Technical Notes and Correspondence

Robust Nonblocking Supervisory Control of Discrete-Event Systems

Sean E. Bourdon, M. Lawford, and W. M. Wonham

***Abstract*—In this note, we generalize a robust supervisory control framework to deal with marked languages. We show how to synthesize a supervisor to control a family of plant models, each with its own specification. The solution we obtain is the most general in that it provides the closest approximation to the supremal controllable sublanguage for each plant/specification pair. We end the note by extending these results to deal with timed discrete-event systems.**

***Index Terms*—Adaptive control, discrete-event systems, robust control, supervisory control.**

I. Introduction

As with many other areas of control theory, the notion of robust control has been introduced to discrete-event systems (DES) in order to cope with situations in which a plant's dynamics are not precisely known. The true plant model is assumed to be one among a set of possibilities, none of which can be dismissed as unlikely or impossible. In this note, we describe a synthesis procedure to compute the optimal nonblocking controller for a family of plant models, each one with its own specification. The controller thus obtained is optimal in that it is maximally permissive when constrained by the combined control objectives imposed by each of the plant/specification pairs. The first part of the note in which we deal with untimed DES is closely based on [1]. Afterwards, we extend these results to deal with the timed discrete-event systems of [2].

There are three distinct frameworks in which robust supervisory control has previously been studied. The first paradigm is one in which robustness is specified in terms of arbitrary performance measures defined via metrics on the set of states. This work is focused on resilience or error recovery properties of fault-tolerant systems. See [3]–[5] for representative papers in this area. A good survey of related work in the area of manufacturing systems can be found in [6]. The remaining two paradigms for robust supervisory control focus on the specific case in which performance is measured in terms of the largest possible language within specified behavior. Cury and Krogh [7], [8] and Takai [9], [10] start with a nominal model of the system dynamics. Their control objective is to synthesize a controller which maximizes the family of plants for which the closed-loop behavior is within specified bounds. This family of plants must necessarily include the nominal plant model. Park and Lim [11] also formulate their framework based on the use of a nominal plant model, although robustness is introduced via a Δ-transition representing internal and unobservable events of the system. The final framework in which robust supervisory control is studied assumes that while the plant dynamics are not precisely known, they are among a finite set of possibilities. The underlying assumption in this robustness framework is quite similar to the one used in developing program families or product-line software [12].

The results of this note are presented within the context of the latter robustness paradigm since this choice seems most natural in the context of DES. For example, a designer with limited resources may not be able to construct a controller for each of n similar, yet distinct, robots in a given factory. This note provides a framework in which we are able to synthesize a single controller for the entire group of robots.

Lin [13] was the first to study robustness in this context. He considered the case in which the specification language K is a subset of the language of each of the plants belonging to the set of possibilities. His paper gives a necessary and sufficient condition under which the closed-loop behavior of each of the plants is equal to K. The note also provides an adaptive control scheme under which the supervisor can be refined in the presence of observations of the system's behavior. Takai [14] generalized these results to the case where the legal behavior is no longer necessarily a sublanguage of each possible plant language. Takai later generalized his results to deal with timed DES in [15]. Park and Lim [16], [17] generalized Lin's results to deal with nondeterministic automata with marked languages. The specialization of their work to the deterministic case does not yield any significant new results.

The work we present in this note is an extension of Lin's work to the most general case and is closely related to Takai's. As in [14], we deal with the case where both the event set and the language for a given model of the plant may contain elements that do not belong to any other model in the family. Also, as will become clear later, supervisors obtained using the current robustness framework can result in significant improvements over supervisors obtained using the results in [13]. Although our work is similar to Takai's, there are two distinct differences. First, we deal with *marked* behaviors and hence we must prevent blocking in the closed-loop system. Secondly, our framework represents a more natural setting in which to model robust DES, a claim we develop in Remark 12.

II. Supervisory Control of Discrete-Event Systems

We begin with a brief introduction to the supervisory control theory initiated by Ramadge and Wonham [18]. Let Σ be a nonempty finite set of event symbols, or *alphabet*, and Σ^* be the set of all finite sequences of events, or *strings*, over Σ including the empty string, ε. Any subset of Σ^* is called a *language* over Σ.

Suppose L is a language over Σ. A string $u \in \Sigma^*$ is a *prefix* of $s \in L$ if there exists $w \in \Sigma^*$ such that $uw = s$. In this case, we write $u \leq s$. The *prefix closure* (or simply the *closure*) of L consists of the set of strings which are prefixes of strings in L. More precisely, the closure of L, denoted $\overline{L}$, is defined as follows:

$$\bar{L} := \{u \in \Sigma^* \mid u \leq s \text{ for some } s \in L\}.$$

Given a language $L \subseteq \Sigma^*$ and a string $s \in \Sigma^*$, the set of *eligible* events at s in L is given by

$$\Sigma_L(s) := \{\sigma \in \Sigma \mid s\sigma \in \bar{L}\}$$

Manuscript received May 14, 2003; revised February 8, 2005. Recommended by Associate Editor A. Giua.

S. E. Bourdon and W. M. Wonham are with the Systems Control Group, Edward S. Rogers Sr. Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON M5S 3G4, Canada (e-mail: bourdon@control.toronto.edu; wonham@control.toronto.edu).

M. Lawford is with the Department of Computing and Software, McMaster University, Hamilton, ON L8S 4K1, Canada (e-mail: lawford@mcmaster.ca).

Digital Object Identifier 10.1109/TAC.2005.860237

A discrete-event system, or plant, is modeled via a deterministic automaton

$$\mathbf{G} := (Q, \Sigma, \delta, q_o, Q_m)$$

where Q is the (finite) set of states, Σ is the event set, the partial function $\delta : Q \times \Sigma \to Q$ is the transition function, q_o is the initial state, and $Q_m \subseteq Q$ is the set of marked states of G. The function δ is extended to $\delta : Q \times \Sigma^* \to Q$ in the obvious way. The closed language generated by $\mathbf{G}$ is

$$L(\mathbf{G}) := \{s \in \Sigma^* | \delta(q_o, s) \text{ is defined}\}$$

and the marked language of $\mathbf{G}$ is $L_m(\mathbf{G}) := \{s \in L(\mathbf{G}) \,|\, \delta(q_o, s) \in Q_m\}$. All languages in this note are assumed to be *regular*, i.e., generated by finite automata.

A state $q \in Q$ is *reachable* if there exists a string $s_1 \in \Sigma^*$ such that $\delta(q_o, s_1) = q$ and *coreachable* if there exists a string $s_2 \in \Sigma^*$ such that $\delta(q, s_2) \in Q_m$. The DES $\mathbf{G}$ is reachable (coreachable) if every state $q \in Q$ is reachable (coreachable). Finally, we have that $\mathbf{G}$ is *trim* if it is both reachable and coreachable. Throughout the note, we assume that all automata (as well as their graphical representations) are trim, deterministic in their transition structures, and that their state size is minimal. This ensures that the correspondence between the automaton, the generated language, and the graphical representation of $\mathbf{G}$ is unique up to isomorphism.

