build the following set of labels for $\mathbb{A}$: for all $\sigma \in \Sigma_o$, $\ell(0) = \sigma$ and for all $i \in I_o(\sigma)$, $\ell(i) = \sigma$, and for all $j \in I \setminus I_o(\sigma)$, $\ell(j) = \varepsilon$; for all $\sigma \in \Sigma \setminus \Sigma_{o,i}$, $\ell(i) = \sigma$ and for all $j \neq i \in I$, $\ell(j) = \varepsilon$; for all $\sigma \in \Sigma \setminus \Sigma_o$, $\ell(0) = \sigma$ and for all $i \in I$, $\ell(i) = \varepsilon$. We also define the empty vector label $\langle \varepsilon, \ldots, \varepsilon \rangle$, where for all $i \in I \cup \{0\}$, $\ell(i) = \varepsilon$.

A transition $(x, \ell, x') \in T^{\mathcal{U}}$, where $x = (q, q_1, \ldots, q_n)$ and $x' = (q', q_1', \ldots, q_n')$ with label $\ell = \langle \ell(0), \ldots, \ell(n) \rangle$, iff $(q, \ell(0), q') \in T_L$, and for all $i \in I$, $(q_i, \ell(i), q_i') \in T_{L_i}$, where T_{L_i} is the transition system for M_{L_i}. We also assume that the transitions $T_L = T_{\mathsf{Good}} \cup T_{\mathsf{Bad}}$, where $T_{\mathsf{Good}} = T_K$ and $T_{\mathsf{Bad}} = T_L \setminus T_K$[3]

The $\mathcal{U}$ structure for Example 1 is shown in Fig. 3.

B. Verifying Distributed Observability

To ensure that we have a finite-state structure on which to verify distributed observability, we describe state equivalences using the following equivalence relation on the state set of $\mathcal{U}$:

Definition 4: (Adapted from [15].) *Let $\Omega = \mathbb{A} \cup \{\langle \varepsilon, \ldots, \varepsilon \rangle\}$. The relation $\sim_\Omega$ is the least equivalence on the set X such that the following two axioms are satisfied*

$$\begin{aligned} \textit{A}1: &\quad (x, \ell, x') \in T^{\mathcal{U}} \textit{ and } \ell \notin \Omega \Rightarrow x \sim_\Omega x'; \\ \textit{A}2: &\quad (x_1, \ell', x_1') \in T^{\mathcal{U}} \textit{ and} \\ &\qquad (x_2, \ell', x_2') \in T^{\mathcal{U}} \textit{ and } x_1 \sim_\Omega x_2 \Rightarrow x_1' \sim_\Omega x_2'. \end{aligned}$$

*The associated **crushed automaton** is $\kappa_\Omega[\mathcal{U}] = (X_{/\sim_\Omega}, [x_0]_\Omega, \Omega, T^{\kappa_\Omega})$, where $[x_0]_\Omega$ is the equivalence class of the initial state x_0; and $([x]_\Omega, \ell, [x']_\Omega) \in T^{\kappa_\Omega}$ if there are states x_1 and x_1' such that $x_1 \sim_\Omega x$, $x_1' \sim_\Omega x'$ and $(x_1, \ell, x_1') \in T^{\mathcal{U}}$.*

We construct a crushed automaton of $\mathcal{U}$ for each controller $i \in I$ using $\Omega_i = \{\ell \in \mathbb{A} \mid \ell(0) = \ell(i) \in \Sigma_{o,i}\}$. Note that a crushed automaton is based on state equivalences, not observation equivalences, as is the case with the construction of observers for partial observation problems. A crushed automaton of $\mathcal{U}$ can be calculated with a computational complexity of $O(|X| + |T^{\mathcal{U}}|)$.

