Marian Iordache, Panos J.Antsaklis, "Admissible Decentralized Control of Petri Nets," Proceedings of the 2003 American Control Conference, pp. 332-337, Denver, Colorado, June 4-6, 2003.

Admissible Decentralized Control of Petri Nets

Marian V. Iordache and Panos J. Antsaklis[1,2]

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, the supervisors may exchange information consisting of local observed events. We propose efficient algorithms for the design of decentralized supervisors, based on the extension of the SBPI concept of admissibility that we define.

1 Introduction

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, *supervision based on place invariants* (SBPI) [1, 5, 12] 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* Petri net [1, 12]. Furthermore, without loss of any of its benefits, the SBPI has been extended in [2] 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. In SBPI, a set of constraints is *admissible* if it can be directly enforced. On the other hand, inadmissible constraints are first transformed to admissible constraints, and then enforced. Thus, an admissible set of constraints is roughly[3] the equivalent of a controllable and observable specification in the Ramadge and Wonham framework [6].

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 [7]. Second, we show how to enforce *d-admissible* constraints and show how to check whether a constraint is d-admissible. Third, when a constraint is not d-admissible, we provide an algorithmic approach to make the constraint d-admissible by enabling communication of events (transition firings).

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. The problem of finding necessary and sufficient conditions for the existence of a decentralized solution enforcing a state predicate is studied in [10]. In [11], 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. In particular, the idea of information structures in [11] is related to the clustering of subsystems in our paper. Other decentralized control work can be found in [8] and the references therein. Literature on SBPI or closely related to it is found in [1, 5, 9] and the references therein. Finally, the work presented in this paper is continued in the sequel paper [3].

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 introduces the d-admissibility and the related algorithms. Finally, d-admissibility is applied to the design of local supervisors with communication in section 5.

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

[2]The partial support of the Lockheed Martin Corporation, of the National Science Foundation (NSF ECS99-12458), and of DARPA/IXO-NEST Program (AF-F30602-01-2-0526) is gratefully acknowledged.

[3]We define admissibility with respect to the SBPI. In principle, it is possible to have an inadmissible constraint that is "admissible" with respect to another supervision technique.

Marian Iordache, Panos J.Antsaklis, "Admissible Decentralized Control of Petri Nets," Proceedings of the 2003 American Control Conference, pp. 332-337, Denver, Colorado, June 4-6, 2003.

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).

The specification of the SBPI [1, 5, 12] 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 $\mathcal{N}$. 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

$$D_s = -LD \tag{2}$$

$$\mu_{0,s} = b - L\mu_0 \tag{3}$$

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:

$$D_c = \begin{bmatrix} D \\ -LD \end{bmatrix} \tag{4}$$

An example is shown in Figure 4(a), 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 $L_a\mu \leq b_a \Rightarrow L\mu \leq b$ [5]. 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 detail in section 4. We will denote $\mathcal{N}$ with sets of uncontrollable and unobservable transitions T_{uc} and T_{uo} by $(\mathcal{N}, T_{uc}, T_{uo})$.

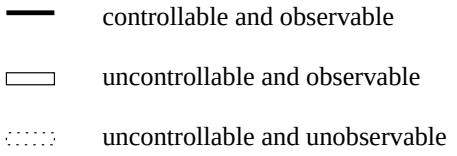

Figure 1: Graphical representation of the transition types.

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 $\mathcal{S}_1, \mathcal{S}_2, \ldots \mathcal{S}_n$, 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* $T_{uc,1}, T_{uc,2}, \ldots T_{uc,n}$ *and* $T_{uo,1}, T_{uo,2}, \ldots T_{uo,n}$, *find a set of local supervisors* $\mathcal{S}_1, \mathcal{S}_2, \ldots \mathcal{S}_n$ *whose simultaneous operation guarantees that the global specification is satisfied, where each* $\mathcal{S}_i$ *can control* $T \setminus T_{uc,i}$ *and can 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 $(\mathcal{N}, T_{uc,1}, \ldots T_{uc,n}, T_{uo,1}, \ldots T_{uo,n})$.

As an example, we consider the manufacturing system of [4], 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).

