Decentralized Control of Petri Nets

Technical Report of the ISIS Group
at the University of Notre Dame
ISIS-2002-005
October, 2002

Marian V. Iordache and Panos J. Antsaklis
Department of Electrical Engineering
University of Notre Dame
Notre Dame, IN 46556, USA
E-mail: iordache.1@nd.edu, antsaklis.1@nd.edu

Interdisciplinary Studies of Intelligent Systems

Decentralized Control of Petri Nets

Marian V. Iordache and Panos J. Antsaklis*

Abstract

Supervision based on place invariants (SBPI) is an efficient technique for the supervisory control of Petri nets. In this paper we propose extensions of the SBPI to a decentralized control setting. In our setting, a decentralized supervisor consists of local supervisors, each controlling and observing a part of the Petri net. We consider both versions of decentralized control, with communication, and with no communication. In the case of communication, a local supervisor may receive observations of events that are not locally observable and send enabling decisions concerning events that are not locally controllable. In the first part of the paper we propose efficient algorithms for the design of decentralized supervisors, based on the extension of the SBPI concept of admissibility that we define. Then, in the second part of the paper, we propose the design of decentralized supervisors based on transformations to admissible constraints. The feasibility of this problem is demonstrated with a simple integer programming approach. This approach can incorporate communication between local supervisors as well as communication constraints.

1 Introduction

The decentralized control of discrete event systems (DES) has received considerable attention in the recent years [12]. The current research effort has been focused on the automata setting, and has considered both versions of decentralized control, with communication and with no communication. This paper considers the decentralized control of Petri nets by means of the supervision based on place invariants (SBPI) [3, 10, 19].

Petri nets are compact models of concurrent systems, as they do not represent explicitly the state space of the system. Petri net models arise naturally in a variety of applications, such as manufacturing systems and communication networks. Petri net methods relying on the structure of the net rather than the state space are of special interest, as the size of the state space, when finite, can be exponentially related to the size of the net. Among such methods, the SBPI offers an efficient technique for the design of supervisors enforcing on Petri nets a particular class of state predicates, called *generalized mutual exclusion constraints*. Note that the generalized mutual exclusion constraints can represent any state predicate of a *safe*[1] Petri net [19]. Furthermore,

*Department of Electrical Engineering, University of Notre Dame, IN 46556, USA. E-mail: iordache.1, antsaklis.1@nd.edu.

[1]A Petri net is safe if for all reachable markings no place has more than one token.

without loss of any of its benefits, the SBPI has been extended in [5] to handle any constraints that can be enforced by control (monitor) places. While SBPI has been considered so far in a centralized setting, this paper proposes extensions of SBPI to a decentralized setting.

Admissibility is a key concept in the SBPI of Petri nets with uncontrollable and unobservable transitions. When dealing with such Petri nets, the SBPI approach classifies the specifications as admissible and inadmissible, where the former can be directly enforced, and the latter are first transformed to an admissible form and then enforced. In the automata setting [11], admissibility corresponds to controllability and observability, and the transformation to an admissible form to the computation of a controllable and observable sublanguage.

The main contributions of this paper are as follows. First, we define *d-admissibility* (decentralized admissibility), as an extension of admissibility to the decentralized setting. Our concept of d-admissibility extends the admissibility concept in the sense that a set of constraints that is d-admissible can be *directly* enforced via SBPI (i.e., without computational overhead) in a decentralized setting. Since d-admissibility identifies constraints for which the supervisors can be easily computed, rather than the class of constraints for which supervisors can be computed, it does not parallel controllability and coobservability in the automata setting [14]. Second, we show how to enforce *d-admissible* constraints and show how to check whether a constraint is d-admissible. Third, to deal with constraints that are not d-admissible, we provide an algorithmic approach to make the constraints d-admissible by enabling communication of events (transition firings). Fourth, to deal with the case in which the constraints are not d-admissible and communication is restricted or unavailable, we propose a simple linear integer programming approach for the design of the decentralized control. The design process generates both the local supervisors and the communication policy. Communication enables the local supervisors to observe events that are not locally observable and to control events that are not locally controllable. The communication policy specifies for each local supervisor the events it remotely observes and the events it remotely controls. This approach allows communication constraints to be incorporated in the design process and can be used to minimize the communication. With regard to our use of integer programming, note that while the development of alternative methods that are less computationally intensive are a direction for future research, in the automata setting it was shown that a decentralized solution cannot be found with polynomial complexity [12]. Note also that the size of the integer program depends on the size of the Petri net structure, and not on the size of its state space (i.e. the size of its equivalent automaton), which may not be finite.

To our knowledge, the decentralized supervisory control of Petri nets has not been yet considered in the literature. In the automata setting, the related work is as follows. Decentralized control when the specification is already given in a decomposed form is studied in [9]. The paper proposes a coordinator for the enforcement of additional specifications not given in the decomposed form. The existence of decentralized supervisors exactly implementing a given language has been first studied in [2]. In [14] coobservability was defined, and it was shown that a decentralized supervisor exactly enforcing a language exists iff the language is controllable and coobservable. In [17], the problem of finding a decentralized solution with the same performance as a centralized solution is considered in a setting in which communication is allowed. The communication consists of local supervisors sending to other local supervisors observation strings. The decentralized control problem with communication is studied in [1]. In this problem both the communication policy and the supervisors are designed. The communication setting consists of supervisors broadcasting their state estimates. Other decentralized control work can be found in the survey [12] and the references therein. Literature on SBPI or closely related to it is found in [3, 19, 10, 7, 8, 4, 15] and the references therein.

Compared to the related work, the idea of information structures in [17] is related to the clustering of subsystems in our paper. However, unlike [17, 1], our communication setting involves sending observed events rather than observation strings or state estimates. This kind of communication has also been considered in [13]. Furthermore, note that in this paper our focus is on computationally efficient or tractable methods for the decentralized control of Petri nets. Therefore, while the optimality of the result and the generality of the solution are also matters of interest, they are not the primary goals of our approach. This differentiates our work from the fundamental results of [14, 2], concerning optimal solutions in the general DES framework. The vast majority of the decentralized control papers consider language specifications. In this paper most developments are focused on a particular class of state predicate specifications on Petri nets. In the automata setting, the existence of a decentralized solution enforcing state predicates is studied in [16]. The relation between state predicate specifications and language specifications is as follows: any language can be represented as a state predicate on a system consisting of the plant and a "memory" DES [7].

The paper is organized as follows. Section 2 describes the notation and outlines the SBPI. Section 3 describes the decentralized setting of our approach. Section 4 defines the d-admissibility, shows how d-admissible constraints can be enforced, and presents the algorithm checking whether a constraint is d-admissible. Then, d-admissibility is applied to the design of local supervisors with communication in section 5. The algorithm presented in section 5 uses communication in order to reduce (when possible) the enforcement of constraints that are not d-admissible to the enforcement of d-admissible constraints. Section 6 describes the supervisory approach for the enforcement of constraints that are not d-admissible in the case in which the communication is restricted or not available. Section 7 shows how the results obtained in the previous sections extend to the generalized type of constraints described in [5] and to the automata setting. Finally, section 8 illustrates our approach on a manufacturing example from [9].

2 Preliminaries

A Petri net structure is denoted by $\mathcal{N} = (P, T, F, W)$, where P is the set of places, T the set of transitions, F the set of transition arcs, and W the weight function. The incidence matrix of $\mathcal{N}$ is denoted by D (places correspond to rows and transitions to columns). A place (transition) denoted by p_j (t_i) is the place (transition) corresponding to the j'th (i'th) row (column) of the incidence matrix.

The specification of the SBPI [3, 10, 19] consists of the state constraints

$$L\mu \leq b \tag{1}$$

where $L \in \mathbb{Z}^{n_c \times |P|}$, $b \in \mathbb{Z}^{n_c}$, and μ is the marking of $\mathcal{N}$. To distinguish between the case $n_c = 1$ and $n_c > 1$, we say that (1) represents *a constraint* when $n_c = 1$, and that (1) represents *a set of constraints* when $n_c > 1$. Note that $\mathcal{N}$ represents the **plant**. The SBPI provides a supervisor in the form of a Petri net $\mathcal{N}_s = (P_s, T, F_s, W_s)$ with

$$\begin{aligned} D_s &= -LD & (2)\\ \mu_{0,s} &= b - L\mu_0 & (3) \end{aligned}$$

where D_s is the incidence matrix of the supervisor, $\mu_{0,s}$ the initial marking of the supervisor, and μ_0 is the initial marking of $\mathcal{N}$. The places of the supervisor are called **control places**. The supervised system, that is the **closed-loop** system, is a Petri net of incidence matrix:

$$D_c = \left[\begin{array}{c} D \\ -LD \end{array} \right] \tag{4}$$

An example is shown in Figure 4(b), in which the supervisor enforcing $\mu_1 + \mu_2 \leq 1$ and $\mu_3 + \mu_4 \leq 1$ consists of the control places C_1 and C_2.

Note that (3) implies that when the plant and the supervisor are in closed-loop, the initial marking of the plant satisfies (1). Let μ_c be the marking of the closed-loop, and let $\mu_c|_{\mathcal{N}}$ denote μ_c restricted to the plant $\mathcal{N}$. Let $t \in T$ be a transition. t is **closed-loop enabled** if μ_c enables t. t is **plant-enabled**, if $\mu_c|_{\mathcal{N}}$ enables t in $\mathcal{N}$. The supervisor **detects** t if t is closed-loop enabled at some reachable marking μ_c and firing t changes the marking of some control place. The supervisor **controls** t if there is a reachable marking μ_c such that t is plant-enabled but not closed-loop enabled. Given μ_c, the supervisor **disables** t if there is a control place C such that $(C, t) \in F_s$ and $\mu_c(C) < W_s(C, t)$.

In Petri nets with uncontrollable and unobservable transitions, admissibility issues arise. Indeed, a supervisor generated as shown above may include control places preventing plant-enabled uncontrollable transitions to fire, and may contain control places with marking varied by firings of closed-loop enabled unobservable transitions. Such a supervisor is clearly not implementable. We say that a supervisor is admissible, if it only controls controllable transitions, and it only detects observable transitions. The constraints $L\mu \leq b$ are **admissible** if the supervisor defined by (2–3) is admissible. When inadmissible, the constraints $L\mu \leq b$ are transformed (if possible) to an admissible form $L_a\mu \leq b_a$ such that $L_a\mu \leq b_a \Rightarrow L\mu \leq b$ [10]. Then, the supervisor enforcing $L_a\mu \leq b_a$ is admissible, and enforces $L\mu \leq b$ as well. Our discussion on admissibility is carried out in more

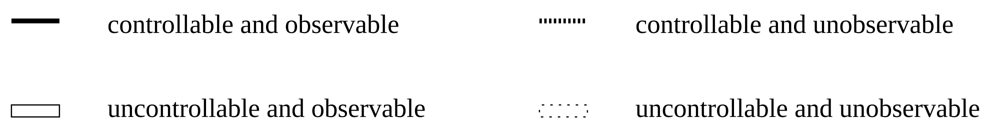

controllable and observable
controllable and unobservable
uncontrollable and observable
uncontrollable and unobservable

Figure 1: Graphical representation of the transition types.

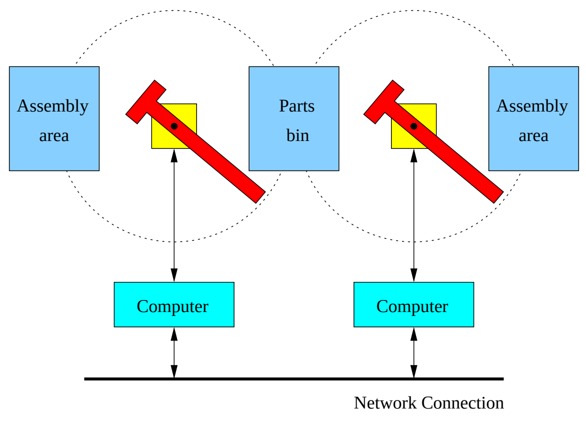


Figure 2: Robotic manufacturing system.

detail in section 4. We will denote $\mathcal{N}$ with sets of uncontrollable and unobservable transitions T_{uc} and T_{uo} by $(\mathcal{N}, T_{uc}, T_{uo})$.

Finally, Figure 1 shows the graphical representation of the uncontrollable and/or unobservable transitions that is used in this paper.