To ensure that the crushed automata $\kappa_{\Omega_i}(\mathcal{U})$ are correctly constructed, $\kappa_{\Omega_i}(\mathcal{U})$ must satisfy the following axioms related to Theorem 2.3 in [15]:

$$\begin{aligned} &A3: \forall x, x' \in X \Rightarrow \exists \Omega_i \subset \mathbb{A} \text{ such that } x \not\sim_{\Omega_i} x'; \\ &A4: \text{if } x \xrightarrow{\ell} x' \notin T^{\mathcal{U}} \Rightarrow \exists \Omega_i \subset \mathbb{A} \text{ and } \ell \in \mathbb{A} \text{ such that} \\ &\qquad\qquad ([x]_{\Omega_i}, \ell, [x']_{\Omega_i}) \notin T^{\kappa_{\Omega_i}}. \end{aligned}$$

The first axiom says that every pair of distinct states in $\mathcal{U}$ is distinguished by one of the crushes. The second axiom states that when a label ℓ is not defined at state $x \in X$, then all states in the equivalence class induced by Ω_i containing x must also not have an outgoing transition of ℓ. It is straightforward to show that $\kappa_{\Omega_i}(\mathcal{U})$ satisfies these axioms.

We introduce Algorithm 1 to verify distributed observability. At the beginning of Algorithm 1, we perform a classification of states $x \in X$ of $\mathcal{U}$ based on whether or not $\sigma \in \Sigma_c$ should be disabled or enabled at $x(0)$ in M_K. For our example, we have only to be concerned with event σ. States in $\mathsf{G}(\sigma)$ are outlined in green in Fig. 3 whereas those in $\mathsf{B}(\sigma)$ are outlined in red. The crushed automaton, for each $i \in I$, is shown in Fig. 4.

Proposition 2: Algorithm 1 correctly verifies distributed observability.

[3]Because M_K is a sub-automaton of M_L, this implies that $\forall t, t' \in K \cap L$ such that $(q_0, t, q) \in T_L$ for some $q \in Q$, $\forall \sigma \in \Sigma_{uc}$, if $t\sigma \in L \setminus K$, then $t'\sigma \in L \setminus K$.

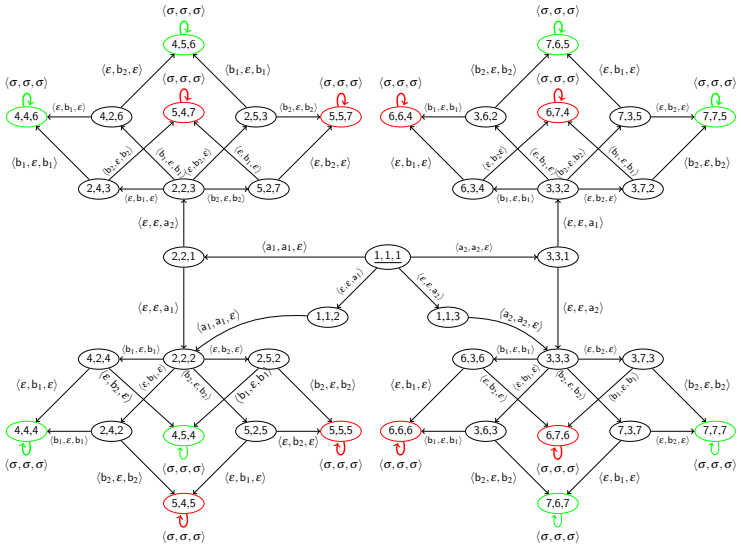

Fig. 3. $\mathcal{U}$ for the example in Fig. 2. Initial state is underlined. Red transitions are those for which the global control decision for σ at the exit state must be disable, whereas at those in green, the global decision must be enable. Therefore, the language-based control problem can be reduced to a state-based control problem, where the transitions of T_L that are to be removed by control are the ones that do not belong to T_K.