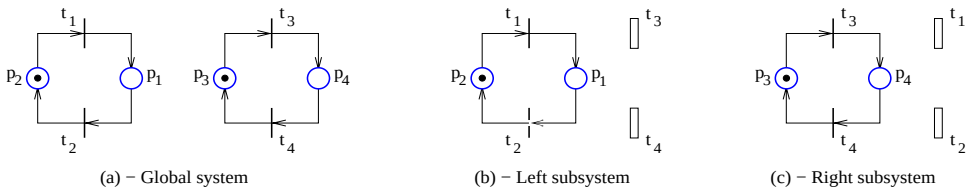

Figure 3: A Petri net model of the robotic manufacturing system.

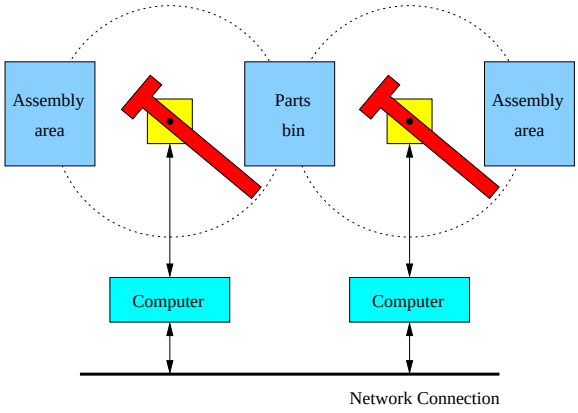

Figure 2: Robotic manufacturing system.

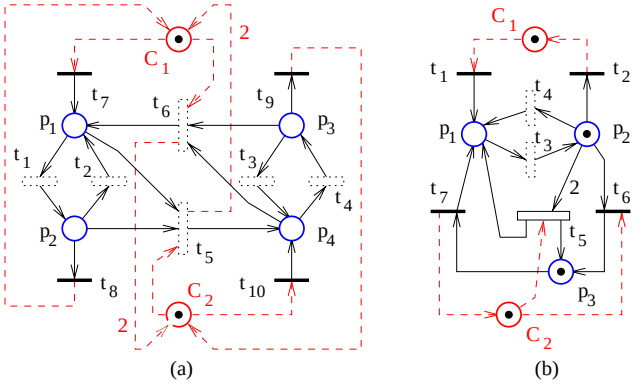

Figure 4: Examples of c-admissible supervision.

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 Figure 4(a) consists of the control places C_1 and C_2. By definition, the supervisor is admissible, though 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 Figure 4(b) consists of the control places C_1 and C_2. 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*[4]*; 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.*

In the decentralized case, we are interested in defining admissibility with respect to a Petri net $(\mathcal{N}, \mu_0)$,

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

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. The following definition achieves this.

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 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. Formally:

Proposition 4.1 *Any set of constraints is d-admissible if* $T_{uo,i} = \emptyset$ *for all* $i = 1 \ldots n$ *and* $\bigcap_{i=1 \ldots n} T_{uc,i} = \emptyset$*.*

The construction of a decentralized supervisor for d-admissible constraints is illustrated on the Petri net of Figure 3. The mutual exclusion constraint $\mu_1 + \mu_3 \leq 1$ 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 is composed of the places p_1 and p_2, and the second one of the places p_3 and p_4. Assume $T_{uo,1} = T_{uo,2} = \emptyset$, $T_{uc,1} = \{t_3, t_4\}$ and $T_{uc,2} = \{t_1, t_2\}$. Note that the constraint 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. The decentralized solution is shown also in Figure 5. There are two control places C_1 and C_2, each representing the supervisor of the left and right subsystem, respectively. Note that C_1 and C_2 and their connections represent two copies of the control place C and its connections. As C_1 and C_2 have the same initial marking as C, their markings stay equal at all times. So, at all times C_1 disables the same transitions as C_2. However, as discussed before, the disablement of t_1 (t_4) is implemented by C_1 (C_2), while (C_1, t_4) and (C_2, t_1) are interpreted as observation arcs.

