Control of Nondeterministic Discrete-Event Systems for Bisimulation Equivalence

Changyan Zhou, *Student Member, IEEE*, Ratnesh Kumar, *Senior Member, IEEE*, and Shengbing Jiang

***Abstract*—Most prior work on supervisory control of discrete event systems is for achieving *deterministic specifications*, expressed as formal languages. In this paper we study supervisory control for achieving *nondeterministic specifications*. Such specifications are useful when designing a system at a higher level of abstraction so that lower level details of system and its specification are omitted to obtain higher level models that may be nondeterministic. Nondeterministic specifications are also meaningful when the system to be controlled has a nondeterministic model due to the lack of information (caused for example by partial observation or unmodeled dynamics). Language equivalence is not an adequate notion of behavioral equivalence for nondeterministic systems, and instead we use the finest known notion of equivalence, namely the bisimulation equivalence. Choice of bisimulation equivalence is also supported by the fact that bisimulation equivalence specification is equivalent to a specification in the temporal logic of μ-calculus that subsumes the complete branching-time logic CTL*. Given nondeterministic models of system and its specification, we study the design of a supervisor (possibly nondeterministic) such that the controlled system is bisimilar to the specification. We obtain a small model theorem showing that a supervisor exists if and only if it exists over a certain finite state space, namely the power set of Cartesian product of system and specification state spaces. Also, the notion of state-controllability is introduced as part of a necessary and sufficient condition for the existence of a supervisor. In the special case of deterministic systems, we provide an existence condition that can be verified polynomially in both system and specification states, when the existence condition holds.**

***Index Terms*—Bisimulation equivalence, controllability, discrete-event systems (DESs), nondeterministic systems, supervisory control.**

I. Introduction

DISCRETE-event systems (DESs) are systems with discrete states and are event-driven. Communication networks, manufacturing systems, and traffic systems are examples of DESs. In supervisory control [33], a DES is modeled as an automaton and its behavior is represented by the language of the automaton. A controller, called a supervisor, is also modeled as another automaton that exercises control by operating in synchrony with the system under control. The control objective is to ensure that the language of the controlled system is as desired.

Nondeterminism in plant model can arise from unmodeled dynamics or abstraction. A nondeterministic plant can have a language or a finer specification such as: Failures [11], refusal-trace (same as trajectory) [31], [10], ready-trace [4], simulation and bisimulation equivalences [29]. Also, the supervisors can be deterministic as well as nondeterministic.

The control of nondeterministic plant subject to language specification is studied in [37], [21], and [22], where plant is modeled using the trajectory model. In [30] and [9], both plant and specification are nondeterministic and are represented using failures and trajectory models, respectively. The authors in [9] showed how to transform their control problem of nondeterministic setting to one of deterministic setting with an added partial observability. Control of plants modeled using nondeterministic state machines for language specification is also studied in [19] and [15]. All these work used deterministic supervisors.

The use of nondeterministic supervisors for specification represented using language model was explored in [13] and [38]. The notion of nondeterministic control was formalized in [20] and used for control under partial observation for language specification, and the notion of achievability (a property weaker than controllability and observability combined) was introduced. The problem of finding a nondeterministic supervisor so that it's parallel-composition with plant conforms to a deterministic specification (via language containment) is studied in [41]. Nondeterministic supervisors were also used in [16] where nondeterministic specification was specified in the temporal logic of CTL*, generalizing the work reported in [1] which used CTL to express specification. Other work related to control subject to temporal logic-based specification include [23]–[25], [2], and [34].

In general, plant, specification, and supervisor all can be nondeterministic. Nondeterministic plant and specification are useful when designing a system at a higher level of abstraction so that lower level details of system and its specification are omitted to obtain higher level models that are nondeterministic. Nondeterministic specifications are also meaningful when the system to be controlled has a nondeterministic model due to the lack of information (caused for example by partial observation or unmodeled dynamics). For nondeterministic systems numerous notions of behavioral equivalence that are finer than the language equivalence have been proposed ([40] provides a classification of these equivalences). In this paper, we study the control of nondeterministic plants subject to the requirement

Manuscript received May 25, 2004; revised March 21, 2005 and November 17, 2005. Recommended by Associate Editor A. Giua. The work was supported in part by the National Science Foundation under Grants NSF-ECS-9709796, NSF-ECS-0099851, NSF-ECS-0218207, NSF-ECS-0244732, NSF-EPNES-0323379, and NSF-0424048, and in part by a DoD-EPSCoR grant through the Office of Naval Research under Grant N000140110621.

C. Zhou and R. Kumar are with the Department of Electrical and Computer Engineering, Iowa State University, Ames, IA 50011 USA (e-mail: czhou@iastate.edu; rkumar@iastate.edu).

S. Jiang is with General Motors R&D and Planning, Warren, MI 48090-9055 USA (e-mail: shengbing.jiang@gm.com).

Digital Object Identifier 10.1109/TAC.2006.875036

of bisimulation equivalence with respect to a nondeterministic specification using nondeterministic supervisors.

Bisimulation equivalence was first introduced in communicating systems by Milner [28]. A bisimulation equivalence specification is equivalent to a specification in the temporal logic of μ-calculus that subsumes the complete branching-time logic CTL* [8]. So if a supervisor is designed to ensure that the controlled system is bisimilar to a specification system, then this is equivalent to ensuring that the controlled system satisfies the same μ-calculus or CTL* specification that is satisfied by the specification system. On the other hand, a language equivalence based control only guarantees the satisfaction of a linear temporal logic (LTL) specification which is a strict subclass of μ-calculus and CTL*.

Control for achieving CTL* specification was studied by Jiang and Kumar in [14], under the assumption that plant model is deterministic. [3] studied the synthesis of controllers for deterministic plants subject to μ-calculus based specifications under partial observation, where the observation mask is restricted to be projection type. The control problem is solved by reduction to a discrete-event game problem, and explicit conditions for the existence of a supervisor are not provided. In this paper, we allow both plant and specification models to be nondeterministic. Furthermore, our approach is quite different: In [14], the control problem was reduced to a decision problem of CTL*, whereas our results are based on the properties of the automata models of the plant and the specification. Given nondeterministic models of plant and its specification, we study the design of a supervisor (possibly nondeterministic) such that the controlled system is bisimilar to the specification system.

The input–output model matching control studied in [7] also uses the notion of simulation, and as shown in [6] it can be casted as an instance of standard supervisory control problem of deterministic setting. Bisimulation relation has been used as a technique for supervisory control of deterministic systems subject to language equivalence in [36], [18], [5], [27], and [17]. In [36], [17], the controllability and observability is characterized as a bisimulation type relation. [32] studied the problem of synthesizing a supervisor so that the controlled system is bisimilar to a deterministic specification. The event set of the system and specification need not be same, and all events are treated controllable. [26] studied control for bisimulation equivalence for a partial specification (defined over an "external event set"). The plant is taken to be deterministic and all events are treated controllable. Further it is required that all events treated indistinguishable from the partial specifications point of view be either all enabled or all disabled at a state. Such a requirement does not make sense in supervisory control context. The author of [39] studied the controller synthesis problem for deterministic plants subject to a possibly nondeterministic partial specification such that the controlled system is bisimulation equivalent to the specification. This is the same problem as that studied in [26] except the aforementioned control requirement is removed.

To summarize the contribution of our work, we study a more general bisimulation equivalence control problem, namely, in which both the plant as well as the specification are nondeterministic. No prior work addresses this problem in this generality—they impose determinism either on the plant or on the specification. To understand the nature of the problem when both the plant and the specification are nondeterministic, note that even when the specification is the same as the plant (and so trivially bisimilar to the plant), the specification itself may not be work as a supervisor, for the composition of two of the same system need not be bisimilar to the system itself. Note this complication does not arise when either the plant or the specification is deterministic, since in this case the specification (assumed without loss of generality to be plant-simulated and state-controllable) itself can be chosen as a supervisor. This is because the composition of plant and specification will be bisimilar to the specification (when plant is deterministic), or language equivalent to the specification (when specification is deterministic). In the more general case, the option of choosing the specification itself as a supervisor is not necessarily available, which introduces a nontrivial added complexity to the nature of the control problem.

Our main result is a small model theorem showing that a supervisor for enforcing bisimulation equivalence between the specification and the controlled system exists if and only if it exists over a certain finite state space, namely the power set of Cartesian product of the plant and the specification state spaces. Also, a stronger notion of controllability, called state-controllability, is introduced as part of the necessary and sufficient condition for the existence of such a supervisor. State-controllability is stronger than the "language-controllability," where the latter is a property of language models, and the former is a property of the automata models. We present an algorithm of linear complexity for testing state-controllability matching the complexity of testing the language-controllability.

For the special case of deterministic plants we obtain a necessary and sufficient condition for the existence of a bisimilarity enforcing supervisor which can be verified linearly in both the plant and the specification sizes. Moreover, when the existence conditions are satisfied, the complexity of synthesizing a supervisor is linear in the size of the specification. These happen to be the same as the complexity of verifying the existence and of performing the synthesis of a supervisor when both the plant and the specification are deterministic.

The rest of this paper is organized as follows. Section II presents an illustrative example. Section III presents the required notation and a preliminary background. Section IV develops the theory for bisimulation equivalence control and presents the small model theorem. Section V presents a polynomial test for state-controllability, a condition that appears in our small model theorem. Section VI discusses the special case of deterministic plants, and shows the polynomiality of the solution in this case. Conclusions are given in Section VII, and in Appendix A examples are given to show some differences with the work in [26].

II. An Example

In this section, we give an example to illustrate some of the issues that are prevalent when controlling a nondeterministic system.

Example 1: Consider an automatic check-out scanner in a shopping center, a state machine model G of which is shown in

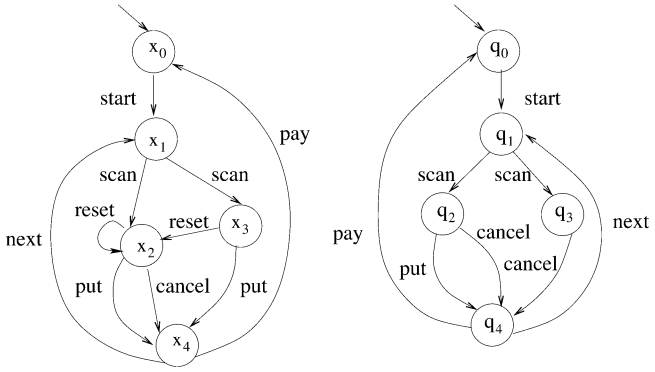

Fig. 1. Plant G (left) and specification R (right).

