Supervision Based On Place Invariants: A Survey

Technical Report of the ISIS Group
at the University of Notre Dame
ISIS-2004-003
July, 2004

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

Submitted for Journal Publication

Supervision Based On Place Invariants: A Survey

Marian V. Iordache and Panos J. Antsaklis*†

July 30, 2004

Abstract

The supervision based on place invariants (SBPI) is an efficient technique for the supervisory control of Petri nets. This paper reveals the significance of the SBPI based on a literature survey, applications, and an analysis of problems and supervisory settings that can be addressed using SBPI. Much of the application and analysis parts of the paper consist of new results. Special attention is given to the various settings within which the problem can be formulated. Such settings can be distinguished based on the concurrency type, the type of controllability and observability, and the centralized or decentralized type of supervision. As we show, it is possible to approach the most general settings in a purely structural way, without resorting to reachability analysis. We begin by describing the SBPI problem and the literature methods that address this problem or are related to it. Then, we proceed to show classes of problems that can be reduced to the SBPI problem. In the SBPI, the specification is described as a system of inequalities that the Petri net marking must satisfy at any time. However, as we show, problems involving more general specifications can be approached in the SBPI setting, sometimes under additional assumptions, by performing net and/or specification transformations. Four of the specifications we will consider are logic constraints, language specifications, disjunctions of linear constraints, and liveness. We conclude with a presentation of possible applications of the SBPI approach to programming with semaphores, fault tolerance, and synchronic-distance based designs.

1 Introduction

Petri nets (PNs) are an important class of discrete event systems, allowing a compact representation of concurrent systems. The literature on the supervision of PNs contains numerous references to the enforcement of specifications consisting of linear inequalities on the PN marking. Such constraints have the form

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

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

†The authors gratefully acknowledge the partial support of the Lockheed Martin Corporation and of the National Science Foundation (NSF CCR01-13131).

where μ is the marking, $L \in \mathbb{Z}^{n_c \times m}$, $b \in \mathbb{Z}^{n_c}$, $\mathbb{Z}$ is the set of integers, m is the number of places of the PN, and n_c the number of constraints. The constraints (1) are sometimes called *generalized mutual exclusion constraints* [18], since a simpler form of (1) correspond to mutual exclusion specifications.

Originally arising in the context of a constrained optimal control problem of chemical processes [67], the use of the constraints (1) has been proposed also for various other applications: the coordination of AGVs [42], manufacturing constraints [51], and mutual exclusion in batch processing [63]. Moreover, by considering also classes of constraints that can be reduced to (1) on transformed PNs, specifically the generalized linear constraints of [39], other applications can be mentioned here as well: supervisory control of railway networks [19] and fairness enforcement, such as bounding the difference between the number of occurrences of two events, in protocols [13] and manufacturing [48].

Other interesting qualities of the constraints (1) are as follows. They can describe any forbidden marking specification on safe Petri nets [68, 18], where a Petri net is *safe* if all reachable markings are binary vectors (i.e. consisting of 0 and 1 elements). This property is very interesting for supervision problems on certain subclasses of Petri nets, and notably on marked graphs. Further, as we show in this paper, more general specifications can be reduced to specifications (1) on transformed PNs. Such specifications include language specifications on labeled PNs and disjunctions of constraints (1), under certain boundedness assumptions. Note that a labeled PN is a PN in which the transitions are labeled with (not necessarily distinct) events, just as in the automata setting. Further, a disjunction of constraints (1) is described by $L_1\mu \leq b_1 \vee L_2\mu \leq b_2 \vee \ldots L_p\mu \leq b_p$, requiring the marking μ to satisfy at least one of $L_i\mu \leq b_i$, $i = 1 \ldots p$. Note also that the constraints (1) are also interesting in the representation of deadlock prevention and liveness specifications [37, 38].

From a historical perspective, the supervisory control of discrete event systems has been related to the problem of Church [9], in Computer Science. In Computer Science, this line of thought was continued with work on program synthesis for open systems (e.g. [54]), with focus on automata models and specifications on infinite sequences of events (temporal logic, ω-languages). In Control Systems, the supervisory control was proposed by Ramadge and Wonham [57], with focus on automata and specifications on finite sequences of events. The results of Ramadge and Wonham prompted also research work on the supervisory control of PNs. However, note that the supervision of PNs can also be traced back to earlier work, such as the use of monitors for liveness enforcement in [44]. The initial work on the supervision of PNs considered forbidden state problems [41] and specifications requiring a PN to reach a target state with additional constraints on the firing sequence [31, 30]. In the subsequent developments on the supervision of PNs, several major approaches can be identified, as follows. First, the supervision of PNs for forbidden state specifications has been approached with path-based methods, as in [27, 42, 69], and also with monitor-based solutions, as in the supervision based on place invariants [18, 68, 52]. Then, there is also an extension of the Ramadge and

Wonham supervisory control [57] to Petri nets [48, 49, 50] as well as work on the enforcement of languages on labeled PNs, e.g. [16, 17, 43]. An excellent survey on the methods proposed for the supervision of Petri nets can be found in [26] and also in [25]. While [26] focuses on the path-based approach, here we survey work that is most relevant to the SBPI and present new results that emphasize the significance of this approach. Note also that much of the work surveyed here was not available at the time of [26].

The contribution and the organization of the paper is as follows. First, the supervision based on place invariants (SBPI) is introduced in section 3. The notation of the paper and several important definitions are also included in this section. To simplify the introduction of the SBPI, section 3 considers the simpler case of full controllability and observability. Then, section 4 presents the various ways partial controllability and partial observability is modeled in the literature. These include individually controllable/observable transitions, controlled PNs (CtlPNs), labeled PNs, and marking observation. In section 4 we also introduce a new concept, which we call *double-labeled PNs*. Double-labeled PNs are shown to be able to represent the systems described by any of the previous modeling techniques (CtlPNs, labeled PNs, PNs with individually controllable/observable transitions). Further, as shown in the following section 5, the admissibility based methods for the SBPI (e.g. in [51]) can be adapted for double-labeled PNs. This is another new result presented in this paper.

After outlining the principle of the admissibility-based methods and presenting structural admissibility tests in section 5, the literature approaches that can deal with specifications (1) are overviewed in section 6. The literature survey of section 6 is classified according to the type of the methods, such as methods based on structural conditions for admissibility, or on a path analysis, or on the computation of the maximal controlled-invariant set, or for decentralized control. Section 7 deals with the expressiveness of the constraints (1). This section overviews several results showing how various supervision problems can be reduced to the enforcement of constraints (1). Some of the results overviewed here are known (logic constraints, representing liveness constraints by (1), reducing the generalized constraints of [39] to (1) by PN transformations). However, there are also two new results presented in section 7: reducing language specifications and disjunctions of constraints to (1), by PN transformations. We emphasize the significance of these results, as they expand the area of applicability of the supervision methods for constraints (1).

In section 8 we show three applications of the constraints (1). First, we examine the relation between the SBPI and programming with semaphores, discussing also the implications to automated code generation in software engineering. Then, we show that constraints (1) and one of their extensions can be used to represent redundant embeddings for fault tolerant applications. Finally we show that one of the extensions of (1) can represent a class of specifications arising in the context of the Theory of Synchrony.

2 Notation

A Petri net (PN) will be denoted by the structure $\mathcal{N} = (P, T, D^-, D^+)$, where P is the set of places, T the set of transitions, $D^-, D^+ \in \mathbb{N}^{|P|\times|T|}$ are the input and output matrices, and $\mathbb{N}$ is the set of nonnegative integers. Further, we denote by $D = D^+ - D^-$ the incidence matrix and by μ the marking. A Petri net with initial marking μ_0 will be denoted by $(\mathcal{N}, \mu_0)$.

In this survey we will distinguish between the *firing vector* q, the *Parikh vector* v and the *firing count vector* $\underline{\sigma}$. The firing vector q describes the transition(s) that fire at a firing instance. The Parikh vector is a state variable, indicating how often each transition has fired since the initialization of the system. Finally, the firing count vector $\underline{\sigma}$ is defined with respect to a finite firing sequence σ, indicating how many times each transition t occurs in σ. In particular, if σ is the sequence fired since the initialization of the system, $v = \underline{\sigma}$.

The set of reachable markings of $(\mathcal{N}, \mu_0)$ will be denoted by $\mathcal{R}(\mathcal{N}, \mu_0)$. Recall, a Petri net $(\mathcal{N}, \mu_0)$ in which all reachable markings are binary vectors, i.e. $\mathcal{R}(\mathcal{N}, \mu_0) \subseteq \{0, 1\}^{|P|}$ is said to be *safe*.

We call (1) a *set of constraints*, because it consists of the constraints $L(i, \cdot)\mu \leq b(i)$, for $i = 1 \ldots k$, and k the number of rows of L. Further, we also say that (1) is a *conjunction of constraints*, since all $L(i, \cdot)\mu \leq b(i)$, $i = 1 \ldots k$, must be satisfied when (1) is satisfied. In contrast, a *disjunction of constraints* $l_i\mu \leq c_i$, $i = 1 \ldots k$, describes the requirement that at all times there is i such that μ satisfies $l_i\mu \leq c_i$. We denote the disjunction of constraints by $\bigvee_i l_i\mu \leq c_i$.

3 The Supervision Based on Place Invariants

This section introduces the supervision based on place invariants (SBPI). The presentation of this section focuses on the simplest case: no concurrency and full controllability and observability. At the end of the section we will present also various concurrency settings, together with the simple extension of the SBPI for concurrency. The SBPI under partial controllability and observability is more involved, and will be presented in subsequent sections.

In the SBPI, the system to be controlled is called **plant**, and is assumed to be given in the form of a PN $\mathcal{N} = (P, T, D^-, D^+)$. The SBPI provides a supervisor enforcing (1) in the form of a PN $\mathcal{N}_s = (P_s, T, D_s^-, D_s^+)$ 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 **monitors**[1]. The supervised system, that is the

[1]In much of the literature, the monitors are called *control places*. In this paper we do not call them control places, to avoid

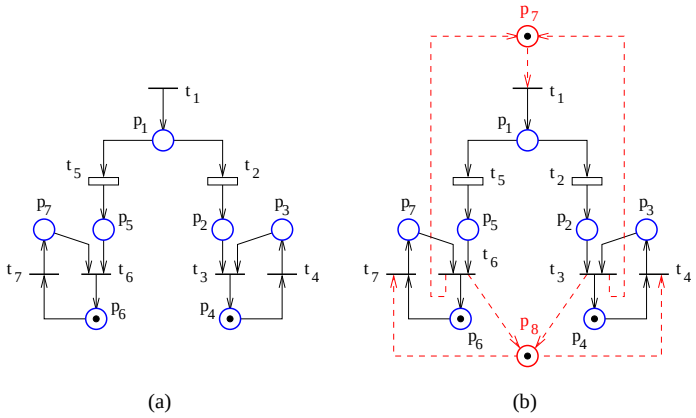

Figure 1:

closed-loop system, is a PN $\mathcal{N}_c$ of incidence matrix:

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

The relation to place invariants is as follows. Recall, a **place invariant** of $\mathcal{N}$ is an integer vector $x \in \mathbb{Z}^{1\times|P|}$ such that $xD = 0$. Recall also that for a place invariant x, $x\mu = x\mu_0$ for all reachable markings μ. Note that all rows of $[L,\ I]$ are place invariants of $\mathcal{N}_c$. Then, from (3) it follows that at all reachable markings of the closed loop:

$$\mu_s = b - L\mu \tag{5}$$

Since μ_s, the marking of the monitors, is nonnegative, (1) is enforced. Furthermore, we can say that (1) is enforced by creating the invariants $[L,\ I]$ in the closed-loop. This why the approach is "based on place invariants."

Example 3.1 The PN of Figure 1(a) adapts a PN model of [51] of an unreliable machine [12, 51]. Let's denote $\mu(p_i)$ by μ_i. Assume we desire to enforce

$$\mu_1 + \mu_2 + \mu_5 \leq 1 \tag{6}$$

$$\mu_3 + \mu_7 \leq 1 \tag{7}$$

By (2–3), we obtain the supervisor shown in Figure 1(b), consisting of the monitors p_7 and p_8. Thus, (5) is described by

$$\mu_7 = 1 - \mu_1 - \mu_2 - \mu_5 \tag{8}$$

$$\mu_8 = 1 - \mu_3 - \mu_7 \tag{9}$$

confusion with the quite different concept of control places of the CtlPN approach to the supervision of PNs [41, 27, 42].

where (8) and (9) correspond to the two place invariants created by p_7 and p_8. □

The supervisors constructed as above are optimal [18, 51, 68]. Following [51], the optimality can be stated as follows:

Theorem 3.1 [51] *If $L\mu_0 \leq b$ then the PN supervisor with incidence matrix $D_s = -LD$ and initial marking $\mu_{0,s} = b - L\mu_0$ enforces the constraint $L\mu \leq b$ when included in the closed-loop system $D_c = [D^T, D_s^T]^T$. Furthermore, the supervision is least restrictive.*

Much of the PN literature is written under the assumption that only one transition may fire at a time. This is known as the **no concurrency assumption**. This will also be the usual assumption in this survey. However, since many results are not limited to this setting, we will consider also other concurrency assumptions. A very good presentation of the various concurrency settings can be found in [60].

Let q denote the firing vector. Under the no concurrency assumption, $q \in \{0,1\}^{|T|}$, $\sum_{t\in T} q(t) = 1$, and the entry with $q(t) = 1$ indicates the transition that is to fire. Another concurrency setting is under the **concurrency assumption**. Under this assumption, groups of transitions may fire at the same time. In this case, $q \in \{0,1\}^{|T|}$ and $\{t : q(t) = 1\}$ identifies the transitions t that are to be fired at the same time. Still another setting corresponds to the **transition-bag assumption**. Under this assumption, the transitions in a group may be fired each several times, at the same firing instance. Thus, $q \in \mathbb{N}^{|T|}$ and for each t, $q(t)$ indicates how many times t is fired. Following [60], we can incorporate these concurrency assumptions in a general setting in which we require $q \in \Delta$, for a given $\Delta \subseteq \mathbb{N}^{|T|}$.

Under any concurrency setting, a firing vector q is enabled by the plant at the marking μ when

$$\mu \geq D^- q \tag{10}$$

While under the no concurrency assumption it is convenient to consider that a supervisor enables transitions, for the general case we consider that a supervisor enables firing vectors. Following [60], we restrict our attention to the supervisors with the property that if they enable q, then they enable every $q' \leq q$.

