A Method to Verify the Controllability of Language Specifications in Petri Nets Based on Basis Marking Analysis

Ziyue Ma, Zhiwu Li, and Alessandro Giua

December 15, 2015

Abstract

In this paper we propose an effective method based on basis marking analysis to verify the controllability of a given language specification in Petri nets. We compute the product, i.e., the concurrent composition, of the plant and of the specification nets and enumerate a subset of its reachable states, called basis markings. Each of these basis markings can be classified as controllable or uncontrollable by solving an integer programming problem. We show that the specification is controllable if and only if no basis marking is uncontrollable. The method can lead to practically efficient verification because it does not require an exhaustive enumeration of the state space.

Published as:

["A Method to Verify the Controllability of Language Specifications in Petri Nets Based on Basis Marking Analysis," 54nd IEEE Conf. on Decision and Control (Osaka, Japan), Dec. 15-18, 2015.]
DOI: 10.1109/CDC.2015.7402451.

Z. Ma is with the School of Electro-Mechanical Engineering, Xidian University, Xi'an 710071, China (e-mail: maziyue@gmail.com), and also with Dipartimento di Ingegneria Elettrica ed Elettronica, Università degli Studi di Cagliari, Cagliari 09123, Italy.

Z. Li is with the School of Electro-Mechanical Engineering, Xidian University, Xi'an 710071, China, and also with Institute of Systems Engineering, Macau University of Science and Technology, Taipa, Macau (e-mail: zhwli@xidian.edu.cn, systemscontrol@gmail.com).

A. Giua is with Aix Marseille Université, CNRS, ENSAM, Université de Toulon, LSIS UMR 7296, Marseille 13397, France (Email: alessandro.giua@lsis.org), and also with Dipartimento di Ingegneria Elettrica ed Elettronica, Università degli Studi di Cagliari, Cagliari 09123, Italy (Email: giua@diee.unica.it).

1 Introduction

Supervisory Control Theory, originated by the work of Ramadge and Wonham [1], is a system theory approach that provides a unifying framework for the control of Discrete Event Systems (DESs). In this setting, the control demands are characterized by two types of specifications. A *state specification* consists in a forbidden set of states, while a *language specification* consists in a forbidden set of event sequences. The control problem is usually complicated by the presence of uncontrollable events that cannot be disabled by a control agent, called supervisor, to prevent violating a specification.

In the work of Ramadge and Wonham, *finite state automata* were used to model the plant, i.e., the system to be controlled. A state specification can be enforced by preventing the plant to reach the set of weakly forbidden states, i.e., those states from which a forbidden one can be reached by firing uncontrollable events. When language specifications are considered, a *monolithic supervisor candidate* (MSC) is first constructed as the product[1] of the plant and of the automaton generating the language specification. By this approach, enforcing the original language specification can be transformed into the problem of enforcing a related state specification in the MSC. In particular if the MSC is controllable, the specification can be used as a supervisor. If the MSC is not controllable, it can be refined to determine the structure of a minimally restrictive supervisor.

Petri nets extend the class of control problems that can be solved by automata and provide many efficient and well founded approaches for supervisory control. Several interesting results have been obtained for state specifications: in this case efficient algorithms exist to compute a controller in the presence of uncontrollable transitions [3–8]. However, relatively few works have discussed how Petri net models may be efficiently used to design supervisors for arbitrary language specifications. Although the product of Petri net modules can be efficiently computed without enumerating its state space [9], there is no efficient general method to determine the controllability of a given MSC or to refine it. Thus, analyzing and refining a Petri net MSC requires a brute-force approach to compute its entire reachability graph, and is thus hindered by the well known *state explosion problem*.

Recently a state compression approach [10] has been used for state estimation and fault diagnosis in Petri nets [11]. Here the set of transitions (i.e., events) is partitioned into *observable* and *unobservable* ones. The advantages of this technique is that only part of the reachability space, the so-called *basis markings*, is enumerated; all other markings reachable from them firing only unobservable transitions can be characterized by a linear system.

In this paper we show that the basis marking approach can be generalized to solve other problems in which the set of transitions can be appropriately partitioned into two disjoint sets. In particular, we use this approach to verify the controllability of a given language specification by constructing the *basis reachability*

[1] Following [2] we use the term *product* to denote the counterpart on the systems structure (be it an automaton or a Petri net) of the *concurrent composition* operator on languages.

graph (BRG) of a given MSC.

We show that the set of uncontrollable markings of an MSC can always be characterized by a special class of state specifications called *OR-AND GMECs* which can be easily enforced by Petri net controllers [12]. This allows one to partition the set of basis marking into the set of controllable and uncontrollable ones by solving a series of integer programming problems.

Finally we prove that the MSC (and hence the corresponding specification) is controllable if and only if all its basis marking are controllable: as a result the exhaustive enumeration of the entire reachability graph of the MSC can be avoided. Should the MSC fail to be controllable, the basis marking approach can also be used to efficiently design a supervisor. However, due to space constraints, this problem is not discussed in this manuscript and will be addressed in the future.

Our approach is presented for nets that satisfy the following assumption: the uncontrollable subnet of the MSC is acyclic. We observe, however, that this assumption may be relaxed at the cost of increasing the number of basis markings.

This paper is organized in five sections. The notions of labelled Petri nets and product operator are recalled in Section II. Section III introduces the notions of language specification and the construction of monolithic supervisor candidate. Section IV introduces the method to construct the BRG followed by the method to verify the controllability of a specification by analyzing the BRG, while some simulation results are presented. Conclusions are drawn in Section V.

