Decentralized Supervisory Control of Discrete Event Systems
Moving Decisions Closer to Actions

Ahmed Khoumsi and Hicham Chakib
Department of Electrical & Computer Engineering, University of Sherbrooke, Sherbrooke, Canada

Keywords: Decentralized Supervisory Control, Discrete Event Systems, Local Supervisors, Fusion Modules, Sensors, Actuators, Moving Decision Closer to Actions, Mixed Architecture, Mixed-observability.

Abstract: In decentralized control of discrete event systems, two main agents contribute to the computation of decisions: local supervisors and fusion modules. The local supervisors process the information detected on the plant and its environment, and transmit their results to the fusion modules. The latter process what is received from the local supervisors in order to decide actions to be applied to the plant. In the existing decentralized control architectures, the local supervisors execute complex operations, while the fusion modules execute simple operations. In the present article, we propose to move the decision computation complexity from local supervisors to fusion modules, that is what we term: moving decisions closer to actions. We justify this movement of decision and develop a simple architecture based on it. With the proposed architecture, the local supervisors are simple local observers, while all decisions are computed by the fusion modules. We characterize the class of languages achievable with the new architecture and compare it with the classes of languages achievable with the existing decentralized architectures and the centralized architecture.

1 INTRODUCTION

In supervisory control, the behavior of a plant, modeled as a discrete event systems (DES), is restricted by a supervision system (or supervisor) so that it respects a global specification (Ramadge and Wonham, 1987). In decentralized supervisory control (more briefly: decentralized control) of DES, observation and control tasks are shared among several agents that cooperate to take adequate decisions that restrict the behavior of the plant as desired. In this article, we consider decentralized control as it has been initiated in (Cieslak et al., 1988; Lin and Wonham, 1988; Lin and Wonham, 1990; Rudie and Wonham, 1992) and then continued in many studies, such as (Rudie and Willems, 1995; Prosser et al., 1997; Ricker and Rudie, 2000; Overkamp and van Schuppen, 2001; Jiang and Kumar, 2000; Yoo and Lafortune, 2002; Yoo and Lafortune, 2004; Ricker and Rudie, 2007; Kumar and Takai, 2007; Chakib and Khoumsi, 2011).

In decentralized control, two categories of agents contribute to the computation of decisions, which we call *local supervisors* and *fusion modules*. The local supervisors process detected information on the plant and its environment, and transmit their results to the fusion modules. The latter process what is received from the local supervisors in order to decide enablement/disablement of events of the plant.

Figure 1 outlines the conceptual structure of decentralized control, by an example with three local supervisors $(Sup_i)_{i=1\cdots3}$ and two fusion modules FM_σ and FM_μ. Each Sup_i observes the behavior of the plant through a mask (or projection) P_i, and processes the observed information before providing the result to one or both fusion modules. Each fusion module combines what it receives from two local supervisors, in order to take enablement/disablement decisions.

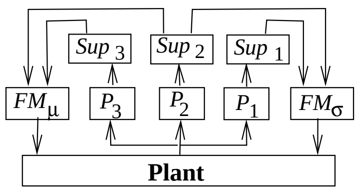

Figure 1: Decentralized control.

Definition 1. *A local supervisor or fusion module is said* powerful *if it has a high processing capacity and models of the dynamics of the plant and the specification. On the other hand, it is said* powerless *if it has a minimal processing capacity and no model of the dynamics of the plant and the specification.*

DOI: 10.5220/0005010702800287
In *Proceedings of the 11th International Conference on Informatics in Control, Automation and Robotics* (ICINCO-2014), pages 280-287
ISBN: 978-989-758-040-6

A typical minimal processing capacity is to be able to execute simple operations, for example boolean operations. A typical high processing capacity is to be able to manipulate complex automata-based models of DES, by executing various automata-based operators.

The following assumption has been made by the existing architectures of decentralized control:

Assumption 1. *The local supervisors are powerful and the fusion modules are powerless.*

Due to assumption 1, the local supervisors use complex operations (e.g. automata-based) to compute local decisions which are made accessible to fusion modules. Due to assumption 1, the fusion modules merge by simple (e.g. boolean) operations the local decisions of the local supervisors, in order to compute global decisions of enabling/disabling controllable events which are effectively applied to the plant. In a word, the most complex part of decision computation is done by powerful local supervisors, while the simplest part is done by powerless fusion modules. A possible justification of this approach is that it is consistent with hierarchical organizations, considering that fusion modules are at a higher level than local supervisors. Besides, this approach gives a good insight of the limit of control that can be reached with powerless fusion modules. In the sequel, decentralized control based on assumption 1 (which includes the previously mentioned references) will be referred to as *conventional decentralized control*.