Note that the SBPI design remains optimal under concurrency, as Theorem 3.1 still applies. This has been formally proven in [60].

In the literature, the study of the SBPI began with the work of Giua et al [18]. The reference [18] considers the enforcement of constraints (1) when all elements of L and b are nonnegative. The paper deals with the redundancy, equivalence and modeling power of the specifications (1), and the enforcement of (1) for fully controllable and observable PNs. The authors show how to construct the supervisor based on L, D and μ_0, and prove a result equivalent to Theorem 3.1. While the results of [18] assume that L and b in (1) have nonnegative elements, most results there apply also in the general case.

The benefits of the SBPI were further detailed in [68], which considers also a more general set of linear constraints that involve both the marking μ and the firing vector. A simple approach for the conversion of boolean expressions to (1) for safe PNs appears also in [68]. Further, a very accessible presentation of the SBPI appears in [51, 52], together with extensions for PNs with uncontrollable and unobservable transitions. For the most part, our notation follows that of [51, 52]. Moreover, as we will see later, some of the new results included in this paper extend results of [51, 52] on the design of supervisors for PNs with uncontrollable and unobservable transitions.

As seen in this section, the SBPI design is both simple and optimal in the case of fully controllable and observable PNs. Thus, in the literature, the focus has been on the development of design methods for PNs with partial controllability and partial observability. Before surveying this part of the literature, we present the various concepts of controllability and observability that have been used.

4 Uncontrollability and Unobservability

Our developments in the previous section rely on the assumptions that (a) all transitions of the PN can be disabled at will, that is, are controllable; (b) the firings of any transition can be detected; (c) each transition firing produces a distinct event. By relaxing (a), (b), and (c) we obtain PNs with partial controllability, partial observability, and with a labeling, respectively.

In the literature, we can distinguish two main types of uncontrollability and unobservability. In the first one, events can be controlled and observed, as in the Ramadge and Wonham setting [57]. Thus, when the transitions have distinct event labels, individual transitions can be controlled/observed. Another view of partial controllability has been introduced by Krogh [41], who proposed the *controlled PNs*. In the controlled PN setting, sets of transitions (as opposed to individual transitions) may be disabled. Further, a different kind of partial observability results when the supervisor is assumed to rely on the state (marking) rather than transition firings. In this section we describe and compare the various controllability and observability settings. In particular, we will introduce a class of PNs, called double-labeled PNs, and show that it encompasses all types of uncontrollability while modeling also event unobservability. This result is significant, as we will show in the next section that double-labeled PNs can be approached by structural methods of supervisor design.

4.1 Individually Controllable and Observable Transitions

In this setting, the set of transitions T is partitioned in $T = T_c \cup T_{uc}$ and $T = T_o \cup T_{uo}$, where T_c (T_o) is the set of controllable (observable) transitions and T_{uc} (T_{uo}) is the set of observable (unobservable) transitions. Thus, a supervisor has the ability to control only the transitions $t \in T_c$ and to observe only the firings of

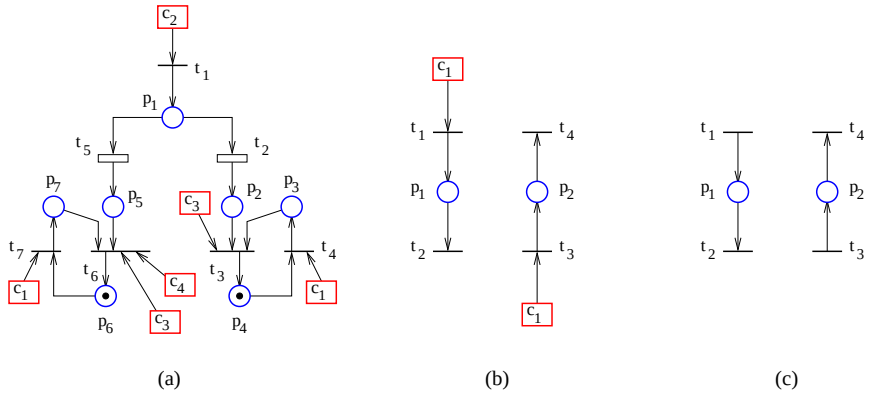

Figure 2:

$t \in T_o$.

As an illustration, consider the PN of Figure 2(c), modeling a part of a manufacturing system. In the manufacturing system, AGVs going in opposite directions can enter a common loading area. In the model, firing t_1 (t_2) corresponds to AGVs entering (exiting) in one direction, and firing t_3 (t_4) corresponds to AGVs entering (exiting) from the other direction. Thus, $t_1 \in T_o$ corresponds to the case in which we can detect when an AGV enters the loading area from one of the directions. Further, $t_1, t_3 \in T_c$ corresponds to the case when we can prevent AGVs from entering the loading area from either direction.

A possible way to generalize this setting appears in [3], which proposes replacing T_{uc} and T_{uo} with control and observation costs. Other ways to generalize this setting are discussed in the remaining part of this section.

4.2 Controlled PNs (CtlPNs)

This setting introduces a different kind of uncontrollability. Following [26], a **controlled PN (CtlPN)** is a triple $\mathcal{N}^c = (\mathcal{N}, \mathcal{C}, \mathcal{B})$, where $\mathcal{N} = (P, T, F)$ is an ordinary PN, $\mathcal{C}$ is a finite set of **control places**, $\mathcal{C} \cap P = \emptyset$, and $\mathcal{B} \subseteq \mathcal{C} \times T$ is a set of directed arcs. As expected, given a marking μ of $\mathcal{N}$, a transition t is enabled by the plant, or **state enabled**, when for all places $p \in P$, $(p, t) \in F \Rightarrow \mu(p) \geq 1$. A **control** for a CtlPN is a function $u : \mathcal{C} \to \{0, 1\}$. Given a control u, a transition t is **control enabled** when for all control places c, $(c, t) \in \mathcal{B} \Rightarrow u(c) = 1$. Of course, a transition can be fired only when it is both control and state enabled. In a concurrency setting, the control allows all control-enabled transitions to be fired simultaneously.

Note that firing a transition has no effect on the control (there are no tokens flowing out of the control places). This distinguishes the control places of CtlPNs from the monitors in the context of the SBPI, which behave completely like the normal places of a Petri net.

As an example, consider Figure 2(a). The control places in the CtlPN shown there are $c_1 \ldots c_4$. The

firings of t_2 and t_5 cannot be controlled, as no control places are connected to t_2 and t_5. On the other hand, $c_1 \ldots c_4$ control the firings of the other transitions. For instance, t_7 may be fired only if $u(c_1) = 1$ and t_6 only if both $u(c_3) = 1$ and $u(c_4) = 1$. In a CtlPN, it may not be possible to disable individually each "controllable" transition. For instance, if $u(c_1) = 0$, both t_7 and t_4 are disabled, and if $u(c_1) = 1$, both t_7 and t_4 are control-enabled. This makes the controllability concept of CtlPNs more general than the one of section 4.1.

A modeling example illustrating this kind of controllability is as follows. Consider a train-gate controller, at the crossing of a railway with a two-way road. There are two gates, one for each direction of the traffic. This system is modeled in Figure 2(b): firing t_1 corresponds to a vehicle entering the crossing from one direction, and firing t_3 corresponds to a vehicle entering from the other direction. The controller is only given the ability to either lower *both* gates or raise *both* gates. Thus, the controller cannot have one gate lowered and the other raised. This is modeled by controlling t_1 and t_3 with the same control place c_1.

4.3 State Observation

In the structural setting, transition firings are observed. However, we could observe instead markings (the state). In this case, limited observability corresponds to limited information on the marking of the system. As shown in [26], this can be modeled by a function $O : \mathcal{M} \to \{o_1, o_2, \ldots o_n\}$, mapping the set of markings $\mathcal{M}$ onto a set of observability classes o_1, o_2, $\ldots$ o_n.

As an illustration, consider again the manufacturing model of Figure 2(c). Recall, AGVs going in opposite directions can enter a common loading area; firing t_1 (t_2) corresponds to AGVs entering (exiting) in one direction, while firing t_3 (t_4) corresponds to AGVs entering (exiting) from the other direction. Assume only one AGV can be in the loading area at any time. Assume also that we can only detect the presence of an AGV in the loading area, but not its direction. Then, we cannot distinguish between the markings $\mu = [1, 0]^T$ and $\mu = [0, 1]^T$. So we can associate an observation class o_1 for the markings $[1, 0]^T$ and $[0, 1]^T$, and a class o_2 for the marking $[0, 0]^T$.

4.4 Labeled Petri Nets

The controllability and observability concepts of section 4.1 can be extended to labeled PNs. A **labeled PN** is a PN enhanced with a labeling function $\rho : T \to 2^{\Sigma} \cup \{\lambda\}$, where Σ is the set of events, ρ the labeling function, and λ the null event. Following the Ramadge-Wonham setting, Σ can be partitioned into controllable and uncontrollable events, $\Sigma = \Sigma_c \cup \Sigma_{uc}$ and observable and unobservable events $\Sigma = \Sigma_o \cup \Sigma_{uo}$. In this setting, when a transition t fires, an event $e \in \rho(t)$ is generated. If $e \in \Sigma_c$ ($e \in \Sigma_o$), the supervisor is able to disable (observe) this event. Note that t is disabled by the supervisor only when all events $e \in \rho(t)$

are disabled. Compared to section 4.1, one difference is that two transitions t_1 and t_2 may produce the same event when fired. Here, a supervisor controls/observes transitions indirectly, by disabling/observing events.

The Ramadge-Wonham setting is usually associated with the no concurrency assumption. However, in the context of labeled PNs we can use also the other concurrency settings, allowing control-enabled transitions to fire at the same time, including multiple firings of individual transitions.

As an illustration, recall the train-gate controller example of section 4.2, modeled in Figure 2(b). We can model the same example with the structure of Figure 2(c) and a labeling ρ such that $\rho(t_1) = \rho(t_3)$. However, note that this model assumes not only that t_1 and t_3 cannot be individually controlled, but also that they cannot be individually observed (firing t_1 or t_3 produces the same event). This motivates introducing double-labeled PNs next.

4.5 Double-Labeled PNs

Double-labeled PNs combine the concepts of transition controllability and observability of CtlPNs and labeled PNs, respectively. A **double-labeled PN** is a PN enhanced with two labeling functions: $\rho : T \to 2^{\Sigma} \cup \{\lambda\}$ and $o : T \to \Omega \cup \{\lambda\}$, where ρ labels transitions with subsets of control events $e \in \Sigma$, and o labels transitions with observation events $o \in \Omega$. Thus, Σ (Ω) is the set of control (observation) events. The meaning of the two labellings is as follows: a transition $t \in T$ is control-enabled when there is an event $e \in \rho(t)$ that is enabled. Further, when t fires, the event $o(t)$ is generated. Note that when the underlying PN is safe and has a state machine structure, a double-labeled PN corresponds to a Mealy type automaton. In fact, the reachability graph of any double-labeled PN is a Mealy automaton.

By using the two labeling functions in the train-gate example (Figure 2(c)), we can model the situation in which vehicles entering from different directions produce different observation symbols ($o(t_1) \neq o(t_3)$), while the flow from one direction cannot be interrupted apart from the flow of the other direction ($\rho(t_1) = \rho(t_3)$).

4.6 Comparison

Clearly, the controllability concept of CtlPNs is more general than that of section 4.1, as in section 4.1 we assume each controllable transition can be individually disabled. Further, the setting of labeled PNs does not capture the ability to control only certain groups of transitions either. Indeed, it is known [26] that enabling groups of transitions as opposed to individual transitions, corresponds to a more unusual setting in the Ramadge-Wonham framework, in which not all combinations of controllable events can be disabled [22]. However, double-labeled PNs can model the type of uncontrollability of CtlPNs, as we show next. This is an important observation, for as we show in section 5.3, structural methods can deal with double-labeled PNs.

As an example, Figure 3(b) shows a double-labeled PN. The observation events are shown in Greek

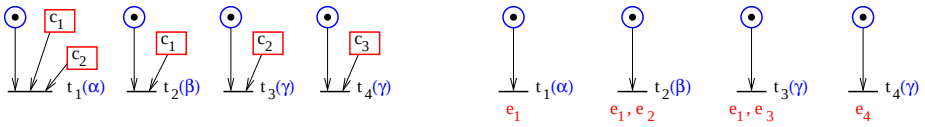

Figure 3:

letters. For instance, $o(t_1) = \alpha$ and $o(t_4) = \gamma$. The control events are the events e_i, $i = 1 \ldots 4$. For instance, $\rho(t_2) = \{e_1, e_2\}$ and $\rho(t_1) = \{e_1\}$. Note that the PN of Figure 3(b) implements the behavior of the CtlPN of Figure 3(a). In fact, it can be seen that any CtlPN can be converted to a double-labeled PN. Indeed, given a CtlPN $\mathcal{N}^c = (\mathcal{N}, \mathcal{C}, \mathcal{B})$, let $\mathcal{U}$ be the set of controls. The observation labeling o of $\mathcal{N}$ is assumed to be given, as we know from the beginning the observation events generated by transitions. It remains to construct the control labeling ρ. For each transition t, let $u_t \in \mathcal{U}$ denote the minimal control enabling t: $u_t(c) = 1$ when $(c, t) \in \mathcal{B}$ and $u_t(c) = 0$ otherwise. Let $\mathcal{U}_{min}$ be the set of minimal controls: $\mathcal{U}_{min} = \{u \in \mathcal{U} : u \text{ minimal for some } t \in T\}$. The set of control events Σ is constructed as follows: for each $u_k \in \mathcal{U}_{min} \setminus \{0\}$ associate a distinct event $e_k \in \Sigma$. Let's denote by $u[e_k]$ the control associated to an event e_k. Note that given a control u_j, a transition t of $\mathcal{N}^c$ is control-enabled when its minimal control u_t satisfies $u_t \leq u_j$. So we define $\rho(t) = \{e_j \in \Sigma : u_t \leq u[e_j]\}$. Thus, applying a control u to the CtlPN corresponds to enabling all events e_j with $u[e_j] \leq u$, resulting in the same transitions being control-enabled in the CtlPN and the double-labeled PN.

The converse is also true: a double-labeled PN can be converted to a CtlPN enhanced with an observation labeling o. That is, it is possible to replace the control labeling with control places.

From a supervision viewpoint, the definition of CtlPNs limits CtlPNs to the concurrency assumption. Indeed, under more general concurrency settings (which allow also multiple firings of the same transition at one time), the controls become very liberal, as they allow an unlimited number of firings for all enabled transitions. However, there is no such limitation for double-labeled PNs. For instance, in Figure 1(b), the number of simultaneous firings of t_1 can be limited by the marking of the monitor p_7.