Fig. 1. Initially, a customer presses the start button to start the check-out process. Then it scans an item, upon which, owing to a malfunction, the scanner nondeterministically transitions to one of two states. In the first state, the scanner allows the customer to either put the item in a bag, or cancel; whereas in the second state the only option offered is to put the item in the bag. Not giving an option to cancel in the second state is unacceptable. A reset button may be pressed in either of the states to return to the first state. After this, the scanner waits for either a request for a next item, or if there is no more items then a request to pay. In the latter case, scanner returns to its initial state, and in the former case it goes back to the state from where check-out process resumes. Since a customer must pay at the end of the check-out process, the event "pay" is deemed uncontrollable. All other events are controllable.

The partial specification R, also shown in Fig. 1, is given in order to restrict the plant to exhibit only an acceptable behavior. According to the specification, after start and scan, two possible states may be reached nondeterministically. In both states, cancel is an available option which is what we desire of the scanner, while put is an additional option at the first state. The rest of the behavior is the same as the one feasible in the scanner. Note that the "reset" event does not appear in the specification state machine since an occurrence or nonoccurrence of it is immaterial to the specification. This implies that the specification R is for the plant G projected on to the event set $\hat{\Sigma} := \Sigma - \{\text{reset}\}$, denoted $G{\uparrow}\hat{\Sigma}$. (Projecting an automaton onto $\hat{\Sigma}$ replaces each event label outside $\hat{\Sigma}$ of the automaton by ϵ.)

Note that $L(R) = L(G \uparrow \hat{\Sigma})$, i.e., $G \uparrow \hat{\Sigma}$ is language equivalent to R. Thus if we use language equivalence as a notion of behavioral equivalence, then there is no need to control. However, as mentioned previously, G can exhibit some behavior that is not acceptable (i.e., not always giving the option to cancel after scan). We develop a theory in this paper that lets us design a supervisor S such that $(G\|S)\uparrow\hat{\Sigma}$ is bisimilar to R.

III. Notation and Preliminaries

Automata are used to model discrete event systems at the logical level. A nondeterministic automaton is a 5-tuple [12], [33], $G = (X, \Sigma, \alpha, X_0, X_m)$, where X is the set of states, Σ is the alphabet of events, $\alpha : X \times \overline{\Sigma} \to 2^X$ is the state transition function, where $\overline{\Sigma} := \Sigma \cup \{\epsilon\}$ with ϵ being a label for "silent" transitions, $X_0 \subseteq X$ is the set of initial states, and $X_m \subseteq X$ is the set of final states. Σ^* denotes the set of all finite sequences of events in Σ, called event-traces, and includes the zero length trace, denoted ϵ. The ϵ-closure (denoted as $\epsilon^*(\cdot)$) of $x \in X$ is the set of states reached by the execution of a sequence of ϵ-transitions from state x. By using ϵ-closure map, we can extend the definition of transition function from events to traces, $\alpha^* : X \times \Sigma^* \to 2^X$, which is defined inductively as:

$$\forall x \in X, \alpha^*(x, \epsilon) := \epsilon^*(x);$$
$$\forall s \in \Sigma^*, \sigma \in \Sigma : \alpha^*(x, s\sigma) := \epsilon^*(\alpha(\alpha^*(x, s), \sigma)).$$

The language generated (respectively, marked) by G, is denoted as $L(G)$ (respectively, $L_m(G)$). $L(G)$ is the sequence of events generated starting from the initial state, i.e., $L(G) = \{s \in \Sigma^* \mid \alpha^*(X_0, s) \neq \emptyset\}$, and $L_m(G)$ is the set of generated sequences that end in a marked state, i.e., $L_m(G) = \{s \in L(G) \mid \alpha^*(X_0, s) \cap X_m \neq \emptyset\}$. For $x \in X$, we define

$$\Sigma(x) := \{\sigma \in \Sigma \mid \alpha(x, \sigma) \neq \emptyset\}$$

to denote the set of events defined at state x.

A DES, called a plant, is controlled to restrict its behavior so as to prevent any undesirable behavior by dynamically disabling certain controllable events [33]. Such a controller is called a supervisor. The supervisor can be modeled as another automaton operating in synchronous composition with the plant. The synchronous composition of two automata is defined as follows.

Definition 1: The *synchronous composition* of two automata G_1 and G_2, where $G_1 = (X_1, \Sigma, \alpha_1, X_{01}, X_{m1})$ and $G_2 = (X_2, \Sigma, \alpha_2, X_{02}, X_{m2})$, is the automaton

$$G_1\|G_2 = (X_1 \times X_2, \Sigma, \alpha, X_{01} \times X_{02}, X_{m1} \times X_{m2})$$

where for $x_1 \in X_1, x_2 \in X_2, \sigma \in \overline{\Sigma}$

$$\alpha((x_1, x_2), \sigma) = \begin{cases} \alpha_1(x_1, \sigma) \times \alpha_2(x_2, \sigma), & \text{if } \sigma \neq \epsilon \\ (\alpha_1(x_1, \epsilon) \times \{x_2\}) \cup (\{x_1\} \times \alpha_2(x_2, \epsilon)), & \text{if } \sigma = \epsilon \end{cases}.$$

We next introduce the notion of behavioral equivalence based on bisimulation.

Definition 2: Given $G_1 = (X_1, \Sigma, \alpha_1, X_{01}, X_{m1})$, $G_2 = (X_2, \Sigma, \alpha_2, X_{02}, X_{m2})$, a *simulation relation* is a binary relation $\Phi \subseteq X_1 \times X_2 \subseteq (X_1 \cup X_2)^2$ such that $(x_1, x_2) \in \Phi$ implies

1) $\sigma \in \overline{\Sigma}, x_1' \in \alpha_1^*(x_1, \sigma) \Rightarrow \exists x_2' \in \alpha_2^*(x_2, \sigma)$, such that $(x_1', x_2') \in \Phi$;
2) $x_1 \in X_{m1} \Rightarrow x_2 \in X_{m2}$.

We write $x_1 \sqsubseteq_\Phi x_2$ to denote that there exists a simulation relation Φ with $(x_1, x_2) \in \Phi$, read as x_1 is simulated by x_2. We sometimes omit the subscript Φ from $\sqsubseteq_\Phi$ when it is clear from the context.

A bisimulation relation is a symmetric simulation relation which is captured by the following definition.

Definition 3: Given two automata G_1 and G_2 as defined above, a *bisimulation relation* is a binary relation $\Phi \subseteq (X_1 \cup X_2)^2$ such that for $x_1 \in X_1, x_2 \in X_2, (x_1, x_2) \in \Phi$ implies

1) $\sigma \in \overline{\Sigma}, x_1' \in \alpha_1^*(x_1, \sigma) \Rightarrow \exists x_2' \in \alpha_2^*(x_2, \sigma)$, such that $(x_1', x_2') \in \Phi$;

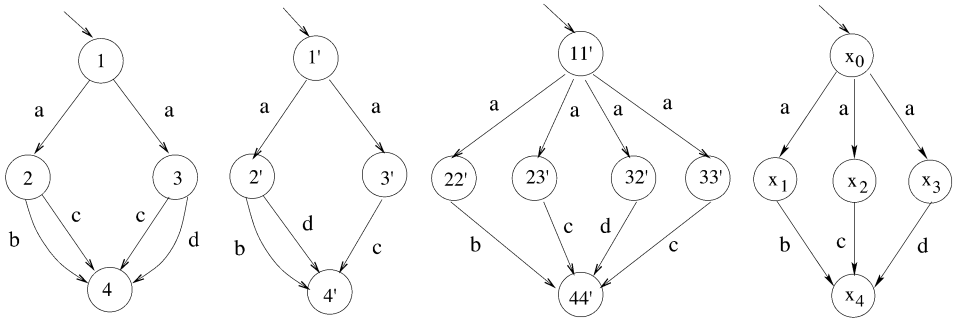

Fig. 2. G_1 (first), G_2 (second), $G_3 = G_1 \| G_2$ (third), and G (fourth).

2) $\sigma \in \overline{\Sigma}, x_2' \in \alpha_2^*(x_2, \sigma) \Rightarrow \exists x_1' \in \alpha_1^*(x_1, \sigma)$, such that $(x_1', x_2') \in \Phi$;
3) $x_1 \in X_{m1} \Leftrightarrow x_2 \in X_{m2}$.

We write $x_1 \simeq_\Phi x_2$ to denote that there exists a bisimulation relation Φ with $(x_1, x_2) \in \Phi$, read as x_1 is bisimilar to x_2. We sometimes omit the subscript Φ from $\simeq_\Phi$ when it is clear from the context. From the definition of bisimulation relation and simulation relation, we easily observe that $x_1 \simeq_\Phi x_2$ if and only if $x_1 \sqsubseteq_\Phi x_2$, $x_2 \sqsubseteq_\Phi x_1$ and Φ is symmetric. Next, we give the definition of simulation and bisimulation relation between two automata.

Definition 4: Given G_1 and G_2 as defined previously, G_1 is simulated by G_2 (denoted as $G_1 \sqsubseteq_\Phi G_2$) if there exists a simulation relation $\Phi \subseteq X_1 \times X_2 \subseteq (X_1 \cup X_2)^2$ such that

$$x_{01} \in X_{01} \Rightarrow \exists x_{02} \in X_{02}, \text{ such that } (x_{01}, x_{02}) \in \Phi$$

i.e., $X_{01} \sqsubseteq_\Phi X_{02}$. Further G_1 and G_2 are said to be bisimilar (denoted as $G_1 \simeq_\Phi G_2$) if Φ is symmetric so that $X_{01} \simeq_\Phi X_{02}$.

It is easily observed that given two automata G_1 and G_2, $G_1 \simeq_\Phi G_2$ if and only if $G_1 \sqsubseteq_\Phi G_2$, $G_2 \sqsubseteq_\Phi G_1$ and Φ is symmetric. Next, we give an example to illustrate these concepts.

Example 2: Consider two automata G_1 and G_2 shown in Fig. 2. Their synchronous composition $G_3 = G_1 \| G_2$ is shown in Fig. 2. Note that $L(G_1) = L(G_2) = L(G_3)$. However, $G_3 \not\simeq G_1 \not\simeq G_2$. Thus although synchronous composition preserves "language equivalence," it may not preserve bisimulation equivalence.