A control theory of DES is based on partitioning the event set Σ into two disjoint subsets, of controllable and uncontrollable events, Σ_c and Σ_u respectively. A language $K \subseteq \Sigma^*$ is *controllable* with respect to $\mathbf{G}$ if

$$\bar{K}\Sigma_u \cap L(\mathbf{G}) \subseteq \bar{K} \tag{1}$$

or, equivalently

$$\Sigma_u \cap \Sigma_{L(\mathbf{G})}(s) \subseteq \Sigma_K(s)$$

for every $s \in \bar{K}$. Note that (1) is also equivalent to $\bar{K}\Sigma_u \cap L(\mathbf{G}) \subseteq \bar{K} \cap L(\mathbf{G})$. Namely, the controllability condition on K only constrains strings in the set $\bar{K} \cap L(\mathbf{G})$.

Control is achieved by means of a controller, or supervisor, which is allowed to disable any subset of controllable events after having observed an arbitrary string $s \in L(\mathbf{G})$. Formally, a supervisory control for $\mathbf{G}$ is any map $V : \Sigma^* \to \Gamma_\Sigma$, where

$$\Gamma_\Sigma = \{\gamma \in \mathcal{P}(\Sigma) \,|\, \Sigma_u \subseteq \gamma\}$$

represents the set of all *control patterns* on Σ. Here, as in the remainder of the note, $\mathcal{P}(\cdot)$ denotes the power set notation. In this specific instance, $\mathcal{P}(\Sigma)$ represents the set of all subsets of Σ. Note that this is an extension of the original definition from [18] where the mapping V is only defined on strings in $L(\mathbf{G})$. The extension is precisely what allows us to control multiple plants simultaneously. With either definition, an event $\sigma \in \Sigma_c$ is *enabled* after s if $\sigma \in V(s)$ and *disabled* otherwise. Uncontrollable events can never be disabled. The closed-loop system is denoted $V/\mathbf{G}$; the *closed behavior* of $V/\mathbf{G}$ is the language $L(V/\mathbf{G}) \subseteq L(\mathbf{G})$ obtained inductively as follows:

- $\varepsilon \in L(V/\mathbf{G})$;
- for any $s \in \Sigma^*$ and $\sigma \in \Sigma$, $s\sigma \in L(V/\mathbf{G})$ if and only if $\sigma \in V(s)$ and $s\sigma \in L(\mathbf{G})$.

The *marked behavior* of $V/\mathbf{G}$ is obtained as

$$L_m(V/\mathbf{G}) = L(V/\mathbf{G}) \cap L_m(\mathbf{G}).$$

In addition, we say that V is *nonblocking* if $\overline{L_m(V/\mathbf{G})} = L(V/\mathbf{G})$.

We now present two results from [18]. The first characterizes the supremal controllable sublanguage of a specification language E, while the second gives conditions under which $L_m(V/\mathbf{G}) = K$, for some $K \subseteq \Sigma^*$.

Proposition 1: Given a language $E \subseteq \Sigma^*$, the set

$$\mathcal{C}_\mathbf{G}(E) := \{K \subseteq E \,|\, K \text{is controllable with respect to } \mathbf{G}\}$$

is nonempty and is closed under arbitrary unions. In particular, $\mathcal{C}_\mathbf{G}(E)$ contains a supremal element $\sup\mathcal{C}_\mathbf{G}(E)$.

A supervisor implementing the supremal element is *maximally permissive* in that it disables the least number of controllable events while maintaining legal behavior in the closed loop. Before stating the second result, we need one more definition. Let $K \subseteq L \subseteq \Sigma^*$. The language K is *L-closed* if $K = \bar{K} \cap L$.

Theorem 2: Let $K \subseteq L_m(\mathbf{G}), K \neq \emptyset$. There exists a nonblocking supervisory control V for $\mathbf{G}$ such that $L_m(V/\mathbf{G}) = K$ if and only if K is controllable with respect to $\mathbf{G}$ and K is $L_m(\mathbf{G})$-closed.

The statement of Theorem 2 requires a test of $L_m(\mathbf{G})$-closedness in order to ensure that there exists a nonblocking supervisory controller V such that $L_m(V/\mathbf{G}) = K$. However, this property can be guaranteed a priori. Given a language $E \subseteq \Sigma^*$, we define $\mathcal{R}_\mathbf{G}(E)$ to be the set of sublanguages of E that are $L_m(\mathbf{G})$-closed. Namely

$$\mathcal{R}_\mathbf{G}(E) := \{K \subseteq E | K = \bar{K} \cap L_m(\mathbf{G})\}.$$

We then have the following characterization of $\mathcal{R}_\mathbf{G}(E)$. See, for example, [19].

Proposition 3: The set $\mathcal{R}_\mathbf{G}(E)$ is closed under arbitrary unions and

$$\sup\mathcal{R}_\mathbf{G}(E) = E - (L_m(\mathbf{G}) - E)\Sigma^*.$$

We end this section by presenting three results which are simple extensions of results previously obtained in [20]. We say that languages L and M are *nonconflicting* if $\overline{L \cap M} = \bar{L} \cap \bar{M}$. The first result, Lemma 4, characterizes the largest sublanguage of $E \subseteq \Sigma^*$ that is nonconflicting with languages $L_i \subseteq \Sigma^*$ for all i in some finite index set $\mathcal{I}$.

Lemma 4: Given languages $E, L_i \subseteq \Sigma^*$ for i in some finite index set $\mathcal{I} = \{0, 1, \ldots, n-1\}$, let $\mathcal{L} := \{L_0, \ldots, L_{n-1}\}$ and

$$\mathcal{N}_\mathcal{L}(E) := \{M \subseteq E | \overline{M \cap L_i} = \bar{M} \cap \overline{L_i} \text{ for all } i \in \mathcal{I}\}.$$

Then, $\sup\mathcal{N}_\mathcal{L}(E)$ is well-defined and

$$\sup\mathcal{N}_\mathcal{L}(E) = \lim_{k \to \infty} E_k$$

where

$$E_0 := E$$
$$E_{k+1} := \Phi_k(E_k)$$

and, for each nonnegative integer k, Φ_k is defined by

$$\Phi_k(X) := X - \left(\bar{X} \cap \bar{L}_{[k]} - \overline{X \cap L}_{[k]}\right)\Sigma^*$$

with

$$[k] := k(\mathrm{mod}\ n).$$

Before presenting the proof, we note that from [20, Th. 3(ii)] it can be deduced that $\Phi_k(Z)$ is the largest sublanguage of Z which is nonconflicting with $L_{[k]}$. For future reference, we call an element in $\mathcal{N}_{\mathcal{L}}(E)$ a *nonconflicting language with respect to L_i for all $i \in \mathcal{I}$*. The next lemma adds controllability and $L_m(\mathbf{G})$-closedness requirements to the previous result.