Comparing the two observability settings, state observation and event observation, note that no setting is more general than the other, in the sense that a problem formulated in one setting may not be approachable in the other.

An example of problem that can be dealt with in the event observation setting, but not in the state observation setting, is as follows. Figure 4 shows a PN in which only t_1 is observable and only t_6 is controllable. Assume the initial marking is known and corresponds to the marking shown in Figure 4(a). The specification requires t_6 be disabled until t_1 fires, and then enabled. This problem is trivial in the event observation setting: the supervisor disables t_6 until it observes a firing of t_1. In the state observation setting,

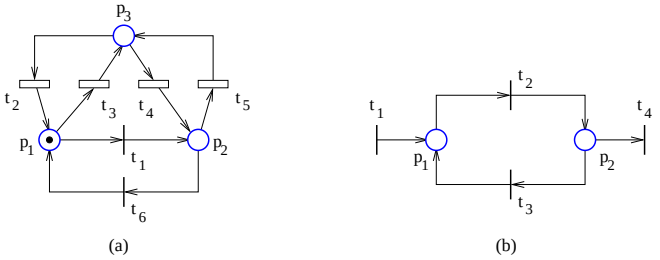

Figure 4:

note the following. There are three reachable markings: μ^1, μ^2, and μ^3, each corresponding to one token being in p_1, p_2, and p_3, respectively. Since t_2 and t_3 are unobservable, we must define the observation map O such that $O(\mu^1) = O(\mu^3)$. Similarly, we need $O(\mu^2) = O(\mu^3)$. It follows that all reachable markings must belong to the same observation class! Therefore, we have no information regarding whether t_6 should be enabled or not. In this particular example, it is still possible to solve the problem by changing the structure of the PN: let p_4 be a sink place added to t_1 (i.e. $\bullet p_4 = t_1$ and $p_4\bullet = \emptyset$). Since t_1 is observable, we can define two classes of markings: o_1 for markings with $\mu_4 = 0$, and o_2 for markings with $\mu_4 \geq 1$. Then, we disable (enable) t_6 whenever the marking is in o_1 (o_2).

On the other hand, a problem that cannot be treated in the event observation framework is as follows. In the PN of Figure 4(b), assume we have three observation classes: o_1 for markings with $\mu_2 = 0$, o_2 for $1 \leq \mu_2 \leq 2$ and o_3 for $\mu_2 \geq 3$. The specification is that t_4 may fire only if $\mu_2 \geq 3$, where t_4 is controllable. Regardless of the observation labels we choose for t_2, t_3, and t_4, there is no

Finally, note that from events we can estimate the state by means of observers. Work on PN observers appears in [20]. There, the initial marking is unknown, and the marking of the plant is estimated by observing the transitions. Supervising the PN for specifications (1) based on the estimated marking is also considered in [20]. However, as shown in [21], deadlock may arise in the enforcement of (1) due to estimation errors. Thus, a deadlock recovery solution is proposed in [20], based on integer programming and timing information on the firing delays of enabled transitions.

5 A Structural Approach to Supervision

When dealing with fully observable and controllable systems, we have seen that the SBPI provides a very simple and optimal solution for the design of supervisors. The result is summarized in Theorem 3.1. However, when uncontrollability and unobservability is present, the supervisor designed as in Theorem 3.1 may not be *admissible*. For instance, the supervisor may include monitors that are supposed to prevent plant-enabled uncontrollable transitions from firing, and may contain monitors with marking varied by firings of closed-loop

enabled unobservable transitions. Such a supervisor is clearly not implementable. A supervisor is admissible, when it respects the uncontrollability and unobservability constraints of the plant. 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

$$L_a\mu \leq b_a \Rightarrow L\mu \leq b \tag{11}$$

Then, the supervisor enforcing $L_a\mu \leq b_a$ is admissible, and enforces $L\mu \leq b$ as well.

Example 5.1 Assume t_2 and t_5 uncontrollable in Fig. 1(a). Then $\mu_2+\mu_5 \leq 1$ is not admissible, as enforcing it may attempt controlling either of t_2 and t_5. However, it can be checked that $\mu_1+\mu_2+\mu_5 \leq 1$ is admissible and $\mu_1+\mu_2+\mu_5 \leq 1 \Rightarrow \mu_2+\mu_5 \leq 1$. □

Various conditions on the constraints $L\mu \leq b$ could be used to guarantee $L\mu \leq b$ are admissible, such as the conditions presented later in this section. Given some admissibility conditions, the design approach is as follows:

Algorithm 5.2

1. Check whether the admissibility conditions are satisfied by the supervisor (2–3). If so, the supervisor is optimal and admissible.
2. If not, transform the specification $L\mu \leq b$ to $L_a\mu_a \leq b_a$ such that the admissibility conditions and (11) are satisfied.
3. Design the supervisor enforcing $L_a\mu \leq b_a$ as in (2–3).

Various design methods result, depending on the admissibility conditions used in the algorithm and the approach used at the second step. Being known that a minimally restrictive solution may not correspond to a convex region $L_a\mu \leq b_a$ [18], one can give up the requirement that the transformed specification is a conjunction $L_a\mu \leq b_a$, and allow disjunctions $\bigvee_i L_{a,i}\mu \leq b_{a,i}$. In either case, the method is suboptimal whenever the admissibility conditions used in the algorithm are not necessary. In this section we describe "structural" admissibility constraints, which are sufficient for admissibility, but not necessary. While they may result in suboptimal designs, the structural admissibility conditions have the advantage that they have allowed the development of computationally efficient methods for supervisor design. The rest of this section presents structural admissibility conditions for three cases:

1. Individually controllable and observable transitions
2. Labeled PNs

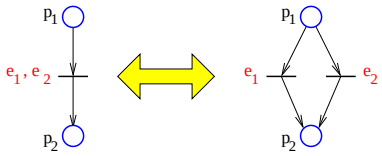

Figure 5:

3. Double-labeled PNs

We will see that in the first two cases the conditions have the form $LA \leq 0$, and in the third disjunctions $\bigvee_{i=1}^{n} LA_i \leq 0$. Note that any method that is applied at the second step of the Algorithm 5.2 and that relies on conditions $LA \leq 0$, can also be used for admissibility conditions $\bigvee_{i=1}^{n} LA_i \leq 0$. This is how. First, given a constraint $l\mu \leq c$ of $L\mu \leq b$, find for every $i = 1 \ldots n$ a solution $l_a\mu \leq c_a$ satisfying $l_a A_i \leq 0$ and (11). Then, select the "best" solution $l_a\mu \leq c_a$ out of the n cases $i = 1, 2, \ldots n$. Finally, take $L_a\mu \leq b_a$ as the conjunction of the constraints $l_a\mu \leq c_a$ that were selected for each constraint $l\mu \leq c$ of $L\mu \leq b$.

5.1 Individually controllable and observable transitions

If T_{uc} denotes the set of uncontrollable transitions, the supervisor (2–3) controls only the controllable transitions if all elements of $LD(\cdot, T_{uc})$ are nonnegative [8, 51, 52], which is written as:

$$LD(\cdot, T_{uc}) \leq 0 \tag{12}$$

Further, to ensure that the supervisor (2–3) detects only the observable transitions it is sufficient to require [51, 52]:

$$LD(\cdot, T_{uo}) = 0 \tag{13}$$

where T_{uo} is the set of unobservable transitions. Given (1) and an initial marking μ_0, (12) and (13) are only sufficient for admissibility. However, if L is fixed and μ_0 and b are variables, we have the following optimality property.

Theorem 5.1 [32] *The supervisor of (2–3) is admissible for all μ_0 and $b \geq L\mu_0$ iff L satisfies (12–13).*

This result can be exploited for fault-tolerant supervisory control [36].

5.2 Labeled PNs

Without loss of generality, we may assume the labeling to be defined as $\rho : T \to \Sigma \cup \{\lambda\}$ instead of $\rho : T \to 2^{\Sigma} \cup \{\lambda\}$, as illustrated in Figure 5. In this way, each transition is labeled with a single event[2]

[2]The transitions that originally had no label are labeled with λ, the null event.

$e \in \Sigma \cup \{\lambda\}$. Then we can write the following sufficient conditions for admissibility:

$$\forall t_1, t_2 \in T,\ \rho(t_1) = \rho(t_2) \quad \Rightarrow \quad LD(\cdot, t_1) = LD(\cdot, t_2) \tag{14}$$

$$\forall t \in T,\ \rho(t) \in \Sigma_{uc} \cup \{\lambda\} \quad \Rightarrow \quad LD(\cdot, t) \leq 0 \tag{15}$$

$$\forall t \in T,\ \rho(t) \in \Sigma_{uo} \cup \{\lambda\} \quad \Rightarrow \quad LD(\cdot, t) = 0 \tag{16}$$

Note that (14–16) can be written compactly as $LA \leq 0$, for some matrix A. This means that the same methods used for finding L_a and b_a subject to (11) and (12) or (11–13) can be applied also here, by replacing (12) with $LA \leq 0$.

5.3 Double-Labeled PNs

Again, without loss of generality, we may assume the control labeling to be defined as $\rho : T \to \Sigma \cup \{\lambda\}$ instead of $\rho : T \to 2^{\Sigma} \cup \{\lambda\}$. Here, it is more convenient to write the conditions in terms of single constraints $l\mu \leq c$ instead of sets of constraints $L\mu \leq b$. We have the following sufficient conditions for the admissibility of $l\mu \leq c$ (note that $L\mu \leq b$ is admissible if all its constraints $l\mu \leq c$ are admissible):

$$\forall t_1, t_2 \in T,\ o(t_1) = o(t_2) \quad \Rightarrow \quad lD(\cdot, t_1) = lD(\cdot, t_2) \tag{17}$$

$$\forall t \in T,\ o(t) \in \Sigma_{uo} \cup \{\lambda\} \quad \Rightarrow \quad lD(\cdot, t) = 0 \tag{18}$$

$$\forall t \in T,\ \rho(t) = \lambda \quad \Rightarrow \quad lD(\cdot, t) \leq 0 \tag{19}$$

$$\begin{aligned} &\forall t_1, t_2 \in T,\ \forall \alpha \in \Sigma_{uc},\ \rho(t_1) = \rho(t_2) = \alpha \Rightarrow \\ &\qquad lD(\cdot, t_1) = lD(\cdot, t_2) \vee [lD(\cdot, t_1) \leq 0 \wedge lD(\cdot, t_1) \leq 0] \end{aligned} \tag{20}$$

Due to the constraint (20), the conditions for the admissibility of $L\mu \leq b$ are no longer linear. Instead, they have the form $\bigvee_{i=1}^{n} LA_i \leq 0$.

6 Supervision Methods

6.1 Admissibility Based Methods

Here, we refer to the Algorithm 5.2, and describe methods that can implement the second step of the algorithm. The methods presented here are based on the admissibility conditions (12) and (13). They assume free-labeled PNs with individually controllable and observable transitions. However, as noticed in section 5, such methods can be easily adapted to the more general settings of labeled or double-labeled PNs.

The design of admissible constraints has been approached in [51, 52] using the following parameterization:

$$L_a = R_1 + R_2 L \tag{21}$$

$$b_a = R_2(b + \mathbf{1}) - \mathbf{1} \tag{22}$$

where R_1 is an integer matrix with nonnegative elements and R_2 is a diagonal matrix with positive integers on the diagonal. This parameterization is used as a sufficient condition for (11). Thus, at the step 2 of Algorithm 5.2, the constraints (21–22) replace (11). Now, the problem is to find L_a and b_a subject to (21–22), (12) and (13). This is a linear integer programming problem for which, sometimes, solutions may be found using an efficient matrix row operation algorithm [51, 52]. Note that this integer programming formulation of the problem allows introducing additional requirements of interest. For instance, communication constraints and a minimum-communication objective were used in a distributed version of this problem [35]. While the approach of [51, 52] is computationally efficient, it is also suboptimal. That is, a solution may not be found when solutions exist, and if one is found, it may not be the least restrictive solution. A source of suboptimality is that the computation is not constrained to ensure that if L'_a and b'_a are another solution to (21–22), (12) and (13), then $L_a\mu \leq b_a \not\Rightarrow L'_a\mu \leq b'_a$.

The approach of [51, 52] can been improved in several ways. First, it should be noticed that it is difficult to express by linear inequalities the requirement that $L_a\mu \leq b_a$ should be as permissive as possible. However, it is easy to constrain the computation of L_a and b_a to guarantee some weaker properties: (a) that a set of markings of interest is included in $\{\mu : L_a\mu \leq b_a\}$ and (b) that a set of firing count vectors x is included in $\{x : D_c x \geq 0\}$, where D_c is the incidence matrix of the closed-loop. These simple extensions can be found in [35]. As noticed in [2], the admissible constraints $L_a\mu \leq b_a$ satisfying (11) may not have a unique supremal element. Thus, further work has been done by the authors of [1] towards finding the supremal constraints $L_a\mu \leq b_a$ subject to (21–22), (12) and (13) by means of a parameterization.

Another way to control the selection of L_a and b_a is by means of observation and control costs. Thus, in [3], the optimal design of supervisors is considered, where optimality here is with respect to control and observation costs. Here, instead of having sets of uncontrollable and unobservable transitions T_{uc} and T_{uo}, we have maps $z_c : T \to \mathbb{R}^+$ and $z_o : T \to \mathbb{R}^+$, associating control and observation costs to each transition. The setting of [3] is general, as we can still consider some transitions as uncontrollable/unobservable by associating with them very large control or observation costs. The design problem of [3] is solved by an integer programming approach, using (21–22) and admissibility conditions equivalent to (12) and (13).

The optimal design of supervisors with respect to the admissibility constraints (12) and (13) is approached also in chapter 8 of [60]. The proposed method applies to specifications (1) in which for all rows of L, all elements on a row have the same sign. Note that the solution is given in the form of a disjunction of constraints.