Consider another automaton G shown in Fig. 2, then there exists a simulation relation $\Phi_1 \subseteq Q \times Q_3$:

$$\Phi_1 = \{(x_0, 11), (x_1, 22), (x_2, 23), (x_2, 33), (x_3, 32), (x_4, 44)\}.$$

Thus, $G \sqsubseteq_{\Phi_1} G_3$. Also, there exists a simulation relation $\Phi_2 \subseteq Q_3 \times Q$:

$$\Phi_2 = \{(11, x_0), (22, x_1), (23, x_2), (33, x_2), (32, x_3), (44, x_4)\}.$$

Thus, $G_3 \sqsubseteq_{\Phi_2} G$. Therefore, there exists a symmetric simulation relation $\Phi \subseteq (Q_3 \cup Q)^2$ given by

$$\Phi = \Phi_1 \cup \Phi_2.$$

So, we conclude $G_3 \simeq_\Phi G$.

Our construction of a bisimilarity enforcing control requires merger of states of a certain automaton as defined below.

Definition 5: We use $\overline{G_{\langle x,x' \rangle}} = (\overline{X}, \Sigma, \overline{\alpha}, \overline{X}_0, \overline{X}_m)$ to denote $G = (X, \Sigma, \alpha, X_0, X_m)$ in which two states $x, x' \in X$ are merged. Use $\langle x, x' \rangle$ to denote the merger of two states x and x'. Then

$$\overline{X} = (X - \{x, x'\}) \cup \{\langle x, x' \rangle\}$$

and $\forall \hat{x}, \tilde{x} \in X - \{x, x'\}, \forall \sigma \in \overline{\Sigma}$

$$\begin{aligned}
&x \in \alpha(\hat{x}, \sigma) \text{ in } G \Rightarrow \langle x, x' \rangle \in \overline{\alpha}(\hat{x}, \sigma) \text{ in } \overline{G_{\langle x,x' \rangle}} \\
&x' \in \alpha(\tilde{x}, \sigma) \text{ in } G \Rightarrow \langle x, x' \rangle \in \overline{\alpha}(\tilde{x}, \sigma) \text{ in } \overline{G_{\langle x,x' \rangle}}; \\
&\hat{x} \in \alpha(x, \sigma) \text{ in } G \Rightarrow \hat{x} \in \overline{\alpha}(\langle x, x' \rangle, \sigma) \text{ in } \overline{G_{\langle x,x' \rangle}}; \\
&\tilde{x} \in \alpha(x', \sigma) \text{ in } G \Rightarrow \tilde{x} \in \overline{\alpha}(\langle x, x' \rangle, \sigma) \text{ in } \overline{G_{\langle x,x' \rangle}}; \\
&\tilde{x} \in \alpha(\hat{x}, \sigma) \text{ in } G \Rightarrow \tilde{x} \in \overline{\alpha}(\hat{x}, \sigma) \text{ in } \overline{G_{\langle x,x' \rangle}}; \\
&x \in \alpha(x', \sigma) \text{ in } G \Rightarrow \langle x, x' \rangle \in \overline{\alpha}(\langle x, x' \rangle, \sigma) \text{ in } \overline{G_{\langle x,x' \rangle}}; \\
&x' \in \alpha(x, \sigma) \text{ in } G \Rightarrow \langle x, x' \rangle \in \overline{\alpha}(\langle x, x' \rangle, \sigma) \text{ in } \overline{G_{\langle x,x' \rangle}}.
\end{aligned}$$

It is known (see [35]) that merger of bisimilar states in an automaton yields a bisimilar automaton.

Theorem 1: [35] Given an automaton G, if $x, x' \in X$ are such that $x \simeq x'$, then $G \simeq \overline{G_{\langle x,x' \rangle}}$.

IV. Supervisory Control for Bisimilarity

In this section, we study the control of a nondeterministic plant to ensure bisimilarity of the controlled plant and the given specification. The set of events is partitioned into *uncontrollable* and *controllable events*: $\Sigma = \Sigma_u \cup (\Sigma - \Sigma_u)$. The events in $\Sigma - \Sigma_u$ can be disabled when desired, while those in Σ_u are events that the supervisory controller cannot disable. This is ensured by requiring the supervisor to be Σ_u-compatible. Unless otherwise stated, we will use $G = (X, \Sigma, \alpha, X_0, X_m)$, $R = (Q, \Sigma, \delta, Q_0, Q_m)$, and $S = (Y, \Sigma, \beta, Y_0, Y_m)$ to denote the plant, the specification, and the supervisor, respectively. The controlled system is denoted by $G \| S = (X \times Y, \Sigma, \gamma, X_0 \times Y_0, X_m \times Y_m)$.

Definition 6: A supervisor S is said to be Σ_u-*compatible* if each uncontrollable event is defined at each state of S.

In the deterministic setting, the controllability of specification language $L(R)$ with respect to plant language $L(G)$ and uncontrollable event set Σ_u is defined as

$$L(R)\Sigma_u \cap L(G) \subseteq L(R).$$

This definition of "language-controllability" requires the following extension to the nondeterministic setting where instead of language models, automata models are used for plant and specification.

Definition 7: Given plant automaton G and specification automaton R with $L(R) \subseteq L(G)$, we say R is *state-controllable with respect to G and Σ_u* if

$$s \in L(R), \sigma \in \Sigma_u \text{ such that } s\sigma \in L(G) \Rightarrow \forall q \in \delta^*(Q_0, s), \sigma \in \Sigma(q).$$

R is state-controllable with respect to G if for trace s in $L(R)$ and uncontrollable event σ defined at *some* state reachable by s in G, σ is defined at *all* states reachable by s in R. Clearly, state-controllability implies language-controllability; the converse need not hold.

The following lemma establishes a type of equivalence between Σ_u-compatibility and state-controllability.

Lemma 1: Suppose S is state-controllable with respect to G and Σ_u. Define S' as S augmented with self-loops at each state on undefined uncontrollable events at the state. Then, S' is Σ_u-compatible and $G||S \simeq G||S'$.

Proof: Since S is state-controllable, for any state (x, y) of $G||S$ such that x has uncontrollable events defined, S also has those events defined at y. Therefore, adding self-loops at each state on undefined uncontrollable events in S does not change the result of synchronous composition. It follows that $G||S = G||S'$. Thus, $G||S \simeq G||S'$. ■

Before we give the main result of this section, we first give some preliminary results.

Lemma 2: For G, S, and R defined as before, consider $x \in X, y \in Y, q \in Q$.

1) If $q \sqsubseteq_\Phi (x, y)$, then $q \sqsubseteq_{\Phi'} x$ and $q \sqsubseteq_{\Phi''} y$.
2) If $q \sqsubseteq_{\Phi'} x$ and $q \sqsubseteq_{\Phi''} y$, then $q \sqsubseteq_\Phi (x, y)$.

Proof:

1) $q \sqsubseteq_\Phi (x, y)$ implies there exists a simulation relation Φ such that $(q, (x, y)) \in \Phi$. Also any pair $(q', (x', y')) \in \Phi$ implies

$$\sigma \in \overline{\Sigma}, q'_\sigma \in \delta^*(q', \sigma) \Rightarrow \exists (x'_\sigma, y'_\sigma) \in \gamma^*((x', y'), \sigma) \text{ and } (q'_\sigma, (x'_\sigma, y'_\sigma)) \in \Phi.$$
$$q' \in Q_m \Rightarrow (x', y') \in X_m \times Y_m.$$

By Definition 1

$$\sigma \in \overline{\Sigma}, (x'_\sigma, y'_\sigma) \in \gamma^*((x', y'), \sigma) \Rightarrow x'_\sigma \in \alpha^*(x', \sigma).$$
$$(x', y') \in X_m \times Y_m \Rightarrow x' \in X_m.$$

Define a relation $\Phi' = \{(q', x') \mid (q', (x', y')) \in \Phi\}$. Then, $(q', x') \in \Phi'$ implies

$$\sigma \in \overline{\Sigma}, q'_\sigma \in \delta^*(q', \sigma) \Rightarrow \exists x'_\sigma \in \alpha^*(x', \sigma) \text{ such that } (q'_\sigma, x'_\sigma) \in \Phi'.$$
$$q' \in Q_m \Rightarrow x' \in X_m.$$

Clearly, Φ' is simulation relation. By Definition 2, $q \sqsubseteq_{\Phi'} x$. Similarly, we can prove $q \sqsubseteq_{\Phi''} y$.

2) $q \sqsubseteq_{\Phi'} x$ implies there exists a simulation relation Φ' with $(q, x) \in \Phi'$. Also, for any pair $(q', x') \in \Phi'$

$$\sigma \in \overline{\Sigma}, q'_\sigma \in \delta^*(q', \sigma) \Rightarrow \exists x'_\sigma \in \alpha^*(x', \sigma) \text{ such that } (q'_\sigma, x'_\sigma) \in \Phi'.$$
$$q' \in Q_m \Rightarrow x' \in X_m.$$

Similarly, $q \sqsubseteq_{\Phi''} y$ implies there exists a simulation relation Φ'' with $(q, y) \in \Phi''$. Also for any pair $(q', y') \in \Phi''$

$$\sigma \in \overline{\Sigma}, q'_\sigma \in \delta^*(q', \sigma) \Rightarrow \exists y'_\sigma \in \beta^*(y', \sigma) \text{ such that } (q'_\sigma, y'_\sigma) \in \Phi''.$$
$$q' \in Q_m \Rightarrow y' \in Y_m.$$

Define a relation $\Phi = \{(q', (x', y')) \mid (q', x') \in \Phi' \text{ and } (q', y') \in \Phi''\}$. Then $(q', (x', y')) \in \Phi$ implies $\forall \sigma \in \overline{\Sigma}, \forall q'_\sigma \in \delta^*(q', \sigma)$

$$\exists (x'_\sigma, y'_\sigma) \in \alpha^*(x', \sigma) \times \beta^*(y', \sigma) \text{ such that } (q'_\sigma, (x'_\sigma, y'_\sigma)) \in \Phi.$$
$$q' \in Q_m \Rightarrow (x', y') \in X_m \times Y_m.$$

Thus, $q \sqsubseteq_\Phi (x, y)$. ■

The following corollary follows from Lemma 2 and serves as a necessary condition for the existence of a supervisor for enforcing bisimulation equivalence.

Corollary 1: Given G, R and S, if $G||S \simeq R$, then $R \sqsubseteq G$.

