Control of Nondeterministic Discrete Event Systems for Bisimulation Equivalence

Changyan Zhou[1], Ratnesh Kumar[1], Shengbing Jiang[2]
[1]Department of ECE, Iowa State University, Ames, IA 50011
[2]GM R&D and Planning, Warren, MI 48090-9055

Abstract— **In this paper we study supervisory control for enforcing *nondeterministic specifications*. 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 linearly in both system and specification states.**
Keywords: Discrete event systems, supervisory control, nondeterministic systems, bisimulation equivalence, controllability

I. Introduction

Discrete event systems (DES) are systems with discrete states and are event-driven. Computer, communication, manufacturing and traffic systems are examples of discrete event systems. DES research deals with its modeling, design, control, verification and testing, diagnosis, performance evaluation, and optimization. Ramadge and Wonham [13] initiated supervisory control of discrete event systems. A DES is modeled as an automaton and its behavior 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.

Extensive research has been done for nondeterministic systems [3], [11], [15], [8], [16], [9], [7], [6] using language model as a means of specification. 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 are nondeterministic. Nondeterministic specifications are also meaningful when the system to be controlled has a nondeterministic model due to lack of information (caused for example by partial observation or unmodeled dynamics).

This work was supported in part by the National Science Foundation under the grants NSF-ECS-0218207, NSF-ECS-0244732 and NSF-EPNES-0323379, and a DoD-EPSCoR grant from the Office of Naval Research under the grant N000140110621.

While language equivalence is an adequate notion of behavioral equivalence for deterministic systems, it is not so for nondeterministic systems, and instead we use the finest known notion of equivalence, namely the bisimulation equivalence. Several other models such as failures model [4], refusal-trace model [12], ready-trace model [1], etc., have been proposed in the literature for representing qualitative behavior of nondeterministic DES's. A model more refined than the language model, namely the trajectory-model, has been used as specification in [3]. Authors in [3] show how to transform their control problem of nondeterministic setting to one of deterministic setting with an added partial observability.

We use the notion of bisimulation equivalence as specifications for nondeterministic systems. Bisimulation equivalence is the finest known notion of behavioral equivalence and it was first introduced in communicating systems by Milner [10]. 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* [2]. 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 LTL (Linear Temporal Logic) specification which is a strict subclass of μ-calculus and CTL*.

Control for achieving CTL* specification was studied by Jiang and Kumar in [5], under the assumption that plant model is deterministic. In this paper we allow both plant and specification models to be nondeterministic. Furthermore, our approach is quite different: In [5], 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.

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 plant and specification states. This happens to be the same as the complexity of verifying the existence of a supervisor when both plant and specification are deterministic. Further, the specification model can itself be used as a supervisor when the plant is deterministic.

The rest of the paper is organized as follows. Section 2 presents a motivating example. Section 3 presents notations and preliminary background. Section 4 provides some preliminary results needed for developing the theory for bisimulation equivalence control presented in section 5. Section 6 extends the results of section 5 to general nondeterministic state machines with multiple initial and final states and with epsilon transitions. Section 7 discusses the special case of deterministic plants. The paper concludes with section 8.

II. A Motivating 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 Figure 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 specification R, also shown in Figure 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 non-occurrence of it is immaterial to the specification. This implies that the specification R is for the plant G projected on to the event set $\hat{\Sigma} := \Sigma - \{reset\}$, denoted $G \uparrow \hat{\Sigma}$. Note that $L(R) = L(G \uparrow \hat{\Sigma})$, i.e., $G \uparrow \hat{\Sigma}$

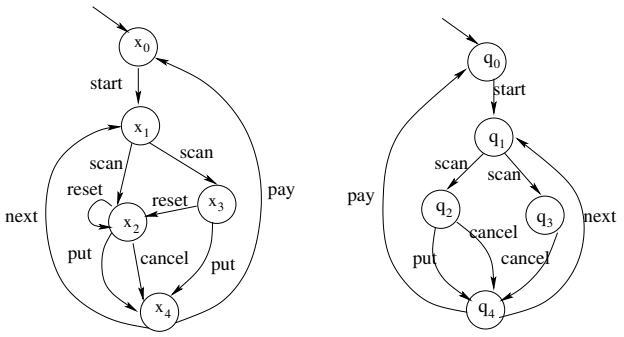

Fig. 1. The plant G (left) and specification R (right)