Still another approach appears in [7]. The setting of [7] assumes full observability. Essentially, given the constraint $l\mu \leq c$ with $l \in \mathbb{N}^m$ and $c \in \mathbb{N}$, $l\mu \leq c$ is replaced with the disjunction

$$\bigvee_{l_i \in SD_{min}(l)} [l_i\mu \leq c] \tag{23}$$

where $SD_{min}(l)$ is the set of minimal integer vectors x satisfying $x \geq l$ and $xD(\cdot, T_{uc}) \leq 0$. In particular, $l\mu \leq c$ is replaced with the single admissible constraint $l_1\mu \leq c$ when $SD_{min}(l)$ is the singleton $\{l_1\}$. Under the conditions of [8, 6], which are discussed later in section 6.2, the resulting supervisor is least restrictive. It is interesting to notice that some of the assumptions of [7] can be dropped. Indeed, (23) is still a valid supervisor even if $l \in \mathbb{Z}^m$ and $c \in \mathbb{Z}$ (as opposed to $l \in \mathbb{N}^m$ and $c \in \mathbb{N}$). Further, partial observability can be incorporated by defining $SD_{min}(l)$ as the set of minimal integer vectors x satisfying $x \geq l$, $xD(\cdot, T_{uc}) \leq 0$ and $xD(\cdot, T_{uo}) = 0$.

6.2 Path-based approaches

Here we outline other structural approaches from the literature. In the literature, there are several results dealing with the supervision of marked graphs. We begin by outlining how these results can be used for enforcing specifications (1), when the plant is a marked graph and various other modeling assumptions are satisfied. Then, we will present some other results that deal with more general PN models.

Powerful results for the supervision of marked graphs were first obtained in [27, 42]. The setting of [27, 42] is as follows. The plant is a CtlPN (see section 4.2), in which the underlying PN is a cyclic marked graph with an initial marking that places exactly one token in every directed cycle. Thus, the PN is safe (i.e., all reachable markings are binary vectors). Full observability is implicitly assumed. The supervisory goal is to avoid a set of forbidden markings $\mathcal{M}_F$. While in [27] $\mathcal{M}_F$ is specified in terms of *place*, *set* and *class conditions*, [42] specifies $\mathcal{M}_F$ as the complement of the feasible set a set of constraints (1):

$$\mathcal{M}_F = \bigcup_{(F,k)\in\mathcal{F}} \left\{ \mu : \sum_{p\in F} \mu(p) > k \right\} \tag{24}$$

Note that both *class conditions* and (24) can specify any set $\mathcal{M}_F$, due to the fact that the PN model is a safe cyclic marked graph. (In this survey we will show how to obtain inequalities (1), not (24) though, from any set $\mathcal{M}_F$ of a safe PN. This is done in section 7.3.) There are some mild assumptions on the set $\mathcal{M}_F$ in [27, 42]. As mentioned in [28], these assumptions guarantee the supervisor designed is least restrictive.

The design of supervisors in [27] is approached by analyzing the paths of the marked graph that do not involve controllable transitions. This solution is simplified in [42]. The solution of [42] involves the following: identify a number of paths in the marked graph, offline; evaluate certain place and path predicates, online. Note that in [27, 42] the supervisor is not represented as a Petri net.

In [5], the design of supervisors is studied for a setting similar to that of [27, 42], in which the CtlPN has a state machine structure. The forbidden set here is described by disjunctions of constraints of the form (1). The use of disjunctions is necessary in order to describe arbitrary sets of forbidden states, as the PN is not assumed to be safe. The supervisors obtained in [5] are not represented as Petri nets.

The results of [27, 42] are generalized in [29], by extending the plant model from marked graphs to arbitrary ordinary PNs. The type of specifications is similar to that of [27]. However, as the PNs may not be safe, it cannot capture all possible sets of forbidden markings. Further, compared with (1), the specifications of [29] are neither a subset nor a superset of the specifications expressed by (1). As in the previous work [27, 42], in [29] the least restrictive supervisor is found by a path based approach.

6.3 Controlled-invariant approaches

The setting of [27, 42] and also of other subsequent papers can be described as follows. A supervisor can avoid the states in $\mathcal{M}_F$ if it avoids a larger set $\mathcal{A}_F \supseteq \mathcal{M}_F$, where $\mathcal{A}_F \setminus \mathcal{M}_F$ describe the markings μ that lead to $\mu' \in \mathcal{M}_F$ by firing only uncontrollable transitions. Let T_{uc} be the set of uncontrollable transitions and $\mathcal{N}_u = (P, T_{uc}, D^-(\cdot, T_{uc}), D^+(\cdot, T_{uc}))$ a subnet of the plant $\mathcal{N}$ that does not contain the controllable transitions. Then $\mathcal{A}_F$ can be expressed as:

$$\mathcal{A}_F = \{\mu : \mathcal{R}(\mathcal{N}_u, \mu) \cap \mathcal{M}_F = \emptyset\} \tag{25}$$

This set is known as the *maximal controlled-invariant set* [42, 56]. The approaches discussed above design supervisors such that the states leading to $\mathcal{A}_F$ are avoided. Thus, $\mathcal{A}_F$ is not explicitly computed. However, a possible approach to supervision is to compute $\mathcal{A}_F$. Once we know $\mathcal{A}_F$, the control task is simply to disable any control actions that lead to a marking in $\mathcal{A}_F$. Note that avoiding $\mathcal{A}_F$, as opposed to some superset $\mathcal{E} \supseteq \mathcal{A}_F$, corresponds to least restrictive supervision. In particular, as noticed in [18], solutions replacing a specification $L\mu \leq b$ with an admissible $L_a\mu \leq b_a$ correspond to supervisors that avoid supersets $\mathcal{E} \supseteq \mathcal{A}_F$, since $\mathcal{A}_F$ may not be representable as the complement of a set of constraints of the form (1) even when $\mathcal{M}_F$ is given as the complement of a set of constraints (1). We discuss briefly below literature methods that compute $\mathcal{A}_F$.

In [6] specifications (1) are considered, where L and b are restricted to have only nonnegative elements. Given $l\mu \leq c$ as one of the constraints of (1), (so $l \in \mathbb{N}^m$ and $c \in \mathbb{N}$), the influential subnet $\mathcal{N}_u^l$ is defined, which is the subnet of $\mathcal{N}_u$ containing the places p with $l(p) \neq 0$ and the directed paths of $\mathcal{N}_u$ to these places. The main result of the paper shows how to express $\mathcal{A}_F$ as the set of markings satisfying a disjunction of linear marking inequalities. This result relies on two conditions, as follows. First, $\mathcal{N}_u^l$ should be a marked graph. (Note that $\mathcal{N}_u^l$, not $\mathcal{N}$, is restricted to a marked graph structure.) Second, for all reachable markings of $(\mathcal{N}, \mu_0)$, every directed circuit of $\mathcal{N}_u^l$ should have at least one token. In [6] the supervisor is not represented as a PN. However, the subsequent work of [7] proposes an extended PN representation of the supervisor, in which negative markings are allowed. Note that a similar result was obtained in [8] for the case in which $\mathcal{N}_u^l$ is a state machine, instead of a marked graph. For this case, it is shown that $\mathcal{A}_F$ has the form $\mathcal{A}_F = \{\mu : l_a\mu \leq c\}$, where l_a can be easily computed. Thus, the monitor enforcing $l_a\mu \leq c$ is the least

restrictive supervisor.

The efficient computation of [8] for PNs and specifications for which the subnets $\mathcal{N}_u^l$ are state machines, may not be surprising in light of the complexity findings of [55]. The model of [55] is as follows. The plant consists of p components that do not interact with each other, where the components are represented by deterministic Büchi automata $G_i = (Q_i, \Sigma_i, \delta_i, q_{0i}, Q_{mi})$ over disjoint alphabets Σ_i. Given the subsets of states $\overline{Q}_i \subset Q_i$, a mutual exclusion specification requires less than k components to have their states q_i in $\overline{Q}_i$ at the same time. Note that the plant of [55] can be represented by a safe labeled PN with a state machine structure, and the mutual exclusion constraint by a constraint $l\mu \leq c$ in which $c = k$ and all elements of l are 0 or 1. One of the problems considered in [55] is to find nonblocking coordinators that enforce the mutual exclusion constraint. Roughly, a nonblocking coordinator is a supervisor that guarantees certain strong liveness properties. The paper shows that the existence of a solution can be decided in polynomial time in p and n, where $n = \max_i |Q_i|$. Further, it is shown that if a solution exists, the minimally restrictive solution can be found in polynomial time in p and n. It is interesting to note that in the equivalent PN representation of the plant, the supervisor found in [55] corresponds to a monitor place enforcing a constraint $l_a\mu \leq c$, provided the PN is free-labeled. Note also that in view of [23], the assumption that the sets Σ_i are disjoint seems to be critical for polynomial complexity. In [23] it is shown that when the components of the plant have a shared event, the solvability of the problem can no longer be decided in polynomial time. A restriction of the problem for which polynomial complexity is maintained is also proposed.

A method that finds the optimal design for specifications (1) appears in [49]. Several assumptions are made, as seen from the following outline of the method. Let $\mathcal{L}(\mathcal{N}_u, \mu)$ denote the set of firing sequences σ of $\mathcal{N}_u$ that are enabled at the marking μ. Let $\underline{\sigma}$ be the firing count vector with respect to $\mathcal{N}$ (not $\mathcal{N}_u$). Finally, let $l\mu \leq c$, $l \in \mathbb{Z}^{|P|}$ and $c \in \mathbb{Z}$, denote a single constraint of (1). The set $\mathcal{A}_F$ corresponding to $l\mu \leq c$ is given by $\mathcal{A}_F = \{\mu : (\forall \sigma \in \mathcal{L}(\mathcal{N}_u, \mu))\, l\mu + lD\underline{\sigma} \leq c\}$. By assuming $\mathcal{N}_u$ (not $\mathcal{N}$) to be acyclic, $\mathcal{A}_F = \{\mu : l\mu + lDv^*(\mu) \leq c\}$, where $v^*(\mu)$ is the solution of the linear integer program $\max lDv$ subject to $D(\cdot, T_{uc})v \geq -\mu$ and $v \geq 0$. As shown in [49], a closed-form expression of $\mathcal{A}_F$ can be computed under additional assumptions. First, [49] defines subnets for each $t \in T_{uc}$, consisting of all paths of $\mathcal{N}_u$ ending in t. Denoting by $\hat{T}_{uc} = \{t \in T_{uc} : lD(\cdot, t) > 0\}$, [49] requires all subnets of $t \in \hat{T}_{uc}$ be independent (disjoint). Further, when the subnets have the $TS1$ structure described in [49], $\mathcal{A}_F$ can be expressed by a disjunction of inequalities: $\mathcal{A}_F = \{\mu : \bigvee_{i=1}^{k} l_i\mu \leq c\}$ for some k and $l_i \in \mathbb{Z}^{|P|}$. Moreover, when the subnets have the $TS2$ structure described in [49], then $\mathcal{A}_F = \{\mu : l_a\mu \leq c\}$ for some $l_a \in \mathbb{Z}^{|P|}$. Thus, in the $TS1$ case the optimal supervisor of $l\mu \leq c$ enforces $\bigvee_{i=1}^{k} l_i\mu \leq c$, and in the $TS2$ case $l_a\mu \leq c$. The approach of [49] is computationally efficient, as $\mathcal{A}_F$ is calculated independently of μ and without resorting to the traditional methods for solving integer programs.

Results on the supervision of marked graphs appear in [15]. Compared to [27, 42], the marked graphs

considered here may not be safe. However, unlike to [27, 42], the results are presented in the no concurrency setting and the uncontrollability model is simpler: the set of transitions is partitioned into controllable (T_c) and uncontrollable (T_{uc}) transitions. The specifications have the form (1). A least restrictive supervision policy is computed first for several particular cases. This policy is very efficient, as it involves little online computations. Finally, a supervision policy is proposed for the general case, which involves solving online linear programs, for every reachable marking. This last result is based on the observation that given a constraint $l\mu \leq c$, $l \in \mathbb{Z}^{1\times m}$ and $b \in \mathbb{Z}$, finding $\max\{l\mu^* : \mu^* \in \mathcal{R}(\mathcal{N}_u, \mu)\}$ is equivalent to the integer linear program $\max\{l\mu^* : \mu^* = \mu + D(\cdot, T_{uc})q,\, q \in \mathbb{N}^{|T_{uc}|}\}$, which is equivalent to the linear program $\max\{l\mu^* : \mu^* = \mu + D(\cdot, T_{uc})q,\, q \in \mathbb{R}_+^{|T_{uc}|}\}$. These two equivalences result from the fact that the plant is a live marked graph.

6.4 Methods for partial observability

Partial observability, as long as described by some events being observable and others not, can be easily dealt with in the setting of the admissibility-based methods. The admissibility-based methods were presented in section 6.1. However, more substantial extensions are needed in order to incorporate partial observability in the methods of sections 6.2 and 6.3. This section presents several methods, which are not admissibility-based, and which can deal with partial observability.

The extension of the path-based approach of Holloway and Krogh [27, 42] to partial observability appears in [69]. Ordinary PN structures are considered, instead of marked graphs. The set of transitions is partitioned in controlled and uncontrolled transitions, $T = T_c \cup T_{uc}$, and in observed and unobserved transitions, $T = T_o \cup T_{uo}$, with $T_o \supseteq T_c$. Note that a transition is controlled if connected to some control place. Further, each transition is labeled by one event, and a transition is observed if its label is not the null event. The authors propose a path algebra, described in more detail in [29]. This algebra is used to define reachability predicates, which are then used to define the least restrictive control policy. (The supervision is nondeterministic, so least restrictive control policies exist.)

Several important results on the control of live marked graphs appear in [11]. The specification considered there is more powerful than (1), as it has the form $av \leq c$, where v is the Parikh vector, $a \in \mathbb{Z}^{1\times n}$ and $c \in \mathbb{Z}$. In [11], the set of transitions T is partitioned into the disjoint subsets: $T = T_c \cup T_f \cup T_i$, where T_c is the set of controllable transitions, and $T_o = T_c \cup T_f$ the set of observable transitions. The approach of the paper is as follows. *Suspect* vectors are defined as Parikh vectors v such that $v|_{T_o} = v'|_{T_o}$ for some v' with the property that after firing v', a forbidden state v'' could be reached (i.e. $av'' > c$), by firing only uncontrollable transitions. The paper shows that any deterministic supervisor has to avoid reaching the set of suspect vectors, and that the projections of these vectors on T_o form a convex set (that is, the set of integral points of a polyhedron). The paper shows also how to compute this set. Since the complement

of this set may not be convex, it follows that the least restrictive supervisor may not be implementable by control (monitor) places. Even when monitors can be used, the paper shows that the number of monitors may be exponential. Another observation of the authors is that the number of linear constraints defining the set of suspect vectors may depend exponentially on the size of $D(\cdot, T_{uc})$. The alternative to the computation of this set is as follows. Given a state v_0, a linear program can be solved in order to decide whether $t \in T_c$ should be enabled. Since linear (not linear integer) programming is used, the computation has polynomial complexity.