Proof: $G||S \simeq R$ implies $R \sqsubseteq G||S$. By Definition 4, $Q_0 \sqsubseteq (X_0, Y_0)$. By Lemma 2, $Q_0 \sqsubseteq X_0$. Then by Definition 4, $R \sqsubseteq G$. ■

Remark 1: For a deterministic G and any R, it can be verified that $R \sqsubseteq G$ is equivalent to $L(R) \subseteq L(G)$ and $L_m(R) \subseteq L_m(G)$.

Next, we present our main result on the existence of a supervisor S for plant G such that $G||S$ is bisimilar to the specification R.

Theorem 2: Given nondeterministic G and R, there exists a Σ_u-compatible supervisor S such that $G||S \simeq R$ if and only if there exists a state-controllable automaton T with state space $2^{X \times Q}$ such that $G||T \simeq R$.

Proof:

($\Rightarrow$): Given G, R, and S such that $G||S \simeq R$, we construct a Σ_u-compatible T such that $G||T \simeq R$ and state space of T is $2^{X \times Q}$. We assume without loss of generality that all transitions of S participate in the composition with G. If a transition of S never participates in $G||S$, then we can remove this transition from S, and call the result as $\langle S \rangle$. Then $G||\langle S \rangle = G||S$ and every transition of $\langle S \rangle$ participates in the composed automaton $G||\langle S \rangle$. We compute T from G, R and S as follows.

1) For $y \in Y$, we define

$$X_{\text{syn}}(y) := \{x \in X \mid \exists s \in \Sigma^*, x \in \alpha^*(X_0, s) \text{ and } y \in \beta^*(Y_0, s)\}$$

to be the states in G that are reachable by a same trace as is the state y of S. In other words, $x \in X_{\text{syn}}(y)$ if and only if (x, y) is a reachable state in $G||S$. Since $G||S \simeq R$, each such state (x, y) is bisimilar to some state $q \in Q$. Collection of all such states is denoted as $Q_{\text{sim}}(y)$, i.e.,

$$Q_{\text{sim}}(y) := \{q \in Q \mid \exists x \in X_{\text{syn}}(y), (x, y) \simeq q\}.$$

2) Label each state y of S by $lbl(y) \subseteq X \times Q$, such that $(x,q) \in lbl(y)$ if and only if $x \in X_{\rm syn}(y), q \in Q_{\rm sim}(y)$ and $(x,y) \simeq q$.
3) Define $[S]_0 := S$. For $k \geq 0$, $[S]_{k+1}$ is obtained by merging two states of $[S]_k$ carrying the same label, stop when $[S]_k = [S]_{k+1} =: [S]$.

Define T to be $[S]$. Then each state of T carries a unique label that is an element of $2^{X\times Q}$, and so the state space of T can be thought to be $2^{X\times Q}$. Next, we prove that $G||[S] \simeq R$ by induction on the number of mergers.

Base case: $[S]_0 = S$. So $G||[S]_0 = G||S \simeq R$.

Induction step: Suppose at step k, $G||[S]_k \simeq R$. Denote the state of $[S]_k$ as $y^{(k)}$ with label $lbl(y^{(k)})$. At step $k+1$ of merging, suppose we merge $y^{(k)}, y'^{(k)}$, where $lbl(y^{(k)}) = lbl(y'^{(k)})$. Now we prove $G||[S]_{k+1} \simeq G||[S]_k$. Merging $y^{(k)}$ and $y'^{(k)}$ causes the merger of states $(x,y^{(k)})$ and $(x,y'^{(k)})$ of $G||[S]_k$ for all $x \in X_{\rm syn}(y^{(k)})$. Define automata $P_1, \ldots P_{|X_{\rm syn}(y^{(k)})|}$ as follows.

1) $n := 0$, $P_n := G||[S]_k$, $X_n := X_{\rm syn}(y^{(k)})$.
2) If $X_n \neq \emptyset$, then $P_{n+1} := \overline{P_n}_{\langle (x,y^{(k)}),(x,y'^{(k)})\rangle}$, where $x \in X_n$, $X_{n+1} := X_n - \{x\}$, $n := n+1$; else stop, and $G||[S]_{k+1} = P_n$.

Since $lbl(y^{(k)}) = lbl(y'^{(k)})$, it follows that both $(x,y^{(k)})$ and $(x,y'^{(k)})$ are bisimilar to same state of Q, i.e., $(x,y^{(k)}) \simeq (x,y'^{(k)})$. From the repeated application of Theorem 1 it follows that $G||[S]_k = P_0 \simeq P_1 \simeq \cdots \simeq P_n = G||[S]_{k+1}$. This proves the induction step. It remains to show that $[S]$ is Σ_u-compatible. Since S is Σ_u-compatible, and since Σ_u-compatibility is preserved under state mergers, $[S] = T$ is Σ_u-compatible. Since Σ_u-compatibility implies state-controllability, T is state-controllable.

($\Leftarrow$): Define S as T augmented with self-loops at each state on all undefined uncontrollable events at the state (as in Lemma 1). Then S is Σ_u-compatible and the result follows from Lemma 1. ∎

Remark 2: Note that $G_1 \simeq G_2$ implies G_1 is trim (a marked state can be reached from every reachable state) if and only if G_2 is trim. So, bisimilarity of controlled system and specification implies that the supervisor is nonblocking if and only if the specification automaton is trim. In other words, requiring a bisimilarity enforcing supervisor to be nonblocking is equivalent to requiring that the supervisor be bisimilarity enforcing and specification be trim. Thus, there is no need to separately study nonblocking control in context of bisimulation equivalence specification (all that is needed is the specification automaton be trim).

Remark 3: From Theorem 2, an exhaustive search can be performed to determine the existence of a supervisor S over the state space $2^{X\times Q}$, the complexity of which is $O(2^{2^{|X|\times|Q|}})$. Since there may exist more systematic ways of searching for a desired S, the tightness of the upper bound complexity remains open.

In the following example, we illustrate the computation of T through labeling states of S by $lbl(\cdot) \in 2^{X\times Q}$ as in the proof of Theorem 2. Next, we also show that labeling states of S just by $Q_{\rm sim}(\cdot) \in 2^Q$ and using such labels to perform mergers of the states of S can yield a $[S]$ for which $G||[S] \simeq R$ need not hold. This example illustrates that it may not be possible to replace the labeling function $lbl(\cdot)$ used in the proof of Theorem 2 by something simpler such as $Q_{\rm sim}(\cdot)$.

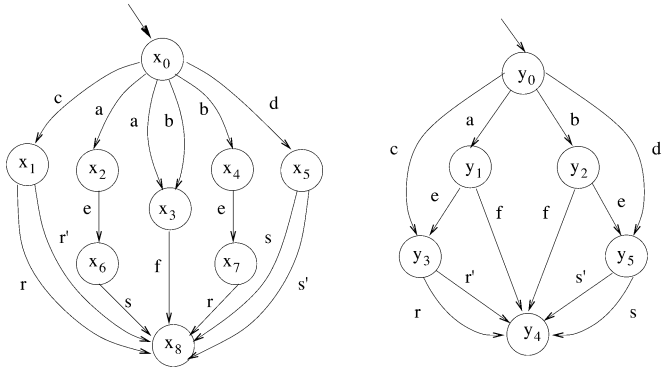

Fig. 3. Plant G (left) and supervisor S (right).

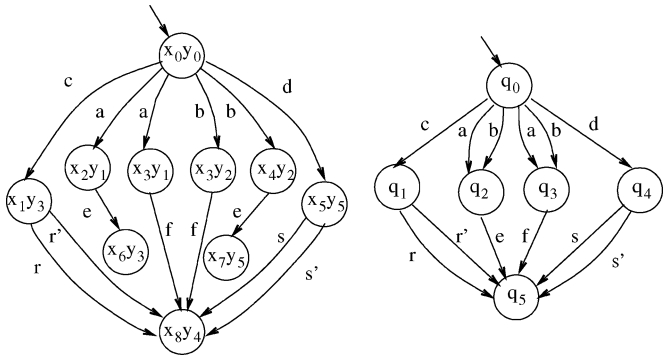

Fig. 4. Controlled system $G||S$ (left) and specification R (right).

Example 3: Consider nondeterministic G and S as shown in Fig. 3.

$G||S$ and R are shown in Fig. 4, and we can easily see that $G||S \simeq R$. By step 1), we compute $X_{\rm syn}(y)$:

$$X_{\rm syn}(y_0) = \{x_0\} \quad X_{\rm syn}(y_1) = \{x_2, x_3\}$$
$$X_{\rm syn}(y_2) = \{x_3, x_4\} \quad X_{\rm syn}(y_3) = \{x_1, x_6\}$$
$$X_{\rm syn}(y_5) = \{x_5, x_7\} \quad X_{\rm syn}(y_4) = \{x_8\}.$$

Since

$$(x_0,y_0) \simeq q_0, (x_2,y_1) \simeq q_2, (x_3,y_1) \simeq q_3, (x_3,y_2) \simeq q_3$$
$$(x_4,y_2) \simeq q_2, (x_1,y_3) \simeq q_1, (x_6,y_3) \simeq q_5, (x_8,y_4) \simeq q_5$$
$$(x_5,y_5) \simeq q_4, (x_7,y_5) \simeq q_5$$

$Q_{\rm sim}(y)$ is computed as:

$$Q_{\rm sim}(y_0) = \{q_0\}, Q_{\rm sim}(y_1) = \{q_2, q_3\}, Q_{\rm sim}(y_2) = \{q_2, q_3\}$$
$$Q_{\rm sim}(y_3) = \{q_1, q_5\}, Q_{\rm sim}(y_4) = \{q_5\}, Q_{\rm sim}(y_5) = \{q_4, q_5\}.$$

Then by step 2), label of each y is given by:

$$lbl(y_0) = \{(x_0, q_0)\} \quad lbl(y_1) = \{(x_2, q_2), (x_3, q_3)\}$$
$$lbl(y_2) = \{(x_3, q_3), (x_4, q_2)\}$$
$$lbl(y_3) = \{(x_1, q_1), (x_6, q_5)\}$$
$$lbl(y_4) = \{(x_8, q_5)\} \quad lbl(y_5) = \{(x_5, q_4), (x_7, q_5)\}.$$

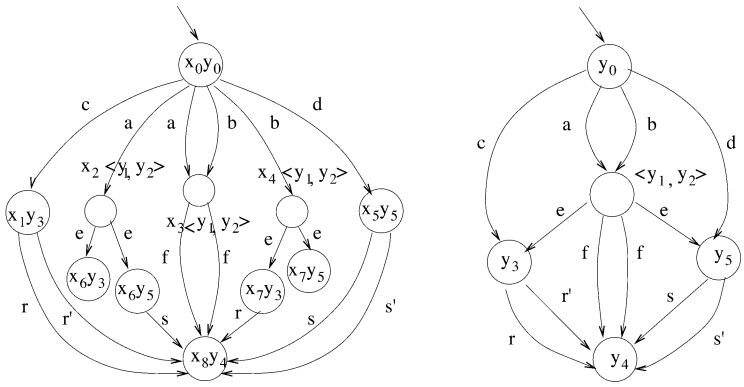

Fig. 5. $G||[S]$ (left) and $[S]$ (right).