Algorithm 1 Verification of distributed observability

```
distObs ← True
for σ ∈ Σ_c do
    B(σ) ← {x ∈ X | ∃x' ∈ X s.t.(x(0), σ, x'(0)) ∈ T_Bad}
    G(σ) ← {x ∈ X | ∃x' ∈ X s.t.(x(0), σ, x'(0)) ∈ T_Good}
end for
for σ ∈ Σ_uc do
    G(σ) ← {x ∈ X | ∃x' ∈ X s.t.(x(0), σ, x'(0)) ∈ T_Good
end for
for i ∈ I do
    Construct κ_{Ω_i}(U), thus yielding [x]_{Ω_i}, for all x ∈ X
end for
for σ ∈ Σ_c do
    for x ∈ X s.t. ∃x' ∈ X and (x, ℓ, x') ∈ T^U and ℓ(0) = σ do
        if (∩_{i∈I_c(σ)} [x]_{Ω_i} ∩ G(σ) ≠ ∅) and (∩_{i∈I_c(σ)} [x]_{Ω_i} ∩ B(σ) ≠ ∅) then
            distObs ← False; Exit
        end if
    end for
end for                                   ▷ Complexity O(| X || T^U |)
```

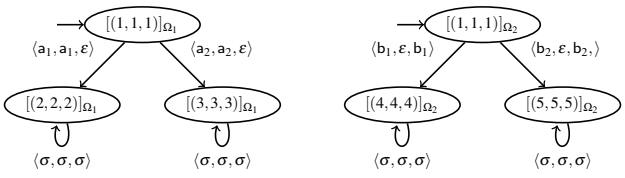

Fig. 4. $\kappa_{\Omega_1}(\mathcal{U})$ [left] and $\kappa_{\Omega_2}(\mathcal{U})$ [right].

C. A finite system of equations for XORSAT

In Algorithm 2, we are constructing the system of equations $A\underline{y} = \underline{b}$ for XORSAT. We will only continue from Algorithm 1 if the system is distributed observable. We continue to use $\mathsf{B}(\sigma)$ and $\mathsf{G}(\sigma)$, as calculated in Algorithm 1, to guide the construction of $\underline{b}$. We assign a variable in $\underline{y}$ based on $[x]_{\Omega_i}$, for $i \in I$. Since we are constructing equations based on decisions that must be taken at state-equivalences of X, we have a finite number of equations.

Below are the equations that are constructed for our example using Algorithm 2, which correspond to the system of equations previously noted in (5).

$$
\begin{aligned}
&H([(4,4,4)]_{\Omega_0})(\sigma) = \\
&\quad h_1([(4,4,4)]_{\Omega_1})(\sigma) \oplus h_2([(4,4,4)]_{\Omega_2})(\sigma) = \mathsf{0}, \\
&H([(5,5,5)]_{\Omega_0})(\sigma) = \\
&\quad h_1([(5,5,5)]_{\Omega_1})(\sigma) \oplus h_2([(5,5,5)]_{\Omega_2})(\sigma) = \mathsf{1}, \\
&H([(6,6,6)]_{\Omega_0})(\sigma) = \\
&\quad h_1([(6,6,6)]_{\Omega_1})(\sigma) \oplus h_2([(6,6,6)]_{\Omega_2})(\sigma) = \mathsf{1}, \\
&H([(7,7,7)]_{\Omega_0})(\sigma) = \\
&\quad h_1([(7,7,7)]_{\Omega_1})(\sigma) \oplus h_2([(7,7,7)]_{\Omega_2})(\sigma) = \mathsf{0}.
\end{aligned}
$$

Note that the global control decision is the same for each sequence that leads to $[x]_{\Omega}$, and also all local control decisions are the same for all the partial observations that lead to $[x]_{\Omega_i}$.

Proposition 3: Algorithm 2 produces a finite system of equations that can be rewritten in the form $A\underline{y} = \underline{b}$ for purposes of solving XORSAT.

VI. Relationship to multi-decision framework

In the multi-decision framework of [6], one examines the following two sets of sequences for every $\sigma \in \Sigma_c$, :

$$\mathbb{E}_\sigma = \{t \in \overline{K} \mid t\sigma \in \overline{K}\} \text{ and } \mathbb{D}_\sigma = \{t \in \overline{K} \mid t\sigma \in L \setminus \overline{K}\}.$$