Lemma 5: Given languages E and L_i as in Lemma 4 and a DES $\mathbf{G}$ defined over Σ, let

$$\mathcal{RNC}_{\mathbf{G}}(E) := \mathcal{R}_{\mathbf{G}}(E) \cap \mathcal{N}_{\mathcal{L}}(E) \cap \mathcal{C}_{\mathbf{G}}(E). \tag{2}$$

Then, $\sup\mathcal{RNC}_{\mathbf{G}}(E)$ is well-defined and is the largest fixed point of the operator (on sublanguages of E) $\Omega : \mathcal{P}(E) \to \mathcal{P}(E)$ defined by

$$\Omega(Z) := \sup\mathcal{R}_{\mathbf{G}}(\sup\mathcal{N}_{\mathcal{L}}(\sup\mathcal{C}_{\mathbf{G}}(Z))).$$

The fixed point from Lemma 5 can be computed by iteratively imposing the nonconflicting, controllability, and $L_m(\mathbf{G})$-closedness properties as follows.

1) Let $E_0 = E$.
2) Compute $E_{3k-2} = \sup\mathcal{C}_{\mathcal{G}}(E_{3k-3}), E_{3k-1} = \sup\mathcal{N}_{\mathbf{L}}(E_{3k-2})$, and $E_{3k} = \sup\mathcal{R}_{\mathbf{G}}(E_{3k-1})$ for every positive integer k until $E_{3m} = E_{3m-3}$ for some integer $m > 0$.

Lemma 6: The above iteration scheme terminates after a finite number of steps and $E_\infty := \lim_{k-\infty} E_k = \sup\mathcal{RNC}_{\mathbf{G}}(E)$.

III. Robust Nonblocking Supervisory Control

In the sequel, we assume that the plant dynamics are not precisely known. Rather, we suppose that the true plant model $\mathbf{G}_{\text{true}}$ belongs to the set $\mathcal{G} := \{\mathbf{G}_i | i \in \mathcal{I}\}$, where $\mathcal{I}$ is a finite index set. For maximum flexibility at the modeling stage, each of the plant models is equipped with its own alphabet Σ_i, which may or may not have elements in common with the other plant models in $\mathcal{G}$. Namely, we have that $L(\mathbf{G}_i) \subseteq \Sigma_i^*$ for each $i \in \mathcal{I}$. The set of all events Σ is defined via

$$\Sigma := \bigcup_{i \in \mathcal{I}} \Sigma_i.$$

As usual, we denote the set of controllable and uncontrollable events by Σ_c and Σ_u, respectively. In addition, we require that all plants agree on which events are controllable and which are uncontrollable. Hence we have that for each $i \in \mathcal{I}$, the subset of controllable events of Σ_i can be computed by $(\Sigma_c)_i = \Sigma_c \cap \Sigma_i$.

We now define the set $\mathcal{V}$ of robust nonblocking supervisory controls (RNSCs). Specifically, $\mathcal{V}$ consists of supervisors which guarantee that the closed-loop behavior is nonblocking for each plant. Thus, we have

$$\mathcal{V} := \{V : \Sigma^* \to \Gamma_\Sigma | \overline{L_m(V/\mathbf{G}_i)} = L(V/\mathbf{G}_i) \text{ for each } i \in \mathcal{I}\} \tag{3}$$

Notice that we have a slight abuse of notation in the previous definition. In our setting, a supervisor for $\mathbf{G}_i$ is a map $V_i : \Sigma_i^* \to \Gamma_{\Sigma_i}$, where the equality $\Sigma_i = \Sigma$ need not hold. However, obtaining a supervisor V_i from $V \in \mathcal{V}$ is straightforward (simply let $V_i(s) := V(s) \cap \Sigma_i$ for every $s \in \Sigma_i^*$) and hence we will not distinguish between V and V_i when the context is obvious.

We now seek to characterize the set of languages over Σ that admit a robust nonblocking supervisory controller. To this end, we define a DES $\mathbf{G}$ via

$$L_m(\mathbf{G}) := \bigcup_{i \in \mathcal{I}} L_m(\mathbf{G}_i) \quad \text{and} \quad L(\mathbf{G}) := \bigcup_{i \in \mathcal{I}} L(\mathbf{G}_i).$$

We can now state the following theorem.

Theorem 7: Suppose $K \subseteq L_m(\mathbf{G})$ and $K \neq \emptyset$. There exists $V \in \mathcal{V}$ such that $L_m(V/\mathbf{G}) = K$ if and only if K is controllable with respect to $\mathbf{G}$, $L_m(\mathbf{G})$-closed, and nonconflicting with respect to $L_m(\mathbf{G}_i)$ for each $i \in \mathcal{I}$.

Throughout the sequel, we assume that $L_i = L_m(\mathbf{G}_i)$ in the definition of $\mathcal{N}_{\mathcal{L}}(E)$ and in (2). The theorem then says that given a specification language $E \subseteq \Sigma^*$, the set $\mathcal{RNC}_{\mathbf{G}}(E)$ consists precisely of all sublanguages of E that admit an RNSC. Before proving this result, we present Lemma 8, a straightforward generalization of [7, Prop. 2]. It provides us with a property that is useful in the proof of the above theorem.

Lemma 8: Suppose $\mathbf{G}$ and $\mathbf{G}'$ are DES over Σ with $L(\mathbf{G}') \subseteq L(\mathbf{G})$ and $L_m(\mathbf{G}') \subseteq L_m(\mathbf{G})$. If $V : \Sigma^* \to \Gamma_\Sigma$, then $L_m(V/\mathbf{G}') = L_m(V/\mathbf{G}) \cap L_m(\mathbf{G}')$.

Suppose K is a controllable sublanguage of $L_m(\mathbf{G})$ and that V is a supervisory controller (not necessarily robust) such that $L_m(V/\mathbf{G}) = K$. Lemma 8 says that $L_m(V/\mathbf{G}_i) = K \cap L_m(\mathbf{G}_i)$. Namely, through Theorem 2, we know that if $K \cap L_m(\mathbf{G}_i) \neq \emptyset$, then the language $K \cap L_m(\mathbf{G}_i)$ is controllable with respect to $\mathbf{G}_i$ and that V is a supervisor synthesizing this language. The following lemma additionally says that if $K \cap L_m(\mathbf{G}_i)$ is controllable with respect to $\mathbf{G}_i$ for each $i \in \mathcal{I}$, then K is controllable with respect to $\mathbf{G}$.

Lemma 9: Suppose $K \subseteq L_m(\mathbf{G})$ and $K \in \mathcal{N}_{\mathcal{L}}(E)$. Then K is controllable with respect to $\mathbf{G}$ if and only if $L_m(\mathbf{G}_i) \cap K$ is controllable with respect to $\mathbf{G}_i$ for each $i \in \mathcal{I}$.