Since most of the important results in decentralized control have been obtained in conventional decentralized control, assumption 1 has been considered almost as a standard. But we had a long reflection which led us to think that assumption 1 is not always indispensable and may sometimes have disadvantages. This is confirmed by our examination of several examples used in the literature of conventional decentralized control, in which we see no practical reason that obliges to make assumption 1. This assumption is even sometimes artificial, for example when the separation between local supervisors and fusion modules is logical but *not* physical. Indeed, we found examples where the same physical module hosts a local supervisor and a fusion module. We will return on this point in Section 2.5.

The above observation has motivated this article, where our objective is to propose a new decentralized architecture without the limitative assumption 1. Our first idea has then been to use powerful local supervisors as well as powerful fusion modules. But as we will explain in Section 2.5 and prove in Section 5.2, if the fusion modules are powerful, then it is not restrictive to have powerless local supervisors That is, the following assumption is not restrictive:

Assumption 2. *The local supervisors are powerless and the fusion modules are powerful.*

We use the following figurative definition:

Definition 2. *To* move decisions closer to actions *means to replace assumption 1 by assumption 2, because such a replacement implies moving the decision computation complexity from the local supervisors to the fusion modules.*

The remainder of the paper is organized as follows: In Section 2, we present a reflection which led us to deduce that moving decisions closer to actions is realistic and has advantages. A decentralized architecture based on this displacement of decisions is proposed in Section 3. The proposed architecture is called mixed architecture, because it has similarities with both decentralized and centralized architectures. In Section 4, we study the existence of solutions and define the notion of mixed-observability which, with controllability, characterizes the class of languages achievable under the mixed architecture. Section 5 compares this class with the classes of languages achievable under conventional decentralized control and centralized control. Finally, Section 6 contains a conclusion and propositions of future work.

2 MOVE DECISIONS CLOSER TO ACTIONS: REALISTIC AND ADVANTAGEOUS

This section presents a reflection on decentralized control, which leads us to deduce that assumption 1 is not always indispensable. More precisely, our deduction is that moving decisions closer to actions (i.e. replacing assumption 1 by assumption 2) is realistic and has advantages. The most important advantage is that such a movement of decision will permit to obtain a simpler and more general control architecture.

2.1 Structure based on Sensors and Actuators

In practice, a control system interacts with the plant under control through two categories of modules: sensors and actuators.

- Sensors are parts of the plant through which the control system detects information on the behavior and the environment of the plant. Examples of sensors: an obstacle detector that warns when an object is too close to the plant, and an accelerometer that informs of the acceleration of the plant.

- Actuators are parts of the plant through which the control system can influence the behavior of the plant. Examples of actuators: an electric motor that gives motion to a car, and a valve actuator that opens and closes a valve of a water system.

In decentralized control, several sensors and actuators are distributed at several parts of the plant, according to given requirements which may be: *physical*, i.e. to satisfy mandatory objectives due to physical considerations; or *logical*, i.e. to satisfy desirable objectives, e.g. to decrease cost or increase efficiency. Let us present concrete sensor and actuator neighborhoods and their usual theoretical models in conventional decentralized control.

2.1.1 Sensors Neighborhood

A sensor plays the role of an interface through which the control system obtains information on the behavior and the environment of the plant. According to given requirements, sensors are regrouped by sites identified by numbers $i = 1 \cdots n$. Each site i has a given *memory and processing capability* (more briefly: processing capacity) to process information obtained from sensors. The processed information is then made accessible to controllers of actuators. Let us make the link with the usual theoretical models:

- The information obtained in site i from sensors is modeled as the alphabet of events observable at site i, usually denoted $\Sigma_{o,i}$.
- The sensors of site i are modeled as a mask (or projection) P_i that shows (all and only) the events of $\Sigma_{o,i}$, among the events executed by the plant.
- The processing capacity of site i (that processes information obtained from sensors of site i) is modeled as a local supervisor Sup_i.

Hence, the neighborhood of the sensors of a site consists of the local supervisor of that site.

2.1.2 Actuator Neighborhood

An actuator is an interface through which the control system can influence the behavior of the plant. Each actuator is directly controlled by the means of a given processing capacity which, from information provided by local supervisors, decides which enablement/disablement actions to apply to the actuator. Let us make the link with the usual theoretical models.

- An actuator is associated to a set of controllable events and is modeled as a module that receives orders of enabling/disabling these controllable events. For example, a motor that receives the commands “start moving” and “stop moving”.
- The processing capacity used to control an actuator is modeled as a set of fusion modules, typically one fusion module FM_σ for each controllable event σ associated to the actuator.
- Each fusion module FM_σ merges (i.e. collects and processes) information from sites (more precisely, from local supervisors), in order to compute (enablement/disablement) decisions on σ that are applied to the actuator.
- The set of controllable events whose fusion modules are connected to (i.e. receive information from) a given site i, is specified by the The connectivity between local supervisors and fusion modules can also be specified by the *index set* of a controllable σ: $I_\sigma = \{i \mid \sigma \in \Sigma_{c,i}\}$. That is, I_σ specifies to which local supervisors is connected the fusion module FM_σ.

