Supervisor Synthesis for Discrete Event Systems under Partial Observation and Arbitrary Forbidden State Specifications

Yu Ru, Maria Paola Cabasino, Alessandro Giua, Christoforos N. Hadjicostis *

Abstract

In this paper, we consider the forbidden state problem in discrete event systems modeled by partially observed and partially controlled Petri nets. Assuming that the reverse net of the uncontrollable subnet of the Petri net is structurally bounded, we compute a set of weakly forbidden markings from which forbidden markings can be reached by firing a sequence of uncontrollable/unobservable transitions. We then use reduced consistent markings to represent the set of consistent markings for Petri nets with structurally bounded unobservable subnets. We determine the control policy by checking if the firing of a certain controllable transition will lead to a subsequent reduced consistent marking that belongs to the set of weakly forbidden markings; if so, we disable the corresponding controllable transition. This approach is shown to be minimally restrictive in the sense that it only disables behavior that can potentially lead to a forbidden marking. The setting in this paper generalizes previous work by studying supervisory control for partially observed and partially controlled Petri nets with a general labeling function and a finite number of arbitrary forbidden states. In contrast, most previous work focuses on either labeling functions that assign a unique label to each observable transition or forbidden states that are represented using linear inequalities. More importantly, we demonstrate that, in general, the separation between observation and control (as considered in previous work) may not hold in our setting.

Published as:
Y. Ru, M.P. Cabasino, A. Giua, C.N. Hadjicostis, "Supervisor Synthesis for Discrete Event Systems under Partial Observation and Arbitrary Forbidden State Specifications," *Discrete Event Dynamic Systems*, 2012. DOI: 10.1007/s10626-012-0152-0. The original publication is available at www.springerlink.com.

*Y. Ru is with the Dept. of Mechanical and Aerospace Eng. Univ. of California, San Diego, CA, USA. E-mail:yuru2@ucsd.edu. M.P. Cabasino and A. Giua are with the Dept. of Electrical and Electronic Eng. Univ. of Cagliari, Piazza D'Armi, 09123 Cagliari, Italy. E-mail:{cabasino,giua}@diee.unica.it. C.N. Hadjicostis is with the Dept. of Electrical and Computer Eng. at the Univ. of Cyprus and at the Univ. of Illinois at Urbana-Champaign.E-mail:chadjic@ucy.ac.cy
This work was supported in part by the United States National Science Foundation (NSF), under NSF CNS Award 0834409. The research leading to these results has also received funding from the European Commission (EC) International Curriculum Option on Hybrid Control for Complex, Distributed and Heterogeneous Embedded Systems (http://www.piaggio.ccii.unipi.it/ICO/), and the European Commission Seventh Framework Programme (FP7/2007-2013) under grant agreements INFSO-ICT-223844 and PIRG02-GA-2007-224877. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of NSF or EC.

1 Introduction

1.1 Overview

A discrete event system (DES) is a dynamic system that evolves in accordance with the abrupt occurrence of events, at possibly unknown and irregular intervals [29]. Such systems arise in a variety of contexts, including manufacturing, robotics, vehicular traffic, and computer systems, as well as communication networks.

In many DESs, there may exist system states that are undesirable (e.g., a deadlock state from which there are no further state transitions, or a state that is reached through faulty state transitions). When certain activity in the system can be enabled/disabled, the problem of devising a control strategy via enabling/disabling transitions in order to avoid forbidden states is called the forbidden state problem. In this paper we study the forbidden state problem in which an arbitrary finite set of forbidden states needs to be avoided. The underlying system is modeled as a partially observed and partially controlled Petri net with a *general* labeling function, where transitions can be uncontrollable and/or unobservable and the net is not necessarily bounded. The problem is challenging in that (i) the set of forbidden states can be an arbitrary finite set that is not necessarily described by a set of linear constraints; (ii) possible system states following sensor observations are not necessarily unique, due to unobservable transitions and/or observable transitions that share the same label; and (iii) there can be interleaving of uncontrollable and unobservable transitions.

Given an arbitrary finite set of forbidden markings, we first compute (offline) the set of weakly forbidden markings (refer to Definition 5.1) which effectively allows us to deal with uncontrollable/unobservable transitions and to check the existence of a maximally permissive control policy (this is defined in Section 3). Then, we use reduced consistent markings (refer to Definition 4.2) to represent consistent markings (namely, the set of all possible system states at the current epoch), and the control policy is determined by checking whether a controllable transition will lead to a subsequent reduced consistent marking that belongs to the set of weakly forbidden markings.

1.2 Literature Review

The forbidden state problem was first introduced by Ramadge and Wonham in [28] in the context of finite automata. Later, this problem was also studied in the Petri net framework. When the set of forbidden states can be represented by linear inequalities (this is possible, for example, when the constraint relates to resource limitations in manufacturing systems), many applicable methods have been devised (e.g., [14, 37, 26, 6, 35, 13, 4, 21]). Though this assumption is not always viable, there is only a limited number of supervisor synthesis methods that address more general – possibly arbitrary – forbidden state specifications directly (e.g., [19, 33, 2, 31]). Most of these earlier approaches exhibit some limitations: for example, the plant is restricted to cyclic controlled marked graphs in [19] and to bounded Petri nets in [33, 2, 31] or the uncontrollable subnet is a backward conflict free choice net [4]. Another category of methods dealing with arbitrary forbidden state specifications (e.g., the work in [10, 9, 38, 25]) uses a two stage procedure, in which arbitrary forbidden state specifications are first transformed into linear inequality constraints and then handled using existing approaches for such linear constraints. The approaches in [10, 9] only deal with safe Petri nets (i.e., 1-bounded Petri nets; refer to Section 2.1 for its definition). Later on, non-safe Petri nets are considered in [38]; due to the necessity of enumerating states that are not forbidden (called authorized states in [38]), its applicability is still limited to bounded Petri nets. In [25], general procedures for transforming forbidden states into (a potentially large number of) linear constraints are proposed along with methods to simplify such linear constraints. In Example 5.6, it is shown that such simplification methods might not be effective for certain specifications.

Besides the arbitrary forbidden state specification, another challenge is that of partial observation. In the framework of Petri nets, the partial observation issue has been addressed in [39, 15, 1, 30]. In [15, 1, 30], the firing of each observable transition generates a unique label and the firings of unobservable transitions go unrecorded, whereas [39] considers the more general case where multiple observable transitions are allowed to share the same label.

The setting considered in [39] is similar to the setting in this paper with two key differences: i) a transition is control-enabled only when all control places that are connected to this transition have one token, and as a result, ii) a maximally permissive control policy as defined in [39] is nondeterministic in general. The approach followed in [39] uses a reduced state estimate set for control calculation (this set is equivalent to the set $\mathcal{E}$ of minimal cardinality as explained in Section 4); however, the characterization and update of the reduced state estimate set are not resolved. In addition, the supervisor is synthesized based on path predicates with a restrictive assumption[1] on the set of forbidden states. Furthermore, the computational complexity of the supervisor synthesis is not characterized. Finally, a survey of the main contributions appeared in the literature on the implementation of supervisory control techniques has been presented in [3].

The control policy proposed in this paper is based on a limited lookahead strategy. This strategy has been used in the context of finite state machines (e.g., the work in [7, 8, 24]). In [7, 8, 24], different lookahead strategies have been proposed to synthesize a supervisor which can ensure a set of legal behaviors (described as a language). In [18], a supervisor modification algorithm is proposed to modify an optimal full-observation supervisor to operate under partial observation, and the algorithm can be carried out on-line with linear computational complexity. However, all of the above mentioned approaches are based on the observation-control separate architecture (refer to Fig. 3(a)), which may not be applicable to the setting considered in this paper (this is explained in more detail in Section 3).

The contributions of this paper include the following: i) partially observed and partially controlled Petri nets with *general* labeling functions are used in supervisor synthesis; in contrast to previous work in [15, 1, 30], the separation between observation and control (in the sense

[1]More specifically, this assumption requires the so-called precedence path input condition; for details, refer to Definition 4 and Theorem 3 in [39].

of [1]) does not necessarily hold in this more general setting (refer to Section 3 for details); ii) reduced consistent markings are introduced not only to represent consistent markings but also to synthesize supervisors; iii) a detailed analysis of the computational complexity of the supervisor synthesis algorithm is provided, which is polynomial in the length of the observation and could be exponential in the Petri net size (such as the number of places and/or the number of transitions) in the worst case.

The paper is organized as follows. Section 2 presents basic definitions of Petri nets and introduces partially observed and partially controlled Petri nets. In Section 3, we formulate the supervisor synthesis problem with partial observation and an arbitrary finite set of forbidden states. In Section 4, we propose the concept of reduced consistent markings to represent the set of consistent markings given a sequence of observed labels and a sequence of control values. The supervisor synthesis method is presented in Section 5 and is demonstrated with a simple subway track system in Section 6. Conclusions are provided in Section 7.

2 Preliminaries

2.1 Basic Concepts of Petri Nets

In this subsection we recall notation and basic concepts about Petri nets. For more details, refer to [27].

Definition 2.1 *A Petri net structure is a 4-tuple $N = (P, T, F, W)$ where $P = \{p_1, p_2, ..., p_n\}$ is a finite set of n places; $T = \{t_1, t_2, ..., t_m\}$ is a finite set of m transitions; $F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs; $W : F \to \{1, 2, 3, ...\}$ is a weight function; $P \cap T = \emptyset$ and $P \cup T \neq \emptyset$.*

A *marking* is a function $M : P \to \mathbb{N}$ that assigns to each place a nonnegative integer number of tokens, where $\mathbb{N}$ denotes the set of nonnegative integers; $M(p)$ denotes the number of tokens in place p. Pictorially, places are represented by circles, transitions by bars and tokens by black dots, as shown in Fig. 1(a). A *Petri net* $G = \langle N, M_0\rangle$ is a Petri net structure N with an initial marking M_0.

The set of all input (or output) places of a transition $t \in T$ is defined as[2] ${}^\bullet t = \{p \in P \mid W(p,t) > 0\}$ (or $t^\bullet = \{p \in P \mid W(t,p) > 0\}$). A transition t is said to be a source transition if ${}^\bullet t = \emptyset$. Similarly, the set of all input (or output) transitions of a place $p \in P$ is defined as ${}^\bullet p = \{t \in T \mid W(t,p) > 0\}$ (or $p^\bullet = \{t \in T \mid W(p,t) > 0\}$). For example, for the Petri net in Fig. 1(a), we have $p_1^\bullet = \{t_1\}$, ${}^\bullet p_4 = \{t_3, t_5\}$, $t_3^\bullet = \{p_4\}$, and ${}^\bullet t_5 = \{p_2\}$. A Petri net structure is *acyclic* if it has no directed circuits (e.g., the nets in Fig. 2).

A transition t is *state-enabled* at marking M if each input place $p \in {}^\bullet t$ is marked with at least $W(p,t)$ tokens; this is denoted by $M[t\rangle$. The firing of state-enabled transition t removes $W(p,t)$ tokens from each input place $p \in {}^\bullet t$ and adds $W(t,p')$ tokens to each output place $p' \in t^\bullet$,

[2]If $W(p,t)$ or $W(t,p)$ is not defined for a specific place p and transition t, it is taken to be 0.

resulting in a marking M'; this is denoted by $M[t\rangle M'$. In this paper, we assume that at most one transition can fire at any instant; in other words, no concurrency is allowed. A k-length *firing sequence* $S = t_{s_1}t_{s_2}\cdots t_{s_k}$, $t_{s_i} \in T$, is enabled at marking M if $M[t_{s_1}\rangle M_1[t_{s_2}\rangle M_2 \cdots [t_{s_k}\rangle M'$; this is denoted by $M[S\rangle M'$. Marking M' can also be written as

$$M' = M + D\sigma, \tag{1}$$

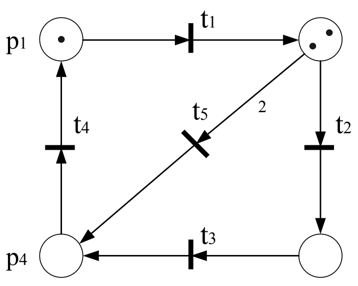
where (i) D is the $n \times m$ incidence matrix of N satisfying $D(i,j) = -W(p_i,t_j) + W(t_j,p_i)$, and (ii) σ is the $m \times 1$ *firing vector* of S with its ith entry being the number of times transition t_i appears in S.