We can now introduce the modeling framework for our robustness paradigm. This naturally leads to the robust nonblocking supervisory control problem which is the main thrust of this note. In our setting each of the plant models has its own specification language $E_i \subseteq \Sigma_i^*$ which constitutes the set of legal behaviors for that given model. Clearly, it is unreasonable to expect that in general there exists a RNSC V satisfying $L_m(V/\mathbf{G}_i) = \sup\mathcal{C}_{\mathbf{G}_i}(E_i \cap L_m(\mathbf{G}_i))$ for each $i \in \mathcal{I}$. Thus, we always seek the best approximation to this idealized case. In this context, we make the following definition.

Definition 10: We call $V \in \mathcal{V}$ a *maximally permissive robust nonblocking supervisory control* (MPRNSC) for $\{(\mathbf{G}_i, E_i) | i \in \mathcal{I}\}$ if for each $i \in \mathcal{I}, L_m(V/\mathbf{G}_i) \subseteq E_i$ and $L_m(V'/\mathbf{G}_i) \subseteq E_i$ implies $L_m(V'/\mathbf{G}_i) \subseteq L_m(V/\mathbf{G}_i)$ for every $V' \in \mathcal{V}$.

We are now able to state the robust nonblocking supervisory control problem which is the focus of this note.

Robust Nonblocking Supervisory Control Problem (RNSCP): Given the plants $\mathbf{G}_i$ and specifications E_i with $L(\mathbf{G}_i), E_i \subseteq \Sigma_i^*$ for each $i \in \mathcal{I}$, synthesize a maximally permissive robust nonblocking supervisory control $V : \Sigma^* \to \Gamma_\Sigma$, where $\Sigma = \bigcup_{i \in \mathcal{I}} \Sigma_i$.

Suppose $V \in \mathcal{V}$ and $s \in L(V/\mathbf{G}_i)$. The key to solving the above problem is the observation that if the string $s \in L_m(\mathbf{G}_i)$ for some $i \in \mathcal{I}$, we must then have that $s \in C_i := \sup\mathcal{C}_{\mathbf{G}_i}(E_i \cap L_m(\mathbf{G}_i))$. Fig. 1(a) shows this for the case where there are two plants, each with its own specification. The white areas in the diagram indicate which strings are necessarily prevented from occurring based on the above observation. In Theorem 11, $E := \bigcap_{i \in \mathcal{I}}(C_i \cup (\Sigma^* - L_m(\mathbf{G}_i))) \cap L_m(\mathbf{G})$ is used as the initial point of a fixed point calculation aimed at solving the RNSCP. Note that E represents the largest sublanguage of $L_m(\mathbf{G})$ satisfying $L_m(\mathbf{G}_i) \cap E \subseteq C_i$ for each $i \in \mathcal{I}$. Thus, from the definition of E, we also see that in this robustness framework the strings

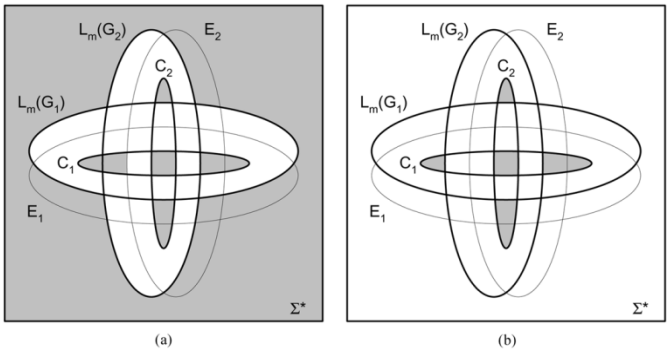

Fig. 1. Construction of E for two plant, two specification arrangement.

$w \in (\Sigma^* - L_m(\mathbf{G}_i))$ are used to enlarge the closed-loop behavior of our plant under the control of $V \in \mathcal{V}$ in the presence of the constraints simultaneously imposed by $\mathbf{G}_i$ and E_i for each $i \in \mathcal{I}$. The shaded area in Fig. 1(b) shows the language E for the two plant, two specification arrangement of Fig. 1(a). This language is obtained by taking the intersection of the shaded region in Fig. 1(a) with $L_m(\mathbf{G}_1) \cup L_m(\mathbf{G}_2)$.

Theorem 11: Let $C_i := \sup\mathcal{C}_{G_i}(E_i \cap L_m(\mathbf{G}_i))$, for each $i \in \mathcal{I}$, and let

$$E := \bigcap_{i\in\mathcal{I}}(C_i \cup (\Sigma^* - L_m(\mathbf{G}_i))) \cap L_m(\mathbf{G}). \tag{7}$$

Further, let K be the largest fixed point of the operator (on sublanguages of E) $\Omega : \mathcal{P}(E) \to \mathcal{P}(E)$ defined by

$$\Omega(Z) := \sup\mathcal{RNC}_{\mathbf{G}}(Z). \tag{8}$$

If $K \neq \emptyset$, a supervisor $V \in \mathcal{V}$ solves the RNSCP if and only if $L_m(V/\mathbf{G}) = K$.

The result says that the RNSCP has a solution for the given plant/specification pairs if and only if the supremal controllable sublanguage of E that is nonconflicting with each of the plants $\mathbf{G}_i$ is nonempty and $L_m(\mathbf{G})$-closed. Notice that the solution K must be computed using the fixed point operator Ω. This is necessary since we cannot otherwise guarantee that the resulting supremal controllable sublanguage of E will be nonblocking with respect to the individual plants $\mathbf{G}_i$.

Proof: Suppose V solves the RNSCP. By definition of C_i, we must then have that

$$L_m(V/\mathbf{G}_i) \subseteq C_i$$

for each $i \in \mathcal{I}$. By Lemma 8, this implies that

$$L_m(V/\mathbf{G}) \cap L_m(\mathbf{G}_i) \subseteq C_i$$

from which it follows that

$$L_m(V/\mathbf{G}) \subseteq C_i \cup (\Sigma^* - L_m(\mathbf{G}_i))$$

for each $i \in \mathcal{I}$. Since we also have that $L_m(V/\mathbf{G}) \subseteq L_m(\mathbf{G})$, it follows from (7) that

$$L_m(V/\mathbf{G}) \subseteq E$$

Appealing to Lemma 5, we conclude that $L_m(V/\mathbf{G}) \subseteq \sup\mathcal{RNC}_{\mathbf{G}}(E) = K$.

Also, since K is controllable, nonempty, and $L_m(\mathbf{G})$-closed, Theorem 7 says there exists $V \in \mathcal{V}$ such that $L_m(V/\mathbf{G}) = K$. From Lemma 8, we note that $L_m(V/\mathbf{G}_i) = L_m(V/\mathbf{G}) \cap L_m(\mathbf{G}_i) = K \cap L_m(\mathbf{G}_i)$ which in turn implies that

$$\begin{aligned} L_m(V/\mathbf{G}_i) &\subseteq E \cap L_m(\mathbf{G}_i) \\ &\subseteq C_i \\ &\subseteq E_i. \end{aligned}$$

Thus, if V solves the RNSCP, then $L_m(V/\mathbf{G}) = K$.