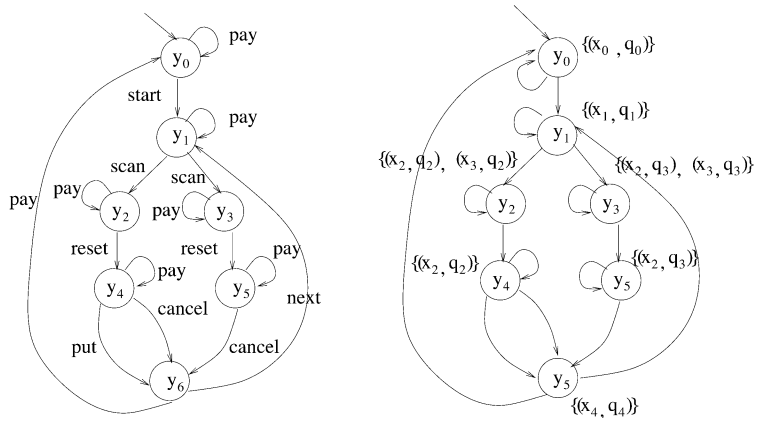

Fig. 6. Supervisor S (left) and labeling of its states (right).

No states can be merged since no states have the same label, so step 3) simply yields $T = S$.

In contrast, if we label each y of S by $Q_{\mathrm{sim}}(y)$, and merge two states y and y' having the same label, then since $Q_{\mathrm{sim}}(y_1) = Q_{\mathrm{sim}}(y_2)$, we merge y_1 and y_2. The resulting state machine $[S]$ is shown in Fig. 5. The synchronous composition $G||[S]$ is shown in Fig. 5. It can be seen that $G||[S] \not\simeq R$, since the states $(x_2, \langle y_1, y_2\rangle)$ and $(x_4, \langle y_1, y_2\rangle)$ of $G||[S]$ are bisimilar to no state of R.

Now, we revisit the motivating example.

Example 4: We need to find a Σ_u-compatible supervisor S such that $(G||S) \uparrow \hat{\Sigma} \simeq R$, where $\hat{\Sigma} = \Sigma - \{\mathrm{reset}\}$. Such a supervisor S is shown in Fig. 6. The synchronous composition of G and S is drawn in Fig. 7. The following bisimulation relation Φ exists between $(G||S) \uparrow \hat{\Sigma}$ and R:

$$\begin{aligned}\Phi = \{&(x_0y_0, q_0), (x_1y_1, q_1), (x_2y_2, q_2), (x_2y_3, q_3), (x_3y_2, q_2)\\ &(x_3y_3, q_3), (x_2y_4, q_2), (x_2y_5, q_3), (x_4y_6, q_4), (q_0, x_0y_0)\\ &(q_1, x_1y_1), (q_2, x_2y_2), (q_2, x_3y_2), (q_3, x_2y_3), (q_3, x_3y_3)\\ &(q_2, x_2y_4), (q_3, x_2y_5), (q_4, x_4y_6)\}.\end{aligned}$$

Thus, the controlled system is bisimilar to the specification with respect to $\hat{\Sigma}$. States in S can be labeled by elements of $2^{X\times Q}$ as guaranteed by Theorem 2 (shown in Fig. 6). A state $(x, q) \in X\times Q$ belongs to the label of a state $y \in Y$ of S if (x, y) appears in $G||S$ (i.e., exists a common trace from x_0 to x in G and from y_0 to y in S so (x, y) is a reachable state of $G||S$), and

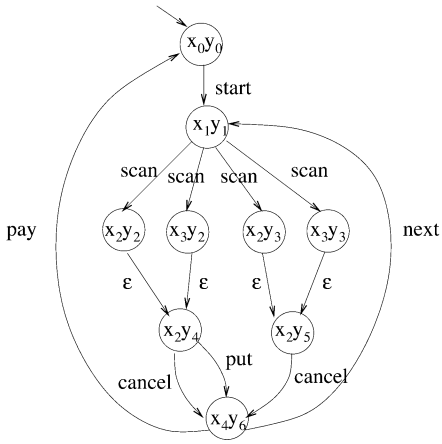

Fig. 7. Controlled system $(G||S) \uparrow \hat{\Sigma}$.

(x, y) is bisimilar to state q of R. All states of S with identical labels may be merged to obtain the state machine T stated in Theorem 2. T is same as S in this case, and so $G||T = G||S$.

V. Test for State-Controllability

A condition in Theorem 2 is that of state-controllability. We present here an algorithm of polynomial complexity for verifying state-controllability of an automaton R with respect to a plant G.

Algorithm 1: Algorithm for testing state-controllability of $R = (Q, \Sigma, \delta, Q_0)$ with respect to $G = (X, \Sigma, \alpha, X_0)$.

1) Construct $\overline{R}$ by augmenting R with a new state called *dump*, and by adding transitions at each state of R on each undefined uncontrollable event at that state to the *dump* state, i.e., $\overline{R} = (Q \cup \{dump\}, \Sigma, \overline{\delta}, Q_0)$, where

$$\forall q \in Q, \sigma \in \Sigma:$$
$$\overline{\delta}(q,\sigma) = \begin{cases} \delta(q,\sigma), & \text{if } \sigma \in \Sigma(q) \\ \text{dump}, & \text{if } \sigma \in \Sigma_u - \Sigma(q) \end{cases}$$
$$\overline{\delta}(q,\epsilon) = \delta(q,\epsilon).$$

2) Obtain $G||\overline{R}$.

3) R is state-controllable with respect to G if and only if there does not exist $\overline{x} \in X$ such that $(\overline{x}, dump)$ is reachable in $G||\overline{R}$.

The following lemma is needed in order to prove Algorithm 1.

Lemma 3: R is state-controllable with respect to G if and only if

$$\forall q \in Q, \forall x \in X_{\text{syn}}(q) : \Sigma(x) \cap \Sigma_u \subseteq \Sigma(q) \cap \Sigma_u. \quad (1)$$

Proof:

($\Leftarrow$): For each state $q \in Q$ of R define $X_{\text{syn}}(q) \subseteq X$ to be the set of states of G that are reachable by a common trace, i.e.,

$$X_{\text{syn}}(q) := \{x \in X \mid \exists s \in L(R) \cap L(G) \text{ s.t. } q \in \delta^*(Q_0, s), \text{ and } x \in \alpha^*(X_0, s)\}.$$

Pick $s \in L(R), x \in \alpha^*(X_0, s)$ such that $\sigma \in \Sigma(x) \cap \Sigma_u$. Then, for any $q \in \delta^*(Q_0, s)$, $x \in X_{\text{syn}}(q)$ and so from hypothesis, $\sigma \in \Sigma(q) \cap \Sigma_u$. It follows that R is state-controllable with respect to G.

($\Rightarrow$): Pick $q \in Q, x \in X_{\text{syn}}(q), \sigma \in \Sigma(x) \cap \Sigma_u$. It suffices to show that $\sigma \in \Sigma(q)$. Since $x \in X_{\text{syn}}(q)$, exists $s \in L(R)$ such that $q \in \delta^*(Q_0, s)$ and $x \in \alpha^*(X_0, s)$. Then, from state-controllability, $\sigma \in \Sigma(q)$. ∎

The following theorem establishes the correctness of Algorithm 1.

Theorem 3: Algorithm 1 is correct.

Proof: Let $(x, q) \in X \times Q$ be a state reachable in $G||R$. Then it is obvious that $(x, q) \in X_{\text{syn}}(q) \times \{q\}$, and so for state-controllability to hold condition of (1) must hold. On the other hand, if this condition is violated, i.e., if exists $\sigma \in \Sigma(x) \cap \Sigma_u - \Sigma(q)$, then a transition $(x, \sigma, \overline{x})$ for some $\overline{x} \in X$ is defined in G and the transition $(q, \sigma, dump)$ is defined in $\overline{R}$. So the transition $((x, q), \sigma, (\overline{x}, dump))$ is defined in $G||\overline{R}$. It follows that a state $(\overline{x}, \text{dump})$ is reachable in $G||\overline{R}$ if and only if R is not state-controllable with respect to G. ∎

Remark 4: Since G and R are nondeterministic, their number of transitions is $O(|X|^2)$ and $O(|Q|^2)$, respectively. So the complexity of constructing $G||\overline{R}$ is $O(|X|^2 \times |Q|^2)$, and the complexity of checking the reachability of $(\cdot, \text{dump})$ in $G||\overline{R}$ is also $O(|X|^2 \times |Q|^2)$. So the complexity of the algorithm for testing state-controllability of R with respect to G is $O(|X|^2 \times |Q|^2)$, i.e., it is quadratic in the number of states of both G and R (equivalently, linear in size of G and R).

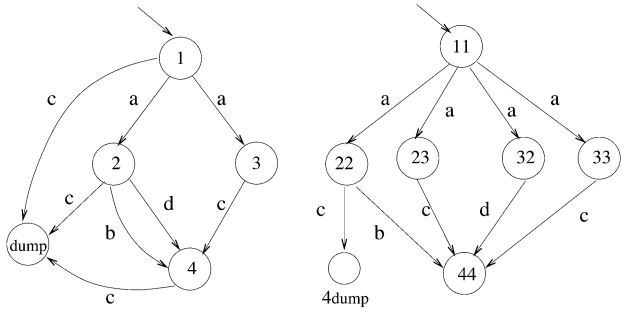

Fig. 8. $\overline{R}$ (left) and $G||\overline{R}$ (right).

Example 5: Consider the automata G_1 and G_2 shown in Fig. 2. Set $G = G_1$ and $R = G_2$, and suppose event c is uncontrollable. Then, following the definition of the state-controllability, it is obvious that R is not state-controllable with respect to G ($a \in L(G) \cap L(R), ac \in L(G)$, but c is not defined at y_1, one of the states reached by the execution of a in R).