A marking M is *reachable* in $\langle N, M_0\rangle$ if there exists a firing sequence S such that $M_0[S\rangle M$. The set of all markings reachable from M_0 defines the *reachability set* of $\langle N, M_0\rangle$ and is denoted by $R(N, M_0)$. A Petri net $\langle N, M_0\rangle$ is *K-bounded* (or simply, *bounded*) if there exists a positive constant integer K such that $\forall M \in R(N, M_0)$, $\forall p \in P$, $M(p) \leq K$. A Petri net is safe if it is *1-bounded*. A Petri net is *structurally bounded* if it is bounded for any initial marking. Equivalently, a Petri net is structurally bounded if and only if there exists an n-dimensional column vector y with strictly positive integer entries such that $y^T D \leq \mathbf{0}_m^T$, where y^T denotes the transpose of y, $\mathbf{0}_m$ denotes an m-dimensional column vector with all entries being 0, and the inequality is taken elementwise (see Theorem 29 in [27]). A structurally bounded Petri net is said to be *deadlock structurally bounded* if there exists an n-dimensional column vector y with strictly positive integer entries such that $y^T D < \mathbf{0}_m^T$ [32]. Acyclic Petri nets without source transitions have been shown to be deadlock structurally bounded [32].

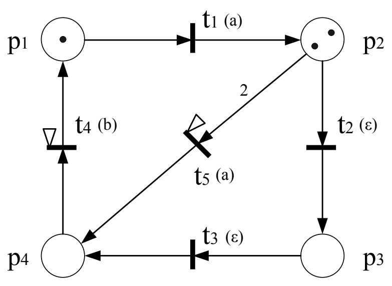
2.2 Partially Observed and Partially Controlled Petri Nets

In this paper, we consider Petri nets in which state transitions are partially observed and/or partially controlled.

Definition 2.2 *A partially observed and partially controlled Petri net Q is a 4-tuple* (N, T_o, T_c, M_0), *where*



(a) A simple Petri net.



(b) The net in (a) with observation labels and control inputs.

Figure 1: A partially observed and partially controlled Petri net.

- $N = (P, T, F, W)$ *is a Petri net structure with* n *places and* m *transitions;*
- $T_o \subseteq T$, *is the nonempty set of observable transitions;*
- $T_c \subseteq T$, *is the nonempty set of controllable transitions;*
- M_0 *is the initial state.*

Remark 2.3 *If* T_o *is empty, then there is no observation from the system; if* T_c *is empty, then there is no way to control the system. Therefore, we require both* T_o *and* T_c *to be nonempty.* ■

In a partially observed and partially controlled Petri net, the set of transitions T is partitioned in two distinct ways[3]:

- $T = T_o \uplus T_{uo}$: T_o (or T_{uo}) consists of all observable (or unobservable) transitions;
- $T = T_c \uplus T_{uc}$: T_c (or T_{uc}) consists of all controllable (or uncontrollable) transitions.

The partitioning of T into observable and unobservable transitions indicates what transitions can generate observation outputs. Unobservable transitions are transitions that cannot be directly observed given current sensor availability (no sensors exist for such transitions). On the other hand, an observable transition t can have a sensor to monitor its firing, but this sensor might be shared with other observable transitions. The association between sensors and transitions can be captured by a *labeling function* $L : T \to \Sigma \cup \{\varepsilon\}$, which assigns a label to each transition, and satisfies[4] $L(t) \in \Sigma$ for any $t \in T_o$ and $L(t) = \varepsilon$ for any $t \in T_{uo}$. Here, Σ is the set of labels and ε is the empty label. We denote the set of transitions that are associated with any label $e \in \Sigma$ as $T_e := \{t \in T \mid L(t) = e\}$. Without loss of generality, we assume that L is onto Σ so that $T_e \neq \emptyset$ for any $e \in \Sigma$. If $|T_{L(t)}| = 1$ (i.e., the label associated with transition t is unique to that transition) for any $t \in T_o$ and $L(t) = \varepsilon$ for any $t \in T_{uo}$, the mapping L is called a *natural projection*.[5] An example of a partially observed and partially controlled Petri net can be found in Fig. 1(b), and is discussed in detail in Example 2.10.

Given a firing sequence $S = t_{s_1} t_{s_2} \cdots t_{s_k}$, the corresponding *observation sequence* is

$$\omega = L(S) := L(t_{s_1}) L(t_{s_2}) \cdots L(t_{s_k}),$$

i.e., a string in Σ^* (the set of all possible strings generated from the alphabet Σ). Note that the empty label ε does not appear in a nonempty observation sequence and, therefore, the occurrence of unobservable transitions in an execution of a Petri net goes unrecorded. Therefore, due to the possible presence of unobservable transitions in the firing sequence $S = t_{s_1} t_{s_2} \cdots t_{s_k}$, the observation sequence $L(S)$ could have any length between 0 and k.

Now we introduce the definition of the set of consistent markings [5].

[3]For sets A, B, and C, $A = B \uplus C$ means $A = B \cup C$ and $B \cap C = \emptyset$.

[4]If there is no sensor for an observable transition t, one could enlarge T_{uo} as $T_{uo} \cup \{t\}$ to satisfy the requirements in the definition of a labeling function.

[5]The term "natural projection" is widely used in the context of automata, refer to [36] for details.

Definition 2.4 *Given a partially observed and partially controlled Petri net Q with labeling function L and an observed sequence of labels ω, the set of consistent markings is*

$$\mathcal{C}(\omega) = \{M \in \mathbb{N}^n \mid \exists S \in T^* : M_0[S\rangle M \text{ and } L(S) = \omega\} \ .$$

An observed sequence of labels $\omega = e_1 e_2 \cdots e_k$ is equivalent to a sequence of subsets of observable transitions $T_{e_1} T_{e_2} \cdots T_{e_k}$ with the interpretation that only one transition in T_{e_i} should occur at time epoch i and any sequence of unobservable transitions might occur between two consecutive time epochs, before time epoch 1, or after time epoch k. More generally, we define markings consistent with a sequence of subsets of observable transitions $\phi = T_1 \cdots T_i \cdots T_k$ as follows, where $T_i \subseteq T_o$ for $i = 1, ..., k$.

Definition 2.5 *Given a partially observed and partially controlled Petri net Q and a sequence of subsets of observable transitions $\phi = T_1 \cdots T_i \cdots T_k$, where $T_i \subseteq T_o$ for $i = 1, ..., k$, the set of consistent markings is*

$$\begin{aligned}\mathcal{C}(\phi) = \{&M \in \mathbb{N}^n \mid \exists S = S_{uo}^1 t_{s_1} \cdots S_{uo}^i t_{s_i} \cdots S_{uo}^k t_{s_k} S_{uo}^{k+1} : \\ &M_0[S\rangle M, S_{uo}^i \in T_{uo}^* \text{ for } i = 1, ..., k+1, \text{ and } t_{s_i} \in T_i \\ &\text{for } i = 1, ..., k\} \ .\end{aligned}$$

With this definition, we have $\mathcal{C}(e_1 e_2 \cdots e_k) = \mathcal{C}(T_{e_1} T_{e_2} \cdots T_{e_k})$. The introduction of $\mathcal{C}(\phi)$ enables us to deal with the interleaving between observation and control in a cleaner way (for details, refer to Section 4).

The partitioning of T into controllable and uncontrollable transitions indicates what transitions can be influenced by an external supervisor. Uncontrollable transitions are transitions that cannot be disabled by a supervisor. For example, state transitions in chemical reactions are usually uncontrollable; similarly, actuator failures can also be modeled by uncontrollable transitions.

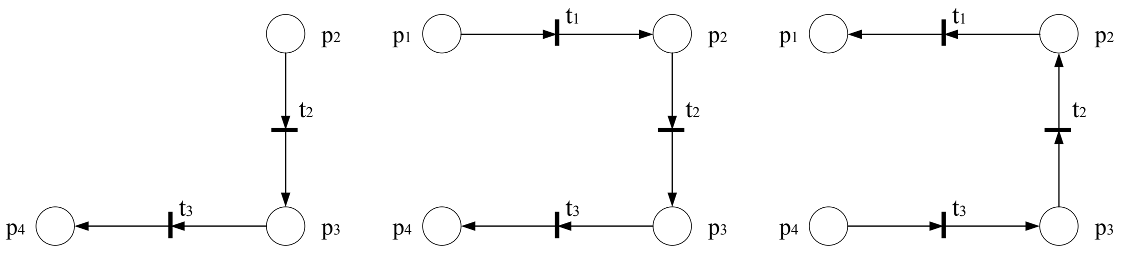
A controllable transition can be disabled by an external supervisor even if it is state-enabled. We take the set of possible control actions to be all subsets of T_c. More formally, we define the *control set* as

$$U = \{u \mid u \subseteq T_c\},$$

where u is called a *control value*. A controllable transition t is said to be control-disabled (or control-enabled) if $t \in u$ (or $t \notin u$). If a transition t is state-enabled and is not control-disabled, it is enabled and can fire following the state equation (1). By definition, uncontrollable transitions are always control-enabled.

To handle unobservable transitions, we need the concept of the unobservable subnet.

Definition 2.6 *Given a partially observed and partially controlled Petri net Q, we define the unobservable subnet as a net $N_{T_{uo}} = (P_{uo}, T_{uo}, F_{uo}, W_{uo})$, where $P_{uo} = \{p \in P \mid \exists t \in T_{uo} : p \in {}^\bullet t \cup t^\bullet\}$, F_{uo} is the restriction of F to $(P_{uo} \times T_{uo}) \cup (T_{uo} \times P_{uo})$, and W_{uo} is the restriction of W to F_{uo}.*



(a) Unobservable subnet of the net in Fig. 1(b).

(b) Uncontrollable subnet of the net in Fig. 1(b).

(c) Reverse net of the uncontrollable subnet of the net in Fig. 1(b).

Figure 2: Illustration of the unobservable subnet, the uncontrollable subnet, and the reverse net of the uncontrollable subnet of the Petri net in Fig. 1(b).

Remark 2.7 *In the definition of unobservable subnets, we use P_{uo} (a subset of P) instead of P due to the necessity of applying Lemma 7.1 (which is stated in the Appendix) to the analysis of computational complexity.* ■

To handle uncontrollable transitions, we need the concepts of the uncontrollable subnet and the reverse net [12].

Definition 2.8 *Given a partially observed and partially controlled Petri net Q, we define the uncontrollable subnet as a net $N_{T_{uc}} = (P, T_{uc}, F_{uc}, W_{uc})$, where F_{uc} is the restriction of F to $(P \times T_{uc}) \cup (T_{uc} \times P)$, and W_{uc} is the restriction of W to F_{uc}.*

Definition 2.9 *Given a Petri net structure $N = (P, T, F, W)$, $N' = (P, T, F', W')$ is said to be its reverse net if $F' = \{(x, y) | (y, x) \in F\}$, and $W'(x, y) = W(y, x)$ for any $(x, y) \in F'$.*

Note that if the incidence matrix for N is D, then the incidence matrix for its reverse net N' is $-D$.

