Robust Nonblocking Supervisory Control of Discrete-Event Systems

Sean E. Bourdon[*†] M. Lawford [‡] W.M. Wonham[†]

Abstract

In the first part of this paper, we generalize a notion of robust supervisory control 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. The second part of this paper extends these results to deal with timed discrete-event systems.

1 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 paper, 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.

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 [10, 9, 3] for representative papers in this area. 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 behaviour. Cury and Krogh [4, 5], and Takai [16] 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 behaviour is within specified bounds. This

[*]Corresponding author

[†]Systems Control Group, Edward S. Rogers Sr. Department of Electrical and Computer Engineering, University of Toronto, Toronto, Ontario, Canada, M5S 3G4, e-mail:{bourdon,wonham}@control.toronto.edu

[‡]Department of Computing and Software, McMaster University, Hamilton, Ontario, Canada, L8S 4K1, e-mail: lawford@mcmaster.ca

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 results of this paper 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 paper provides a framework in which we are able to synthesize a single controller for the entire group of robots. As a second example, consider an assembly line in which a piece is first worked on by one of n_1 identical input machines, placed in a k slot buffer, and then worked on by one of n_2 identical output machines. Further suppose that at any given time, any given number of input and output machines may be taken out of service. Each of the $2^{n_1+n_2}$ configurations for the assembly line can then be considered a set of possible dynamics for the system.

Lin [8] 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 behaviour of each of the plants is equal to K. The paper also provides an adaptive control scheme under which the supervisor can be refined in the presence of observations of the system's behaviour. Takai [14] generalized these results to the case where the legal behaviour is no longer necessarily a sublanguage of each possible plant language. Takai later generalized his own robustness results to deal with timed DES in [15]. Park and Lim [12] 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 paper is an extension of Lin's work to the most general case and is closely related to Takai's. Specifically, 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 plant in the family of models. Also, as will become clear later, we are able to achieve significant gains over the results obtained using [8]. Although our work is similar to Takai's, there are two distinct differences. First, we deal with *marked*behaviours 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 point we develop in Remark 9.

2 Supervisory Control of Discrete-Event Systems

We begin with a brief introduction to the supervisory control theory initiated by Ramadge and Wonham [13]. 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*, in Σ including the empty string, ε. Any subset of Σ^* is called a *language* over Σ. Given a language $L \subseteq \Sigma^*$ and a string $s \in \Sigma^*$, the set of *eligible* events at s in L is given by

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

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:

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

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

$$\mathbf{G} := (Q, \Sigma, \delta, q_{\mathrm{o}}, Q_m),$$

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

$$L(\mathbf{G}) := \{s \in \Sigma^* \mid \delta(q_{\mathrm{o}}, s) \text{ is defined}\},$$

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

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

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

or equivalently

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

for every $s \in \overline{K}$. Note that the controllability condition on K only constrains strings in the set $\overline{K} \cap L(\mathbf{G})$, since from (1), we have that $\overline{K}\Sigma_u \cap L(\mathbf{G}) \subseteq \overline{K} \cap L(\mathbf{G})$.

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

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

represents the set of all *control patterns* on Σ. Note that this is an extension of the original definition from [13] 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 behaviour* 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 behaviour* 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 [13]. 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 $K \subseteq \Sigma^*$. Given a language $E \subseteq \Sigma^*$, let

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

We then have the following result.

Proposition 1 *The set $\mathcal{C}_{\mathbf{G}}(E)$ 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 behaviour 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 $K = \overline{K} \cap L$.

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.*

We end this section by presenting two results which are simple extensions of results previously obtained in [2]. We say that languages L and M are *nonconflicting* if $\overline{L \cap M} = \overline{L} \cap \overline{M}$. The first result, Lemma 3 below, 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 3 *Given languages E, $L_i \subseteq \Sigma^*$ for i in some finite index set $\mathcal{I} = \{0, 1, \ldots, n-1\}$, let*

$$\mathcal{N}_E := \{M \subseteq E \mid \overline{M \cap L_i} = \overline{M} \cap \overline{L_i} \text{ for all } i \in \mathcal{I}\}.$$

Then, $\sup \mathcal{N}_E$ is well-defined and

$$\sup \mathcal{N}_E = \lim_{k \to \infty} E_k,$$

where

$$E_{-1} := E,$$
$$E_k := \Phi_k(E_{k-1}),$$

and, for each non-negative integer k, Φ_k is defined by

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

with

$$[k] := k \pmod{n}.$$

Before presenting the proof, we note that from Theorem 3 (ii) in [2] it can be deduced that $\Phi_k(Z)$ is the largest sublanguage of Z which is nonconflicting with $L_{[k]}$.

Proof: First we show that $\sup \mathcal{N}_E$ exists. We accomplish this by showing that $\mathcal{N}_E$ is closed under arbitrary unions. Let $M_j \in \mathcal{N}_E$ for all j in some arbitrary index set $\mathcal{J}$. It then follows that $\bigcup_{j\in\mathcal{J}} M_j \subseteq E$. Also, by distributivity of language intersection and language closure over unions of languages, we have that

$$\overline{\Big(\bigcup_{j\in\mathcal{J}} M_j\Big) \cap L_i} = \overline{\bigcup_{j\in\mathcal{J}} (M_j \cap L_i)} = \bigcup_{j\in\mathcal{J}} (\overline{M_j \cap L_i}).$$

Now, since $M_j \in \mathcal{N}_E$ for each j, it follows that $\overline{M_j} \cap \overline{L_i} = \overline{M_j \cap L_i}$. Using this last equality and distributivity, we have that

$$\bigcup_{j\in\mathcal{J}} (\overline{M_j \cap L_i}) = \bigcup_{j\in\mathcal{J}} (\overline{M_j} \cap \overline{L_i}) = \Big(\bigcup_{j\in\mathcal{J}} \overline{M_j}\Big) \cap \overline{L_i} = \overline{\bigcup_{j\in\mathcal{J}} M_j} \cap \overline{L_i},$$

from which we conclude that

$$\overline{\Big(\bigcup_{j\in\mathcal{J}} M_j\Big) \cap L_i} = \overline{\bigcup_{j\in\mathcal{J}} M_j} \cap \overline{L_i}$$

for each $i \in \mathcal{I}$. Thus, $\bigcup_{j\in\mathcal{J}} M_j \in \mathcal{N}_E$ which completes this part of the proof.

We now show that $\sup \mathcal{N}_E = E_\infty$, where $E_\infty := \lim_{k\to\infty} E_k$. First, we show that E_∞ exists. Clearly $\{E_n\}$ is a monotonically decreasing sequence bounded from below by $\emptyset$. Thus, E_∞ exists and $E_\infty = \bigcap_{n\in\mathbb{N}_o\cup\{-1\}} E_n$, where $\mathbb{N}_o = \{0, 1, 2, \dots\}$ is the set of natural numbers.

Next we show $\sup \mathcal{N}_E \subseteq E_\infty$ using mathematical induction. By definition, $\sup \mathcal{N}_E \subseteq E_{-1} = E$. Now assume that $\sup \mathcal{N}_E \subseteq E_k$ for some integer $k \geq -1$. We complete this part of the proof by showing that $\sup \mathcal{N}_E \subseteq E_{k+1} = \Phi_{k+1}(E_k)$. Now, $\Phi_{k+1}(E_k)$ is the largest sublanguage of E_k which is nonconflicting with $L_{[k+1]}$ and so the inclusion follows since $\sup \mathcal{N}_E$ is necessarily nonconflicting with L_i for each $i \in \mathcal{I}$.

To show inclusion in the opposite direction, we prove that there exists an integer N such that $E_\infty = E_k$ for all $k \geq N$. With this being the case, we have that E_∞ is nonconflicting with L_i for each $i \in \mathcal{I}$. Namely, $E_\infty \in \mathcal{N}_E$ from which it follows that $E_\infty \subseteq \sup \mathcal{N}_E$. Now, we say that an automaton $\mathbf{G}$ *represents* a language K if $L_m(\mathbf{G}) = K$ and $L(\mathbf{G}) = \overline{K}$. In [2] it is shown that an automaton representing E_{k+1} can be obtained from an automaton representing E_k simply by deleting some of its states and transitions. By regularity of $E_{-1} = E$, it follows that an automaton representing E_{-1} has only finitely many states and transitions. Hence, there must exist an integer N such that $E_n = E_N$ for every $n \geq N$. $\square$

The next lemma adds a controllability requirement to the previous result.