In [14], the supervisory control problem is approached based on the reachability graph. Here, the supervisor is designed as a set of monitors acting upon the PN plant. First, a subset of the reachability graph is obtained, such that from any of the markings of the subgraph, forbidden states and blocking states cannot be reached by firing uncontrollable transitions. This subgraph becomes the desired reachability graph that is to be achieved by the closed-loop. Then, the authors deal with the design of supervisors that ensure the closed-loop has the specified reachability graph. Given a set Ω containing the pairs (μ, t) such that t should be disabled at the marking μ, monitors are designed, such that each monitor deals with at least one of the pairs (μ, t) of Ω. The connections of a monitor to the plant are determined by finding an integer solution to a system of inequalities (which corresponds to integer programming, [58]).

6.5 Decentralized Control

The decentralized control of PNs is approached in [34, 35]. The setting is as follows. A PN $\mathcal{N} = (P, T, D^-, D^+)$ is given, representing the plant. The plant has m subsystems, each having a set of controllable transitions $T_{c,i} \subseteq T$ and a set of observable transitions $T_{o,i} \subseteq T$, for $i = 1 \ldots m$. In this setting, we are to design m supervisors $\mathcal{S}_i$, each allowed to disable transitions $t \in T_{c,i}$ and observe transitions $t \in T_{o,i}$, such that the joint operation of the supervisors $\mathcal{S}_i$ ensures the specification (1) is satisfied. In [34] a decentralized admissibility concept is introduced, called **d-admissibility**. In [34] it is shown that d-admissibility can be checked by a structural approach similar to that of section 4.1. Several cases have been studied:

1. The specification (1) is d-admissible.
2. The specification (1) is not d-admissible and communication of transition firings is allowed.
3. The specification (1) is not d-admissible and communication is not allowed or is restricted.

Case 1 is solved by a construction similar to that of (2) and (3). Case 2 is reduced to case 1 by allowing event communication add more elements to the sets $T_{c,i}$ and $T_{o.i}$. However, case 3 is more involved. Assuming no communication is allowed, the problem is to decompose the specification (1) into sets of constraints $L_1\mu \leq b_1$

$\ldots L_r\mu \leq b_r$ such that each $L_i\mu \leq b_i$ is d-admissible and

$$(L_1\mu \leq b_1 \wedge L_2\mu \leq b_2 \wedge \ldots L_r\mu \leq b_r) \Rightarrow L\mu \leq b \tag{26}$$

The d-admissibility requirement can be tested by inequalities similar to (12) and (13). In [35] this problem is approached using the parameterization (21–22), by replacing (26) with the conservative requirement that

$$L_1 + L_2 + \ldots L_m = R_1 + R_2 L \tag{27}$$

$$b_1 + b_2 + \ldots b_m = R_2(b + \mathbf{1}) - \mathbf{1} \tag{28}$$

where R_1 has nonnegative integer elements and R_2 is diagonal with positive integer elements on the diagonal. Integer programming is then used to find L_i, b_i, R_1 and R_2. The problem is solved in a similar way when communication is allowed.

7 Expressiveness of the constraints

This section reports several situations in which problems involving constraints of a different form than (1) can be reduced to problems involving constraints (1). First, we consider a class of general linear constraints that correspond to the languages of the free-labeled PNs. Next, we consider constraints expressing general PN languages. Then, we consider logical constraints for safe PNs. We continue with disjunctions of constraints (1) under some boundedness assumptions. Finally, results showing (1) can express constraints for liveness enforcement are presented. The section ends with a discussion concerning the implications of the presented results.

7.1 Generalized Constraints

An interesting class of linear constraints that can be represented in the form (1) by PN transformations is given by

$$L\mu + Hq + Cv \leq b \tag{29}$$

where q is the firing vector and v the Parikh vector. To simplify our presentation, we will focus on the no concurrency assumption. The general case can be found in [32]. Under the no concurrency assumption, $q \in \{0,1\}^n$, $n = |T|$, identifies the transition that is to be fired next: $q_i = 1$ if t_i is to be fired next, and $q_i = 0$ otherwise. Recall, the Parikh vector $v \in \mathbb{N}^n$ records how many times each transition has fired. For instance, $v_1 = 4$ indicates t_1 has fired four times. q and v are illustrated in Figure 6. Further, $H \in \mathbb{Z}^{n_c \times n}$ and $C \in \mathbb{Z}^{n_c \times n}$ are matrices, and n_c is the number of constraints.

The constraints (29) are interpreted as follows. A supervisor enforcing (29) ensures that: (i) all states (μ, v) satisfy $L\mu + Cv \leq b$; (ii) if q is the firing vector of a transition t_i, $\mu \xrightarrow{t_i} \mu'$, and $v' = v + q$, then $L\mu + Hq + Cv \leq b$ and $L\mu' + Cv' \leq b$.

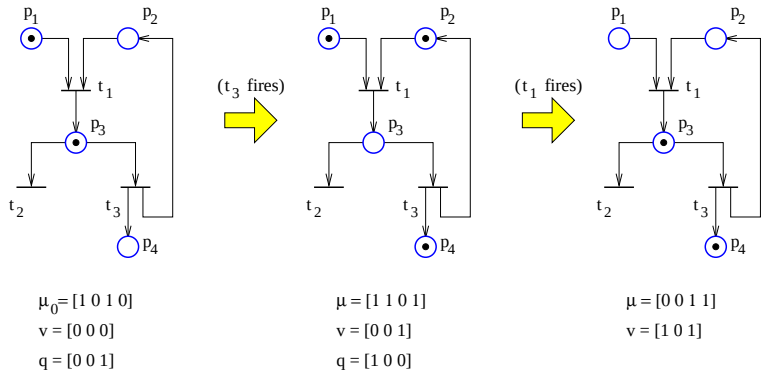

Figure 6: Illustration of the q and v parameters.

In [39] it is shown that:

- the class of constraints

$$Hq + Cv \leq b \tag{30}$$

  is as general as the class $L\mu + Hq + Cv \leq b$. That is, given $L\mu + Hq + Cv \leq b$, there is C' such that $L\mu + Hq + Cv \leq b$ and $Hq + C'v \leq b$ are equivalent.

- any monitor arbitrarily connected to the places of a Petri net can be described as enforcing a constraint of the form (30), where b corresponds to the initial marking of the monitor.

- in fact, any PN $(\mathcal{N}, \mu_0)$, $\mathcal{N} = (P, T, D^-, D^+)$, can be described by constraints (30), for $H = D^-$, $C = D^- - D^+$ and $b = \mu_0$.

- Consequently, the specifications (29) correspond to the P-type languages of the free-labeled PNs! (Following [53], a labeled PN is freely-labeled when each transition of the net has a unique and distinct label, different from λ, the null symbol; further, a language $\mathcal{L}$ is a P-type PN language if there is a PN with an initial marking such that $\mathcal{L}$ consists of the words associated with the firing sequences enabled by the initial marking.)

Let $\mathcal{L}$ be the language corresponding to all behaviors accepted by a specification (29). It is important to note that the specification (29) does not require the closed-loop to generate $\mathcal{L}$. Rather, it requires the closed-loop language to be a sublanguage of $\mathcal{L}$. Further, note that the least restrictive supervisor enforcing (29) can be easily designed under full controllability and observability assumptions [39, 32].

Another important result that appears in [39, 32] shows that under the partial controllability and observability setting of section 4.1, the design of supervisors enforcing (29) can be reduced to the design of supervisors enforcing (1) [39, 32]. Thus, if (29) is to be enforced on a PN $\mathcal{N}$, the problem is transformed into the design of a supervisor enforcing constraints of the form (1) on a PN $\mathcal{N}_H$. The solution to this problem is

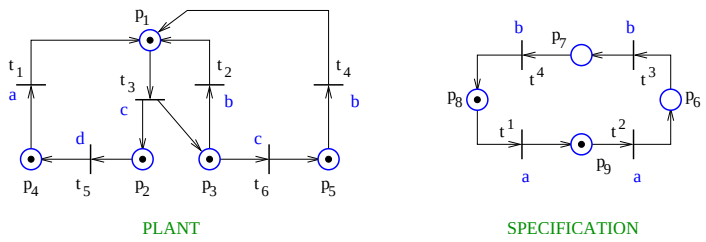

Figure 7:

then used to obtain the solution to the original problem of designing a supervisor enforcing (29) on $\mathcal{N}$. The uncontrollability and unobservability setting used in [39, 32] is that of section 4.1.

7.2 Language Constraints

As shown above, we can reduce the problem of enforcing certain PN languages to the enforcement of constraints (1). The plants considered there are free-labeled PNs and the specifications are P-type PN languages of free-labeled PNs. This section shows that we can approach in a similar way more general problems, that do not assume free-labeling for the plant and the specification. As in the previous considerations, the requirements here are that the closed-loop generates a sublanguage of the specification.

As an example, consider the PN and the specification shown in Figure 7. In this example, the specification is described by a PN labeled by the events a and b. To simplify the notation, it is assumed that all events of the plant that do not appear in the specification are always enabled in the specification. The closed-loop in our example can be computed immediately by a parallel composition of the plant and specification, and is shown in Figure 8(a). Note that in the closed-loop, the transition t_1 of the plant appears in the form of t_1^1 and t_1^2, corresponding to the synchronization of t_1 with the transitions t^1 and t^2 of the supervisor. Similarly, t_2^3 and t_2^4 correspond to the synchronization of t_2 with t_3 and t^4. A formal description of the algorithm composing PN plants with PN specifications can be found in [19].

The supervision is interpreted as follows. The plant and the supervisor have each a distinct set of transitions, T_p and T_s, respectively. The supervisor cannot observe/control the plant transitions directly, but it can observe/control events generated by the plant. When the plant generates the event a, the supervisor picks one of its own enabled transitions $t \in T_s$ that is labeled by a, and fires it. Note that the supervisor is free to choose which of its enabled transitions labeled by a fires. For instance, in Figure 7, when the plant generates a, the supervisor can select either of t^1 or t^2, since both are enabled and labeled by a. So we can relabel the closed-loop, to indicate the supervisor can distinguish between its own transitions that have the same label. Thus, in Figure 8 we have the following new labels: a^1 for t_1^1, a^2 for t_1^2, b^3 for t_2^3 and t_4^3, and b^4 for t_2^4 and t_4^4.

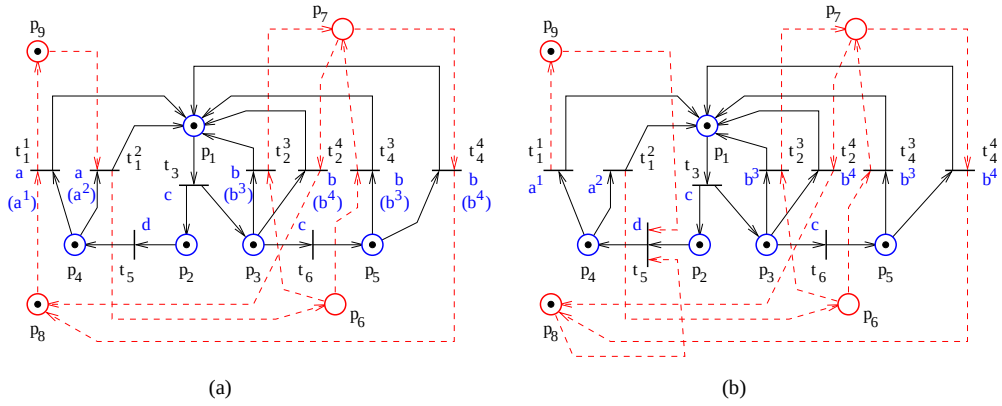

Figure 8:

According to our previous section 7.1, each place of the supervisor that appears in the closed-loop corresponds to a specification in terms of constraints (29). For instance, p_9 enforces $v_1^2 - v_1^1 \leq 1$ and p_8 enforces $v_1^1 - v_2^4 - v_4^4 \leq 1$. This gives us a readily available approach for supervisor design in the case of partial controllability and partial observability:

- Compose the PN plant and the PN specification (supervisor).
- Relabel the closed-loop, to take in account the supervisor can distinguish between its own transitions.
- Find the constraints (29) corresponding to the constraints enforced by the monitors of the closed-loop.
- Transform the constraints (29) to an admissible form, which is at least as restrictive.

For instance, assume in our example that t_1 (the event a) is uncontrollable but the other transitions are controllable. Assume all other events are observable. Notice that in Figure 8(a) p_8 and p_9 may attempt disabling t_1. So, the specification is inadmissible. However, the constraints enforced by p_8 and p_9, namely $v_1^1 - v_2^4 - v_4^4 \leq 1$ and $v_1^2 - v_1^1 \leq 1$, can be transformed to the admissible form $v_1^1 - v_2^4 - v_4^4 + \mu_4 \leq 1$ and $v_1^2 - v_1^1 + \mu_4 \leq 1$. The resulting closed-loop and supervisor are shown in Figure 8(b) and Figure 9, respectively. The supervision is admissible, while ensuring the plant generates only words that satisfy the original specification of Figure 7.

It is known that the supremal controllable sublanguage of a P-type PN language may not be a P-type PN language [16]. This is an indication that the approach presented here is suboptimal, in the sense that it may not lead to the least restrictive supervisor. Note that in the literature it has been shown that the computation of the least restrictive supervisor can be reduced to a forbidden marking problem, provided both the plant and specification generate deterministic languages [43]. (Given a labeled PN $(\mathcal{N}, \rho, \mu_0)$, the P-language it generates is deterministic if for any of its strings w, there is a unique transition sequence σ

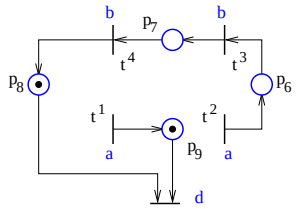

Figure 9:

enabled by μ_0 that generates w: $\rho(\sigma) = w$.) In the setting of [43], partial controllability and full observability are assumed.

7.3 Logical Constraints

This section shows that for safe PNs, the enforcement of logical constraints can be reduced to the enforcement of constraints (1). Recall, a PN $(\mathcal{N}, \mu_0)$ is safe if all reachable markings are binary vectors. In the literature, the observation that logic constraints on the marking can be reduced to (1) was made in [18, 67, 68]. The derivation of inequalities from logic expressions is rather easy, as shown in [67, 68].