In order to prove the converse implication, we note that the second paragraph of the proof implies that any solution to the RNSCP must satisfy the relationship $L_m(V/\mathbf{G}) \supseteq K$. However, the first paragraph of the proof showed that $L_m(V/\mathbf{G}) \subseteq K$. Therefore, we conclude that the $V \in \mathcal{V}$ satisfying $L_m(V/\mathbf{G}) = K$ solves the RNSCP. □

Remark 12: Lin [13] gave necessary and sufficient conditions for the existence of a *marking* robust nonblocking supervisory controller V such that $L_m(V/\mathbf{G}_i) = K$ for each $i \in \mathcal{I}$ when $K \subseteq \bigcap_{i\in\mathcal{I}} L_m(\mathbf{G}_i)$ and $\Sigma_i = \Sigma$ for each $i \in \mathcal{I}$. Hence, his work is much more restrictive in general than that presented here. Specifically, in this case we have that (7) is replaced by the more restrictive condition $E = \bigcap_{i\in\mathcal{I}} C_i$ while (8) reduces to $\Omega(Z) = \sup\mathcal{R}_{\mathbf{G}}(\sup\mathcal{C}_{\mathbf{G}}(Z))$ since K and $L_m(\mathbf{G}_i)$ are nonconflicting by definition. Thus, we have that if $K \neq \emptyset$ and K is $L_m(\mathbf{G})$-closed, a supervisor $V \in \mathcal{V}$ solves the RNSCP if and only if $L_m(V/\mathbf{G}) = K$. If in addition, we place the burden of marking on the controller V, we can drop the $L_m(\mathbf{G})$-closedness requirement on K. (See [18] for additional details.) Park and Lim's work [16], [17] deals with a problem very similar to Lin's, only they extend Lin's framework to deal with nondeterministic systems.

Takai meanwhile relaxes the assumption on E in much the same way we do here, although he begins with a *closed* specification language E (K_{legal} in [14]) and only considers the case where $L_m(\mathbf{G}_i) = L(\mathbf{G}_i)$ for each $i \in \mathcal{I}$. He provides necessary and sufficient conditions for the existence of a maximally permissive robust supervisory controller. In this case, we find that $L(\mathbf{G}_i)$ and $L_m(V/\mathbf{G})$ are automatically nonconflicting. Moreover, the $L_m(\mathbf{G})$-closedness condition is also automatically satisfied and so (8) reduces to $\Omega(Z) = \sup\mathcal{C}_{\mathbf{G}}(Z)$. In this case, we deduce that if $K \neq \emptyset$, a supervisory controller $V \in \mathcal{V}$ solves the RNSCP if and only if $L(V/\mathbf{G}) = K$.

Another fundamental difference in the approach of [14] and the current note is that Takai assumes from the outset that the specification language K_{legal} is given. The framework of this note provides a natural setting for the modeling of plant/specification pairs. Namely, each specification is tailored directly to the plant to which it pertains. This note provides a systematic way of computing the language E from the plant/specification pairs, a task that may prove difficult in Takai's framework when there are competing requirements. ♢

We end this section with a discussion of the adaptive supervisory control scheme first introduced in [13]. In particular, we note that the framework introduced earlier in this section resolves some model uncertainties without having to recompute the supervisory control.

Adaptive supervision seeks to eliminate uncertainties in the plant model using observed strings of the system's behavior. If the string s has been observed in the closed-loop behavior and $s \notin L(\mathbf{G}_j)$, then we know that $\mathbf{G}_{\text{true}} \neq \mathbf{G}_j$ and we can recompute an optimal supervisor working only with the subset of plants $\{\mathbf{G}_i | i \in \mathcal{I} - \{j\}\}$. In [13], the proposed scheme requires that the optimal supervisor be recomputed after each new event appears in the string. However, in our framework

this is unnecessary. It is obvious from the definition of language closure that if $s \notin L(\mathbf{G}_j)$, then $sw \notin L(\mathbf{G}_j)$, for every $w \in \Sigma^*$. This means that after having observed the string s, the MPRNSC obtained via Theorem 11 is no longer constrained by $L(\mathbf{G}_j)$ and also no longer disables events in $\Sigma_j - \bigcup_{i\in\mathcal{I}-\{j\}} \Sigma_i$. In this case, uncertainty in the system is automatically resolved and the controller becomes optimal given its information regarding $\mathbf{G}_{\rm true}$. This is the subject of the following result.

Proposition 13: Suppose $C_i, L(\mathbf{G}), V$, and K are as defined in the statement of Theorem 11 and that the string s is observed in the closed-loop system $L(V/\mathbf{G})$. If $s \notin L(V/G_j)$ for some $j \in \mathcal{I}$, then $\mathbf{G}_{\rm true} \neq \mathbf{G}_j$. Moreover, we have that for every $t \in \Sigma^*$, $st \in L_m(V/\mathbf{G})$ if and only if $st \in L_m(V'/\mathbf{G})$, where V' is the maximally permissive robust supervisor over the set $\{(\mathbf{G}_i, E_i) \,|\, i \in \mathcal{I} - \{j\}\}$.

Proof: First, suppose there exists $t \in \Sigma^*$ such that $st \in C_j \cap L_m(V/\mathbf{G})$ for some $j \in \mathcal{I}$. This then implies that $st \in L_m(V/\mathbf{G}_j)$ since

$$\begin{aligned} C_j \cap L_m(V/\mathbf{G}) &= C_j \cap L_m(\mathbf{G}_j) \cap L_m(V/\mathbf{G}) \\ &\qquad \text{since } C_j \subseteq L_m(\mathbf{G}_j) \\ &= C_j \cap L_m(V/\mathbf{G}_j), \text{ by Lemma 8} \\ &= L_m(V/\mathbf{G}_j), \text{ by definition of } C_j. \end{aligned}$$

However, this means that $s \in L(V/\mathbf{G}_j)$, which is a contradiction. Thus, $st \notin C_j$ for every $t \in \Sigma^*$ which, by supremality of C_j, implies that $st \notin L_m(V/\mathbf{G}_j)$. It must then be the case that $st \in L_m(V/\mathbf{G})$ if and only if $st \in L_m(V/\mathbf{G}_i)$ for some $i \in \mathcal{I} - \{j\}$, since $st \notin C_j$ implies that $st \notin L_m(V/\mathbf{G}_j)$. Hence, $\mathbf{G}_{\rm true} \neq \mathbf{G}_j$.