3 The Model

We assume that the system is given as a Petri net structure $\mathcal{N} = (P, T, F, W)$. A decentralized supervisor consists of a set of local supervisors $\mathcal{S}_1$, $\mathcal{S}_2$,... $\mathcal{S}_n$, each acting upon individual parts of the system, called *subsystems*, where the simultaneous operation of the local supervisors achieves a global specification. A local supervisor $\mathcal{S}_i$ observes the system through the set of locally observable transitions $T_{o,i}$, and controls it through the set of locally controllable transitions $T_{c,i}$. So, from the viewpoint of $\mathcal{S}_i$, the sets of uncontrollable and unobservable transitions are $T_{uc,i} = T \setminus T_{c,i}$ and $T_{uo,i} = T \setminus T_{o,i}$. This is the design problem: *Given a global specification and the sets of uncontrollable and unobservable transitions $T_{uc,1}$, $T_{uc,2}$, ... $T_{uc,n}$ and $T_{uo,1}$, $T_{uo,2}$, ... $T_{uo,n}$, find a set of local supervisors $\mathcal{S}_1$, $\mathcal{S}_2$,... $\mathcal{S}_n$ whose simultaneous operation guarantees that the global specification is satisfied, where each $\mathcal{S}_i$ can control $T \setminus T_{uc,i}$ and observe $T \setminus T_{uo,i}$.* A system $\mathcal{N}$ with subsystems of uncontrollable and unobservable transitions $T_{uc,i}$ and $T_{uo,i}$ will be denoted by $(\mathcal{N}, T_{uc,1}, \ldots T_{uc,n}, T_{uo,1}, \ldots T_{uo,n})$.

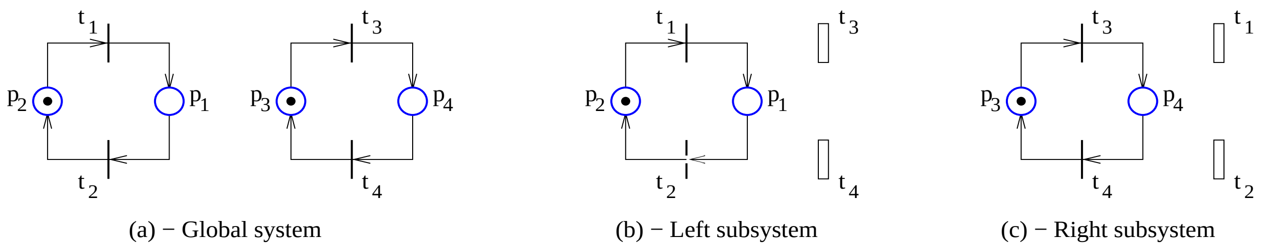


Figure 3: A Petri net model of the robotic manufacturing system.

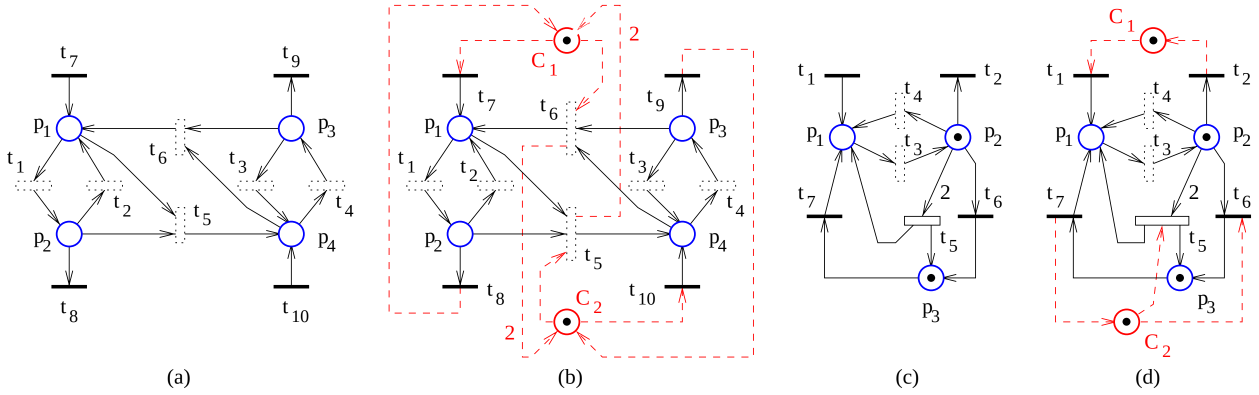


Figure 4: Examples of c-admissible supervision.

As an example, we consider the manufacturing system of [6], shown in Figure 2. In this example, two robots access a common parts bin. The system can be modeled by the Petri net of Figure 3(a), where $\mu_2 = 1$ ($\mu_4 = 1$) when the left (right) robot is in the assembly area, and $\mu_1 = 1$ ($\mu_3 = 1$) when the left (right) robot is in the parts bin. The set of controllable transitions of the left (right) subsystem may be taken as $T_{c,1} = \{t_1, t_2\}$ ($T_{c,2} = \{t_3, t_4\}$). Assume that the subsystem of each robot knows when the other robot enters or leaves the parts bin. Then each subsystem contains the controllable transitions of the other subsystem as observable transitions; a possible graphical representation of the subsystems is shown in Figure 3(b) and (c).

4 Admissibility

To distinguish between admissibility in the centralized case and admissibility in the decentralized case (to be defined later), we denote by **c-admissibility** the admissibility property in the centralized case. Therefore, *c-admissibility* is taken with respect to a Petri net $(\mathcal{N}, \mu_0)$ of uncontrollable transitions T_{uc} and unobservable transitions T_{uo}. The significance of c-admissibility is as follows. A c-admissible set of constraints (1) can be implemented with the simple construction of (2–3), as in the fully controllable and observable case.

It is essential for the understanding of this paper to see that supervisors defined by (2–3) may be admissible even when they have control places connected to unobservable transitions, and control places connected to uncontrollable transitions by place-to-transition arcs. We first illustrate this fact by two examples, and then, in the next paragraph, we show how such admissible supervisors can be (physically) implemented. In the first example, the supervisor enforcing $\mu_1 + \mu_2 \leq 1$ and $\mu_3 + \mu_4 \leq 1$ in the Petri net of Figure 4(a) is shown in Figure 4(b). By definition, the supervisor is admissible, in spite of the fact that it is connected to the uncontrollable and unobservable transitions t_5 and t_6. The reason is that, on one side, whenever the supervisor disables t_5 (or t_6), t_5 (t_6) is anyway disabled by the plant and, on the other side, t_5 and t_6 are dead (they require $\mu_1 + \mu_2 \geq 2$ and $\mu_3 + \mu_4 \geq 2$, respectively, in order to be plant-enabled) and so their observation is not necessary. In the second example, the supervisor enforcing $\mu_1 + \mu_2 + \mu_3 \leq 3$ and $\mu_3 \leq 2$ in the Petri net of Figure 4(c) is shown in Figure 4(d). Again, the supervisor is admissible, in spite of the fact that it may disable the uncontrollable transition t_5. Indeed, the supervisor never disables t_5 when t_5 is plant-enabled, and so its disablement decision does not need to be physically implemented. In fact, the arc (C, t_5) can be seen as corresponding to an observation action only, as the supervisor decrements the marking of C_2 whenever t_5 fires.

The previous examples motivate the following interpretation of the arcs between the control places of an *admissible* supervisor and the uncontrollable and/or unobservable transitions. Let C be a control place and t a transition. *If t is uncontrollable, an arc (C, t) models observation only, due to the fact that an admissible supervisor never disables a plant-enabled transition; physically, this means that the supervisor has a sensor to monitor t but no actuator to control t. If t is unobservable and controllable, an arc (C, t) models control only, as the fact that an admissible supervisor does not observe closed-loop enabled unobservable transitions indicates that t is dead in the closed-loop*[2]*; physically, the supervisor has an actuator to control t but no sensor to monitor t. If t is unobservable and uncontrollable, arcs between C and t can be ignored, as the fact that an admissible supervisor would never disable or observe t if plant-enabled, implies that in the closed-loop t is never plant-enabled.* A summary of the interpretation of the arcs between control places and transitions is found in Appendix A.

In the decentralized case, we are interested to define admissibility with respect to a Petri net $(\mathcal{N}, \mu_0)$, and the sets of uncontrollable and unobservable transitions of the subsystems: $T_{uc,1} \ldots T_{uc,n}$ and $T_{uo,1} \ldots T_{uo,n}$. Admissibility in the decentralized case is called **d-admissibility**. As in the case of c-admissibility, we would like d-admissibility to guarantee that we are able to construct the (decentralized) supervisor without employing constraint transformations. This is achieved by the following definition.

Definition 4.1 *A constraint is* **d-admissible** *with respect to $(\mathcal{N}, \mu_0, T_{uc,1} \ldots T_{uc,n}, T_{uo,1} \ldots T_{uo,n})$, if there is a collection of subsystems $\mathcal{C} \subseteq \{1, 2, \ldots n\}$, $\mathcal{C} \neq \emptyset$, such that the constraint is c-admissible*

[2]Self-loops do not arise as long as we limit ourselves to the constraints of the type (1).

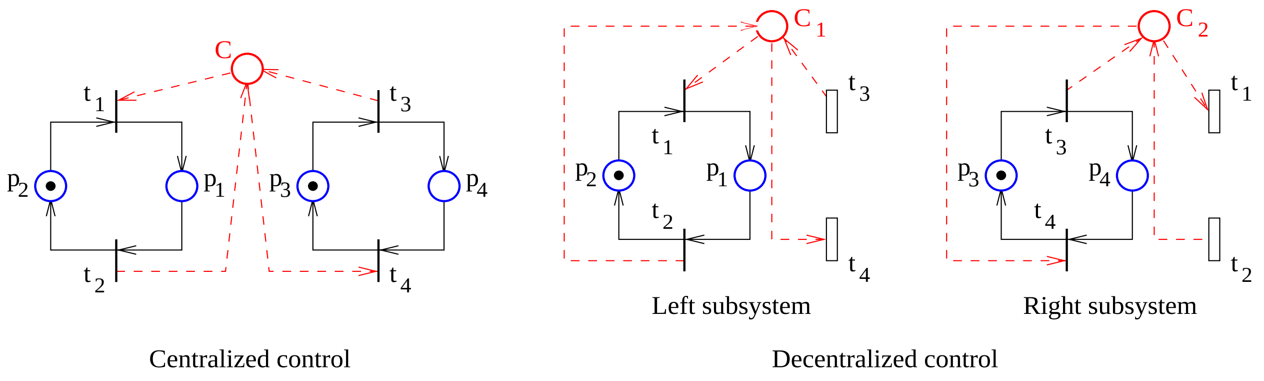


Figure 5: Centralized control versus decentralized control.

with respect to $(\mathcal{N}, \mu_0, T_{uc}, T_{uo})$*, where* $T_{uc} = \bigcap_{i\in\mathcal{C}} T_{uc,i}$ *and* $T_{uo} = \bigcup_{i\in\mathcal{C}} T_{uo,i}$*. A set of constraints is* **d-admissible** *if each of its constraints is d-admissible.*

To illustrate the definition, assume that we have a constraint that is c-admissible only with respect to the first subsystem. Then, it is d-admissible, as we can select $\mathcal{C} = 1$. An interesting consequence is that when each subsystem has full observability of the net and every transition is controllable with respect to some subsystem, any constraint is d-admissible. This consequence is formally stated next.

Proposition 4.2 *Any set of constraints is d-admissible if* $T_{uo,i} = \emptyset$ *for all* $i = 1 \dots n$ *and* $\bigcap_{i=1\dots n} T_{uc,i} = \emptyset$*.*

The construction of a decentralized supervisor, given a d-admissible set of constraints, is illustrated on the Petri net of Figure 3. The mutual exclusion constraint

$$\mu_1 + \mu_3 \leq 1 \tag{5}$$

is to be enforced. The centralized control solution is shown in Figure 5. In the case of decentralized supervision, there are two subsystems: the first one has $T_{uo,1} = \emptyset$ and $T_{uc,1} = \{t_3, t_4\}$, and the other has $T_{uo,2} = \emptyset$ and $T_{uc,2} = \{t_1, t_2\}$. Note that (5) is not c-admissible with respect to any of $(\mathcal{N}, T_{uc,1}, T_{uo,1})$ or $(\mathcal{N}, T_{uc,2}, T_{uo,2})$. However, it is d-admissible for $\mathcal{C} = \{1, 2\}$. Given two variables $x_1, x_2 \in \mathbb{N}$, a decentralized supervisor $\mathcal{S}_1 \wedge \mathcal{S}_2$ enforcing (5) can be defined by the following rules:

The supervisor $\mathcal{S}_1$:

- initialize x_1 to 0.
- disable t_1 if $x_1 = 0$
- increment x_1 if t_2 or t_3 fires.
- decrement x_1 if t_1 or t_4 fires.

