On Petri Net Models of Infinite State Supervisors

R. S. Sreenivas and B. H. Krogh

Abstract—We consider a class of supervisory control problems that require infinite state supervisors and introduce Petri nets with inhibitor arcs (PN's) to model the supervisors. We compare this PN-based approach to supervisory control to automata-based approaches. The primary advantage of a PN-based supervisory controller is that a PN-based controller provides a finite representation of an infinite state supervisor. For verification, implementation, and testing reasons, a finite PN-based representation of an infinite state supervisor is preferred over an automata-based supervisor. We show that this modeling advantage is accompanied by a decision disadvantage, in that in general the controllability of a language that can be generated by the closed-loop system is undecidable.

I. INTRODUCTION

Supervisory control of discrete-event systems (DES's) was introduced by Ramadge and Wonham [1], [2] and has since been studied extensively [3]–[7]. A DES is a dynamical system with a (possibly infinite) state set and a finite set of events. For a given DES G, it is of interest to synthesize a supervisor DES Θ that prevents the occurrence of certain events in G to enforce some specifications on the behavior of the controlled DES. The classes of specifications that have been considered hitherto fall into two categories: state avoidance problems [2], where the objective is to avoid a collection of states, and string avoidance problems [1], where the objective is to avoid a collection of event strings. In this note, we consider string avoidance problems.

We introduce a method of modeling infinite state supervisors by Petri nets (PN's) of finite size. We relate supervisory control in the PN-based models to those in the current literature that use automata-based models. We then introduce a class of supervisory control problems that require infinite state supervisors. We show that for these problems there is a PN-based supervisor that has a finite representation. For verification, implementation, and testing reasons, a finite PN-based representation of an infinite state supervisor is preferred over an automata-based supervisor.

This note is organized as follows. In Section II, we present an overview of automata-based supervisory control. Section III states two results on the use of infinite state models for plants and supervisors. Section IV introduces Petri nets with inhibitor arcs as finite-size models of infinite state supervisors. We show that any Turing acceptable language can be realized with such supervisors.

The concluding section indicates directions for future research.

II. AN OVERVIEW OF AUTOMATA-BASED SUPERVISORY CONTROL

Following Ramadge and Wonham [7], we define a controlled DES or a plant as a 4-tuple G = (Q, Σ, δ, q0), where Q is a (possibly infinite) state set, q0 ∈ Q is the initial state, Σ is a finite alphabet used to label transitions between states (events), and δ: Q × Σ → Q is a partial function that describes the dynamics of the system. Events are assumed to be instantaneous and asynchronous. We extend the function δ to a function δ*: Q × Σ* → Q, in the usual way. Also, we define the size of an automaton to be card(Q).

The language generated by G is denoted by the symbol L(G).

Given a language L ⊆ Σ* we use the symbol L to denote its prefix closure.

To control a DES we assume the set Σ is partitioned into two sets ΣP and ΣN, where ΣP is the set of events that can be disabled. We let Σ denote the set of control patterns, where

Γ = {γ | γ: Σ → {0, 1} and γ(a) = 1 for all a ∈ ΣP}.

An event a ∈ Σ is said to be enabled by γ when γ(a) = 1, and is disabled otherwise.

A supervisor Θ is a system that changes the control patterns dynamically. An automata-based supervisor Θ is a pair (S, ϕ), where S = (X, Σ, ε, x0) is an automaton with a (possibly infinite) state set X, input alphabet Σ, partial transition function ε: X × Σ → X, initial state x0, and ϕ: X → Γ. As defined by Ramadge and Wonham [1], the supervisor state transitions are synchronous with identically labeled events in the plant G. At each state of S a control pattern is selected based on the function ϕ. We use the symbol Θ | G to represent the closed-loop plant-supervisor system described above, and the symbol L(Θ | G) to represent the language generated by the system Θ | G.