Example 2.10 *Consider the partially observed and partially controlled Petri net Q in Fig. 1(b). In this Petri net, the labeling function L is given as $L(t_1) = L(t_5) = a$, $L(t_4) = b$, and $L(t_2) = L(t_3) = \varepsilon$, which implies that $T_o = \{t_1, t_4, t_5\}$ and $T_{uo} = \{t_2, t_3\}$. Furthermore, we can only disable transitions t_4 and t_5, which implies that $T_c = \{t_4, t_5\}$ and $T_{uc} = \{t_1, t_2, t_3\}$. Note that controllable transitions are indicated by attached empty triangles in the figure. The unobservable subnet of the net in Fig. 1(b) is shown in Fig. 2(a), its uncontrollable subnet in Fig. 2(b), and the reverse net of its uncontrollable subnet in Fig. 2(c).* ■

3 Problem Formulation

In the forbidden state problem we consider, the system is modeled by a partially observed and partially controlled Petri net Q with labeling function L, and the goal is to synthesize a control policy such that the system is guaranteed to avoid entrance to any state within a given finite set of forbidden states denoted by M_F. In most of previous work (e.g., [14, 37, 26, 6, 35, 13, 21]), the set of states that are not forbidden (called admissible states) is represented by a set of linear constraints, called generalized mutual exclusion constraints (GMEC). A GMEC is a linear constraint that limits the weighted sum of tokens in a subset of places of a given Petri net. Each GMEC is a couple (w, k), where $w : P \to Z$ is an $n \times 1$ weight vector and $k \in Z$, and defines admissible states as $\mathcal{M}(w, k) = \{M \in N^n \mid w^T \cdot M \leq k\}$. We discuss such constraints in more detail in Example 5.6.

To solve the forbidden state problem, we assume that the Petri net Q satisfies the following assumptions:

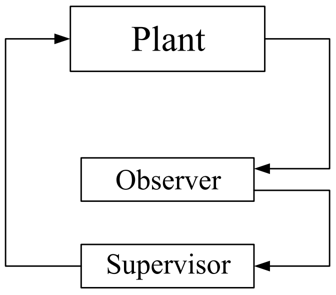
- **A1** $T_{uo} \subseteq T_{uc}$ (or equivalently, $T_c \subseteq T_o$);
- **A2** the unobservable subnet is structurally bounded;
- **A3** the reverse net of the uncontrollable subnet is structurally bounded;
- **A4** the set of forbidden markings M_F has finite cardinality.

The assumption that $T_{uo} \subseteq T_{uc}$ (or equivalently $T_c \subseteq T_o$) is quite common (e.g., the work in [39, 1, 30], and implicitly in [15]) because even if one disables a controllable but unobservable transition t, it is possible that the transition t might have occurred before the control action. As we will see later, Assumption **A2** guarantees that the set of consistent markings can be tracked (with a finite representation), while Assumptions **A3** and **A4** ensure that the existence of a control policy can be checked.

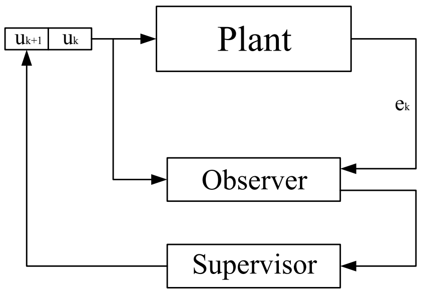
The trivial control policy in which all controllable transitions are always disabled can guarantee the avoidance of any forbidden state as long as this is possible (refer to Section 5.1, which discusses the conditions for the existence of a supervisory control policy); however, this policy might be too restrictive. To impose an optimality criterion, we introduce the concept of a maximally permissive control policy. At time epoch k, and assuming we have observed the sequence of labels $\omega = e_1 e_2 ... e_k$ and have applied the sequence of control values $\pi = u_1 u_2 ... u_k$, we define a *control policy* as a function $f(\omega, \pi) \subseteq T_c$. Note that the control sequence $u_1 u_2 ... u_k$ has the same length as the sequence of labels $e_1 e_2 ... e_k$, where u_i could be $\emptyset$ for $1 \leq i \leq k$. A control policy f is *maximally permissive* if the cardinality of $f(\omega, \pi)$ is minimal for any ω and any π. Therefore, our goal at any given time epoch k is to disable the minimal number of controllable transitions. Note that a similar optimality criterion has been used in [39, 34].

When the labeling function is the natural projection, the observer can be designed solely based on the output of the plant and the observer's output is used in supervisor synthesis, as shown in[6] Fig. 3(a); this is referred to as the separation between observation and control in [1]. This supervisory control structure is valid because any previously disabled controllable transition cannot appear as the current observed transition. However, in the partially observed and partially controlled Petri net (with a general labeling function) we consider here, the observer must be

[6]Fig. 3(a) is essentially Fig. 1 in [1], which is redrawn here for comparison.



(a) Observation-control separate architecture used in [15, 1, 30].



(b) Observation-control nonseparate architecture used in this paper.

Figure 3: Supervisory control architectures.

designed based on not only the current output of the plant but also based on the previous control action, as shown in Fig. 3(b). For example, if e_k is the current observed label and u_k is the previous control value, then $T_{e_k} \setminus u_k$ captures the set of observable transitions that might have generated the label e_k. Since $u_k \cap T_{e_k}$ is not necessarily empty in general, we need to consider both the sequence of observations and the sequence of control actions (up to the current epoch) in order to obtain (a more accurate estimate of the set of possible states and then use it to obtain) a maximally permissive policy.

Example 3.1 *Consider again the Petri net in Fig. 1(b). Suppose the control action u_1 is $\{t_5\}$ at the initial state $M_0 = (1\ 2\ 0\ 0)^T$, i.e., t_5 is control-disabled before we observe the next label. If we next observe the label a, then the only possible explanation is the firing of transition t_1 (even though t_5 is also associated with the label a) because of the previous control action u which disabled t_5. Therefore, when computing all possible current states, previous control actions should be taken into account (in addition to the current observed label). In turn, calculation of the next control action will depend on the observed labels and previous control actions (because the determination of the next control action depends on the set of all possible current states if state feedback based policies are considered). This shows that observation and control are not necessarily separable in our setting. In contrast, if each of the transitions t_1 and t_5 is assigned a unique label, then we will only observe the label corresponding to t_1 after the control action $\{t_5\}$. In this case, observation and control can be separated.* ■

4 Interleaving Between Observation and Control

To determine the control value at the current epoch given a sequence of observed labels and a sequence of previous control values, we need to first obtain an accurate estimate of possible current states, ideally developing a compact and easily computable representation of such state estimates.

Assume for now that there exists a maximally permissive control policy (for details regarding how to check the existence of such a policy, refer to the discussion in Section 5.1). The calculation of a control action is triggered by an observed label $e \in \Sigma$, except at the initial state M_0. At the very beginning when there is no observation (the observation sequence is $\omega = \varepsilon$), the system could be in any state within the set

$$\mathcal{D}_0 = \{M \in \mathbb{N}^n \mid \exists S \in T_{uo}^* : M_0[S\rangle M\}$$

due to the possible occurrence of an unobservable firing sequence. To determine if a controllable transition $t \in T_c$ should be disabled, we need to check if there exists a marking in the set

$$\mathcal{D}_{0t} = \{M' \in \mathbb{N}^n \mid \exists M \in \mathcal{D}_0, \exists S \in T_{uo}^* : M[tS\rangle M'\}$$

that is a weakly forbidden marking (i.e., a marking which can lead to forbidden markings via the firing of uncontrollable transition sequences; for the precise definition of weakly forbidden markings, refer to Definition 5.1). $\mathcal{D}_{0t}$ can also be represented as

$$\mathcal{D}_{0t} = \{M \in \mathbb{N}^n \mid \exists S_{uo}^1, S_{uo}^2 \in T_{uo}^* : M_0[S_{uo}^1 t S_{uo}^2\rangle M\}.$$

Based on $\mathcal{D}_{0t}$, we can determine the control action u_1. Assume that u_1 has been applied and next we observe the label e_1 (now the observation sequence is $\omega = e_1$ and the control sequence is $\pi = u_1$). Before observing e_1, only an unobservable firing sequence $S_{uo}^1 \in T_{uo}^*$ can fire. Because of Assumption **A1**, S_{uo}^1 is also an uncontrollable firing sequence and the control action u_1 has no effect on S_{uo}^1. Therefore, the control action u_1 can only possibly affect the controllable transitions that might have generated the observed label e_1. In other words, since a transition t satisfying $L(t) = e_1$ might be control disabled (i.e., $t \in u_1$), only the firing of transitions in $T_{e_1} \setminus u_1$ can possibly generate label e_1. Now the system could be in any state in the set of markings

$$\begin{aligned}\mathcal{D}_1 =& \{M' \in \mathbb{N}^n \mid \exists M \in \mathcal{D}_0, \exists t \in T_{e_1} \setminus u_1, \exists S_{uo}^2 \in T_{uo}^* : \\ & M[tS_{uo}^2\rangle M'\} = \{M' \in \mathbb{N}^n \mid \exists S_{uo}^1, S_{uo}^2 \in T_{uo}^*, \\ & \exists t \in T_{e_1} \setminus u_1 : M_0[S_{uo}^1 t S_{uo}^2\rangle M'\}\ .\end{aligned}$$

This procedure can be repeated by first calculating the control action u_2 based on $\mathcal{D}_1$ (or, equivalently, based on e_1 and u_1, as shown in Fig. 3(b)), and by then updating the set of all possible states upon the subsequently observed label e_2. In general, we could define the set of markings that are consistent with the sequence of observed labels ω and the sequence of control values π as described below.

Definition 4.1 *Given a partially observed and partially controlled Petri net Q with labeling function L, a sequence of observed labels $\omega = e_1 e_2 \cdots e_k$ (or, equivalently, a sequence of subsets of observable transitions $\phi = T_{e_1} T_{e_2} \cdots T_{e_k}$) and a sequence of control values $\pi = u_1 u_2 \cdots u_k$, $\mathcal{C}(\omega, \pi)$ (or equivalently, $\mathcal{C}(\phi, \pi)$), the set of markings consistent with ω (or equivalently, ϕ)* and *π, is defined as*

$$\begin{aligned}\mathcal{C}(\omega, \pi) = \{M \in \mathbb{N}^n \mid & \exists S = S_{uo}^1 t_o^1 \cdots S_{uo}^i t_o^i \cdots S_{uo}^k t_o^k S_{uo}^{k+1} \\ & \textit{such that } S_{uo}^i \in T_{uo}^* \textit{ for } i = 1, ..., k+1,\ t_o^i \in \\ & T_{e_i} \setminus u_i \textit{ for } i = 1, ..., k,\ \textit{and } M_0[S\rangle M\}\ .\end{aligned}$$

If ω and π are both empty, then

$$\mathcal{C}(\varepsilon, \lambda) = \{M \in \mathbb{N}^n \mid \exists S_{uo}^1 \in T_{uo}^* : M_0[S_{uo}^1\rangle M\} ,$$

where λ denotes an empty sequence of control values π.

It can be verified that $\mathcal{D}_0 = \mathcal{C}(\varepsilon, \lambda) = \mathcal{C}(\nu, \lambda)$, $\mathcal{D}_1 = \mathcal{C}(e_1, u_1)$, and $\mathcal{D}_{0t} = \mathcal{C}(\{t\}, \emptyset)$, where ν denotes an empty sequence of subsets of observable transitions ϕ. The calculation of $\mathcal{C}(\omega, \pi)$ (or $\mathcal{C}(\phi, \pi)$) is important not only for updating the set of consistent markings but also for calculating the control actions.

Now we establish that $\mathcal{C}(\omega, \pi)$ is identical to $\mathcal{C}(\phi')$ (refer to Definition 2.5) for a sequence of subsets of observable transitions ϕ' which is determined by ω and π. By definition, $\mathcal{C}(\varepsilon, \lambda) = \mathcal{C}(\nu)$. After applying u_1 and observing e_1, only the firing of transition $t \in T_{e_1} \setminus u_1$ can possibly generate label e_1 as argued previously. We can use the sequence of observed labels $\omega = e_1 e_2 ... e_k$ and the sequence of control values $\pi = u_1 u_2 ... u_k$, and apply the above argument repeatedly k times to construct an equivalent new sequence of subsets of observable transitions $\phi' = T_1 T_2 \cdots T_k$ where $T_i = T_{e_i} \setminus u_i$ for $i = 1, 2, ..., k$. Therefore, $\mathcal{C}(\omega, \pi)$ is equal to $\mathcal{C}(\phi')$. This way, control actions can only potentially affect the observed labels, and the construction of the new sequence of subsets of observable transitions guarantees that all control-disabled transitions are excluded in the calculation of consistent markings.