The supervisor $\mathcal{S}_2$:

- initialize x_2 to 0.
- disable t_4 if $x_2 = 0$
- increment x_2 if t_2 or t_3 fires.
- decrement x_2 if t_1 or t_4 fires.

A graphical representation of $\mathcal{S}_1$ and $\mathcal{S}_2$ is possible, as shown in Figure 5. Thus, $\mathcal{S}_1$ is represented by C_1 and $\mathcal{S}_2$ by C_2; x_1 is the marking of C_1 and x_2 the marking of C_2. Graphically, C_1 and C_2 are copies of the control place C of the centralized supervisor. However, as discussed earlier in the section, (C_1, t_4) and (C_2, t_1) represent observation arcs. This corresponds to the fact that $\mathcal{S}_1$ never disables t_4 and $\mathcal{S}_2$ never disables t_1. As C_1 and C_2 have the same initial marking as C, their markings stay equal at all times. So, whenever t_1 should be disabled, the disablement action is implemented by C_1, and whenever t_4 is to be disabled, the disablement action is implemented by C_2.

In the general case, the construction of a supervisor enforcing a d-admissible constraint $l\mu \leq c$ ($l \in \mathbb{N}^{1\times|P|}$ and $c \in \mathbb{N}$) is as follows: (Note that the notation of Definition 4.1 is used)

Algorithm 4.3 *Supervisor Design for a D-admissible Constraint*

1. Let μ_0 the initial marking of $\mathcal{N}$, C the control place of the centralized SBPI supervisor $\mathcal{N}_s = (P_s, T, F_s, W_s)$ enforcing $l\mu \leq c$, and $\mathcal{C}$ the set of Definition 4.1.

2. For all $i \in \mathcal{C}$, let $x_i \in \mathbb{N}$ be a state variable of $\mathcal{S}_i$.

3. Define $\mathcal{S}_i$, for all $i \in \mathcal{C}$, by the following rules:

   - Initialize x_i to $c - l\mu_0$.
   - If $t \in T_{c,i}$, $t \in C\bullet$ and $x_i < W_s(C,t)$, then $\mathcal{S}_i$ disables t.
   - If t fires, $t \in T_{o,i}$ and $t \in \bullet C$, then $x_i = x_i + W_s(t,C)$.
   - If t fires, $t \in T_{o,i}$ and $t \in C\bullet$, then $x_i = x_i - W_s(C,t)$.

To enforce a d-admissible set of constraints $L\mu \leq b$, the construction above is repeated for each constraint $l\mu \leq c$. Note that in the graphical representation of the supervisors $\mathcal{S}_i$ corresponds to $|\mathcal{C}|$ copies of the control place C of the centralized supervisor, where each copy has the same initial marking as C.

Next we prove that the resulting decentralized supervisor is feasible (physically implementable) and as performant as the centralized supervisor. First, we formalize the feasibility concept. The decentralized supervisor $\bigwedge_{i\in\mathcal{C}} \mathcal{S}_i$ is said to be **feasible** if for all reachable markings μ_c of the closed-loop and for all transitions t: (i) for all $i = 1 \ldots n$, if t is closed-loop enabled and $t \notin T_{o,i}$, firing t does not change the state (marking) of $\mathcal{S}_i$; (ii) if t is plant-enabled but not closed-loop enabled, there is an $\mathcal{S}_i$ disabling t such that $t \in T_{c,i}$.

Theorem 4.4 *The decentralized supervisor constructed in Algorithm 4.3 is feasible, enforces the desired constraint, and is as permissive as the centralized supervisor of* $(\mathcal{N}, T_{uc}, T_{uo})$.

Proof: Feasibility is an immediate consequence of the construction of Algorithm 4.3. To prove the remaining part of the theorem, we show that a firing sequence σ is enabled by the centralized supervisor at the initial marking iff enabled by the decentralized supervisor at the initial marking. The proof uses the notation of the Algorithm 4.3 and of Definition 4.1. In addition, let $\mathcal{S}$ be the centralized supervisor implemented by the control place C, and $\mathcal{S}_d$ the decentralized supervisor $\bigwedge_{i\in C} \mathcal{S}_i$. Given a firing sequence $\sigma = t_{i_1}t_{i_2}\dots t_{i_k}$ enabled from μ_0 in the open-loop $(\mathcal{N}, \mu_0)$, we denote by μ_j the markings reached while firing σ: $\mu_0 \xrightarrow{t_{i_1}} \mu_1 \xrightarrow{t_{i_2}} \mu_2 \xrightarrow{t_{i_3}} \dots \mu_k$.

First, note that for all firing sequences $\sigma = t_{i_1}t_{i_2}\dots t_{i_k}$ enabled by both $\mathcal{S}$ and $\mathcal{S}_d$ from μ_0, we have that at all markings μ_j reached while firing σ

$$x_i = c - l\mu_j \quad \forall i \in \mathcal{C} \tag{6}$$

This is proven by induction. For $i = 0$, (6) is satisfied, due to the way the variables x_i are initialized. Assume (6) satisfied for $j < k$. According to the SBPI, when the plant has the marking μ_j the marking of C is $c - l\mu_j$, the same as x_i $\forall i \in \mathcal{C}$. In view of Definition 4.1, the d-admissibility of $l\mu \le c$ implies that $\mathcal{S}$ is c-admissible with respect to $(\mathcal{N}, \mu_0, T_{uc}, T_{uo})$. Then, since t_{i_j} is not dead, $t_{i_j} \notin T_{uc}$. Then, $t_{i_j} \notin T_{uc} \Rightarrow (\forall i \in \mathcal{C})\ t_{i_j} \notin T_{uo,i}$. Hence t_{i_j} is observable to all $\mathcal{S}_i$, and so all x_i are changed in the same way. Moreover, according to the SBPI, firing t_{i_j} changes the marking of C the same way as x_i are changed. From the SBPI we know that the new marking of C is $c - l\mu_{j+1}$. It follows that when μ_{j+1} is reached, $x_i = c - l\mu_{j+1}$ $\forall i \in \mathcal{C}$.

Finally, we prove by contradiction that the firing sequences enabled by $\mathcal{S}$ from μ_0 are the firing sequences enabled by $\mathcal{S}_d$ from μ_0. Assume the contrary, that there is σ that is enabled by one supervisor and not enabled by the other. We decompose σ into $\sigma = \sigma_x t_x \sigma_y$, $t_x \in T$, where σ_x is enabled by both supervisors and $\sigma_x t_x$ is not. If $\mu_0 \xrightarrow{\sigma_x} \mu_x$, then (6) is satisfied at $\mu_j = \mu_x$; the marking of C is also $c - l\mu_x$. There are two cases: (a) t_x enabled by C; (b) t_x not enabled by C. As in the previous part of the proof, case (a) leads to the conclusion that $\mathcal{S}_d$ enables also t_x, which contradicts the assumption that not both $\mathcal{S}$ and $\mathcal{S}_d$ enable t_x. In case (b), according to the SBPI, we have that $W_s(C, t_x) < c - l\mu_x$ and $t_x \notin T_{uc}$, by the d-admissibility of $l\mu \le c$. It follows that there is $i \in \mathcal{C}$ such that $\mathcal{S}_i$ disables t_x, and hence that $\mathcal{S}_d$ does not enable t_x. This contradicts the fact that one of $\mathcal{S}$ and $\mathcal{S}_d$ enables t_x. □

Next we turn our attention to checking whether a constraint is d-admissible. Let $\mathcal{S}$ be the centralized supervisor that enforces the constraint in the fully controllable and observable version of $\mathcal{N}$. Let T_{uo}^M be the set of transitions that are *not* detected by $\mathcal{S}$ and T_{uc}^M the set of transitions that are *not* controlled by $\mathcal{S}$.

Algorithm 4.5 *Checking whether a Constraint is D-admissible*

1. Find T_{uo}^M and T_{uc}^M.

2. Find the largest set of subsystems $\mathcal{C}$ such that $\forall i \in \mathcal{C}$: $T_{uo,i} \subseteq T_{uo}^M$.

3. If $\mathcal{C} = \emptyset$, declare that the constraint is not d-admissible and exit.

4. Define $T_{uc} = \bigcap_{i \in \mathcal{C}} T_{uc,i}$.
5. Does T_{uc} satisfy $T_{uc} \subseteq T_{uc}^M$? If yes, declare the constraint d-admissible. Otherwise, declare that the constraint is not d-admissible.

In the algorithm above, as long as a constraint is d-admissible, the constraint can be implemented for a minimal set $\mathcal{C}_{min} \subseteq \mathcal{C}$ containing the minimal number of subsystems such that $T_{uc}^M \supseteq \bigcap_{i \in \mathcal{C}_{min}} T_{uc,i}$.

Note that checking whether a set of constraints is d-admissible involves checking each constraint individually. The reason the algorithm checks single constraints is that checking sets of constraints as a whole may cause d-admissible sets of constraints to be declared not d-admissible. To see this, note that the overall set T_{uo}^M of a d-admissible set of constraints may be empty, which would cause the algorithm to declare the constraints not d-admissible (see step 3). Indeed, $T_{uo}^M = \emptyset$ is possible in spite of d-admissibility since $T_{uo}^M = \bigcap_i T_{uo}^{i,M}$, where $T_{uo}^{i,M}$ is the T_{uo}^M parameter for the i'th constraint of the set of constraints.

Proposition 4.6 *The algorithm checking d-admissibility is correct.*

Proof: It is sufficient to prove that the algorithm declares a constraint d-admissible only if it is d-admissible, and that all d-admissible constraints are declared d-admissible. Let $T_{uo} = \bigcup_{i \in \mathcal{C}} T_{uo,i}$. By construction, $T_{uo} \subseteq T_{uo}^M$.

A constraint is declared d-admissible if $\mathcal{C} \neq \emptyset$ and $T_{uc} \subseteq T_{uc}^M$. The definition of T_{uo}^M and T_{uc}^M implies that the constraint is c-admissible with respect to $(\mathcal{N}, T_{uc}, T_{uo})$. Then, in view of Definition 4.1, the algorithm is right to declare the constraint d-admissible.

Next, assume a d-admissible constraint. Then, there is a set of subsystems $\mathcal{C}' \neq \emptyset$ such that the constraint is c-admissible with respect to $(\mathcal{N}, T'_{uc}, T'_{uo})$ (where $T'_{uc} = \bigcap_{i \in \mathcal{C}'} T_{uc,i}$ and $T'_{uo} = \bigcup_{i \in \mathcal{C}'} T_{uc,i}$). Then $T'_{uo} \subseteq T_{uo}^M$; $T'_{uo} \subseteq T_{uo}^M \Rightarrow \mathcal{C}' \subseteq \mathcal{C} \Rightarrow T_{uc} \subseteq T'_{uc} \Rightarrow T_{uc} \subseteq T_{uc}^M$. Consequently, the algorithm declares the constraint to be d-admissible. □

In general, it may be difficult to compute the sets T_{uc}^M and T_{uo}^M. Then estimates $T_{uc}^e \subseteq T_{uc}^M$ and $T_{uo}^e \subseteq T_{uo}^M$ can be used in the algorithm instead. In this case the algorithm only checks a sufficient condition for d-admissibility, and so it can no longer detect constraints that are not d-admissible. In the case of the SBPI, such estimates can be found from the structural admissibility test of [10], stating that $L\mu \leq b$ is c-admissible if $LD_{uc} \leq 0$ and $LD_{uo} = 0$, where D_{uc} and D_{uo} are the restrictions of D to the columns of T_{uc} and T_{uo}.

Note that when it is possible and convenient to communicate in a reliable fashion with each subsystem of a decentralized system, a centralized solution with $T_{uc} = \bigcap_{i=1\ldots n} T_{uc,i}$ and $T_{uo} = \bigcap_{i=1\ldots n} T_{uo,i}$ is possible. Finally, note that in the implementation of d-admissible constraints, each supervisor $\mathcal{S}_i$ with $i \in \mathcal{C}$ relies on the proper operation of the other supervisors $\mathcal{S}_j$ with $j \in$

$\mathcal{C}$. By itself, a local supervisor may not be able to implement a d-admissible constraint or its implementation may be overrestrictive. For instance, in the example of Figure 3, the supervisor of the first subsystem can only enforce $\mu_1+\mu_3 \leq 1$ by itself by enforcing $\mu_1 = 0$. However, this solution is overrestrictive. D-admissibility illustrates the fact that more can be achieved when supervisors cooperate to achieve a given task, rather than when a supervisor tries on its own to achieve it (cf. "two heads better than one" in [14]).

5 Supervision with Communication

Obviously, communication can be used to change the attributes of otherwise inaccessible transitions to observable or even controllable. We begin with an illustration.

