DECENTRALIZED SUPERVisory CONTROL OF DISCRETE EVENT SYSTEMS: INVOLVING THE FUSION SYSTEM IN THE DECISION-MAKING

Ahmed Khoumsi  Hicham Chakib
Department of Electical and Computer Engineering
University of Sherbrooke, CANADA

ABSTRACT
In this paper, we propose a new architecture for decentralized control of discrete event systems, whose basic principle is as follows: for every event which is controllable by several supervisors, each of the latter takes an enabling/disabling decision only when it is sure that this is the right decision which can be applied to the plant. Otherwise, the supervisor transmits its local information to the fusion system which will thus be involved in the enabling/disabling decision-making. We compare our approach with the previous decentralized approaches.

KEY WORDS
Decentralized Supervisory Control, Local supervisors, Fusion system, Distributing the decision-making.

1 Introduction
This paper deals with decentralized supervisory control (for brevity, the term supervisory can be omitted) of discrete event systems (DES), where a set of supervisors cooperate in order to restrict the behavior of a plant so that it respects a specification. Each supervisor $\text{Sup}_i$ observes only a part $\Sigma_{o,i}$ of the events of the plant and can control only a part $\Sigma_{c,i}$ of the events of the plant.

Many articles have studied decentralized control of DES [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]. The principle used in all these architectures, is that each each supervisor $\text{Sup}_i$ takes a control decision for each event $\sigma \in \Sigma_{c,i}$, according to its local observations. When several supervisors take decisions concurrently for the same event, a global decision is synthesized by “fusing” these local decisions. The difference between these architectures is on the type of decisions (e.g., permissive, anti-permissive, unconditional, conditional) taken by the supervisors, and on the type of fusion (e.g., conjunctive, disjunctive).

In all previous decentralized architectures, each supervisor that controls an event $\sigma$ always takes an enabling decision on $\sigma$, even when it is not sure of its decision. And when $\sigma$ is controllable by several supervisors, a fusion module $\text{Fuse}_\sigma$ is involved as follows: $\text{Fuse}_\sigma$ receives decisions from each supervisor that controls $\sigma$, and then $\text{Fuse}_\sigma$ combines these decisions and generates an actual enabling decision on $\sigma$. Our point of view about this approach is the following: the decision taken by each supervisor is in fact a coarse element of information and is constructed from the more precise information which is what has been observed by the supervisor. In other words, each supervisor converts a precise information into a coarse information before sending it to a fusion module who takes the actual decision.

The above point of view has inspired us for proposing an architecture based on the following approach: The information sent from a supervisor to a fusion module will be richer (i.e., more precise) than a simple decision. The richer information can for example be the event sequence observed by the supervisor or some related information. Consequently, the decision taken by a fusion module will be based on a more complete information than in previous architectures.

The contribution and organization of this paper are:

1. In Section 2, we present our new control architecture.
2. Section 3 presents the notions of $n$-observability and feasibility which, together with controllability, characterize the class of languages achievable with our control architecture. We show that this class includes the classes achievable with the previous architectures.
3. In Section 4, we present some algebraic properties of $n$-observability. We also discuss about the synthesis of $n$-observable languages when the specification is non-achievable.
4. And we conclude in Section 5.

2 New Control Approach

2.1 Motivation of Our Architecture
In previous decentralized architectures, when an event $\sigma$ is controllable by several supervisors, a fusion module $\text{Fuse}_\sigma$ receives decisions from each of these supervisors, and then $\text{Fuse}_\sigma$ combines these decisions and generates an actual enabling decision on $\sigma$. Our point of view about this approach is that the decision taken by each supervisor is a coarse element of information and is constructed from the more precise information which is the observation of the supervisor.
That is, each supervisor converts a precise information into a coarse information before sending it to a fusion module who takes the actual decision.

In this paper, we propose a new architecture which is motivated by the following questioning: why does a supervisor transmit a coarse information instead of transmitting a richer information? The latter can for example be the event sequence the supervisor has observed or some related information.

The supervisors and the fusion modules are presented in Subsection 2.2. We will see how these components behave in our architecture for controlling a plant so that it conforms to a specification.

**Remark 2.1** Note that we do not propose to add new communication channels between supervisors and fusion modules. We just propose to transmit richer information on the existing channels.

**Remark 2.2** For clarity and simplicity, we will consider uniquely controllable events. It is implicitly assumed that uncontrollable events are always enabled.