Indeed, let the conjunctive normal form of the specification be $\Phi_1 \wedge \Phi_2 \wedge \ldots \wedge \Phi_g$ with $\Phi_i \equiv \Psi_{i_1} \vee \Psi_{i_2} \vee \ldots \vee \Psi_{i_{h_i}}$, for $i = 1 \ldots g$. This can be expressed by

$$\sum_{k=1}^{h_i} \Psi_{i_k} \geq 1 \quad \text{for all } i = 1 \ldots g \tag{31}$$

Note that negation is algebraically represented as $\neg\Psi_{i_k} = 1 - \Psi_{i_k}$.

This approach can be applied to specifications described by logic constraints in the marking of a safe PN. The specifications can also include q, provided the concurrency setting ensures q is also a binary variable. As an example, assume the markings $[0,0,0]^T$, $[1,0,0]^T$, $[1,1,0]^T$ and $[1,1,1]^T$ are to be forbidden. Then, the specification can be expressed in the conjunctive normal form as $(\mu_1 \vee \mu_2 \vee \mu_3) \wedge (\neg\mu_1 \vee \mu_2 \vee \mu_3) \wedge (\neg\mu_1 \vee \neg\mu_2 \vee \mu_3) \wedge (\neg\mu_1 \vee \neg\mu_2 \vee \neg\mu_3)$, which can be simplified to $(\mu_2 \vee \mu_3) \wedge (\neg\mu_1 \vee \neg\mu_2)$. So, we obtain the constraints $\mu_2 + \mu_3 \geq 1$ and $-\mu_1 - \mu_2 \geq -1$.

7.4 Disjunctions of Constraints

Here we show that under certain boundedness assumptions, disjunctions of constraints can be expressed by conjunctions of constraints by adding not only places, but also transitions to the PN. A disjunction of constraints has the form:

$$\bigvee_i L_i\mu \leq b_i \tag{32}$$

where $L_i \in \mathbb{Z}^{m_i \times n}$ and $b_i \in \mathbb{Z}_i^m$. This can be written as

$$\bigwedge_j \bigvee_{i \in A_j} l_i \mu \leq c_i \tag{33}$$

where $l_i \in \mathbb{Z}^{1 \times n}$, $c_i \in \mathbb{Z}$ and A_j is a set of integers. The idea is to include additional binary variables δ_i for each constraint $l_i \mu \leq c_i$ such that:

$$[l_i \mu \leq c_i] \leftrightarrow [\delta_i = 1] \tag{34}$$

Then the disjunction (32) can be replaced by

$$\sum_{i \in A_j} \delta_i \geq 1 \tag{35}$$

for all indices j. If we know that $l_i \mu$ is between the bounds m_i and M_i, (34) becomes:

$$l_i \mu + (M_i - c_i)\delta_i \quad \leq \quad M_i \tag{36}$$

$$l_i \mu + (c_i + 1 - m_i)\delta_i \quad \geq \quad c_i + 1 \tag{37}$$

Note that this technique of adding auxiliary variables has been used to solve propositional logic via integer programming in [65, 66]. This technique has also been applied to Hybrid Systems in [4]. In our Petri net context, the variables δ_i will be interpreted as markings of additional "observer" places. This is the algorithm:

1. Let $T_i^+ = \{t : lD(\cdot, t) < 0\}$ and $T_i^- = \{t : lD(\cdot, t) > 0\}$.
2. Add an additional place d_i and copies t^+ for each transition $t \in T^+$ and copies t^- for each transition $t \in T^-$.
3. Add the arcs (d_i, t^-) and (t^+, d_i) with the weight 1. Note that (34) is satisfied by enforcing (36–37).

This construction is illustrated on the following example. Assume we desire to enforce

$$[\mu_2 \leq 0] \vee [\mu_4 \leq 0] \tag{38}$$

on the Petri net of Figure 11(a). Assume also the following bounds are known: $\mu_2 \leq 2$ and $\mu_4 \leq 3$. Note that (38) cannot be represented by conjunctions of inequalities that use only the variables μ_2 and μ_4 (Figure 10). For $\mu_2 \leq 2$, the relations (36–37) become (for $c_i = 0$, $m_i = 0$ and $M_i = 2$):

$$\mu_2 + 2\delta_1 \quad \leq \quad 2 \tag{39}$$

$$\mu_2 + \delta_1 \quad \geq \quad 1 \tag{40}$$

Similarly, for $\mu_4 \leq 3$ we have

$$\mu_4 + 3\delta_2 \quad \leq \quad 3 \tag{41}$$

$$\mu_4 + \delta_2 \quad \geq \quad 1 \tag{42}$$

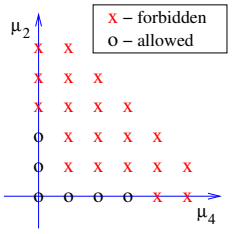

Figure 10:

The places d_1 and d_2 are shown in Figure 11(b). Figure 11(c) shows also the monitors a_1, e_1, a_2 and e_2, which correspond to (39–42), in this order. Finally, our disjunction (38) can be implemented by enforcing $d_1 + d_2 \geq 1$ (Figure 11(d)).

Unlike to the supervision based on place invariants, here not only places but also transitions are added to the net. However, it can be seen that the additional transitions are only used to represent disjunctions in the supervisory rule. For instance, in our example in Figure 11, (d) illustrates the closed-loop of a supervisor with the plant (a). The supervisor itself cannot be represented as a PN, since it involves disjunctions. The operation of the supervisor in terms of the plant (a) is represented in Figure 12, for the transitions t_2 and t_3. Similar operations are performed for t_4 and t_5. Note also that in the closed-loop Petri net the transitions t_2 and t_2^- can be distinguished for supervisory purposes, that is, they don't need to be labeled by the same event. Indeed, the supervisor distinguishes between the firings of t_2 and t_2^-, by checking whether a_1 is nonzero or not.

7.5 Liveness Enforcement

Here we consider an approach that designs liveness enforcing supervisors as supervisors enforcing constraints (1). This approach has appeared in [32, 38]. While the approach is very general, in that it makes no assumptions on the PN structure, it does not have guaranteed termination.

Given a PN $\mathcal{N}$ of initial marking μ_0, a transition t is live if for all reachable markings μ, there is an enabled firing sequence that includes t. Given $\mathcal{T} \subseteq T$, $(\mathcal{N}, \mu_0)$ is $\mathcal{T}$-live if all $t \in \mathcal{T}$ are live. Further, $(\mathcal{N}, \mu_0)$ is live if T-live (i.e., all transitions t are live).

Example 7.1 Note that the PN of Fig. 1(b) is not live, and not even deadlock-free: the sequence t_1, t_2, t_7 leads to deadlock. Here, the supervisor causes deadlock, as the plant in Fig. 1(a) is live. So we consider enhancing a specification $L\mu \leq b$ with additional constraints $L'\mu \leq b'$ such that the resulting supervised system is live. □

The procedure proposed in [32, 38] has the following input:

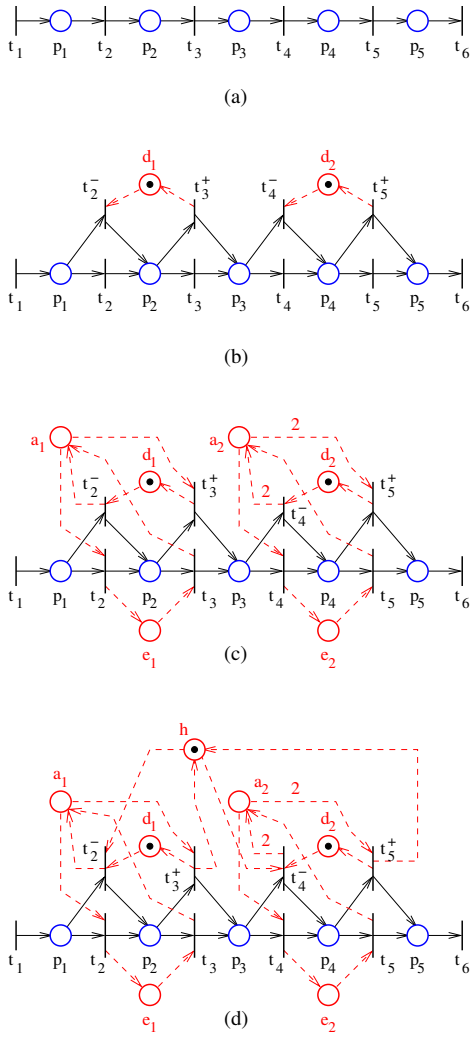

Figure 11:

1. A PN $\mathcal{N}$ and the set $\mathcal{T} \subseteq T$;

2. The sets of uncontrollable and unobservable transitions, T_{uc} and T_{uo};

3. Optionally, the set of reachable-marking constraints (RMC) $G\mu \leq h$.

Note that the RMC describe constraints that the reachable markings are known to satisfy. Formally, given a set of initial markings of interest $\mathcal{M}_I$, the RMC satisfy that $\forall \mu_0 \in \mathcal{M}_I \ \forall \mu \in \mathcal{R}(\mathcal{N}, \mu_0)$: $G\mu \leq h$, where $\mathcal{R}(\mathcal{N}, \mu_0)$ is the set of reachable markings of $(\mathcal{N}, \mu_0)$. The RMC is an optional argument, and its implicit value corresponds to $\mathbb{N}^m$ (all possible markings). The output of the procedure is the following:

1. Two sets of constraints $C\mu \leq d$ and $C_0\mu \leq d_0$, describing the supervisor.

enable t_2 in plant if $a_1 \vee (d_1 \wedge h)$

if t_2 fires in plant then
  if a_1 /* t_2 fires in closed-loop */
    $a_1 \rightarrow a_1 - 1$, $e_1 \rightarrow e_1 + 1$
  else /* t_2^- fires in closed-loop */
    $a_1 \rightarrow a_1 + 1$, $d_1 \rightarrow d_1 - 1$, and $h \rightarrow h - 1$
  end
end

enable t_3 in plant if $a_1 \vee e_1$

if t_3 fires in plant then
  if a_1 /* t_3^+ fires in closed-loop */
    $a_1 \rightarrow a_1 - 1$, $d_1 \rightarrow d_1 + 1$, and $h \rightarrow h + 1$
  else /* t_3 fires in closed-loop */
    $a_1 \rightarrow a_1 + 1$, $e_1 \rightarrow e_1 - 1$
  end
end

Figure 12:

2. A boolean variable LR, where $LR = TRUE$ indicates least-restrictive supervision.[3] (LR is set by checking sufficient conditions for least-restrictive supervision; in principle, the supervision could be least-restrictive also when $LR = FALSE$).

3. A boolean variable $TERM$, where $TERM = TRUE$ indicates successful termination.

The role of the constraints $C\mu \leq d$ and $C_0\mu \leq d_0$ is described in the following theorem from [32, 38].

Theorem 7.1 *If the procedure terminates and $TERM = TRUE$, then $C\mu \leq d$ is admissible and $(\mathcal{N}, \mu_0)$ supervised according to $C\mu \leq d$ is $\mathcal{T}$-live for all initial markings $\mu_0 \in \mathcal{M}_I$ satisfying $C_0\mu_0 \leq d_0$ and $C\mu_0 \leq d$.*

Note that $\mathcal{M}_I = \mathbb{N}^m$ when no RMC is given. On the other hand, when an RMC is given, the supervisor design may rely on it, and so $\mathcal{T}$-liveness enforcement is not guaranteed for $\mu_0 \notin \mathcal{M}_I$.

As Theorem 7.1 shows, the initial marking is a variable, not a given input, just as in the SBPI. In this context, this is what "least restrictive supervision" means. The supervisor defined by $C\mu \leq d$ and $C_0\mu \leq d_0$ is least restrictive if for all initial markings μ_0

- if $C\mu_0 \not\leq d$ or $C_0\mu_0 \not\leq d_0$, no $\mathcal{T}$-liveness enforcing supervisor of $(\mathcal{N}_0, \mu_0)$ exists.

- if $C\mu_0 \leq d$ and $C_0\mu_0 \leq d_0$, the supervisor enforcing $C\mu \leq d$ is the least restrictive $\mathcal{T}$-liveness enforcing supervisor of $(\mathcal{N}_0, \mu_0)$.

Note that if the procedure terminates and certain sufficient conditions are satisfied, the supervisor given by $C\mu \leq d$ and $C_0\mu \leq d_0$ is guaranteed to be least restrictive. In particular, when $\mathcal{T} = T$ (full liveness enforcement), $\mathcal{N}$ is fully controllable and observable ($T_{uc} = \emptyset$ and $T_{uo} = \emptyset$) and the procedure terminates,

[3]For the simplicity of the presentation, LR has not been included in the procedures of [32, 38]; however, it is implemented in the package [33].

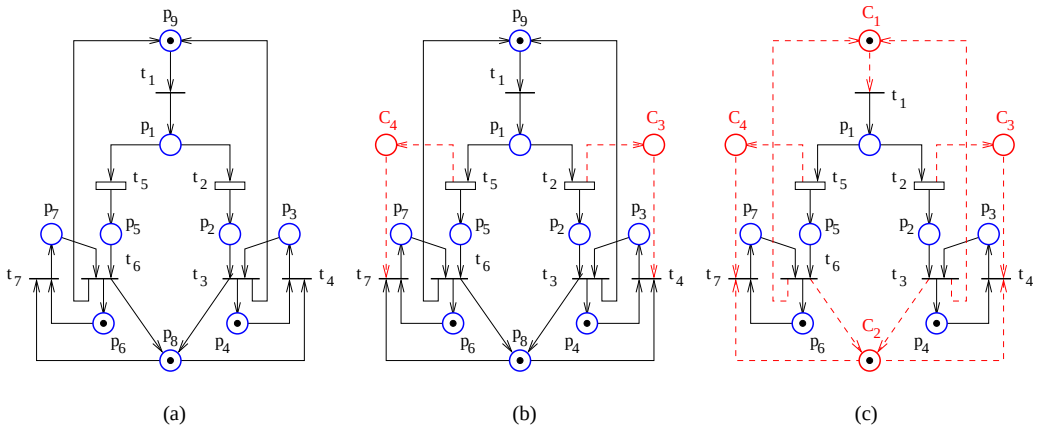

Figure 13:

the procedure generates the least restrictive liveness enforcement supervisor, if a liveness enforcing supervisor exists.