Lemma 4 *Given languages E and L_i as in Lemma 3 and a DES $\mathbf{G}$ defined over Σ, let*

$$\mathcal{CN}_{\mathbf{G}}(E) := \mathcal{N}_E \cap \mathcal{C}_{\mathbf{G}}(E). \tag{2}$$

Then, $\sup \mathcal{CN}_{\mathbf{G}}(E)$ 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

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

Proof: We show that $\mathcal{CN}_{\mathbf{G}}(E)$ is closed under arbitrary unions. We already know that $\mathcal{N}_E$ is closed under arbitrary unions from the proof of Lemma 3 while Proposition 1 shows that the same is true of $\mathcal{C}_{\mathbf{G}}(E)$. Thus, $\sup \mathcal{CN}_{\mathbf{G}}(E)$ exists and is well-defined. We conclude the proof by showing that $\sup \mathcal{CN}_{\mathbf{G}}(E)$ is the largest fixed point of Ω.

Let E_Ω denote the largest fixed point of Ω. By definition of Ω, $E_\Omega \subseteq E$ must be controllable with respect to $\mathbf{G}$ and nonconflicting with L_i for each $i \in \mathcal{I}$. Namely, $E_\Omega \in \mathcal{CN}_{\mathbf{G}}(E)$ from which it is immediate that $E_\Omega \subseteq \sup \mathcal{CN}_{\mathbf{G}}(E)$. Also, it is clear that $\sup \mathcal{CN}_{\mathbf{G}}(E)$ is a fixed point of Ω from which we have that $\sup \mathcal{CN}_{\mathbf{G}}(E) \subseteq E_\Omega$. □

The fixed point from Lemma 4 can be computed by iteratively imposing the nonconflicting and controllability properties as follows:

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

Lemma 5 *The above iteration scheme terminates after a finite number of steps and $E_\infty := \lim_{k\to\infty} E_k = \sup \mathcal{CN}_{\mathbf{G}}(E)$.*

Proof: The proof proceeds much as the proof of Lemma 3. First, we show that E_∞ exists. Clearly $\{E_k\}$ is a monotonically decreasing sequence bounded from below by $\emptyset$. Thus, E_∞ exists and $E_\infty = \bigcap_{k \in \mathbb{N}_o} E_k$.

Next we show $\sup \mathcal{CN}_{\mathbf{G}}(E) \subseteq E_\infty$. By definition, $\sup \mathcal{CN}_{\mathbf{G}}(E) \subseteq E_0 = E$. Now assume that $\sup \mathcal{CN}_{\mathbf{G}}(E) \subseteq E_k$ for some integer $k \geq 0$. We complete this part of the proof by showing that $\sup \mathcal{CN}_{\mathbf{G}}(E) \subseteq E_{k+1}$. If k is even, then $E_{k+1} = \sup \mathcal{N}_{E_k}$, the largest sublanguage of E_k which is nonconflicting with L_i for each $i \in \mathcal{I}$. Similarly, if k is odd, $E_{k+1} = \sup \mathcal{C}_{\mathbf{G}}(E_k)$, the largest sublanguage of E_k which is controllable with respect to $\mathbf{G}$. The inclusion follows in both cases since $\sup \mathcal{CN}_{\mathbf{G}}(E)$ is necessarily nonconflicting with L_i for each $i \in \mathcal{I}$ and controllable with respect to $\mathbf{G}$.

To show inclusion in the opposite direction, we prove that there exists an integer N such that $E_\infty = E_k$ for all $k \geq N$. With this being the case, we have that E_∞ is nonconflicting with L_i for each $i \in \mathcal{I}$ and controllable with respect to $\mathbf{G}$. It then follows that $E_\infty \subseteq \sup \mathcal{CN}_{\mathbf{G}}(E)$. In [2], it is shown that an automaton representing E_{k+1} when k is odd can be obtained from an automaton representing E_k simply by deleting some of its states and transitions. Moreover, it was shown in the proof of Lemma 3 that the same is true when k is even. By regularity of $E_0 = E$, it follows that an automaton representing E_0 has only finitely many states and transitions. Hence, there must exist an integer N such that $E_k = E_N$ for every $k \geq N$. □

3 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 $\mathbf{G}_{true}$ belongs to the set $\mathcal{G} := \{\mathbf{G}_i \mid i \in \mathcal{I}\}$, 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

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

In this setting each of the plant models has its own specification language $E_i \subseteq \Sigma_i^*$ which constitutes the set of legal behaviours for that given model. As usual, we denote the set of controllable and uncontrollable events by Σ_c and Σ_u, respectively. In addition, we assume that all plants must 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 (RNSC). Specifically, $\mathcal{V}$ consists of supervisors which guarantee that the closed-loop behaviour is nonblocking and consists only of legal strings for each plant/specification pair. Thus, we have

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

Notice that we have a slight abuse of notation in the above 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.

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 call $V \in \mathcal{V}$ a *maximally permissive robust nonblocking supervisory control* (MPRNSC) if for any $V' \in \mathcal{V}$, $L_m(V'/\mathbf{G}_i) \subseteq L_m(V/\mathbf{G}_i)$ for each $i \in \mathcal{I}$. We are now able to state the robust nonblocking supervisory control problem which is the focus of this paper.

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 $s \in C_i := \sup \mathcal{C}_{\mathbf{G}_i}(E_i \cap L_m(\mathbf{G}_i))$. Figure 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 this robustness framework, we use strings $w \in (\Sigma^* - L_m(\mathbf{G}_i))$ to enlarge the closed-loop behaviour 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 $i \in \mathcal{I}$. This is borne out in the definition of E in the statement of Theorem 6 below, the main result of the paper. The shaded area in Figure 1(b) shows the language E for the two plant, two specification arrangement of Figure 1(a). This language is obtained simply by taking the intersection of the shaded region in Figure 1(a) with $L_m(\mathbf{G}_1) \cup L_m(\mathbf{G}_2)$.

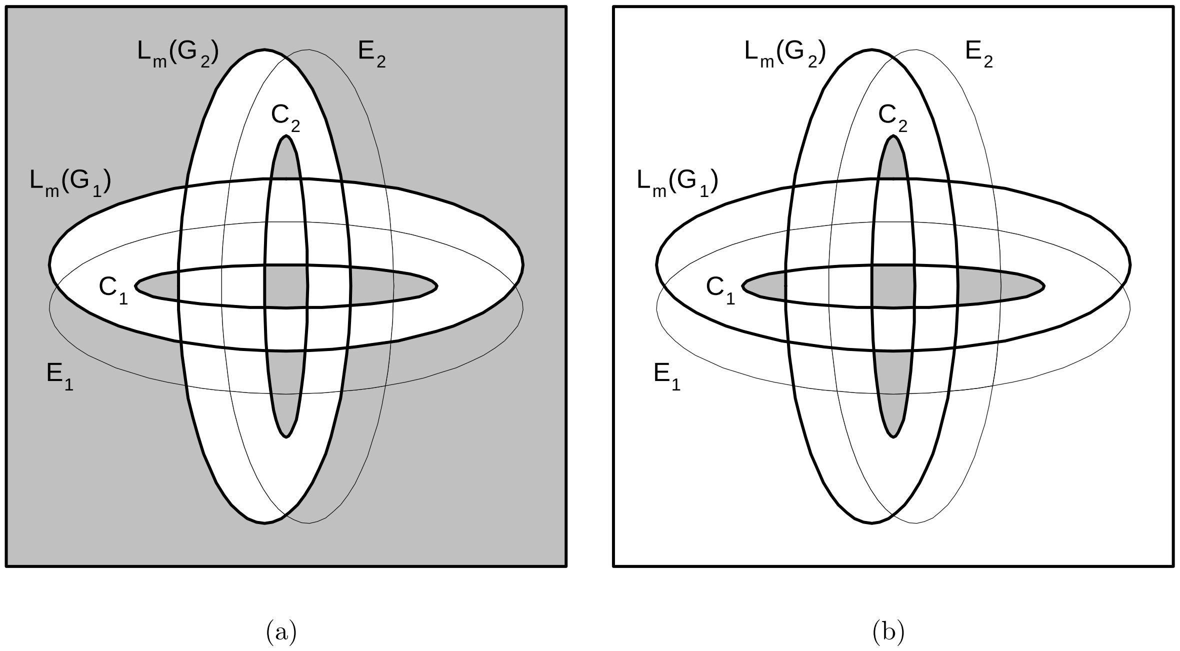

Figure 1: Construction of E for two plant, two specification arrangement

Theorem 6 *Let the DES* $\mathbf{G}$ *be defined via*

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

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*

$$E := \bigcap_{i\in\mathcal{I}} \Big(C_i \cup \big(\Sigma^* - L_m(\mathbf{G}_i)\big)\Big) \cap L_m(\mathbf{G}) \tag{4}$$

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*

$$\Omega(Z) := \sup \mathcal{CN}_{\mathbf{G}}(Z), \tag{5}$$

where the L_i *in* (2) *are replaced by* $L_m(\mathbf{G}_i)$. *If* $K \neq \emptyset$ *is* $L_m(\mathbf{G})$*-closed, a supervisor* $V \in \mathcal{V}$ *solves the Robust Nonblocking Supervisory Control Problem 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 also nonempty. 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 $\mathbf{G}_i$.