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 above, 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,

$$G = (X, \Sigma, \alpha, X_0, X_m),$$

where X is the set of states, Σ is the alphabet of events, $\alpha : X \times (\Sigma \cup \{\epsilon\}) \to 2^X$ is the state transition function, where ϵ is a label for "silent" or "unobservable" 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 sequences 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 (resp., marked) by G, is denoted as $L(G)$ (resp., $L_m(G)$). $L(G)$ 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., $L_m(G) = \{s \in L(G) \mid \alpha^*(X_0, s) \cap X_m \neq \emptyset\}$. Given $\hat{\Sigma} \subseteq \Sigma$, we use $G \uparrow \hat{\Sigma}$ to denote G obtained by replacing each transition $(x, \sigma, x') \in X \times (\Sigma - \hat{\Sigma}) \times X$ by (x, ϵ, x'). For notational convenience, we define $\overline{\Sigma} = \Sigma \cup \{\epsilon\}$. For

$x \in X$, we define $\Sigma(x) := \{\sigma \in \overline{\Sigma} \mid \alpha(x,\sigma) \neq \emptyset\}$ to denote the set of events defined at state x.

A DES, called a plant, is controlled to restrict it's behavior so as to prevent any undesirable behavior by dynamically disabling certain controllable events [14]. 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 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 $x_1 \in X_1, x_2 \in X_2, \sigma \in \overline{\Sigma}$,

$\alpha((x_1,x_2),\sigma) =$

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

Most prior work on supervisory control of DESs deals with language equivalence between the controlled system and the specification. However, in nondeterministic setting the language can not distinguish the structural differences of two automata that generate the same language. Numerous notions of behavioral equivalence that are finer than the language equivalence have been proposed. Bisimulation equivalence is the finest equivalence among them, and has been established as an appropriate notion of behavioral equivalence for nondeterministic systems [17]. We first introduce the concept of a simulation relation. For simplicity of presentation we first consider nondeterministic state machines with single initial state, all states marked and no epsilon transitions. Extension to general nondeterministic state machines is discussed in Section 6.

Definition 1: Given two automata G_1 and G_2 as defined above, a *simulation relation* is a binary relation $\Phi \subseteq X_1 \times X_2 \subseteq (X_1 \cup X_2)^2$ such that for $x_1 \in X_1, x_2 \in X_2$, $(x_1,x_2) \in \Phi$ implies

$$\sigma \in \Sigma(x_1), x_1' \in \alpha_1(x_1,\sigma) \Rightarrow$$

$$\exists x_2' \in \alpha_2(x_2,\sigma) \text{ such that } (x_1',x_2') \in \Phi.$$

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.

A bisimulation relation is a symmetric simulation relation. 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 for the simulation and bisimulation relation between two automata.

Definition 2: Given two automata G_1 and G_2, G_1 is simulated by G_2 (denoted as $G_1 \sqsubseteq_\Phi G_2$) if there exists a simulation relation $\Phi \subseteq X_1 \times X_2 \subseteq (X_1 \cup X_2)^2$ with $(x_{01},x_{02}) \in \Phi$, i.e., $x_{01} \sqsubseteq_\Phi x_{02}$. G_1 and G_2 are said to be bisimilar (denoted as $G_1 \simeq_\Phi G_2$) if there exists a bisimulation relation $\Phi \subseteq (X_1 \cup X_2)^2$ such that $(x_{01},x_{02}) \in \Phi$, i.e., $x_{01} \simeq_\Phi x_{02}$.

IV. Bisimilarity Preserving State-merger

We show that when two bisimilar states of a state machine are merged, the resulting state machine is bisimilar to the original one.

The next lemma shows simulation and bisimulaiton relations preserve transitive property. For space consideration, all proofs are omitted.

Lemma 1: Given G, consider $x_1, x_2, x_3 \in X$.

1) If $x_1 \sqsubseteq_\Phi x_2$ and $x_2 \sqsubseteq_{\Phi'} x_3$, then $x_1 \sqsubseteq_{\Phi''} x_3$.
2) If $x_1 \simeq_\Phi x_2$ and $x_2 \simeq_{\Phi'} x_3$, then $x_1 \simeq_{\Phi''} x_3$.

We use $\overline{G_{\langle x,x' \rangle}}$ to denote the automaton G in which two states $x, x' \in X$ have been merged to form the merged state $\langle x,x' \rangle$.