### 2.2 Formal Description of Our Architecture

Like previous architectures, the proposed architecture consists of two parts: a set of supervisors and a fusion system. Let us see how the supervisors and the fusion system behave with the objective to control a plant so that it conforms to a specification.

In the sequel, $G$ is a plant generating the prefix-closed language $\mathcal{L}(G)$ and the marked language $\mathcal{L}_m(G)$, and we consider a specification generating the language $\mathcal{K} \subseteq \mathcal{L}_m(G)$. We write $\overline{\mathcal{L}}$ for the prefix-closure of a language $L$. Let us define formally the supervisors and the fusion modules in the following subsections 2.2.1 and 2.2.2.

#### 2.2.1 Formal Description of the Supervisors

Sup$_i$ observes continuously (and partially) the plant. For every $\sigma \in \Sigma$, and every observed event sequence $\mu_i \in P_i(\overline{\mathcal{K}})$, Sup$_i$ computes the following two languages:

\[
\begin{align*}
REJ^{\mu_i}_{i,\sigma} &= (P^{-1}_i(\mu_i) \cap \mathcal{K})\sigma \cap (\mathcal{L}(G) \setminus \overline{\mathcal{K}}) \\
ACC^{\mu_i}_{i,\sigma} &= P^{-1}_i(\mu_i)\sigma \cap \overline{\mathcal{K}}
\end{align*}
\]

Note that $ACC^{\mu_i}_{i,\sigma} \subseteq \overline{\mathcal{K}}$ and $REJ^{\mu_i}_{i,\sigma} \subseteq \mathcal{L}(G) \setminus \overline{\mathcal{K}}$. The interpretation of these two languages is that:

- If $\sigma$ is accepted by $\overline{\mathcal{K}}$, then $\sigma$ leads to a sequence of $ACC^{\mu_i}_{i,\sigma}$.
- If $\sigma$ is accepted by $\mathcal{L}(G) \setminus \overline{\mathcal{K}}$ (we will also say: $\sigma$ is rejected by $\overline{\mathcal{K}}$), then $\sigma$ leads to a sequence of $REJ^{\mu_i}_{i,\sigma}$.

Therefore, Sup$_i$ generates one of the following three types of outputs, for every $\sigma \in \Sigma$:

1. When $REJ^{\mu_i}_{i,\sigma} = \emptyset$, Sup$_i$ deduces with certainty that $\sigma$ is accepted by $\overline{\mathcal{K}}$ if it is accepted by $\mathcal{L}(G)$, and thus, its output is “Enable $\sigma$”.
2. When $ACC^{\mu_i}_{i,\sigma} = \emptyset$, Sup$_i$ deduces with certainty that $\sigma$ is rejected by $\overline{\mathcal{K}}$, and thus, its output is “Disable $\sigma$”.
3. When $ACC^{\mu_i}_{i,\sigma} \neq \emptyset$ and $REJ^{\mu_i}_{i,\sigma} \neq \emptyset$, Sup$_i$ cannot determine with certainty if $\sigma$ is accepted or not by $\overline{\mathcal{K}}$. In this case, its output is the pair $(ACC^{\mu_i}_{i,\sigma}, REJ^{\mu_i}_{i,\sigma}) \subseteq (\overline{\mathcal{K}} \times (\mathcal{L}(G) \setminus \overline{\mathcal{K}}))$.

Since the two situations $REJ^{\mu_i}_{i,\sigma} = \emptyset$ and $ACC^{\mu_i}_{i,\sigma} = \emptyset$ generate contradictory decisions ($En$ and $Dis$), it may seem that the following situation is problematic:

$$\forall \sigma \in \Sigma, Sup_i(\mu_i) = (ACC^{\mu_i}_{i,\sigma}, REJ^{\mu_i}_{i,\sigma}),$$

More precisely, we have for $\mu_i \in P_i(\overline{\mathcal{K}})$:

\[
Sup_{i,\sigma}(\mu_i) = \begin{cases} 
En, & \text{if } REJ^{\mu_i}_{i,\sigma} = \emptyset \\
Dis, & \text{if } ACC^{\mu_i}_{i,\sigma} = \emptyset \\
(ACC^{\mu_i}_{i,\sigma}, REJ^{\mu_i}_{i,\sigma}), & \text{otherwise}
\end{cases}
\]