Remark 7 The statement of Theorem 6 requires a test for $L_m(\mathbf{G})$-closedness of the fixed point K. However, the fixed point calculation can be modified to ensure $L_m(\mathbf{G})$-closedness a priori. Let $\mathcal{R}_{\mathbf{G}}(Z)$ denote the lattice of sublanguages of Z that are $L_m(\mathbf{G})$-closed and let $\mathcal{RCN}_{\mathbf{G}}(Z) := \mathcal{R}_{\mathbf{G}}(Z) \cap \mathcal{CN}_{\mathbf{G}}(Z)$. It follows (e.g. [7]) that $\mathcal{R}_{\mathbf{G}}(Z)$ is closed under arbitrary unions and that

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

By replacing $\mathcal{CN}_{\mathbf{G}}(Z)$ with $\mathcal{RCN}_{\mathbf{G}}(Z)$ in Theorem 6, we find that there exists a solution to RNSCP whenever $K \neq \emptyset$. The proof that $\sup \mathcal{RCN}_{\mathbf{G}}(Z)$ exists and is well-defined is similar to the proof of Lemma 4. Moreover, $\sup \mathcal{RCN}_{\mathbf{G}}(Z)$ can be computed by adding a third step to the iteration scheme discussed in Lemma 5. Specifically, the fixed point can be computed by iteratively imposing the nonconflicting, controllability, and $L_m(\mathbf{G})$-closedness properties. The proof that this scheme converges in a finite number of iterations is similar to the proof of Lemma 5. $\Diamond$

Lemma 8 below is a straightforward generalization of Proposition 2 in [4] and provides us with a property that is useful in the proof of the above theorem. From this result, we can deduce that under the conditions of Theorem 6 the closed-loop behaviour of each of the component plants consists precisely of those strings in the closed-loop behaviour of $\mathbf{G}$ which are also in $L_m(\mathbf{G}_i)$.

Lemma 8 *Suppose* $\mathbf{G}$ *and* $\mathbf{G}'$ *are DES over the alphabet* 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* $L_m(V/\mathbf{G}') = L_m(V/\mathbf{G}) \cap L_m(\mathbf{G}')$.

Proof: By Proposition 2 of [5], we can deduce the following equality:

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

Since $L_m(V/\mathbf{G}') := L(V/\mathbf{G}') \cap L_m(\mathbf{G}')$ and $L_m(\mathbf{G}') \subseteq L(\mathbf{G}')$, the above implies that

$$\begin{aligned} L_m(V/\mathbf{G}') &= L(V/\mathbf{G}) \cap L(\mathbf{G}') \cap L_m(\mathbf{G}') \\ &= L(V/\mathbf{G}) \cap L_m(\mathbf{G}'). \end{aligned}$$

Finally, we find that

$$\begin{aligned} L_m(V/\mathbf{G}') &= L(V/\mathbf{G}) \cap L_m(\mathbf{G}) \cap L_m(\mathbf{G}') \\ &= L_m(V/\mathbf{G}) \cap L_m(\mathbf{G}') \end{aligned}$$

since $L_m(\mathbf{G}') \subseteq L_m(\mathbf{G})$ and $L_m(V/\mathbf{G}) := L(V/\mathbf{G}) \cap L_m(\mathbf{G})$. $\Box$

Proof of Theorem 6: Suppose V solves 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 (4) that

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

Appealing to Lemma 4, we conclude that $L_m(V/\mathbf{G}) \subseteq \sup \mathcal{CN}_{\mathbf{G}}(E) = K$.

Also, since K is controllable, nonempty, and $L_m(\mathbf{G})$-closed, Theorem 2 says there exists a nonblocking supervisory control V such that $L_m(V/\mathbf{G}) = K$. Thus, we need only show that $V \in \mathcal{V}$ in order to complete the proof. From Lemma 8, we note that $L_m(V/\mathbf{G}_i) = L_m(V/\mathbf{G}) \cap L_m(\mathbf{G}_i) = K \cap L_m(\mathbf{G}_i)$ from which we find that

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

Once again, we use Lemma 8, this time to deduce that $\overline{L_m(V/\mathbf{G}_i)} = \overline{K \cap L_m(\mathbf{G}_i)}$. By definition, K is nonconflicting with $L_m(\mathbf{G}_i)$ so that

$$\begin{aligned} \overline{L_m(V/\mathbf{G}_i)} &= \overline{L_m(V/G)} \cap \overline{L_m(\mathbf{G}_i)} \\ &= L(V/\mathbf{G}) \cap L(\mathbf{G}_i), \end{aligned}$$

since V is nonblocking. Recall (Proposition 2 of [5]) that $L(V/\mathbf{G}_i) = L(V/\mathbf{G}) \cap L(\mathbf{G}_i)$ so that $\overline{L_m(V/\mathbf{G}_i)} = L(V/\mathbf{G}_i)$. Hence, our proof is complete. $\Box$

Remark 9 Lin [8] 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 (4) is replaced by the more restrictive condition $E = \bigcap_{i \in \mathcal{I}} C_i$ while (5) reduces to $\Omega(Z) = \sup \mathcal{C}_{\mathbf{G}}(Z)$ 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 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 [13] for additional details.) Park and Lim's work [12] is similar, although they extend Lin's work to deal with nondeterministic systems.

Takai meanwhile relaxes the assumption on E much in 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 and so (5) once again reduces to $\Omega(Z) = \sup \mathcal{C}_{\mathbf{G}}(Z)$. Moreover, the $L_m(\mathbf{G})$-closedness condition is also automatically satisfied and so we deduce that in this case, if $K \neq \emptyset$, a supervisory controller $V \in \mathcal{V}$ solves RNSCP if and only if $L(V/\mathbf{G}) = K$.

Another fundamental difference in the approach of [14] and the current paper is that Takai assumes from the outset that the specification language K_{legal} is given. The framework of this paper 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 paper 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. $\Diamond$

We end this section with a discussion of the adaptive supervisory control scheme first introduced in [8]. 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 behaviour. If the string s has been observed in the closed-loop behaviour and $s \notin L(\mathbf{G}_j)$, then we know that $\mathbf{G}_{true} \neq \mathbf{G}_j$ and we can recompute an optimal supervisor working only with the subset of plants $\{\mathbf{G}_i \mid i \in \mathcal{I} - \{j\}\}$. In [8], 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 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 6 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. Namely, if $su \in C_i$, then we also have that $su \in L(V/\mathbf{G}) = L(V/\mathbf{G}_i)$. This is the subject of the following result.

Proposition 10 *Suppose C_i, $L(\mathbf{G})$, V, and K are as defined in the statement of Theorem 6 and that the string s is observed in the closed-loop system $L(V/\mathbf{G})$. If $s \in L(V/\mathbf{G}_i)$ for some $i \in \mathcal{I}$ and $s \notin L(V/G_j)$ for every $j \in \mathcal{I} - \{i\}$, then $\mathbf{G}_{true} = \mathbf{G}_i$ and, for every $t \in \Sigma^*$, $st \in L_m(V/\mathbf{G})$ if and only if $st \in C_i$.*

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

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

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

$st \in L_m(V/\mathbf{G}_i)$, since for every $j \in \mathcal{I} - \{i\}$, $st \notin C_j$ implies that $st \notin L_m(V/\mathbf{G}_j)$. Hence, $\mathbf{G}_{true} = \mathbf{G}_i$.

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

Remark 11 The central theme of this paper is robustness of discrete-event systems, although adaptive supervision could just as easily have been chosen instead. This choice is motivated by the fact that a supervisor obtained through Theorem 6 is always robust (in the sense discussed earlier this section) but may not be adaptive. An example of a robust supervisor which is not adaptive is shown below. $\diamondsuit$

4 Timed Discrete Event Systems

We now extend the results of the last section to deal with the timed discrete event systems (TDES) of [1]. The notion of forcible events, which we introduce shortly, make TDES fundamentally different from the DES we have dealt with thus far. However, we will also make it clear that in the absence of forcible events modifying the theory of robust DES to handle TDES is not difficult.