Hence, the neighborhood of an actuator consists of the set of fusion modules acting on it.

2.2 Knowledge of the Plant and Processing Capacity

We have seen that in decentralized control, two types of processing systems are used: local supervisors and fusion modules. Hence, to evaluate the capacity of control that can be achieved, it is necessary to specify, for each local supervisor and each fusion module: what knowledge it has of the plant and the specification, and what is its processing capacity.

2.2.1 Knowledge of the Plant and the Specification

It is necessary to specify what knowledge of the plant and the specification, each local supervisor and fusion module has. When it is not null, such a knowledge is usually in the form of automata-based models of the dynamics of the plant and the specification.

In conventional decentralized control, due to assumption 1, only the local supervisors have a knowledge of the plant and the specification.

2.2.2 Processing Capacity

It is necessary to specify what is the processing capacity of each local supervisor and fusion module. By processing capacity, we mean memory and processing capability. The fusion modules associated to the same actuator must be assumed having the same processing capacity, because they are physically in the control module of the same actuator. A typical minimal processing capacity is to be able to execute simple operations, for example boolean operations. A typical high processing capacity is to be able to manipulate complex automata-based models of DES by executing various automata-based operators.

In conventional decentralized control, due to assumption 1, only the local supervisors have a high processing capacity, while the fusion modules have a minimal processing capacity.

2.3 Functioning

Let $\Sigma_c = \bigcup_{i=1\cdots n} \Sigma_{c,i}$ denote the set of all controllable events. Each Sup_i (modeling the processing system in site i) uses its processing capacity and its knowledge of the plant and the specification, to process information taken from sensors of site i (modeled by the events of $\Sigma_{o,i}$). The processing result of Sup_i is then an information accessible by some means to every fusion module FM_σ connected to Sup_i (i.e. such that $\sigma \in \Sigma_{c,i}$, or equivalently $i \in I_\sigma$).

Each FM_σ (which models the processing system that decides on enablement/disablement of $\sigma \in \Sigma_c$) uses its processing capacity and its knowledge of the plant, to process information from supervisors Sup_i connected to it. The processing result of FM_σ is a command that is applied to an actuator of the plant to disable/enable σ.

In conventional decentralized control, due to assumption 1, powerful local supervisors compute local decisions from their knowledge of the plant and the specification, and powerless fusion modules compute effective decisions by combining the local decisions using operations independent of the plant and the specification.

2.4 Illustrative Example

Figure 2 outlines by an example our [sensor,actuator]-based vision of decentralized control. The plant has three sensors and two actuators. The three sensors detect information on the plant and transmit it in the form of events $a_1, \{a_2, b_2\}, c_3$, respectively. The two actuators are controlled by enablement/disablement of the events $\{\sigma, \mu\}$ and ρ, respectively. Hence, $\Sigma_c = \{\sigma, \mu, \rho\}$. Dec$(x)$ denotes the enablement/disablement decision taken on an event x.

The control system consists of three supervisors and three fusion modules. The three supervisors $(Sup_i)_{i=1\cdots 3}$ process the events obtained from the three sensors, respectively. These events are modeled by the local observable alphabets $\Sigma_{o,1} = \{a_1\}$, $\Sigma_{o,2} = \{a_2, b_2\}$ and $\Sigma_{o,3} = \{c_3\}$, An arrow connecting Sup_i to FM_x means that the processing result of Sup_i is accessible to FM_x Hence, FM_σ and FM_μ have access to processing results from Sup_1 and Sup_2, and FM_ρ has access to processing results from Sup_2 and Sup_3. This connectivity is modeled by the controllable alphabets $\Sigma_{c,1} = \{\sigma, \mu\}$, $\Sigma_{c,2} = \{\sigma, \mu, \rho\}$ and $\Sigma_{c,3} = \{\rho\}$. This connectivity is also modeled by the index sets: $I_\sigma = I_\mu = \{1, 2\}$ and $I_\rho = \{2, 3\}$.

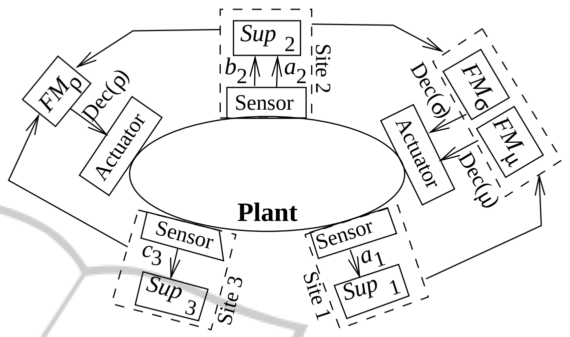

Figure 2: Example of our [sensor,actuator]-based vision of decentralized control.