Example 7.2 As shown before, enforcing the specification (6–7) on the PN of Fig. 1(a) leads to deadlock. To add new constraints that ensure liveness, we start with the PN of Fig. 13(a), corresponding to the closed-loop of Fig. 1(b). Consider applying the $\mathcal{T}$-liveness enforcing procedure with $\mathcal{T} = T$ (full liveness desired), $T_{uo} = \emptyset$ and $T_{uc} = \{t_2, t_5\}$. Due to (8–9), the RMC are $\mu_1 + \mu_2 + \mu_5 + \mu_9 = 1$ and $\mu_3 + \mu_7 + \mu_8 = 1$. The procedure terminates with the following constraints $C\mu \leq d$:

$$\mu_1 + 2\mu_2 + \mu_5 + \mu_7 + \mu_8 + \mu_9 \quad \geq \quad 2 \tag{43}$$

$$\mu_1 + \mu_2 + \mu_3 + 2\mu_5 + \mu_8 + \mu_9 \quad \geq \quad 2 \tag{44}$$

and the following constraints $C_0\mu \leq d_0$

$$\mu_3 + \mu_4 \quad \geq \quad 1 \tag{45}$$

$$\mu_6 + \mu_7 \quad \geq \quad 1 \tag{46}$$

In view of the RMC, μ_8 and μ_9 can be substituted, and then (43) and (44) become

$$\mu_2 - \mu_3 \quad \geq \quad 0 \tag{47}$$

$$\mu_5 - \mu_7 \quad \geq \quad 0 \tag{48}$$

The supervised PN is shown in Fig. 13(b), while Fig. 13(c) shows the original plant supervised with (6–7) and the additional constraints (47–48) for liveness enforcement. □

7.6 Discussion

This section has shown that various types of specifications can be approached by structural methods and SBPI. Among language specifications, we have only considered specifications requiring the language of the closed-loop to be a sublanguage of the specification. Specifications requiring the closed-loop to generate a given PN language are more difficult. The existence of supervisors for this class of specifications is considered in [16].

Further, we have not considered the languages of labeled PNs with final states. For such PNs, a word is accepted only if it leads to a marking contained in the set of the final states. For such problems the supervision is to be nonblocking, that is, words leading to states from which the final states are unreachable should not be allowed. Thus, if $\mathcal{L}$ is the language describing the specification, the approach of section 7.2 can be used to ensure all sequences of plant events are in $\overline{\mathcal{L}}$. However, the approach of section 7.2 may allow the plant to deadlock after generating a word $w \in \overline{\mathcal{L}} \setminus \mathcal{L}$. Thus, a final state may never be reached. A topic of further research is to enhance the approach of section 7.2 to guarantee this situation cannot occur. This topic is related to [31, 30], dealing with specifications requiring target states to be reached and prespecified sequences to be fired.

Another type of languages considered in the literature deal with the infinite behavior of a plant. They express the requirement that there are no deadlocks and for all infinite words, some final state is infinitely often visited. Automata with this acceptance rule are called Büchi automata. It is known that specifications expressed in LTL (linear-time temporal logic) can be translated into Büchi automata [10]. This result is interesting, as it suggests temporal logic can be approached in our PN setting. Thus, given a Büchi automaton $\mathcal{A}$, we can first apply the approach of section 7.2, to generate a supervisor enforcing the part of the specification described by the structure of $\mathcal{A}$. Then, deadlock prevention or $\mathcal{T}$-liveness enforcement methods could be applied to guarantee some final states are infinitely often visited. The application of PN structural methods to temporal logic is an interesting topic of further research.

The application to temporal logic highlights the importance of a reliable tool for $\mathcal{T}$-liveness enforcement. As mentioned in section 7.5, the procedure of [32, 38] does not have guaranteed termination. In practice, the termination issue can be mitigated by using transformations to "EAC-nets" instead of "AC-nets" [32]. However, the total elimination of this issue is a matter of further research.

The procedure of section 7.5 is rather unique in that it offers a structural approach for $\mathcal{T}$-liveness enforcement (not just liveness enforcement), it makes no assumptions on the structure of the PN, and supports partial controllability and partial observability. Other approaches in the literature are on liveness (not $\mathcal{T}$-liveness) enforcement. Further, they typically make various assumptions on the structure of the PN and assume full controllability and observability. While valuable for the class of problems they were developed for, they may not be applicable to the closed-loop PNs resulting by enforcing specifications (1) or the more

general specifications discussed in this section. There are also some notable exceptions, from the area of reachability based methods. First, under boundedness assumptions and for a fixed initial marking, the $\mathcal{T}$-liveness enforcement problem is obviously solvable based on the reachability graph, which is finite in this case. However, even when these assumptions do not hold true, it is still possible to find a $\mathcal{T}$-liveness enforcing supervisor, as shown in [64]. The approach of [64] can be used for $\mathcal{T}$-liveness enforcement for arbitrary PNs but under full controllability and observability. The algorithm of [64] searches the marking space to find a set of minimal markings; based on this set the least restrictive $\mathcal{T}$-liveness enforcing supervisor can be immediately derived. While this approach has guaranteed termination, it has the following computational limitation: (a) the coverability graph is to be evaluated for every marking considered during the search; (b) the number of minimal markings as well as the size of a coverability graph may be large (e.g. exponential in the size of the net).

8 Applications

The constraints (1) have been proposed for various applications, such as in chemical processes [67], AGV coordination [42], manufacturing constraints [51], and mutual exclusion in batch processing [63]. Moreover, the class of constraints $L\mu + Hq \leq b$ has also been applied for the supervisory control of railway networks [19]. The constraints $Cv \leq b$ have also been used for fairness enforcement, such as bounding the difference between the number of occurrences of two events, in protocols [13] and manufacturing [48].

In this section we mention some other areas of application for the constraints (1). We consider here:

- an application to semaphores in Operating Systems.
- an application to fault-tolerance.
- the relation to synchronic distances.

8.1 Semaphores

The application of supervisory control techniques in software engineering has been proposed in [47, 45]. There, the supervisor can be seen as a plug-in to other software modules, ensuring certain specifications are satisfied. The approach there is to use the unfolding[4] of PN models for supervisor design. Obviously, other approaches could be applied as well for the supervisor design of software modules. In this section we consider monitor-based supervisors, we show such supervisors can be implemented in software by means of *semaphores*, and we discuss some of the potential benefits of the supervisory control approach for automatic code generation.

[4]Unfolding is a partial order method that constructs a reduced reachability graph

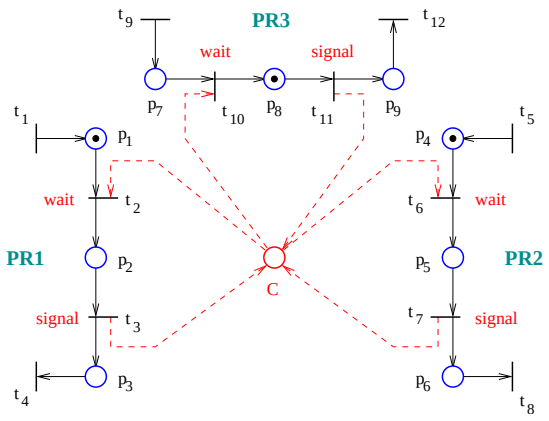

Figure 14:

Semaphores, monitors and rendez-vous mechanisms have been used in the context of Operating Systems for synchronization and control of access to shared resources. The PN modeling of these mechanisms has been considered in the literature [70]. In particular, the relation between PNs and semaphores has been known for a long time [40]. The observation that monitors correspond to semaphores is also known [26, 46].

Semaphores are nonnegative integer variables that can be accessed by means of two indivisible operations provided by the operating system: *wait* and *signal*. Given a semaphore x, when a process calls $wait(x)$, the operating system can act in two ways: (a) if $x \geq 1$, $x \rightarrow x - 1$; (b) if $x = 0$, the process calling $wait(x)$ is suspended. The calls $signal(x)$ result in two possible actions: (a) if there are processes suspended on $wait(x)$, one of them is selected to resume its execution; (b) otherwise, $x \rightarrow x + 1$.

Semaphores can easily be modeled by monitors, as illustrated in Figure 14. The figure shows three processes $PR1$, $PR2$, and $PR3$, that share a memory location. The process $PR1$ may access the memory when p_2 is marked, $PR2$ when p_5 is marked, and $PR3$ when p_8 is marked. To ensure the memory is not read and written at the same time by different processes, a semaphore is added, which is represented by the marking of the place C. Thus, the transitions $t \in C\bullet$ correspond to wait calls and the transitions $t \in \bullet C$ to signal calls. For the marking shown in the figure, $PR3$ is running, while $PR1$ and $PR2$ are suspended, as they cannot execute t_2 and t_6. However, after $PR3$ executes t_{11} (signal), one of $PR1$ or $PR2$ may resume its execution. Note also that the constraint enforced by the semaphore is $\mu_2 + \mu_5 + \mu_8 \leq 1$.

This example illustrates also that given a specification, such as a memory location may only be accessed by one process at a time, the semaphores implementing it may be automatically generated using the supervisory techniques of this paper. Some issues arise:

- Specifications (1) may result in more complex control structures, with more than one monitor connected to a single transition. This requires some simple extensions of the semaphore operations.

- Some other minor extensions to the semaphore operations are needed to implement specifications (29), which may produce self-loops.
- The extensions are somewhat more involved in the case of specifications represented by disjunctions (section 7.4).
- The usage of semaphores may lead to deadlocks. However, a liveness enforcing approach (section 7.5) could be used to automatically enhance the code with calls to additional semaphores such that no deadlock can occur.
- Semaphores have been typically used in a centralized setting. However, by means of the approach of section 6.5, a centralized specification can be decomposed for enforcement in a decentralized/distributed setting.

8.2 Fault Tolerance

Recent research on the robustness of the SBPI based designs to faults in a plant appears in [36]. There it is shown that the designs based on the SBPI and the related liveness enforcing approach of [38] have remarkable built-in qualities that simplify the fault accommodation process. In fact, only minor updates may be required for certain faults and reconfigurations. The kind of faults/reconfigurations considered in [36] are: faults modeled by token loss/gain, a class of changes in the form of the constraints, and changes in the controllability/observability of the system.

In what follows, we focus on a different approach to fault tolerance [24, 59, 61], in which additional places are added to a PN that allow detecting and correcting errors. Namely, we show that these places can be described by constraints $L\mu \leq b$ and $L\mu + Hq \leq b$.

An embedding of a PN $\mathcal{N} = (P, T, D^-, D^+)$ is a PN $\mathcal{N}_E = (P_E, T, D_E^-, D_E^+)$ such that $P \subseteq P_H$, and the input/output matrices are related by:

$$D_E^- = \begin{bmatrix} D^- \\ X_E^- \end{bmatrix} \quad D_E^+ = \begin{bmatrix} D^+ \\ X_E^+ \end{bmatrix}$$

As defined in [24], $\mathcal{N}_E$ is a **separate redundant embedding** if for every initial marking μ_0 of $\mathcal{N}$ and initial marking $\mu_{0,E} = G\mu_0$ of $\mathcal{N}_E$, all firing sequences enabled by μ_0 in $\mathcal{N}$ are also possible from $\mu_{0,E}$ in $\mathcal{N}_E$. The matrix G is required to have the form:

$$G = \begin{bmatrix} I_n \\ C \end{bmatrix}$$

for $n = |P|$. Note that for a separate redundant embedding, the places of the embedding are implicit, as their marking always enable a transition t if the marking of P enables t. Given a matrix X, let's write $X \geq 0$

if all elements of X are nonnegative; given the matrices X and Y, let's write $X \leq Y$ if $Y - X \geq 0$, and let's denote $\min(X, Y)$ the minimum taken element by element, that is, $\min(X, Y)$ denotes the matrix Z such that $Z_{i,j} = \min(X_{i,j}, Y_{i,j})$. Let's define also $\max(X, Y)$ in a similar way. The following result appears in [24]:

Theorem 8.1 [24] *$\mathcal{N}_E$ is a separate redundant embedding iff $C \geq 0$, $X_E^- = CD^- - A$ and $X_E^+ = CD^+ - A$, where $0 \leq A \leq \min(CD^-, CD^+)$.*

Here we show that constructing a redundant embedding is equivalent to designing the supervisor enforcing $L\mu + Hq \leq 0$ for $L \leq 0$ and $H \leq -LD^-$. From [39] we know that in the fully controllable and observable setting, the least restrictive supervisor enforcing $L\mu + Hq \leq 0$ corresponds to a PN of input and output matrices $X^- = \max(0, LD, H)$ and $X^+ = \max(0, -LD) + \max(0, H - \max(0, LD))$. Thus, the closed-loop is given by the input and output matrices:

$$D_C^- = \begin{bmatrix} D^- \\ X^- \end{bmatrix} \quad D_C^+ = \begin{bmatrix} D^+ \\ X^+ \end{bmatrix}$$

It turns out that we have the following result:

Theorem 8.2 *$\mathcal{N}_E$ is a separate redundant embedding iff there are $L\mu + Hq \leq 0$ with $L \leq 0$ and $H \leq -LD^-$ such that $X^- = X_E^-$ and $X^+ = X_E^+$.*

The result can be proven based on Theorem 8.1: if $\mathcal{N}_E$ is a separate redundant embedding, we can define $L = -C$ and $H = CD^- - A$, and then prove $X^- = X_E^-$ and $X^+ = X_E^+$. On the other hand, if $L\mu + Hq \leq 0$ with $L \leq 0$ and $H \leq -LD^-$, we can define $C = -L$ and $A = \min(-LD^-, -LD^+) - \max(0, H - \max(0, LD))$, and then prove $0 \leq A \leq \min(CD^-, CD^+)$.

In [24], two types of faults are considered. The first one, place failures, results in a change in the number of tokens. The second one, transitions failures, result in marking errors when the postcondition or the precondition of a transition t is not executed, that is, when we have either $\mu'_E = \mu_E - D_E^-(\cdot, t)$ or $\mu'_E = \mu_E + D_E^+(\cdot, t)$ instead of $\mu'_E = \mu_E + D_E^+(\cdot, t) - D_E^-(\cdot, t)$. As shown in [24], the detection and identification failures relies on C for place failures and on A for transition failures. Note that if we limit ourselves to place failures, we are free to chose any A such that $0 \leq A \leq \min(CD^-, CD^+)$. In particular, the choice $A = \min(CD^-, CD^+)$ corresponds to an embedding that do not adds self-loops to the Petri net. This corresponds to constraints with $L = -C$ and $H = CD^- - A$, that is, $H = \max(0, LD)$. However, the constraints $L\mu + Hq \leq b$ with $H = \max(0, LD)$ can be simply expressed (under the no concurrency assumption) as $L\mu \leq b$ (see chapter 3 of [32].) This shows that the constraints (1) can also be used in the context of fault detection and identification.