As in [1], we model a TDES $\mathbf{G}$ in two steps. The first is to construct an activity transition structure $\mathbf{G_{act}} := (A, \Sigma, \xi, a_o, A_m)$, which is simply a finite deterministic automaton [13]. In this definition, A is the set of *activities*, $a_o \in A$ is the *initial activity*, and A_m denotes the set of *marked activities*. We use Σ to denote the set of system events. The partial function $\xi : A \times \Sigma \to A$ is called the *activity transition function* and we write $\xi(a, \sigma)!$ whenever $\xi(a, \sigma)$ is defined.

The second step in constructing a TDES consists of introducing time into the activity transition structure. In a TDES, time is measured via the *tick* of a discrete global clock, relative to which the enablement of each of the events in Σ is defined. Specifically, to each event $\sigma \in \Sigma$ we assign lower and upper time bounds $l_\sigma \in \mathbb{N}_o$ and $u_\sigma \in \mathbb{N}_o \cup \{\infty\}$, respectively, which satisfy $l_\sigma \leq u_\sigma$. In the sequel, we refer to $\Sigma_{tim} := \{(\sigma, l_\sigma, u_\sigma) : \sigma \in \Sigma\}$ as the set of *timed events* of $\mathbf{G}$.

Following [1], we give the TDES $\mathbf{G}$ an automaton structure. For this, we need some further notation. Let $\Sigma_{spe} := \{\sigma \in \Sigma : u_\sigma < \infty\}$ and $\Sigma_{rem} := \{\sigma \in \Sigma : u_\sigma = \infty\}$. We say that an event σ is prospective (remote) if $\sigma \in \Sigma_{spe}$ ($\sigma \in \Sigma_{rem}$). It is clear from the above definitions that $\Sigma = \Sigma_{spe} \dot\cup \Sigma_{rem}$ and that a remote event has no deadline attached to it. Next, we introduce the special event *tick* to denote the passage of one unit of time in our global clock, and write $\Sigma_t := \Sigma \cup \{tick\}$. Finally, we define the *timer interval*, T_σ, for an event σ via

$$T_\sigma := \begin{cases} [0, u_\sigma], & \sigma \in \Sigma_{spe} \\ [0, l_\sigma], & \sigma \in \Sigma_{rem} \end{cases},$$

where $[j, k]$ denotes the set of integers i satisfying $j \leq i \leq k$.

We are now able to formally write $\mathbf{G} := (Q, \Sigma_t, \delta, q_o, Q_m)$. The state set in this case is defined to be $Q = A \times \prod_{\sigma \in \Sigma} T_\sigma$, with the marked states given by $Q_m := \{(a, t_\sigma) \in Q \mid a \in A_m\}$. Similarly, we have that the initial state $q_0 := (a_o, \{t_{\sigma 0} : \sigma \in \Sigma\})$, where $t_{\sigma 0} := \max(T_\sigma)$. Now, suppose $q = (a, \{t_\sigma : \sigma \in \Sigma\}) \in Q$. We have that $\delta(q, \sigma)!$ if and only if

- $\sigma = tick$ and $(\forall \tau \in \Sigma_{spe})\ \xi(a, \tau)!$ implies $t_\tau > 0$; or
- $\sigma \in \Sigma_{spe}$, $\xi(a, \sigma)!$, and $0 \leq t_\sigma \leq u_\sigma - l_\sigma$; or
- $\sigma \in \Sigma_{rem}$, $\xi(a, \sigma)!$, and $t_\sigma = 0$

When one of the above conditions is satisfied, we have that $q' = \delta(q, \sigma) := (a', \{t'_\sigma : \sigma \in \Sigma\})$ where either

- $\sigma = tick$ implies $a' := a$ and $t'_\tau := \begin{cases} \max(0, t_\tau - 1), & \xi(a, \tau)! \\ t_{\tau 0}, & \text{otherwise} \end{cases}$
- $\sigma \in \Sigma$ implies $a' := \xi(a, \sigma)$ and $t'_\tau := \begin{cases} t_\tau, & \xi(a', \tau)! \text{ and } \tau \neq \sigma \\ t_{\tau 0}, & \tau = \sigma \text{ or } \neg(\xi(a', \tau)!) \end{cases}$

Using the above definitions, we note that an event $\sigma \in \Sigma_t$ is *eligible* at state $q = (a, \{t_\sigma : \sigma \in \Sigma\})$ if $\delta(q, \sigma)$ exists whereas an event $\sigma \in \Sigma$ is *enabled* at state q if $\xi(a, \sigma)$ exists. Thus, $\sigma \in \Sigma$ must be continuously enabled for at least l_σ, but no more than u_σ, *ticks* of the clock in order to occur, provided its occurrence is not preempted by the occurrence of another competing event.

A final technical condition is placed on TDES in order to ensure that the *tick* of the clock is not indefinitely preempted by activity transitions. Namely, if $\delta(q, s) = q$ for some state q and string $s = \sigma_1 \sigma_2 \cdots \sigma_n$, we must have that $\sigma_i = tick$ for some $1 \leq i \leq n$.

A control theory on TDES is established by defining the set of controllable events to be those which can be indefinitely disabled by a supervisor. Note that this necessarily implies that $\Sigma_c \subseteq \Sigma_{rem}$. The set of controllable events is given by $\Sigma_u := \Sigma - \Sigma_c$. It is also natural to define a set of *forcible* events for TDES, $\Sigma_{for} \subseteq \Sigma$, which are used to preempt the *tick* event. No relationship is postulated between Σ_{for} and each of Σ_{spec}, Σ_{rem}, and Σ_c. Note that the ability to preempt *ticks* makes it customary to declare $tick \in \Sigma_c$ for simplicity [1, 15]. However, we explicitly acknowledge that *tick* is really neither controllable nor uncontrollable. The motivation behind this decision will become apparent below.

A language $K \subseteq \Sigma_t^*$ is said to be controllable with respect to a TDES $\mathbf{G}$ if for all $s \in \overline{K}$,

$$\Sigma_K(s) \supseteq \begin{cases} \Sigma_{L(\mathbf{G})}(s) \cap (\Sigma_u \cup \{tick\}), & \text{if } \Sigma_K(s) \cap \Sigma_{for} = \emptyset \\ \Sigma_{L(\mathbf{G})}(s) \cap \Sigma_u, & \text{if } \Sigma_K(s) \cap \Sigma_{for} \neq \emptyset \end{cases}. \tag{6}$$

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

$$[\Sigma_{L(V/\mathbf{G})}(s) \cap \Sigma_{for} = \emptyset] \wedge [tick \in \Sigma_{L(\mathbf{G})}(s)] \Rightarrow [tick \in V(s)]. \tag{7}$$

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.

Note that the introduction of forcible events into the robustness paradigm complicates the robustness results obtained in the previous section. For this reason, our discussion on the robust control of TDES is divided into two cases: $\Sigma_{for} = \emptyset$ and $\Sigma_{for} \neq \emptyset$.

4.1 Special Case: $\Sigma_{for} = \emptyset$

In the absence of forcible events, we find that robustness results for TDES are quite similar to those for untimed DES. Before presenting these results, we note that under the general assumption of this subsection (6) and (7) reduce to

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

and

$$[tick \in \Sigma_{L(\mathbf{G})}(s)] \Rightarrow [tick \in V(s)],$$

respectively. Thus, with respect to controllability and supervision, the *tick* event is treated no differently than the uncontrollable events of the system. This should not have been unexpected since the absence of forcible events precludes the ability to preempt *ticks*.

We can now generalize Lemma 8. Note that it is easy to show that a supervisor which is admissible for each $\mathbf{G}_i$, $i \in \mathcal{I}$, is an admissible supervisor for $\mathbf{G}$. The lemma below states that when $\Sigma_{for} = \emptyset$, the reverse is also true.

Lemma 12 *Suppose* $\mathbf{G}$ *and* $\mathbf{G}'$ *are TDES over the alphabet* 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* $L_m(V/\mathbf{G}') = L_m(V/\mathbf{G}) \cap L_m(\mathbf{G}')$. *Moreover, if* $\Sigma_{for} = \emptyset$ *and* V *is an admissible supervisor for* $\mathbf{G}$, *then* V *is also an admissible supervisor for* $\mathbf{G}'$.