5.1 Illustration

As an illustration, consider again the robotic system of Figure 2. We assume that the computers controlling the two robots are able to communicate through a network connection. The specification is that the robots should not access at the same time the parts bin. By requiring each of the computers to signal any transition firing in the subsystem it controls, the sets of observable transitions become $T_{o,1} = T_{o,2} = \{t_1, t_2, t_3, t_4\}$. Then the decentralized supervisory solution of Figure 5 can be applied.[3]

The realization of a program implementing a local supervisor is illustrated on the left subsystem. The marking of C_1 may be implemented by a variable c_1. Each time the right subsystem signals that t_3 fires, c_1 is incremented, and each time the right subsystem announces that t_4 fires, c_1 is decremented. Furthermore, t_1 is the only transition controlled by the left subsystem; t_1 is allowed to fire only when $c_1 \geq 1$. When t_1 fires, the right subsystem is announced and c_1 is decremented. When t_2 fires, the right subsystem is announced and c_1 is incremented.

5.2 Decentralized Supervisors with Communication

The purpose of communication is to reduce the set of unobservable transitions $T_{uo,i}$ such that, if possible, the given constraints are c-admissible with respect to $(\mathcal{N}, T_{uc}, T_{uo})$. Note that communication cannot reduce T_{uo} below the attainable lower bound $T_{uo,L} \subseteq T_{uo}$, where $T_{uo,L} = \bigcap_{i=1\ldots n} T_{uo,i}$. T_{uc} can be changed by selecting a different set $\mathcal{C}$. However, it cannot be reduced below $T_{uc,L} = \bigcap_{i=1\ldots n} T_{uc,i}$. Indeed, $T_{uc,L}$ ($T_{uo,L}$) is the set of transitions uncontrollable (unobservable) in all subsystems.

[3]Note that in this example an arbitration procedure should be available, to ensure that a transition in the left subsystem does not fire at the same time as one in the right subsystem. Such an arbitration method could be, for instance, that transitions in the left subsystem may fire only at a time $t_0 + 2k\delta$, while transitions in the right subsystem only at a time $t_0 + (2k+1)\delta$. Furthermore, the communication between the two computers is to be reliable (e.g. no lost transition-firing messages).

Algorithm 5.1 *Decentralized Supervisor Design*

1. Is the specification admissible with respect to $(\mathcal{N}, T_{uc,L}, T_{uo,L})$? If not, transform it to be admissible (an approach of [10] could be used) or use the decentralized design approach of section 6.

2. Let $\mathcal{S}$ be the centralized SBPI supervisor enforcing the specification. Let T_c be the set of transitions controlled by $\mathcal{S}$ and T_o the set of transitions detected by $\mathcal{S}$.

3. Find a set $\mathcal{C}$ such that $T_{uc} = \bigcap_{i \in \mathcal{C}} T_{uc,i} \subseteq T \setminus T_c$.[4]

4. Design the decentralized supervisor by applying Algorithm 4.3 to $\mathcal{N}$ and $\mathcal{C}$.

5. The communication can be designed as follows: for all $t \in T_o \cap (\bigcup_{i \in \mathcal{C}} T_{uo,i})$, a subsystem j such that $t \in T_{o,j}$ transmits the firings of t to all supervisors $\mathcal{S}_k$ with $t \in T_{uo,k}$ and $k \in \mathcal{C}$.

Note the following. First, no communication arises when $T_o \cap (\bigcup_{i \in \mathcal{C}} T_{uo,i}) = \emptyset$. Second, the algorithm does not take in account communication limitations, such as bandwidth limitations of the communication channel. Bandwidth limitations can be considered in the approach considered next in section 6. Third, in this solution communication is used only to make some locally unobservable transitions observable; there is no remote control of locally uncontrollable transitions. Fourth, this solution tends to require less communication than a centralized solution. Indeed, a central supervisor not only needs to send the control decisions to the local subsystems, but also to remotely observe *all* transitions in T_o. Fifth, the main limitation of the algorithm is that in the case of inadmissible specifications, the transformation at the step 1 may result in constraints that are too restrictive. If so, the alternative solution we propose in section 6 could be used. Finally, the only way the algorithm can fail is at step 1, when the specification is inadmissible and the transformations to an admissible form fail.

Proposition 5.2 *The decentralized supervisor is feasible and equally permissive to the centralized supervisor $\mathcal{S}$ enforcing the specification on $(\mathcal{N}, T_{uc}, T_{uo,L})$.*

Proof: Since $\mathcal{S}$ is admissible, $T_c \cap T_{uc} = \emptyset$ and $T_o \cap T_{uo,L} = \emptyset$. Communication ensures that the sets of locally unobservable transitions become $T'_{uo,i} = T_{uo,i} \setminus T_o$. It follows that the specification is d-admissible with respect to $(\mathcal{N}, T_{uc,1}, \ldots T_{uc,n}, T'_{uo,1}, \ldots T'_{uo,n})$ and so the conclusion follows by Theorem 4.4. □

[4]At least one solution exists, $\mathcal{C} = \{1 \ldots n\}$. This can be seen from the fact that $\mathcal{S}$ admissible w.r.t. $(\mathcal{N}, T_{uc,L}, T_{uo,L})$ implies $T_{uc,L} \cap T_c = \emptyset$, and from $T_{uc,L} = \bigcap_{i=1\ldots n} T_{uc,i}$.

6 Constraint Transformations for Supervisor Design

Given a d-admissible set of constraints, a supervisor enforcing it can be easily constructed, as shown in Algorithm 4.3. This section considers transformations of sets of constraints that are not d-admissible. These transformations aim to obtain (more restrictive) d-admissible constraints, in order to reduce the problem to the enforcement of d-admissible constraints. Two approaches are proposed: transformations to single sets of constraints and transformations to multiple sets of constraints. The former is a particular case of the latter, and can be done using techniques from the literature [10, 15]. As the transformation to a single set of constraints cannot deal effectively with some interesting problems, we will focus on the transformation to multiple sets of constraints. This approach will be presented in both supervisory frameworks, with communication and with no communication.

6.1 Transformation to a single set of constraints

A possible approach to transform a set of constraints to a d-admissible set of constraints is:

1. Select a nonempty subset $\mathcal{C}$ of $\{1, 2, \ldots n\}$.
2. Transform[5] the set of constraints to a c-admissible set of constraints with respect to $(\mathcal{N}, T_{uc}, T_{uo})$, for $T_{uc} = \bigcap_{i \in \mathcal{C}} T_{uc,i}$ and $T_{uo} = \bigcup_{i \in \mathcal{C}} T_{uo,i}$.

In practice, it may not be trivial to select the “best” set $\mathcal{C}$. However, for some particular cases the choice of $\mathcal{C}$ is more obvious:

- If $T_{uo,1} = T_{uo,2} = \ldots T_{uo,n}$ (in particular, this is true when full observability is available in each subsystem: $T_{uo,i} = \emptyset\ \forall i = 1 \ldots n$), then $\mathcal{C}$ can be chosen as $\mathcal{C} = \{1, 2, \ldots n\}$, to minimize the number of transitions in T_{uc}.
- If $T_{o,i} \cap T_{o,j} = \emptyset$ for all distinct $i, j = 1 \ldots n$, then we could attempt to set $\mathcal{C}$ to each of $\{1\}$, $\{2\}$, $\ldots$ $\{n\}$, do in each case the transformation to admissible constraints, and then select the one yielding the least restrictive constraints.

The main drawback of this approach is that it fails for many interesting systems and constraints. For instance, it fails to provide a solution for the system of Figure 6, with $T_{uc,2} = T_{uo,2} = \{t_1, t_2\}$, $T_{uc,1} = T_{uo,1} = \{t_3, t_4\}$, initial marking as shown in figure, and specification

$$\mu_1 + \mu_3 \leq 2 \tag{7}$$

Indeed, no matter how $\mathcal{C}$ is chosen, no d-admissible inequality implying (7) is satisfied by the initial

[5]Techniques that can be used to perform this transformation appear in [10, 15].

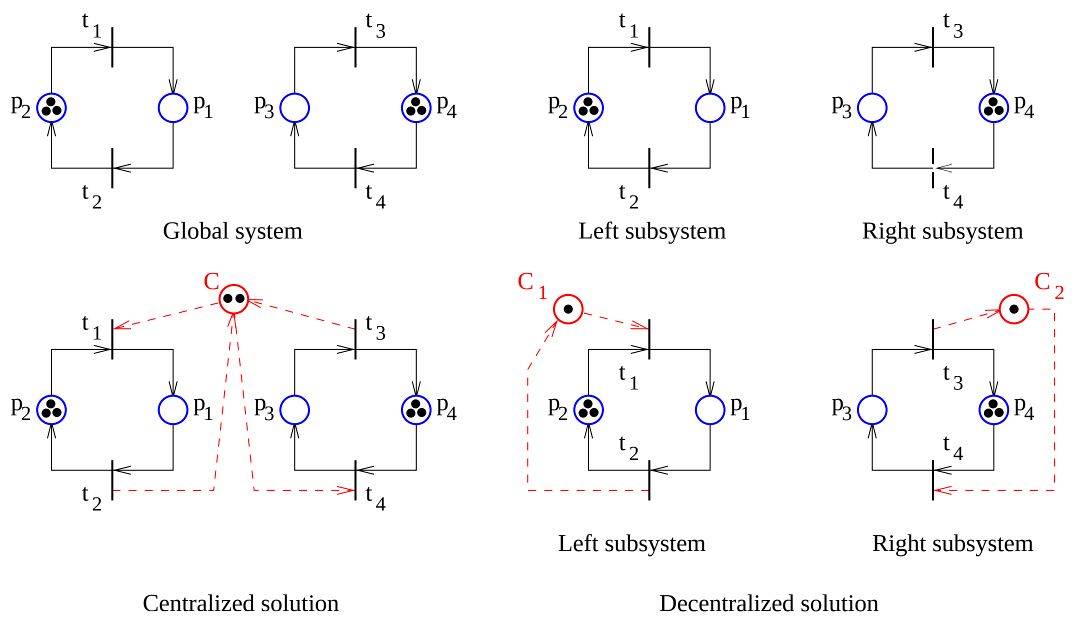


Figure 6: Decentralized control example.

marking. However, it is possible to enforce (7) with two d-admissible inequalities

$$\mu_1 \leq 1 \tag{8}$$

$$\mu_3 \leq 1 \tag{9}$$

as shown in Figure 6. To see that (8) and (9) are d-admissible, note that (8) satisfies Definition 4.1 for $\mathcal{C} = \{1\}$, and (9) satisfies Definition 4.1 for $\mathcal{C} = \{2\}$. Note also that none of (8) and (9), by itself, implies (7). This example motivates the transformation to multiple constraints, which is presented next.

6.2 Transformation to multiple sets of constraints

The problem can be stated as follows: *Given a set of constraints $L\mu \leq b$ that is not d-admissible, find d-admissible sets of constraints $L_1\mu \leq b_1 \ \ldots \ L_m\mu \leq b_m$ such that*

$$(L_1\mu \leq b_1 \wedge L_2\mu \leq b_2 \wedge \ldots L_m\mu \leq b_m) \Rightarrow L\mu \leq b \tag{10}$$

Compared to the previous approach, we now use several sets $\mathcal{C}_1$, $\mathcal{C}_2$, ..., $\mathcal{C}_m$ to design each of the $L_1\mu \leq b_1$, $L_2\mu \leq b_2$, ..., $L_m\mu \leq b_m$, instead of a single set $\mathcal{C}$. For instance, if $T_{o,i} \cap T_{o,j} = \emptyset$ for all $i \neq j$, then we may take $\mathcal{C}_i = \{i\}$, for all i. Furthermore, note that this framework includes the case when not all constraints $L_i\mu \leq b_i$ are necessary to implement $L\mu \leq b$, by allowing $L_i = 0$ and $b_i = 0$.

In general, (10) may have many solutions, not all equally interesting. In order to have a more interesting solution, we can use a set of markings of interest $\mathcal{M}_I$, and constrain each L_i and b_i to satisfy $L_i\mu \leq b_i \ \forall \mu \in \mathcal{M}_I$. This condition can be written as

$$L_i M \leq b_i \mathbf{1}^T \tag{11}$$

where $\leq$ means that each element of L_iM is less or equal to the element of the same indices in $b_i\mathbf{1}^T$, M is a matrix whose columns are the markings of interest, and $\mathbf{1}^T$ is a row vector of appropriate dimension in which all elements are 1.