One simple approach for calculating $\mathcal{C}(\phi')$ is to enumerate all these markings. Since $\mathcal{C}(\phi') \subseteq \mathcal{C}(\omega)$ and (under Assumption **A2** given in Section 3) $|\mathcal{C}(\omega)|$ increases at most polynomially in the length of the observation sequence ω [32], $\mathcal{C}(\phi')$ can also be calculated with complexity that is polynomial in the length $|\phi'|$ of the sequence ϕ' (note that $|\phi'|$ is equal to $|\omega|$, i.e., the length of ω). However, only a subset $\mathcal{E} \subseteq \mathcal{C}(\phi')$ is necessary to represent $\mathcal{C}(\phi')$ as

$$\{M' \in \mathbb{N}^n \mid \exists M \in \mathcal{E}, \exists S \in T_{uo}^* : M[S\rangle M'\} \tag{2}$$

(in other words, any marking M' in $\mathcal{C}(\phi')$ can be reached from some marking M in $\mathcal{E}$ by a firing sequence of unobservable transitions), and the representation of $\mathcal{C}(\phi')$ using $\mathcal{E}$ is sufficient for determining the control action based on the idea of weakly forbidden markings (to be introduced in Definition 5.1). It can be shown that there could be many sets that can serve as $\mathcal{E}$. Ideally, one would like to get a set $\mathcal{E}$ of minimal cardinality and/or with minimal computational effort.

Under the assumption that the unobservable subnet is acyclic, one candidate for $\mathcal{E}$ is the set of markings called[7] basis markings. Under certain assumptions (e.g., Petri nets that are marked graphs [1], or Petri nets whose unobservable subnets are acyclic and backward conflict-free [17]), the basis marking is unique. However, in general, there could be multiple basis markings, and the procedure for calculating basis markings in [5] could be involved. Instead, we propose another candidate for $\mathcal{E}$, called the set of reduced consistent markings.

[7]Given a partially observed and partially controlled Petri net Q with labeling function L, and a sequence ω of observed transition labels, a *basis marking* M_b in [5] is a marking reached from the initial marking M_0 by firing ω and all those unobservable transitions that are strictly necessary to enable ω. Note that in [5], the labeling function used is the natural projection. The definition can be extended to a partially observed and partially controlled Petri net Q with a sequence of subsets of observable transitions ϕ.

Definition 4.2 *Given a partially observed and partially controlled Petri net Q with a sequence of subsets of observable transitions $\phi = T_1 \cdots T_i \cdots T_k$, where $T_i \subseteq T_o$ for $i = 1, ..., k$, the set of reduced consistent markings is defined as*

$$\begin{aligned}\mathcal{C}_r(\phi) = \{M \in \mathbb{N}^n \mid \exists S = S^1_{uo} t_{s_1} \cdots S^i_{uo} t_{s_i} \cdots S^k_{uo} t_{s_k} :\\ M_0[S\rangle M, S^i_{uo} \in T^*_{uo} \text{ for } i = 1, ..., k, \text{ and } t_{s_i} \in T_i\\ \text{for } i = 1, ..., k\}\end{aligned}$$

if $\phi \neq \nu$, and $\mathcal{C}_r(\nu) = \{M_0\}$ otherwise.

In other words, reduced consistent markings are consistent markings that can be reached by a firing sequence whose last transition is observable. The set of consistent markings $\mathcal{C}(\phi)$ can be represented using reduced consistent markings as $\{M \in \mathbb{N}^n \mid \exists M' \in \mathcal{C}_r(\phi), \exists S \in T^*_{uo} : M'[S\rangle M\}$.

Remark 4.3 *The name "reduced consistent marking" is given based on the fact that a reduced consistent marking must be a consistent marking while a consistent marking is not necessarily a reduced consistent marking. In other words, the set of reduced consistent markings is a subset of the set of consistent markings for the same sequence of subsets of observable transitions.* ■

Now we consider the calculation of $\mathcal{C}_r(\phi T')$ given $\mathcal{C}_r(\phi)$ for an arbitrary sequence of subsets of observable transitions ϕ. First we recall the definition of the unobservable reach from a marking M [11]. Given a partially observed and partially controlled Petri net Q, $UR(M)$, the unobservable reach from a marking M, is $\{M' \in \mathbb{N}^n \mid \exists S \in T^*_{uo} : M[S\rangle M'\}$.

Algorithm 4.4 [Update of Reduced Consistent Markings]

Input: *A partially observed and partially controlled Petri net Q with $\mathcal{C}_r(\phi)$ (a set of reduced markings consistent with ϕ), and a new subset of observable transitions T'.*

Output: *$\mathcal{C}_r(\phi T')$.*

```
1. $\mathcal{C}(\phi) = \bigcup_{M \in \mathcal{C}_r(\phi)} UR(M)$.
2. Let $\mathcal{C}_r(\phi T') = \emptyset$.
3. For any $M \in \mathcal{C}(\phi)$
     For any $t$ such that $t \in T'$ and $M[t\rangle$
         Compute $M' = M + D(:, t)$.
         Set $\mathcal{C}_r(\phi T') = \mathcal{C}_r(\phi T') \cup \{M'\}$.
4. Output $\mathcal{C}_r(\phi T')$.
```

Remark 4.5 *Based on the definition of basis markings [5] (as recalled in Footnote 7), a basis marking is also a reduced consistent marking while a reduced consistent marking is not necessarily a basis marking, as shown in the following example.* ■

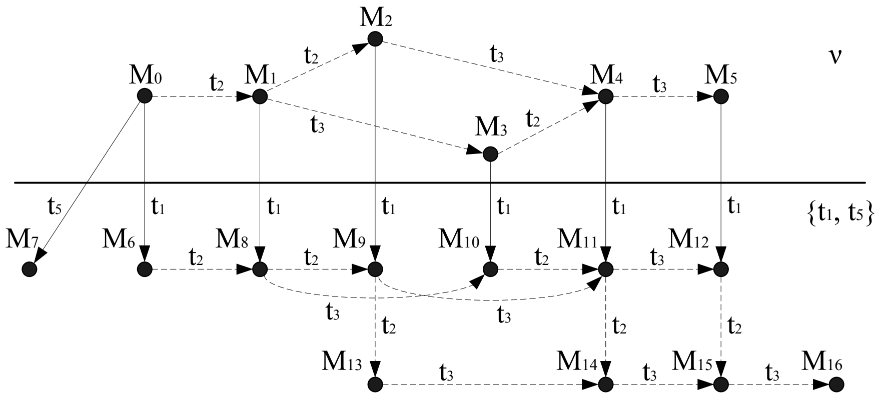


Figure 4: Markings that are consistent with $\phi = \nu$ and $\phi = T' = \{t_1, t_5\}$ for the net in Fig. 1(b); solid lines with arrows represent observable transitions while dashed lines with arrows represent unobservable transitions.

Example 4.6 *For the net in Fig. 1(b), we consider two sequences of subsets of observable transitions starting from M_0, namely, ν and $T' = \{t_1, t_5\}$. The various system trajectories are shown in Fig. 4. If $\phi = \nu$, the set of consistent markings is*

$$\mathcal{C}(\phi) = \{M_0, M_1, M_2, M_3, M_4, M_5\} ,$$

where $M_1 = (1\ 1\ 1\ 0)^T$, $M_2 = (1\ 0\ 2\ 0)^T$, $M_3 = (1\ 1\ 0\ 1)^T$, $M_4 = (1\ 0\ 1\ 1)^T$, and $M_5 = (1\ 0\ 0\ 2)^T$. However, M_0 is both the only basis marking and the only reduced consistent marking. If $\phi = T'$, there are 11 *consistent markings $M_6, M_7, ..., M_{16}$, among which $M_6, M_7, M_8, M_9, M_{10}, M_{11}, M_{12}$ are reduced consistent markings, and M_6, M_7 are basis markings. These markings are $M_6 = (0\ 3\ 0\ 0)^T$, $M_7 = (1\ 0\ 0\ 1)^T$, $M_8 = (0\ 2\ 1\ 0)^T$, $M_9 = (0\ 1\ 2\ 0)^T$, $M_{10} = (0\ 2\ 0\ 1)^T$, $M_{11} = (0\ 1\ 1\ 1)^T$, $M_{12} = (0\ 1\ 0\ 2)^T$, $M_{13} = (0\ 0\ 3\ 0)^T$, $M_{14} = (0\ 0\ 2\ 1)^T$, $M_{15} = (0\ 0\ 1\ 2)^T$, and $M_{16} = (0\ 0\ 0\ 3)^T$.* ■

5 Supervisor Synthesis

In the previous section, we used reduced markings that are consistent with the constructed sequence ϕ' to represent the set of all possible markings that are consistent with the observation sequence ω and the control sequence π. In this section, we consider how to determine a control value to avoid an arbitrary set of forbidden states using the set of reduced consistent markings.

5.1 Existence of Maximally Permissive Supervisor

Besides forbidden states, we also need to prevent the current state from reaching certain states that are not explicitly forbidden. These are states from which one can reach forbidden states by a firing sequence consisting of uncontrollable and/or unobservable transitions. Once a system enters such states, we cannot control the system in a way that guarantees that forbidden states are avoided. We call such markings weakly forbidden markings.

Definition 5.1 *Given a partially observed and partially controlled Petri net Q with a set of forbidden markings M_F, the set of weakly forbidden markings $W(M_F)$ with respect to M_F, is given by*

$$\{M \in \mathbb{N}^n \mid \exists M' \in M_F, \exists S \in (T_{uc} \cup T_{uo})^* : M[S\rangle M'\} \ ,$$

which reduces under Assumption ***A1*** *(namely, $T_{uo} \subseteq T_{uc}$) to*

$$W(M_F) = \{M \in \mathbb{N}^n \mid \exists M' \in M_F, \exists S \in T_{uc}^* : M[S\rangle M'\} \ .$$

This simplification leads to the notion of weakly forbidden markings used in [19]. The set of weakly forbidden markings $W(M_F)$ can be computed using the following proposition (c.f. [33]).

Proposition 5.2 *Given a partially observed and partially controlled Petri net Q with a set of forbidden markings M_F, the set of weakly forbidden markings is given by*

$$W(M_F) = \bigcup_{M \in M_F} R(N'_{T_{uc}}, M),$$

where $N'_{T_{uc}}$ is the reverse net of the uncontrollable subnet $N_{T_{uc}}$.

Following Proposition 5.2, we can obtain $W(M_F)$ by computing all markings reachable from M_F in the reverse net of the uncontrollable subnet. The computation of $W(M_F)$ may be complicated as $W(M_F)$ is not necessarily finite and its computation essentially involves reachability analysis (computational complexity issues are discussed in the Appendix). However, under Assumptions **A3** and **A4** made in Section 3, the set of weakly forbidden markings is finite, because the number of forbidden markings is finite and $N'_{T_{uc}}$ is structurally bounded. More specifically, because $N'_{T_{uc}}$ is structurally bounded, $|R(N'_{T_{uc}}, M)|$ is finite for any M in M_F; thus, $|W(M_F)|$ is also finite as the number of forbidden markings $|M_F|$ is finite.

The existence of a maximally permissive supervisor can be determined by checking whether $M_0 \notin W(M_F)$, which can be directly deduced from Theorem 2 in [20]. More specifically, if $M_0 \notin W(M_F)$, then the control policy that disables all controllable transitions can avoid all forbidden states, which implies that there exists a maximally permissive control policy. Note that this condition is also necessary. That is to say, if $M_0 \in W(M_F)$, there is no control policy which can *guarantee* that the system will not reach a forbidden state.

Example 5.3 *For the net in Fig. 1(b), suppose there is only one forbidden state $(1\ 0\ 0\ 1)^T$. By applying Proposition 5.2 based on the reverse net in Fig. 2(c), we obtain $W(M_F) = \{(1\ 0\ 0\ 1)^T, (1\ 0\ 1\ 0)^T, (1\ 1\ 0\ 0)^T, (2\ 0\ 0\ 0)^T\}$. As M_0 is not in $W(M_F)$, there exists a maximally permissive control policy.* ■

5.2 Determination of Control Policy

In this subsection, we determine the maximally permissive supervisor using reduced consistent markings.