To have a well-defined closed-loop behavior a completeness condition [1] must be imposed on the supervisor Θ, namely,

w ∈ Σ* and o ∈ Σ the following statement must be true:

w ∈ L(Θ | G) and w o ∈ L(G) and 6(6*(xo, w))(q) = 1 ⇒ w o ∈ L(Θ | G).

where * is the string concatenation operator and 6*: X × Σ* → X is an extension of 6: X × Σ → X.

Following Ramadge and Wonham [1] a language θ ⊆ Σ* is said to be controllable with respect to G if

K* ⊆ L(G) ⊆ K.

There exists a complete supervisor Θ such that L(Θ | G) = K if and only if the language K ⊆ Σ* is prefix-closed and controllable with respect to G [1]. The supervisor Θ thus prevents the generation of strings in L(G) that are not in K, while making sure all strings in K can be generated by the system Θ | G. We call the problem of synthesizing a supervisor to realize a restricted language K the string avoidance problem.

III. INFINITE STATE PLANTS AND SUPERVISORS

In this section, we consider two ways in which infinite size automata may be required for the string avoidance problem. First, the plant G might be of infinite size, for example, when the language generated by the plant L(G) is nonregular. Second, the desired closed-loop behavior K ⊆ L(G) may lead to an infinite state supervisor, for example, when K is a controllable, prefix-closed, and nonregular language. The following two propositions deal with each of these cases in turn. The second case is illustrated with an example following Proposition 2.

Proposition 1: If G = (Q, Σ, δ, q0) is an infinite size plant and K ⊆ Σ* is a prefix-closed language such that K ⊆ L(G) and K is controllable with respect to G, then there exists a finite size plant G' = (Q', Σ, δ', q0) (i.e., Q' is finite), such that

1. L(G) ⊆ L(G'),

2. if aK' ⊆ Σ* such that K' ⊆ K, and K' is prefix-closed and controllable with respect to G', and any complete supervisor Θ that satisfies the property L(G' | G) = K' also satisfies the property L(G' | G) = K.

 Manuscript received January 12, 1990. This work was supported in part by the National Science Foundation under Grant DMC-8451493.

The authors are with the Laboratory for Automated Systems and Information Processing, Department of Electrical Engineering and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213.

IEEE Log Number 9104397.
Proof: The first part of the proposition follows trivially from the fact that any language based on a finite alphabet that is always a regular language that is a superset. For example, if $L(G) \subseteq L(G)$, then the language of the trivial automaton $G'$ with only one state for which such that $L(G) = L(G)$.

For the second part, let $L(G) = L(G)$ as described above. Now, consider the language $K' = K'$. Note that $K'$ is prefix-closed since $K$ is prefix-closed. Furthermore, $K'^* \subseteq L(G) = K'^* \subseteq K$. This implies that $K'$ is controllable with respect to $G$. Since $K'$ is prefix-closed and controllable with respect to $G'$, there exists a complete supervisor $\Theta'$ such that $L(\Theta' | G) = L(G)$. Since $L(G) \subseteq L(G)$, $\Theta' | G$ is well-defined and $L(\Theta' | G) = L(\Theta' | G)$. We now show that $L(\Theta' | G) = L(G)$.

Let $w \in L(\Theta' | G)$ and suppose $w \notin K$. Since $L(\Theta' | G) \subseteq L(\Theta' | G) = K'^* \subseteq \Sigma$ such that $w = \omega_1 \omega_2$, and $\omega_1 \in K$, $\omega_2 \in \Sigma^*$, such that $w = \omega_1 \omega_2$. This contradicts the assumption that $K$ is controllable. Therefore, $L(\Theta' | G) = L(G)$.

Proposition 1 implies that in general the procedure of Ramadge and Wonham [1] can be used to construct supervisors for infinite size plants by applying their procedure to a finite size plant (although the proof of the proposition does not necessarily suggest the most computationally efficient method for doing this). The following proposition demonstrates that the size of the supervisor is determined by the desired closed-loop language $K$.