The problem is more tractable if we replace (10) with the stronger condition below:

$$[(\alpha_1 L_1 + \alpha_2 L_2 + \ldots \alpha_m L_m)\mu \leq (\alpha_1 b_1 + \alpha_2 b_2 + \ldots \alpha_m b_m)] \Rightarrow L\mu \leq b \tag{12}$$

where α_i are nonnegative scalars.[6] Without loss of generality, (12) assumes that $L_1 \ldots L_m$ have the same number of rows. Again, without loss of generality, (12) can be replaced by

$$[(L_1 + L_2 + \ldots L_m)\mu \leq (b_1 + b_2 + \ldots b_m)] \Rightarrow L\mu \leq b \tag{13}$$

We further simplify our problem to

$$\begin{aligned} L_1 + L_2 + \ldots L_m &= R_1 + R_2 L & (14)\\ b_1 + b_2 + \ldots b_m &= R_2(b + \mathbf{1}) - \mathbf{1} & (15)\end{aligned}$$

for R_1 with nonnegative integer elements and R_2 diagonal with positive integers on the diagonal. Note that $[(R_1 + R_2 L)\mu \leq R_2(b+\mathbf{1}) - \mathbf{1}] \Rightarrow L\mu \leq b$ has been proved in [10].

It is known that a sufficient condition for the c-admissibility of a set of constraints $L\mu \leq b$ is that $LD_{uc} \leq 0$ and $LD_{uo} = 0$, where D_{uc} and D_{uo} are the restrictions of the incidence matrix D to the sets of uncontrollable and unobservable transitions [10]. The admissibility requirements in our setting can then be written as

$$\begin{aligned} L_i D_{uc}^{(i)} &\leq 0 & (16)\\ L_i D_{uo}^{(i)} &= 0 & (17)\end{aligned}$$

where $D_{uc}^{(i)}$ and $D_{uo}^{(i)}$ are the restrictions of D to the sets $T_{uc}^{(i)} = \bigcap_{i \in \mathcal{C}_i} T_{uc,i}$ and $T_{uo}^{(i)} = \bigcup_{i \in \mathcal{C}_i} T_{uo,i}$. Then our problem becomes: *find a feasible solution of (11) and (14–17).* The unknowns are R_1, R_2, L_i, and b_i, and integer programming can be used to find them. The next result is an immediate consequence of our considerations above.

Proposition 6.1 *Any sets of constraints $L_i\mu \leq b_i$ satisfying (11) and (14–17) are d-admissible and $\bigwedge_{i=1\ldots n} [L_i\mu \leq b_i] \Rightarrow L\mu \leq b$.*

[6]In the literature, a relaxation of a hard problem that is similar to the relaxation from (10) to (12) is the S-procedure mentioned in [18] at page 62.

6.3 Decentralized control with communication

So far, we have ignored the possibility that the local supervisors $\mathcal{S}_i$ may have the ability to communicate. We now consider the case in which the supervisors are able to communicate the firings of certain transitions. Communication is useful, as it relaxes the admissibility constraints (16) and (17) by reducing the number of uncontrollable and unobservable transitions. However, communication constraints may be present, and bandwidth limitations may encourage the minimization of the communication over the network. The analysis of this section, without being comprehensive, serves as an illustration of the fact that such problems can be approached in this framework.

For each set $\mathcal{C}_i$ and transition t_j, let α_{ij} be a binary variable, where $\alpha_{ij} = 1$ if the firing of t_j is made known to the subsystems in $\mathcal{C}_i$. Note that we have the following constraints:

$$\forall t_j \in T_{uo,L} : \alpha_{ij} = 0 \tag{18}$$

where $T_{uo,L} = \bigcap_{i=1\ldots n} T_{uo,i}$ is the set of transitions that cannot be observed anywhere in the system. ($T_{uo,L}$ is the set of transitions whose firing cannot be communicated.)

Let B_L^i and B_U^i be lower and upper bounds of $L_i D$ and $A = [\alpha_{ij}]$ be the matrix of elements α_{ij}. Given a vector v and a matrix M, let $diag(v)$ denote the diagonal matrix of diagonal v, $M(k, \cdot)$ the k'th row of M, and $M|_{T_{uo}^{(i)}}$ the restriction of M to the transitions of $T_{uo}^{(i)}$ (i.e. $M|_{T_{uo}^{(i)}}$ contains the columns $M(\cdot, j)$ such that $t_j \in T_{uo}^{(i)}$). We require

$$L_i D_{uo}^{(i)} \leq [B_U^i diag(A(i,\cdot))]|_{T_{uo}^{(i)}} \tag{19}$$
$$L_i D_{uo}^{(i)} \geq [B_L^i diag(A(i,\cdot))]|_{T_{uo}^{(i)}} \tag{20}$$

instead of $L_i D_{uo}^{(i)} = 0$. In this way, the admissibility requirement $L_i D_{uo}^{(i)} = 0$ is relaxed by eliminating the constraints corresponding to the transitions of $T_{uo}^{(i)}$ that have their firings communicated to the subsystems of $\mathcal{C}_i$.

Similarly, (16) can also be relaxed by communicating enabling decisions of supervisors. Naturally, for each transition t it controls, each supervisor $\mathcal{S}_i$ has two enabling decisions: *enable* and *disable*. They depend on whether all control places C of $\mathcal{S}_i$ that are connected to t satisfy $\mu_c(C) \geq W_s(C, t)$ or not. A communication policy may be that a supervisor announces a remote actuator each time its enabling decision changes. Then the actuator can determine its enabling by taking the conjunction of the decisions corresponding to all supervisors controlling it. In our setting, d-admissibility implies that the supervisors within a cluster $\mathcal{C}_i$ have always the same enabling decisions, and so only communication between clusters needs to be considered. Similarly to α_{ij}, we can introduce binary variables ε_{ij} describing the communication of enabling decisions pertaining to t_j. Thus, $\varepsilon_{ij} = 1$ if a supervisor from $\mathcal{C}_i$ communicates its enabling decisions to t_j. As in the case of α_{ij}, we have

$$\forall t_j \in T_{uc,L} : \varepsilon_{ij} = 0 \tag{21}$$

for $T_{uc,L} = \bigcap_{i=1\ldots n} T_{uc,i}$. Furthermore, if $E = [\varepsilon_{ij}]$, (16) becomes:

$$L_i D_{uc}^{(i)} \leq [B_U^i diag(E(i,\cdot))]|_{T_{uc}^{(i)}} \tag{22}$$

Communication constraints stating that certain transitions cannot be observed by communication or that certain transitions cannot be remotely controlled by communication, can be incorporated by setting coefficients α_{ij} and ε_{ij} to zero. Constraints limiting the average network traffic can be incorporated as constraints of the form:

$$\sum_i A(i,\cdot)g_i + \sum_i E(i,\cdot)h_i \leq p \tag{23}$$

where g_i and h_i are vectors of appropriate dimensions and p is a scalar. As an example, the elements of g_i could reflect average firing counts of the transitions over the operation of the system. Note that (23) can be written more compactly as

$$Tr(AG + EH) \leq p \tag{24}$$

where G and H are the matrices of columns g_i and h_i, and $Tr(M)$ denotes the trace of a matrix M.

We may also choose to minimize the amount of communication involved in the system. Then we can formulate our problem as

$$\min_{L_i, b_i, A, E, R_1, R_2} Tr(AC + EF) \tag{25}$$

where the weight matrices C and F are given, and the minimization is subject to the constraints (11), (14–15), (18–22), and $\alpha_{ij}, \varepsilon_{ij} \in \{0,1\}^{|T|}$. This problem can be solved using linear integer programming.

6.4 Liveness Constraints

One of the difficulties encountered with this approach is that the permissivity of the generated constraints is hard or impossible to be expressed in the cost function. Moreover, the generated constraints may cause parts of the system to unavoidably deadlock. This situation can be prevented by using a special kind of constraints, that we call liveness constraints.

A liveness constraint consists of a vector x such that for all i: $L_i x \leq 0$. A possible way to obtain such constraints is described next. Given a finite firing sequence σ, let x_σ be a vector such that $x_\sigma(i)$ is the number of occurrences of the transition t_i in σ. Given the Petri net of incidence matrix D and the constraints $L\mu \leq b$, let y be a nonnegative integer vector such that $Dy \geq 0$ and $-LDy \geq 0$. A vector y satisfying these inequalities has the following property. If σ is a firing sequence such that (a) σ can be fired without violating $L\mu \leq b$ and (b) $x_\sigma = y$, then σ can be fired infinitely often without violating $L\mu \leq b$. However, if the decentralized control algorithm generates a constraint $L_i\mu \leq b_i$ such that $L_i Dy \not\leq 0$, then any firing sequence σ having $x_\sigma = y$ cannot be infinitely often fired in the closed-loop. If such a situation is undesirable, the matrices L_i can be required to satisfy $L_i x \leq 0$ for $x = Dy$. An illustration will be given in section 8.

6.5 Reducing the Computational Complexity

Obviously, an integer programming approach limits the size of the problems that can be solved. A possible way to reduce the amount of computation is to solve several smaller integer programs instead of a large integer program. This approach is outlined next.

Algorithm 6.2

1. Let $L_0\mu \leq b_0$ be the given set of constraints.
2. Let $\mathcal{I} = \mathcal{J} = \emptyset$ and set $(L_{0,i}, b_{0,i})$ as empty sets of constraints for all $\mathcal{C}_i$, $i = 1 \dots m$.
3. **For** all inequalities $l\mu \leq \beta$ of the set of inequalities $L_0\mu \leq b_0$ **do**
    (a) Set $L = l$, $b = \beta$, and solve (25) subject to (11), (14–15), (18–22), $\alpha_{ij}, \varepsilon_{ij} \in \{0,1\}^{|T|}$, $\alpha_{ij} = 1\ \forall(i,j) \in \mathcal{I}$ and $\varepsilon_{ij} = 1\ \forall(i,j) \in \mathcal{J}$.
    (b) If the integer program is feasible, include (L_i, b_i) in $(L_{0,i}, b_{0,i})$ and set $\mathcal{I} = \{(i,j) : \alpha_{ij} \neq 0\}$ and $\mathcal{J} = \{(i,j) : \varepsilon_{ij} \neq 0\}$.
    (c) If the integer program is infeasible, exit, and declare failure.
4. The output are the constraints $(L_{0,i}, b_{0,i})$, while communication is to ensure that the firing of all transitions t_j with $(i,j) \in \mathcal{I}$ is announced to $\mathcal{C}_i$, and that for all $(i,j) \in \mathcal{J}$, a supervisor from $\mathcal{C}_i$ sends its enabling decisions to t_j.

Other computational savings can be obtained by taking advantage of knowledge on the reachable markings. Indeed, up to now our approach has relied exclusively on structural properties of the net. However, knowledge on the reachable markings can be used to reduce the number of transitions that need to be considered controllable and observable, based on reachability information and the c-admissibility definition. This is shown in Appendix B. Reducing the number of controllable or observable transitions results in a smaller number of constraints in the conditions (16) and (17).

Significant computational savings can be achieved when only a part of the elements of each L_i need to be calculated. For instance, we may choose to set the $j'th$ column of L_i to zero for all places p_j that are not part of the subsystem i.

The number of variables α_{ij} and ε_{ij} is significantly reduced when the communication policy is changed to broadcast. Then observed transitions and enabling decisions that are to be communicated are broadcasted. In this case $\alpha_{ij} = \alpha_j$ and $\varepsilon_{ij} = \varepsilon_j$, where $\alpha_j = 1$ means that a supervisor observing t_j broadcasts the firings of t_j, and $\varepsilon_j = 1$ means that any supervisor that remotely controls t_j broadcasts its enabling decisions for t_j.

7 Extensions

The main results presented so far in the paper consist of the definition of d-admissibility, algorithms for decentralized control, and proofs of correctness. It can be noticed that while the results of section 6 are more specialized, taking advantage of the fact that the constraints have the particular form $L\mu \leq b$, the results of sections 4 and 5 are general. Indeed, in order to apply the latter results to other types of constraints, we only need to replace the SBPI with the corresponding supervision techniques, and then change accordingly the c-admissibility definition. In this section two extensions of our results are discussed. First the extension to the supervision of generalized linear constraints [5] is discussed. Then, the extension to the automata setting is discussed. Note that while the first extension effectively enhances the practical use of our results, the second extension has an illustrative purpose. The practical significance of the specialization of our approach to the automata setting is a matter of further investigation.

7.1 Extensions to Generalized Linear Constraints

This section applies the results derived so far in the paper to generalized linear constraints [5]. This type of constraints generalizes the constraints $L\mu \leq b$ to the form

$$L\mu + Hq + Cv \leq b \tag{26}$$

where q, the firing vector, and v, the Parikh vector, are defined as follows. The firing vector q identifies a transition that is to fire by the position of its nonzero element; namely, if t_i is the transition to fire, $q_j = 1$ for $j = i$ and $q_j = 0\ \forall j \neq i$. The Parikh vector v consists of elements v_i that count the number of firings of each transition t_i since the initialization of the system.