The basic idea is the following. Given the sequence of observed labels ω seen so far and the sequence of control values π applied so far, we generate the set of reduced consistent markings using the constructed sequence ϕ'. We then compute the set of subsequent reduced consistent markings $\mathcal{C}_r(\phi'\{t\})$ for each controllable transition t, and we disable t if at least one of the subsequent reduced consistent markings is in the set of weakly forbidden markings. Once a new label is observed, we update the set of reduced consistent markings based on this label and the current control value, and then determine the next control action. Note that at the initial state M_0, both ω and π are empty, which implies that we first compute the control value u_1, then observe the label e_1, so that ω becomes e_1 and π becomes u_1. The procedure is then repeated as discussed above. The complete two-stage algorithm is given below.

Algorithm 5.4 [Supervisor Synthesis]

Input: *A partially observed and partially controlled Petri net Q with labeling function L satisfying Assumptions **A1**, **A2**, and **A3**, a finite set of forbidden markings M_F satisfying Assumption **A4**, and a streaming sequence of observed labels ω.*

Output: *A sequence of control values corresponding to ω.*

Stage 1: *Offline Checking of Supervisor Existence*

1) Compute the set of weakly forbidden markings $W(M_F)$ using the reverse net of the uncontrollable subnet.

2) Check if $M_0 \in W(M_F)$. If $M_0 \notin W(M_F)$, the supervisor exists; else, exit.

Stage 2: *Online Determination of Control Policy*

1) Let $\omega = \varepsilon$, $\phi' = \nu$ and $\mathcal{C}_r(\phi') = \{M_0\}$.

2) Let $u = \emptyset$.

For every $t \in T_c$, compute $\mathcal{C}_r(\phi'\{t\})$ based on $\mathcal{C}_r(\phi')$ using

Algorithm 4.4. If $\exists M \in \mathcal{C}_r(\phi'\{t\})$ such that $M \in W(M_F)$

then $u = u \cup \{t\}$.

3) Output the control value u at the current epoch.

4) Wait until a new label e is observed.

5) Define T' such that $T' = T_e \setminus u$ and compute $\mathcal{C}_r(\phi' T')$ based on $\mathcal{C}_r(\phi')$ using Algorithm 4.4; let $\omega = \omega e$ and $\phi' = \phi' T'$.

*6) Goto Step 2 of **Stage 2**.*

At Step 2 in Stage 2, we determine the control value at the current epoch by examining for each controllable transition whether it generates a subsequent reduced consistent marking that is weakly forbidden. We show the correctness of the algorithm by proving the following two facts:

1) $\forall t \in T \setminus u$, the firing of t will not drive the system from a legal state to any forbidden state. There are two possibilities: $t \in T_{uc}$ or $t \in T_c \setminus u$. i) if $t \in T_{uc}$, then the firing of t will not drive the system to any forbidden state due to a) the existence of a maximally permissive control if ω is ε, and b) the previous control action if $\omega \neq \varepsilon$. ii) $t \in T_c \setminus u$: suppose there exists a marking M consistent with $\phi'\{t\}$ (namely $M \in \mathcal{C}(\phi'\{t\})$) such that $M \in W(M_F)$; then, there exists a reduced consistent marking $M' \in \mathcal{C}_r(\phi'\{t\})$, from which M is reached by firing a sequence of unobservable transitions (it is also a sequence of uncontrollable transitions as $T_{uo} \subseteq T_{uc}$), such that M' is in $W(M_F)$ based on the definition of $W(M_F)$. Therefore, t should have been disabled according to Step 2 in Stage 2, which is a contradiction.

2) $\forall t \in u$ given at Step 3 in Stage 2, the firing of t can in fact result in a forbidden state because there exists a marking M, which is consistent with $\phi'\{t\}$ (actually it is a reduced consistent marking) and is in $W(M_F)$, i.e., it can uncontrollably reach a forbidden marking.

The above two facts also establish that the control policy is maximally permissive.

As argued in the Appendix, the online computational complexity is polynomial in k which is the length of the observed sequence of labels. This could be very helpful because given a partially observed and partially controlled Petri net only the observed sequence changes with time. On the other hand, the computational complexity is exponential in the number of places in the given Petri net. The exponential dependency could impose challenges in practical applications for Petri nets of large size. In Appendix D, we discuss these issues in more detail and also present specific scenarios in which our approach could potentially be applicable to large-scale systems. In addition, we also provide implementation details on Algorithm 5.4 in the Appendix.

Example 5.5 *Consider the setting in Example 5.3. As the maximally permissive control policy exists, we try in this example to determine the control policy based on observations and previous control actions. When* $\omega = \varepsilon$, $\phi' = \nu$ *and* $\mathcal{C}_r(\phi') = \{M_0\}$. *For* t_4, $\mathcal{C}_r(\phi'\{t_4\}) = \{(2\ 1\ 0\ 0)^T, (2\ 0\ 1\ 0)^T, (2\ 0\ 0\ 1)^T\}$; *as none of the markings in* $\mathcal{C}(\phi'\{t_4\})$ *is in* $W(M_F)$, u *remains empty. For* t_5, $\mathcal{C}_r(\phi'\{t_5\}) = \mathcal{C}_r(\{t_5\}) = \{M_7\}$ *where* M_7 *is given in Example 4.6. As* M_7 *is in* $W(M_F)$, u *becomes* $\{t_5\}$. *Therefore, the control action given* $\omega = \varepsilon$ *is* $u = \{t_5\}$.

Suppose next we observe the label a, *so that* $\omega = a$. *As the previous control action was* $u = \{t_5\}$, *we define* T' *such that* $T' = T_a \setminus u = \{t_1\}$ *and* $\mathcal{C}_r(\phi' T') = \mathcal{C}_r(T') = \{M_6, M_8, M_9, M_{10}, M_{11}, M_{12}\}$, *where* $M_6, M_8, ..., M_{12}$ *are given in Example 4.6. Now* ω *becomes* a *and* ϕ' *becomes* T' *(this is consistent with the discussion in Example 3.1). By repeating the above procedure, we can calculate control actions as additional labels are observed.* ■

Now we discuss whether previous approaches on transforming forbidden states into generalized mutual exclusion constraints can be applied to Example 5.3 or not.

Example 5.6 *Consider the setting in Example 5.3. There is only one forbidden state* $(1\ 0\ 0\ 1)^T$, *which can be described as*

$$M(p_1) = 1 \wedge M(p_2) = 0 \wedge M(p_3) = 0 \wedge M(p_4) = 1 \tag{3}$$

where $\wedge$ denotes conjunction (namely, logical "and"). Then states that are not forbidden can be described as

$$M(p_1) \leq 0 \vee M(p_1) \geq 2 \vee M(p_2) \geq 1 \vee M(p_3) \geq 1 \vee M(p_4) \leq 0 \vee M(p_4) \geq 2 \tag{4}$$

by taking the complement of Eq. (3) *(a general procedure is given in Lemma 1 in [25]), where $\vee$ denotes disjunction (namely, logical "or"). There are quite a few techniques proposed in [25] to simplify a disjunction of linear constraints (for example, the results in Theorems 4, 5, and 9); however, none of those techniques is immediately applicable to the constraint in Eq.* (4). *Though a conjunction of linear constraints is easy to handle (for example, using the place invariant based approach in [26]), a disjunction of linear constraints is much more difficult especially when there are uncontrollable and unobservable transitions [23].*

A different approach is proposed in [38] to represent forbidden states in non-safe Petri nets as linear constraints (more specifically, the procedure given in Algorithm 1 of [38]). The basic idea is to represent forbidden states as $w_1M(p_1) + w_2M(p_2) + w_3M(p_3) + w_4M(p_4) > k$, and states that are not forbidden (called authorized states in [38]) as $w_1M(p_1) + w_2M(p_2) + w_3M(p_3) + w_4M(p_4) \leq k$ for our specific example. Note that in our example the number of authorized states is finite but in general there could be an infinite number of authorized states. Since $(1\ 0\ 0\ 1)^T$ *is forbidden and* $(3\ 0\ 0\ 0)^T, (0\ 0\ 0\ 3)^T$ *are authorized, we have the following constraints regarding w_1, w_2, w_3, w_4, k:*

$$\begin{aligned} w_1 + w_4 &> k \\ 3w_1 &\leq k \\ 3w_4 &\leq k\ . \end{aligned} \tag{5}$$

It can be verified that there is no feasible solution to the set of inequalities in Eq. (5). *In other words, it is impossible to represent the forbidden state as a single linear constraint as required by the approach in [38].* ■

5.3 Extension to Petri Nets with Control Labels

In Section 2.2, we implicitly assumed that every controllable transition can be disabled separately from any other controllable transition. However, there could be constraints on the disabling of controllable transitions. More specifically, consider a scenario where a command actuator, if used, simultaneously disables a subset of controllable transitions. This could be the case, for example, in a 4-way intersection with a traffic light only showing red or green; a car can turn right, left or go straight if the light is green, and the car must stop otherwise. This scenario can be modeled using the Petri net in Fig. 5: transition t_1 (or t_2, t_3, respectively) models that the car turns left (or goes straight, turns right, respectively); place p_i models the number of cars in a certain location, e.g., p_1 models the number of cars that will enter the intersection, p_2

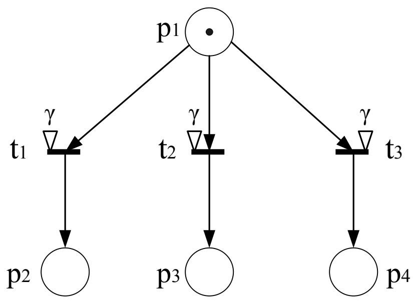


Figure 5: Simple Petri net with a shared control label.

models the number of cars that have turned left (and thus left the intersection), and so forth. All transitions are assigned the same control label γ, which models the action of the traffic light. If γ is disabled, then no car can move; otherwise, cars can turn right, left, or go straight. In general, we could define the control labels as a mapping $\Gamma : T_c \mapsto \Sigma_c$ in which Σ_c is the set of control labels and $\Gamma(t) \in \Sigma_c$ for $t \in T_c$ (similar to an observation label, a control label can be associated with multiple controllable transitions). Then the control set is of the form $U = \{u \mid u \subseteq \Sigma_c\}$. Note that this generalization imposes no changes to the definitions of control policy and maximally permissive control policy that we used earlier. However, a minor change is needed in Algorithm 5.4: at Step 3 in Stage 2, we output the control value[8]

$$u' = \{\gamma \in \Sigma_c \mid \exists t \in u : \gamma = \Gamma(t)\}$$

instead of u. Following the reasoning described after Algorithm 5.4, it can be verified that the resulting control policy is also maximally permissive for this more general scenario.

Remark 5.7 *The use of control labels has appeared in previous work. For example, in [36], different state transitions are allowed to be assigned the same label and some labels are controllable. Another example is the work in [19], in which some control place can control multiple controllable transitions.* ■

6 Practical example

In this section, we consider a simplified subway track system as shown in Fig. 6. The system consists of two tracks with a shared station that has only one track. Train 1 runs on Track 1 and Train 2 runs on Track 2; both trains run in the counterclockwise direction. The Petri net model of the track system is shown in Fig. 7. Transition t_1 (or t_6) corresponds to the event that Train 1 (or 2) is entering within a certain range of the station (this is for safety reasons). Transition t_2 (or t_5) corresponds to the event that Train 1 (or 2) is entering the shared track in the station. Transition t_3 (or t_4) represents the event that Train 1 (or 2) is leaving the shared track. Place

[8]Note that in order to disable a control label $\gamma \in \Sigma_c$, we only need to find one transition t satisfying $\Gamma(t) = \gamma$ such that transition t should be disabled. We could take advantage of this and not check whether we should disable other transitions t' such that $\Gamma(t') = \gamma$ in order to avoid unnecessary computation.

p_i for $i = 1, 2, 3$ (or $i = 4, 5, 6$) represents the state of Train 1 (or 2) in different regions of the track; for example, place p_3 represents that Train 1 is in the shared track. Initially, Train 1 is out of the shared track and is not within a certain range of the station either, whereas Train 2 is within a certain range of the station but has not entered the shared track yet. The objective is to avoid train collisions.