Now, by definition of V', we must have that $L_m(V/\mathbf{G}_i) \subseteq L_m(V'/\mathbf{G}_i)$ for each $i \in \mathcal{I} - \{j\}$. Thus, the proof is complete if we can show that $st \in L_m(V'/\mathbf{G}_i)$ implies that $st \in L_m(V/\mathbf{G}_i)$ for each $i \in \mathcal{I} - \{j\}$. To this end, let $V'' \in \mathcal{V}$ be a supervisor for $\mathbf{G}$ such that $V''(st) = V'(st)$ for every $t \in \Sigma^*$. It then follows that $st \in L_m(V'/\mathbf{G}_i)$ if and only if $st \in L_m(V''/\mathbf{G}_i)$. Since the supervisor V is maximally permissive, we also have that $V''(st) \subseteq V(st)$ from which we can deduce that $st \in L_m(V''/\mathbf{G}_i)$ implies $st \in L_m(V/\mathbf{G}_i)$. This completes the proof. □

Remark 14: The central theme of this note is robustness of discrete-event systems, although many of the results could have been motivated from an adaptive supervisory control viewpoint. This choice stems from the fact that a supervisor obtained through Theorem 11 is always robust (in the sense discussed earlier this section) but may not be adaptive. This situation occurs for example when for one of the candidate plant models $\mathbf{G}_j$, we have $L(\mathbf{G}_i) \subseteq L(\mathbf{G}_j)$ and $L_m(\mathbf{G}_i) \subseteq L_m(\mathbf{G}_j)$ for every $i \in \mathcal{I}$. ◊

IV. Timed Discrete Event Systems

We now extend the results of the last section to deal with the timed discrete event systems (TDES) of [2]. The main difference between TDES and the DES we have dealt with thus far is the introduction of the special tick event which denotes the passage of one unit of time in the global clock. All the other events in a TDES occur relative to the occurrence of ticks.

A control theory on TDES is established in a twofold manner. First, the set of controllable events is defined to be those which can be indefinitely disabled by a supervisor. The set of uncontrollable events is given by $\Sigma_u := \Sigma - \Sigma_c$. Second, we define a set of *forcible* events for TDES, $\Sigma_{\rm for} \subseteq \Sigma$, which are used to preempt the $tick$ event. From a control theoretic perspective, this makes TDES fundamentally different from DES.

In the TDES framework, a supervisor V for $\mathbf{G}$ is said to be *admissible* if the following condition holds for all $s \in L(V/\mathbf{G})$:

$$[\Sigma_{L(V/\mathbf{G})}(s) \cap \Sigma_{\rm for} = \emptyset] \wedge [\text{ tick} \in \Sigma_{L(\mathbf{G})}(s)] \Rightarrow [\text{ tick} \in V(s)].$$

Namely, the $tick$ event can only be disabled if some forcible event is able to preempt it. We note also that Theorem 2 holds for TDES whenever the supervisor involved is admissible.

The robustness framework we use for TDES is similar to the one introduced earlier for untimed DES. Here we must assume that all plants agree on the forcibility of events in addition to controllability and uncontrollability. Moreover, we require that the tick transition represents the passage of the same duration of time in each of the plant models. The final difference is in the definition of $\mathcal{V}$, the set of robust nonblocking supervisory controls. For TDES, we say that a supervisory control $V \in \mathcal{V}$ if in addition to the conditions in (3), V is an admissible supervisor for $\mathbf{G}_i$ for each $i \in \mathcal{I}$. Otherwise, the framework is unchanged.

One of the difficulties with robust TDES is that an admissible supervisor for $\mathbf{G}$ is not necessarily an admissible supervisor for each $\mathbf{G}_i, i \in \mathcal{I}$. See [15, ex. 1]. Clearly, this problem is resolved if V is an admissible supervisor for each $\mathbf{G}_i, i \in \mathcal{I}$. By Theorem 2 and Lemma 8, this is equivalent to having $L_m(V/\mathbf{G}_i)$ be controllable with respect to $\mathbf{G}_i$ for each $i \in \mathcal{I}$. Given a language E, let $\mathcal{L} := \{L_m(\mathbf{G}_0), \ldots, L_m(\mathbf{G}_{n-1})\}$ and

$$\begin{aligned} \mathcal{A}_{\mathcal{L}}(E) := \{K \subseteq E | K \text{ is controllable} \\ \text{with respect to } \mathbf{G}_i \text{ for all } i \in \mathcal{I}\} \end{aligned}$$

Clearly, $\mathcal{A}_{\mathcal{L}}(E)$ is closed under arbitrary unions and so $\sup\mathcal{A}_{\mathcal{L}}(E)$ exists and is well defined.

This suggests that we can generalize Theorem 11 for all TDES by starting with the language E as in (7) and iteratively imposing controllability with respect to $\mathbf{G}$, making the result nonconflicting with each $\mathbf{G}_i$, imposing $L_m(\mathbf{G})$-closedness, and then imposing controllability with respect to $\mathbf{G}_i$ for each $i \in \mathcal{I}$. Namely, to solve the RNSCP for TDES in general, we need to compute $\sup\mathcal{ARNC}_{\mathbf{G}}(E)$, where $\mathcal{ARNC}_{\mathbf{G}}(E) := \mathcal{A}_{\mathcal{L}}(E) \cap \mathcal{R}_{\mathbf{G}}(E) \cap \mathcal{N}_{\mathcal{L}}(E) \cap \mathcal{C}_{\mathbf{G}}(E)$. From [15, Lemma 1], it is easy to deduce that $\mathcal{A}_{\mathcal{L}}(E) \subseteq \mathcal{C}_{\mathbf{G}}(E)$, and so one step in the above four step iteration is redundant.

We are now in a position to present our main result on robust TDES, a generalization of Theorem 11. Note that it generalizes Takai's results in [15] much in the same way that Theorem 11 generalizes the results in [14]. See [21] for the details of the proof.

Theorem 15: Given TDES $\{\mathbf{G}_i | i \in \mathcal{I}\}$, let $\mathbf{G}$ be defined via

$$L_m(\mathbf{G}) = \bigcup_{i\in\mathcal{I}} L_m(\mathbf{G}_i) \quad \text{and} \quad L(\mathbf{G}) = \bigcup_{i\in\mathcal{I}} L(\mathbf{G}_i)$$

and let $C_i := \sup\mathcal{C}_{G_i}(E_i \cap L_m(\mathbf{G}_i))$, for each $i \in \mathcal{I}$. Further let

$$E := \bigcap_{i\in\mathcal{I}} (C_i \cup (\Sigma^* - L_m(\mathbf{G}_i))) \cap L_m(\mathbf{G})$$

and let K be the largest fixed point of the operator (on sublanguages of E) $\Omega : \mathcal{P}(E) \to \mathcal{P}(E)$ defined by

$$\Omega(Z) := \sup\mathcal{ARN}_{\mathbf{G}}(Z)$$

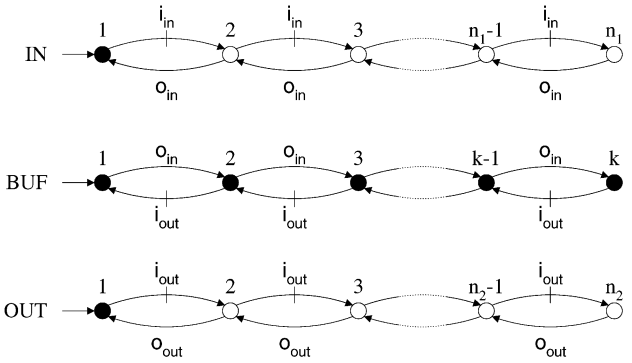

Fig. 2. Assembly line configurations for Example 17.