A supervisor enforcing (26) ensures that (i) all reachable states (μ, v) of the plant satisfy $L\mu + Cv \leq b$ and (ii) a transition t_i can be fired from the state (μ, v) only if $L\mu + Hq^{(i)} + Cv \leq b$ and $L\mu' + Cv' \leq b$, where $q_j^{(i)} = 1$ if $j = i$ and $q_j^{(i)} = 0$ otherwise, μ' is the marking reached by firing t_i, and $v' = v + q^{(i)}$.

As shown in [5], in the fully controllable and observable case, a centralized supervisor enforcing (26) is defined by the input and output matrices

$$D_s^+ = D_{lc}^+ + \max(0, H - D_{lc}^-) \tag{27}$$
$$D_s^- = \max(D_{lc}^-, H) \tag{28}$$

where

$$D_{lc}^+ = \max(0, -LD - C) \tag{29}$$
$$D_{lc}^- = \max(0, LD + C) \tag{30}$$

The initial marking of the supervisor is[7]

$$\mu_{s0} = b - L\mu_0 \tag{31}$$

[7]Note that the initial v is zero.

Similarly to (4), the closed-loop system has the incidence matrix

$$D_c = \left[\begin{array}{c} D \\ D_s \end{array} \right] \tag{32}$$

where $D_s = D_s^+ - D_s^-$. Note that as in the case of the SBPI, the supervisor consists of *control places* connected to the transitions of the plant. Therefore, the definitions for detection and control in section 2 are still valid. So, the c-admissibility definitions can be adapted as follows. A supervisor enforcing (26) is *c-admissible* if it only controls controllable transitions and it only detects observable transitions. The set of constraints (26) is **c-admissible** if the supervisor defined by (27), (28) and (31) is c-admissible.

The results of the sections 4, 5 and 6 apply to specifications (26) as follows:

1. All results in the sections 4 and 5 apply once the references to the SBPI are replaced by references to the supervision method of (27), (28) and (31).

2. The material of section 6 can be applied to specifications in the form of constraints (26) after applying the C- and H-transformations defined in [5]. These transformations transform the Petri net and the constraints to reduce the problem to the enforcement of constraints $L_e\mu \leq b$.

7.2 Extension to Automata

In this section the results of sections 4 and 5 are specialized to the automata setting. Note that the results of section 6 are harder to extend, as they rely on a particular type of specifications. First, the notation is introduced.

An automaton is denoted by the tuple $G = (\Sigma, Q, \delta, q_0, Q_m)$, where Σ is the set of events, Q is the set of states, $\delta : Q \times \Sigma \to Q$ is a (partial) function representing the transition function, $q_0 \in Q$ is the initial state, and $Q_m \subseteq Q$ is the set of marked states. The empty symbol is denoted by ε. Let Σ^* denote the set of all strings over the alphabet Σ. Let $\delta^* : Q \times \Sigma^* \to Q$ be the (partial) function recursively defined as follows: $\forall q \in Q$: $\delta^*(q, \varepsilon) = q$ and $\forall q \in Q$ $\forall \alpha \in \Sigma$ $\forall \sigma \in \Sigma^*$: $\delta^*(q, \sigma\alpha) = \delta(\delta^*(q, \sigma), \alpha)$ when $\delta^*(q, \sigma)$ and $\delta(\delta^*(q, \sigma), \alpha)$ are defined. The fact that $\delta^*(q, \sigma)$ is defined is denoted as $\delta^*(q, \sigma)!$. The language accepted by G is $\mathcal{L}(G) = \{\sigma \in \Sigma^* : \delta^*(q_0, \sigma)!\}$, and the language marked by G is $\mathcal{L}_m(G) = \{\sigma \in \mathcal{L}(G) : \delta^*(q_0, \sigma) \in Q_m\}$.

Consider a prefix-closed specification $K \subseteq \mathcal{L}(G)$. The controllability and observability of K are defined with respect to the set of uncontrollable events $\Sigma_{uc} \subseteq \Sigma$ and the set of unobservable events $\Sigma_{uo} \subseteq \Sigma$. K is controllable if $K\Sigma_{uc} \cap \mathcal{L}(G) \subseteq K$. K is observable if $(\forall \alpha \in \Sigma\ \forall \sigma_1, \sigma_2 \in K)$ $[P(\sigma_1) = P(\sigma_2) \wedge \sigma_1\alpha, \sigma_2\alpha \in \mathcal{L}(G)] \Rightarrow [\sigma_1\alpha, \sigma_2\alpha \in K \vee \sigma_1\alpha, \sigma_2\alpha \notin K]$, where $P : \Sigma^* \to (\Sigma \setminus \Sigma_{uo})^*$ is the projection removing the elements of Σ_{uo} from any string.

A supervisor $\mathcal{S}$ of G is defined as a map $\mathcal{S} : (\Sigma \setminus \Sigma_{uo})^* \to 2^{\Sigma}$. To implement K, the supervisor $\mathcal{S}$ can be defined as follows: $\forall \sigma \in (\Sigma \setminus \Sigma_{uo})^*$, $\mathcal{S}(\sigma) = \{\alpha \in \Sigma : \exists \sigma_x \in P^{-1}(\sigma), \sigma_x\alpha \in K\}$. Then, if K is observable, the closed-loop language is $\mathcal{L}(\mathcal{S}/G) = K$. Note that the control action taken after the occurrence of $\sigma \in \mathcal{L}(G)$ in G is $\mathcal{S}(P(\sigma))$.

In a decentralized setting, G can be partitioned in n subsystems, each with the set of uncontrollable events $\Sigma_{uc,i} \subseteq \Sigma$ and the set of unobservable events $\Sigma_{uo,i} \subseteq \Sigma$, for $i = 1 \dots n$. Compared to the Petri net notation, we have G instead of $\mathcal{N}$, and $\Sigma_{uc,i}$ and $\Sigma_{uo,i}$ instead of $T_{uc,i}$ and $T_{uo,i}$. Thus the decentralized system is $(G, \Sigma_{uc,1}, \dots \Sigma_{uc,n}, \Sigma_{uo,1}, \dots \Sigma_{uo,n})$ instead of $(\mathcal{N}, T_{uc,1}, \dots T_{uc,n}, T_{uo,1}, \dots T_{uo,n})$.

C-admissibility can be defined as follows: K is c-admissible with respect to $(G, \Sigma_{uc}, \Sigma_{uo})$, if controllable and observable with respect to $(G, \Sigma_{uc}, \Sigma_{uo})$. The supervisor $\mathcal{S}$ enforcing K and defined as above is c-admissible if K is c-admissible.

D-admissibility extends as follows. The prefix-closed specification $K \subseteq \mathcal{L}(G)$ is **d-admissible** with respect to $(G, \Sigma_{uc,1}, \dots \Sigma_{uc,n}, \Sigma_{uo,1}, \dots \Sigma_{uo,n})$ if there is a collection of subsystems $\mathcal{C} \subseteq \{1, 2, \dots n\}$, $\mathcal{C} \neq \emptyset$, such that K is c-admissible with respect to $(G, \Sigma_{uc}, \Sigma_{uo})$, for $\Sigma_{uc} = \bigcap_{i \in \mathcal{C}} \Sigma_{uc,i}$ and $\Sigma_{uo} = \bigcup_{i \in \mathcal{C}} \Sigma_{uo,i}$. A decentralized supervisor enforcing the d-admissible specification K consists of the supervisors $\mathcal{S}_i : (\Sigma \setminus \Sigma_{uo,i})^* \to 2^{\Sigma}$, $i \in \mathcal{C}$, such that $\forall \sigma \in (\Sigma \setminus \Sigma_{uo,i})^*$: $\mathcal{S}_i(\sigma) = \Sigma_{uc,i} \cup \{\alpha \in \Sigma : \exists \sigma_x \in P_i^{-1}(\sigma),\ \alpha \in \mathcal{S}(P(\sigma_x))\}$, where P_i is the projection removing the elements of $\Sigma_{uo,i}$ and $\mathcal{S}$ is defined as a map $\mathcal{S} : (\Sigma \setminus \Sigma_{uo})^* \to 2^{\Sigma}$ such that $\forall \sigma \in (\Sigma \setminus \Sigma_{uo})^*$: $\mathcal{S}(\sigma) = \Sigma_{uc} \cup \{\alpha \in \Sigma : \exists \sigma_x \in P^{-1}(\sigma),\ \sigma_x \alpha \in K\}$. Note that $\Sigma_{uo,i} \subseteq \Sigma_{uo}$ implies $\forall \sigma \in \Sigma^*$: $P(P_i^{-1}(P_i(\sigma))) = P(\sigma)$, from which it can be seen that

$$(\forall \sigma \in \Sigma^*) \ \ \mathcal{S}_i(P_i(\sigma)) = \Sigma_{uc,i} \cup \mathcal{S}(P(\sigma)) \tag{33}$$

This construction of the supervisors $\mathcal{S}_i$ is the equivalent of Algorithm 4.3. To state also the equivalent of Theorem 4.4, we need to define feasibility. Note first that by defining the supervisors on $(\Sigma \setminus \Sigma_{uc,i})^*$ rather than on Σ^*, there is no need to include an observability requirement in the feasibility definition. Then, we say that $\mathcal{S}_i$ is feasible with respect to $(G, \Sigma_{uc,i}, \Sigma_{uo,i})$ if $\forall \sigma \in \mathcal{L}(\mathcal{S}_i/G)$ $\forall \alpha \in \Sigma_{uc}$: $\sigma\alpha \in \mathcal{L}(G) \Rightarrow \alpha \in \mathcal{S}_i(P_i(\sigma))$. Furthermore, we say that the decentralized supervisor is feasible if all its components $\mathcal{S}_i$ are feasible. Then Theorem 4.4 extends to:

Theorem 7.1 *The decentralized supervisor $\bigwedge_{i \in \mathcal{C}} \mathcal{S}_i$ is feasible and $\mathcal{L}(\bigwedge_{i \in \mathcal{C}} \mathcal{S}_i/G) = K$.*

Proof: The feasibility of $\mathcal{S}_i$ is satisfied because the way $\mathcal{S}_i$ has been defined guarantees $\forall \sigma \in \mathcal{L}(G)$: $\Sigma_{uc,i} \subseteq \mathcal{S}_i(\sigma)$.

For the second part of the proof, note that in view of (33) we have that for any $\sigma \in \Sigma^*$: $\bigwedge_{i \in \mathcal{C}} \mathcal{S}_i(P_i(\sigma)) = \bigcap_{i \in \mathcal{C}} (\Sigma_{uc,i} \cup \mathcal{S}(P(\sigma))) \Rightarrow \bigwedge_{i \in \mathcal{C}} \mathcal{S}_i(P_i(\sigma)) = \Sigma_{uc} \cup \mathcal{S}(P(\sigma)) \Rightarrow \bigwedge_{i \in \mathcal{C}} \mathcal{S}_i(P_i(\sigma)) = \mathcal{S}(P(\sigma))$. Therefore, $\mathcal{L}(\bigwedge_{i \in \mathcal{C}} \mathcal{S}_i/G) = \mathcal{L}(\mathcal{S}/G)$. Furthermore, by the observability and controllability of K with respect to $(G, \Sigma_{uc}, \Sigma_{uo})$ it follows that $\mathcal{L}(\mathcal{S}/G) = K$. □

As in Theorem 4.4, the decentralized supervisor has the same performance as the centralized supervisor, in the sense that $\mathcal{L}(\bigwedge_{i \in \mathcal{C}} \mathcal{S}_i/G) = \mathcal{L}(\mathcal{S}/G)$. However, the decentralized supervisor does not involve communication, while the implementation of a centralized supervisor in a decentralized setting requires communication.

The remaining results, namely Algorithm 4.5 and Algorithm 5.1, are harder to extend. The reason is that the correspondent of the set T_{uo}^M may not exist in the automata setting. However, for the purpose of illustration, we discuss the direct extensions of the two results for the case in which the set corresponding to T_{uo}^M exists. Σ_{uc}^M and Σ_{uo}^M, the correspondents of T_{uo}^M and T_{uc}^M, are defined as the largest subsets of Σ for which K is controllable and observable with respect to $(G, \Sigma_{uc}^M, \Sigma_{uo}^M)$. Algorithm 4.5, checking whether a specification is d-admissible, can be used without further modification, once the sets T_{uc}^M and T_{uo}^M are replaced with Σ_{uc}^M and Σ_{uo}^M. Algorithm 5.1, proposed for the design of decentralized supervisors with communication that enforce constraints that are not d-admissible, can also be extended:

Algorithm 7.2 *Decentralized Supervisor Design*

1. Let $\Sigma_{uc,L} = \bigcap_{i=1\dots n} \Sigma_{uc,i}$ and $\Sigma_{uo,L} = \bigcap_{i=1\dots n} \Sigma_{uo,i}$. Is the specification K controllable and observable with respect to $(G, \Sigma_{uc,L}, \Sigma_{uo,L})$? If yes, let $L = K$. If not, find a controllable and observable sublanguge L of K.