Suppose that the operator of Train 1 (or 2) reports a symbol a_1 (or a_2) if Train 1 (or 2) is within a certain range of the shared station and it can stop the train if necessary. Also suppose there is a range sensor which can detect if a train is leaving the shared station (and reports symbol b) but cannot distinguish one train from the other one. Correspondingly, we have the labeling function L as $L(t_1) = a_1$, $L(t_2) = L(t_5) = \varepsilon$, $L(t_3) = L(t_4) = b$ and $L(t_6) = a_2$. Transitions t_1 and t_6 are the only controllable transitions. Therefore, $T_o = \{t_1, t_3, t_4, t_6\}$ and $T_c = \{t_1, t_6\}$. Given the initial state $M_0 = (1\ 0\ 0\ 0\ 1\ 0)^T$, we need to design a control policy to avoid the forbidden state $(0\ 0\ 1\ 1\ 0\ 0)^T$, which corresponds to the situation that both trains are on the shared track. By applying Proposition 5.2, we get the set of weakly forbidden markings as $W(M_F) = \{(0\ 0\ 1\ 1\ 0\ 0)^T, (0\ 1\ 0\ 1\ 0\ 0)^T, (0\ 0\ 1\ 0\ 1\ 0)^T, (0\ 1\ 0\ 0\ 1\ 0)^T\}$. Since M_0 is not in $W(M_F)$, there exists a supervisor which can enforce the specification.

Before any observed label becomes available, we need to calculate u_1. Since M_0 is the only reduced consistent marking, we check if t_1 or t_6 need to be disabled. By applying Step 2 in Stage 2 of Algorithm 5.4, we get $u_1 = \{t_1\}$. If we next observe the label b, transition t_4 must have occurred since t_3 cannot occur due to the disabled transition t_1. Now $M = (1\ 0\ 0\ 0\ 0\ 1)^T$ is the only (reduced) consistent marking. Similarly, we can get the control action $u_2 = \emptyset$. We can repeat the above procedure, and calculate control actions as additional labels are observed.

For this specific example, there is no deadlock when applying the synthesized control policy. This can be proved by contradiction: suppose the system reaches deadlock due to the control policy; then it can be verified that the only possibility is that the current state is $M = (1\ 0\ 0\ 0\ 0\ 1)^T$ and the control is $u = \{t_1, t_6\}$. However, at state M, the calculated control action is $u = \emptyset$ as shown earlier. Therefore, the system will never go to deadlock. Note that the no concurrency assumption is key to guaranteeing no deadlock for this example; if concurrency is allowed (i.e., t_1 and t_6 can fire at the same time), $u = \emptyset$ can lead to weakly forbidden states. Even though we can determine the nonexistence of deadlock for this specific (small scale) system, in general it is a very challenging problem since it could involve exploring all reachable states. One practical approach could be the following: i) once a deadlock state is reached, it is added into the set of forbidden states M_F; ii) the set of weakly forbidden states is recalculated; iii) the system is reset to the initial state and then Algorithm 5.4 is applied. The procedure can be repeated until there is no new deadlock state, or the system needs to be redesigned with additional sensors if no supervisor exists.

Remark 6.1 *In large railway networks, there could be parallel tracks (from one station to another) in which there is no sensor or a very limited number of sensors available. For example, for a network that has A parallel tracks, B segments in each track, and no sensor available for each track (the Petri net model is shown in Fig. 8(a)), after observing the label a, there is only one reduced consistent marking in which places $p_{11}, p_{21}, ..., p_{A1}$ are all marked with one token;*

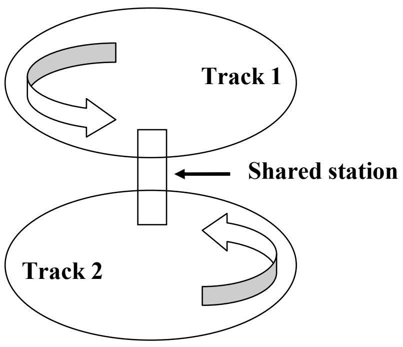


Figure 6: Sketch of two subway tracks with a shared station.

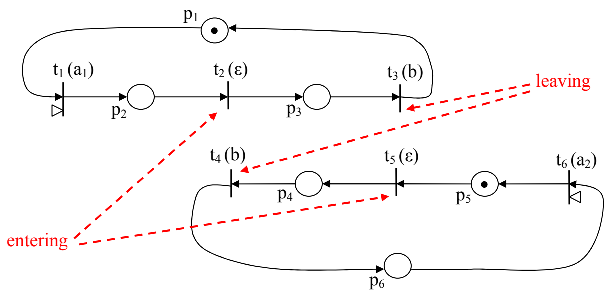


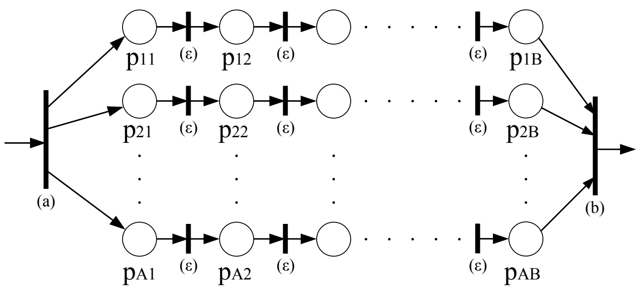
Figure 7: Petri net model of the subway track system in Fig. 6.

in contrast, there are A^B consistent markings.[9] *In general, there could be partial synchronization (one example is shown in Fig. 8(b)); in this case, we still have a single reduced consistent marking while the number of consistent markings is still exponential in the number of parallel tracks. Therefore, the use of reduced consistent markings (instead of consistent markings) can be advantageous in supervisor synthesis. Note that such structures also appear in other settings, such as multiple routes in computer/communication/transportation networks.* ■

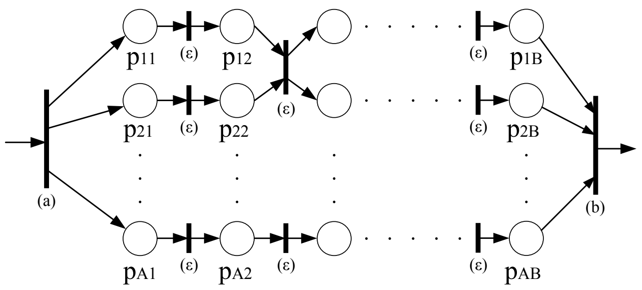
7 Conclusions

In this paper, we consider the arbitrary forbidden state problem in DESs modeled by partially observed and partially controlled Petri nets. Note that the given Petri net can be unbounded, which allows us to deal with infinite state spaces. Under the assumption that $T_{uo} \subseteq T_{uc}$ and certain conditions on the unobservable subnet and the uncontrollable subnet, we show that reduced consistent markings can be used not only to represent the set of consistent markings but also to determine the control policy. We also propose an online algorithm to determine the control policy based on the sequence of observed labels and the sequence of previous control actions, and discuss the extension of our approach to Petri nets with control labels. Note that if

[9]If one of the consistent markings is a forbidden marking, in the worst case there could be A^B weakly forbidden markings. However, since weakly forbidden markings are calculated only once and offline, the online computational complexity is not affected because the checking of whether a reduced consistent marking is weakly forbidden involves linear complexity in the number of places (this is also shown in Appendix C).



(a) Parallel tracks.



(b) Parallel tracks with synchronization.

Figure 8: Reduced consistent markings versus consistent markings.

Assumption **A2** is violated, it could be possible that for a certain observation sequence ω, the set of consistent markings is infinite (which could lead to computational problems); if Assumption **A3** is violated, it could be possible that the set of weakly forbidden markings is infinite even though the set of forbidden markings is finite. We believe that these two assumptions allow us to consider very general scenarios in terms of an arbitrary observation sequence ω and an arbitrary finite set of forbidden states.

There are several interesting future directions. First, it is worth finding the smallest set $\mathcal{E}$ to represent the set of consistent markings because it could significantly reduce the complexity of calculating control actions. Bounding the number of markings in $\mathcal{E}$ (or reduced consistent markings) directly is an interesting problem by iteself. Second, due to the exponential dependency of the complexity functions on Petri net size in Appendix C, we would like to explore special structures in large-scale systems along with specific sensor configurations that make our approach applicable, and further develop software tools for practical applications. One example is the large-scale railway system as studied in [16]. Third, we would like to incorporate the deadlock-free constraint in our problem formulation, and investigate the applicability of other existing deadlock resolution approaches (e.g., the work in [22]).

Appendix: Complexity Analysis of Algorithm 5.4

A. Lemmas

Before discussing the computational complexity of Algorithm 5.4, we first introduce some notation.

Given a partially observed and partially controlled Petri net Q, the unobservable subnet is denoted by $N_{T_{uo}}$, and has $n_{uo} = |P_{uo}|$ places and $m_{uo} = |T_{uo}|$ transitions. Under Assumption **A2**, the unobservable subnet is structurally bounded or, equivalently, there exists an n_{uo}-dimensional column vector y_1 with positive integer entries, such that $y_1^T D_{uo} \leq \mathbf{0}_{m_{uo}}^T$, where D_{uo} is obtained from D by keeping rows that correspond to places in P_{uo} and columns that correspond to unobservable transitions in T_{uo}. Similarly, the uncontrollable subnet is denoted by $N_{T_{uc}}$, and has n places and $m_{uc} = |T_{uc}|$ transitions. Under Assumption **A3**, the reverse net of $N_{T_{uc}}$ is structurally bounded or, equivalently, there exists an n-dimensional column vector y_2 with positive integer entries, such that $y_2^T(-D_{uc}) \leq \mathbf{0}_{m_{uc}}^T$, where D_{uc} is obtained from D by keeping columns that correspond to uncontrollable transitions in T_{uc}.

The following lemma (refer to Theorem 2 in [32]) gives an upper bound on the number of markings consistent with an observation sequence ω in a partially observed and partially controlled Petri net Q with labeling function L and a structurally bounded unobservable subnet.

Lemma 7.1 *Consider a partially observed and partially controlled Petri net Q with labeling function L and a structurally bounded unobservable subnet $N_{T_{uo}}$. If the observation sequence ω has length k, then the number of consistent markings is upper bounded by*

$$(1 + a_1 + a_2 k)^{n - n_{uo}} (1 + c_1 + c_2 k)^{n_{uo}},$$

where[10] *$a_1 = \max_{p \in P \setminus P_{uo}} M_0(p)$, $a_2 = \max_{p \in P \setminus P_{uo}, t \in T_o} D(p,t)$, $c_1 = y_1^T M_0^{uo}$, and c_2 is the maximal entry of $y_1^T D_o^{uo}$ (note that M_0^{uo} is the restriction of M_0 to places in P_{uo}, and D_o^{uo} is the submatrix of the incidence matrix D that has rows that correspond to the places in P_{uo} and columns that correspond to the transitions in T_o).*

In Lemma 1, a_1, a_2, c_1, c_2 are constants which depend on the initial state M_0 and the incidence matrix D, and thus the number of markings consistent with a k-length observation sequence is $\mathcal{O}(k^n)$. Also, in the derivation of the above bound, one obtains that for each place $p \in P_{uo}$ and any $M \in \mathcal{C}(\omega)$, $0 \leq M(p) \leq c_1 + c_2 k$.

B. Reachability Analysis

Both Step 1 of Algorithm 4.4 and Proposition 5.2 involve reachability analysis: in Step 1 of Algorithm 4.4, one needs to calculate all markings that are reachable from a reduced consistent

[10] If $P_{uo} = P$, both a_1 and a_2 can be taken to be 0. If $T_{uo} = \emptyset$, then the value of y_{uo}, c_1 and c_2 are not defined; as the value of y_{uo} is not important in this case, we can take both c_1 and c_2 to be 0.