Now, we verify the state-controllability of R with respect to G by our algorithm. The constructed $\overline{R}$ and $G||\overline{R}$ are depicted in Fig. 8. It can be seen that the state (4, *dump*) is reachable in $G||\overline{R}$. Thus, from Algorithm 1, R is not state-controllable with respect to G. It should be noted that $L(G) = L(R) = pr(ab+ac+ad)$, and so $L(R)$ is language-controllable with respect to $L(G)$.

VI. Specialization to Deterministic Plant

In the earlier sections, we studied the supervisory control problem for enforcing bisimilarity in the setting of nondeterministic plants. In this section, we study the specialized case when plant is deterministic and specification is (possibly) nondeterministic and show that now the problem can be polynomially solved. Suppose exists Σ_u-compatible S such that $G||S \simeq R$. Due to Σ_u-compatibility of S, we find that $G||S$ is state-controllable. The bisimilarity of R with $G||S$ however does not guarantee the state-controllability of R since as shown by the following example the state-controllability may not be preserved under the bisimilarity.

Example 6: Consider the plant G and two specifications R_1 and R_2 shown in Fig. 9. Let $\Sigma_u = \{b\}$. It can be verified that $R_1 \simeq R_2$, and R_2 is state-controllable with respect to G. However, R_1 is not state-controllable since in G, $b \in \Sigma_u$ is defined after $\epsilon \in L(R)$, but in R, b is not defined at state $1 \in \delta^*(1, \epsilon)$.

The reason for the nonpreservation of the state-controllability under bisimulation is the nonpreservation of the set of events defined at a pair of bisimilar states. In the following, we introduce the notion of R^* which is the NSM R with its states renamed and transition function δ replaced by δ^*. We show that R^* possesses the properties that $R^* \simeq R$ and further, if R is state-controllable then R^* remains state-controllable.

Definition 8: Given $R = (Q, \Sigma, \delta, Q_0, Q_m)$, we define $R^* = (Q^*, \Sigma, \delta^*, Q_0^*, Q_m^*)$, where

- $Q^* := \{q^* \mid q \in Q\}$;
- $\forall q^* \in Q^*$: $\delta^*(q^*, \epsilon) := \epsilon^*(q)$, $\delta^*(q^*, \sigma) := \epsilon^*(\delta(\epsilon^*(q), \sigma))$;
- $Q_0^* := \{q^* \mid q \in Q_0\}$;
- $Q_m^* := \{q^* \mid q \in Q_m\}$.

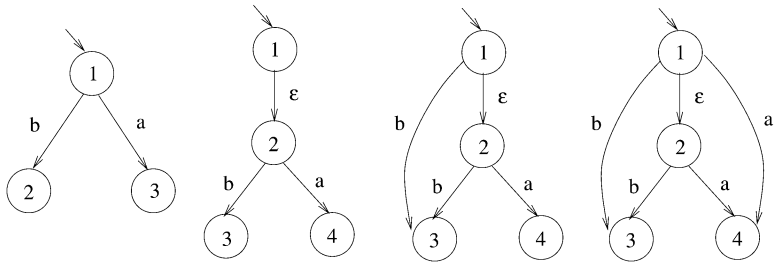

Fig. 9. G (first), R_1 (second), R_2 (third), and R_2^* (fourth).

Note that in the absence of ϵ-transitions, $R^* = R$.

Fig. 9 shows an example illustrating Definition 8.

Since the definition of bisimilarity between two NSM's $G_i = (X_i, \Sigma, \alpha_i, X_{0i}, X_{mi})$ $(i = 1, 2)$ depends on the transition function α_i^* (and not α_i), it is obvious that any two NSMs R and R^* are bisimilar. This and another property is summarized in the following lemma.

Lemma 4: Consider NSMs R and R^*. Then

1) $R \simeq R^*$;
2) $\forall q \in Q$: $\Sigma(q) \subseteq \Sigma(q^*)$.

Proof:

1) It is easy to see that we can pick $\Phi := \{(q, q^*), (q^*, q) \mid q \in Q\}$ to establish $R \simeq R^*$.
2) Follows from the fact that $\delta(q, \sigma) \subseteq \delta^*(q^*, \sigma)$ for all $q \in Q$ and $\sigma \in \Sigma$. ■

Prior to obtaining the main result of this section, we need to prove the following lemmas.

Lemma 5: Given a deterministic plant G, a possibly nondeterministic R, if $R \sqsubseteq G$, then $G||R \simeq R$.

Proof: Choose

$$\Phi := \{((x, q), q), (q, (x, q))|(x, q) \text{ state in } G||R \text{ s.t. } q \sqsubseteq x\}.$$

Since $R \sqsubseteq G$, $L(R) \subseteq L(G)$, and so for each $s \in L(R)$ and $q \in \delta^*(Q_0, s)$, exists singleton $\{x\} = \alpha^*(X_0, s)$ such that $q \sqsubseteq x$. So for each $q \in Q$, exists a unique $x \in X$ such that $((x, q), q) \in \Phi$. Since Φ is symmetric, it suffices to show that it is a simulation relation. Pick $(q, (x, q)) \in \Phi$ and σ-successor $q' \in \delta^*(q, \sigma)$ for some $\sigma \in \overline{\Sigma}$. Since $q \sqsubseteq x$, there exists x' such that $\{x'\} = \alpha(x, \sigma)$, $q' \sqsubseteq x'$ and (x', q') is a state in $G||R$. It follows that $(q', (x', q')) \in \Phi$. Similarly, pick $((x, q), q) \in \Phi$ and σ-successor $(x', q') \in \alpha^*(x, \sigma) \times \delta^*(q, \sigma)$ for some $\sigma \in \overline{\Sigma}$. Then, (x', q') is a state in $G||R$. Also, since $x' \in \alpha^*(x, \sigma)$ is unique and $q \sqsubseteq x$, $q' \sqsubseteq x'$. It follows that $((x', q'), q') \in \Phi$ as desired.

Further, if q is marked, $q \sqsubseteq x$ implies that x is marked. I.e., if q is marked, then (x, q) is marked. Moreover, if q is not marked, then (x, q) is not marked. So, for $x \in X, q \in Q$ such that $q \sqsubseteq x$, $(x, q) \in X_m \times Q_m \Leftrightarrow q \in Q_m$.

Finally, $R \sqsubseteq G$ implies $Q_0 \sqsubseteq X_0$, which further implies $((x_0, q_0), q_0), (q_0, (x_0, q_0)) \in \Phi$ for $q_0 \in Q_0$ and $x_0 \in X_0$. Thus, by definition of bisimulation equivalence, $G||R \simeq R$. ■

Lemma 6: For two bisimilar automata $R_1 \simeq R_2$, if R_1 is state-controllable with respect to G, then R_2^* is state-controllable with respect to G.

Proof: Since R_1 is state-controllable, from Lemma 3, $q_1 \in \delta_1^*(Q_{01}, s)$ and $x \in X_{\text{syn}}(q_1)$ implies

$$\Sigma(x) \cap \Sigma_u \subseteq \Sigma(q_1) \cap \Sigma_u. \tag{2}$$

To prove that R_2^* is state-controllable with respect to G, it suffices to show that for each state q_2^* of R_2^* and for each $x \in X_{\text{syn}}(q_2^*)$ it holds that

$$\Sigma(x) \cap \Sigma_u \subseteq \Sigma(q_2^*) \cap \Sigma_u.$$

Since (2) holds, it suffices to show that $\Sigma(q_1) \subseteq \Sigma(q_2^*)$. Since $x \in X_{\text{syn}}(q_2^*)$, exists a trace $s \in L(R_2^*)$ such that $q_2^* \in \delta_2^*(Q_{02}, s)$ and $x \in \alpha^*(X_0, s)$. Bisimilarity of R_1 and R_2 implies $R_1 \simeq R_2^*$ (from Lemma 4). From the fact that $R_1 \simeq R_2^*$ (which is equivalent to $Q_{01} \simeq Q_{02}^*$), we know $s \in L(R_1) = L(R_2^*)$ and exists a state $q_1 \in \delta_1^*(Q_{01}, s)$ such that $q_1 \simeq q_2^*$ (obtained by inductively extending the definition of bisimulation equivalence from one step to multiple steps).

From Lemma 4, $R_1 \simeq R_2$ implies $R_1^* \simeq R_2^*$. Since

$$\Sigma(q_i^*) = \{\sigma \in \Sigma \mid \delta_i^*(q_i^*, \sigma) \neq \emptyset\} (i = 1, 2)$$

$R_1^* \simeq R_2^*$ implies $\Sigma(q_1^*) = \Sigma(q_2^*)$. From Lemma 4, $\Sigma(q_1) \subseteq \Sigma(q_1^*)$ and so we have $\Sigma(q_1) \subseteq \Sigma(q_2^*)$ as desired. ■

The following lemma holds for a deterministic plant but not in general.

Lemma 7: Let G be a deterministic plant, and S be a Σ_u-compatible supervisor. Then $G||S$ is state-controllable.

Proof: To prove state-controllability of $G||S$ with respect to G, we apply the test of Algorithm 1 to $G||\overline{(G||S)}$. Since G is deterministic if a state $(x, (x', y))$ is reached in $G||(G||S)$, then $x = x'$. We claim that there does not exist a state $(x, (x, y))$ reachable in $G||\overline{(G||S)}$ from where a state $(\cdot, \text{dump})$ in $G||\overline{(G||S)}$ can be reached. For this to hold, every uncontrollable event defined at x in G must also be defined at (x, y) in $G||S$, i.e., we need to show that $\Sigma(x) \cap \Sigma_u$ is a subset of $\Sigma(x, y) \cap \Sigma_u$. This follows since

$$\Sigma(x) \cap \Sigma_u = \Sigma(x) \cap \Sigma(y) \cap \Sigma_u = \Sigma(x, y) \cap \Sigma_u$$

where in the first equality we have used the Σ_u-compatibility of S, which implies $\Sigma_u \subseteq \Sigma(y)$. This completes the proof. ■

Now, we are ready to present a necessary and sufficient condition for the existence of a bisimilarity enforcing supervisor for a deterministic plant.

Theorem 4: Given a deterministic plant G and a possibly nondeterministic specification R, there exists a Σ_u-compatible supervisor S such that $G||S \simeq R$ if and only if $R \sqsubseteq G$ and R^* is state-controllable.