#### 2.2.2 Formal Description of the Fusion System

For every event $\sigma \in \Sigma$, recall that Ind$_\sigma$ is the set of indices $i$ such that $\sigma \in \Sigma_i$. The module Fuse$_\sigma$ can be conceptually seen as a function that, to a combination of outputs of the Sup$_i$ such that $i \in $ Ind$_\sigma$, associates an enabling/disabling decision on $\sigma$. Formally: $\forall \sigma \in \Sigma$, Fuse$_\sigma : \prod_{i \in $ Ind$_\sigma$} Sup_{i,\sigma} \rightarrow \{En, Dis\}$.

More precisely, we have:

\[
\text{Fuse}_\sigma \left( \prod_{i \in $ Ind$_\sigma$} Sup_{i,\sigma}(\mu_i) \right) = \begin{cases} 
En, & \text{if } (\exists i \in $ Ind$_\sigma$ s.t. Sup_{i,\sigma}(\mu_i) = En) \lor (\cap_{i \in $ Ind$_\sigma$} REJ^{\mu_i}_{i,\sigma} = \emptyset) \\
Dis, & \text{if } (\exists i \in $ Ind$_\sigma$ s.t. Sup_{i,\sigma}(\mu_i) = Dis) \lor (\cap_{i \in $ Ind$_\sigma$} ACC^{\mu_i}_{i,\sigma} = \emptyset) \\
\text{Any,} & \text{otherwise}
\end{cases}
\]

In Eq. (3), each of $En$ and $Dis$ has two conditions. Here are some explanations related to this equation:
The first condition of $En$ (resp. $Dis$) means that the module $Fuse_{\sigma}$ has received an $En$ (resp. $Dis$) from at least one $Sup_i$. In this case, $Fuse_{\sigma}$ applies this decision to the plant. Notice that $Fuse_{\sigma}$ may receive two conflicting decisions $En$ and $Dis$ only if $\sigma$ is not accepted by $\mathcal{L}(G)$. In this case, $Fuse_{\sigma}$ knows that its decision does not matter, and thus, can apply any of the two decisions $En$ or $Dis$ (see our discussion on $ACC_{\mu,i,\sigma} = REJ_{\mu,i,\sigma} = \emptyset$ in Subsection 2.2.1).

The second conditions of $En$ and $Dis$ occur when $Fuse_{\sigma}$ receives no decision from the supervisors. Instead, it receives the sets $ACC_{\mu,i,\sigma}$ and $REJ_{\mu,i,\sigma}$, for $i \in Ind_\sigma$. $Fuse_{\sigma}$ computes the intersections $\bigcap_{i \in Ind_\sigma} REJ_{\mu,i,\sigma}$ and $\bigcap_{i \in Ind_\sigma} ACC_{\mu,i,\sigma}$, which are in some sense, refinements of the information elements received from the supervisors. The interpretation of $\bigcap_{i \in Ind_\sigma} REJ_{\mu,i,\sigma}$ and $\bigcap_{i \in Ind_\sigma} ACC_{\mu,i,\sigma}$ is that:

- If $\sigma$ is accepted by $\mathcal{K}$, then it leads to a sequence of $\bigcap_{i \in Ind_\sigma} ACC_{\mu,i,\sigma}$. Therefore, $\bigcap_{i \in Ind_\sigma} ACC_{\mu,i,\sigma} = \emptyset$ implies that $\sigma$ is certainly rejected by $\mathcal{K}$, and thus, is disabled by $Fuse_{\sigma}$.

- If $\sigma$ is accepted by $\mathcal{L}(G) \setminus \mathcal{K}$, then it leads to a sequence of $\bigcap_{i \in Ind_\sigma} REJ_{\mu,i,\sigma}$. Therefore, $\bigcap_{i \in Ind_\sigma} REJ_{\mu,i,\sigma} = \emptyset$ implies that $\sigma$ is certainly accepted by $\mathcal{K}$ if it is accepted by $\mathcal{L}(G)$, and thus, is enabled by $Fuse_{\sigma}$.