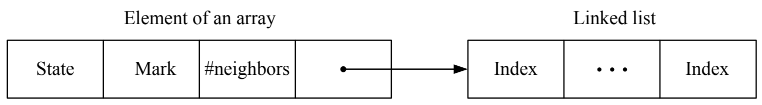


Figure 9: Data structure used for reachability analysis.

marking M by a sequence of unobservable transitions, while in Proposition 5.2, one needs to calculate all markings that are reachable from a forbidden marking M by a sequence of uncontrollable transitions in the reverse net of the uncontrollable subnet. In general, one could formulate the following reachability problem: given a K-bounded Petri net $G = \langle N, M_0 \rangle$ with n places and m transitions, calculate all markings reachable from M_0. To solve the problem efficiently, we propose a method based on a data array $\mathcal{DA}$ of dimension B^n, where $B = K + 1$. Each element of the array is shown in Fig. 9. Before we explain the array in detail, we first recall the following *injective* transformation function as defined in [33].

Definition 7.2 *Given a K-bounded Petri net G, the transformation function* $\Gamma : R(G) \to N_0$ *is defined as,*

$$\Gamma(M) = M(p_1) + M(p_2)B + M(p_3)B^2 + \cdots + M(p_n)B^{n-1}$$

where $B = K + 1$, $M \in R(N, M_0)$.

Given a marking M, the value $\Gamma(M)$ can be calculated via the following recursion

$$\begin{aligned} S_1 &= M(p_n) \\ S_2 &= B \times S_1 + M(p_{n-1}) \\ &\vdots \\ S_k &= B \times S_{k-1} + M(p_{n+1-k}) \\ &\vdots \\ S_n &= B \times S_{n-1} + M(p_1) \end{aligned}$$

and $\Gamma(M) = S_n$. It can be verified that the complexity of this recursion is $2(n-1)$ in terms of the number of additions and multiplications.

For any marking $M \in R(N, M_0)$, we first calculate $\Gamma(M)$ and use $\Gamma(M)$ as the index for M in the array $\mathcal{DA}$. In the element with index $\Gamma(M)$, the field "State" (in Fig. 9) stores the state M, the field "Mark" is binary and is initialized to be 0, the field "#neighbors" is equal to

$$|\{M' \in \mathbb{N}^n \mid \exists t \in T : M[t\rangle M'\}| \ ,$$

and the pointer field points to a linked list representing the set

$$\{\Gamma(M') \mid \exists t \in T : M[t\rangle M'\} \ .$$

Intuitively, the element corresponding to marking M includes necessary information of all one-step reachable markings. Because the Petri net is K-bounded and $B = K + 1$, the array is large enough to store all possible reachable markings for the Petri net. Based on this data array, we propose the following reachability algorithm.

Algorithm 7.3 [Calculation of Reachable Markings]

Input: *A K-bounded Petri net $G = \langle N, M_0 \rangle$ with n places and m transitions.*

Output: *A representation of all reachable markings.*

```
1. Initialize the data array $\mathcal{DA}$ for $B^n$ markings, where $B = K + 1$. $\mathcal{DA}(\Gamma(M))$, the element corresponding to $M$ in the array, has the structure shown in Fig. 9, and is initialized with "state" being $M$, "Mark" being $0$, "#neighbors" being $|\{M' \in \mathbb{N}^n \mid \exists t \in T : M[t\rangle M'\}|$, and the pointer field pointing to a linked list representing the set $\{\Gamma(M') \mid \exists t \in T : M[t\rangle M'\}$.

2. Initialize $Q$ to the set[11] $\{\Gamma(M_0)\}$.

3. While $Q \neq \emptyset$

Pick one element $s$ in $Q$

Change the field "Mark" in the element $\mathcal{DA}(s)$ to be $1$.

For any $s'$ in the linked list pointed by the pointer in $\mathcal{DA}(s)$

    If the "Mark" field of $\mathcal{DA}(s')$ is $0$, add $s'$ to the set $Q$.

Remove $s$ from the set $Q$.

4. The states corresponding to the elements of the array with the field "Mark" being $1$ form the set of all reachable markings.
```

We first explain the algorithm before analyzing its complexity. In Step 1, we build the data array, a process that involves the calculation of all possible one-step state transitions in the Petri net. In Steps 2 and 3, we mark all reachable states starting from M_0 by traversing the data array. The algorithm must stop because the number of reachable states is bounded. In Step 4, we obtain a representation of all reachable markings using the data array.

We now analyze the computational complexity of the algorithm. In Step 1, we build the array for any marking M among B^n possible markings and the complexity is

$$B^n \times (2(n-1) + m \times (n + n + 2(n-1))) \ ,$$

where the first $2(n-1)$ is the complexity of calculating $\Gamma(M)$, m refers to the number of transitions, the first n in $m \times (n+n+2(n-1))$ refers to the number of comparisons to determine if transition t (among m transitions) is enabled or not, the second n in $m \times (n + n + 2(n-1))$ refers to the number of additions to calculate the next state M' if transition t is enabled, and the last $2(n-1)$ refers to the calculation of $\Gamma(M')$. The complexity of Step 1 is roughly $\mathcal{O}(nmB^n)$. The complexity of Steps 2 and 3 is roughly $\mathcal{O}(mB^n)$ because any state can be marked to be 1 at most once and can reach at most m other states. In summary, the complexity of Algorithm 7.3 is $\mathcal{O}(nmB^n)$, where $B = K + 1$.

[11]In practical implementations, one could use a first in first out queue to represent the set Q.

C. Complexity of Algorithm 5.4

Now we examine the computational complexity of Algorithm 5.4.

Stage 1: Offline Checking of Supervisor Existence

In Step 1, we calculate the set of weakly forbidden markings $W(M_F)$ following Proposition 5.2. For any marking M' reachable from some $M \in M_F$, there exists a firing vector σ such that

$$M' = M + (-D_{uc})\sigma \ . \tag{6}$$

Since $N'_{T_{uc}}$ is structurally bounded, there exists a vector y_2 such that $y_2^T(-D_{uc}) \leq \mathbf{0}^T_{m_{uo}}$. If we left-multiply by y_2^T on both sides of Eq. (6), we get

$$y_2^T M' = y_2^T M + y_2^T(-D_{uc})\sigma \leq y_2^T M$$

Therefore, for any place p, $0 \leq M'(p) \leq y_2^T M' \leq y_2^T M$. To apply Algorithm 7.3, we could choose K to be $y_2^T M$. Therefore, the set of markings reachable from M in the reverse net of the uncontrollable subnet can be calculated with complexity $\mathcal{O}(nm_{uc}(y_2^T M + 1)^n)$ using Algorithm 7.3. One straightforward way to calculate all weakly forbidden markings is to apply Algorithm 7.3 $|M_F|$ times and the complexity is

$$\sum_{M \in M_F} \mathcal{O}(nm_{uc}(y_2^T M + 1)^n)$$

which can be relaxed as

$$\mathcal{O}(|M_F| nm_{uc}(\max_{M \in M_F} y_2^T M + 1)^n) \ .$$

However, there is a more efficient implementation. We choose K as

$$\max_{M \in M_F} y_2^T M$$

and change Step 2 of Algorithm 7.3 as follows: let Q be $\{\Gamma(M) \mid M \in M_F\}$. Then the output of the changed algorithm is the set of weakly forbidden markings. The complexity is

$$\mathcal{O}(nm_{uc}(\max_{M \in M_F} y_2^T M + 1)^n) \ .$$

In Step 2, we check if $M_0 \in W(M_F)$. With the data array, the checking can be done by first calculating $\Gamma(M_0)$ and then checking if the field "Mark" of the element with index $\Gamma(M_0)$ is 1 or 0: if the field "Mark" is 1, then $M_0 \in W(M_F)$; otherwise, $M_0 \notin W(M_F)$. The complexity is $\mathcal{O}(n)$ because accessing an array can be done in constant time; the complexity is essentially the complexity of calculating $\Gamma(M_0)$.

Stage 2: Online Determination of Control Policy

The bulk of the online computation is due to Step 2 and Step 5 in Stage 2 of Algorithm 5.4. If ϕ' is of length $k-1$, then $\mathcal{C}_r(\phi') \subseteq \mathcal{C}(\phi') \subseteq \mathcal{C}(\omega)$ and $|\mathcal{C}_r(\phi')| \leq |\mathcal{C}(\omega)| = \mathcal{O}((k-1)^n)$. Similarly, $|\mathcal{C}_r(\phi'T')| \leq \mathcal{O}(k^n)$.

Before analyzing Step 2, we first analyze the complexity of updating the set of reduced consistent markings using Algorithm 4.4. Given $\mathcal{C}_r(\phi')$, we want to compute $\mathcal{C}_r(\phi'T')$. For any $M \in \mathcal{C}_r(\phi')$, we first need to calculate the set of markings reachable from M in the unobservable subnet,[12] which will give us $\mathcal{C}(\phi')$. Recall the bound on the number of tokens that each place $p \in P_{uo}$ can have for any $M' \in \mathcal{C}(\omega)$ in Appendix A, namely $0 \leq M'(p) \leq c_1 + c_2k$, where c_1 and c_2 are given in Lemma 7.1. Therefore, the set of markings reachable from any $M \in \mathcal{C}_r(\phi')$ in the unobservable subnet can be calculated with complexity $\mathcal{O}(n_{uo}m_{uo}(c_1+c_2k+1)^{n_{uo}})$ using the more efficient version of Algorithm 7.3 for a set of initial states. After computing $\mathcal{C}(\phi')$, we compute $\mathcal{C}_r(\phi'T')$ using the procedure in Algorithm 4.4. The complexity is

$$\mathcal{O}(|\mathcal{C}(\phi')| \times |T'| \times (n + n + n|\mathcal{C}_r(\phi'T')|)),$$

where the first n corresponds to checking if the transition is enabled or not, the second n corresponds to the calculation of the next marking M', and the last term corresponds to checking if M' has appeared in $\mathcal{C}_r(\phi'T')$ using linear search. The complexity is roughly $\mathcal{O}(nmk^{2n})$. Putting these results together, we have that the complexity of updating reduced consistent markings is

$$\text{UpdateComplexity} = \mathcal{O}(nmk^{2n}) \ .$$

As only the update of reduced consistent markings is involved in Step 5, this complexity is also the complexity of Step 5. In Step 2, the complexity can be expressed as

$$|T_c| \times (\text{UpdateComplexity} + |\mathcal{C}_r(\phi'T')|n) \ ,$$

where the last n refers to the complexity of checking if a reduced consistent marking is weakly forbidden. The complexity in Step 2 is roughly $\mathcal{O}(nm^2k^{2n} + nmk^n) = \mathcal{O}(nm^2k^{2n})$. In summary, the complexity of one iteration from $k-1$ to k is $\mathcal{O}(nm^2k^{2n})$. If we start with the empty string, the cumulative complexity of the online computation up to an observation sequence of length k is $\mathcal{O}(nm^2k^{2n+1})$. Note that the online computational complexity is polynomial in the length k of the observed sequence of labels, and exponential in the number of places n.

Remark 7.4 *Note that there exist Petri nets such that the number of reduced consistent markings (and basis markings) can increase exponentially in the Petri net size. One example is given in Fig. 10. In this specific Petri net, there are $2n$ places and $2n$ transitions. The n source transitions that are labeled a are observable while the other n transitions are unobservable. If the observation is a sequence $aaa\cdots a$ of length k, then any reduced consistent marking satisfies $\sum_{i=1}^{n} M(p^{1i}) = k$ and $M(p^{2i}) = 0$ for $i = 1, 2, ..., n$. It can be verified that the number of reduced consistent markings is $\binom{k+n-1}{n-1}$, where $\binom{k}{r} = \frac{k!}{r!(k-r)!}$ is the binomial coefficient "k choose*

[12]Note that, in Definition 2.6, $P_{uo} \subseteq P$. Therefore, strictly speaking, calculating the unobservable reach from marking M is not exactly the same as calculating all reachable markings from M in the unobservable subnet. However, for any place $p \in P \setminus P_{uo}$, the number of tokens does not change in the calculation of the unobservable reach.