Proof: The first part of the proof proceeds exactly as the proof of Lemma 8. In order to complete the proof, we need to show that V is an admissible supervisor for $\mathbf{G}'$; namely, that $[tick \in \Sigma_{L(\mathbf{G}')}(s)] \Rightarrow [tick \in V(s)]$. However, this is obvious since $[tick \in \Sigma_{L(\mathbf{G}')}(s)] \Rightarrow [tick \in \Sigma_{L(\mathbf{G})}(s)]$ and V is an admissible supervisor for $\mathbf{G}$. $\Box$

Using this result, it is a straightforward exercise to prove the following generalization of Theorem 6.

Theorem 13 *Given TDES* $\{\mathbf{G}_i \mid i \in \mathcal{I}\}$, *let* $\mathbf{G}$ *be defined via*

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

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*

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

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*

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

where the L_i *in* (2) *are replaced by* $L_m(\mathbf{G}_i)$*. If* $K \neq \emptyset$ *is* $L_m(\mathbf{G})$*-closed, a supervisor* $V \in \mathcal{V}$ *solves the Robust Nonblocking Supervisory Control Problem if and only if* $L_m(V/\mathbf{G}) = K$*.*

4.2 Forced TDES

The introduction of forcible events means that the second conclusion of Lemma 12 does not hold for TDES in general. Namely, an admissible supervisor for $\mathbf{G}$ need not be in $\mathcal{V}$. To see this consider the following example.

Example 14 Suppose we have TDES $\mathbf{G}_1$ and $\mathbf{G}_2$ as shown in Figure 2, in which the marked states are shown using solid nodes. We then have that the languages $L_m(\mathbf{G}_1) = \overline{L(\mathbf{G}_1) = (tick)^*\alpha(tick)^*}$ and $L_m(\mathbf{G}_2) = L(\mathbf{G}_2) = \overline{(tick)^*\beta(tick)^*}$, while $L_m(\mathbf{G}) = L(\mathbf{G}) = \overline{(tick)^*\alpha(tick)^* \cup (tick)^*\beta(tick)^*}$. Further suppose that $\Sigma_{for} = \{\alpha\}$.

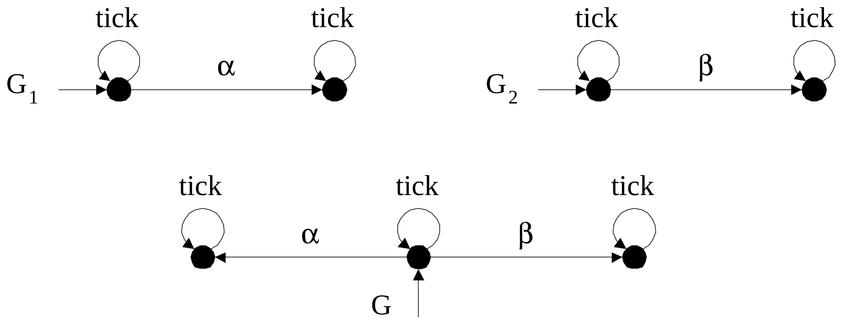

Figure 2: TDES for Example 14

Now define a supervisor V as follows:

$$V(\varepsilon) := \{\alpha, \beta\}$$
$$V(s) := \Sigma = \{\alpha, \beta, tick\}, \textit{ for every } s \in \Sigma^* - \{\varepsilon\}.$$

It is easy to see that V is an admissible supervisor for $\mathbf{G}$ since $\Sigma_{L(V/\mathbf{G})}(\varepsilon) \cap \Sigma_{for} = \{\alpha\} \neq \emptyset$ and $tick \in V(s)$ otherwise. However, V is not an admissible supervisor for $\mathbf{G}_2$ since $\Sigma_{L(V/\mathbf{G}_2)}(\varepsilon) \cap \Sigma_{for} = \emptyset$ and $tick \in \Sigma_{L(\mathbf{G}_2)}(\varepsilon)$, but $tick \notin V(\varepsilon)$. That is, unlike the composite plant $\mathbf{G}$, its component $\mathbf{G}_2$ does not have the ability to preempt the eligible *tick* event in its initial state using α. Thus, although V is an admissible supervisor for $\mathbf{G}$, we have that $V \notin \mathcal{V}$.

The problem exhibited in Example 14 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

$$\mathcal{A}_E := \{K \subseteq E \mid K \text{ is controllable with respect to } \mathbf{G}_i \text{ for all } i \in \mathcal{I}\}.$$

Clearly $\mathcal{A}_E$ is closed under arbitrary unions and so $\sup \mathcal{A}_E$ exists and is well-defined. Moreover, $\sup \mathcal{A}_E$ can be obtained by iteratively applying controllability with respect to each of the plants. This is the subject of the following lemma, the proof of which is straightforward and hence omitted.

Lemma 15 *Given a language E and TDES $\mathbf{G}_i$ for i in the finite index set $\mathcal{I} = \{0, 1, \ldots, n\}$, $\sup \mathcal{A}_E$ is well-defined and*

$$\sup \mathcal{A}_E = \lim_{k\to\infty} E_k,$$

where

$$E_{-1} := E$$

and

$$E_k := \sup \mathcal{C}_{\mathbf{G}_{[k]}}(E_{k-1})$$

for each non-negative integer k, where $[k] := k \pmod{n}$. Moreover, there exists an integer N such that $E_\infty = E_k$ for all $k \geq N$.

The above lemma suggests that we can generalize Theorem 6 for all TDES by starting with the language E as in (4) and iteratively imposing controllability with respect to $\mathbf{G}$, making the result nonconflicting with each $\mathbf{G}_i$, and then imposing controllability with respect to $\mathbf{G}_i$ for each $i \in \mathcal{I}$. Namely, to solve RNSCP for TDES in general, we need to compute $\sup \mathcal{ANC}_{\mathbf{G}}(E)$, where $\mathcal{ANC}_{\mathbf{G}}(E) := \mathcal{A}_E \cap \mathcal{N}_E \cap \mathcal{C}_{\mathbf{G}}(E)$. The next lemma shows that $\mathcal{A}_E \subseteq \mathcal{C}_{\mathbf{G}}(E)$, and so one step in the above three step iteration is redundant.

Lemma 16 *If $K \subseteq \Sigma^*$ is controllable with respect to $\mathbf{G}_i$ for each $i \in \mathcal{I}$, then K is controllable with respect to $\mathbf{G}$.*

Proof: By definition of controllability, we have that

$$\Sigma_K(s) \supseteq \begin{cases} \Sigma_{L(\mathbf{G}_i)}(s) \cap ((\Sigma_u)_i \cup \{tick\}), & \text{if } \Sigma_K(s) \cap (\Sigma_{for})_i = \emptyset \\ \Sigma_{L(\mathbf{G}_i)}(s) \cap (\Sigma_u)_i, & \text{if } \Sigma_K(s) \cap (\Sigma_{for})_i \neq \emptyset \end{cases}$$

for each $i \in \mathcal{I}$. Since it is assumed that all plants agree on controllability of events, the above is equivalent to

$$\Sigma_K(s) \supseteq \begin{cases} \Sigma_{L(\mathbf{G}_i)}(s) \cap (\Sigma_u \cup \{tick\}), & \text{if } \Sigma_K(s) \cap (\Sigma_{for})_i = \emptyset \\ \Sigma_{L(\mathbf{G}_i)}(s) \cap \Sigma_u, & \text{if } \Sigma_K(s) \cap (\Sigma_{for})_i \neq \emptyset \end{cases}. \tag{8}$$

Now, suppose $\Sigma_K(s) \cap \Sigma_{for} = \emptyset$. Since all plants also agree on the forcibility of events, this means that $\Sigma_K(s) \cap (\Sigma_{for})_i = \emptyset$ for each $i \in \mathcal{I}$. We need to show that $\Sigma_{L(\mathbf{G})} \cap (\Sigma_u \cup \{tick\}) \subseteq \Sigma_K(s)$. However, this is obvious from (8) since $\Sigma_{L(\mathbf{G})}(s) = \bigcup_{i \in \mathcal{I}} \Sigma_{L(\mathbf{G}_i)}(s)$. Similarly, $\Sigma_{L(\mathbf{G})}(s) \cap \Sigma_u \subseteq \Sigma_K(s)$ when $\Sigma_K(s) \cap \Sigma_{for} \neq \emptyset$. □

The final lemma we present before giving our main robustness result for TDES shows how to compute the supremal element of $\mathcal{AN}_E := \mathcal{A}_E \cap \mathcal{N}_E$. It is a routine generalization of Lemmas 4 and 5. The proof is omitted.