The situation $Any$ occurs when both $\bigcap_{i \in Ind_\sigma} ACC_{\mu,i,\sigma}$ and $\bigcap_{i \in Ind_\sigma} REJ_{\mu,i,\sigma}$ are not empty, that is, when $Fuse_{\sigma}$ has not enough information to decide. The occurrence of $Any$ is not a problem when $\sigma$ is not accepted by $\mathcal{L}(G)$, because the decision of $Fuse_{\sigma}$ does not matter in such a case. On the other hand, the occurrence of $Any$ when $\sigma$ is accepted by $\mathcal{L}(G)$, means that $\mathcal{K}$ is not achievable with our architecture, that is, the control system cannot force the plant to conform to $\mathcal{K}$. In Section 3, we will present necessary and sufficient conditions for the achievability of $\mathcal{K}$.

For the case $\bigcap_{i \in Ind_\sigma} ACC_{\mu,i,\sigma} = \bigcap_{i \in Ind_\sigma} REJ_{\mu,i,\sigma} = \emptyset$, we can make similar comments as those already made for $REJ_{\mu,i,\sigma} = ACC_{\mu,i,\sigma} = \emptyset$. That is, the situation $\bigcap_{i \in Ind_\sigma} ACC_{\mu,i,\sigma} = \bigcap_{i \in Ind_\sigma} REJ_{\mu,i,\sigma} = \emptyset$ occurs only if $\sigma$ is not accepted by $\mathcal{L}(G)$. In this case, $Fuse_{\sigma}$ knows that its decision does not matter, and thus, can take any of the two decisions $En$ or $Dis$.

The proposed control architecture is illustrated in Figure 1 for a given $\sigma \in \Sigma_c$.

### 2.3 Independence the Fusion System

We have seen that the essential operations executed by a fusion module $Fuse_{\sigma}$ are intersections of languages received from the supervisors. Therefore, these operations are independent of $\mathcal{L}(G)$, $\mathcal{L}_m(G)$ and $\mathcal{K}$, and thus, the fusion modules have not to be adapted when $\mathcal{L}(G)$, $\mathcal{L}_m(G)$ and/or $\mathcal{K}$ are modified.

### 2.4 Variant of our Architecture

In order to help for the comprehension of the proposed architecture, let us present a variant which generates the same class of achievable languages. In the variant architecture, each supervisor behaves as a pure observer, that is, its output is simply the observed sequence. This variant is obtained by simply moving the computation of $(ACC_{\mu,i,\sigma} ; REJ_{\mu,i,\sigma})$ from the supervisors to the fusion system. More precisely, for every $\sigma \in \Sigma_c$, every $Sup_i$ (for $i \in Ind_\sigma$) is an observer that sends to $Fuse_{\sigma}$ its observed sequence $\mu_i$. For each received $\mu_i$, $Fuse_{\sigma}$ computes $(ACC_{\mu,i,\sigma} ; REJ_{\mu,i,\sigma})$. Then $Fuse_{\sigma}$ computes the intersections $(\bigcap_{i \in Ind_\sigma} ACC_{\mu,i,\sigma})$ and $(\bigcap_{i \in Ind_\sigma} REJ_{\mu,i,\sigma})$ and takes an enabling decision on $\sigma$.

Therefore, we can say that in this variant, the control of $\sigma$ is centralized in $Fuse_{\sigma}$ based on a distributed observation by the supervisors $Sup_i$ s.t. $i \in Ind_\sigma$.

### 3 Existence Results and Comparison with Other Architectures

#### 3.1 Existence Results

Before we characterize the class of languages achievable with our architecture, let us recall the notion of controllability and define the new notions of $n$-observability and feasibility.

**Definition 3.1** $\mathcal{K} \subseteq \mathcal{L}(G)$ is said to be controllable w.r.t $\mathcal{L}(G)$, $\Sigma_{uc}$ iff: $\forall \lambda, \mu \in \mathcal{K}, \forall \sigma \in \Sigma_c$ s.t. $(\lambda, \mu, \sigma) \in \mathcal{L}(G)$, we have: $\lambda \sigma \in \mathcal{K} \Leftrightarrow \mu \sigma \in \mathcal{K}$.

