Model abstraction of nondeterministic finite-state automata in supervisor synthesis

Citation for published version (APA):

DOI:
10.1109/TAC.2010.2046931

Document status and date:
Published: 01/01/2010

Document Version:
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:
• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain.
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license above, please follow below link for the End User Agreement:
www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:
openaccess@tue.nl
providing details and we will investigate your claim.

Download date: 30. Mar. 2021
Model Abstraction of Nondeterministic Finite-State Automata in Supervisor Synthesis

Rong Su, Jan H. van Schuppen, Member, IEEE, and Jacobus E. Rooda, Member, IEEE

Abstract—Blockingness is one of the major obstacles that need to be overcome in the Ramadge-Wonham supervisory synthesis paradigm, especially for large systems. In this paper, we propose an abstraction technique to overcome this difficulty. We first provide details of this abstraction technique, then describe how it can be applied to a supervisor synthesis problem, where plant models are nondeterministic but specifications and supervisors are deterministic. We show that a nonblocking supervisor for an abstraction of a plant under a specification is guaranteed to be a nonblocking supervisor of the original plant under the same specification. The reverse statement is also true, if we impose an additional constraint in the choice of the alphabet of abstraction, i.e., every event, which is either observable or labels a transition to a marker state, is contained in the alphabet of abstraction.

Index Terms—Automaton abstraction, discrete-event systems, nondeterministic finite-state automata, supervisor synthesis.

I. INTRODUCTION

The automaton-based Ramadge-Wonham (RW) supervisory control paradigm first appeared in the control literature in 1982, which was subsequently summarized in the well known journal papers [18], [26]. Since then there has been a large volume of literature under the same paradigm. In the RW paradigm one of the main problems is to synthesize a supervisor for a plant such that the closed-loop behavior is nonblocking, controllable [18], observable or normal [11], and satisfies some prescribed requirements. The main difficulty of supervisor synthesis is to achieve nonblockingness because the total number of states of a plant model increases quickly when the number of local components increases, due to the synchronous product which incurs Cartesian product over automata. To overcome this difficulty, some authors attempt to introduce sufficient conditions which allow local supervisor synthesis. For example, in [27] the authors propose the concept of modularity, which is then extended to the concept of local modularity in [17]. When local supervisors are (locally) modular, a globally non-blocking supervisory control is achieved. Nevertheless, testing (local) modularity itself usually imposes prohibitive computational complexity. Another notable work is presented in [10], where, by imposing interface consistency and level-wise controllability among subsystems and local supervisors in a hierarchical setup, a very large nonblocking control problem may be solved, e.g. the size of the state set reaches $10^{21}$ in the Atelier Interétabilissement de Productique (AIP) example [10]. But the approach does not tell how to deliberately and systematically design interfaces that allow synthesis of local supervisors that satisfy those properties. Instead, it assumes that those interfaces are given before synthesis, as mentioned in [9]. In [12] the authors present an interesting approach, which is aimed at synthesizing a state-feedback supervisor. The authors represent product states as state tree structures, upon which the power of symbolic computation (as manifested by the manipulation of binary decision diagrams) is fully utilized. It has been shown in [12] that a system with $10^{24}$ states can be accommodated. Nevertheless, this approach is essentially a centralized approach, and it does not deal with cases when only partial observations of states are available for control. In this paper we will discuss the usage of abstraction to reduce complexity in synthesizing nonblocking supervisors, where partial observation may be present.

Our first contribution is to present a novel automaton-based abstraction technique. The idea of abstraction has been known in the literature, e.g. in [2] abstraction is used in the modular and hierarchical supervisor synthesis; it is also used in [16] for testing the nonblocking property, and in [19] for decentralized control. Nevertheless, their approaches are language-based, and rely on natural projections that satisfy the observer property [23]. Although a natural projection can always be modified to become an observer (with respect to a specific language) [24], such a modification has a potential drawback in the sense that the alphabet of the codomain of the projection may be fairly large for the sake of achieving the observer property, and the consequence is that the size of the projected image may not be small enough to allow supervisor synthesis for large systems. Our abstraction technique is automaton-based, which computes an abstraction for any pre-specified abstraction alphabet, and guarantees that the abstraction is suitable for supervisor synthesis. Thus, the drawback of the language-based abstraction techniques is avoided in our approach. Several strategies for automaton abstraction have been proposed, e.g., in [4], [5], [7], [13], [22]. Among them, [22] aims to achieve weak bisimilarity between an automaton and its abstraction. In [4], [5], [7], [13] the authors first use special events, which are called silent events and usually denoted by $\tau$, or $\tau_e$ and $\tau_u$ when distinguishing controllable and uncontrollable events is necessary, to replace internal events that are not in the abstraction alphabet.

Manuscript received May 19, 2008; revised October 24, 2008; accepted March 22, 2010. First published April 05, 2010; current version published November 03, 2010. Recommended by Associate Editor E. Fabre.

R. Su is with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, 639798 (email: rsu@ntu.edu.sg).

J. H. van Schuppen is with Centrum voor Wiskunde en Informatica (CWI), Amsterdam 1090 GB, The Netherlands (e-mail: j.h.van.schuppen@cwi.nl).

J. E. Rooda is with the Systems Engineering Group, Department of Mechanical Engineering, Eindhoven University of Technology, Eindhoven 5600 MB, The Netherlands (e-mail: j.e.rooda@tue.nl).

Digital Object Identifier 10.1109/TAC.2010.2046931
Then they apply heuristic rewriting rules to ensure that appropriate equivalence relations hold between automata before and after rewriting, e.g., conflict equivalence in [4], [7], supervision equivalence in [5] and synthesis equivalence in [13]. The primary goal of our approach is to create an abstraction for an automaton \( G \), which is not necessarily weak bisimilar to \( G \), such that any automaton \( S \), whose alphabet is the same as that of the abstraction and is nonconflicting with the abstraction, must be nonconflicting with \( G \). If we impose an additional constraint in the choice of the alphabet of abstraction, then it is also true that \( S \) is nonconflicting with \( G \) implies that \( S \) is nonconflicting with the abstraction—at this point, our approach is close to achieving conflict equivalence, but it does not require silent events and heuristic rewriting rules.

Our second contribution is to show how the proposed abstraction technique can be applied to a synthesis problem, where the plant model is nondeterministic but the specification and the supervisor are deterministic. There exists a large body of publications on supervisor synthesis for nondeterministic systems. For example, in [1] both plant and supervisor models are nondeterministic and different types of deterministic or nondeterministic specifications are considered. In [6], [8] the plant is considered to be nondeterministic and both the specification and the supervisor are deterministic. In [15] the plant and the specification are nondeterministic but the supervisor is deterministic. In [28], [29] the plant and the specification are nondeterministic and the supervisor can also be nondeterministic. The main difference between these papers and ours is that, we focus on how to use automaton abstraction in synthesis to reduce computational complexity. We consider a nondeterministic plant because an abstraction of a deterministic plant is usually nondeterministic. We consider a deterministic specification and a deterministic supervisor because they are typical in industrial systems, and they allow automaton abstraction to be used in synthesis. We are still investigating whether the proposed abstraction technique is also applicable to cases with nondeterministic requirements and supervisors. Although [5], [7], [13], [22] also utilize abstraction in synthesis, their abstraction techniques are different from ours. Because the main objective of this paper is to establish a connection between the existence of a nonblocking supervisor for a plant model and the existence of a nonblocking supervisor for an abstract model created by our abstraction technique, details of how to synthesize a nonblocking supervisor based on nondeterministic finite-state automata are not mentioned in this paper, but addressed in [21]. We also introduce the concept of state normality, which allows for the computation of a supremal nonblocking state-normal supervisor for a nondeterministic system.

This paper is organized as follows. In Section II we introduce an abstraction technique over nondeterministic automata. In Section III we show the usage of the proposed abstraction technique in supervisor synthesis. After an illustrative example in Section IV, conclusions are stated in Section V. Long proofs are presented in the Appendix.

**II. AUTOMATON ABSTRACTION AND RELEVANT PROPERTIES**

In this section we follow the notations used in [25]. We first briefly review concepts related to languages and automata, then introduce the concept of automaton abstraction. After that, we present properties of abstraction which are used in supervisor synthesis.

**A. Concepts of Languages, Automata and Abstraction**

Let \( \Sigma \) be a finite alphabet, and \( \Sigma^* \) denote the Kleene closure of \( \Sigma \), i.e., the collection of all finite sequences of events taken from \( \Sigma \). Given two strings \( s, t \in \Sigma^* \), \( s \) is called a **prefix** sub-string of \( t \), written as \( s \preceq t \), if there exists \( s' \in \Sigma^* \) such that \( ss' = t \), where \( ss' \) denotes the concatenation of \( s \) and \( s' \). We use \( \epsilon \) to denote the empty string of \( \Sigma^* \) such that for any string \( s \in \Sigma^* \), \( \epsilon s = s = s \epsilon \). A subset \( L \subseteq \Sigma^* \) is called a **language**. \( L = \{ s \in \Sigma^* | \exists t \in L \text{ s.t. } s \preceq t \} \subseteq \Sigma^* \) is called the prefix closure of \( L \). \( L \) is called prefix closed if \( L = \overline{L} \). Given two languages \( L, L' \subseteq \Sigma^* \), let \( LL' := \{ ss' \in \Sigma^* | s \in L \& s' \in L' \} \) be the concatenation of \( L \) and \( L' \), which contains every string obtainable by concatenating one string from \( L \) and one string from \( L' \).

Let \( \Sigma' \subseteq \Sigma \). A mapping \( P : \Sigma^* \rightarrow \Sigma'^* \) is called the natural projection with respect to \( (\Sigma, \Sigma') \), if

1. \( P(\epsilon) = \epsilon \);
2. \( (\forall s \in \Sigma) P(\sigma) : \sigma \rightarrow \epsilon \) if \( s \in \Sigma' \);
3. \( (\forall s \in \Sigma^*) P(s\sigma) = P(s) P(\sigma) \).

Given a language \( L \subseteq \Sigma^* \), \( P(L) := \{ P(s) | s \in \Sigma^* | s \in L \} \). The inverse image mapping of \( P \) is \( P^{-1} : 2^\Sigma^* \rightarrow 2^\Sigma^* : L \mapsto P^{-1}(L) := \{ s \in \Sigma^* | P(s) \in L \} \).

Given \( L_1 \subseteq \Sigma_1^* \) and \( L_2 \subseteq \Sigma_2^* \), the synchronous product of \( L_1 \) and \( L_2 \) is defined as \( L_1 \| L_2 := P^{-1}(L_1) \cap P^{-1}(L_2) \), where \( P_1 : (\Sigma_1 \cup \Sigma_2)^* \rightarrow \Sigma_1^* \) and \( P_2 : (\Sigma_1 \cup \Sigma_2)^* \rightarrow \Sigma_2^* \) are natural projections. Clearly, \( \| \) is commutative and associative. Next, we introduce automaton product and abstraction.