2 Preliminaries

2.1 Petri Net

A Petri net is a four-tuple $N = (P, T, Pre, Post)$, where P is a set of *m places* represented by circles; *n transitions* represented by bars; $Pre : P \times T \to \mathbb{N}$ and $Post : P \times T \to \mathbb{N}$ are the *pre-* and *post-incidence functions* that specify the arcs in the net and are represented as matrices in $\mathbb{N}^{m \times n}$ (here $\mathbb{N} = \{0, 1, 2, \ldots\}$). The *incidence matrix* of a net is defined by $C = Post - Pre \in \mathbb{Z}^{m \times n}$ (here $\mathbb{Z} = \{0, \pm 1, \pm 2, \ldots\}$).

For a transition $t \in T$ we define its *set of input places* as ${}^\bullet t = \{p \in P \mid Pre(p,t) > 0\}$ and its *set of output places* as $t^\bullet = \{p \in P \mid Post(p,t) > 0\}$. The notion for ${}^\bullet p$ and $p^\bullet$ are analogously defined.

A *marking* is a vector $M : P \to \mathbb{N}$ that assigns to each place of a Petri net a non-negative integer number of tokens, represented by black dots and can also be represented as a m component vector. We denote by $M(p)$ the marking of place p. A *marked net* $\langle N, M_0 \rangle$ is a net N with an initial marking M_0. We denote by $R(N, M_0)$

the set of all markings reachable from the initial one. We also use $x_1p_1 + \cdots + x_np_n$ to denote the marking $[x_1, \ldots, x_n]^T$ for simplicity.

A transition t is *enabled* at M if $M \geq Pre(\cdot, t)$ and may fire reaching a new marking $M' = M_0 + C(\cdot, t)$. We write $M[\sigma\rangle$ to denote that the sequence of transitions σ is enabled at M, and we write $M[\sigma\rangle M'$ to denote that the firing of σ yields M'.

The vector $\mathbf{y}_\sigma$ is the Parikh vector of $\sigma \in T^*$, i.e., $y_\sigma(t) = k$ if transition t occurs k times in σ.

Given a net $N = (P, T, Pre, Post)$ we say that $\hat{N} = (\hat{P}, \hat{T}, \hat{Pre}, \hat{Post})$ is a subnet of N if $\hat{P} \subset P$, $\hat{T} \subset T$ and $\hat{Pre}$ (resp., $\hat{Post}$) is the restriction of Pre (resp., $Post$) to $\hat{P} \times \hat{T}$.

2.2 GMECs

A *Generalized Mutual Exclusion Constraint* (GMEC) is a pair $(\mathbf{w}, k)$ where $\mathbf{w} \in \mathbb{Z}^m$ and $k \in \mathbb{N}$. A GMEC defines a set of legal markings:

$$\mathscr{L}(\mathbf{w}, k) = \{M \in \mathbb{N}^m \mid \mathbf{w}^T \cdot M \leq k\}$$

An *AND-GMEC* is a pair $(\mathbf{W}, \mathbf{k})$, where $\mathbf{W} = [\mathbf{w}_1 \ \cdots \ \mathbf{w}_s] \in \mathbb{Z}^{m \times s}$ and $\mathbf{k} = [k_1 \ \cdots \ k_s]^T \in \mathbb{N}^s$. An AND GMEC defines a set of legal markings $\mathscr{L}(\mathbf{W}, \mathbf{k}) = \{M \in \mathbb{N}^m \mid \mathbf{W}^T \cdot M \leq \mathbf{k}\}$.

An *OR-AND GMEC* is a set $W = \{(\mathbf{W}_1, \mathbf{k}_1), \ldots, (\mathbf{W}_r, \mathbf{k}_r)\}$, where each $(\mathbf{W}_i, \mathbf{k}_i)$ is an AND-GMEC for $i = 1, \ldots, r$. The OR-AND GMEC W defines a set of legal markings

$$\mathscr{L}_W = \bigcup_{i=1}^{r} \mathscr{L}(\mathbf{W}_i, \mathbf{k}_i).$$

2.3 Labelled Petri Nets and Their Products

A *labeled Petri net* is a four-tuple $\langle N, M_0, E, l\rangle$ where $\langle N, M_0\rangle$ is a marked net, E is an *alphabet* and $l : T \to E$ is a *labeling function* that associates a label in E to each transition, i.e., $l(t) = e$. In this paper we also write "$t(e)$" to denote $l(t) = e$ for simplicity.

For $\sigma = t_1 t_2 \cdots t_x$ and $l(t_i) = e_i$, we define $w = l(\sigma) = l(t_1)l(t_2)\cdots l(t_x) = e_1 e_2 \cdots e_x$. Given an event $e \in E$ (resp., a word $w \in E^*$), we write $M[e\rangle$ (resp., $M[w\rangle$) to denote that there exists t (resp., σ) such that $l(t) = e$ and $M[t\rangle$ (resp., $l(\sigma) = w$ and $M[\sigma\rangle$).