**Definition 3.2** $\mathcal{K} \subseteq \mathcal{L}(G)$ is said $n$-observable w.r.t $\mathcal{L}(G)$, $\Sigma_{uc,1}, \Sigma_{uc,1}, \ldots, \Sigma_{uc,n}$, $\Sigma_{en}$ iff $\forall \lambda, \mu \in \mathcal{K}, \forall \sigma \in \Sigma_c$ s.t. $(\lambda, \mu, \sigma) \in \mathcal{L}(G)$, we have: $\lambda \sigma \in \mathcal{K} \Leftrightarrow \mu \sigma \in \mathcal{K}$, s.t. $(\lambda, \mu, \sigma) \in \mathcal{L}(G)$. 
In Section 2, we have explained how the global supervisor (consisting of the supervisors and the fusion modules) interacts with the plant. This global supervisor will be denoted SUP.

**Definition 3.3** SUP is said feasible w.r.t \( K, L(G), \Sigma_{o,1}, \Sigma_{c,1}, \cdots, \Sigma_{o,n}, \Sigma_{c,n} \) iff
\[
\forall \sigma \in \Sigma_c, \forall \lambda \in K \text{ such that } \lambda \sigma \in L(G):
(\bigcap_{i \in Ind_a} ACC_{i,\sigma}^P(\lambda)) \lor (\bigcap_{i \in Ind_a} REJ_{i,\sigma}^P(\lambda)) = \emptyset.
\]

For brevity, in the sequel we omit “w.r.t \( L(G), \Sigma_{o,1}, \Sigma_{c,1}, \cdots, \Sigma_{o,n}, \Sigma_{c,n} \)” when we use n-observability and feasibility. When SUP is feasible, SUP can be seen as a function SUP : \( L(G) \rightarrow 2^K \) which associates to each sequence \( \lambda \in L(G) \) the set of enabled events. Formally:
\[
SUP(\lambda) = \{ \sigma \in \Sigma_c : \text{Fuse}_{\sigma}(\bigcap_{i \in Ind_a} SUP_{i,\sigma}(P_i(\lambda))) = En \} \cup \Sigma_{uc}.
\]

When SUP is feasible, let \( L(SUP(G)) \) denote the prefix-closed language generated by the plant under the control of SUP, and let \( L_m(SUP(G)) = L(SUP(G)) \cap L_m(G) \) be the corresponding marked language. \( L(SUP(G)) \) is formally defined as follows, where \( \varepsilon \) denotes the empty event sequence:
- \( \varepsilon \in L(SUP(G)) \)
- \( (\lambda \in L(SUP(G)) \land (\lambda \sigma \in L(G)) \land (\sigma \in SUP(\lambda))) \Rightarrow \lambda \sigma \in L(SUP(G)). \)

SUP is said non-blocking iff \( L_m(SUP(G)) = L(SUP(G)) \).

**Proposition 3.1** Consider a nonempty \( \mathcal{K} \subseteq L_m(G) \). SUP is feasible w.r.t \( \mathcal{K} \) iff \( \mathcal{K} \) is n-observable.

**Theorem 3.1** Consider a nonempty \( \mathcal{K} \subseteq L_m(G) \). SUP satisfies C0 to C3 iff \( \mathcal{K} \) satisfies K0 to K2:
- C0: SUP is non-blocking; C1: SUP is feasible w.r.t \( \mathcal{K} \);
- C2: \( L(SUP(G)) = \mathcal{K} \); C3: \( L_m(SUP(G)) = \mathcal{K} \).
- K0: \( \mathcal{K} \) is controllable w.r.t \( L(G) \) and \( \Sigma_{uc} \);
- K1: \( \mathcal{K} \) is n-observable; K2: \( \mathcal{K} = \mathcal{K} \cap L_m(G) \).

Theorem 3.1 means that every nonempty \( \mathcal{K} \subseteq L_m(G) \) respecting K0 to K2 can be achieved by controlling the plant G using SUP.

3.2 Comparison with Other Architectures

We have the following two propositions which situate our architecture in comparison with the decentralized architectures of [10, 11, 13] and the centralized architecture [14].

**Proposition 3.2** The class of languages achievable with our new architecture strictly includes the classes achievable with the other decentralized (general, conditional, infer-based) architectures.

**Proposition 3.3** The class of languages achievable with the centralized architecture strictly includes the class achievable with our new architecture.

3.3 Example of Comparison with the Inference-Based Architecture

The inference-based architecture of [13] is the most general among the previous decentralized architectures. Let us give an example which illustrates the fact that our new architecture strictly includes the class of languages achievable with the inference-based architecture.