A **nondeterministic finite-state automaton** is a 5-tuple \( \mathcal{A} = (X, \Sigma, \xi, x_0, X_m) \), where \( X \) stands for the state set, \( \Sigma \) for the alphabet, \( \xi : X \times \Sigma \rightarrow 2^X \) for the nondeterministic transition function, \( x_0 \) for the initial state and \( X_m \subseteq X \) for the marker state set. Usually, the domain of \( \xi \) is extended to \( X \times \Sigma^* \). If for all \( x \in X \) and \( \sigma \in \Sigma \), \( \xi(x, \sigma) \) contains no more than one element, then \( \mathcal{A} \) is called deterministic. Let \( B(G) := \{ s \in \Sigma^* | \exists \exists x \in \xi(x, s) \} \subseteq \Sigma^* \).

Any string \( s \in B(G) \) can lead to a state \( x \), from which no marker state is reachable, i.e. for any \( s' \in \Sigma^*, \xi(x, s') \cap X_m = \emptyset \). Such a state \( x \) is called a **blocking state** of \( G \), and we call \( B(G) \) the **blocking set** of \( G \). A state that is not a blocking state is called a **nonblocking state**. We say \( G \) is nonblocking if \( B(G) = \emptyset \).

For each \( x \in X \), we define another set \( N_G(x) := \{ s \in \Sigma^* | \xi(x, s) \cap X_m = \emptyset \} \), and call \( N_G(x_0) \) the **nonblocking set** of \( G \), which is simply the set of all strings recognized by \( G \). For the notation simplicity, we use \( N(G) \) to denote \( N_G(x_0) \). It is possible that \( B(G) \cap N(G) = \emptyset \), due to nondeterminism. Let \( L(G) := \{ s \in \Sigma^* | \xi(x_0, s) \neq \emptyset \} \) be the **closed behavior** of \( G \).

Given two nondeterministic automata \( G_1 \equiv (X_1, \Sigma_1, \xi_1, x_{1,0}, X_{1,m}, m = 1, 2) \), the product of \( G_1 \) and \( G_2 \), written as \( G_1 \times G_2 \), is an automaton such that \( G_1 \times G_2 = (X_1 \times X_2, \Sigma_1 \cup \Sigma_2, \xi_1 \times \xi_2, (x_{1,0}, x_{2,0}), X_{1,m} \times X_{2,m}) \).
where $\xi_1 \times \xi_2 : X_1 \times X_2 \times (\Sigma_1 \cup \Sigma_2) \to 2^{X_1 \times X_2}$ is defined as follows:

$$(\xi_1 \times \xi_2)((x_1, x_2), \sigma) := \begin{cases} \xi_1(x_1, \sigma) \times \{x_2\} & \text{if } \sigma \in \Sigma_1 - \Sigma_2 \\
\{x_1\} \times \xi_2(x_2, \sigma) & \text{if } \sigma \in \Sigma_2 - \Sigma_1 \\
\xi_1(x_1, \sigma) \times \xi_2(x_2, \sigma) & \text{if } \sigma \in \Sigma_1 \cap \Sigma_2. \end{cases}$$

Clearly, $\times$ is commutative and associative. $\xi_1 \times \xi_2$ is extended to $X_1 \times X_2 \times (\Sigma_1 \cup \Sigma_2)^* \to 2^{X_1 \times X_2}$. By a slight abuse of notations, from now on we use $G_1 \times G_2$ to denote its reachable part, which contains all states reachable from $(x_1,0,x_2,0)$ by $\xi_1 \times \xi_2$ and relevant transitions between each pair of these states. Next, we introduce automaton abstraction, which requires the following concept of marking weak bisimilarity.

Definition 1: Given $G = (X, \Sigma, \xi, x_0, M_m)$, let $\Sigma^* \subseteq \Sigma$ and $P : \Sigma^* \to \Sigma^*$ be the natural projection. A marking weak bisimulation relation on $X$ with respect to $\Sigma^*$ is an equivalence relation $R \subseteq \{(x, y) \in X \times X | x \in X_m \Leftrightarrow y \in X_m\}$ such that, for all $(x, y) \in R$, $s \in \Sigma^*$ and $y \in \xi(x, s)$,

$$(\exists s' \in \Sigma^*) P(s) = P(s') \wedge (\exists y' \in \xi(x', s')) (y, y') \in R.$$ 

The largest marking weak bisimulation relation on $X$ with respect to $\Sigma^*$ is called marking weak bisimilarity on $X$ with respect to $\Sigma^*$, written as $\approx_{\Sigma^* G}$.

Marking weak bisimilarity is almost the same as weak bisimilarity described in [14], except for the special treatment on marker states. We now introduce abstraction.

Definition 2: Given $G = (X, \Sigma, \xi, x_0, M_m)$, let $\Sigma^* \subseteq \Sigma$. The automaton abstraction of $G$ with respect to $\approx_{\Sigma^* G}$ is an automaton $G' = (\Sigma', \approx_{\Sigma^* G}', \xi', x_0, M_m')$ where

1) $Z := X/\approx_{\Sigma^* G} := \{[x] := \{x' \in X | (x, x') \in \approx_{\Sigma^* G}\} | x \in X\}$;
2) $z_0 := [x_0]$;
3) $Z_m := \{z \in Z | z \cap X_m \neq \emptyset\}$;
4) $\delta : Z \times \Sigma^* \to 2^{Z'}$, where for any $(z, \sigma) \in Z \times \Sigma^*$, $\delta(z, \sigma) := \{z' \in Z | (\exists x \in z) (\exists x' \in (\Sigma - \Sigma^*)^{\ast}) (x, x'u \sigma) \in \approx_{\Sigma^* G} \}.$

The total number of transitions in $G'$ is no more than $mn^2$, where $n = |X|$ and $m = |\Sigma^* \cup \{\nu\}|$. Based on a result in [3], the time complexity of computing $X/\approx_{\Sigma^* G}$ is $O(mn^2 \log n)$ if we ignore the complexity caused by checking the condition "$x \in X_m \Leftrightarrow x' \in X_m$" in Def. 1. If we consider this extra condition, which requires comparing at most $(1/2)m(n - 1)$ pairs of states in the worst case, then the overall complexity is $O((1/2)m(n - 1) + mn^2 \log n) = O(mn^2 \log n)$. From now on, when $G$ is clear from the context, we simply use $\approx_{\Sigma^*}$ to denote $\approx_{\Sigma^* G}$, and use $(x)^{\approx_{\Sigma^*}}$ for an element of $X/\approx_{\Sigma^* G}$. If $\Sigma^*$ is also clear from the context, then we simply use $(x)$ for $(x)^{\approx_{\Sigma^*}}$. In other comparable automaton-based abstraction techniques, e.g., [4], [7], [13], [22], the weak bisimilarity is also used, except that in their definition two equivalent states need not have the same marking status, which may potentially make the size of the quotient state set under their construction slightly smaller than the size of $X/\approx_{\Sigma^*}$. On the other hand, in those techniques the definition of $\delta$ utilizes the following standard quotient construction:

$$\delta([z], \sigma) := \begin{cases} \{z' \in Z | (\exists x \in z)(\exists x' \in z') x' \in \xi([x], \sigma) \} & \text{if } \sigma \neq \nu \\
\{z' \in Z | (\exists x \in z)(\exists x' \in z') \in \xi([x], \sigma') \} & \text{if } \sigma = \nu. \end{cases}$$

Our definition of $\delta$ is nonstandard in the sense that, two quotient states are only connected by events in $\Sigma^*$ and $\nu$ is not used. As a result of this nonstandard definition, two different quotient states in $G/\approx_{\Sigma^*}$ may become equivalent in $(G/\approx_{\Sigma^*})/\approx_{\Sigma^*}$, which usually makes $(G/\approx_{\Sigma^*})/\approx_{\Sigma^*}$ smaller than $G/\approx_{\Sigma^*}$.

In the next section we will see that, $G'/\approx_{\Sigma^*}$ can be replaced by $(G/\approx_{\Sigma^*})/\approx_{\Sigma^*}$ in supervisor synthesis. There exists a procedure that computes $(G/\approx_{\Sigma^*})/\approx_{\Sigma^*}$ directly from $G$ without applying the abstraction procedure twice, and the complexity of computing $(X/\approx_{\Sigma^*})/\approx_{\Sigma^*}$ is equal to $O(mn^2 \log n)$, where $m' = |\Sigma'| = m - 1$. Owing to the limited space, we will not discuss this procedure in this paper. As a comparison, we use $G/\approx_{\Sigma^*}$ to denote the standard quotient construction under the weak bisimilarity. Then $(G/\approx_{\Sigma^*})/\approx_{\Sigma^*}$ is the same as $G'/\approx_{\Sigma^*}$ (under automaton isomorphism), whose size is close to that of $G/\approx_{\Sigma^*}$. Thus, in practice our technique can obtain smaller abstractions than the standard quotient construction can achieve, which is illustrated in the following example.

Let $G = (X, \Sigma, \xi, x_0, M_m)$ be a nondeterministic automaton depicted in Fig. 1, where $\Sigma = \{a, b, u\}$. Assume $\Sigma^* = \{a, b\}$. Then we have the quotient state set $X/\approx_{\Sigma^*} = \{1\}, \{2\}, \{3\}, \{4, 5\}, \{5, 6\}, \{6\}, \{7\}, \{11\}, \{10\} = \{10\}$. The abstraction $G/\approx_{\Sigma^*}$ is depicted in Fig. 1. We can check that, in $G/\approx_{\Sigma^*}$ states $1$ and $3$ are equivalent under $\approx_{\Sigma^*}$, and so are $\{5\}$ and $\{6\}$. This happens because the transition map in our definition of abstraction is nonstandard, making the path from state 2 to the blocking state 7 (and from state 6 to state 11) disappears in $G/\approx_{\Sigma^*}$. The abstraction $(G/\approx_{\Sigma^*})/\approx_{\Sigma^*}$ is depicted in Fig. 1, where $(X/\approx_{\Sigma^*})/\approx_{\Sigma^*} = \{\{1\}, \{2\}, \{3\}, \{4\}, \{4, 5\}, \{5, 6\}, \{6\}, \{7\}, \{11\}, \{10\} = \{10\}\}$. As a comparison, we apply the standard quotient construction on $G$. To distinguish elements of $X/\approx_{\Sigma^*}$ from those of $X/\approx_{\Sigma^*}$, we use $\hat{x}$ for a quotient state under $\approx_{\Sigma^*}$. We have $X/\approx_{\Sigma^*} = \{1\}, \{2\}, \{3\}, \{4, 5\}, \{5, 6\}, \{6\}, \{7\}, \{11\}, \{10\} = \{10\}$. We can see that $X/\approx_{\Sigma^*}$ is a quotient set based on the weak bisimilarity. The quotient automaton $G/\approx_{\Sigma^*}$ is depicted in Fig. 1, which is different from $G/\approx_{\Sigma^*}$ and has more states and transitions than $(G/\approx_{\Sigma^*})/\approx_{\Sigma^*}$ has. In this example, we can see that our abstraction technique does enjoy some computational advantage over other automaton-based techniques.
abstraction techniques, which utilize the standard quotient construction. Next, we present properties of automaton abstraction.

**B. Properties of Automaton Abstraction**

We first introduce two more concepts, which are important for applying the aforementioned automaton abstraction in supervisor synthesis.

**Definition 3:** An automaton $G = (X, \Sigma, \xi, x_0, X_m)$ is marking aware with respect to $\Sigma'$, if
\[
(\forall x \in X \setminus X_m)(\forall s \in \Sigma^*)\xi(x, s) \cap X_m \neq \emptyset \Rightarrow P(s) \neq \emptyset
\]
where $P : \Sigma^* \rightarrow \Sigma^*$ is the natural projection.