Proof:

$(\Leftarrow)$: Choose $S = R_u^*$ (where R_u^* is obtained by adding in R^* self-loops at each state on undefined uncontrollable events). Then S is Σ_u-compatible. Also from Lemma 1, $G||S = G||R_u^* = G||R^*$. Further since $R^* \sqsubseteq G$ (since $R^* \simeq R$ and $R \sqsubseteq G$), and G is deterministic, from Lemma 5, $G||R^* \simeq R^*$. And so $G||S = G||R^* \simeq R^* \simeq R$.

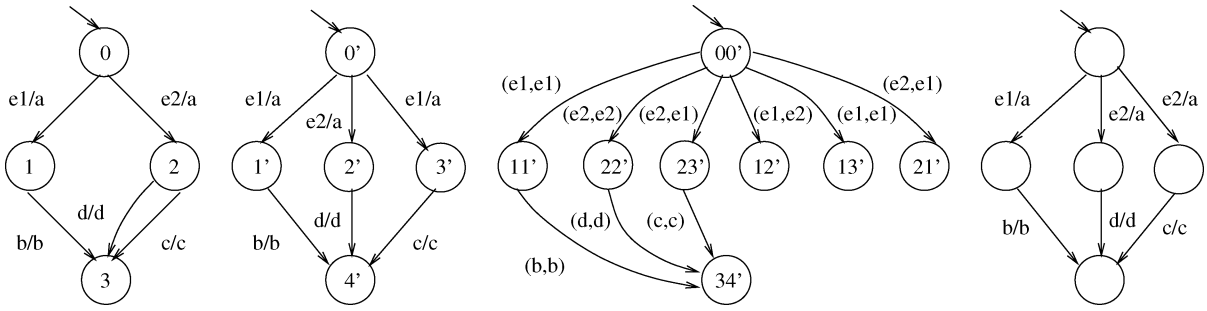

Fig. 10. Plant TS_p (first), spec TS_s (second), $G_{ps} = TS_p \| TS_s$ (third), TS_c (fourth).

($\Rightarrow$): Since S is Σ_u-compatible, from Lemma 7, $G\|S$ is state-controllable. Also, since $R \simeq G\|S$, it follows from Lemma 6 that R^* is state-controllable. Finally, since $G\|S \simeq R$, $R \sqsubseteq G\|S$, which implies that $R \sqsubseteq G$. ■

Remark 5: The statement of Theorem 4 above is a slight refinement of the one appearing in the conference version of [42, Th. 4]. The two statements are identical when there are no ϵ-transitions (so that $R^* = R$).

Remark 6: From Theorem 4, the complexity of checking the existence of a supervisor for enforcing bisimilarity for a deterministic plant is $O(|X| \times |Q|^2)$, which is linear in the sizes of the plant and the specification. Moreover, when the existence conditions are satisfied, R_u^* serves as a supervisor, i.e., the complexity of synthesizing a supervisor is linear in the size of the specification. It is interesting to note that when specification is deterministic but the plant is nondeterministic, the complexity of existence as well as synthesis is again polynomial [20]. In contrast, the situation seems to be different when both the plant and the specification are nondeterministic.

VII. Conclusion

In this paper, we extended the prior work on supervisory control in deterministic as well as nondeterministic setting by allowing both plant and specification to be nondeterministic, and requiring control specification to be bisimulation equivalence of specification and controlled plant. It is known that the language equivalence is not an adequate notion of equivalence for nondeterministic systems, whereas the bisimulation equivalence is the finest known notion of behavioral equivalence. Choice of bisimulation equivalence is also supported by the fact that bisimulation equivalence specification is equivalent to a specification in the temporal logic of μ-calculus that subsumes the complete branching-time logic of CTL*. We obtained a small model theorem showing that a supervisor exists if and only if it exists over a certain finite state space. We extended the controllability concept in language setting to a stronger notion, called state-controllability, as part of the necessary and sufficient condition for the existence of a supervisor and provided a test of complexity linear in the plant and the specification sizes for it. We also obtained a necessary and sufficient condition for the existence of a bisimilarity enforcing supervisor for the special case of deterministic plants. In this case, the complexity of verifying existence using our condition is linear in the size of the plant and the specification. Moreover, when the existence conditions are satisfied, the complexity of synthesizing a supervisor is also linear in the size of the specification.

Appendix
Supporting Examples

Here, we describe the bisimulation equivalence control studied in [26] and demonstrate through examples some of the differences with our work and also the limitations of the requirement of determinism. The control problem studied in [26] can be stated as follows:

> Given *deterministic* plant TS_p and *deterministic* specification TS_s, *empty* set of uncontrollable events ($\Sigma_u = \emptyset$), and a *"partial specification" mask* φ, does there exists a supervisor TS_c such that $\varphi(TS_p \| TS_c) \simeq \varphi(TS_s)$, *with the requirement* that at any state an event e is enabled if and only if all events e' with $\varphi(e') = \varphi(e)$ are enabled.

The first example shows the limitation of requiring plant and specification to be deterministic in [26]. In this example, the specification is nondeterministic. If we apply the algorithm in [26] no supervisor is found. Yet a supervisor exists.

Example 7: Consider deterministic plant TS_p and nondeterministic specification TS_s shown in Fig. 10 with $\varphi(e_1) = \varphi(e_2) = a$, $\varphi(b) = b$, $\varphi(c) = c$, and $\varphi(d) = d$. G_{ps} (as defined in [26]) is shown in Fig. 10, and it can be verified that it does not possess a strong subgraph (defined in [26]) implying that the algorithm is unable to find a supervisor. However there does exist a supervisor TS_c as shown in Fig. 10.

Next example shows the limitation of [26] due to its restriction to supervisors that are deterministic. The example has deterministic plant and specification. By the algorithm in [26], there exists no supervisor. Yet a nondeterministic supervisor exists.

Example 8: Consider deterministic plant TS_p and deterministic specification TS_s shown in Fig. 11 with $\varphi(e_1) = \varphi(e_2) = \varphi(e_3) = a$, $\varphi(b) = b$, $\varphi(c) = c$, and $\varphi(d) = d$. G_{ps} (as defined in [26]) is shown in Fig. 11. By condition (BS3) of the definition of strong subgraph, G_{ps} does not contain a strong subgraph, meaning the algorithm given in [26] does not find a (deterministic) supervisor. However, the nondeterministic supervisor $\overline{TS_c}$ shown in Fig. 11 works.

The previous example illustrates that allowing nondeterminism in the supervisor provides us additional capability of control. In contrast, allowing nondeterminism in the plant and the specification (resulting for example from labeling) can make it harder to control, i.e., while the given plant and specification possess a supervisor, the labled versions of plant

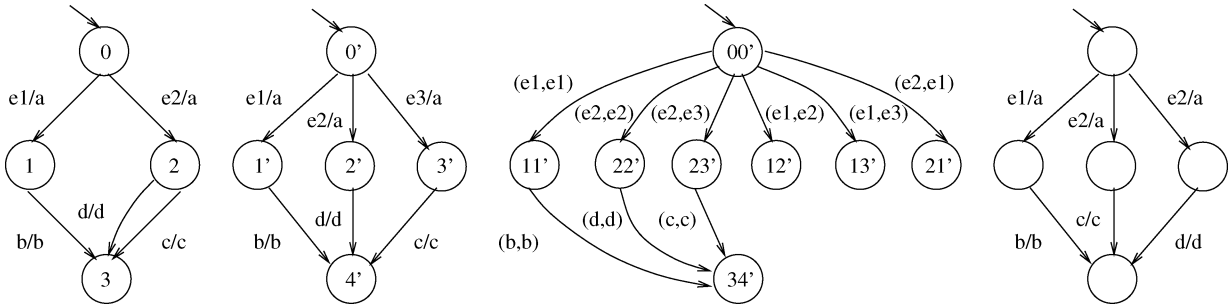

Fig. 11. Plant TS_p (first), spec TS_s (second), $G_{ps} = TS_p \| TS_s$ (third), $\overline{TS_c}$ (fourth).

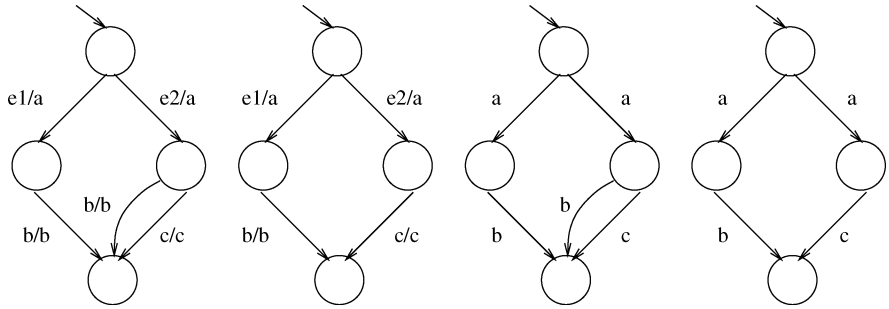

Fig. 12. TS_p (first), TS_s (second), G (third), R (fourth).

and specification may not possess a supervisor (since labeling may introduce nondeterminism). This is illustrated by the next example.

Example 9: Consider plant TS_p and specification TS_s shown in Fig. 12 with $\varphi(e_1) = \varphi(e_2) = a$, $\varphi(b) = b$, and $\varphi(c) = c$. It can be shown that TS_s itself can serve as a supervisor, i.e., $\varphi(TS_p \| TS_s) \simeq \varphi(TS_s)$. In contrast for $G = \varphi(TS_p)$ and $R = \varphi(TS_s)$ (also shown in Fig. 12) there is no supervisor S such that $G \| S \simeq R$. This is because labeling of TS_p yields a nondeterministic G, and any supervisor S that only observes the labeling would be unable to determine which of the two branches the plant initially executed.

REFERENCES

[1] M. Antoniotti, "Synthesis and verification of discrete controllers for robotics and manufacturing devices with temporal logic and control-D systems," Ph.D. dissertation, Dept. Comp. Sci., New York Univ., New York, 1995.

[2] A. Arnold, A. Vincent, and I. Walukiewicz, "Games for synthesis of controllers with partial observation," *Theoret. Comp. Sci.*, pp. 7–34, 2003.

[3] ——, "Games for synthesis of controllers with partial observation," *Theoret. Comp. Sci.*, vol. 303, pp. 7–34, 2003.