Algorithm 2 Construct finite set of equations for XORSAT

for $\sigma \in \Sigma_c$ **do**
  for $x \in \mathsf{B}(\sigma)$ **do**
  ▷ Note that each distinct $h_i[x]_{\Omega_i}$ gives rise to a different variable in y.
    $H([x]_{\Omega_0})(\sigma) = \oplus_{i \in I_c(\sigma)} h_i([x]_{\Omega_i})(\sigma) = \mathsf{1}$
  end for
  for $x \in \mathsf{G}(\sigma)$ **do**
    $H([x]_{\Omega_0})(\sigma) = \oplus_{i \in I_c(\sigma)} h_i([x]_{\Omega_i})(\sigma) = \mathsf{0}$
  end for
end for
for $\sigma \in \Sigma_{uc}$ **do**
  for $x \in \mathsf{G}(\sigma)$ **do**
    $H([x]_{\Omega_0})(\sigma) = \oplus_{i \in I} h_i([x]_{\Omega_i})(\sigma) = \mathsf{0}$
  end for
end for ▷ Complexity $O(\mid \Sigma \mid\mid X \mid)$

The idea is that $\mathbb{E}_\sigma$ can be decomposed into r disjoint partitions $\mathbb{E}_\sigma^k$ such that, for each $k = 1, \ldots, r$:

$$\cap_{i \in I_c(\sigma)} [\![\mathbb{E}_\sigma^k]\!]_i \cap \mathbb{D}_\sigma = \emptyset. \tag{6}$$

For our example, since we have only one controllable event to worry about, we have $\mathbb{D}_\sigma = \{\mathsf{a}_1\mathsf{b}_2, \mathsf{a}_2\mathsf{b}_1\}$, whereas we have the following partition of $\mathbb{E}_\sigma$ that satisfies (6): $\mathbb{E}_\sigma^1 = \{\mathsf{a}_1\mathsf{b}_1\}$ and $\mathbb{E}_\sigma^2 = \{\mathsf{a}_2\mathsf{b}_2\}$.

For each $t \in \mathbb{E}_\sigma \cup \mathbb{D}_\sigma$, each controller makes a local control decision based on its local observations with respect to each decomposition $\mathbb{E}_\sigma^k$ of $\mathbb{E}_\sigma$, for $k = 1, \ldots, r$. The local decision function $h_i^k : \Sigma_{o,i}^* \to LD^\Sigma$, for $k = 1, \ldots, r$, is defined for each $t \in \mathbb{E}_\sigma \cup \mathbb{D}_\sigma$ over $\mathbb{E}_k^\sigma$ (for $k = 1, \ldots, r$) as follows:

$$h_i^k(\pi_i(t))(\sigma) = \begin{cases} 0, & \text{if } [\![t]\!]_i\sigma \cap \mathbb{E}_\sigma^k \neq \emptyset; \\ 1, & \text{otherwise.} \end{cases} \tag{7}$$

Thus each local controller produces a set of multi-decisions for $t\sigma$ over the decomposition of $\mathbb{E}_\sigma$. The global decision function is calculated as follows (adjusted wrt updated Eqs. (1) and (3))

$$H(\pi(t))(\sigma) = \bigwedge_{k \in \{1,\ldots,r\}} [\bigvee_{i \in I} h_i^k(t)(\sigma)].$$

When $t = \mathsf{a}_1\mathsf{b}_2$, we have $H(\pi(t))(\sigma) = [h_1^1(\pi_1(t))(\sigma) \vee h_2^1(\pi_2(t))(\sigma)] \wedge [h_1^2(\pi_1(t))(\sigma) \vee h_2^2(\pi_2(t))(\sigma)] = 1$, as expected.

Although the authors propose a finite-state strategy for deriving the decomposition of $\mathbb{E}_\sigma$, such a decomposition is entirely artificial. The decomposition serves only to isolate the parts of $\mathbb{E}_\sigma$ where a definitive enablement decision can be taken wrt the particular element of the decomposition, leaving any uncertainty to take care of itself. As long as K is distributed observable, a decomposition where each element of $\mathbb{E}_\sigma$ is its own partition will always satisfy Eq.(6).