If $G$ is marking aware with respect to $\Sigma'$, then any string $s$ reaching a marker state from a non-marker state must contain at least one event in $\Sigma'$. A sufficient and necessary condition to make $G$ marking aware with respect to $\Sigma'$ is to put in $\Sigma'$ every event that labels a transition from a non-marker state to a marker state, namely $\{s \in \Sigma | (\exists x \in X \setminus X_m)(\exists \tau' \in X_m)\tau' \in \xi(x, s)\} \subseteq \Sigma'$.

**Definition 4:** Given an alphabet $\Sigma$, we bring in a new event symbol $\tau \notin \Sigma$, and call $G^\tau = (X^\tau, \Sigma \cup \{\tau\}, \xi^\tau, x_0^\tau, X_m)$ standardized if
1) $x_0^\tau \notin X_m \land (\forall x \in X^\tau)[\xi^\tau(x, \tau) \neq \emptyset \iff x = x_0^\tau]$;
2) $(\forall s \in \Sigma)\xi^\tau(x_0^\tau, s) = \emptyset$;
3) $(\forall x \in X^\tau)(\forall \sigma \in \Sigma \cup \{\tau\})x_0^\tau \notin \xi^\tau(x, \sigma)$.

A standardized automaton is nothing but an automaton, in which $x_0^\tau$ is not marked and has only outgoing $\tau$ transitions with no incoming transitions, and no state except $x_0^\tau$ has outgoing $\tau$ transition. For an ordinary automaton $G = (X, \Sigma, \xi, x_0, X_m)$ we can standardize it (i.e., convert it into a standardized automaton) by simply (1) extending the alphabet to $\Sigma \cup \{\tau\}$, (2) adding a new state $x_0^\tau$, and (3) defining a new transition map $\xi^\tau$ such that $\xi^\tau(x_0^\tau, \tau) = \{x_0\}$ and for any $(x, \sigma) \in X \times \Sigma$ we have $\xi^\tau(x, \sigma) = \xi(x, \sigma)$. The resulting automaton $G^\tau = (X \cup \{x_0^\tau\}, \Sigma \cup \{\tau\}, \xi^\tau, x_0^\tau, X_m)$ is a standardized automaton. From now on, unless specified explicitly, we assume that every alphabet contains $\tau$. Thus, if we say $\Sigma_1$ and $\Sigma_2$ are two alphabets, then $\tau \in \Sigma_1 \cap \Sigma_2$; and if we say $\Sigma' \subseteq \Sigma$ is an alphabet, then $\tau \in \Sigma'$. Let $\phi(\Sigma)$ be the collection of all standardized finite-state automata, whose alphabet is $\Sigma$. By a slight abuse of notation, we use $G$ to denote a standardized automaton $G^\tau$. We can easily see that the product of two standardized automata is still a standardized automaton, and abstraction of a standardized automaton is also standardized as long as $\tau$ is in the abstraction alphabet. The concepts of marking awareness and standardized automata are used in the following result, which is extensively used in this paper.

**Proposition 1:** Given alphabets $\Sigma$ and $\Sigma'$ with $\Sigma' \subseteq \Sigma$, let $G \in \phi(\Sigma)$ and $P : \Sigma^* \rightarrow \Sigma^*$ be the natural projection. Then
1) $P(B(G)) = B(G/\equiv_{\Sigma'})$ and $P(N(G)) = N(G/\equiv_{\Sigma'})$.
2) If $G$ is marking aware with respect to $\Sigma'$, then $P(B(G)) = B(G/\equiv_{\Sigma'})$.

The proof is given in the Appendix, which indicates that, if $G$ is not standardized, then we may not always have $P(B(G)) \subseteq B(G/\equiv_{\Sigma'})$ and $P(N(G)) = N(G/\equiv_{\Sigma'})$, which are critically important in abstraction-based synthesis.

As an illustration of Prop. 1, Fig. 2 depicts an example, where $\Sigma = \{\sigma, a, b\}$ and $\Sigma' = \{\tau, b\}$. We can check that $P(N(G)) = \{\tau\} = N(G/\equiv_{\Sigma'})$. But $P(B(G)) = \{\tau\}$ and $B(G/\equiv_{\Sigma'}) = \{\tau, \tau b\}$, namely $P(B(G)) \subseteq B(G/\equiv_{\Sigma'})$. In this example, to make $G$ marking aware with respect to $\Sigma'$, $a$ must be included in $\Sigma'$. If we set $\Sigma' = \{\tau, a\}$ then $P(B(G)) = B(G/\equiv_{\Sigma'})$, as predicted in Prop. 1.

To show the usefulness of automaton abstraction in supervisor synthesis, we need the following concept.

**Definition 5:** Given automata $G_i = (X_i, \Sigma_i, \xi_i, x_{i0}, X_{im})$ ($i = 1, 2$), we say $G_1$ is nonblocking preserving with respect to $G_2$, denoted as $G_1 \sqsubset G_2$, if $B(G_1) \subseteq B(G_2)$, $N(G_1) = N(G_2)$, and for all $s \in N(G_1)$ and all $x_1 \in \xi_1(x_{10}, s)$, there exists $x_2 \in \xi_2(x_{20}, s)$ such that

\[N_{G_2}(x_2) \subseteq N_{G_1}(x_1) \land \{x_1 \in X_{1m} \iff x_2 \in X_{2m}\}.
\]

We say $G_1$ is nonblocking equivalent to $G_2$, denoted as $G_1 \cong G_2$, if $G_1 \sqsubset G_2$ and $G_2 \sqsubset G_1$.

By Def. 5, if $G_1$ is nonblocking preserving w.r.t. $G_2$ then their nonblocking behaviors are equal, but $G_2$’s blocking behavior may be larger. The last condition is used to guarantee that nonblocking preserving is conserved under automaton product and abstraction. If additionally $G_2$ is nonblocking preserving w.r.t. $G_1$, then they are nonblocking equivalent. We now present a few results.

**Fig. 1. Example 1:** $G$, $G/\equiv_{\Sigma'}$, $(G/\equiv_{\Sigma'})/\equiv_{\Sigma'}$ and $G/\equiv_{\Sigma'}$.

**Fig. 2. Example 2:** $G$ and $G/\equiv_{\Sigma'}$. 
Proposition 2: Given $G_1, G_2 \in \phi(\Sigma)$, $G_3 \in \phi(\Sigma)$, if $G_1 \subseteq G_2$ then $G_1 \times G_3 \subseteq G_2 \times G_3$.

Corollary 1: Given $G_1, G_2 \in \phi(\Sigma)$, $G_3 \in \phi(\Sigma)$, if $G_1 \approx G_2$ then $G_1 \times G_3 \approx G_2 \times G_3$.

By Prop. 2 and Cor. 1 nonblocking preserving and equivalence are invariant under automaton product.

Proposition 3: Given $G_i \in \phi(\Sigma_i)$ with $i = 1, 2$, let $\Sigma' \subseteq \Sigma_1 \cup \Sigma_2$. If $\Sigma'_1 \cap \Sigma'_2 \subseteq \Sigma'$, then

1) $(G_1 \times G_2)/\approx_{\Sigma'_1 \cap \Sigma'_2} \subseteq (G_1/\approx_{\Sigma'_1 \cap \Sigma'_2}) \times (G_2/\approx_{\Sigma'_1 \cap \Sigma'_2})$.

2) If additionally $G_i$ ($i = 1, 2$) is marking aware with respect to $\Sigma_i \cap \Sigma'$, then

$$(G_1 \times G_2)/\approx_{\Sigma'} \subseteq (G_1/\approx_{\Sigma'1 \cap \Sigma'2}) \times (G_2/\approx_{\Sigma'_1 \cap \Sigma'_2})$$.

Thus, $B((G \times S)/\approx_{\Sigma'}) \subseteq B((G/\approx_{\Sigma'} \times (S/\approx_{\Sigma'}))$.

Furthermore, $G \times S$ is also marking aware with respect to $\Sigma'$ because both $G$ and $S$ are marking aware with respect to $\Sigma'$. By Prop. 1 we get that

$$P(B(G \times S)) = B((G \times S)/\approx_{\Sigma'})$$.

Thus, if $G$ is marking aware with respect to $\Sigma'$, then $B((G \times S)/\approx_{\Sigma'}) \times S') = \emptyset \iff B(G \times S) = \emptyset$.

Theorem 1: Given two alphabets $\Sigma$ and $\Sigma'$ with $\Sigma' \subseteq \Sigma$, let $G \in \phi(\Sigma)$ and $S \in \phi(\Sigma')$.

1) $B((G/\approx_{\Sigma'}) \times S) = \emptyset \iff B(G \times S) = \emptyset$.

2) If $G$ is marking aware with respect to $\Sigma'$, then $B((G/\approx_{\Sigma'}) \times S) = \emptyset$ if and only if $B(G \times S) = \emptyset$.

Proof: Let $P : \Sigma^* \rightarrow \Sigma^*$ be the natural projection

$$B((G/\approx_{\Sigma'}) \times S) = \emptyset \iff B(G/\approx_{\Sigma'} \times (S/\approx_{\Sigma'})) = \emptyset$$

because $S/\approx_{\Sigma'} \approx S$ and by Corollary 1

$$B((G \times S)/\approx_{\Sigma'}) = \emptyset$$ by Prop. 3

$$P(B(G \times S)) = \emptyset$$ by Equation 2

Thus, $B((G \times S)/\approx_{\Sigma'}) = \emptyset \iff B(G \times S) = \emptyset$.

III. AUTOMATON ABSTRACTION IN SUPERVISOR SYNTHESIS

In this section we first introduce concepts of a supervisor synthesis problem, which is to compute a deterministic nonblocking state-controllable, state-observable (or state-normal) supervisor of a nondeterministic plant under a deterministic specification. Then we achieve our main objective of this paper: to establish a connection between the existence of a nonblocking supervisor of a plant and the existence of a nonblocking supervisor of an abstraction of the plant, generated by the proposed abstraction technique.

A. Concepts of a Supervisor Synthesis Problem

Given $G = (X, \Sigma, \xi, x_0, X_m)$, for each $x \in X$ let

$$E_G : X \times 2^\Sigma : x \mapsto E_G(x) := \{\sigma \in \Sigma | \xi(x, \sigma) \neq \emptyset\}$$.

Thus, $E_G(x)$ is simply the set of all events allowable at $x$ in $G$.

We now bring in the concept of state controllability. Let $\Sigma = \Sigma_1 \cup \Sigma_2$ and $G = (X, \Sigma, \xi, x_0, X_m)$ be a finite-state probabilistic automaton. We say that $G$ is state-controllable if for each $x \in X$ and each $\sigma \in \Sigma$, there exists $y \in X$ such that

$$\xi(x, \sigma) \not\subseteq (X_m \cup \{y\})$$.

The set $\Sigma$ controllable by $G$ is defined by

$$C(G) = \{\sigma \in \Sigma | \exists y \in X \text{ s.t. } \xi(x, \sigma) \not\subseteq (X_m \cup \{y\})\}$$.