Lemma 17 *Given languages E and L_i as in Lemma 3 and the DES $\mathbf{G}_i$ defined over Σ for $i \in \mathcal{I}$, let*

$$\mathcal{AN}_E := \mathcal{A}_E \cap \mathcal{N}_E. \tag{9}$$

Then, $\sup \mathcal{AN}_E$ 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

$$\Omega(M) := \sup \mathcal{A}_M(\sup \mathcal{N}_M).$$

Moreover,

$$\sup \mathcal{AN}_E = \lim_{k \to \infty} E_k,$$

where

$$\begin{aligned} E_0 &:= E, \\ E_{2k-1} &:= \sup \mathcal{N}_{E_{2k-2}} \\ \text{and } E_{2k} &:= \sup \mathcal{A}_{E_{2k-1}} \end{aligned}$$

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

We end our discussion on robust TDES by stating the generalization of Theorem 6 that deals with general TDES. Note that it generalizes Takai's results in [15] much in the same way that Theorem 6 generalizes the results in [14].

Theorem 18 *Given TDES $\{\mathbf{G}_i \mid i \in \mathcal{I}\}$, let $\mathbf{G}$ be defined via*

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

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

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

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

$$\Omega(Z) := \sup \mathcal{AN}_Z,$$

where the L_i in (2) are replaced by $L_m(\mathbf{G}_i)$. If $K \neq \emptyset$ is $L_m(\mathbf{G})$-closed, a supervisor $V \in \mathcal{V}$ solves the Robust Nonblocking Supervisory Control Problem if and only if $L_m(V/\mathbf{G}) = K$.

5 Examples

We now present three examples which help further understand the theory of robustness developed in the previous sections. The first example is one mentioned in the introduction of the paper. It illustrates the flexibility of the robustness paradigm and that in this setting robustness need not always yield conservative results. The second example shows a manufacturing line that utilizes one of four configurations for testing its finished product. In this example, we find that control is necessarily conservative, at least initially. However, under certain special circumstances, the controller can become optimal as described in our earlier discussion on adaptive supervision. Finally, our third example demonstrates the use of robustness in the timed DES framework. Specifically, robustness with respect to variations in the time bounds of events is explored. In all the diagrams that follow, marked states are those denoted by solid nodes.

Example 19 Consider a short assembly line where pieces are worked on by one of n_1 identical input machines and then by one of n_2 identical output machines. However, breakdown or resource allocation may force some of the machines at either end to be temporarily unavailable. This means that at any given time there are $2^{n_1+n_2}$ possible configurations for the assembly line. The pieces are placed in a k-slot buffer between being worked on by the input and output machines. The specification on the plant in this case is to prevent both overflow and underflow of the buffer.

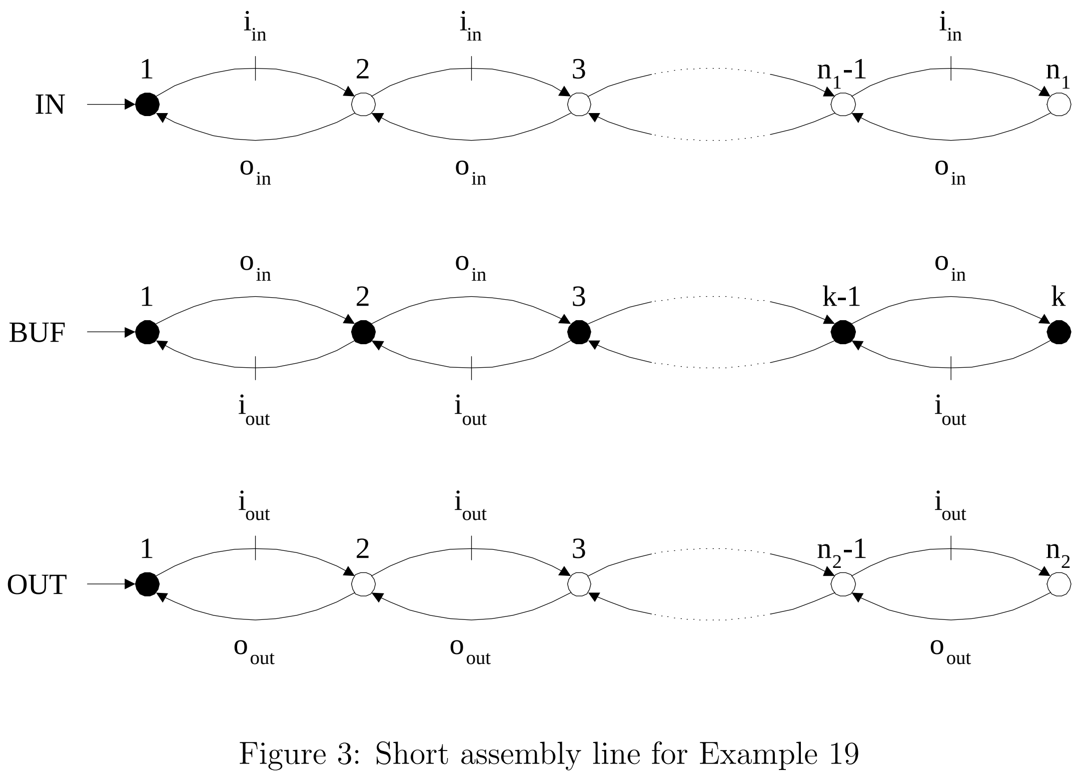

Figure 3: Short assembly line for Example 19

The automata representing the input and output ends of the line, with n_1 and n_2 machines available, as well as the specification on the buffer are shown in Figure 3. We assume for this example that the set of controllable events is $\Sigma_c = \{i_{in}, i_{out}\}$. In this case, the controller obtained is optimal in that the closed-loop behaviour for the robust controller is precisely the closed-loop behaviour obtained for each individual plant configuration.

Example 20 Consider a manufacturing line where 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 fails 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.

The plant configurations are shown in Figure 4. Automata representing each of the components are shown in Figure 5. 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 $\{in_1, in_2, in_r, in_4, in_{TU}\}$ and the uncontrollable events are $\{out_1, out_2, out_4, pass, fail\}$. Once again, the specification is to prevent underflow and overflow of the buffers. These specifications are collectively shown in Figure 6.

In this case, we find that the supervisor is conservative. 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 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 3.

Example 21 In this last example, we explore the robustness of TDES with respect to variations in the time bounds of events in a TDES. Consider the TDES $\mathbf{G}$ whose activity transition graph (ATG) is that shown in Figure 7.

Here, we assume that the time bounds for event α are $(5, 5)$ while the time bounds for event β are $(2, 5)$. The resulting timed transition graph (TTG) explicitly showing *tick* transitions is shown in Figure 8. Also, for this example, we have that $\Sigma_{for} = \beta$.

The specification for this example is shown in Figure 9. Namely, we seek to prevent the occurrence of the α event unless it is immediately preceded by a β event. This is easily accomplished by forcing β to occur prior to the eligibility of event α. Moreover, this is the only means of achieving our goal since α cannot be disabled. A TDES implementing the optimal supervisory controller for this example is shown in Figure 10, where the occurrence of β is forced in the state labelled q.