Given a labeled Petri net G, its *language* $L(G)$ is the set of strings generated by any firing sequence, i.e.,

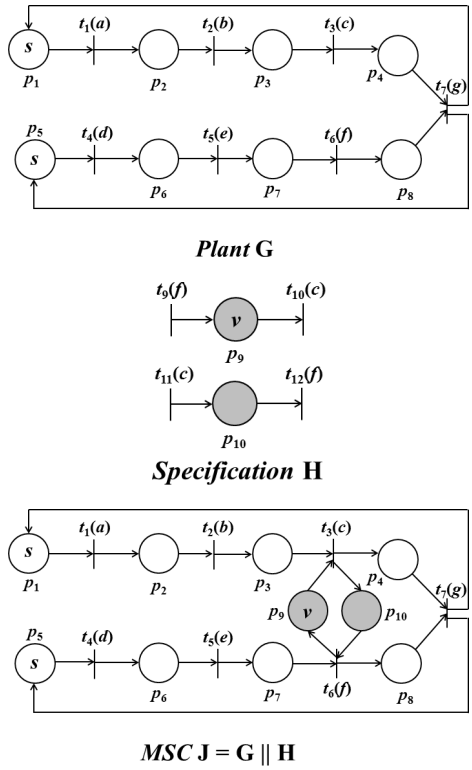

Figure 1: A plant G, a specification H and the MSC $J = G||H$.

$$L(G) = \{l(\sigma) \mid M_0[\sigma\rangle\}$$

A labelled Petri net G is *deterministic* if for all $t_1, t_2 \in T, t_1 \neq t_2$, and for all $M \in R(N, M_0)$, it holds: $(M[t_1\rangle \wedge M[t_2\rangle) \Rightarrow (l(t_1) \neq l(t_2))$.

The event set E can be partitioned into the sets of *controllable* events E_c and *uncontrollable* events E_u. This also induces a partition of the transition set T into the set of controllable transitions $T_c = \{t \in T \mid l(t) \in E_c\}$ and uncontrollable transitions $T_u = T \setminus T_c$.

Given two labelled Petri nets $G = \langle N', M_0', E', l' \rangle$ and $H = \langle N'', M_0'', E'', l'' \rangle$, the *product* of G and H is a labelled Petri net that fuses the transitions in G and H which share the same label. We give a simple example to illustrate the product operator in Example 1. More details can be found in the appendix of this paper and in [9].

3 Plant, Language Specification, and Monolithic Supervisor Candidate

In this section we first review some key notions on supervisory control using Petri nets.

A *plant*, i.e., a system to be controlled, is a deterministically labeled Petri net $G = \langle N', M'_0, E', l' \rangle$ with $N' = \langle P', T', Pre', Post' \rangle$. Given a plant G, a *language specification* on its behavior is described by a deterministically labeled net $H = \langle N'', M''_0, E'', l'' \rangle$ with $N'' = \langle P'', T'', Pre'', Post'' \rangle$ and $E'' \subseteq E'$. A specification H defines a set of legal firing sequences of G given by

$$\Gamma(G,H) = \{w \in L(G) \mid w_{\uparrow E''} \in L(H)\} \tag{1}$$

where $w_{\uparrow E''}$ denotes the natural projection of w on the alphabet E'' (i.e., $e_{\uparrow E''}$ is the string obtained by w removing all events not in E''). A *monolithic supervisor candidate* (MSC) is a labeled Petri net J the product of G and H, i.e., $J = G \parallel H$ (see Appendix).

Example 1 *Consider the Petri nets in Figure 1. The plant G contains two workstations which produce two types of parts to be assembled. Both workstations have a capacity s. Suppose in G events b,c,e,g are uncontrollable and events a,d,f are controllable. Now we want to apply a language specification H such that the production rate of the two stations should not differ too much: in any situations $t_3(c)$ should not fire more than v times than $t_6(f)$, and $t_6(f)$ should not fire more zero times than $t_3(c)$. The product of G and H is the MSC J where uncontrollable transitions $T_u = \{t_2,t_3,t_5,t_7\}$ and controllable transitions $T_c = \{t_1,t_4,t_6\}$.* □

A specification as well as the MSC may only disable controllable transitions to ensure that the controlled plant G only generates sequences in $\Gamma(G,H)$. However, during the evolution of the closed-loop system there may exist some markings at which an uncontrollable transition in the plant is about to fire, while the supervisor attempts to prevent it (i.e., the closed-loop control mode collapses at this moment). Typically to verify the controllability of H the MSC J has to be necessarily constructed and analyzed, since the language of J represents the set of legal words that G can generate. The specification H is uncontrollable if there exist some markings in the reachability set of J at which some uncontrollable transition is enabled in G but is not enabled in J (which implies it is not enabled in H). These markings are called *strongly uncontrollable markings*:

Definition 1 *Given an MSC $J = G||H$, its* strongly uncontrollable marking set *is:*

$$\mathscr{U} = \{M \in \mathbb{N}^m \mid \exists e_u \in E_u, M_{\uparrow G}[e_u\rangle \wedge \neg M_{\uparrow H}[e_u\rangle\} \tag{2}$$

□

For simplicity, in the sequel we use the term "uncontrollable" for "strongly uncontrollable".

Definition 2 *Given an MSC $J = G||H$, its* weakly uncontrollable marking set *is:*

$$\mathscr{W} = \{M \in \mathbb{N}^m \mid \exists \sigma_u \in T_u^*, M[\sigma_u\rangle \hat{M} \in \mathscr{U}\}$$

□

Definition 3 *Given an MSC $J = G||H$, its* controllable marking set *is defined as $\mathscr{C} = \mathbb{N}^m \setminus \mathscr{W}$.* □

We note that Definition 1 is equivalent to the definition of uncontrollable markings in [13]. From the definitions we immediately have $\mathscr{W} \supseteq \mathscr{U}$.