VII. Discussion

In [16], a decentralized architecture using PCX (Prioritized Composition with eXclusion), that includes $\oplus$ as one of the fusion rules, partitions the state space into areas where specific fusion operators can be applied. The requirement for synthesis using $\oplus$ as the fusion operator, requires exactly one controller to have the knowledge to make the correct control decision. In the parity-based architecture we present here, we do not require that exactly one controller has the correct knowledge. Rather, to respect parity, we only allow one controller to issue the disablement decision, even though others may also know the correct decision.

It is not difficult to find languages that are unconditionally co-observable and yield an XORSAT solution, or conditionally co-observable and yield an XORSAT solution. It is equally straightfoward to find languages that are unconditionally co-observable or conditionally co-observable but do not yield an XORSAT solution. Thus, the class of languages for which there are valid parity-based control solutions *intersects* rather than subsumes the other decentralized family of languages. For this reason, it is not straightforward to express our architecture in the ambiguity management framework of [5]. A language-based characterization of the parity-based architecture is ongoing work, and will formally clarify this relationship.

References

[1] K. Rudie and W. M. Wonham, "Think globally, act locally: Decentralized supervisory control," *IEEE Trans. Autom. Control*, vol. 37, no. 11, pp. 1692–1708, 1992.

[2] T.-S. Yoo and S. Lafortune, "A general architecture for decentralized supervisory control of discrete-event systems," *Discrete Event Dyn. S.*, vol. 12, no. 3, pp. 335–377, 2002.

[3] ——, "Decentralized supervisory control with conditional decisions: supervisor existence," *IEEE Trans. Autom. Control*, vol. 49, no. 11, pp. 1886–1904, 2004.

[4] S. L. Ricker and K. Rudie, "Knowledge is a terrible thing to waste: Using inference in discrete-event control problems," *IEEE Trans. Autom. Control*, vol. 52, no. 3, pp. 428–441, 2007.

[5] R. Kumar and S. Takai, "Inference-based ambiguity management in decentralized decision-making: Decentralized control of discrete event systems," *IEEE Trans. Autom. Control*, vol. 52, no. 10, pp. 1783–94, 2007.

[6] H. Chakib and A. Khoumsi, "Multi-decision supervisory control: Parallel decentralized architectures cooperating for controlling discrete event systems," *IEEE Trans. Automat. Control*, vol. 56, no. 11, pp. 2608–2622, 2011.

[7] G. Barrett and S. Lafortune, "Decentralized supervisory control with communicating controllers," *IEEE Trans. Automat. Control*, vol. 45, no. 9, pp. 1620–1638, 2000.

[8] S. L. Ricker, "Knowledge and communication in decentralized discrete-event control," Ph.D. dissertation, Queen's University, 1999.

[9] Y. Wang, T.-S. Yoo, and S. Lafortune, "Diagnosis of discrete event systems using decentralized architectures," *Discrete Event Dyn. S.*, vol. 17, pp. 233–263, 2007.

[10] S. L. Ricker and K. Rudie, "Know means no: Incorporating knowledge into discrete-event control systems," *IEEE Trans. Autom. Control*, vol. 45, no. 9, pp. 1656–1668, 2000.

[11] F. Lin and W. M. Wonham, "On observability of discrete-event systems," *Info. Sci.*, vol. 44, pp. 173–198, 1988.

[12] A. Biere, M. J. H. Heule, H. van Maaren, and T. Walsh, Eds., *Handbook of Satisfiability*, ser. Frontiers in Artificial Intelligence and Applications. IOS Press, 2009, vol. 185.

[13] S. L. Ricker and B. Caillaud, "Mind the gap: Expanding communication options in decentralized discrete-event control," *Automatica*, vol. 47, pp. 2364–2372, 2011.

[14] A. Arnold, *Finite transition systems*. Prentice–Hall, 1994.

[15] R. Morin, "Decompositions of asynchronous systems," in *CONCUR: LNCS 1446*, 1998, pp. 549–564.

[16] W. Qiu, R. Kumar, and V. Chandra, "Decentralized control of discrete event systems using prioritized composition with exclusion," *IEEE Trans. Autom. Control*, vol. 53, no. 10, pp. 2425 – 2430, 2008.