We consider the plant of Figure 2 defined over the alphabet \( \{a_1, a_2, c\} \), where \( \Sigma_{o,1} = \Sigma_{c,2} = \{c\}, \Sigma_{o,1} = \{a_1\}, \Sigma_{o,2} = \{a_2\} \). The specification \( \mathcal{K} \) is obtained from the plant by forbidding \( c \) in the three states represented by a square, that is, by removing states 5, 9 and 10. Let us show that this specification is n-observable with our architecture. The events \( a_1 \) and \( a_2 \) are enabled in all situations, and for the event \( c \) the control system proceeds as follows:

- When both \( SUP_1 \) and \( SUP_3 \) have observed \( c \):
  - \( SUP_1 \) computes \( REJ_{1,c} = \{c\} \) and \( ACC_{1,c} = \{a_2c\} \)
  - \( SUP_2 \) computes \( REJ_{2,c} = \{c\} \) and \( ACC_{2,c} = \{a_1c\} \)
  - Fuse\(_c\) deduces \( REJ_{1,c} \cap REJ_{2,c} = \{c\} \) and \( ACC_{1,c} \cap ACC_{2,c} = \emptyset \). Therefore, \( c \) is disabled.

- When \( SUP_1 \) and \( SUP_2 \) have observed \( a_1 \) and \( a_2 \) resp.:
  - \( SUP_1 \) computes \( REJ_{1,c} = \{a_1a_2c, a_2a_1c\} \) and \( ACC_{1,c} = \{a_1c\} \)
  - \( SUP_2 \) computes \( REJ_{2,c} = \{a_1a_2c, a_2a_1c\} \) and \( ACC_{2,c} = \{a_2c\} \)
  - Fuse\(_c\) deduces \( REJ_{1,c} \cap REJ_{2,c} = \{a_1a_2c, a_2a_1c\} \) and \( ACC_{1,c} \cap ACC_{2,c} = \emptyset \). Therefore, \( c \) is disabled.

- When \( SUP_1 \) and \( SUP_2 \) have observed \( \varepsilon \) and \( \varepsilon \) resp.:
  - \( SUP_1 \) computes \( REJ_{1,c} = \{a_1a_2c, a_2a_1c\} \) and \( ACC_{1,c} = \{a_1c\} \)
  - \( SUP_2 \) computes \( REJ_{2,c} = \{a_1a_2c, a_2a_1c\} \) and \( ACC_{2,c} = \{a_2c\} \)
  - Fuse\(_c\) deduces \( REJ_{1,c} \cap REJ_{2,c} = \emptyset \) and \( ACC_{1,c} \cap ACC_{2,c} = \{a_1c\} \). Therefore, \( c \) is enabled.

- When \( SUP_1 \) and \( SUP_2 \) have observed \( \varepsilon \) and \( a_2 \) resp.:
  - \( SUP_1 \) computes \( REJ_{1,c} = \{c\} \) and \( ACC_{1,c} = \{a_2c\} \)
  - \( SUP_2 \) computes \( REJ_{2,c} = \{a_1a_2c, a_2a_1c\} \) and \( ACC_{2,c} = \{a_2c\} \)
  - Fuse\(_c\) deduces \( REJ_{1,c} \cap REJ_{2,c} = \emptyset \) and \( ACC_{1,c} \cap ACC_{2,c} = \{a_2c\} \). Therefore, \( c \) is enabled.

The control system is feasible because in every situation, we have \( REJ_{1,c} \cap REJ_{2,c} = \emptyset \lor (REJ_{1,c} \cap REJ_{2,c} = \emptyset) \) (see Def. 3.3). Therefore, the specification is n-observable (from Prop. 3.1).

Using the definition of infer-cooererbability of [13], it can be shown \( \mathcal{K} \) is not infer-cooererbable.
3.3.1 Example of Comparison with the Centralized Architecture

We consider the plant $\mathcal{L}_m(G)$ of Figure 3 defined over the alphabet $\{a_1, a_2, c\}$. We take $\Sigma_{a,1} = \{a_1\}, \Sigma_{a,2} = \{a_2\}, \Sigma_c = \Sigma_{c,1} = \Sigma_{c,2} = \{c\}$. The specification $K$ is obtained from the plant by forbidding $c$ in the state represented by a square. That is, $c$ is enabled (resp. disabled) after the execution of the sequence $a_1a_2$ (resp. $a_2a_1$). In a centralized architecture, the only unobservable event is $c$, and thus, the sequences $a_1a_2$ and $a_2a_1$ are distinguishable. Therefore, $K$ is observable.