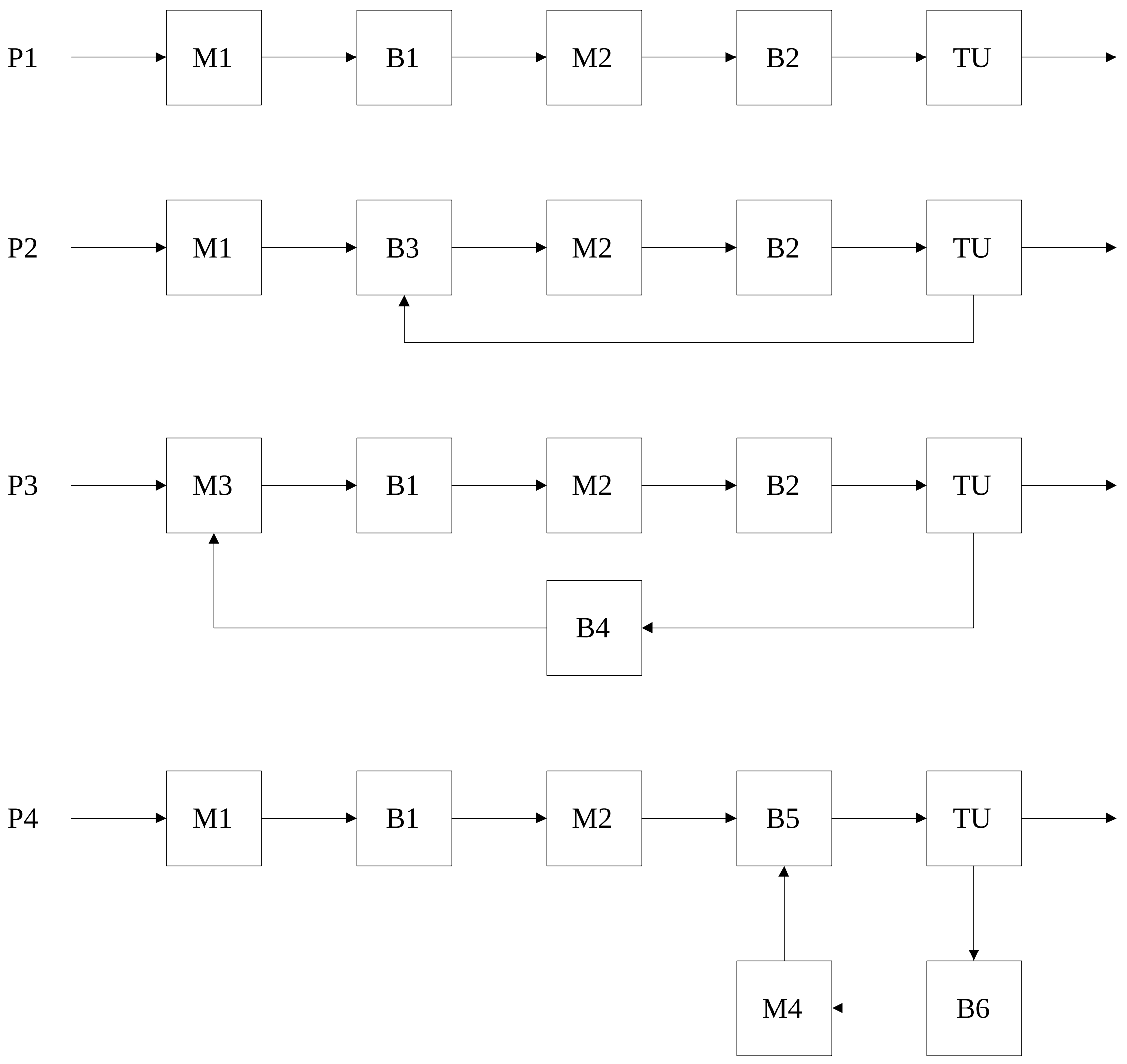

Figure 4: Assembly line configurations for Example 20

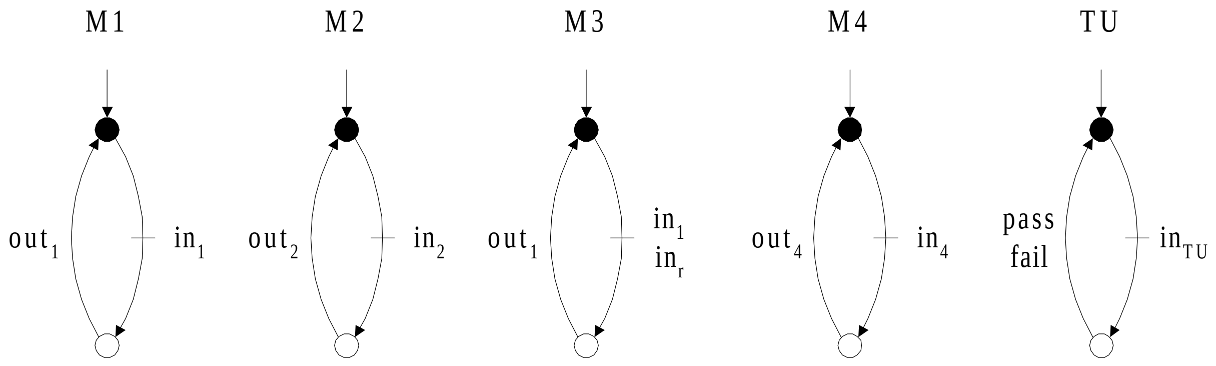

Figure 5: Component machines for Example 20

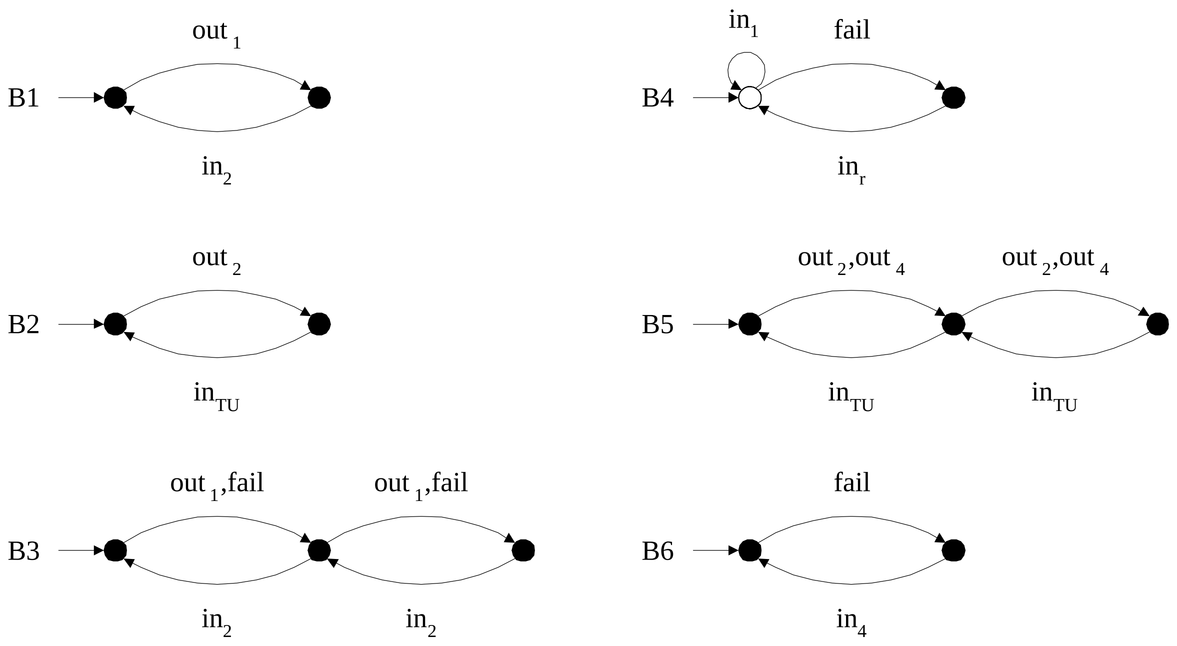

Figure 6: Specifications for Example 20

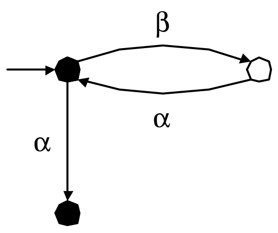

Figure 7: ATG for Example 21

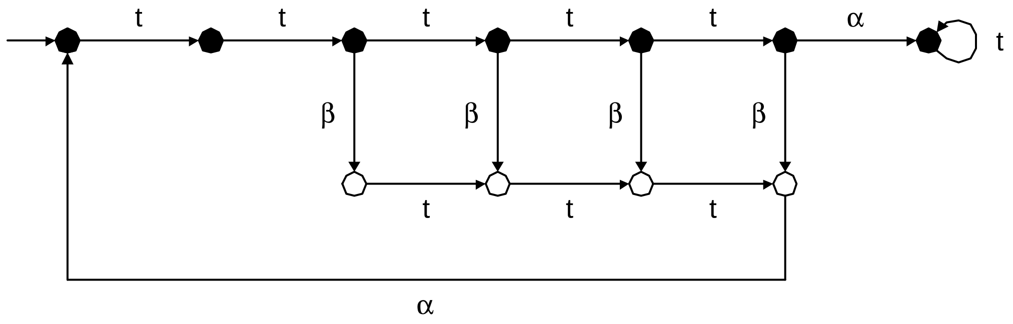

Figure 8: TTG for Example 21

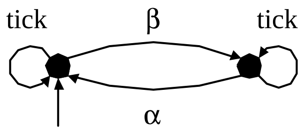

Figure 9: Specification for Example 21

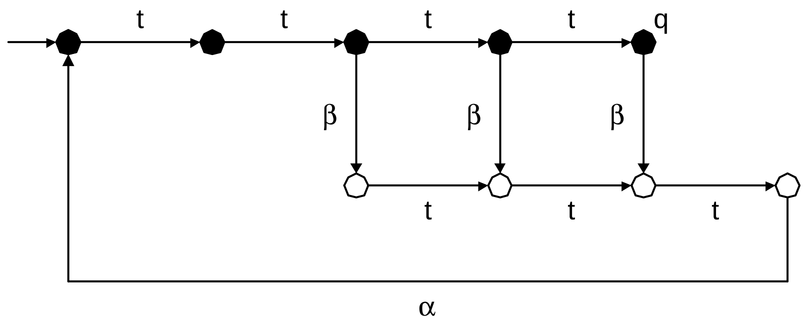

Figure 10: Optimal Supervisor for Example 21

A method for computing suboptimal supervisory controllers is described in [6]. Such a supervisor can be computed by first scaling the time bounds of all events by a common scaling factor n. Time is scaled so that there is a *decrease* in the rate of *tick* events per unit of time. The scaled time bounds may not be integer and so the lower time bounds are rounded down and the upper time bounds are rounded up where appropriate. The only restriction on the choice of scaling factor is that it must be an integer multiple of l_σ for each $\sigma \in \Sigma_{for}$. This means that the only allowable scaling factor in our case is 2 and that the scaled time bounds for α and β are $(2, 3)$ and $(1, 3)$, respectively. Control is then computed using the scaled TDES. Once obtained, this controller is scaled back to its original granularity of *ticks*. The suboptimal controller obtained for our example is shown in Figure 11. Notice that the controller is conservative in that it forces the occurrence of β unnecessarily early in state q_{o2}.