2.5 Discussion and Objective

Intuitively, $\sigma \in \Sigma_{c,i}$ (or equivalently: $i \in I_\sigma$) means that Sup_i is involved in some way in the decision of enabling/disabling σ, by providing information to FM_σ which takes the effective decision.

Definition 3. *Sup_i is said* actively *involved in a decision (of enablement/disablement) on $\sigma \in \Sigma_c$, if it computes a local decision on σ which is used by FM_σ to compute the effective decision on σ. On the other hand, Sup_i is said* passively *involved in a decision on σ, if it just provides an observed information which is used by FM_σ to compute the effective decision on σ.*

Active involvement of a local supervisor is appropriate if it is powerful (Def. 1), because it can use its high processing capacity and its knowledge of the plant and the specification, to compute local decisions. Passive involvement of a local supervisor is appropriate if it is powerless (Def. 1).

In conventional decentralized control, due to assumption 1, the (powerful) local supervisors are actively involved by computing local decisions (using complex automata-based operations), and the (powerless) fusion modules combine the local decisions (using simple operations) in order to compute global decisions of enabling/disabling controllable events. Since most of the important results in decentralized control have been obtained in conventional decentralized control, assumption 1 has been practically considered as a standard in decentralized control. But our [sensor-actuator]-based vision shows no concrete reason that obliges to make assumption 1. Concretely, we see no reason why sensor environments must be powerful, while actuator environments must be powerless. Our impression is confirmed by the fact that the literature of conventional decentralized control presents no concrete justification of the indispensability of assumption 1. In some examples, the separation between local supervisors and fusion modules is logical but *not* physical. For example, an illustrative example of a decentralized traffic control is presented in (Yoo and Lafortune, 2004), where the same physical module hosts a local supervisor and a fusion module.

In addition to the fact that assumption 1 is not indispensable, we think that it is sometimes restrictive, from the following point of view: In conventional decentralized control, the local supervisors convert a precise information (their local observations) into a coarse information (their local decisions) before sending it to the fusion modules.

Our first objective has therefore been to remove assumption 1, by using powerful local supervisors as well as powerful fusion modules. But if the fusion modules are powerful, it is not indispensable to have powerful local supervisors. The explanation is that the maximal information which is available to decide of enabling/disabling a controllable event σ, is the set of events observed by the local supervisors connected to FM_σ. Such information can be provided by powerless local supervisors which just play the role of observers that forward their observations to the fusion modules. Hence our objective in Sections 3 and 4 is:

Objective 1. *To develop a decentralized control architecture which uses assumption 2 (instead of assumption 1).*

3 CONTROL ARCHITECTURE BASED ON ASSUMPTION 2

3.1 Principle of the Proposed Architecture

We will make the following assumption which has also been made in all the control architectures:

Assumption 3. *The plant is slower than the control system, in the sense that between the executions of any pair of consecutive events by the plant, the control system has the time to compute and apply its enablement/disablement decisions.*

For every controllable event σ, we denote by Σ_σ the set of events observable by the local supervisors connected to FM_σ. Formally:

Definition 4. *For every controllable event σ, the observable alphabet of FM_σ is $\Sigma_\sigma = \bigcup_{i \in I_\sigma} \Sigma_{o,i}$.*

In the proposed architecture, each fusion module FM_σ knows the observable alphabets of the local supervisors connected to it, that is, it knows every $\Sigma_{o,i}$ such that $i \in I_\sigma$ (or equivalently: $\sigma \in \Sigma_{c,i}$). Also, FM_σ is informed of the occurrence of every event in Σ_σ by the local supervisors connected to it. For these reasons and due to assumption 3, FM_σ has conceptually a partial observation of the plant through the alphabet Σ_σ, that is, FM_σ can determine the order of the occurrences of the events of Σ_σ. From the latter information, FM_σ determines its decisions on σ. In other words, each FM_σ is conceptually a centralized supervisor under partial observation of the plant through the alphabet Σ_σ. Hence, the proposed architecture is equivalent to a collection of centralized supervisors, each one controlling a single controllable event σ. We call it *mixed architecture*, because it has similarities with both decentralized and centralized controls.

Note that with the mixed architecture, the local supervisors are (powerless) *observers* and the fusion modules are (powerful) *event controllers*. But to facilitate a comparison with conventional decentralized control, we will keep the usual designations *local supervisor* and *fusion module*.

Figure 3 outlines the conceptual representation of the mixed architecture for the example of Figure 1, where P_σ and P_μ denote the projections in the alphabets Σ_σ and Σ_μ, respectively.

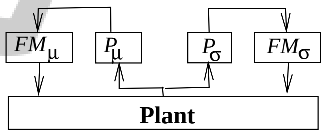

Figure 3: Conceptual representation of the mixed architecture for the example of Fig. 1.