Let us show that this specification is not $n$-observable with our architecture. When $Sup_1$ and $Sup_2$ have observed $a_1$ and $a_2$ respectively:

- The outputs of $Sup_1$ are $REJ_{1,c}^{a_1} = \{a_2a_1c\}$ and $ACC_{1,c}^{a_1} = \{a_1a_2c\}$.
- The outputs of $Sup_2$ are $REJ_{2,c}^{a_2} = \{a_2a_1c\}$ and $ACC_{2,c}^{a_2} = \{a_1a_2c\}$.
- $Fuse_c$ computes $REJ_{1,c}^{a_1} \cap REJ_{2,c}^{a_2} = \{a_2a_1c\}$ and $ACC_{1,c}^{a_1} \cap ACC_{2,c}^{a_2} = \{a_1a_2c\}$.

The control system is not feasible because both intersections are not empty (see Def. 3.3). Therefore, the specification is not $n$-observable (from Prop. 3.1). Intuitively, distinct decisions are to be taken in indistinguishable situations:

- $c$ must be enabled after the execution of $a_1a_2$, i.e., when $Sup_1$ and $Sup_2$ have observed $a_1$ and $a_2$, respectively.
- $c$ must be disabled after the execution of $a_2a_1$, i.e., when $Sup_1$ and $Sup_2$ have observed $a_1$ and $a_2$, respectively.

4 Algebraic Properties of $n$-observability

Similarly to the $*-\text{coobservabilities}$ of the previous architectures (i.e., general, conditional, inference based), $n$-observability satisfies the following two propositions:

**Proposition 4.1** $n$-observability is not necessarily preserved under union of languages.

**Proposition 4.2** $n$-observability is preserved under intersection of languages when the latter are prefix-closed.

Proposition 4.2 implies that when $K$ is not $n$-observable, the infimal $n$-observable language containing $K$ exists when the languages considered are prefix-closed. Proposition 4.1 implies that when $K$ is not $n$-observable, the supremal $n$-observable language included in $K$ does not necessarily exist. A solution would be to find a property, let us call it $n$-normality, which is:

1. preserved under union of languages,
2. stronger than $n$-observability,
3. weaker than the inference-coobservability, and thus characterizes a class of languages which includes those of the previous decentralized architectures.

In this case, if $K$ is not achievable (i.e., is not at the same time controllable, $n$-observable and $\mathcal{L}_m(G)$-closed), then the aim will be to synthesize the supremal $M \subseteq K$ which is controllable, $n$-normal and $\mathcal{L}_m(G)$-closed.

In [15], a joint-observability has been defined and shown to be undecidable. Although joint-observability seems quite similar to our $n$-observability, it can be easily proved that the two notions are distinct. Indeed, unlike $n$-observability, joint-observability is not based on the controllability of events.

5 Conclusion

In this paper, we proposed a new architecture for decentralized control of DES whose principle is as follows: when an event $\sigma$ is controllable by several supervisors, each of the latter takes an enabling/disabling unconditional decision if and only if it is sure that this is the right decision which can be applied to the plant. Otherwise, the supervisor transmits its local information to the fusion system. Consequently, when no supervisor can take a decision, the fusion system combines the elements of information transmitted by the supervisors in order to synthesize a decision which is applied to the plant.

We have defined the property of $n$-observability, which characterizes the class of languages which are achievable with our architecture. Like the $*\text{-coobservability}$ properties of the previous architectures, our $n$-observability is not preserved under union of languages, and thus the supremal $n$-observable language which is included in $K$ does not exist in general.
The main advantage of our new architecture is that \( n \)-observability characterizes a class of languages which strictly includes those of the previous decentralized architectures. But the price to be paid for this advantage, is that the computations for generating a control decision are done on-line. More precisely, the computations and intersections of the languages \( ACC_{\mu_i, \sigma} \) and \( REJ_{\mu_i, \sigma} \) (or their corresponding automata) are done on-line during the control of the plant. We are investigating how to solve (or at least reduce) this problem.

An other relevant issue is to investigate decidability of \( n \)-observability, and to propose some alternative in the case where \( n \)-observability is undecidable.

**Acknowledgements**

This work is partially financed by a grant from the National Sciences and Engineering Research Council (NSERC) of Canada.

**References**