**Proposition 2:** Given a finite size plant $G$, and $K \subseteq L(G)$, a prefix-closed, language that is controllable with respect to $G$. If $K$ is nonregular, then there does not exist a complete finite size supervisor $\Theta$ such that $L(\Theta | G) = K$.

**Proof:** The proposition follows from the fact that for any finite size supervisor, the composite system $\Theta | G$ is necessarily finite. Thus if $K$ is a nonregular language, any complete supervisor $\Theta$ such that $L(\Theta | G) = K$ can be obtained only if $\Theta$ is of infinite size.

As an example consider the plant $G = (Q, \Sigma, \delta, q_0)$, where $Q = \{q_0, q_1\}$, $\Sigma = \{a, b\}$, and $\delta$ is the function tabulated in Table 1. The state diagram of the plant $G$ is shown in Fig. 1. The initial state is represented by an arrow head on the vertex representing $q_0$. Let $\Sigma = \{b\}$, and $\Sigma = \{a\}$. $L(G)$ is the regular language denoted by the regular expression $a^*b^*$. Consider the language $K \subseteq L(G)$, where $K = \{a^nb^m \mid n \geq 0 \}$. $K$ is prefix-closed and controllable because if $w \in K$, $w^*a \in L(G)$ implies the fact that $w$ does not contain any $b$'s, so $w^*a \in K$. However, $K$ is nonregular. From the Proposition 2 we know that any complete supervisor $\Theta$ that produces a closed-loop language $K$ has to have an infinite size. In the next section, we illustrate a supervisor for this example which is of finite size within the PN framework (but infinite state) and produces the desired closed-loop behavior $K$.

**IV. PN-BASED SUPERVISORY CONTROL**

In this section, we introduce a modeling framework for DES's using PN's. For the sake of uniformity we assume the plant DES to be a finite state automaton as in the previous section. However, the supervisor is a PN. In the following paragraphs, we describe the closed-loop behavior when a PN based supervisor is used in conjunction with a finite automata based plant.

**Definition 1:** A Petri net with inhibitor arcs (PN) is an ordered 5-tuple, $N = (\Pi, T, \Phi, \Psi, m_0)$, where $\Pi = \{p_1, p_2, \ldots, p_n\}$ is a set of $n$ places, $T = \{t_1, t_2, \ldots, t_m\}$ is a collection of $m$ transitions, $\Phi \subseteq (\Pi \times T) \cup (T \times \Pi)$ is a set of arcs, $\Psi \subseteq (\Pi \times T)$ is a set of inhibitor arcs, $m_0: \Pi \rightarrow N$ is the initial marking function (or the initial marking), and $N$ is the set of nonnegative integers.

The state of a PN is defined by the marking $m: \Pi \rightarrow N$ which indicates the distribution of tokens in each place. We define the size of a PN as max{card($\Pi$), card($T$)}.

A string of transitions $t_1, t_2, \ldots, t_i$, where $t_i \in T$ ($i = \{1, 2, \ldots, k\}$) is said to be a valid firing sequence starting from the marking $m$ if

1) the transition $t_i$ is enabled, and
2) for $i = \{1, 2, \ldots, k\}$, the firing of the transition $t_i$ produces a marking under which the transition $t_{i+1}$ is enabled.

Given an initial marking $m_0$, the set of reachable markings for $m_0$ denoted by $R(m_0)$, is defined as the set of markings generated by all valid firing sequences starting with marking $m_0$. We use the notation $m \rightarrow \omega \rightarrow m'$ if the firing sequence $\omega$ starting from the marking $m$ results in the marking $m'$.
where \( S = (\Pi, T, \Phi, \gamma, m_0) \) is a PN, \( \Sigma \) is an alphabet set (same as the alphabet set associated with the plant), and \( \alpha: T \rightarrow \Sigma \cup \{\epsilon\} \) is a total function that identifies a plant event or the null event \( \epsilon \) with each transition in \( S \). The function \( \alpha^n: T^n \rightarrow \Sigma^n \) is the extension of the \( \alpha \) function to strings. The function \( \Phi: \mathcal{R}(m_0^n) \rightarrow \Gamma \) defines a control pattern for each marking in the set of reachable markings, and \( \Gamma \) is the set of control patterns defined in the previous section. The function \( \Phi: \mathcal{R}(m_0^n) \rightarrow \Gamma \) is defined explicitly for \( m_r \in \mathcal{R}(m_0^n) \), \( \alpha \in \Sigma \) as follows:

\[
\Phi(m_r)(\alpha) = \begin{cases} 1 & \text{if } \alpha \in \Sigma_u \text{ or } \alpha = \epsilon(t) \text{ for some } t \in \mathcal{N}_s(m_r), \\ 0 & \text{otherwise} \end{cases}
\]

where \( \mathcal{N}_s(m_r) = \{t | t \in T_s(m_r) \text{ or } t \in T_r(m_r') \} \), \( m_r \rightarrow \omega \rightarrow m_r' \), and \( \epsilon(t) = |\omega| \) denotes the length of the firing sequence \( \omega \).

As with automata based supervisors, events (transitions) in the supervisor are triggered by events in the plant according to the following convention which takes into account the null event symbols defined above. Suppose the supervisor has a marking \( m_r \) and \( \Phi(m_r)(\alpha) = 1 \) for event \( \alpha \in \Sigma \). If the event \( \alpha \) occurs in the plant, a firing sequence occurs on the associated transitions and \( \Phi(m_r)(\alpha) = 1 \) for event \( \alpha \in \Sigma \). The firing sequence occurs simultaneously in the PN-based supervisor where \( \alpha^n(\omega) = (\epsilon)^{n-1} \sigma \). We note that the PN-based supervisor can be non-deterministic since for a given marking of the PN \( S \), there can be many permissible transitions with the same symbol associated with them via the \( \alpha \) function. Furthermore, deterministic PN-based supervisors as defined above that recognize controllable languages are always complete.

As in the previous section, we denote the closed-loop supervisor-plant combination by \( \Theta(G) \) and its language by \( L(\Theta | G) \). A plant \( G \) can be considered as a language generator, while a supervisor \( \Theta \) can be thought of as a language acceptor. We observe that \( L(\Theta | G) = L(\Theta) \cap L(G) \), where \( L(\Theta) \) denotes the controllable language accepted by a complete PN-based supervisor \( \Theta \).

As an example of PN-based supervisory control, consider the example described in the previous section. Let \( \Sigma = \{a, b\} \) and \( \Sigma_u = \{\epsilon\} \). The plant \( G \) shown in Fig. 1 generates the language denoted by the regular expression \( a^*b^n \). Let \( K = \{a^*b^n | n \geq m \geq 0\} \). Fig. 3 shows a supervisor \( \Theta = (S, \Sigma, \alpha, \phi) \), where \( \alpha(t_1) = \{a\} \), and \( \alpha(t_2) = \alpha(t_3) = \{b\} \). This definition of \( \alpha \) is represented by the symbol \( \alpha \) being placed alongside the transition \( t_2 \) and the symbol \( \{b\} \) being placed alongside the other transitions. By definition, the event \( \{a\} \) is enabled all the time, and the event \( \{b\} \) is enabled only when the marking of the supervisor \( \Theta \) is such that one of these transitions is enabled and only if the place \( p_4 \) has a nonzero token load. This is illustrated graphically in the Fig. 4.

From the figure we observe that every firing of \( t_3 \) (event \( a \)), puts a token in place \( p_3 \) and re-enables \( t_4 \). Upon the first firing of \( t_k \) (event \( b \)), \( t_4 \) is disabled permanently (since no more \( a \)'s can occur in the plant) and \( t_4 \) (event \( b \)) can fire repeatedly until the tokens in place \( p_4 \) are depleted. Firing \( t_4 \) terminates the string of \( b \)'s before \( p_4 \) is emptied. Thus, the closed-loop system generates the language \( \{a^*b^n | n \geq m \geq 0\} \). This is an example of a PN supervisor which realizes a non-regular language. Also, since the language \( K \) is a L-type PN language (cf. [8]) the PN in the definition of the supervisor did not require inhibitor arcs.