3.2 Mixed Supervisor

As already explained, each FM_σ is a centralized supervisor under partial observation through the alphabet Σ_σ. As usual, consider an automaton G modeling the plant and the regular language $\mathcal{K}$ modeling the specification, both defined over the same alphabet Σ. Here are a few necessary definitions and notations:

- Σ^* is the set of all finite sequences of events of Σ, including the empty sequence.
- For two sequences $\lambda, \mu \in \Sigma^*$, μ is said a prefix of λ, if there exists $\rho \in \Sigma^*$ such that $\lambda = \mu\rho$.
- $\mathcal{L}(A)$ is the prefix-closed language of an automaton A, that is, the set of sequences starting in the initial state of A and terminating in any state of A.
- $\mathcal{L}_m(A)$ is the marked language of A, that is, the set of sequences starting in the initial state of A and terminating in a marked state of A.
- $\overline{L}$ consists of the sequences of L and their prefixes.

- $\Sigma_c = \bigcup_{i=1\cdots n}\Sigma_{c,i}$ and $\Sigma_{uc} = \Sigma \setminus \Sigma_c$ are the sets of controllable and uncontrollable events, resp.
- $\Sigma_o = \bigcup_{i=1\cdots n}\Sigma_{o,i}$ is the set of observable events.
- P (resp. P_i, P_σ) is the natural projection that hides, from any $\lambda \in \Sigma^*$, the events which are not in Σ_o (resp. $\Sigma_{o,i}$, Σ_σ).
- $\mathcal{E}_\sigma = \{\lambda \in \overline{\mathcal{K}} \,|\, \lambda\sigma \in \overline{\mathcal{K}}\}$.
- $\mathcal{D}_\sigma = \{\lambda \in \overline{\mathcal{K}} \,|\, \lambda\sigma \in \mathcal{L}(G) \setminus \overline{\mathcal{K}}\}$.
- *En* and *Dis* denote the decisions of enabling and disabling an event, respectively.

Intuitively, $\mathcal{E}_\sigma$ (resp. $\mathcal{D}_\sigma$) contains the sequences executed by the plant after which σ must be enabled (resp. disabled) by FM_σ to guarantee that (all and only) the sequences in the specification are accepted.

A supervisor of a control architecture can be defined as a function *SUP* that associates one of the decisions *En* or *Dis* to every pair (λ, σ), where λ is a sequence executed by the plant and σ is an event. $SUP(\lambda, \sigma)$ is the decision taken by *SUP* for σ when the plant has executed λ. As usual, $SUP(\lambda, \sigma) = En$ when $\sigma \in \Sigma_{uc}$.

A supervisor of a mixed architecture, which we call *mixed supervisor*, is the set of the fusion modules used in the mixed architecture. We consider that after the execution of a sequence λ by the plant, each fusion module FM_σ takes its decision $SUP(\lambda, \sigma)$ which depends on two things: 1) the observed execution $P_\sigma(\lambda)$, and 2) its knowledge of the plant and the specification, which is formalized by $P_\sigma(\mathcal{E}_\sigma)$ and $P_\sigma(\mathcal{D}_\sigma)$. Hence, a mixed supervisor is any supervisor that takes decisions $SUP(\lambda, \sigma)$ depending on the above points 1 and 2. Here is an example of mixed supervisor:

$$\forall\sigma \in \Sigma_c, SUP(\lambda,\sigma) = \left\{\begin{array}{ll} En, & \text{if } P_\sigma(\lambda) \in P_\sigma(\mathcal{E}_\sigma)\setminus P_\sigma(\mathcal{D}_\sigma) \\ Dis, & \text{if } P_\sigma(\lambda) \in P_\sigma(\mathcal{D}_\sigma)\setminus P_\sigma(\mathcal{E}_\sigma) \\ En \text{ or } Dis, & \text{if } P_\sigma(\lambda) \notin (P_\sigma(\mathcal{E}_\sigma)\cup P_\sigma(\mathcal{D}_\sigma)) \\ \text{don't know}, & \text{if } P_\sigma(\lambda) \in P_\sigma(\mathcal{E}_\sigma)\cap P_\sigma(\mathcal{D}_\sigma) \end{array}\right\} \quad (1)$$

Intuitively, the above mixed supervisor enables (resp. disables) σ when it is certain that enabling (resp. disabling) σ will not violate the specification. When FM_σ is certain that σ is impossible (i.e. σ is not accepted by the plant), it can decide any of *En* and *Dis*, because σ will not occur whatever the decision taken by FM_σ. When FM_σ is uncertain of the adequate decision, it takes a "don't know" decision which may concretely correspond to an alarm. We have considered this "don't know" situation explicitly, because it is undesirable and we will present in Section 4 the condition that avoids it.