Definition 4 *An MSC $J = G||H$ is* controllable *(as well as H) if $R_J(N, M_0) \cap \mathscr{U} = \emptyset$. Otherwise J (as well as H) is* uncontrollable. □

Theorem 1 *[9, 13] The controllability of H as well as J is decidable.*

Theorem 2 *[9, 14] Given an uncontrollable bounded $J = \langle N, M_0\rangle = G||H$, it is always possible to find an OR-AND GMEC W such that $\mathscr{L}_W = \mathscr{C}$.*

From these two theorems, the controllability of a specification H (and of the MSC J) with respect to a plant G is always decidable. However, to verify the controllability of H is not easy. A straightforward method is to enumerate all reachable markings of the MSC (i.e., $R(N, M_0)$) to check if there exist any uncontrollable markings. However, even in a medium-sized system it may be impossible to construct its reachability graph due to the *state explosion*, and the complexity of such a controller may be up to $O(R(N, M_0))$ [9].

On the other hand, before the end of this section we propose an important result. The following theorem shows that a strongly uncontrollable marking set $\mathscr{U}$ of a monolithic supervisor candidate J can always be characterized by an OR-AND GMEC. Therefore checking if some markings in $\mathscr{U}$ are reachable from a given marking can be done by solving an integer programming problem. On this result we built the efficient method presented in the next section to solve the controllability checking problem. The method is based on the *basis reachability graph* analysis, without enumerating the whole state space of J.

Theorem 3 *Given a plant G and a language specification H, there always exists an OR-AND W on $J = G||H$ such that $\mathscr{L}_W = \mathscr{U}$.*

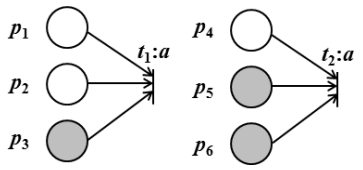

Figure 2: Illustration for Example 2.

Proof: According to Eq. (2), for any uncontrollable marking $M \in \mathscr{U}$, there exists an uncontrollable event e_u such that: $M_{\uparrow P'}[e_u\rangle, \neg M_{\uparrow P''}[e_u\rangle$. Therefore for a certain event $e_u \in E_u$, we can write a corresponding $\mathscr{U}(e_u)$ (which is a subset of $\mathscr{U}$) as:

$$\begin{cases} \exists t : l(t) = e_u, \forall p \in ({}^\bullet t \cap P'), M(p) \geq Pre(p,t) \\ \forall t : l(t) = e_u, \exists p \in ({}^\bullet t \cap P''), M(p) \leq Pre(p,t) - 1 \end{cases} \tag{3}$$

The set $\mathscr{U}(e_u)$ contains markings at which e_u is enabled in $J_{\uparrow G}$ but not enabled enabled $J_{\uparrow H}$. This set can always be characterized by an OR-AND GMEC $W(e_u)$ (see Example 2). Therefore the set $\mathscr{U}$ can be written as $\mathscr{U} = \bigcup_{e_u \in E_u} \mathscr{U}(e_u)$, and there always exists a $W = \bigvee_{e_u \in E_u} W(e_u)$, which can always be equivalently transformed into a standard OR-AND form. ■

Example 2 *Consider the labelled Petri net in Figure 2 as a part of an MSC, in which $p_1, p_2, p_4 \in P'$ (i.e., belong to G) and $p_3, p_5, p_6 \in P''$ (i.e., belong to H). For event a, the uncontrollable marking set $\mathscr{U}(a)$ can be characterized by the following linear inequalities:*

$$\begin{cases} [(M(p_1) \geq 1) \wedge (M(p_2) \geq 1) \vee (M(p_4) \geq 1)] \\ \quad \wedge [(M(p_3) \leq 0) \wedge ((M(p_5) \leq 0) \vee (M(p_6) \leq 0))] \end{cases} \tag{4}$$

which can be automatedly transformed into its equivalent standard disjunctive normal form*:*

$$\bigvee \begin{cases} (M(p_1) \geq 1) \wedge (M(p_2) \geq 1) \wedge (M(p_3) \leq 1) \\ \qquad\qquad\qquad\qquad \wedge (M(p_5) \leq 1) \\ (M(p_1) \geq 1) \wedge (M(p_2) \geq 1) \wedge (M(p_3) \leq 1) \\ \qquad\qquad\qquad\qquad \wedge (M(p_6) \leq 1) \\ (M(p_4) \geq 1) \wedge (M(p_3) \leq 1) \wedge (M(p_5) \leq 1) \\ (M(p_4) \geq 1) \wedge (M(p_3) \leq 1) \wedge (M(p_6) \leq 1) \end{cases} \tag{5}$$

To verify if such an OR-AND GMEC is feasible, one could either solve an IPP for each conjunctive parts or use the method in [15] to convert it into an equivalent conjunction linear integer constraints in polynomial time. □

4 Verifying Controllability by Basis Marking Analysis

In the work of Cabasino et al. [10, 11], a compact way to represent the reachability set of a Petri net is proposed to solve the diagnosis problems. The transition set T is partitioned into the observable transition set T_o and the unobservable transition set T_{uo}, since solving a diagnosis problem requires to reconstruct the firing sequences of the unobservable transitions from the observable firings. In [10, 11] only a subset of the reachable markings, called *basis markings*, are computed, and a non-deterministic finite state automaton called *basis reachability graph* (BRG) is constructed. All non-basis markings and the unobservable firing sequences can be characterized by the expansion of a set of linear equalities depending on those basis markings.