If $K \neq \emptyset$, a supervisor $V \in \mathcal{V}$ solves the RNSCP if and only if $L_m(V/\mathbf{G}) = K$. Moreover

$$\sup\mathcal{ARN}_{\mathbf{G}}(E) = \sup\mathcal{A}_{\mathcal{L}}(\sup\mathcal{R}_{\mathbf{G}}(\sup\mathcal{N}_{\mathcal{L}}(E))) = \lim_{k\to\infty} E_k$$

where

$$\begin{aligned} E_0 &:= E \\ E_{3k-2} &:= \sup\mathcal{N}_{\mathcal{L}}(E_{3k-3}) \\ E_{3k-1} &:= \sup\mathcal{R}_{\mathbf{G}}(E_{3k-2}) \\ \text{and } E_{3k} &:= \sup\mathcal{A}_{\mathcal{L}}(E_{3k-1}) \end{aligned}$$

for every positive integer k, and there exists an integer N such that $E_\infty = E_k$ for all $k \geq N$.

V. Example

We now present an example which helps further understand the theory of robustness developed in the previous sections. The example shows a manufacturing line that utilizes one of four configurations for testing its finished product.

In the manufacturing line, the parts are sequentially processed by a pair of machines. In order to enforce quality control, the plant employs feedback from a test unit (TU) in order to gauge its process. Four testing configurations are possible and on a given day, any of these configurations is possible. In the first configuration (P1), there is no feedback. Parts that pass the test are kept while parts that fail are simply discarded. In configuration P2, a failed test results in a part being passed through M2 a second time. In configuration P3, a part that fails will pass through the entire line again. In this case, a failed part is given priority over a new part. Finally, in P4, a part that fails the test is passed to a new machine (M4) for refining and is subsequently retested.

Automata representing each of the components in Fig. 3 are shown in Fig. 3. The marked states in Fig. 4 are those represented by solid nodes. In the test unit, pass denotes a passed test, whereas fail indicates that the part has failed. In this example, each of the buffers is assumed to have a capacity equal to the number of machines that feed into it. The set of controllable events is $\{\mathrm{in}_1, \mathrm{in}_2, \mathrm{in}_r, \mathrm{in}_4, \mathrm{in}_{\mathrm{TU}}\}$ and the uncontrollable events are $\{\mathrm{out}_1, \mathrm{out}_2, \mathrm{out}_4, \mathrm{pass}, \mathrm{fail}\}$. The specifications are used to prevent underflow and overflow of the and are collectively shown in Fig. 4.

In this example, we find that the supervisor is conservative, at least initially. For instance, machine M1 is initially prevented from producing a second work piece until M2 begins work on a part. This is because the supervisor does not know whether there is a one-slot buffer (B1) or a two-slot buffer (B3) following M1. More interestingly, regardless of which two (or more) configurations we admit initially, the supervisor remains conservative. For example, if we remove P3 from the set of possible plants before computing the robust supervisor, our controller does not become more permissive. However, if the supervisor observes the event in_4, it "knows" that configuration P4 is the correct configuration and subsequently behaves optimally as described in Section III.

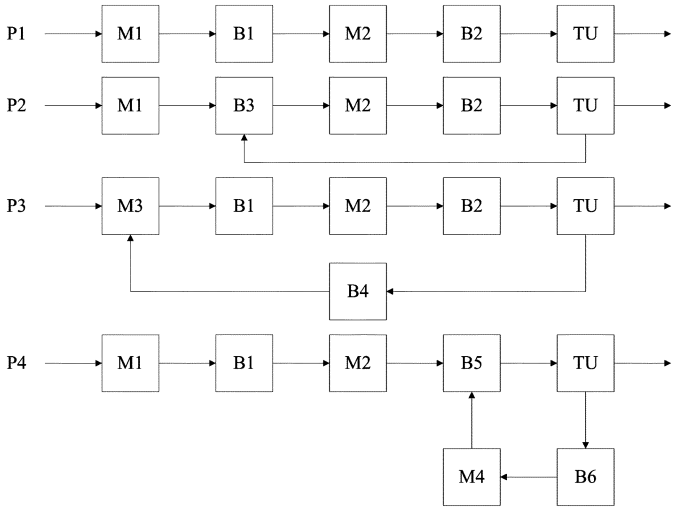

Fig. 3. Component machines for Example 17.

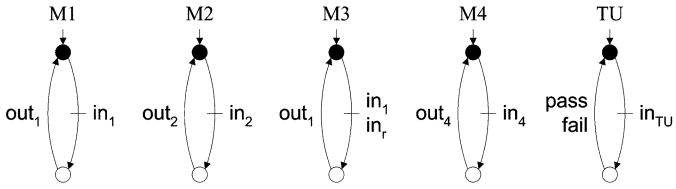

Fig. 4. Specifications for Example 17.

VI. Conclusion

In this note, we have extended the results of [13], [14], and [15]. We introduce a framework for the modeling of robust control of DES which is both natural and expressive. Within this framework, we solve the robust nonblocking supervisory control problem for both timed and untimed systems. The keys to the solution are the resolution of conflict in the various specifications and that strings not belonging to the closed behavior of a system require no control intervention.

References

[1] S. Bourdon, M. Lawford, and W. Wonham, "Robust nonblocking supervisory control of discrete-event systems," in *Proc. Amer. Control Conf.*, 2002, pp. 730–735.

[2] B. Brandin and W. Wonham, "The supervisory control of timed DES," *IEEE Trans. Autom. Control*, vol. 39, no. 2, pp. 329–342, Feb. 1994.

[3] K.-H. Cho and J.-T. Lim, "Stability and robustness of discrete event dynamic systems," *Int. J. Syst. Sci.*, vol. 28, no. 7, pp. 691–703, 1997.

[4] C. Özveren and A. Willsky, "Output stabilizability of discrete-event dynamic systems," *IEEE Trans. Autom. Control*, vol. 36, no. 8, pp. 925–935, Aug. 1991.

[5] ——, "Stability and stabilizability of discrete event dynamic systems," *J. Assoc. Comput. Mach.*, vol. 38, no. 3, pp. 730–752, 1991.

[6] M. Lawley and W. Sulistyono, "Robust supervisory control policies for manufacturing systems with unreliable resources," *IEEE Trans. Robot. Automat.*, vol. 18, no. 3, pp. 346–359, Jun. 2002.

[7] J. Cury and B. Krogh, "Design of robust supervisors for discrete event systems with infinite behaviors," in *Proc. IEEE Conf. Decision and Control*, 1996, pp. 2219–2224.