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

V. 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 $G = (X, \Sigma, \alpha, x_0)$, $R = (Q, \Sigma, \delta, q_0)$, and $S = (Y, \Sigma, \beta, y_0)$ to denote the plant, the specification, and the supervisor, respectively.

Definition 3: 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 4: 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

$$s \in L(R), \sigma \in \Sigma_u \text{ such that } s\sigma \in L(G)$$

$$\Rightarrow \forall q_s \in \delta(q_0,s), \sigma \in \Sigma(q_s).$$

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

Lemma 2: 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 $G \| S \simeq G \| S'$.

We next present our main result on existence of supervisor S for plant G such that $G \| S$ is bisimilar to 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 $G\|T \simeq R$.

Remark 1: 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.

A condition in Theorem 2 is that of state-controllability. We present below 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 $\forall q \in Q, \sigma \in \Sigma$:

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

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 $G\|\overline{R}$.

Remark 2: The complexity of the algorithm for testing state-controllability of R with respect to G is $O(|X|\times|Q|)$, i.e., it is linear in the number of states of both G and R. This complexity is the same as that of testing language-controllability in the deterministic setting.

VI. Extension to General Nondeterministic Setting

In this section, we extend our earlier results on supervisory control for bisimulation equivalence to allow for general NSM models that can consist of marked states, multiple initial states, and ϵ-transitions. The definition of simulation relation can be extended as follows.

Definition 5: 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) $x_1 \in X_{m1} \Rightarrow x_2 \in X_{m2}$.

A bisimulation relation $\Phi \subseteq (X_1 \times X_2)^2$ is a symmetric simulation relation. Definition of simualtion and bisimulation relations for two automata in the general setting is given as follows.

Definition 6: Given G_1 and G_2 as in Definition 5, 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 $x_{01} \in X_{01} \Rightarrow \exists x_{02} \in X_{02}$ such that $(x_{01}, x_{02}) \in \Phi$, i.e., $X_{01} \sqsubseteq_\Phi X_{02}$. G_1 and G_2 are said to be bisimilar (denoted as $G_1 \simeq_\Phi G_2$) if there exists a bisimulation relation $\Phi \subseteq (X_1 \times X_2)^2$ such that $X_{01} \simeq_\Phi X_{02}$.

We say that $G_1 \simeq G_2$ *with respect to* $\hat{\Sigma} \subseteq \Sigma$ if $G_1\uparrow \hat{\Sigma} \simeq G_2\uparrow \hat{\Sigma}$. All results of the previous two sections hold for the setting of this section, including the following extension of the small model theorem.

Theorem 3: Given plant G and specification 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 $G\|T \simeq R$.

Now we revisit the motivating example.

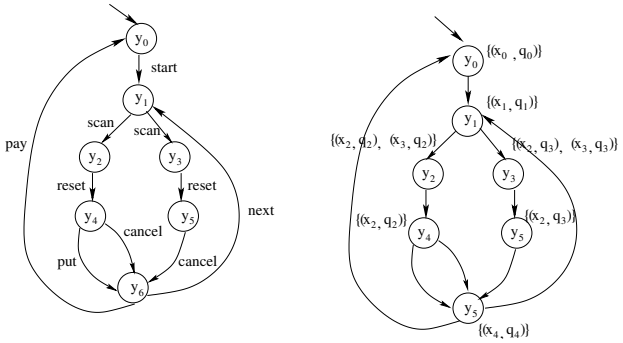

Fig. 2. Supervisor S (left) and labeling of its states (right)

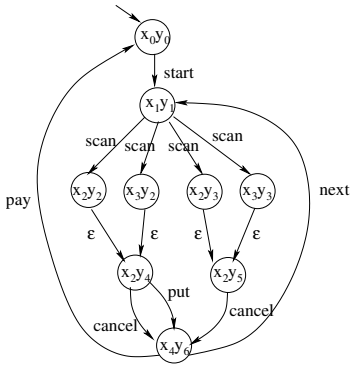

Fig. 3. Controlled system $(G\|S)\uparrow\hat{\Sigma}$

Example 2: We need to find a Σ_u-compatible supervisor S such that $(G\|S)\Uparrow\hat{\Sigma} \simeq R$, where $\hat{\Sigma} = \Sigma - \{reset\}$. Such a supervisor S is shown in Figure 2. (S is state-controllable, and can be made Σ_u-compatible by adding appropriate uncontrollable event self-loops.) The synchronous composition of G and S is drawn in Figure 3. 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 Figure 2). 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 (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 $G\|T = G\|S$.