Up to now, the BRG approach has only been used to solve diagnosability problems and the related opacity problems [16, 17]. However, we point out that this basis marking analysis can be efficiently generalized. In general, the BRG-based approach can be applied if (1) the firing information of a certain set of transitions, say $T_x \subseteq T$, can be abstracted with respect to the problem we study, and (2) the subnet of T_x (i.e., $T_x = (P, T_x, Pre_x, Post_x)$ where Pre_x and $Post_x$ are Pre and $Post$ matrices reduced to $P \times T_x$, respectively) is acyclic. For instance, in the original diagnosability issue the transition set T is partitioned into observable transitions T_o and unobservable transitions T_{uo}, the firing of the latter can be abstracted. Here we partition T into controllable transitions T_c and uncontrollable ones T_u.

Definition 5 *Given a marking M and a controllable transition $t : l(t) \in E_c$, we define*

$$\Sigma(M,t) = \{\sigma \in T_u^* \mid M[\sigma\rangle M_x, M_x \geq Pre(\cdot,t)\}$$

the set of explanations *of t at M, and we define $Y(M,t) = \{\boldsymbol{y}_\sigma \mid \sigma \in \Sigma(M,t)\}$ the set of* explanation vectors*, i.e., to enable t some $\sigma \in \Sigma(M,t)$ must fire.* □

Definition 6 *Given a marking M and a controllable transition $t : l(t) \in E$, we define*

$$\Sigma_{min}(M,t) = \{\sigma \in \Sigma(M,t) \mid \nexists \sigma' \in \Sigma(M,t) : \boldsymbol{y}_{\sigma'} \lneq \boldsymbol{y}_\sigma\}$$

the set of minimal explanations *of t at M, and we define $Y_{min}(M,t) = \{\boldsymbol{y}_\sigma \mid \sigma \in \Sigma_{min}(M,t)\}$ the corresponding set of* minimal explanation vectors*.* □

The definitions of *explanations* and *explanation vectors* here are different from their definitions first proposed in [10, 11], in which the transitions are classified as observable/unobservable. By using the method in [11] to compute $Y_{min}(M,t)$ only algebraic manipulations are required. We also note that typically there exists more than one minimal explanation in $\Sigma_{min}(M,t)$.

In the following we present an algorithm to construct the *Basis Reachability Graph* (BRG) of a given Petri net. Although we also use the name BRG for consistency, we note that the BRG we constructe here is not exactly the same as that in [10], in which many definitions are tailored to reconstruct the firings of unobservable transitions for diagnosability problems.

Algorithm 1 BRG for Controllability

Input: A bounded labelled net $G = (N, M_0, E, l)$
Output: The BRG $\mathscr{B}_G = (\mathscr{M}, E, \Delta, M_0)$

1: $\mathscr{M} := \{M_0\}$ and assign no tag to M_0;
2: **while** state with no tag exist, **do**
3: select a state $M \in \mathscr{M}$ with no tag;
4: **for all** t such that $l(t) \in E_c$ and $Y_{min}(M,t) \neq \emptyset$, **do**
5: **for all** $\mathbf{y}_u \in Y_{min}(M,t)$, **do**
6: $\hat{M} := M + C \cdot \mathbf{y}_u + C(\cdot, t_u)$;
7: If $\nexists \hat{M} \in \mathscr{M}$, then $\mathscr{M} := \mathscr{M} \cup \{\hat{M}\}$ and assign no tag to $\hat{M}$;
8: $\Delta := \Delta \cup (M, l(t), \hat{M})$;
9: **end for**
10: tag the node M as "old";
11: **end for**
12: **end while**
13: Remove all tags.

Definition 7 *Give a bounded labelled net $J = (N, M_0, E, l)$, its* basis reachability graph *(BRG) is a non-deterministic finite state automaton $\mathscr{B}_J$ output by Algorithm 1. The BRG $\mathscr{B}_J$ is a four-tuple $(\mathscr{M}, E, \Delta, M_0)$, where:*

- *$\mathscr{M}$ is the state set containing a group of markings, these markings are called* basis markings*;*
- *$\Delta \subseteq (\mathscr{M} \times E \times \mathscr{M})$ is the transition relation;*
- *E and M_0 are the alphabet and the initial marking of J, respectively.*

□

Example 3 (Ex. 1 Continued) *Consider the MSC J in Figure 1 which is the product of G and H with $s = 2$ and $v = 1$. This MSC has 67 reachable markings. It is too large and we do not present it. By applying Algorithm 1 we obtain its BRG in Figure 3 which only contains 14 basis markings in $\mathscr{M}$.* □

By the construction procedure in Algorithm 1, one can verify that $(M_1[e\rangle M_2) \Rightarrow ((M_1, l(t), M_2) \in \Delta$. An important property of the basis marking set $\mathscr{M}$ is:

Theorem 4 *Given a labelled net $J = (N, M_0, E, l)$ in which the uncontrollable subnet is acyclic: (1) for any $M \in R(N, M_0)$, there exists an $M_b \in \mathscr{M}$ such that $M_b + C \cdot \mathbf{y}_{\sigma_u} = M$, where $\sigma_u \in T_u^*$; (2) for any basis marking $M_b \in \mathscr{M}$ and any marking M such that $M_b + C \cdot \mathbf{y}_{\sigma_u} = M$, it holds $M \in R(N, M_0)$.*

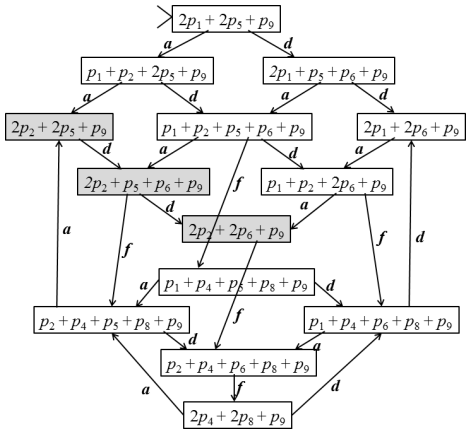

Figure 3: The BRG $\mathscr{B}_J$ of the MSC J in Figure 1. This BRG contains 14 basis markings while the reachability graph of J has 67 markings. Shaded markings are uncontrollable basis markings.