Algorithm 4.1 *Supervisor design for a d-admissible constraint* (uses the notation of Definition 4.1)

1. Create $|\mathcal{C}|$ copies of the centralized supervisor enforcing the constraint in $(\mathcal{N}, T_{uc}, T_{uo})$.
2. Associate each copy to one of the subsystems $i \in \mathcal{C}$; it represents the supervisor of subsystem i.
3. Set the initial state of each copy to the initial state of the centralized supervisor.

To enforce a d-admissible set of constraints, the construction is repeated for each constraint. Next we prove that the resulting decentralized supervisor is feasible (physically implementable). Let $\mathcal{S}_1$, $\mathcal{S}_2$, $\ldots$ $\mathcal{S}_n$ denote the local supervisors of a decentralized supervisor, were each of $\mathcal{S}_i$ can control and observe the transitions in $T_{c,i}$ and $T_{o,i}$, respectively. The decentralized supervisor is **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 4.1 *The decentralized supervisor constructed in Algorithm 4.1 is feasible, enforces the desired constraint, and is as permissive as the centralized supervisor of* $(\mathcal{N}, T_{uc}, T_{uo})$*.*

Proof: The notation of Definition 4.1 is assumed. Note that all supervisors $\mathcal{S}_i$ have the same connections to the net and the same initial marking as the centralized supervisor. Therefore, all of $\mathcal{S}_i$ have equal marking at all times.

Feasibility: If $t \in T_{uo,i}$ fires, for $i \in \mathcal{C}$, its firing would not affect the marking of the centralized supervisor, as the set of constraints is c-admissible with respect to $(\mathcal{N}, T_{uc}, T_{uo})$, and $T_{uo,i} \subseteq T_{uo}$. Therefore, it does not affect either the marking of any of $\mathcal{S}_i$.

If t is plant-enabled but not closed-loop enabled, the centralized supervisor would also prevent the firing t. Since the centralized supervisor is admissible, $t \notin T_{uc}$. Then, by the definition of T_{uc}, there is $j \in \mathcal{C}$ such that $t \notin T_{uc,j}$. So $\mathcal{S}_j$ disables t and $t \in T_{c,j}$.

Enforcement and permissivity: True, since for any transition t, t is enabled by the decentralized supervisor iff it is enabled by the centralized supervisor. ■

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} be the set of transitions that are *not* detected by $\mathcal{S}$ and T^*_{uc} the set of transitions that are *not* controlled by $\mathcal{S}$.

Algorithm 4.2 *Checking whether a constraint is d-admissible*

1. Find T^*_{uo} and T^*_{uc}.

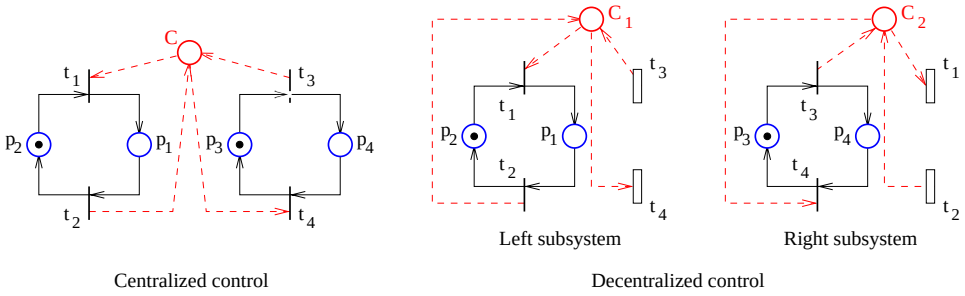

Figure 5: Centralized control versus decentralized control.

2. Find the largest set of subsystems $\mathcal{C}$ such that $\forall i \in \mathcal{C}$: $T_{uo,i} \subseteq T^*_{uo}$.
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 $T_{uc} \subseteq T^*_{uc}$? 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 $T^*_{uc} \supseteq \bigcap_{i \in \mathcal{C}_{min}} T_{uc,i}$.

Proposition 4.2 *The algorithm checking d-admissibility is correct.*

Proof: We 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, $T_{uo} \subseteq T^*_{uo}$.