2. Let Σ_{uc}^M and Σ_{uo}^M be the largest subsets of Σ such that L is controllable and observable with respect to $(G, \Sigma_{uc}^M, \Sigma_{uo}^M)$. Let $\mathcal{S}$ be the centralized supervisor enforcing the specification L on $(G, \Sigma_{uc}^M, \Sigma_{uo}^M)$.

3. Find a set $\mathcal{C}$ such that $\Sigma_{uc} = \bigcap_{i\in\mathcal{C}} \Sigma_{uc,i} \subseteq \Sigma_{uc}^M$.[8]

4. The decentralized supervisor consists of the supervisors $\mathcal{S}_i$, for $i \in \mathcal{C}$, defined as $\mathcal{S}_i(\sigma) = \mathcal{S}(\sigma) \cup \Sigma_{uc,i}\ \forall \sigma \in \Sigma^*$.

5. The communication can be designed as follows: for all events $\alpha \in (\Sigma \setminus \Sigma_{uo}^M) \cap (\bigcup_{i\in\mathcal{C}} \Sigma_{uo,i})$, a subsystem j such that $\alpha \in \Sigma \setminus \Sigma_{uo,j}$ transmits the occurrences of α to all supervisors $\mathcal{S}_k$ with $\alpha \in \Sigma_{uo,k}$ and $k \in \mathcal{C}$.

As previously mentioned, the extensions of the Algorithms 4.5 and 5.1 require the sets Σ_{uc}^M and Σ_{uo}^M. In Appendix C it is shown that Σ_{uc}^M exists, however Σ_{uo}^M may not exist. Therefore, these extensions can be used when the structure of G guarantees the existence of Σ_{uo}^M.

8 Example

This section illustrates our approach on the manufacturing example from [9], shown in Figure 7. The system consists of two machines (M_1 and M_2), four robots ($H_1 \dots H_4$), and four buffers of finite capacity ($B_1 \dots B_4$). The events associated with the movement of the parts within the

[8]At least one solution exists, $\mathcal{C} = \{1 \dots n\}$. This can be seen from the fact that L controllable and observable w.r.t. $(G, \Sigma_{uc,L}, \Sigma_{uo,L})$ implies $\Sigma_{uc,L} \subseteq \Sigma_{uc}^M$, and from $\Sigma_{uc,L} = \bigcap_{i=1\dots n} \Sigma_{uc,i}$.

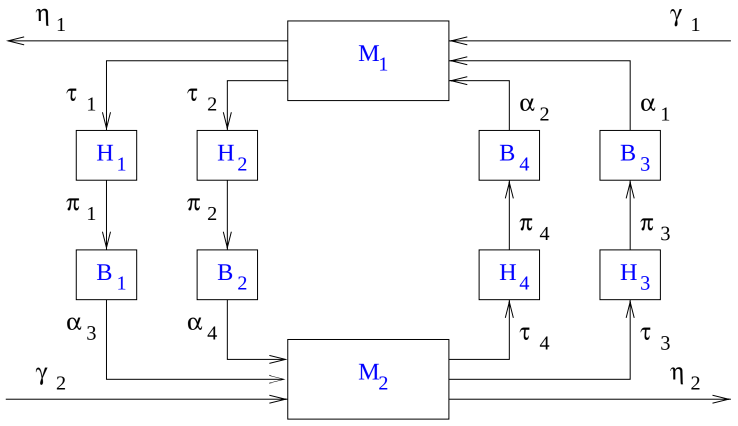


Figure 7: A manufacturing system.

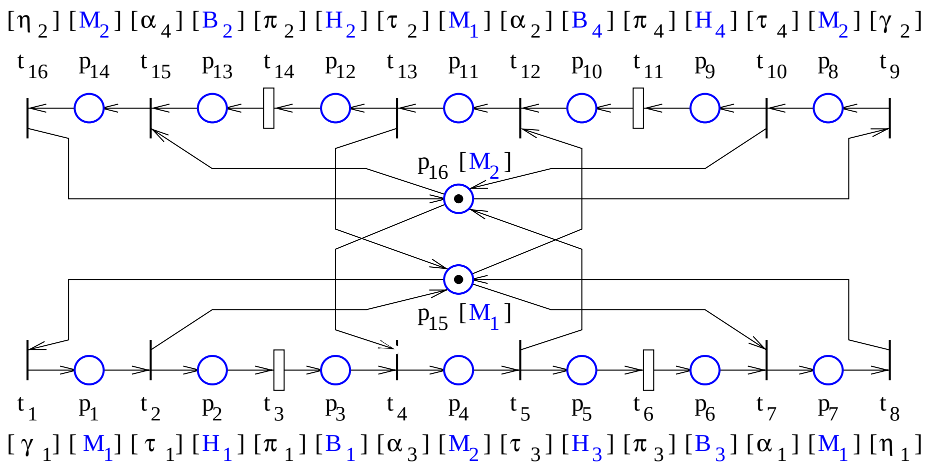

[η 2] [M2] [α 4] [B 2] [π 2] [H2] [τ 2] [M1] [α 2] [B 4] [π 4] [H 4] [τ 4] [M2] [γ 2]
t16 p14 t15 p13 t14 p12 t13 p11 t12 p10 t11 p9 t10 p8 t9
p16 [M2]
p15 [M1]
t1 p1 t2 p2 t3 p3 t4 p4 t5 p5 t6 p6 t7 p7 t8
[γ 1] [M1] [τ 1] [H1] [π 1] [B 1] [α 3] [M2] [τ 3] [H3] [π 3] [B 3] [α 1] [M1] [η 1]

Figure 8: Petri net model of the system.

system are marked with Greek letters. There are two types of parts. The manufacturing process of the first type of parts is represented by the following sequence of events: $\gamma_1\tau_1\pi_1\alpha_3\tau_3\pi_3\alpha_1\eta_1$. The manufacturing process of the second kind of parts is represented by $\gamma_2\tau_4\pi_4\alpha_2\tau_2\pi_2\alpha_4\eta_2$. These processes can be represented by the Petri net of Figure 8. In the Petri net, the transitions are labeled by the events they represent, and the places by the names of the manufacturing components. For instance, a token in p_{16} indicates that M_2 is idle, and a token in p_8 indicates that M_2 is working on a part of type 2 that has just entered the system. Furthermore, the number of parts in a buffer is the marking of the place modeling the buffer; for instance, μ_{13} represents the number of parts in B_2 at the marking μ. The number of parts the machines M_1 and M_2 can process at the same time is $\mu_1 + \mu_7 + \mu_{11} + \mu_{15} = n_1$ and $\mu_4 + \mu_8 + \mu_{14} + \mu_{16} = n_2$, respectively. In [9], $n_1 = n_2 = 1$.

The first supervisory requirements are that the buffers do not overflow. If the capacity of the

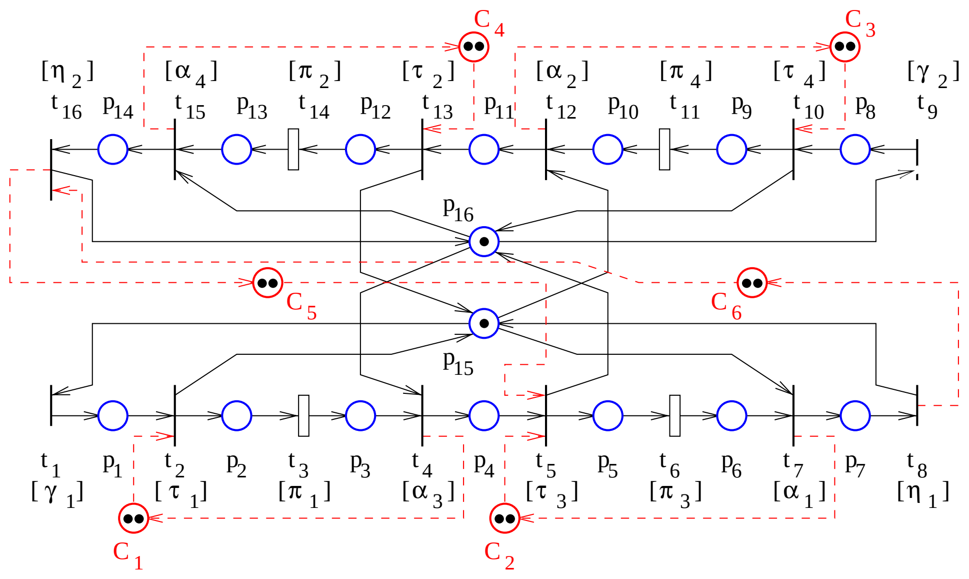


Figure 9: Decentralized supervision.

buffers is k, the requirement can be written as:

$$\mu_i \leq k \text{ for } i \in \{3, 6, 10, 13\} \tag{34}$$

In [9] the capacity of the buffers is $k = 2$. Another requirement is that the number of completed parts of type 1 is about the same as the number of completed parts of type 2:

$$v_8 - v_{16} \leq u \tag{35}$$

$$v_{16} - v_8 \leq u \tag{36}$$

where v_8 and v_{16} denote the number of firings of t_8 and t_{16}, respectively. In [9], $u = 2$. Note that constraints involving the vector v can be easily represented as marking constraints in a transformed Petri net [5].

Following [9], the constraints (34) are enforced assuming that the system consists of the subsystems: $T_{c,1} = \{t_2, t_4\}$ and $T_{o,1} = \{t_2, t_3, t_4\}$, $T_{c,2} = \{t_5, t_7\}$ and $T_{o,2} = \{t_5, t_6, t_7\}$, $T_{c,3} = \{t_{10}, t_{12}\}$ and $T_{o,3} = \{t_{10}, t_{11}, t_{12}\}$, $T_{c,4} = \{t_{13}, t_{15}\}$ and $T_{o,4} = \{t_{13}, t_{14}, t_{15}\}$. We take $\mathcal{C}_i = \{i\}$ for $i = 1 \ldots 4$. Enforcing (34) results in the control places C_1, C_2, C_3, and C_4 shown in Figure 9. They correspond to the subsystems 1,2,3 and 4, respectively, and enforce $\mu_2 + \mu_3 \leq 2$, $\mu_5 + \mu_6 \leq 2$, $\mu_9 + \mu_{10} \leq 2$, and $\mu_{12} + \mu_{13} \leq 2$.

In [9], (35–36) are assumed to be enforced at a higher hierarchical level at which all transitions locally observable and controllable are observable and controllable, except for t_7 and t_8, which are uncontrollable due to communication problems. In this context, (35) is inadmissible and (36) is admissible. The design of an admissible constraint for (35) illustrates the need for liveness constraints. Indeed, our computer implementation of the decentralized algorithms generates the admissible constraint $\mu_5 + \mu_6 + \mu_7 + v_8 \leq 2$ for (35). However, this is clearly an unacceptable constraint, as it causes the closed-loop to unavoidably deadlock. As discussed in section 6.4, a remedy is to add liveness constraints. So we added the liveness constraint $L_i x \leq 0$ for $x = Dy$ and $y = [1, 1, \ldots 1]^T$. This is to prevent the constraints generated by the algorithm from blocking the firing sequence $t_1 t_2 \ldots t_{16}$ to occur infinitely often. Then, the generated constraints for (35–36) are

$$\mu_5 + \mu_6 + \mu_7 + v_8 - v_{16} \leq 2 \tag{37}$$

$$v_{16} - v_8 \leq 2 \tag{38}$$

They are enforced by the control places C_5 and C_6 in Figure 9. Note that compared to the solution of [9], our solution is equivalent. However, in our case the supervisor can be reused for other values of n_1, n_2, k and u, by changing accordingly the initial markings of $C_1 \ldots C_6$.