The set of all events allowable at $x$ in $G$ is called the allowable set at $x$ and is denoted by $\Sigma_x$. The allowable set at $x$ is defined by

$$\Sigma_x = \sigma \in \Sigma | \xi(x, \sigma) \not\subseteq (X_m \cup \{y\})\}$$.
\[ \Sigma_c \cup \Sigma_u \text{, where } \Sigma_c \text{ is the set of controllable events, } \Sigma_u \text{ is the set of uncontrollable events and } \tau \in \tau_{uc}. \]

**Definition 6:** Given \( G = (X, \Sigma_c, \xi, x_0, X_M) \) and \( \Sigma' \subseteq \Sigma \), let \( A = (Y, \Sigma', \eta_y, y_0, Y_M) \) and \( P : \Sigma' \rightarrow \Sigma^* \) be the natural projection. \( A \) is state-controllable with respect to \( G \) and \( \tau_{uc} \) if for all \( s \in L(G \times A) \), \( x \in \xi((x_0, s), y) \in \eta_y(P(s)), \) and \( \eta(x, \sigma) \neq \emptyset \). Therefore, \( \sigma \in E_G(x) \cap \Sigma_{uc} \). There are two cases: (1) \( \sigma \in \Sigma' \). Then since \( A \) is state-controllable, by Def. 6 we have \( \sigma \in E_A(y) \), which means \( \sigma \in E_G(x, y) \). Thus, let \( n \in L(G \times A) \); (2) \( \sigma \in \Sigma - \Sigma'. \) Then \( \xi(x, \sigma) \neq \emptyset \) implies that \( \xi \times \eta((x_0, y_0), s) \neq \emptyset \). Therefore, we have \( \sigma \in E_G(x) \). In either case we have \( \sigma \in E_G(x) \). Thus, it is always true that state controllability implies language controllability described in the RW paradigm. But the reverse statement is not true unless both \( A \) and \( G \) are autonomous. We now introduce the concept of state observability. Let \( \Sigma = \Sigma_{uc} \cup \Sigma_{uo} \), where \( \Sigma_0 \) is the set of observable events, \( \Sigma_{uo} \) is the set of unobservable events and \( \tau \in \tau_{uo} \). Let \( P_0 : \Sigma^{*} \rightarrow \Sigma' \) be the natural projection.

**Definition 7:** Given \( G = (X, \Sigma_c, \xi, x_0, X_M) \) and \( \Sigma' \subseteq \Sigma \), let \( A = (Y, \Sigma', \eta_y, y_0, Y_M) \). \( A \) is state-observable with respect to \( G \) and \( P_0 \), if for all \( s, s' \in L(G \times A) \) with \( P_0(s) = P_0(s') \), and for all \( (x, y), (x', y') \in \xi \times \eta((x_0, y_0), s) \), \( (x', y') \in \xi \times \eta((x_0, y_0), s') \), we have \( E_G(x, y) \cap E_G(x', y') \subseteq E_A(y') \).

State observability defined in Def. 7 is more general than the one defined in [7], as the authors in [7] consider a subautomaton of \( G \) and only one event is unobservable. By Def. 7, if \( A \) is state observable then for any two states \( (x, y) \) and \( (x', y') \) in \( G \times A \) reachable by two strings \( s \) and \( s' \) having the same projected image (i.e. \( P_0(s) = P_0(s') \)), any event \( \sigma \) allowed at \( (x, y) \) and \( x' \) must be allowed at \( y' \) as well. We can check that, if \( A \) is state-observable then for all \( s, s' \in L(G \times A) \) with \( P_0(s) = P_0(s') \) and \( \sigma \in \Sigma \), \( s \sigma \in L(G \times A) \) and \( s' \sigma \in L(G \times A) \).

This can be briefly shown as follows. Let \( s, s' \in L(G \times A) \) with \( P_0(s) = P_0(s') \) and \( \sigma \in \Sigma \), \( s \sigma \in L(G \times A) \) and \( s' \sigma \in L(G \times A) \). There must exist \( x, x' \in X \) and \( y, y' \in Y \) such that \( (x, y) \in \xi \times \eta((x_0, y_0), s), (x', y') \in \xi \times \eta((x_0, y_0), s'), \xi(x, \sigma) \neq \emptyset, (x', \sigma') \neq \emptyset \). Clearly, \( \sigma \in E_G(x, y) \) and \( \sigma \in E_G(x', y') \). There are two cases: (1) \( \sigma \in \Sigma' \). Then since \( A \) is state-observable, by Def. 7 we have \( \sigma \in E_G(x, y) \cap E_G(x', y') \subseteq E_A(y') \). Thus, \( \sigma \in E_G(x') \), which means \( \sigma \sigma \in L(G \times A) \). (2) \( \sigma \in \Sigma - \Sigma' \). Then \( \xi(x, \sigma) \neq \emptyset \) implies that \( \xi(\sigma', x') \neq \emptyset \), which means \( \sigma \sigma \in L(G \times A) \). In either case, we have \( \sigma \sigma \in L(G \times A) \). Thus, state observability implies observability defined in [11]. But the inverse statement is not always true unless both \( A \) and \( G \) are deterministic. Notice that, if \( \Sigma_0 = \Sigma \), namely every event is observable, \( A \) may still not be state-observable, owing to nondeterminism. In many applications we are interested in an even stronger observability property called state normality which is defined as follows.

**Definition 8:** Given \( G = (X, \Sigma_c, \xi, x_0, X_M) \) and \( \Sigma' \subseteq \Sigma \), let \( A = (Y, \Sigma', \eta_y, y_0, Y_M) \) and \( P : \Sigma' \rightarrow \Sigma^* \) be the natural projection. \( A \) is state-normal with respect to \( G \) and \( P_0 \), if for all \( s \in L(G \times A) \), \( s' \in \bigcap \sigma \sigma' \subseteq L(G \times A) \) and for all \( (x, y) \in \xi \times \eta((x_0, y_0), s), (x, y') \in \xi((x_0, y_0), s') \), and \( \sigma \in \Sigma^* \), if \( P_0(s) = P_0(s') \), and \( \xi(x, \sigma) \neq \emptyset \), then \( \eta(y, P_0(s')) \neq \emptyset \).

We can check that, if \( A \) is state-normal with respect to \( G \) and \( P_0 \), then \( L(G) \cap \bigcap \sigma \sigma' \subseteq L(G \times A) \), which implies \( L(G \times A) \) is normal with respect to \( L(G) \) and \( P_0 \), as defined in [11]. This can be briefly shown as follows. Let \( s \in L(G \times A) \) and \( \xi(x, \sigma) \neq \emptyset \) such that \( t \in L(G \times A) \). Since \( s, t \in L(G \times A) \), there exist \( s', t' \in L(G \times A) \) such that \( P_0(s) = P_0(s') \). Then, there exist \( \xi(x, \sigma) \neq \emptyset \) implies that \( \xi \times \eta((x_0, y_0), s) \neq \emptyset \). Therefore, we have \( \sigma \in E_G(x) \). In either case we have \( \sigma \in E_G(x) \). Thus, it is always true that state controllability implies language controllability described in the RW paradigm. But the reverse statement is not true unless both \( A \) and \( G \) are autonomous. Furthermore, we can check that state normality implies state observability. But the inverse statement is not true.

**Definition 9:** Given \( G \in \phi(\Sigma) \) and \( H \in \phi(\Delta) \) with \( \Delta \subseteq \Sigma' \subseteq \Sigma \), let \( \alpha \in \phi(\Sigma') \) and \( \alpha \) is a nonblocking supervisor of \( G \) under \( H \), if \( S \) is deterministic and the following conditions hold:

1. \( N(G \times S) \subseteq N(G \times H) \);
2. \( N(G \times S) = \emptyset \);
3. \( S \) is state-controllable w.r.t. \( G \) and \( \tau_{uc} \);
4. \( S \) is state-observable (or state-normal) w.r.t. \( G \) and \( P_0 \).

The first condition of Def. 9 indicates that \( G \times S \), which represents the closed-loop system in the sense that \( G \) is supervised by \( S \), complies with the specification \( H \) in terms of language inclusion. Because of this condition we only consider \( H \) to be deterministic. The use of a nondeterministic specification is described in, e.g. [15], where the goal is to achieve a closed-loop system \( G \times S \) that reduces the requirement \( H \) in terms of failure semantics. Because this paper is about the usage of abstraction in synthesis, which may or may not be applicable to cases with nondeterministic specifications, we decide to use deterministic specifications. For practical applications, it is not necessary that \( \tau \in \Delta \). The second condition indicates \( G \times S \) is nonblocking. The third and fourth ones are self-explanatory. Later we will use the term "nonblocking state-normal supervisor," when we want to emphasize that \( S \) is state-normal with respect to \( G \) and \( P_0 \). The following result provides a sufficient and necessary condition for the existence of a nonblocking supervisor.

**Theorem 2:** Given \( G \in \phi(\Sigma) \) and \( H \in \phi(\Delta) \) with \( \Delta \subseteq \Sigma' \subseteq \Sigma \), there exists a nonblocking supervisor \( S \in \phi(\Sigma') \) of \( G \) under \( H \) if and only if there exists \( A \in \phi(\Sigma') \) with \( L(A) = N(\bar{A}) \) such that

1. \( N(G \times A) \subseteq N(G \times H) \);
2. \( N(G \times A) = \emptyset \);
3. \( A \) is state-controllable w.r.t. \( G \) and \( \tau_{uc} \);
4. \( A \) is state-observable (or state-normal) w.r.t. \( G \) and \( P_0 \).
The proof of Theorem 2 indicates that a nonblocking supervisor is simply a recognizer of an automaton A which satisfies those four conditions. In [11], [18] we know that controllability and normality are closed under language union. The following result shows that state controllability and state normality bear a similar feature.

Proposition 4: Given \( G \in \phi(\Sigma) \) and \( H \in \phi(\Delta) \) with \( \Delta \subseteq \Sigma' \subseteq \Sigma \), let \( S_i \in \phi(\Sigma_i') \), then \( \bigwedge_{i=1,2} \) be a nonblocking state-normal supervisor of \( G \) under \( H \) and \( L(S_i) = N(S_i) \). Let \( S \in \phi(\Sigma') \) be a deterministic automaton with \( N(S) = N(S_1) \cup N(S_2) \) and \( L(S) = N(S) \). Then \( S \) is a nonblocking state-normal supervisor of \( G \) w.r.t. \( H \).

By Prop. 4 the union of two nonblocking state-normal (NSN) supervisors is still a NSN supervisor. We define a set

\[ \mathcal{N}(G, H, \Sigma') := \{ S \in \phi(\Sigma') | S \text{ is a NSN supervisor of } G \text{ under } H \land L(S) \subseteq N(G/\approx_{\Sigma'}) \} \]
fetching a work piece \((a_i)\); 2) preprocessing \((b_i)\); 3) postprocessing \((c_i)\); 4) polishing \((e_i)\); 5) packaging \((d_i)\). After preprocessing \(b_i\), there are two choices: to be postprocessed directly \((c_i)\) or to be polished first \((e_i)\) before postprocessing. The latter gives a product with better quality. The negative aspect is that polishing may cause the machine \(G_1\) to fail \((f_1)\). If failure does happen, \(G_1\) will stop automatically and wait for repair. Among each alphabet \(\Sigma_i\), the controllable alphabet is \(\Sigma_{c_i} = \{a_i, e_i\}\), and for the purpose of simplicity the observable alphabet \(\Sigma_{d_i} = \Sigma_i - \{\tau\}\), namely every event except for \(\tau\) is observable. There is one specification \(H \in \phi(\Delta)\) with \(\Delta = \{e_1, e_2\}\), depicted in Fig. 7, indicating that if a work piece is polished in \(G_1(e_1)\), then a work piece must be polished in \(G_2\) afterwards \((e_2)\). We now start to synthesize a nonblocking supervisor for \(G_1 \times G_2\) that complies with the specification \(H\).