A constraint is declared d-admissible if $\mathcal{C} \neq \emptyset$ and $T_{uc} \subseteq T^*_{uc}$. The definition of T^*_{uo} and T^*_{uc} 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 $\mathcal{C}' \neq \emptyset$ 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 $T'_{uo} \subseteq T^*_{uo}$; $T'_{uo} \subseteq T^*_{uo} \Rightarrow \mathcal{C}' \subseteq \mathcal{C}$ $\Rightarrow T_{uc} \subseteq T'_{uc} \Rightarrow T_{uc} \subseteq T^*_{uc}$. So, the algorithm declares the constraint to be d-admissible. ■

In general, it may be difficult to compute the sets T^*_{uc} and T^*_{uo}. Then estimates $T^e_{uc} \subseteq T^*_{uc}$ and $T^e_{uo} \subseteq T^*_{uo}$ 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 [5], stating that $L\mu \leq b$ is 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 [7].

5 Supervision with Communication

Obviously, communication can be used to change the attributes of otherwise inaccessible transitions to observable or even controllable. As an illustration, consider again the robotic system of Figure 2. The computers controlling the two robots are able to communicate through a network connection. The specification is that the robots should not access the parts bin at the same time. By requiring each computer 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. 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 allowed to fire only when $c_1 \geq 1$. When t_1 fires, the right subsystem is announced and c_1 is decremented, and when t_2 fires, the right subsystem is announced and c_1 is incremented.

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 $\mathcal{C}$, but it cannot be reduced below $T_{uc,L} = \bigcap_{i=1\ldots n} T_{uc,i}$. $T_{uc,L}$ ($T_{uo,L}$) is the set of transitions uncontrollable (unobservable) in all subsystems.

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 [5] could be used) or use the decentralized design approach of the sequel paper [3].
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 $\bigcap_{i\in\mathcal{C}} T_{uc,i} \subseteq T \setminus T_c$.[5]
4. Design the decentralized supervisor by applying Algorithm 4.1 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 of the sequel paper [3]. Third, 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. Fourth, 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 [3] 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 5.1 *The decentralized supervisor is feasible and equally permissive to the centralized supervisor $\mathcal{S}$ enforcing the specification on $(\mathcal{N}, T_{uc}, T_{uo,L})$.*

[5] 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 $T_{uc,L} = \bigcap_{i=1\ldots n} T_{uc,i}$.

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 4.1. ∎

6 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 is considered in the sequel paper [3].

References

[1] A. Giua, F. DiCesare, and M. Silva. Generalized mutual exclusion constraints on nets with uncontrollable transitions. In *Proc. IEEE Conf. Syst., Man, Cybern.*, pp. 974–979, 1992.

[2] M. V. Iordache and P. J. Antsaklis. Synthesis of supervisors enforcing general linear vector constraints in Petri nets. In *Proc. 2002 Amer. Contr. Conf.*, pp. 154–159, 2002.

[3] M. V. Iordache and P. J. Antsaklis. Decentralized control of Petri nets with constraint transformations. In *Proc. 2003 Amer. Contr. Conf.*

[4] X. Koutsoukos and P. Antsaklis. Hybrid control of a robotic manufacturing system. In *Proc. 7th IEEE Mediterranean Conf. Contr. Automat.*, pp. 144–159, 1999.

[5] J. O. Moody and P. J. Antsaklis. Petri net supervisors for DES with uncontrollable and unobservable transitions. *IEEE Trans. Automat. Contr.*, 45(3):462–476, 2000.

[6] P. Ramadge and W. Wonham. The control of DES. *Proc. IEEE*, 77(1):81–98, 1989.

[7] K. Rudie and W. Wonham. Think globally, act locally: Decentralized supervisory control. *IEEE Trans. Automat. Contr.*, 37(11):1692–1708, 1992.

[8] K. Rudie. The current state of decentralized discrete-event control systems. In *Proc. of the 10th IEEE Mediterranean Conf. Contr. Automat.*, 2002.

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

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

[11] J.H. van Schuppen. Decentralized supervisory control with information structures. In *Proc. Intern. Worksh. DES (WODES98)*, pp. 36–41, 1998.

[12] 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, 1996.