This suboptimal controller is robust in that it could be applied to TDES having time bounds $(4, 5)$, $(4, 6)$, $(5, 5)$, or $(5, 6)$ for α and time bounds $(2, 5)$ or $(2, 6)$ for β since each of these possibilities yields time bounds $(2, 3)$ and $(1, 3)$ when scaled by 2. Note that in the absence of the divisibility condition imposed by l_β, these scaled time bounds would have also been obtained if the original time bounds on β were $(3, 5)$ or $(3, 6)$.

The time bounds given in the last paragraph can be used to define a family of 16 unique TDES. We now consider the problem of controlling this family of TDES subject to the specification given in Figure 9. The supervisor obtained using the methods of Section 4 is shown in Figure 12.

Notice that this supervisor is less conservative than the one shown in Figure 11 since event β is no longer prematurely forced to occur in state q_{o2}. There are two other interesting points to note before concluding this example. First, unlike the suboptimal supervisor of [6],

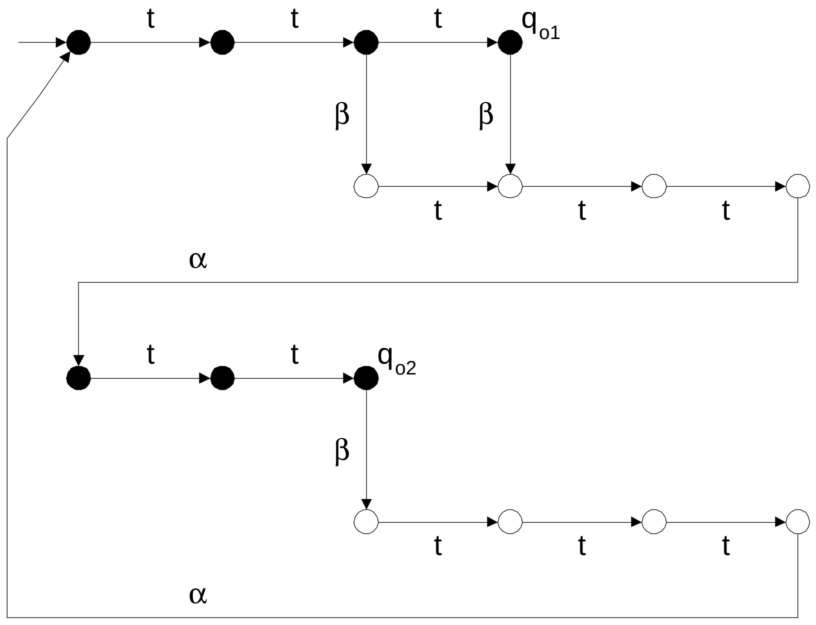

Figure 11: Suboptimal Controller of [6] for Example 21

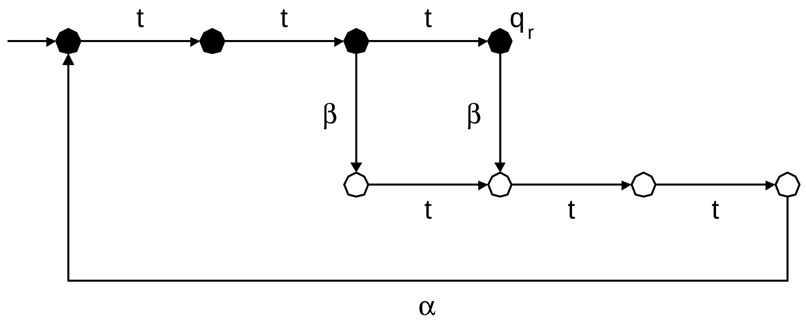

Figure 12: Robust Controller for Example 21

the robust formulation is not constrained by the divisibility condition on forcible events. Namely, we are completely free to choose any scaling factor we want, although large scaling factors (5 or more in this example) will yield trivial results. A second observation to make in this case is that the robust supervisor is not adaptive since we have that $L_m(\mathbf{G}_i) \subseteq L_m(\mathbf{G}')$ and $L(\mathbf{G}_i) \subseteq L(\mathbf{G}')$ for $i = 1, \ldots, 16$, where $\mathbf{G}'$ is the TDES with $(l_\alpha, u_\alpha) = (4, 6)$ and $(l_\beta, u_\beta) = (2, 6)$. This is despite the fact that after finite time, we are easily able to rule out many of the candidate plant models.

6 Conclusions

In this paper, we have extended the results of [8], [14], and [15]. We introduce a framework for the modeling of robust control of discrete-event systems 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 behaviour of a system require no control intervention.

References

[1] B.A. Brandin and W.M. Wonham. The supervisory control of timed DES. *IEEE Transactions on Automatic Control*, **39**(2), pages 329–342, 1994.

[2] E. Chen and S. Lafortune. On nonconflicting languages that arise in supervisory control of discrete event systems. *Systems & Control Letters*, **17**(2), pages 105–113, 1991.

[3] K.-H. Cho and J.-T. Lim. Stability and robustness of discrete event dynamic systems. *International Journal of Systems Science*, **28**(7), pages 691–703, 1997.

[4] J.E.R. Cury and B.H. Krogh. Design of robust supervisors for discrete event systems with infinite behaviors. In *Proceedings of the IEEE Conference on Decision and Control*, pages 2219–2224, 1996.

[5] J.E.R. Cury and B.H. Krogh. Robustness of supervisors for discrete-event systems. *IEEE Transactions on Automatic Control*, **44**(2), pages 376–379, 1999.

[6] P. Gohari and W.M. Wonham. Reduced supervisors for timed discrete-event systems. In R. Boel and G. Stremersch, editors, *Discrete Event Systems: Analysis and Control*, SECS 569, pages 119–130, 2000.

[7] R. Kumar and V.K. Garg. *Modeling and Control of Logical Discrete Event Systems.* Kluwer Academic Publishers, Boston, 1995.

[8] F. Lin. Robust and adaptive supervisory control of discrete event systems. *IEEE Transactions on Automatic Control*, **38**(12), pages 1848–1852, 1993.

[9] C.M. Özveren and A.S. Willsky. Output stabilizability of discrete-event dynamic systems. *IEEE Transactions on Automatic Control*, **36**(8), pages 925–935, 1991.

[10] C.M. Özveren and A.S. Willsky. Stability and stabilizability of discrete event dynamic systems. *Journal of the Association for Computing Machinery*, **38**(3), pages 730–752, 1991.

[11] S.-J. Park and J.-T. Lim. Robust nonblocking supervisor for discrete event systems with model uncertainty under partial observation. *IEEE Transactions on Automatic Control*, **45**(12), pages 2393–2396, 2000.

[12] S.-J. Park and J.-T. Lim. Robust and nonblocking supervisory control of nondeterministic discrete event systems using trajectory models. *IEEE Transactions on Automatic Control*, **47**(4), pages 655–658, 2002.

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

[14] S. Takai. Maximally permissive robust supervisors for a class of specification languages. In *Proceedings of the IFAC Conference on System Structure and Control*, volume 2, pages 429–434, 1998.

[15] S. Takai. Robust supervisory control of a class of timed discrete event systems under partial observation. *Systems & Control Letters*, **39**(4), pages 267–273, 2000.

[16] S. Takai. Synthesis of maximally permissive and robust supervisors for prefix-closed language specifications. *IEEE Transactions on Automatic Control*, **47**(1), pages 132–136, 2002.