8.3 Synchronic Distances

An area of interest in the study of PNs is the Theory of Synchrony. Introductions to the field may be found in [13, 62]. The main issue here is the dependence between transition firings, such as, for instance, how many times can one transition t_1 be fired without firing another transition t_2. An important concept in this theory is the synchronic distance, defined below. We show here that specifications requiring bounds on synchronic distances can be implemented by specifications of the form $Cv \leq b$. This observation is important because, as mentioned also in section 7.1, enforcing constraints $Cv \leq b$ can be reduced to enforcing constraints (1).

Given a finite firing sequence σ of firing count vector $\underline{\sigma}$, let $\underline{\sigma}_i = \underline{\sigma}(t_i)$. Thus, $\underline{\sigma}_i$ denotes the number of occurrences of t_i in σ. Recall also that the Parikh vector v equals $\underline{\sigma}$ when σ is the sequence of firings since the initialization of the system. Given a PN with an initial marking and given two transitions t_1 and t_2, the *synchronic distance* can be defined by $\delta(t_1, t_2) = \sup_\sigma |\underline{\sigma}_1 - \underline{\sigma}_2|$, where the supremum is taken over all finite sequences σ enabled from some reachable marking. For some systems, a transition t_1 may fire twice as often as a transition t_2. Then, it would be natural to evaluate the difference $\sup_\sigma |\underline{\sigma}_1 - 2\underline{\sigma}_2|$. For this reason and in order to compare sets of transitions instead of just single transitions, the *synchronic distance* is defined with respect to weight vectors W_1 and W_2 as $\delta(W_1, W_2) = \sup_\sigma |W_1\underline{\sigma} - W_2\underline{\sigma}|$.

As shown on an example in [13], it may be useful to have specifications of the form $\delta(W_1, W_2) \leq d$. Note that $|W_1 v - W_2 v| \leq d/2 \Rightarrow \delta(W_1, W_2) \leq d$. Indeed, given σ enabled by μ, let σ^0 be the sequence by which μ was reached from the initial state, and let $\sigma^1 = \sigma^0\sigma$. We have $|W_1\underline{\sigma} - W_2\underline{\sigma}| \leq |W_1\underline{\sigma^0} - W_2\underline{\sigma^0}| + |W_1\underline{\sigma^1} - W_2\underline{\sigma^1}| \leq d$, where we have taken in account that for any firing sequence σ^0 from the initial marking, there is a reachable state v such that $v = \underline{\sigma^0}$. Thus, constraints $Cv \leq b$ can be used to describe synchronic distance specifications.

9 Conclusions

The problem of enforcing specifications $L\mu \leq b$ can be approached by numerous methods, as shown in this survey. We have emphasized a subset of structural methods, showing that they can be extended to very general plant models, including labeled Petri nets. Enforcing specifications $L\mu \leq b$ is of interest to numerous problems, as more general specifications can be reduced to the form $L\mu \leq b$ by transforming the plant model. Such general specifications that are treated in this paper include languages and disjunctions of linear inequalities.

Finally, it should be noted that some of the methods mentioned in this paper have been implemented in software. In particular, the SPNBOX [33] is a Matlab toolbox available on the web, that implements supervisor design approaches of [52, 39, 38] for SBPI design and liveness enforcement.

References

[1] F. Basile, P. Chiacchio, and A. Giua. On the choice of suboptimal monitor places for supervisory control of Petri nets. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, pages 752–757, 1998.

[2] F. Basile, P. Chiacchio, and A. Giua. Supervisory control of Petri nets based on suboptimal monitors places. In *Proceedings of the 4th International Workshop on Discrete Event Systems*, pages 85–87, 1998.

[3] F. Basile, P. Chiacchio, and A. Giua. Optimal control of Petri net monitors with control and observation costs. In *Proceedings of the 39'th IEEE International Conference on Decision and Control*, pages 424–429, 2000.

[4] A. Bemporad and M. Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407–427, 1999.

[5] R. K. Boel, L. Ben-Naoum, and V. Van Breusegem. On forbidden state problems for a class of controlled Petri nets. *IEEE Transactions on Automatic Control*, 40(1):1717–1731, 1995.

[6] H. Chen. Net structure and control logic synthesis of controlled Petri nets. *IEEE Transactions on Automatic Control*, 43(10):1446–1450, 1998.

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

[8] H. Chen and B. Hu. Monitor-based control of a class of controlled Petri nets. In *Proceedings of the 3rd International Conference on Automation, Robotics and Computer Vision*, 1994.

[9] A. Church. Logic, arithmetics, and automata. In *Proceedings of the International Congress of Mathematicians*, pages 23–35. Institut Mittag-Leffler, 1963.

[10] E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.

[11] P. Darondeau and X. Xie. Linear control of live marked graphs. *Automatica*, 39(3):429–440, 2003.

[12] A.A. Desrochers and R.Y. Al'Jaar. *Applications of Petri nets in Manufacturing Systems: Modelling, Control and Performance Analysis*. IEEE Press, 1995.

[13] H. J. Genrich, K. Lautenbach, and P. S. Thiagarajan. Elements of general net theory. In Brauer, W., editor, *Net Theory and Applications*, volume 84 of *Lecture Notes in Computer Science*, pages 21–163. Springer-Verlag, 1980.

[14] A. Ghaffari, N. Rezg, and X. Xie. Design of a live and maximally permissive petri net controller using the theory of regions. *IEEE Transactions on Robotics and Automation*, 19(1):137–142, 2003.

[15] A. Ghaffari, N. Rezg, and X. Xie. Feedback control logic for forbidden-state problems of marked graphs: application to a real manufacturing system. *IEEE Transactions on Automatic Control*, 48(1):2–17, 2003.

[16] A. Giua and F. DiCesare. Blocking and controllability of Petri nets in supervisory control. *IEEE Transactions on Automatic Control*, 39(4):818–823, 1994.

[17] A. Giua and F. DiCesare. Decidability and closure properties of weak Petri net languages in supervisory control. *IEEE Transactions on Automatic Control*, 40(5):906–910, 1995.

[18] 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.

[19] A. Giua and C. Seatzu. Supervisory control of railway networks with Petri nets. In *Proceedings of the 40'th IEEE Conference on Decision and Control*, pages 5004–5009, December 2001.

[20] A. Giua and C. Seatzu. Observability of place/transition nets. *IEEE Transactions on Automatic Control*, 47(9):1424–1437, 2002.

[21] A. Giua, C. Seatzu, and F. Basile. Observer-based state feedback control of timed Petri nets with deadlock recovery. *IEEE Transactions on Automatic Control*, 49(1):17–29, 2004.

[22] C. H. Golaszewski and P. J. Ramadge. Discrete event processes wtih arbitrary controls. In M. J. Denham and A. J. Laub, editors, *Advanced Computing Concepts and Techniques in Control Engineering*, pages 459–469. 1988.

[23] C. H. Golaszewski and P. J. Ramadge. Mutual exclusion problems for discrete event systems with shared events. In *Proceedings of the 27th IEEE Conference on Decision and Control*, pages 234–239, 1988.

[24] C. N. Hadjicostis and G. C. Verghese. Monitoring discrete event systems using Petri net embeddings. In *Application and Theory of Petri Nets 1999*, volume 1639 of *Lecture Notes in Computer Science*, pages 188–207. Springer-Verlag, 1999.

[25] L. E. Holloway and B. H. Krogh. Controlled Petri nets: A tutorial survey. In *11th International Conference on Analysis and Optimization of Systems: Discrete Event Systems*, volume 199 of *Lecture Notes in Control and Information Sciences*, pages 158–168. Springer-Verlag, 1994.

[26] L. E. Holloway, B. H. Krogh, and A. Giua. A survey of Petri net methods for controlled discrete event systems. *Discrete Event Dynamic Systems*, 7(2):151–190, 1997.

[27] L.E. Holloway and B.H. Krogh. Synthesis of feedback control logic for a class of controlled Petri nets. *IEEE Transactions on Automatic Control*, 35(5):514–523, 1990.

[28] L.E. Holloway and B.H. Krogh. On closed-loop liveness of discrete-event systems under maximally permissive control. *IEEE Transactions on Automatic Control*, 37(5):692–697, 1992.

[29] L.E. Holloway and B.H. Krogh. A generalization of state avoidance policies for controlled Petri nets. *IEEE Transactions on Automatic Control*, 41(6):804–816, 1996.

[30] A. Ichikawa and K. Hiraishi. Analysis and control of discrete event systems represented by Petri nets. In P. Varaiya and A. B. Kurzhanski, editors, *Discrete Event Systems: Models and Applications*, volume 103 of *Lecture Notes in Control and Information Sciences*, pages 115–134. Springer Verlag, 1988.

[31] A. Ichikawa, K. Yokoyama, and S. Kurogi. Reachability and control of discrete event systems represented by conflict-free Petri nets. In *Internationl Symposium on Circuits and Systems, Proceedings*, pages 487–490, 1985.

[32] M. V. Iordache. *Methods for the Supervisory Control of Concurrent Systems Based on Petri Net Abstractions*. PhD thesis, University of Notre Dame, 2003.

[33] M. V. Iordache and P. J. Antsaklis. Software tools for the supervisory control of Petri nets based on place invariants. Technical report isis-2002-003, University of Notre Dame, April 2002.

[34] M. V. Iordache and P. J. Antsaklis. Admissible decentralized control of Petri nets. In *Proceedings of the 2003 American Control Conference*, pages 332–337, 2003.

[35] M. V. Iordache and P. J. Antsaklis. Decentralized control of Petri nets with constraint transformations. In *Proceedings of the 2003 American Control Conference*, pages 314–319, 2003.

[36] M. V. Iordache and P. J. Antsaklis. Resilience to failures and reconfigurations in the supervision based on place invariants. In *Proceedings of the 2004 American Control Conference*, pages 4477–4482, 2004.

[37] M. V. Iordache, J. O. Moody, and P. J. Antsaklis. Synthesis of deadlock prevention supervisors using Petri nets. *IEEE Transactions on Robotics and Automation*, 18(1):59–68, February 2002.

[38] M.V. Iordache and P.J. Antsaklis. Design of T-liveness enforcing supervisors in Petri nets. *IEEE Transactions on Automatic Control*, 48(11):1962–1974, 2003.

[39] M.V. Iordache and P.J. Antsaklis. Synthesis of supervisors enforcing general linear vector constraints in Petri nets. *IEEE Transactions on Automatic Control*, 48(11):2036–2039, 2003.

[40] S. R. Kosaraju. Limitations of Dijkstra's semaphore primitives and Petri nets. *Operating Systems Review*, 7(4):122–126, 1973.

[41] B. Krogh. Controlled Petri nets and maximally permissive feedback logic. In *Proceedings of the 25th Annual Allerton Conference, University of Illinois, Urbana*, 1987.

[42] B.H. Krogh and L.E. Holloway. Synthesis of feedback control logic for manufacturing systems. *Automatica*, 27(4):641–651, 1991.

[43] R. Kumar and L. Holloway. Supervisory control of Petri nets with regular specification languages. *IEEE Transactions on Automatic Control*, 41(2):245–249, 1996.

[44] K. Lautenbach and P. S. Thiagarajan. Analysis of a resource allocation problem using Petri nets. In *Proceedings of the 1st European Conference on Parallel and Distributed Processing*, pages 260–266. Cepadues Editions, 1979.

[45] M. Lemmon and K. He. Supervisory plug-ins for distributed software. In Pezze, M. and Shatz, M., editors, *Proceedings of the Workshop on Software Engineering and Petri Nets*, pages 155–172. University of Aarhus, Department of Computer Science, 2000.

[46] M. Lemmon and K. He. Liveness enforcing monitors for safe and controllable petri nets. In *Proceedings of the 41st IEEE Conference on Decision and Control*, pages 775–780, 2002.

[47] M. Lemmon, K. He, and S. Shatz. Dynamic reconfiguration of software objects using Petri nets and network unfolding. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, pages 3069–3074, 2000.

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

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

[50] Y. Li and W. Wonham. Concurrent vector discrete-event systems. *IEEE Transactions on Automatic Control*, 40(4):628–638, 1995.

[51] J. O. Moody and P. J. Antsaklis. *Supervisory Control of Discrete Event Systems Using Petri Nets*. Kluwer Academic Publishers, 1998.

[52] 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.

[53] J. L. Peterson. *Petri Net Theory and the Modeling of Systems*. Englewood Cliffs, New Jersey: Prentice Hall, Inc., 1981.

[54] A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proceedings of the ACM Symposium on Principles of Programming Languages*, pages 179–190, 1989.

[55] P. Ramadge. Some tractable supervisory control problems for discrete-event systems modeled by Büchi automata. *IEEE Transactions on Automatic Control*, 34(1):10–19, 1989.

[56] P. Ramadge and W. Wonham. Modular feedback logic for discrete-event systems. *SIAM Journal on Control and Optimization*, 25(5):1202–1218, 1987.

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

[58] A. Schrijver. *Theory of linear and integer programming*. Wiley, 1986.

[59] J. Sifakis. Realization of fault-tolerant systems by coding Petri nets. *Journal of Design Automation and Fault-Tolerant Computing*, 3:93–107, April 1979.

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

[61] M. Silva Suarez. Error detection and correction on Petri net models of discrete events control systems. In *International Symposium on Circuits and Systems*, pages 921–924. IEEE, 1985.

[62] M. Silva Suarez. Towards a synchrony theory for P/T nets. In Voss, K., Genrich, H.J., and Rozenberg, G., editors, *Concurrency and Nets–Advances in Petri Nets*, pages 435–460. Springer-Verlag, 1987.

[63] M. Tittus and B. Egardt. Hierarchical supervisory control for batch processes. *IEEE Transactions on Control Systems Technology*, 7(5):542–554, 1999.

[64] R. Valk and M. Jantzen. The residue of vector sets with applications to decidability problems in Petri nets. *Acta Informatica*, 21:643–674, 1985.

[65] H. P. Williams. Linear and integer programming applied to the propositional calculus. *International Journal of Systems Research and Information Science*, 2:81–100, 1987.

[66] H. P. Williams. *Model building in mathematical programming.* Wiley, 1993. (3rd ed.).

[67] E. Yamalidou and J. Kantor. Modeling and optimal control of discrete-event chemical processes using Petri nets. *Computers and Chemical Engineering*, 15(7):503–519, 1991.

[68] 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.

[69] L. Zhang and L. E. Holloway. Forbidden state avoidance in controlled Petri nets under partial observation. In *Proceedings of the 33rd Annual Allerton Conference on Communications, Control, and Computing*, pages 146–155, 1995.

[70] W. M. Zuberek. Petri net models of process synchronization mechanisms. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, pages 841–847, 1999.