3.3 Example

Consider the prefix-closed plant of Figure 4 where all the states are marked. The specification is obtained by removing state 9. We have $\Sigma_{o,1} = \{a_1\}, \Sigma_{o,2} = \{a_2\}, \Sigma_{o,3} = \{a_3\}$, $\Sigma_o = \Sigma_{o,1} \cup \Sigma_{o,2} \cup \Sigma_{o,3} = \{a_1, a_2, a_3\}$, $\Sigma_{c,1} = \Sigma_{c,2} = \Sigma_c = \{\sigma\}$, $\Sigma_{c,3} = \emptyset$. Hence, $I_\sigma = \{1,2\}$. Therefore, the observable alphabet of FM_σ is $\Sigma_\sigma = \{a_1, a_2\}$. We compute $\mathcal{E}_\sigma = \{a_1a_2a_3\}$, $\mathcal{D}_\sigma = \{a_2a_1a_3\}$, $P_\sigma(\mathcal{E}_\sigma) = \{a_1a_2\}$ and $P_\sigma(\mathcal{D}_\sigma) = \{a_2a_1\}$. If we apply Eq. (1) to this example, we obtain the mixed supervisor of Eq (2).

$$\forall\sigma \in \Sigma_c, SUP(\lambda,\sigma) = \left\{\begin{array}{ll} En, & \text{if } P_\sigma(\lambda) = a_1a_2 \\ Dis, & \text{if } P_\sigma(\lambda) = a_2a_1 \\ En \text{ or } Dis, & \text{if } P_\sigma(\lambda) \in \{a_1, a_2\} \end{array}\right\} \quad (2)$$

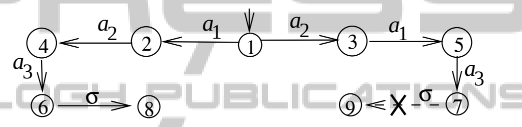

Figure 4: Example of plant and specification.

4 EXISTENCE OF SOLUTIONS

4.1 Mixed-Observability

Consider an automaton G modeling the plant over an alphabet Σ. We define a notion of observability, here called mixed-observability, which (with controllability) characterizes the class of languages achievable under the mixed architecture.

Definition 5. *Given a mixed architecture specified by the alphabets $(\Sigma_\sigma)_{\sigma\in\Sigma_c}$, a regular language $\mathcal{K} \subseteq \mathcal{L}_m(G)$ is said mixed-observable if: $\forall\sigma \in \Sigma_c$: $P_\sigma(\mathcal{E}_\sigma) \cap P_\sigma(\mathcal{D}_\sigma) = \emptyset$.*

Intuitively, $\mathcal{K}$ is mixed-observable if every FM_σ has at any time enough information to decide on the enablement/disablement of σ without violating the specification. This means that for every $\sigma \in \Sigma_c$, the decision "don't know" OF Eq. 1 is never taken.

When $\Sigma_\sigma = \Sigma_o$ for every $\sigma \in \Sigma_c$, mixed-observability is equivalent to observability of the centralized architecture. This is not surprising, because this case occurs when each FM_σ observes all the events that are observable by a centralized supervisor.

4.2 Existence Result

A language $\mathcal{K} \subseteq \mathcal{L}_m(G)$ is said $\mathcal{L}_m(G)$-closed if $\mathcal{K} = \overline{\mathcal{K}} \cap \mathcal{L}_m(G)$, and it is said $(\mathcal{L}(G), \Sigma_{uc})$-controllable if

$\overline{\mathcal{K}}\Sigma_{uc} \cap \mathcal{L}(G) \subseteq \overline{\mathcal{K}}$. Let $\mathcal{L}(SUP/G)$ denote the prefix-closed language generated by the plant under the control of a supervisor SUP, and let $\mathcal{L}_m(SUP/G) = \mathcal{L}(SUP/G) \cap \mathcal{L}_m(G)$ be the corresponding marked language. SUP is said non-blocking if $\overline{\mathcal{L}_m(SUP/G)} = \mathcal{L}(SUP/G)$. $\mathcal{L}(SUP/G)$ is formally defined as follows, where ε denotes the empty event sequence:

- $\varepsilon \in \mathcal{L}(SUP/G)$
- $[(\lambda \in \mathcal{L}(SUP/G)) \wedge (\lambda\sigma \in \mathcal{L}(G)) \wedge (SUP(\lambda,\sigma) = En)] \Leftrightarrow \lambda\sigma \in \mathcal{L}(SUP/G)$.

The following theorem states a necessary and sufficient condition for the existence of a mixed supervisor SUP that controls a plant G so that it respects a specification $\mathcal{K}$.