VII. Specialization to Deterministic Plant

In this section, we study the specialized case when plant is deterministic. It turns out that for a deterministic plant, whenever a supervisor exists, the specification itself can serve as a supervisor, i.e., $G\|R \simeq R$, which leads to a polynomial complexity result for the existence as well as the synthesis of a supervisor.

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

1) R is state-controllable, and
2) $L(R) \subseteq L(G)$.

Remark 3: From Theorem 4, the complexity for checking the existence of a supervisor for enforcing bisimilarity for a deterministic plant is $O(|X| \times |Q|)$, which is linear in the number of states of plant and specification. This is the same as the existence complexity when both plant and specification are deterministic. Moreover, when the existence conditions are satisfied, the specification itself serves as a supervisor, i.e., the complexity of synthesizing a supervisor is linear in the specification states. It is interesting to note that when specification is deterministic but plant is nondeterministic, the complexity of existence is again polynomial [3]. In contrast, we have shown the situation is different when both plant and specification are nondeterministic.

VIII. 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. We have shown that the bisimilarity is preserved under the merger of bisimilar states. 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 the supervisor and provided a polynomial complexity test for it. We obtained a small model theorem showing that a supervisor exists if and only if it exists over a certain finite state space. 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 both the plant and specification states. Also, in this case, the specification can itself be used as a supervisor, implying the size of supervisor is linear in the size of specification.

References

[1] J. C. M. Baeten, J. A. Bergstra, and J. W. Klop. Ready-trace semantics for concrete process algebra with the priority operator. *The Computer Journal*, 30(6):498–506, 1987.

[2] M. Hennessey and R. Milner. Algebraic laws for nondeterminism and concurrency. *Journal of ACM*, 32:137–161, 1985.

[3] M. Heymann and F. Lin. Discrete-event control of nondeterministic systems. *IEEE Transactions on Automatic Control*, 43(1):3–17, January 1998.

[4] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice Hall, Inc., Englewood Cliffs, NJ, 1985.

[5] S. Jiang and R. Kumar. Supervisory control of discrete event systems with CTL* temporal logic specification. In *2001 IEEE Conference on Decision and Control*, pages 4122–4127, FL, December 2001.

[6] S. Jiang and R. Kumar. Supervisory control of nondeterministic discrete event systems with driven events via masked prioritized synchronization. *IEEE Transactions on Automatic Control*, 47(9):1438–1449, 2002.

[7] R. Kumar and M. Heymann. Masked prioritized synchronization for interaction and control of discrete event systems. *IEEE Transactions on Automatic Control*, 45(11):1970–1982, 2000.

[8] R. Kumar and M. A. Shayman. Non-blocking supervisory control of nondeterministic systems via prioritized synchronization. *IEEE Transactions on Automatic Control*, 41(8):1160–1175, August 1996.

[9] R. Kumar and M. A. Shayman. Centralized and decentralized supervisory control of nondeterministic systems under partial observation. *SIAM Journal of Control and Optimization*, 35(2):363–383, March 1997.

[10] R. Milner. *A Calculus of Communicating Systems*. Springer Verlag, 1980.

[11] A. Overkamp. Supervisory control for nondeterministic systems. In Guy Cohen and Jean-Pierre Quadrat, editors, *Lecture Notes in Control and Information Sciences 199*, pages 59–65. Springer-Verlag, New York, 1994.

[12] I. Phillips. Refusal testing. *Theoretical Computer Science*, 50:241–284, 1987.

[13] P. J. Ramadge and W. M. Wonham. Supervisory control of a class of discrete event processes. *SIAM Journal of Control and Optimization*, 25(1):206–230, 1987.

[14] P. J. Ramadge and W. M. Wonham. The control of discrete event systems. *Proceedings of IEEE: Special Issue on Discrete Event Systems*, 77:81–98, 1989.

[15] M. Shayman and R. Kumar. Supervisory control of nondeterministic systems with driven events via prioritized synchronization and trajectory models. *SIAM Journal of Control and Optimization*, 33(2):469–497, March 1995.

[16] M. A. Shayman and R. Kumar. Process objects/masked composition: An object oriented approach for modeling and control of discrete event systems. *IEEE Transactions on Automatic Control*, 44(10):1864–1869, 1999.

[17] Y. Willner and M. Heymann. Supervisory control of concurrent discrete-event systems. *International Journal of Control*, 54(5):1143–1169, 1991.