Proof: This theorem follows from a theorem in [10] (in which it is Theorem 3.8), by changing observable/unobservable transitions as controllable/uncontrollable ones. ■

By Theorem 4, any reachable marking must be reachable from some basis marking in $\mathscr{M}$ by firing only uncontrollable transitions, and *vice versa*.

Definition 8 *Given an MSC $J = G||H$, a basis marking $M_b \in \mathscr{M}$ is call an* uncontrollable basis marking *if $M_b \in \mathscr{W}$.* □

The following proposition can be used to determine if a basis marking is uncontrollable. We note that since there always exists an OR-AND GMEC W such that $\mathscr{L}_W = \mathscr{U}$ by Theorem 3, the first constraint in (6) can be given as a set of $\leq$ inequalities.

Proposition 1 *Given an MSC $J = G||H$, a basis marking $M_b \in \mathscr{W}$ if and only if the following integer programming problem (IPP) has a feasible solution:*

$$\begin{cases} \min & \boldsymbol{1}^T \cdot \boldsymbol{y}_u \\ s.t. & M_b + C \cdot \boldsymbol{y}_u \in \mathscr{U} \\ & \boldsymbol{y}_u \geq \boldsymbol{0} \end{cases} \tag{6}$$

Proof: If the IPP (6) has a feasible solution $\mathbf{y}_u$, since T_u-induced subnet is acyclic, there must exist a $\sigma_u \in T_u^*$ whose Parikh vector is $\mathbf{y}_u$ and $M_b[\sigma_u\rangle M_x \in \mathscr{U}$. Therefore $M_b \in \mathscr{W}$. On the other hand, if $M_b \in \mathscr{W}$, from the definition of weakly controllability there exists $M_x \in \mathscr{U}$ and M_x is reachable from M_b by firing $\sigma_u \in T_u$, therefore the IPP has a feasible solution. ■

Now we present the main result of this section.

Theorem 5 *Given a plant G, a specification H and the MSC $J = (N, M_0, E, l)$ such that $J = G||H$ and the uncontrollable subnet is acyclic. H (as well as J) is controllable if and only if all basis markings are controllable.*

Proof: (If) Suppose there exists $M \in R_J(N, M_0) \cap \mathscr{U}$, i.e. J is uncontrollable. By Theorem 4 part 1, there must exist a basis marking M_b such that $M_b + C \cdot \mathbf{y}_{\sigma_u} = M$. Since the uncontrollable subnet is acyclic, M is reachable from M_b. This indicates that $M_b \in \mathscr{W}$.

(Only If) Since $\mathscr{M} \subseteq R_J(N, M_0)$, if there exists $M_b \in \mathscr{W}$, obviously $R_J(N, M_0) \cap \mathscr{W} \neq \emptyset$. ■

By Theorem 5, to verify the controllability of J, instead of checking all reachable markings in the reachability graph, we only need to verify if all the basis markings are controllable. Now we can present an algorithm to determine the controllability of a give MSC J.

Algorithm 2 Determine the Controllability of an MSC

Input: A bounded plant G, a bounded specification H
Output: YES (controllable)/NO (uncontrollable)

```
Compute J = G||H,
Compute W according to Theorem 3,
Compute the BRG B_J of J by Algorithm 1,
for all M_b in M, do
    if IPP (6) has a feasible solution, then
        Tag M_b as "Uncontrollable";
    else
        Tag M_b as "Controllable";
    end if
end for
if M_b ∈ M with "Uncontrollable" tag exists, then
    Output NO and Exit;
else
    Output YES and Exit.
end if
```

In Algorithm 2 for a given MSC J we construct its BRG $\mathscr{B}_J$ and then verify that from each basis marking if some uncontrollable marking in $\mathscr{U}$ is reachable by solving an IPP. If there exists some basis marking in $\mathscr{M}$ from which some uncontrollable markings in $\mathscr{U}$ can be reached, it implies that the MSC may evolve to such uncontrollable markings so that the specification H is uncontrollable. Otherwise H is controllable.

Example 4 (Ex. 3 Continued) *Consider again the MSC $J = G||H$ in Figure 1 with $s = 2, v = 1$ and its BRG in Figure 3. The uncontrollable marking set $\mathscr{U}$ can be written as: $(M(p_3) \geq 1) \wedge (M(p_9) = 0)$. By applying*

Algorithm 2, for each basis marking M_b in $\mathscr{M}$ we solve the following IPP:

$$\begin{cases} \min & \boldsymbol{I}^T \cdot \boldsymbol{y}_u \\ s.t. & M_b(p_3) + C(p_3,\cdot)\cdot \boldsymbol{y}_u \geq 1 \\ & M_b(p_9) + C(p_9,\cdot)\cdot \boldsymbol{y}_u = 0 \\ & M_b + C\cdot \boldsymbol{y}_u \geq \boldsymbol{0} \\ & \boldsymbol{y}_u \geq \boldsymbol{0} \end{cases} \tag{7}$$