[4] J. C. M. Baeten, J. A. Bergstra, and J. W. Klop, "Ready-trace semantics for concrete process algebra with the priority operator," *Comp. J.*, vol. 30, no. 6, pp. 498–506, 1987.

[5] G. Barrett and S. Lafortune, "Using bisimulation to solve discrete event control problem," in *Proc. Amer. Control Conf.*, Albuquerque, NM, Jun. 1997, pp. 2337–2341.

[6] ——, "Bisimulation, the supervisory control problem and strong model matching for finite state machines," *J. Discrete Event Syst.: Theory Appl.*, vol. 8, pp. 377–429, 1998.

[7] M. D. Di Benedetto, A. L. Sangiovanni-Vincentelli, and T. Villa, "Model matching for finite state machines," *IEEE Trans. Autom. Control*, vol. 46, no. 11, pp. 1726–1743, Nov. 2001.

[8] M. Hennessey and R. Milner, "Algebraic laws for nondeterminism and concurrency," *J. ACM*, vol. 32, pp. 137–161, 1985.

[9] M. Heymann and F. Lin, "Discrete-event control of nondeterministic systems," *IEEE Trans. Autom. Control*, vol. 43, no. 1, pp. 3–17, Jan. 1998.

[10] M. Heymann and G. Meyer, "Algebra of discrete event processes," NASA Ames Research Center, Moffett Field, CA, Tech. Rep. NASA 102 848, 1991.

[11] C. A. R. Hoare, *Communicating Sequential Processes*. Englewood Cliffs, NJ: Prentice-Hall, 1985.

[12] J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Languages and Computation*. Reading, MA: Addison-Wesley, 1979.

[13] K. Inan, *Nondeterministic Supervision Under Partial Observations*. ser. Lecture Notes in Control and Information Sciences, G. Cohen and J.-P. Quadrat, Eds. New York: Springer-Verlag, 1994, vol. 199, pp. 39–48.

[14] S. Jiang and R. Kumar, "Supervisory control of discrete event systems with CTL* temporal logic specification," in *Proc. IEEE Conf. Decision and Control*, FL, Dec. 2001, pp. 4122–4127.

[15] ——, "Supervisory control of nondeterministic discrete event systems with driven events via masked prioritized synchronization," *IEEE Trans. Autom. Control*, vol. 47, no. 9, pp. 1438–1449, Sep. 2002.

[16] ——, "Supervisory control of discrete event systems with CTL* temporal logic specification," *SIAM J. Control Optim.*, vol. 44, no. 6, pp. 2079–2103, 2006.

[17] J. Komenda, "Coalgebra and supervisory control of discrete-event systems with partial observations," in *Proc. Int. Symp. MTNS*, Notre Dame, IN, Aug. 2002, pp. 12–16.

[18] ——, "Computation of supremal sublanguages of supervisory control using coalgebra," in *Workshop on Discrete-Event Systems (WODES'02)*, Zaragoza, Spain, Oct. 2002, pp. 26–33.

[19] R. Kumar and M. Heymann, "Masked prioritized synchronization for interaction and control of discrete event systems," *IEEE Trans. Autom. Control*, vol. 45, no. 11, pp. 1970–1982, Nov. 2000.

[20] R. Kumar, S. Jiang, C. Zhou, and W. Qiu, "Polynomial synthesis of supervisor for partially observed discrete-event systems by allowing nondeterminism in control," *IEEE Trans. Autom. Control*, vol. 50, no. 4, pp. 463–475, Apr. 2005.

[21] R. Kumar and M. A. Shayman, "Nonblocking supervisory control of nondeterministic systems via prioritized synchronization," *IEEE Trans. Autom. Control*, vol. 41, no. 8, pp. 1160–1175, Aug. 1996.

[22] ——, "Centralized and decentralized supervisory control of nondeterministic systems under partial observation," *SIAM J. Control Optim.*, vol. 35, no. 2, pp. 363–383, Mar. 1997.

[23] O. Kupferman, P. Madhusudan, P. S. Thiagarajan, and M. Y. Vardi, "Open systems and reactive environments: Control and synthesis," in *Proc. 11th Conf. Concurrency Theory*, vol. 1877, Lecture Notes in Computer Science, Aug. 2000, pp. 92–107.

[24] O. Kupferman and M. Y. Vardi, "Robust satisfaction," in *Proc. 10th Conf. Concurrency Theory*, vol. 1664, Lecture Notes in Computer Science, Aug. 1999, pp. 382–398.

[25] O. Kupferman, M. Y. Vardi, and P. Wolper, "Module checking," *Inform. Comput.*, vol. 164, pp. 322–344, 2001.
[26] P. Madhusudan and P. S. Thiagarajan, "Branching time controllers for discrete event systems," *Theoret. Comput. Sci.*, vol. 274, pp. 117–149, 2002.
[27] H. Marchand and S. Pinchinat, "Supervisory control problem using symbolic bisimulation techniques," in *Proc. 2000 Amer. Control Conf.*, Chicago, IL, Jun. 2000, pp. 4067–4071.
[28] R. Milner, *A Calculus of Communicating Systems*. New York: Springer-Verlag, 1980.
[29] ——, *Communication and Concurrency*. Upper Saddle River, NJ: Prentice-Hall, 1989.
[30] A. Overkamp, "Supervisory control for nondeterministic systems," in *Lecture Notes in Control and Information Sciences*, G. Cohen and J.-P. Quadrat, Eds. New York: Springer-Verlag, 1994, vol. 199, pp. 59–65.
[31] I. Phillips, "Refusal testing," *Theoret. Comp. Sci.*, vol. 50, pp. 241–284, 1987.
[32] H. Qin and P. Lewis, *Factorization of Finite State Machines Under Observational Equivalence*, ser. Lecture Notes in Computer Science. New York: Springer-Verlag, 1990, vol. 458.
[33] P. J. Ramadge and W. M. Wonham, "The control of discrete event systems," *Proc. IEEE*, vol. 77, no. 1, pp. 81–98, Jan. 1989.
[34] S. Riedweg and S. Pinchinat, "Quantified mu-calculus for control synthesis," in *Mathematical Foundations of Computer Science*. Berlin, Germany: Springer, 2003.
[35] J. J. M. M. Rutten, "A calculus of transition systems (toward universal coalgebra)," in *Modal Logic and Process Algebra—A Bisimulation Perspective*, A. Ponse, M. de Rijke, and Y. Venema, Eds. Stanford, CA: CSLI Publications, 1995, vol. 53, CSLI Lecture Notes, pp. 231–256.
[36] ——, "Coalgebra, concurrency, and control," in *Proc. 5th Workshop on Discrete Event Systems*, 2000, pp. 31–38.
[37] M. A. Shayman and R. Kumar, "Supervisory control of nondeterministic systems with driven events via prioritized synchronization and trajectory models," *SIAM J. Control Optim.*, vol. 33, no. 2, pp. 469–497, Mar. 1995.
[38] ——, "Process objects/masked composition: An object oriented approach for modeling and control of discrete event systems," *IEEE Trans. Autom. Control*, vol. 44, no. 10, pp. 1864–1869, Oct. 1999.
[39] P. Tabuada, "Open maps, alternating simulations and control synthesis," in *Proc. Int. Conf. Concurrency Theory*, 2004, pp. 466–480.
[40] R. van Glabbeek, "The linear time–Branching time spectrum," in *Proc. CONCUR'90*, vol. 458, Lecture Notes in Computer Science, Amsterdam, The Netherlands, 1990, pp. 278–297.
[41] N. Yevtushenko, T. Villa, R. Brayton, A. Petrenko, and A. Sangiovanni-Vincentelli, "Solution of parallel logic equations for logic synthesis," in *Proc. ICCAD*, Nov. 2001, pp. 103–110.
[42] C. Zhou, R. Kumar, and S. Jiang, "Control of nondeterministic discrete event systems for bisimulation equivalence," in *Proc. Amer. Control Conf.*, Boston, MA, Jun. 2004, pp. 4488–4492.

Changyan Zhou (S'03) received the B.S. degree in mechanical engineering from the Northwestern Polytechnical University, Xian, China, and the M.S. degree in Mechatronic engineering from the Harbin Institute of Technology, Harbin, China, in 1993 and 1996, respectively. She is currently working toward the Ph.D. degree in electrical and computer engineering at Iowa State University, Ames.

Her research interests include formal methods, control and diagnosis of reactive/event-driven systems and their applications.

Ratnesh Kumar (SM'88) received the B.Tech. degree in electrical engineering from the Indian Institute of Technology, Kanpur, India, in 1987, and the M.S. and the Ph.D. degrees in electrical and computer engineering from the University of Texas at Austin, in 1989 and 1991, respectively.

From 1991 to 2002, he was on the faculty of University of Kentucky, Lexington, and since 2002, he has been on the faculty of the Iowa State University, Ames. He has held visiting positions at the Institute of Systems Research at the University of Maryland, College Park, the Applied Research Laboratory at Pennsylvania State University, State College, the NASA Ames Research Center, and the Argonne National Laboratory—West. His primary research interest is modeling, verification, control, and diagnosis of reactive/event-driven, real-time, and hybrid systems and their applications. He is a coauthor of the book *Modeling and Control of Logical Discrete Event Systems* (Norwell, MA: Kluwer, 1995).

Dr. Kumar was a recipient of the Microelectronics and Computer Development (MCD) Fellowship from the University of Texas at Austin, and was awarded the Lalit Narain Das Memorial Gold Medal for the Best EE Student and the Ratan Swarup Memorial Gold Medal for the Best All-Around Student from the Indian Institute of Technology at Kanpur, India. He is a recipient of the NSF Research Initiation Award and a NASA-ASEE summer faculty fellowship award. He serves on the program committee for the IEEE Control Systems Society, and the International Workshop on Discrete Event Systems. He is an Associate Editor of the *SIAM Journal on Control and Optimization*, the *Journal of Discrete Event Dynamical Systems*, and the IEEE Control Systems Society. He is a past Editor of the IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION.

Shengbing Jiang received the B.S. degree in electrical engineering from the University of Science and Technology of China, Hefei, the M.S. degree in electrical engineering from East China Institute of Technology, Nanjing, and the Ph.D. degree in electrical engineering from the University of Kentucky, Lexington, in 1987, 1990, and 2002, respectively.

He joined General Motors R&D, Warren, MI, in 2002. His research interests include formal methods, formal verification, supervisory control, and failure diagnosis of discrete-event and hybrid systems, and their applications in embedded software design.