First, we create an appropriate abstraction of \(G_1 \times G_2\). We pick \(\Sigma' = \{\tau, a_1, a_2, e_1, e_2\}\). The rationality is that, since \(\Delta \subseteq \Sigma'\), the abstraction \((G_1 \times G_2) / \approx_{\Sigma'}\) can capture constraints imposed by the specification \(H\); and since all controllable events are in \(\Sigma'\), the abstraction \((G_1 \times G_2) / \approx_{\Sigma'}\) also contains all means of control available to \(G_1 \times G_2\) itself. Since \(\Sigma_1 \cap \Sigma_2 = \{\tau\} \subseteq \Sigma'\), by Prop. 3

\[
\left( (G_1 \times G_2) / \approx, \Sigma' \right) \subseteq \left( (G_1 / \approx_{\Sigma_1} \times G_2 / \approx_{\Sigma_2}) \right) \times \left( (G_1 / \approx_{\Sigma_1} \times G_2 / \approx_{\Sigma_2}) \right).
\]

The results of \((G_1 / \approx_{\Sigma_1} \times G_2 / \approx_{\Sigma_2})\) are depicted in Fig. 8. The product of two abstractions \(G' := (G_1 / \approx_{\Sigma_1} \times G_2 / \approx_{\Sigma_2})\) is depicted in Fig. 9. We now use \(G'\) and \(H\) to synthesize a supervisor. The product \(G' \times H\) is depicted in Fig. 9. Clearly, the transitions \(e_1\) from state \((2,0)\) to state \((3,1)\), and from \((5,0)\) to \((4,1)\) in \(G' \times H\) must be disabled. Otherwise, blocking states \((3,1)\) and \((4,1)\) will be reached. Once these two transitions are disabled, transitions \(e_2\) from \((2,0)\) to \((1,1)\), and from \((5,0)\) to \((6,1)\) must be disabled as well because, otherwise, the remaining automaton is neither state-normal nor state-observable. After removing transitions \(e_1\) at states \((2,0)\) and \((5,0)\) in Fig. 9, the remaining reachable part \(A\) is depicted in Fig. 10, which is nonblocking, state-controllable, state-normal (and state-observable). By Theorem 2 we get that a recognizer \(S\) of the marked behavior \(N(A)\), depicted in Fig. 11, is a nonblocking supervisor of \(G'\) under \(H\). We can see that \(S\) does not allow events \(e_1\) and \(e_2\) to happen. It is not difficult to check that \(S\) is a nonblocking supervisor of \(G_1 \times G_2\) under \(H\), as predicted by Theorem 3. We can verify that the maximum number of states of any intermediate automaton computed is 13, which occurs when we compute \(G' \times H\). Clearly, abstractions help to reduce the computational complexity in this example because otherwise we will have to face the product \(G_1 \times G_2\) directly, which has 61 states. The abstraction technique has been applied to a semiconductor cluster tool example in [21], where the monolithic plant model has about \(2.68 \times 10^6\) states and, as a contrast, the largest abstraction has only 985 states. Thus, the abstraction-based synthesis shows a significant computational advantage over centralized synthesis. It has also been applied to a cable service network example in [20], where the ratio of the sizes of state sets of abstractions obtained by using our approach and the observer-based approach is \((1.25)^n\), where \(n\) denotes the number

**IV. EXAMPLE**

Suppose we have models of two machines, which are part of one processing unit and functionally identical, except for individual event labels. The system is depicted in Fig. 6. Each machine \(G_i\) \((i = 1, 2)\) has the following standard operations: 1)
of residents in a community. Clearly, our abstraction approach enjoys a computational advantage over the observer-based abstraction approach. We are applying this technique to other case studies at the moment to test its efficiency compared with other automaton-based abstraction techniques in the literature.

V. CONCLUSION

In this paper, we first present a new technique that computes an abstraction of a nondeterministic finite-state automaton and provide some relevant properties. Then we show the usage of this technique in a synthesis problem, where supervisors and specifications are deterministic but plant models are nondeterministic. After introducing the concepts of state controllability, state observability and state normality, we show that a nonblocking supervisor of an abstraction $G'/\approx_{\Sigma'}$ under a specification $H$ is also a nonblocking supervisor of the original plant $G$ under the same specification. The inverse statement is true, if all observable events are contained in $\Sigma'$ and the plant $G$ is marking aware with respect to $\Sigma'$. In this paper we also present a sufficient and necessary condition for the existence of a nonblocking supervisor and show that the supremal nonblocking state-normal supervisor exists for a plant $G$ and a specification $H$. The concrete procedure to compute such a supremal supervisor is not provided, owing to the different objective of this paper and the page limit as well. It is addressed in another paper of the authors [21].

Although results in this paper are about standardized automata, they are applicable in a supervisor synthesis problem, where $G$ is non-standardized in the sense that $\tau \not\in \Sigma$. To do this, we first standardize $G$ to obtain $G^\tau$, then synthesize a standardized nonblocking supervisor $S^\tau$ based on $G^\tau/\approx_{\Sigma'}$. Since $S^\tau$ is deterministic, we can convert it to a non-standardized automaton by simply removing the $\tau$ transition and setting the target state of the $\tau$ transition as the initial state of the resultant automaton $S$. Since $\tau$ is uncontrollable and unobservable, we can show that $S$ is a nonblocking supervisor of $G$, which is introduced in [21] for aggregative synthesis of distributed supervisors.

APPENDIX

1) Proof of Prop. 1: Let $\xi'$ be the transition map of $G/\approx_{\Sigma'}$. First we show that $P(B(G)) \subseteq B(G/\approx_{\Sigma'})$. For each string $s \in P(B(G))$, there exists $t \in B(G)$ with $P(t) = s$ such that

$$\left(\exists x \in \xi(x_0, t) \right) \left(\forall t' \in \Sigma^* \right) \xi(x, t') \cap X_m = \emptyset.$$  

Since $G$ is standardized, $P(t) \neq \varepsilon$ iff $t \neq \varepsilon$. Thus, we get that $\langle x \rangle \in \xi'((x_0), P(t))$. Because

$$\left(\forall t' \in \Sigma^* \right) \xi(x, t') \cap X_m = \emptyset,$$

we have that, $\left(\forall s' \in \Sigma^* \right) \xi'((x), s') \cap (X_m/\approx_{\Sigma'}) = \emptyset$. Thus, $s = P(t) \in B(G/\approx_{\Sigma'})$. To show $P(N(G)) \subseteq N(G/\approx_{\Sigma'})$, let $s \in P(N(G))$. Then

$$\left(\exists t \in N(G) \right) P(t) = s \wedge \xi(x_0, t) \cap X_m \neq \emptyset.$$  

Since $G$ is standardized, from $\xi(x_0, t) \cap X_m \neq \emptyset$ we have $\xi'((x_0), P(t)) \cap (X_m/\approx_{\Sigma'}) \neq \emptyset$. Thus, $s \in N(G/\approx_{\Sigma'})$. To show $N(G/\approx_{\Sigma'}) \subseteq P(N(G))$, let $s \in N(G/\approx_{\Sigma'})$. Then we have $\xi'((x_0), s) \cap (X_m/\approx_{\Sigma'}) \neq \emptyset$, which means, there exists $t \in \Sigma^*$ with $P(t) = s$ such that $\xi(x_0, t) \cap X_m \neq \emptyset$. Thus, $t \in N(G)$, namely $P(t) = s \in P(N(G))$. Therefore, we have $P(N(G)) = N(G/\approx_{\Sigma'})$. 

\[\]
Finally, suppose $G$ is marking aware with respect to $\Sigma^*$. To show $B(G/\approx_{\Sigma^*}) = P(B(G))$, we only need to show that $B(G/\approx_{\Sigma^*}) \subseteq P(B(G))$. For each string $s \in B(G/\approx_{\Sigma^*})$, there exists $\langle x \rangle \in \xi^t_0(x_0)0$ such that

$$\forall s' \in \Sigma^* \xi^t_0(x, s') \cap (x_m/\approx_{\Sigma^*}) = \emptyset$$

from which we can derive that, there exists $t \in \Sigma^*$ such that $P(t) = s$ and

$$x \in \xi(x_0, t) \land (\forall t' \in \Sigma^*) (x, t') \cap X_m \neq \emptyset \Rightarrow t' \in (\Sigma - \Sigma^*)^*.$$ Clearly, $x \not\in X_m$, because otherwise $\xi^t_0(x, s') \cap (x_m/\approx_{\Sigma^*}) \neq \emptyset$. We claim that $x$ is a blocking state of $G$. Otherwise, there exists $t' \in \Sigma^*$ such that $\xi(x, t') \cap X_m \neq \emptyset$. Since $G$ is marking aware with respect to $\Sigma^*$, we have that $t' \not\in (\Sigma - \Sigma^*)^*$, which contradicts the fact that

$$\forall t' \in \Sigma^* \xi(x, t') \cap X_m \neq \emptyset \Rightarrow t' \in (\Sigma - \Sigma^*)^*.$$ Thus, the claim is true. Since $x$ is a blocking state, we have $t \in B(G)$, namely $s = P(t) \in B(G).$

2) Proof of Prop. 2: Let $G_i = (X_i, \Sigma_i, \xi_i, x_{i0}, X_{im})$ with $i = 1, 2, 3$, where $\Sigma_1 = \Sigma_2 = \Sigma$ and $\Sigma_3 = \Sigma'$. Let $P : (\Sigma \cup \Sigma')^* \rightarrow \Sigma^*$ and $P' : (\Sigma \cup \Sigma')^* \rightarrow \Sigma'^*$ be natural projections. We first show that $N(G_1 \times G_3) = N(G_2 \times G_3)$. Clearly, we have $N(G_1 \times G_3) = N(G_1) \cap N(G_3)$. Since $G_1 \subseteq G_2$, we have $N(G_1) = N(G_2)$. Thus, we have $N(G_1 \times G_3) = N(G_1) \cap N(G_3) = N(G_2 \times G_3).$

To show that $B(G_1 \times G_3) \subseteq B(G_2 \times G_3)$, let $s \in B(G_1 \times G_3)$. By the definition of automaton product, there exists $x_1 \in X_1$ such that $x_1 \in \xi_1(x_{10}, P(s))$. There are two cases to consider. Case 1: $x_1$ is a blocking state. Then $P(s) \in B(G_1) \subseteq B(G_2)$. Thus, $s \in B(G_2 \times G_3)$. Case 2: $x_1$ is a nonblocking state. Since $G_1 \subseteq G_2$, there exists $x_2 \in \xi_2(x_{20}, P(s))$ such that $N_{G_1}(x_1) \subseteq N_{G_2}(x_2)$. Since $s \in B(G_1 \times G_3)$, there exists $x_3 \in X_3$ such that $\langle x_1, x_3 \rangle \in \xi_3((x_{10}, x_{30}), s)$ and $N_{G_1 \times G_3}(x_1, x_3) = \emptyset$. We have

$$N_{G_2 \times G_3}(x_2, x_3) = N_{G_2}(x_2) \cap N_{G_3}(x_3) \subseteq N_{G_1}(x_1) \cap N_{G_3}(x_3).$$

Thus, $(x_2, x_3)$ is a blocking state of $G_2 \times G_3$, which means $s \in B(G_2 \times G_3)$. Therefore, in either case we have $B(G_1 \times G_3) \subseteq B(G_2 \times G_3)$.