We find that there are three basis markings (located in shaded boxes) for which (6) *has feasible solutions. From Theorem 5 it indicates that H is uncontrollable. For instance, for $\tilde{M}_b = 2p_2 + 2p_5 + p_9$ the corresponding IPP has a feasible solution $\boldsymbol{y}_u$ that is: $y_u(t_2) = 2, y_u(t_3) = 1, y_u(t_1) = y_u(t_4) = y_u(t_5) = y_u(t_6) = y_u(t_7) = 0$, and there exists a reachable marking $M = \hat{M}_b + C\cdot \boldsymbol{y}_u = p_3 + p_4 + 2p_5 + p_{10}$ which is reachable and in $\mathscr{U}$.*

On the other hand, consider a new specification $\bar{H}$ with $s = 2, v = 2$. The new MSC $\bar{J}$ will have the same structure as J in Figure 1 except $s = 2, v = 2$. The resulting BRG $\bar{\mathscr{B}}_J$ is isomorphic to the one in Figure 3 but there will be one more token in p_9 for all basis markings in $\bar{\mathscr{M}}$. For the limit of space we do not present a figure to show this. One can verify that by Algorithm 2 for all basis markings in $\bar{\mathscr{M}}$ the IPP (6) *does not have a feasible solution. Therefore the new specification $\bar{H}$ with $v = 2$ is controllable.* □

Let us briefly discuss the complexity of the BRG-based method. In the worst case the BRG may have the same size as its reachability graph, i.e., $|\mathscr{M}| = |R(N, M_0)|$. However, in most cases the BRG is much smaller than the reachability graph, i.e., $|\mathscr{M}| \ll |R(N, M_0)|$. Therefore Algorithm 2 can be used to efficiently check the controllability of a language specification in Petri nets, as shown in the following example.

Example 5 (Ex. 4 Continued) *Still consider the MSC $J = G||H$ in Figure 1. For different parameters s and v the performances of the classical reachability graph analysis and our approach by Algorithm 2 are listed in Table 1. The simulation is done on a workstation with Core-II 2.2G CPU, using the standard Matlab toolboxes.*

From Tabel 1 one can see that with the increase of s and v, the size of the reachability graph ($|R(N, M_0)|$) grows much faster than that of the BRG ($|\mathscr{M}|$). Hence the reachability-graph-based algorithm takes a much longer time to determine the controllability of J than using Algorithm 2. Moreover, when $s, v > 8$ the reachability graph analysis is not possible to be applied, while the BRG apporach, i.e., Algorithm 2, is capable for even larger systems (for $s = v = 30$ the cardinality of $R(N, M_0)$ is approximately 10^9). □

At the end of this section we want to make two comments concerning our approach.

First, this approach is presented for nets that satisfy the following assumption: the uncontrollable subnet

Table 1: Performance of Algorithm 2.

s	v	$\lvert R(N,M_0)\rvert$	Time(s)[1]	$\lvert\mathscr{M}\rvert$	Time(s)	Result[2]
2	1	67	<1	14	<1	No
2	2	73	<1	14	<1	Yes
4	3	783	7	55	<1	No
4	4	798	7	55	<1	Yes
6	5	4298	273	140	<1	No
6	6	4326	251	140	<1	Yes
8	7	16026	3816	285	2	No
8	8	16071	4123	285	2	Yes
10	9	-	o.t.	506	4	No
10	10	-	o.t.	506	4	Yes
20	19	-	o.t.	3311	97	No
20	20	-	o.t.	3311	91	Yes
30	29	-	o.t.	10416	830	No
30	30	-	o.t.	10416	774	Yes

[1]We define *overtime* (o.t.) if the Matlab program does not halt within 8 hours.
[2]"Yes" and "No" indicate that the language specification (i.e., the corresponding MSC) is controllable and uncontrollable, respectively.

of the MSC is acyclic. However, we are also working at removing this assumption. In fact, if the uncontrollable subnet is not acyclic, we consider a new partition $T = T_u' \cup T_c'$ where $T_u' \subset T_u$ — this means that some uncontrollable transitions that create cycles in the uncontrollable nets are now treated as controllable so that the T_u'-induced subnet is acyclic. In such a case a modified BRG can be constructed and all the above results holds. However, in such a case the number of basis markings will increase. The scalability of this approach is an open issue for our future research.

Second, in the supervisory control framework, if H is uncontrollable, then J must be further refined to determine a supervisor. Typically this step requires reachability graph. However, the BRG can be used to design an on-line supervisor, since the BRG the transition function Δ contains all controllable transitions.

5 Conclusion

In this paper an effective method based on basis marking analysis to determine and enhance the controllability of a given language specification is proposed. By constructing the basis reachability graph, the controllability of the given specification can be verified by solving a series of integer programming problems.

References

[1] P. J. Ramadge and W. M. Wonham, "The control of discrete event systems," *Proceedings of IEEE*, vol. 77, no. 1, pp. 81–98, 1989.

[2] C. G. Cassandras and S. Lafortune, *Introduction to discrete event systems*. Springer, 2008.

[3] A. Giua, F. DiCesare, and M. Silva, "Generalized mutual exclusion constraints for Petri nets with uncontrollable transitions," in *In Proceedings of the IEEE Int. Conf. on Systems, Man, and Cybernetics*, Chicago, USA, 1992, pp. 947–949.

[4] J. Moody and P. Antsaklis, "Petri net supervisors for DES with uncontrollable and unobservable transitions," *IEEE Transactions on Automatic Control*, vol. 45, no. 3, pp. 462–476, 2000.