The next theorem describes the class of closed-loop languages that can be obtained by using a finite size PN as the supervisor.

**Theorem 1:** Given a finite size plant \( G = (Q, \Sigma, \delta, q_0) \), let \( K \subseteq L(G) \) be a prefix-closed, Turing acceptable language that is controllable with respect to \( G \), then there exists a complete PN based supervisor \( \Theta = (S, \Sigma, \alpha, \phi) \) such that \( S \) is of finite size and \( L(\Theta | G) = K \).

**Proof:** Hack [9] proved that for every Turing acceptable language \( K' \) there exists a (possibly non-deterministic) finite size PN with inhibitor arcs which accepts \( K' \). Since \( K \) is prefix-closed, Turning acceptable and controllable with respect to \( G \), \( K' = K' \cup \Sigma_u \) is also prefix-closed, Turning acceptable and controllable with respect to \( G \). Also, there exists a complete supervisor \( \Theta = (S, \Sigma, \alpha, \phi) \) such that \( L(\Theta) = K' = K' \cup \Sigma_u \) and \( L(\Theta | G) = L(\Theta) \cap L(G) = K' \cap L(G) = K \cap L(G) = K \), as \( K \subseteq L(G) \).

Ramadge and Wonham have shown that a procedure to test for the controllability of a prefix-closed regular language \( K \) with respect to \( L(G) \), where \( G \) is of finite size is \( O(n^2k^2) \), where \( k \) is the cardinality of the state set for a trim recognizer of \( K \) and \( n \) is the cardinality of the state set of \( G \) [7]. Consider the language \( K \subseteq \{a, b\}^n \), where \( K = \{a^*b^n | n \geq m \geq 0\} \) of the previous example. It is easy to show this language is controllable by using an argument that is specific to this language. However, as the following
On Damping Ratio of Polynomials with Perturbed Coefficients

C. B. Soh

Abstract—Recently, Soh and Berger [1] have derived a sufficient condition for a family of real interval polynomials to have a damping ratio of $\epsilon$ using Kharitonov's theorem for complex polynomials. This note points out that the transformations used by Soh and Berger [1] to obtain the sufficient conditions also guarantees a simplification of Kharitonov’s theorem for complex polynomials. That is, the number of required polynomials to be Hurwitz is half the number specified by Soh and Berger [1].

I. INTRODUCTION

Consider the characteristic polynomial of a linear continuous-time system

$$P(s) = t_0 s^n + t_1 s^{n-1} + \cdots + t_n$$

where
t' = [t_0, \ldots, t_n].$

Soh and Berger [1] have shown that a sufficient condition for a family of real interval polynomials (of the form (1)) to have only roots in the left sector (see Fig. 1) defining the damping ratio of continuous-time systems is that eight specified complex polynomials are Hurwitz. We would like to point out that only four of the eight specified complex polynomials are required to be Hurwitz to guarantee the family of real interval polynomials has only roots in the left sector. Similarly, only eight of the sixteen specified complex polynomials given by Soh and Berger [1] are required to be Hurwitz to guarantee a family of complex interval polynomials has only roots in the left sector.

This is because the transformations used by Soh and Berger [1] have been incorrectly printed. Furthermore $P(s)$ and $Q(s)$ denotes, respectively, polynomials of real and complex coefficients.

II. SUPPORTING RESULTS

In this section, we develop the tools necessary for attaining the ultimate objective.

**Lemma 1:** Every polynomial $f(s) \in K^n$ does not have imaginary roots $jw_i$ with $w_i \leq 0$ if and only if the pair polynomials

$$\{(h_1, e_1), (h_2, e_2), (h_3, e_3)\}$$

are Hurwitz polynomials.

REFERENCES