Finally, from the above argument in Case 2, for any $s \in (\Sigma \cup \Sigma')^*$ and $(x_1, x_3) \in \xi_1(\xi_1(x_{10}, x_{30}), s)$, we have $(x_2, x_3) \in \xi_2(\xi_2(x_{20}, x_{30}), s)$ such that $N_{G_2 \times G_3}(x_2, x_3) \subseteq N_{G_1 \times G_3}(x_1, x_3).$

3) Proof of Prop. 3: Let $G_i = (X_i, \Sigma_i, \xi_i, x_{i0}, X_{im}) \in \phi(\Sigma_i)$ with $i = 1, 2$. For notation simplicity let $\Sigma_3 = \Sigma \cap \Sigma'$, and $P : (\Sigma \cup \Sigma')^* \rightarrow \Sigma^*$, $P_1 : \Sigma \rightarrow \Sigma$, $P_2 : \Sigma \rightarrow \Sigma$, $P_3 : \Sigma \rightarrow \Sigma$, $P_4 : \Sigma \rightarrow \Sigma$, and $Q_1 : (\Sigma \cup \Sigma')^* \rightarrow \Sigma^*$ be natural projections, $\xi^t_1$ for the transition map of $G_1/\approx_{\Sigma_1}$, $\xi^t_2$ for the transition map of $G_2/\approx_{\Sigma_2}$.

First, we have the following:

$$N((G_1 \times G_2)/\approx_{\Sigma^*}) = P(N(G_1 \times G_2)) \text{ by Prop. 1 } = P(N(G_1)) \cap N(G_2) = P_1(N(G_1)) \cap P_2(N(G_2)) \text{ because } \Sigma_1 \cap \Sigma_2 \subseteq \Sigma^* = N((G_1/\approx_{\Sigma_1}) \cap N((G_2/\approx_{\Sigma_2}) \text{ by Prop. 1 } = N((G_1/\approx_{\Sigma_1}) \times (G_2/\approx_{\Sigma_2})).$$

Next, we show that

$$B((G_1 \times G_2)/\approx_{\Sigma^*}) \subseteq B((G_1/\approx_{\Sigma_1}) \times (G_2/\approx_{\Sigma_2})).$$

Let $s \in B((G_1 \times G_2)/\approx_{\Sigma^*})$. Then there exists $(x_1, x_2) \in X_1 \times X_2$ such that $\langle x_1, x_2 \rangle \in \xi^t_1((x_{10}, x_{20})_\Sigma, s)$ and for all $s' \in \Sigma^*$

$$\xi^t_1((x_{10}, x_{20})_\Sigma, s') \cap (\langle X_{1m}, X_{2m} \rangle/\approx_{\Sigma^*}) = \emptyset$$

which means $(x_1, x_2) \not\in X_{1m} \times X_{2m}$ and there exists $t \in (\Sigma_1 \cup \Sigma_2)^*$ with $P(t) = s$ such that $(x_1, x_2) \in \xi_1 \times \xi_2((x_{10}, x_{20})_\Sigma, t)$ and for all $t' \in (\Sigma_1 \cup \Sigma_2)^*$.

$$\xi_1 \times \xi_2((x_1, x_2), t') \cap (\langle X_{1m}, X_{2m} \rangle/\approx_{\Sigma^*}) = \emptyset$$

Since $G_1$ and $G_2$ are standardized, from $(x_1, x_2) \in \xi_1 \times \xi_2((x_{10}, x_{20})_\Sigma, t)$ and the fact that $\Sigma_1 \cap \Sigma_2 \subseteq \Sigma$ we can derive that

$$\langle (x_1, x_2) \xi_3, \xi_4 \rangle \Xi^t_1 \Xi^t_2 \langle (x_{10}, x_{20})_\Sigma, s \rangle.$$

We claim that $(\xi_3, \xi_4) \subseteq (\langle x_{10}, x_{20} \rangle, s)$ is a blocking state of $G_i/\approx_{\Sigma_i}$, and otherwise, there exists $s' \in \Sigma^*$ such that

$$\xi^t_1 \times \xi^t_2 \langle (\langle x_{10}, x_{20} \rangle, s'), \xi^t_1 \cap (\langle X_{1m}, X_{2m} \rangle/\approx_{\Sigma^*}) \neq \emptyset.$$

Since $(x_1, x_2) \not\in X_{1m} \times X_{2m}$, we get that $(\langle x_{10}, x_{20} \rangle, s') \not\in (\langle X_{1m}, X_{2m} \rangle/\approx_{\Sigma^*})$. Thus, $s' \not\in s$, which means there exists $t' \in \Sigma^*$ with $P(t) = s' \not\in (\xi_1 \times \xi_2)^t \cap (\langle X_{1m}, X_{2m} \rangle/\approx_{\Sigma^*})$—contradict the fact that for all $t' \in \Sigma^*$

$$\xi_1 \times \xi_2((x_1, x_2), t') \cap (\langle X_{1m}, X_{2m} \rangle/\approx_{\Sigma^*}) \neq \emptyset$$

From the claim we get that $s \in B((G_1/\approx_{\Sigma_1}) \times (G_2/\approx_{\Sigma_2}))$. Let $s \in N((G_1/\approx_{\Sigma_1}) \times (G_2/\approx_{\Sigma_2}))$. For any $(x_1, x_2) \in X_1 \times X_2$ with $\langle x_1, x_2 \rangle \not\in \xi^t_1((x_{10}, x_{20})_\Sigma, s)$, there exists $t' \in \Sigma^*$ such that $P(t') = s$ and $(x_1, x_2) \in \xi^t_1((x_{10}, x_{20}), t)$. Since $G_1$ and $G_2$ are standard, if $s = \epsilon$, then $t = \epsilon$, which means $(x_1, x_2) = (x_{10}, x_{20})$. Clearly, we have the following:

$$\langle (x_{10}, x_{20}, s), (x_{20}, s) \rangle \in \xi^t_1 \times \xi^t_2 \langle \langle (x_{10}, x_{20})_\Sigma, s \rangle, \epsilon \rangle.$$
If \(s \neq \varepsilon\), then by the definition of automaton abstraction and the assumption that \(\Sigma_1 \cap \Sigma_2 \subseteq \Sigma'\), we get

\[
\left(\langle x_1, x_2 \rangle, \langle x_3, x_4 \rangle \right) \in \mathcal{E}_1 \times \mathcal{E}_2 \left(\left(\langle x_1, 0 \rangle, \langle x_3, 0 \rangle \right), s\right).
\]

Thus, in either case we have

\[
\left(\langle x_1, x_2 \rangle, \langle x_3, x_4 \rangle \right) \in \mathcal{E}_1 \times \mathcal{E}_2 \left(\left(\langle x_1, 0 \rangle, \langle x_3, 0 \rangle \right), s\right).
\]

We now show that

\[
N(G_1, \approx_{\Sigma_1}) \times (G_2, \approx_{\Sigma_2}) \left(\langle x_1, x_2 \rangle, \langle x_3, x_4 \rangle \right) \subseteq N(G_1 \times G_2, \approx_{\Sigma'}) \left(\langle x_1, x_2 \rangle, \langle x_3, x_4 \rangle \right).
\]

Let \(s' \in N(G_1, \approx_{\Sigma_1}) \times (G_2, \approx_{\Sigma_2}) \subseteq (G_1, \approx_{\Sigma_1}) \times (G_2, \approx_{\Sigma_2})\). If \(s' = \varepsilon\), then \(\left(\langle x_1, x_2 \rangle, \langle x_3, x_4 \rangle \right) \subseteq (X_1 \times X_2, \approx_{\Sigma})\), from which we can derive that \(X_1 \times X_2 \subseteq \Sigma_1 \cap \Sigma_2\). Thus, \(\langle x_1, x_2 \rangle, \langle x_3, x_4 \rangle \in \Sigma'\), namely \(\varepsilon \in N(G_1 \times G_2, \approx_{\Sigma}) \subseteq (X_1 \times X_2, \approx_{\Sigma})\). If \(s' \neq \varepsilon\), then there exists \(t' \in \Sigma'\) with \(P(t') = s'\) such that \(\xi_1 \times \xi_2 \left(\langle x_1, x_2 \rangle, t' \right) \cap (X_1 \times X_2, \approx_{\Sigma}) \neq \emptyset\). Since \(P(t') \neq \varepsilon\), by the definition of automaton abstraction, we get that

\[
\xi'_1 \left(\langle x_1, x_2 \rangle, \langle x_3, x_4 \rangle \right) \cap \xi'_2 \left(\langle x_1, x_2 \rangle, \langle x_3, x_4 \rangle \right) \neq \emptyset
\]

which contradicts Expression (4). Thus, the claim is true, from which we have \(s \in B(G_1 \times G_2, \approx_{\Sigma})\).

Let \(s \in N(G_1, \approx_{\Sigma_1}) \times (G_2, \approx_{\Sigma_2})\). For all \(\langle x_1, x_2 \rangle \in X_1 \times X_2\) with \(\left(\langle x_1, x_2 \rangle, \langle x_3, x_4 \rangle \right) \subseteq \xi_1 \times \xi_2 \left(\langle x_1, x_2 \rangle, \langle x_3, x_4 \rangle \right), s\), there exists \(t \in \Sigma'\) such that

\[
P(t) = s \wedge \langle x_1, x_2 \rangle \in \xi'_1 \left(\langle x_1, x_2 \rangle, \langle x_3, x_4 \rangle \right), t\).
\]

Since \(G_1\) and \(G_2\) are standardized, if \(s = \varepsilon\), then \(t = \varepsilon\), which means \(\langle x_1, x_2 \rangle = \langle x_3, x_4 \rangle\). Clearly, we have

\[
\left(\langle x_1, x_2 \rangle, \langle x_3, x_4 \rangle \right) \in \xi'_1 \left(\langle x_1, x_2 \rangle, \langle x_3, x_4 \rangle \right), s\).
\]

Thus, in either case we have

\[
\left(\langle x_1, x_2 \rangle, \langle x_3, x_4 \rangle \right) \in \xi'_1 \left(\langle x_1, x_2 \rangle, \langle x_3, x_4 \rangle \right), s\).
\]

We claim that \(\langle (x_1, x_2) \rangle \subseteq \text{blocking state of } (G_1 \times G_2) / \approx_{\Sigma'}\). Otherwise, there exists \(s' \in \Sigma'\) such that

\[
\xi'_1 (\langle (x_1, x_2) \rangle, \langle x_3, x_4 \rangle) \subseteq (X_1, x_2)^m / X_2, m / \approx_{\Sigma'} \neq \emptyset.
\]

Since \((x_1, x_2) \not\subseteq X_1, m \times X_2, m\), we get \(\langle (x_1, x_2) \rangle \not\subseteq (X_1, x_2)^m / \approx_{\Sigma'}\). Thus, \(s' \neq \varepsilon\). Furthermore, since \(G_i (i = 1, 2)\) is marking aware with respect to \(\Sigma_i \cap \Sigma'_i\), we have \(P_i (s') \neq \varepsilon\). Thus, there exists \(t' \in \Sigma'\) with \(P(t') = s'\) such that

\[
\xi_1 \times \xi_2 (\langle (x_1, x_2), t' \rangle) \cap (X_1, m \times X_2, m) \neq \emptyset.
\]