[5] F. Basile, R. Cordone, and L. Piroddi, "Integrated design of optimal supervisors for the enforcement of static and behavioral specifications in Petri net models," *Automatica*, vol. 49, no. 11, pp. 3432–3439, 2013.

[6] J. L. Luo, W. M. Wu, H. Y. Su, and J. Chu, "Supervisor synthesis for enforcing a class of generalized mutual exclusion constraints on Petri nets," *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, vol. 39, no. 6, pp. 1237–1246, 2009.

[7] M. V. Iordache and P. J. Antsaklis, "Petri net supervisors for disjunctive constraints," in *In Proceedings of the 26th American Control Conference*, New York, USA, 2007, pp. 4951–4956.

[8] M. V. Iordache, P. Wu, F. Zhu, and P. J. Antsaklis, "Efficient design of Petri-net supervisors with disjunctive specifications," in *In Proceedings of the IEEE Int. Conf. on Automation Science and Engineering*, Madison, USA, 2013, pp. 936–941.

[9] A. Giua, "Supervisory control of Petri nets with language specifications," in *Control of discrete-event systems*, C. Seatzu, M. Silva, and J. van Schuppen, Eds. London: Springer, 2013, vol. 433, pp. 235–255.

[10] M. Cabasino, A. Giua, and C. Seatzu., "Fault detection for discrete event systems using Petri nets with unobservable transitions," *Automatica*, vol. 46, no. 9, pp. 1531–1539, 2010.

[11] M. Cabasino, A. Giua, M. Pocci, and C. Seatzu, "Discrete event diagnosis using labeled Petri nets: an application to manufacturing systems," *Control Engineering Practice*, vol. 19, no. 9, pp. 989–1001, 2011.

[12] Z. Y. Ma, Z. W. Li, and A. Giua, "Design of optimal Petri net controllers for disjunctive generalized mutual exclusion constraints," *IEEE Transactions on Automatic Control*, vol. 60, pp. 1774–1785, 2015.

[13] B. Lacerda and P. U. Lima, "On the notion of uncontrollable marking in supervisory control of Petri nets," *IEEE Transactions on Automatic Control*, vol. 59, no. 11, pp. 53–61, 2014.

[14] Z. Y. Ma, Z. W. Li, and A. Giua, "Linear algebraic characterization of legal markings for Petri net supervisors," in *In Proceedings of the 14th IFAC World Congress*, Capetown, South Africa, 2014, pp. 2429–2434.

[15] M. Cabasino, A. Giua, and C. Seatzu, "Identification of Petri nets from knowledge of their language," *Discrete Event Dynamic Systems*, vol. 17, no. 4, pp. 447–474, 2007.

[16] Y. Tong, Z. W. Li, C. Seatzu, and A. Giua, "Verification of current-state opacity using Petri nets," in *In Proceedings of the 34th American Control Conference*, Chicago, USA, 2015, pp. 1935–1940.

[17] ——, "Verification of initial-state opacity in DES," in *In Proceedings of the 54th IEEE Conf. on Decision and Control*, Osaka, Japan, 2015, pp. 344–349.

Appendix

Definition 9 (Concurrent composition of languages) *Given two languages $L_1 \subseteq E_1^*$, $L_2 \subseteq E_2^*$ and $E = E_1 \cup E_2$, their* concurrent composition *is the language L:*

$$L = L_1 || L_2 = \{w \in E^* \mid w_{\uparrow E_1} \in L_1, w_{\uparrow E_1} \in L_2\}.$$

This operator has a counterpart on the transition structure of Petri nets. In fact, given two labelled Petri nets G', G'', the following algorithm constructs their *product* $G = G' \parallel G''$. The net G generates the language $L(G) = L(G')||L(G'')$ concurrent composition of the languages of G' and G''.

Here λ denotes the empty sequence and is used to denote that a transition in G' (resp., G'') is not synchronized with a transition in G'' (resp., G'). The transitions of the form (t', λ) or (λ, t'') are usually simply denoted t' or t''. If for a plant transition t' there exists only one $(t', \cdot)$ in G, $(t', \cdot)$ is also denoted as t' for simplicity.

Algorithm 3 Product of G' and G''

Input: $G' = (N', M_0', E', l'), G'' = (N'', M_0'', E'', l'')$

Output: $G = (N, M_0, E, l)$ such that $L(G) = L(G')||L(G'')$

1: Let $P = P' \cup P''$;

2: Let $T = \{(t', t'') \mid t' \in T', t'' \in T'', l'(t') = l''(t'')\}$

$$\begin{array}{l} \cup \{(t', \lambda) \mid t' \in T', (\nexists t'' \in T'') l'(t') = l''(t'')\} \\ \cup \{(\lambda, t'') \mid t'' \in T'', (\nexists t' \in T') l'(t') = l''(t'')\} \end{array} \tag{8}$$

3: Let $Pre(p,t) =$

$$\begin{cases} Pre'(p,t') & \text{if } p \in P', t' \neq \lambda, t = (t', t'') \\ Pre''(p,t'') & \text{if } p \in P'', t'' \neq \lambda, t = (t', t'') \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

4: Let $Post(p,t) =$

$$\begin{cases} Post'(p,t') & \text{if } p \in P', t' \neq \lambda, t = (t', t'') \\ Post''(p,t'') & \text{if } p \in P'', t'' \neq \lambda, t = (t', t'') \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

5: Let $E = E' \cup E''$;

6: Let $l((t', t'')) = l'(t')$ if $t' \neq \lambda$, else $l((t', t'')) = l''(t'')$;

7: Let $M_0 = (M_0', M_0'')$.