Theorem 1. *Consider a nonempty $\mathcal{K} \subseteq \mathcal{L}_m(G)$. There exists a nonblocking mixed supervisor SUP satisfying $\mathcal{L}(SUP/G) = \overline{\mathcal{K}}$ and $\mathcal{L}_m(SUP/G) = \mathcal{K}$ if and only if $\mathcal{K}$ is mixed-observable, $(\mathcal{L}(G),\Sigma_{uc})$-controllable and $\mathcal{L}_m(G)$-closed.*

4.3 Example

We return to the example of Sect. 3.3 represented in Fig. 4. The specification is mixed-observable because $P_\sigma(\mathcal{E}_\sigma) \cap P_\sigma(\mathcal{D}_\sigma) = \{a_1a_2\} \cap \{a_2a_1\} = \emptyset$. Since the specification is also $(\mathcal{L}(G),\Sigma_{uc})$-controllable and $\mathcal{L}_m(G)$-closed, then from Theorem 1, there exists a nonblocking mixed supervisor that controls the plant so that it respects the specification. An example of such a mixed supervisor is given by Eq. (2).

5 COMPARISON WITH OTHER ARCHITECTURES

As in Theorem 1 for the mixed architecture, the class of languages achievable by any control architecture is characterized by three notions: $(\mathcal{L}(G),\Sigma_{uc})$-controllability, $\mathcal{L}_m(G)$-closure and a notion of observability. Since $(\mathcal{L}(G),\Sigma_{uc})$-controllability and $\mathcal{L}_m(G)$-closure are independent of the considered architecture, we can compare control architectures by comparing their respective observabilities.

5.1 Comparison with the Centralized Control

The observability of a specification $\mathcal{K}$ in a centralized architecture, which we denote cent-observability, is defined as follows, where P is the projection on the observable alphabet Σ_o: $\forall\sigma \in \Sigma_c : P(\mathcal{E}_\sigma) \cap P(\mathcal{D}_\sigma) = \emptyset$.

Since $\Sigma_\sigma \subseteq \Sigma_o$ for every $\sigma \in \Sigma_c$, mixed-observability implies cent-observability. Therefore, the class of languages achievable with the centralized architecture subsumes the class of languages achievable with the mixed architecture.

We return to the example of Sect. 3.3 represented in Fig. 4, where we make the following modifications: $\Sigma_{c,1} = \Sigma_{c,3} = \Sigma_c = \{\sigma\}$ and $\Sigma_{c,2} = \emptyset$, and hence $I_\sigma = \{1,3\}$. Since $\Sigma_o = \{a_1,a_2,a_3\}$, we have $P(\mathcal{E}_\sigma) = \mathcal{E}_\sigma = \{a_1a_2a_3\}$ and $P(\mathcal{D}_\sigma) = \mathcal{D}_\sigma = \{a_2a_1a_3\}$. Hence, $P(\mathcal{E}_\sigma) \cap P(\mathcal{D}_\sigma) = \emptyset$, that is, the specification is cent-observable. The observable alphabet of FM_σ is now $\Sigma_\sigma = \{a_1,a_3\}$, $P_\sigma(\mathcal{E}_\sigma) = \{a_1a_3\}$ and $P_\sigma(\mathcal{D}_\sigma) = \{a_1a_3\}$. Hence, $P_\sigma(\mathcal{E}_\sigma) \cap P_\sigma(\mathcal{D}_\sigma) = \{a_1a_3\} \neq \emptyset$, that is, the specification is not mixed-observable.

Intuitively, since it observes $\Sigma_o = \{a_1,a_2,a_3\}$, a centralized supervisor can distinguish the sequences $a_1a_2a_3$ and $a_2a_1a_3$, where distinct decisions must be taken on σ. On the other hand, a mixed supervisor cannot distinguish these two sequences because it does not observe a_2.

5.2 Comparison with Conventional Decentralized Control

Let us consider any conventional decentralized architecture and denote by *coobservability* the corresponding notion of observability. A sufficient condition that a specification $\mathcal{K}$ is not coobservable is the existence of a pair of sequences necessitating distinct decisions and which cannot be distinguished by the local observations. More formally:

$$\mathcal{K} \text{ coobservable } \Rightarrow (\forall\sigma\in\Sigma_c) \\ (\forall\lambda\in\mathcal{E}_\sigma,\forall\mu\in\mathcal{D}_\sigma)(\exists i\in I_\sigma) : P_i(\lambda)\neq P_i(\mu) \quad (3)$$

On the other hand, from the fact that $\forall\sigma \in \Sigma_c, \forall i \in I_\sigma : \Sigma_{o,i} \subseteq \Sigma_\sigma$, we deduce that:

$$(\forall\sigma\in\Sigma_c)(\forall i\in I_\sigma)(\forall\lambda,\mu\in\Sigma^*) : \\ (P_i(\lambda) \neq P_i(\mu)) \Rightarrow (P_\sigma(\lambda)\neq P_\sigma(\mu)) \quad (4)$$

Equations (3,4) imply the following equation:

$$\mathcal{K} \text{ coobservable } \Rightarrow \\ (\forall\sigma\in\Sigma_c)(\forall\lambda\in\mathcal{E}_\sigma,\forall\mu\in\mathcal{D}_\sigma) : P_\sigma(\lambda)\neq P_\sigma(\mu) \quad (5)$$

From Def. 5 and Eq. (5), we have: coobservability implies mixed-observability, from which we deduce that the class of languages achievable with the mixed architecture subsumes the class of languages achieved by any conventional decentralized architecture. Hence, assumption 2 is not restrictive.