Since \(P(t') \neq \varepsilon\) for \(i = 1, 2\) and \(\Sigma_1 \cap \Sigma_2 \subseteq \Sigma'\), we have

\[
\xi'_1 \times \xi'_2 (\langle (x_1, x_2), \langle x_3, x_4 \rangle \rangle, s') \subseteq \emptyset
\]

From Expression (3) we get that

\[
\left(\langle x_1, x_2 \rangle, \langle x_3, x_4 \rangle \right) \in \xi'_1 \times \xi'_2 \left(\left(\langle x_1, 0 \rangle, \langle x_3, 0 \rangle \right), s\right).
\]

From Expression (4) we get that \(\langle x_1, x_2 \rangle \not\subseteq X_1, m \times X_2, m\). Since \(G_1\) and \(G_2\) are standardized, from Expression (3) and the fact that \(\Sigma_1 \cap \Sigma_2 \subseteq \Sigma'\) we have

\[
\langle (x_1, x_2) \rangle \in \xi'_1 \left(\langle (x_1, 0, x_2, 0) \rangle, s\right).
\]
abstraction, we have
\[
\xi_1' \times \xi_2' \left( \left( \langle x_1 \rangle_{\Sigma_x} , \langle x_2 \rangle_{\Sigma_x} , \xi', s' \right) \cap \left( \left( x_{1, m} / \approx_{\Sigma_1} \right) \times \left( x_{2, m} / \approx_{\Sigma_2} \right) \right) \right) \neq \emptyset.
\]
Thus, \( s' \in N(G_1 / \approx_{\Sigma_1}) \times (G_2 / \approx_{\Sigma_2}) \). Case 2: \( \tilde{P}_1(s') = \varepsilon \) and \( \tilde{P}_2(s') \neq s' \neq \varepsilon \). Since \( G_1 \) is marking aware with respect to \( \Sigma_1 \) \( \cap \Sigma_2 \), \( \tilde{P}_1(s') = \varepsilon \) implies that \( x_1 \in X_{1,m} \).
Since \( \tilde{P}_2(s') \neq s' \neq \varepsilon \) we have
\[
\left( \langle x_2 \rangle_{\Sigma_x} \in X_{2, m} / \approx_{\Sigma_2} \right) \left( \langle x_2 \rangle_{\Sigma_x} \in \xi_2' \left( \langle x_2 \rangle_{\Sigma_x} , \tilde{P}_2(s') \right) \right).
\]
Thus
\[
\left( \langle x_1 \rangle_{\Sigma_x} , \langle x_2 \rangle_{\Sigma_x} \right) \in \xi_1' \times \xi_2' \left( \left( \langle x_1 \rangle_{\Sigma_x} , \langle x_2 \rangle_{\Sigma_x} , \xi', s' \right) \right)
\]
which means \( s' \in N(G_1 / \approx_{\Sigma_1}) \times (G_2 / \approx_{\Sigma_2}) \left( \left( \langle x_1 \rangle_{\Sigma_x} , \langle x_2 \rangle_{\Sigma_x} \right) \right) \). Case 3: \( \tilde{P}_1(s') \neq \varepsilon \) or \( \tilde{P}_2(s') = \varepsilon \). This case is similar to Case 2. In either case, we have
\[
N(G_1 \times G_2) / \approx_{\Sigma_2} \left( \left( \langle x_1 \rangle_{\Sigma_x} , \langle x_2 \rangle_{\Sigma_x} \right) \right) \subseteq \left( \left( \langle x_1 \rangle_{\Sigma_x} , \langle x_2 \rangle_{\Sigma_x} , \xi', s' \right) \right).
\]
Thus, \( (G_1 / \approx_{\Sigma_1}) \times (G_2 / \approx_{\Sigma_2}) \subseteq \left( G_1 \times G_2 \right) / \approx_{\Sigma_2} \).

4) Proof of Theorem 2: The ONLY IF part is obvious. We only need to show the IF part. Let \( S \) be a recognizer of \( N(A) \), i.e. \( N(S) = N(A) \) and \( L(S) = \overline{N(S)} \). Then we have
\[
N(G \times S) = \overline{N(G)} \mid (N(S) = \overline{N(G)} \mid (N(A) \subseteq N(G \times H)).
\]
Next, we show \( B(G \times S) = \emptyset \). Let \( G = (X, \Sigma_x, \xi, x_0, X_m) \), \( A = (Z, \Sigma_2, \xi_2, z_0, Z_m) \) and \( S = (Y, \Sigma_2, \eta, y_0, Y_m) \). Suppose \( B(G \times S) \neq \emptyset \). Then there exists \( s \in B(G \times S) \) and \( (x, y) \in \xi \times \eta(x_0, y_0, s) \) such that for all \( s' \in \Sigma^* \nexists \xi \times \eta((x, y), s') \cap (X_m \times Y_m) \neq \emptyset \).

Let \( P : \Sigma^* \rightarrow \Sigma^* \) be the natural projection. Since \( P(s) \in L(S) = \overline{N(A)} \), there exists \( z \in \delta(x_0, P(s)) \). Thus, \( (x, z) \in \xi \times \delta(x_0, z_0, s) \). Since \( B(G \times A) = \emptyset \), we get that
\[
(\exists s' \in \Sigma^*) \xi \times \delta((x, z), s') \cap (X_m \times Z_m) \neq \emptyset.
\]
Thus, \( \xi((x, s') \cap X_m \neq \emptyset \) and \( P(s') \in N(A) \subseteq N(S) \). Since \( S \) is deterministic, \( \eta(y, P(s')) \cap Y_m \neq \emptyset \). Therefore, \( \xi \times \eta((x, y), s') \cap (X_m \times Y_m) \neq \emptyset \) contradicting the fact that \( (x, y) \) is a blocking state. Thus, \( B(G \times S) = \emptyset \).

For each \( s \in L(G \times S) \), \( x \in \xi(x_0, s) \) and \( y \in \eta(y_0, P(s)) \). Since \( A \) is state-controllable, for any \( z \in \delta(x_0, P(s)) \), we have \( E_G(x) \cap \Sigma_{uc} \subseteq E_A(z) \). Since \( E_S(y) = \bigcup_{z \in \delta(x_0, P(s))} E_A(z) \), we have
\[
E_G(x) \cap \Sigma_{uc} \subseteq E_S(y).
\]
Thus, \( S \) is state controllable with respect to \( G \) and \( \Sigma_{uc} \).

Next, we show that \( S \) is state-observable w.r.t. \( G \) and \( P_0 \) if \( A \) is state-observable w.r.t. \( G \) and \( P_0 \). Suppose it is not true. Then there exist \( s, s' \in I(G \times S) \subseteq L(A) \) with \( P_0(s) = P_0(s') \), \( (x, y) \in \xi \times \eta((x_0, y_0), s) \) and \( (x', y') \in \xi \times \eta((x_0, y_0), s') \) such that \( E_G \times_S (x, y) \cap E_G(x') \subseteq E_S(y') \). Since \( S \) is deterministic, there exists \( \sigma \in \Sigma^* \) such that
\[
s \in L(G) \cap s' \in L(G) \cap P(s) \sigma \in L(S) \cap P(s') \sigma \notin L(S).
\]
Since \( L(S) = \overline{N(A)} = L(A) \), we have that there exist \( s, s' \in L(A) \) with \( P_0(s) = P_0(s') \) such that
\[
s \in L(G) \cap s' \in L(G) \cap P(s) \sigma \in L(A) \cap P(s') \sigma \notin L(A).
\]
Pick \( z \in \delta(x_0, P(s)) \) and \( z' \in \delta(x_0, P(s')) \), then \( (x, z) \in \xi \times \delta((x_0, z_0), s) \) and \( (x', z') \in \xi \times \delta((x_0, z_0), s') \). Furthermore, we have that \( \sigma \in E_G \times_S (x, z) \cap E_G(x') \cap \Sigma^* \) but \( \sigma \notin E_A(z') \), namely \( E_G \times_S (x, z) \cap E_G(x') \cap \Sigma^* \subseteq E_A(z') \), which contradicts that \( A \) is state-observable w.r.t. \( G \) and \( P_0 \). Thus, \( S \) is state-observable w.r.t. \( G \) and \( P_0 \).

Finally, we show that \( S \) is state-normal w.r.t. \( G \) and \( P_0 \). Let \( s \in I(G \times S) \) and \( s' \in \overline{P_0^{-1}(P_0(s))} \cap L(G \times S) \). For any \( (x, y) \in \xi \times \eta((x_0, y_0), s') \) and \( s'' \in \Sigma^* \) with \( P_0(s'' s') = P_0(s) \), we need to show that
\[
\xi((x, s'') \neq \emptyset \Rightarrow \eta(y, P(s'')) \neq \emptyset.
\]
which means \( S \) is state-normal w.r.t. \( G \) and \( P_0 \).

5) Proof of Prop. 4: Since \( N(G \times S_i) \subseteq N(G \times H) \) for \( i = 1, 2 \), we have
\[
N(G \times S) = \overline{N(G)} \mid (N(S_1) \cup N(S_2)) = \overline{N(G)} \mid (N(S_1) \cup N(S_2)) \subseteq N(G \times H).
\]
Next, we show \( B(G \times S) = \emptyset \). Let \( G = (X, \Sigma_x, \xi, x_0, X_m) \)
\( S_1 = (Y_1, \Sigma_2, \eta_1, y_0, Y_m) \) and \( S = (Y, \Sigma_2, \eta, y_0, Y_m) \). Suppose \( B(G \times S) \neq \emptyset \). Then there exist \( s \in B(G \times S) \) and \( (x, y) \in \xi \times \eta((x_0, y_0), s) \) such that for all \( s' \in \Sigma^* \nexists \xi \times \eta((x, y), s') \cap (X_m \times Y_m) \neq \emptyset \).

Let \( P : \Sigma^* \rightarrow \Sigma^* \) be the natural projection. Then \( P(s) \in L(S) = \overline{N(S_1)} \cup \overline{N(S_2)} \). Thus, either \( P(s) \in N(S_1) \) or \( P(s) \in N(S_2) \).
Without loss of generality, suppose \( P(s) \in \overline{N(S_0)} \). Then there exists \( y_1 \in \eta_l(y_{10}, P(s)) \), namely \((x, y_1) \in \xi \times \eta_l((x_0, y_{10}), s)\). Since \( B(G \times S_1) = \emptyset \), we have

\[
\exists s' \in \Sigma^* \xi \times \eta_l ((x, y_1), s') \cap (X_m \times X_{1m}) \neq \emptyset.
\]

Thus, \( \xi(x, s') \cap X_m \neq \emptyset \) and \( P(ss') \in N(S_1) \subset N(S) \). Since \( S \) is deterministic, we get \( \eta_l(y_1, P(s')) \cap Y_{1m} \neq \emptyset \). Therefore, \( \xi(x, y_1), s') \cap (X_m \times X_{1m}) \neq \emptyset \) — contradicting the fact that \((x_1, y_1)\) is a blocking state. Thus, \( B(G \times S) = \emptyset \).