[8] ——, "Robustness of supervisors for discrete-event systems," *IEEE Trans. Autom. Control*, vol. 44, no. 2, pp. 376–379, Feb. 1999.
[9] S. Takai, "Synthesis of maximally permissive and robust supervisors for prefix-closed language specifications," *IEEE Trans. Autom. Control*, vol. 47, no. 1, pp. 132–136, Jan. 2002.
[10] ——, "Maximizing robustness of supervisors for partially observed discrete event systems," *Automatica*, vol. 40, no. 3, pp. 531–535, 2004.
[11] S.-J. Park and J.-T. Lim, "Robust nonblocking supervisor for discrete event systems with model uncertainty under partial observation," *IEEE Trans. Autom. Control*, vol. 45, no. 12, pp. 2393–2396, Dec. 2000.
[12] D. Hoffman and D. Weiss, Eds., *Software Fundamentals: Collected Papers of David L. Parnas*. Boston, MA: Addison-Wesley, 2001.
[13] F. Lin, "Robust and adaptive supervisory control of discrete event systems," *IEEE Trans. Autom. Control*, vol. 38, no. 12, pp. 1848–1852, Dec. 1993.
[14] S. Takai, "Maximally permissive robust supervisors for a class of specification languages," in *Proc. IFAC Conf. System Structure and Control*, vol. 2, 1998, pp. 429–434.
[15] ——, "Robust supervisory control of a class of timed discrete event systems under partial observation," *Syst. Control Lett.*, vol. 39, no. 4, pp. 267–273, 2000.
[16] S.-J. Park and J.-T. Lim, "Robust and nonblocking supervisory control of nondeterministic discrete event systems using trajectory models," *IEEE Trans. Autom. Control*, vol. 47, no. 4, pp. 655–658, Apr. 2002.
[17] ——, "On robust and nonblocking supervisor for nondeterministic discrete event systems," *IEICE Trans. Inf. Syst.*, vol. E86-D, no. 2, pp. 330–333, 2003.
[18] P. Ramadge and W. Wonham, "Supervisory control of a class of discrete-event processes," *SIAM J. Control Optim.*, vol. 25, no. 1, pp. 206–230, 1987.
[19] R. Kumar and V. Garg, *Modeling and Control of Logical Discrete Event Systems*. Boston, MA: Kluwer, 1995.
[20] E. Chen and S. Lafortune, "On nonconflicting languages that arise in supervisory control of discrete event systems," *Syst. Control Lett.*, vol. 17, no. 2, pp. 105–113, 1991.
[21] S. Bourdon, M. Lawford, and W. Wonham, "Robust nonblocking supervisory control of discrete-event systems," Dept. Comput. Software, McMaster Univ., Software Quality Research Lab. Tech. Rep. 9, Feb. 2003.

On the Equivalence of Three Independently Developed Phase-Locked Loops

Alireza K. Ziarani and Masoud Karimi-Ghartemani

***Abstract*—Three independent research groups have proposed three apparently different architectures for an improved phase-locked loop (PLL) structure through three entirely different approaches. It is shown that all three PLLs are structurally and mathematically the same. A linear analysis is performed and concludes that the available theory for the design of conventional PLLs can be leveraged into the design of the new class of PLLs.**

***Index Terms*—Costas loop, frequency estimation, phase detection, phase-locked loop (PLL), signal decomposition.**

I. Introduction

The phase-locked loop (PLL) is a fundamental concept that finds application in diverse areas such as communications, computers, instrumentation devices, control and power systems [1]–[7]. Its structure is usually thought of as being comprised of three building blocks: phase detector (PD), loop filter (LF) and voltage-controlled oscillator (VCO), as depicted in Fig. 1. The PD is conventionally a multiplier and the LF is a low-pass filter. The VCO generates an output signal, the phase angle of which is in lock with that of the input signal.

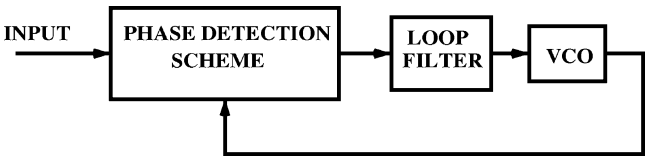

Fig. 1. Conventional PLL structure.

Despite the long history of its usage dating back to 1930s, the PLL is the subject of an active research aiming at improving its performance. Among others, three research groups have developed three apparently different PLL architectures alternative to the conventional structure. Although emerged from entirely different areas using entirely different approaches, the three PLL structures are equivalent, a fact which is shown in this note.

The first of the three PLL structures studied in this note was reported briefly in [8] with no elaboration on how it was developed. A circuit capable of locking to more than one frequency was disclosed that is useful in signal separators, notch filters and tracking filter circuitries.

The second PLL discussed in this note is a magnitude/phase-locked loop presented in [9], [10] where its stability is analyzed and its functionality for frequency estimation is verified. This PLL is controlled by three parameters g_ω, g_m and K_f, and is a modified version of the one used in [11] for the rejection of sinusoidal disturbances. This work has somewhat heuristically emerged from the studies in the area of active noise control [12].

The third PLL structure studied here has emerged from the works of the authors and their colleagues in the area of adaptive filtering [13] and was initially viewed as a signal processing technique for the extraction of nonstationary sinusoids [14], [15]. Some of its applications in the areas of biomedical [16], [17], mechanical [18], and power engineering [19] were developed, and proofs of its mathematical properties were furnished [20]. Soon, the authors realized that the proposed signal processing algorithm can be thought of as a PLL [21], and as such presented this view in [22].

The following section revisits the three PLL systems and shows their mathematical as well as structural equivalence. In Section III, it is shown that the new system can be envisaged as a conventional PLL further equipped with an amplitude estimator and external control loop. Section III also establishes an analogy between the new PLL system and the well-established Costas PLL. Some issues related to system stability are presented in Section IV where a linear analysis of the system is also presented. Finally, it is shown how the available theory for the design of conventional PLLs can be leveraged into this class of PLLs.

II. Overview of the Three PLL Topologies

This section provides an overview of the three PLL systems. In each case, its basic block diagram is presented and governing differential equations are derived. Equivalence of the three PLL topologies is discussed.

A. Residual Mode PLL

The residual mode phase-locked loop described in [8] is shown in Fig. 2(a). The input signal is subtracted from an estimated signal to

Manuscript received April 9, 2004; revised November 9, 2004. Recommended by Associate Editor M. Demetriou. This work was supported in part by the National Science Foundation under Award BES-0447705.

A. K. Ziarani is with the Department of Electrical and Computer Engineering, Clarkson University, Potsdam, NY USA 13699-5720 (e-mail: aziarani@clarkson.edu).

M. Karimi-Ghartemani is with the Edward S. Rogers Sr. Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON M5S 3G4, Canada (e-mail: masoud@ele.utoronto.ca).

Digital Object Identifier 10.1109/TAC.2005.860239