Let us return to the example of Section 3.3 represented in Figure 4. We have shown in Section 4.3

that the specification is mixed-observable. On the other hand, for $\lambda = a_1a_2a_3 \in \mathcal{E}_\sigma, \mu = a_2a_1a_3 \in \mathcal{D}_\sigma$, we have $P_1(\lambda) = P_1(\mu)$ and $P_2(\lambda) = P_2(\mu)$. From Eq. (3), we deduce that the specification is not coobservable in conventional decentralized control. Therefore, no conventional decentralized architecture can achieve the specification of this example.

6 CONCLUSION

We have proposed a decentralized controlled architecture, called mixed architecture, where the local supervisors are simple local observers, while all computations of decisions are made by the fusion modules which are conceptually equivalent to centralized supervisors. We prove that this approach is realistic and that it is simpler and more general than conventional decentralized control. Actually, this idea has already been studied in (Khoumsi and Chakib, 2008), but the corresponding developed architecture was indecidable, due to the fact that the plant was not assumed slower than the control systems.

In a near future, we intend to study the applicability of the mixed architecture in complex real life system.

REFERENCES

Chakib, H. and Khoumsi, A. (2011). Multi-decision Supervisory Control: Parallel Decentralized Architectures Cooperating for Controlling Discrete Event Systems. *IEEE Transactions on Automatic Control*, 56(11):2608–2622.

Cieslak, R., Desclaux, C., Fawaz, A., and Varaiya, P. (1988). Supervisory control of discrete event processes with partial observations. *IEEE Transactions on Automatic Control*, 33(3):249–260.

Jiang, S. and Kumar, R. (2000). Decentralized control of discrete event systems with specializations to local control and concurrent systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 30(5):653–660.

Khoumsi, A. and Chakib, H. (2008). A New Architecture for Decentralized Control of Discrete Event Systems: Decidability and Synthesis Issues. In *8ème conf. francophone de MOdélisation et SIMulation (MOSIM)*, Paris, France.

Kumar, R. and Takai, S. (2007). Inference-Based Ambiguity Management in Decentralized Decision-Making: Decentralized Control of Discrete Event Systems. *IEEE Transactions on Automatic Control*, 52(10):1783–1794.

Lin, F. and Wonham, W. M. (1988). Decentralized supervisory control of discrete event systems. *Information Sciences*, 44:199–224.

Lin, F. and Wonham, W. M. (1990). Decentralized control and coordination of discrete-event systems with partial observation. *IEEE Transactions on Automatic Control*, 35(12):1330–1337.

Overkamp, A. and van Schuppen, J. H. (2001). Maximal solutions in decentralized supervisory control. *SIAM Journal on Control and Optimization*, 39(2):492–511.

Prosser, J. H., Kam, M., and Kwatny, H. G. (1997). Decision fusion and supervisor synthesis in decentralized discrete-event systems. In *American Control Conference (ACC)*, pages 1313–1319.

Ramadge, P. J. and Wonham, W. M. (1987). Supervisory control of a class of discrete event processes. *SIAM J. Control and Optimization*, 25(1):206–230.

Ricker, S. and Rudie, K. (2000). Know means no: Incorporating knowledge into discrete-event control systems. *IEEE Transactions on Automatic Control*, 45(9):1656–1668.

Ricker, S. L. and Rudie, K. (2007). Knowledge is a terrible thing to waste: using inference in discrete-event control problems. *IEEE Transactions on Automatic Control*, 52(3):428–441.

Rudie, K. and Willems, J. C. (1995). The computational complexity of decentralized discrete event control problems. *IEEE Transactions on Automatic Control*, 40(7):1313–1319.

Rudie, K. and Wonham, W. M. (1992). Think globally, act locally: decentralized supervisory control. *IEEE Transactions on Automatic Control*, 31(11):1692–1708.

Yoo, T.-S. and Lafortune, S. (2002). A General Architecture for Decentralized Supervisory Control of Discrete-Event Systems. *Discrete Event Dyna. Syst.: Theory Applicat.*, 12:335–377.

Yoo, T.-S. and Lafortune, S. (2004). Decentralized Supervisory Control With Conditional Decisions: Supervisor Existence. *IEEE Transactions on Automatic Control*, 49(11):1886–1904.