For each \( s \in L(G \times S) \), let \( x \in \xi(x_0, s) \) and \( y \in \eta_l(y_{10}, P(s)) \).

Since \( P(s) \in L(S) = N(S_1) \cup N(S_2) \), without loss of generality, suppose \( P(s) \in N(S_1) \) then \( B_1 \) because \( \eta_l(y_{10}, P(s)) \).

Thus, \( S \) is state-controllable with respect to \( G \) and \( S_{\text{unc}} \).

Finally, we show that \( S \) is state-normal with respect to \( G \) and \( P_{o} \). Let \( s \in L(G \times S) \) and \( s' \in P_{o}^{-1}(P_{o}(s)) \cap L(G \times S) \). For any \((x, y) \in \xi \times \eta_l((x_0, y_{10}), s')\), suppose \((s', s'') \in \Sigma^* \) with \( P_{o}(s'') = P_{o}(s) \), then \( s' \in L(S_1) \) or \( \psi \in L(S_1) \).

Thus, \( \eta_l(y_1, P(s'')) = \emptyset \), which contradicts the fact that \( S_1 \) is state-normal with respect to \( G \) and \( P_{o} \).

Thus, \( \xi(x, s'') \neq \emptyset \) or \( \eta_l(y_1, P(s'')) \neq \emptyset \).

Suppose it is not true. Then there exist \( x \in X \) and \( s'' \in \Sigma^* \) such that \( \xi(x, s'') \neq \emptyset \) or \( \eta_l(y_1, P(s'')) \neq \emptyset \). Since \( S \) is deterministic, \( P(s'') \notin L(S) \). Since \( s \in L(G \times S) \), we get that \( P(s) \in L(S_1) \) or \( L(S_2) \). Without loss of generality, suppose \( P(s) \in L(S_1) \). Let \( s \in L(S_1) \) and \( \psi \in L(S_1) \).

Thus, \( \xi(x, s) \neq \emptyset \) or \( \eta_l(y_1, P(s)) \neq \emptyset \).

Suppose \( \xi(x, s) \neq \emptyset \) or \( \eta_l(y_1, P(s)) \neq \emptyset \).

which means \( S \) is state-normal with respect to \( G \) and \( P_{o} \).

\[E_G(x) \cap \Sigma_{\text{unc}} \cap \Sigma' \subseteq E_S(y)\]

where \( \Sigma' \rightarrow \Sigma^* \) is the natural projection, and there exist \( t \in L(G/\Sigma' \times S) \), \( (x', y) \in \xi'((x_0, t), y) \cap \eta_l(y, P(s)) \) such that \( E_{G/\Sigma'\times S}(x') \cap \Sigma_{\text{unc}} \cap \Sigma' \subseteq E_S(y) \).

where \( \xi' \) is the transition map of \( G/\Sigma' \). By the definition of automaton abstraction we have

\[E_{G/\Sigma'\times S}(x') \cap \Sigma_{\text{unc}} \cap \Sigma' \subseteq E_S(y) \]

Thus, \( E_{G/\Sigma'\times S}(x') \cap \Sigma_{\text{unc}} \cap \Sigma' \subseteq E_S(y) \).

From expression (7) we also get that

\[E_{G/\Sigma'\times S}(x') \cap \Sigma_{\text{unc}} \cap \Sigma' \subseteq E_S(y) \]

which contradicts expression (6). Thus, the IF part is true.

Next, we show the ONLY IF part. Suppose it is not true. Then \( S \) is state-controllable w.r.t. \( G/\Sigma' \) and \( \Sigma_{\text{unc}} \), but it is not state-controllable w.r.t. \( G/\Sigma' \) and \( \Sigma_{\text{unc}} \). Thus, for all \( t \in L(G/\Sigma' \times S) \), \( x'' \in \xi''((x_0, t), y) \cap \eta_l(y, P(s)) \) such that

\[E_{G/\Sigma'\times S}(x'') \cap \Sigma_{\text{unc}} \cap \Sigma' \subseteq E_S(y) \]

which contradicts expression (6). Thus, the ONLY IF part is true.

7 Proof of Lemma 2: (1) Let \( G = (X, \Sigma, \xi, x_{00}, X_m) \) and \( S = (Y, \Sigma, \eta_l(y_{10}, Y_m)) \). We first show the IF part. Suppose it is not true. Then \( S \) is state-controllable w.r.t. \( G \) and \( S_{\text{unc}} \) but it is not state-controllable w.r.t. \( G/\Sigma' \) and \( \Sigma_{\text{unc}} \). Thus, for all \( s \in L(G \times S) \), \( x \in \xi(x_0, s) \) and \( y \in \eta_l(y_{10}, P(s)) \).

\[E_G(x) \cap \Sigma_{\text{unc}} \cap \Sigma' \subseteq E_S(y) \]

where \( P : \Sigma^* \rightarrow \Sigma^* \) is the natural projection, and there exist \( t \in L(G/\Sigma'' \times S) \), \( (x', y) \in \xi'((x_0, t), y) \cap \eta_l(y, P(s)) \) such that

\[E_{G/\Sigma''\times S}(x') \cap \Sigma_{\text{unc}} \cap \Sigma' \subseteq E_S(y) \]

Assume that \( S \) is not state-observable w.r.t. \( G \) and \( P_{o} \). Then there are \( t, t' \in L(G \times S) \) with \( P_{o}(t) = P_{o}(t') \), and \( (x, y) \in \xi \times \eta_l((x_0, y_0), t) \) and \( (x', y') \in \xi \times \eta_l((x_0, y_0), t') \) such that

\[E_{G\times S}(x, y) \cap E_G(x') \cap \Sigma' \subseteq E_S(y') \]

Since \( G \) is standardized, we get \( (x) \in \xi''((x_0, t), P(t)) \) and \( (x') \in \xi'((x_0, P(t'))) \). We also have \( y \in \eta_l(y_{10}, P(t)) \) and \( y' \in \eta_l(y_{10}, P(t')) \).
Suppose $S$ is not state-normal w.r.t. $G$ and $P_0$. Then there exist $t \in L(G \times S)$ and $t' \in \mathcal{P}_0^{-1}(P_0(t)) \cap L(G \times S)$ such that there exist $(x,y) \in \xi \times \eta((x_0,y_0),P(t))$ and $t'' \in \Sigma^*$ such that

$$P_0(t'') = P_0(t) \land \eta((x,t'') \neq \emptyset \land \eta(y,P(t'')) = \emptyset.$$  

Let $P_0^* : \Sigma_0 \rightarrow (\Sigma_0 \cap \Sigma^*)$ be the nature projection. Since $G$ is guaranteed, we have $P_0(t) \in L(G/\approx_{\Sigma_0}) \times S$. From $(x,y) \in \xi \times \eta((x_0,y_0),P(t'))$, we can derive that $((x,y),P(t')) \in \xi \times \eta(((x_0,y_0),P(t'))$. Since $t'' \in \mathcal{P}_0^{-1}(P_0(t)) \setminus L(G \times S)$, we have $P_0(t'') \leq P_0(t)$ and $P_0(t) \in L(G/\approx_{\Sigma_0}) \times S$, which means $P_0^*(P(t')) = P_0^*(P_0(t)) \leq P_0^*(P_0(t)) = P_0^*(P(t)).$ Therefore, $P(t') \in \mathcal{P}_0^{-1}(P_0^*(s)) \cap L(G/\approx_{\Sigma_0}) \times S$. Since $P_0(t''t') = P_0(t)$, we have

$$P_0^*(P(t''t')) = P_0^*(P_0(t''t')) = P_0^*(P_0(t)) = P_0^*(P_0(t)).$$

Since $\xi(t'',t') \neq \emptyset$, we have $P(t'') = \emptyset$ if $P(t'') = \emptyset$ and we have $\xi(t'',t') \neq \emptyset$. Thus, there exist $s = P_0(t) \in L(G/\approx_{\Sigma_0}) \times S$, $s' = P_0(t') \in \mathcal{P}_0^{-1}(P_0^*(s)) \cap L(G/\approx_{\Sigma_0}) \times S$, and $t'' \in \Sigma^*$ such that there exist $(x,y) \in \xi \times \eta(((x_0,y_0),P(t'))$, and $s'' = t'' \in \Sigma^*$ which contradicts expression (13). Thus, (1) is true.

(2) Let $S$ be state-normal w.r.t. $G$ and $P_0$. Then for any $t \in L(G \times S)$, $t' \in \mathcal{P}_0^{-1}(P_0(t)) \cap L(G \times S)$, and $t'' \in \Sigma^*$ such that $P_0^*(s'') = P_0^*(s) \land \eta((x_0,y_0),s) \neq \emptyset$ and $s'' \neq \emptyset$, we have $P_0^*(s'') = P_0^*(s) \land \eta((x_0,y_0),s') \neq \emptyset$.

8) Proof of Lemma 3: (1) Let $G = (X,\Sigma,\xi;x_0,X_M)$ and $S = (Y,\Sigma_0,\eta_0,Y_M)$. Suppose $S$ is state-normal w.r.t. $G/\approx_{\Sigma_0}$ and $P_0^*$. Then for any $s \in L(G/\approx_{\Sigma_0}) \times S$, $s' \in \mathcal{P}_0^{-1}(P_0^*(s)) \cap L(G/\approx_{\Sigma_0}) \times S$, we have, for each $(x,y) \in \xi \times \eta(((x_0,y_0),s'),$ and $s'' \in \Sigma^*$

$$P_0^*(s'') = P_0^*(s) \Rightarrow \xi((x,t''),s'') \neq \emptyset \land \eta(y,P(t'')) = \emptyset.$$  

which contradicts expression (11). Thus, (2) is true.

REFERENCES


Jan H. van Schuppen (M’73) is affiliated with Centrum voor Wiskunde en Informatica (CWI), Amsterdam, The Netherlands, and as Full Professor with the Department of Mathematics, Delft University of Technology (part time), Delft, The Netherlands. He is Editor-in-Chief of Mathematics of Control, Signals, and Systems and was Department Editor of Discrete Event Dynamic Systems. His research interests include control of hybrid systems and of discrete-event systems, stochastic control, realisation, and system identification. In applied research his interests include engineering problems of control of motorway traffic, of communication networks, and control and system theory for the life sciences.

Dr. van Schuppen was Associate Editor-at-Large of the IEEE TRANSACTIONS AUTOMATIC CONTROL.

Rong Su received the M.A.Sc. and Ph.D. degrees in electrical engineering from the University of Toronto, Toronto, ON, Canada, in 2000 and 2004, respectively.

He is currently affiliated with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. His research interests include modeling, fault diagnosis and supervisory control of discrete-event dynamic systems.

Dr. Su has been a member of IFAC technical committee on discrete event and hybrid systems (TC 1.3) since 2005.

Jacobus E. Rooda (M’90) received the M.Sc. degree from Wageningen University of Agriculture Engineering, Wageningen, The Netherlands and the Ph.D. degree from Twente University, Enschede, The Netherlands.

Since 1985, he has been a Professor of (manufacturing) systems engineering with the Department of Mechanical Engineering, Eindhoven University of Technology, Eindhoven, The Netherlands. His research fields of interest are modeling and analysis of manufacturing systems. His interest is especially in control of (high-tech) manufacturing lines and in supervisory control of high-tech (manufacturing) machines.