Assuming that the higher level supervisor implementing C_5 and C_6 uses direct links to access each transition, the communication cost depends only on the number of links, that is, the number of transitions it controls and/or observes. Figure 9 shows that the communication between the plant and C_5 and C_6 involves t_5, t_8 and t_{16}. Is three the minimal number of transitions? While our approach is suboptimal, the minimization it employs could be used to find a solution with communication that involves less transitions. To do so, we could attempt to design a supervisor for (35-36) minimizing communication. The setting is as follows. At the higher level, no transition is controllable or observable. Communication can make all transitions observable and controllable, except for the transitions of $T_{uc,L} = \{t_3, t_6, t_{11}, t_{14}, t_7, t_8\}$, which cannot be controlled. With the notation of section 6.3, we have $c = \mathbf{1}^T$, $f_i = \mathbf{0}^T$ (here $i = 1$), and we constrain ε_{ij} to $\varepsilon_{ij} = 0$ $\forall t_j \in T_{uc,L}$ and $\varepsilon_{ij} = \alpha_{ij} = \alpha_j$ $\forall t_j \notin T_{uc,L}$. By solving the integer program the following constraints were obtained

$$\mu_1 + \mu_2 + \mu_3 + 2\mu_4 + \mu_5 + \mu_6 + \mu_7 + \mu_8 + \mu_{16} + v_8 - v_{16} \leq 2 \tag{39}$$

$$\mu_{14} + v_{16} - v_8 \leq 2 \tag{40}$$

which are enforced by control places C'_5 and C'_6 such that $C'_5\bullet = \{t_1\}$, $\bullet C'_5 = \{t_{15}\}$, $C'_6\bullet = \{t_{15}\}$ and $\bullet C'_6 = \{t_8\}$. In this solution, the communication involves t_1, t_8 and t_{15}. While this solution is obviously less permissive than that of (37–38), it shows that our approach cannot find a solution involving the communication of less transitions. In this sense, (37–38) are optimal. Finally, (39–40) illustrate once more that the permissivity of the solutions is hard to control. However, in this particular case, a second integer program can be used to improve the permissivity, by minimizing the sum of the positive coefficients in (39–40), while requiring the other coefficients to stay less or equal to zero (the integer program is also subject to the constraints of the previous integer program and to $\sum \alpha_i = 3$, which constrains the communication cost to the minimum value). After solving this second integer program, the solution of (37–38) is again obtained.

Note that the total amount of time taken by our software implementation to design the supervisors described in this section is less than one second on an average computer.

9 Conclusions

The design of decentralized supervisors is computationally easy for the class of specifications identified as d-admissible. When communication between the local supervisors is allowed, the concept of d-admissibility can also be used for the design of supervisors enforcing specifications that are not d-admissible, by the identification of the events that need to be communicated. In the decentralized settings with no communication or with restricted communication, the enforcement of specifications that are not d-admissible can be done via linear integer programming. The integer programming approach is suboptimal, as it may not produce the least restrictive solution, when it exists. However, it allows to design both the supervisors and their communication policy. Moreover, it can be used to minimize the communication of the local supervisors. Future work may explore possibilities to increase the computational efficiency of the supervisor design. Another direction to be considered is decentralized deadlock prevention, to avoid the deadlock possibilities existing in the system and those caused by supervision. Finally, many of the results presented in this paper can be extended to other supervisory techniques and/or DES models.

References

[1] G. Barrett and S. Lafortune. Decentralized supervisory control with communicating controllers. *IEEE Transactions on Automatic Control*, 45(9):1620–1638, 2000.

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

[3] A. Giua, F. DiCesare, and M. Silva. Generalized mutual exclusion constraints on nets with uncontrollable transitions. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, pages 974–979, 1992.

[4] C. Haoxun. Control synthesis of Petri nets based on s-decreases. *Discrete Event Dynamic Systems: Theory and Applications*, 10(3):233–250, 2000.

[5] M. V. Iordache and P. J. Antsaklis. Synthesis of supervisors enforcing general linear vector constraints in Petri nets. In *Proceedings of the 2002 American Control Conference*, pages 154–159, 2002.

[6] X. Koutsoukos and P. Antsaklis. Hybrid control of a robotic manufacturing system. In *Proceedings of the 7th IEEE Mediterranean Conference on Control and Automation*, pages 144–159, 1999.

[7] Y. Li and W. Wonham. Control of Vector Discrete-Event Systems I - The Base Model. *IEEE Transactions on Automatic Control*, 38(8):1214–1227, August 1993.

[8] Y. Li and W. Wonham. Control of Vector Discrete-Event Systems II - Controller Synthesis. *IEEE Transactions on Automatic Control*, 39(3):512–530, March 1994.

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

[10] J. O. Moody and P. J. Antsaklis. Petri net supervisors for DES with uncontrollable and unobservable transitions. *IEEE Transactions on Automatic Control*, 45(3):462–476, 2000.

[11] P. Ramadge and W. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77(1):81–98, 1989.

[12] K. Rudie. The current state of decentralized discrete-event control systems. In *Proceedings of the 10th Mediteranean Conference on Control and Automation*, 2002.

[13] K. Rudie, S. Lafortune, and F. Lin. Minimal communication in a distributed discrete-event system. In *Proceedings of the 1999 American Control Conference*, pages 1965–1970, 1999.

[14] K. Rudie and W. Wonham. Think globally, act locally: Decentralized supervisory control. *IEEE Transactions on Automatic Control*, 37(11):1692–1708, 1992.

[15] G. Stremersch. *Supervision of Petri Nets.* Kluwer Academic Publishers, 2001.

[16] S. Takai and S. Kozama. Decentralized state feedback control of discrete event systems. *Systems & Control Letters*, 22(5):369–375, 1994.

[17] J.H. van Schuppen. Decentralized supervisory control with information structures. In *Proceedings of the International Workshop on Discrete Event Systems (WODES98)*, pages 36–41, 1998.

[18] L. Vandenberge and S. Boyd. Semidefinite programming. *SIAM Review*, 38(1):49–95, 1996.

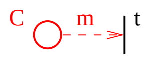
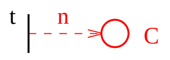
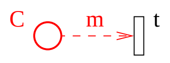
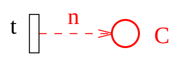
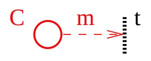
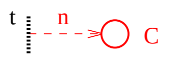
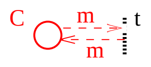
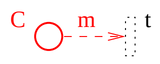
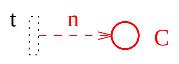
[19] E. Yamalidou, J. O. Moody, P. J. Antsaklis, and M. D. Lemmon. Feedback control of Petri nets based on place invariants. *Automatica*, 32(1):15–28, January 1996.

APPENDIX

A Summary of the Interpretation of the Transition Arcs

Table 1 summarizes the interpretation of the transition arcs connected to admissible supervisors. The notation is as follows: t is a transition of the plant and C is a control place of the supervisor. The first column of the table shows the configurations under consideration. The second column shows the interpretation of the transition arcs. The third column shows the admissibility property of the configuration that makes the interpretation valid. The more obvious interpretations, e.g. those for arcs of controllable and observable transitions, are not included.

Table 1: Supervisor Implementation

Configuration	Supervisor Action	Admissibility requirements
	inhibit t when $\mu(C) < m$; when t fires, $\mu(C) \longrightarrow \mu(C) - m$	—
	when t fires, $\mu(C) \longrightarrow \mu(C) + n$	—
	when t fires, $\mu(C) \longrightarrow \mu(C) - m$	t must not be plant-enabled when $\mu(C) < m$
	when t fires, $\mu(C) \longrightarrow \mu(C) + n$	—
	inhibit t when $\mu(C) < m$	t must be dead in the closed-loop
	—	t must be dead in the closed-loop
	inhibit t when $\mu(C) < m$	—
	—	t must be dead in the closed-loop
	—	t must be dead in the closed-loop

B Reducing the Number of Uncontrollable or Unobservable Transitions

This appendix shows how knowledge on the reachable markings can be used to eliminate transitions from the list of uncontrollable transitions and the list of unobservable transitions. Thus, transitions that a supervisor does not need to control or observe are identified.

Assume that a supervisor is to enforce $L\mu \leq b$, while it is known that all reachable markings will satisfy $L_R\mu \leq b_R$. While $L_R\mu \leq b_R \Rightarrow L\mu \leq b$, additional constraints in $L_R\mu \leq b_R$ can be added if information on the initial marking is available. The following computable tests can be used:

> Given a transition t, let w be the column corresponding to t in D^-, the output matrix, and q the firing vector associated to the firing of t. A supervisor enforcing $L\mu \leq b$ does not need to control an uncontrollable transition t if the system
>
> $$\begin{aligned} L_R\mu &\leq b_R \\ \mu &\geq w \\ L(\mu + Dq) &\not\leq b \end{aligned}$$
>
> is infeasible. Furthermore, t never fires (and so never needs to be observed) if the system
>
> $$\begin{aligned} L_R\mu &\leq b_R \\ \mu &\geq w \end{aligned}$$
>
> is infeasible.

C The Sets Σ_{uc}^M and Σ_{uo}^M

This appendix studies the existence of the sets Σ_{uc}^M and Σ_{uo}^M. These are defined for a prefix-closed specification $K \subseteq \mathcal{L}(G)$ as the maximal subsets of Σ such that K is controllable and observable with respect to $(G, \Sigma_{uc}^M, \Sigma_{uo}^M)$. We show that Σ_{uc}^M exists, but Σ_{uo}^M may not exist.

Note first that regardless of K, sets Σ_{uc} and Σ_{uo} such that K is controllable and observable with respect to $(G, \Sigma_{uc}, \Sigma_{uo})$ exist. Indeed, this is true of the fully controllable and observable case, for which $\Sigma_{uc} = \emptyset$ and $\Sigma_{uo} = \emptyset$.

The existence of Σ_{uc}^M can be proven based on the fact that if K is controllable with respect to Σ_{uc}^1 and Σ_{uc}^2, then it is controllable with respect to $\Sigma_{uc}^1 \cup \Sigma_{uc}^2$. Indeed, according to the definition of controllability: $K\Sigma_{uc}^1 \cap \mathcal{L}(G) \subseteq K$ and $K\Sigma_{uc}^2 \cap \mathcal{L}(G) \subseteq K$. Then $K(\Sigma_{uc}^1 \cup \Sigma_{uc}^2) \cap \mathcal{L}(G) \subseteq K$ can be immediately checked, by noticing that $K(\Sigma_{uc}^1 \cup \Sigma_{uc}^2) \cap \mathcal{L}(G) = (K\Sigma_{uc}^1 \cap \mathcal{L}(G)) \cup (K\Sigma_{uc}^2 \cap \mathcal{L}(G))$.

To prove the existence of Σ_{uc}^M, let's denote by Σ_{uc}^i all distinct subsets of Σ, $i = 1 \ldots p$, such that for all $i = 1 \ldots p$, K is controllable with respect to G and Σ_{uc}^i. In view of the previous result, there is $\Sigma_{uc}^j = \Sigma_{uc}^{max}$, for $\Sigma_{uc}^{max} = \bigcup_{i=1\ldots p} \Sigma_{uc}^i$. Then note that Σ_{uc}^{max} satisfies the definition of Σ_{uc}^M, so $\Sigma_{uc}^M = \Sigma_{uc}^{max}$.

The fact that Σ_{uo}^M may not exist can be seen in an example. Consider G with the language $\mathcal{L}(G) = \overline{\{abcd, ad\}}$, $K = \overline{\{abcd\}}$, $\Sigma_{uo}^1 = \{b\}$ and $\Sigma_{uo}^2 = \{c\}$. It can be easily checked that K is observable with respect to Σ_{uo}^1 and Σ_{uo}^2, however not observable with respect to $\Sigma_{uo} = \Sigma_{uo}^1 \cup \Sigma_{uo}^2$.

Indeed, if P is the projection removing the elements of Σ_{uo}, we have that $P(a) = P(abc) = a$, $ad, abcd \in \mathcal{L}(G)$, $ad \notin K$ and $abcd \in K$. It follows that Σ_{uo}^M does not exist, since it cannot contain both b and c.

Note that the same is true even if we define Σ_{uo}^M with respect to normality, instead of with respect to observability. (K is normal with respect to G and Σ_{uo} if $K = \mathcal{L}(G) \cap P^{-1}(P(K))$, where P is the projection removing the elements of Σ_{uo}.) The previous example can also be used to show that Σ_{uo}^M may not exist. If P_1 and P_2 are the projections with respect to Σ_{uo}^1 and Σ_{uo}^2, note that $K = \mathcal{L}(G) \cap P_1^{-1}(P_1(K))$, $K = \mathcal{L}(G) \cap P_2^{-1}(P_2(K))$, but $K \neq \mathcal{L}(G) = \mathcal{L}(G) \cap P^{-1}(P(K))$. So in this example Σ_{uo}^M does not exist.

With regard to the computational complexity, the set Σ_{uc}^M can be computed in polynomial time, since controllability can be checked in polynomial time. This is also true of Σ_{uo}^M, when it exists.