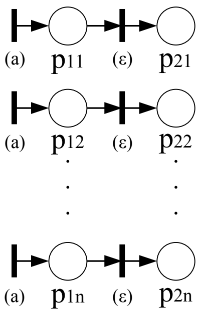


Figure 10: Example for reduced consistent markings.

r''. Therefore, the number of reduced consistent markings is $\mathcal{O}(k^{n-1})$, which is exponential in the number of places. It can also be verified that any reduced consistent marking for this Petri net is also a basis marking based on the definition in Footnote 7, and that the set of reduced consistent markings is a set $\mathcal{E}$ of minimum cardinality. This example shows that, in general, the complexity is exponential in Petri net size, and this appears to be unavoidable even when using the smallest set of markings that can represent the set of consistent markings, as in Eq. (2). In the next subsection, we will discuss different special types of Petri nets in which the exponential dependency could potentially be alleviated. ■

D. Alleviation of Exponential Dependency on Petri Net Size

In general, the complexity of the online computation is exponential in n, the number of places in the given Petri net. However, if we impose further assumptions, this exponential dependency could be alleviated. For example, if the unobservable subnet is acyclic and backward conflict free and if the labeling function is the natural projection, then the basis marking is unique as shown in [17]. Another example is that if the Petri net model is a marked graph and if the labeling function is the natural projection, then there exists a unique marking which serves as the set $\mathcal{E}$ of minimum cardinality [1]. If we use such a unique marking instead of reduced consistent markings for supervisor synthesis, then the complexity function could be a linear function of k.

For certain Petri nets with large n, but small n_{uo} (recall that n_{uo} is the cardinality of P_{uo} as defined in Definition 2.6) and small number of transitions m, the algorithm could be useful under slightly stronger assumptions: specifically, if the unobservable subnet is deadlock structurally bounded (which is stronger than Assumption **A2**), the number of markings consistent with ω of length k is $\mathcal{O}(k^{j(\bar{l}-1)+m_{uo}})$, where j is the number of nondeterministic labels (i.e., nonempty labels that can be associated with more than one transition), and $\bar{l}$ is the maximum number of transitions that can be associated with a nondeterministic label [32]. With this bound, we can show that the online computational complexity becomes $\mathcal{O}(n_{uo}m_{uo}k^{n_{uo}+1}+nm^2k^{2(j(\bar{l}-1)+m_{uo})+1})$ following the same reasoning as before (note that the exponent $2(j(\bar{l}-1)+m_{uo})+1$ is $\mathcal{O}(m^2)$). The values of m_{uo}, n_{uo}, j and $\bar{l}$ are critical in determining the complexity of the algorithm.

Notice that for any given positive constant integer C satisfying[13]

$$1 \leq C \leq \lfloor \frac{2m}{3} + 1 \rfloor \ ,$$

there exist labeling functions that can make the exponents in $\mathcal{O}(n_{uo} m_{uo} k^{n_{uo}+1} + nm^2 k^{2(j(\bar{l}-1)+m_{uo})+1})$ less than or equal to C. More specifically, we can construct a labeling function in the following way:

- There is a nonempty label for any transition $t \in T$; in other words, $m_{uo} = n_{uo} = 0$ so that $n_{uo} + 1$ is 1 and $2(j(\bar{l}-1) + m_{uo}) + 1$ is $2j(\bar{l}-1) + 1$.
- If we choose $j(\bar{l}-1) \leq \lfloor \frac{C-1}{2} \rfloor$, then $2(j(\bar{l}-1) + m_{uo}) + 1 = 2j(\bar{l}-1) + 1 \leq C$ and $n_{uo} + 1 = 1 \leq C$. Note that $\lfloor \frac{C-1}{2} \rfloor$ is a nonnegative integer.
- To guarantee $j(\bar{l}-1) \leq \lfloor \frac{C-1}{2} \rfloor$, we can first factorize[14] $\lfloor \frac{C-1}{2} \rfloor$ to be $C_1 \times C_2$ where C_1 and C_2 are both nonnegative integers, and then set

  $$j = C_1, \text{ and } \bar{l} = C_2 + 1 \ .$$

  If $\lfloor \frac{C-1}{2} \rfloor = 0$, we set $C_1 = C_2 = 0$.
- If $j = 0$ (i.e., there is no nondeterministic label), then we can construct labeling function $L(t) = t$ for any $t \in T$. If $j \geq 1$, $\bar{l}$ must be larger than 1 since $C_2 > 0$ (note that if $C_2 = 0$, $j = C_1$ must be zero). Given j and $\bar{l}$, there could be multiple labeling functions which result in the same j and $\bar{l}$. We can construct a specific one by associating $t_1, t_2, ..., t_{\bar{l}}$ with label e_1, associating $t_{\bar{l}+2i-1}, t_{\bar{l}+2i}$ with label e_{1+i} for $i = 1, ..., j-1$, and associating t_i with label t_i for $i = \bar{l} + 2j - 1, ..., m$. Note that $\bar{l} + 2j - 1 = C_2 + 1 + 2C_1 - 1 \leq \frac{C-1}{2} + 2\frac{C-1}{2} \leq m$.

For example, if C is chosen to be 3, then we can set j to be 1 and $\bar{l}$ to be 2. The labeling function constructed according to the above procedure is $L(t_1) = L(t_2) = e_1$ and $L(t_i) = t_i$ for $i = 3, ..., m$. More generally, we could construct a labeling function as $L(t_i) = L(t_j) = e_1$ for some $i, j \in \{1, 2, ..., m\}$, and $L(t_k) = e_k$ for $k \in \{1, 2, ..., m\} \setminus \{i, j\}$. The implication of the above analysis is that for Petri nets with large n and large m, there exist sensor configurations (namely, labeling functions) that allow the application of our approach with prescribed computational complexity. The above construction of sensor configurations assumes that there is no unobservable transition. If we do allow unobservable transitions, we can perform the same analysis except that C cannot be made as small as 1 (instead, C will be lower bounded by one plus the number of unobservable transitions).

[13]The floor function $\lfloor x \rfloor$ is the largest integer which is smaller than or equal to x.

[14]Note that in general the factorization of $\lfloor \frac{C-1}{2} \rfloor$ into a pair of factors is not unique. If $\lfloor \frac{C-1}{2} \rfloor \geq 1$, it can always be factorized as $1 \times \lfloor \frac{C-1}{2} \rfloor$. Every pair of factors can generate a class of labeling functions in the sense that these labeling functions share the same parameters j and $\bar{l}$.

References

[1] Z. Achour, N. Rezg, and Xiaolan Xie. Supervisory control of partially observable marked graphs. *IEEE Transactions on Automatic Control*, 49:2007–2011, November 2004.

[2] Z. Achour, N. Rezg, and Xiaolan Xie. On the existence of Petri net controller for discrete event systems under partial observation. In *Proc. of the 16th IFAC World Congress 2005*, July 2005.

[3] F. Basile and P. Chiacchio. On the implementation of supervised control of discrete event systems. *IEEE Trans. on Control Systems Technology*, 15(4):725 –739, July 2007.

[4] F. Basile, P. Chiacchio, and C. Carbone. Feedback control logic for backward conflict free choice nets. *IEEE Trans. on Automatic Control*, 52(3):387–400, March 2007.

[5] M.P. Cabasino, A. Giua, M. Pocci, and C. Seatzu. Discrete event diagnosis using labeled Petri nets. An application to manufacturing systems. *Control Engineering Practice*, 19(9):989–1001, September 2011.

[6] H. Chen. Control synthesis of Petri nets based on S-decreases. *Discrete Event Dynamic Systems: Theory and Applications*, 10:233–249, 2000.

[7] S.-L. Chung, S. Lafortune, and F. Lin. Limited lookahead policies in supervisory control of discrete event systems. *IEEE Transactions on Automatic Control*, 37:1921–1935, 1992.

[8] S.-L. Chung, S. Lafortune, and F. Lin. Supervisory control using variable lookahead policies. *Discrete Event Dynamic Systems: Theory and Applications*, 4:237–268, 1994.

[9] A. Dideban, M. Zareiee, and H. Alla. Controller synthesis with very simplified linear constraints in PN model. In *2nd IFAC workshop on Dependable Control of Discrete Systems*, pages 265–270, Bari, Italy, 2009.

[10] Abbas Didebana and Hassane Alla. Reduction of constraints for controller synthesis based on safe Petri nets. *Automatica*, 44:1697–1706, July 2008.

[11] S. Genc and S. Lafortune. Distributed diagnosis of place-bordered Petri nets. *IEEE Transactions on Automation Science and Engineering*, 4:206–219, April 2007.

[12] H. J. Genrich and E. Stankiewicz-Wiechno. A dictionary of some basic notions of net theory. *Lecture Notes in Computer Science, Vol. 84: Net Theory and Applications*, pages 519–535, 1980.

[13] A. Ghaffari, N. Rezg, and Xiaolan Xie. Design of a live and maximally permissive Petri net controller using the theory of regions. *IEEE Transactions on Robotics and Automation*, 19:137–142, February 2003.

[14] A. Giua, F. DiCesare, and M. Silva. Generalized mutual exclusion constraints on nets with uncontrollable transitions. In *Proc. of the IEEE Int. Conf. on Systems, Man and Cybernetics*, pages 974–979, October 1992.

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

[16] A. Giua and C. Seatzu. Modeling and supervisory control of railway networks using Petri nets. *IEEE Transactions on Automation Science and Engineering*, 5:431–445, July 2008.

[17] A. Giua, C. Seatzu, and D. Corona. Marking estimation of Petri nets with silent transitions. *IEEE Transactions on Automatic Control*, 52:1695–1699, September 2007.

[18] M. Heymann and F. Lin. On-line control of partially observed discrete event systems. *Discrete Event Dynamic Systems: Theory and Applications*, 4:221–236, 1994.

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

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

[21] Hesuan Hu, Mengchu Zhou, and Zhiwu Li. Algebraic synthesis of timed supervisor for automated manufacturing systems using Petri nets. *IEEE Transactions on Automation Science and Engineering*, 7(3):549–557, 2010.

[22] Hesun Hu, Mengchu Zhou, and Zhiwu Li. Supervisor optimization for deadlock resolution in automated manufacturing systems with Petri nets. *IEEE Transactions on Automation Science and Engineering*, 8(4):794–804, 2011.

[23] M. V. Iordache and P. J. Antsaklis. Petri net supervisors for disjunctive constraints. In *Proc. of American Control Conference*, pages 4951–4956, 2007.

[24] R. Kumar, H. M. Cheung, and S. I. Marcus. Extension based limited lookahead supervision of discrete event systems. *Automatica*, 34:1327–1344, November 1998.

[25] Jiliang Luo and K. Nonami. Approach for transforming linear constraints on Petri nets. *IEEE Transactions on Automatic Control*, 56:2751–2765, December 2011.

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

[27] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77:541–580, April 1989.

[28] P. J. Ramadge and W. M. Wonham. Modular feedback logic for discrete event systems. *SIAM J. Control and Optimization*, 25:1202–1218, September 1987.

[29] P. J. Ramadge and W. M. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77:81–98, January 1989.

[30] Y. Ru, M. P. Cabasino, A. Giua, and C. N. Hadjicostis. Supervisor synthesis for discrete event systems with arbitrary forbidden state specifications. In *Proc. of the 47th IEEE Conf. on Decision and Control*, pages 1048–1053, December 2008.

[31] Y. Ru and C. N. Hadjicostis. Fault-tolerant supervisory control of discrete event systems modeled by bounded Petri nets. In *Proc. of 2007 American Control Conference*, pages 4945–4950, July 2007.

[32] Y. Ru and C. N. Hadjicostis. Bounds on the number of markings consistent with label observations in Petri nets. *IEEE Transactions on Automation Science and Engineering*, 6:334–344, April 2009.

[33] Y. Ru, W. Wu, H. Su, and J. Chu. Supervisor synthesis for bounded Petri nets based on a transformation function. In *Proc. of 2004 American Control Conference*, pages 4493–4498, June 2004.

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

[35] G. Stremersch and R. K. Boel. Structuring acyclic Petri nets for reachability analysis and control. *Discrete Event Dynamic Systems: Theory and Applications*, 12:7–41, 2002.

[36] W. M. Wonham. *Supervisory Control of Discrete-Event Systems.* Toronto, Canada, 2009.

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

[38] M. Zareiee, A. Dideban, and P. Nazemzadeh. From forbidden states to linear constraints. In *World Academy of Science, Engineering and Technology*, pages 167–173, 2011.

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