Supervision Localization of Timed Discrete-Event Systems

Renyuan Zhang\textsuperscript{a}, Kai Cai\textsuperscript{b}, Yongmei Gan\textsuperscript{a}, Zhaoan Wang\textsuperscript{a}, W.M. Wonham\textsuperscript{b}

\textsuperscript{a}School of Electrical Engineering, Xi'an Jiaotong University, Xi'an, Shaanxi 710049, China
\textsuperscript{b}Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON M5S 3G4, Canada

Abstract

We study supervisor localization for real-time discrete-event systems (DES) in the Brandin-Wonham framework of timed supervisory control. We view a real-time DES as comprised of asynchronous agents which are coupled through imposed logical and temporal specifications; the essence of supervisor localization is the decomposition of monolithic (global) control action into local control strategies for these individual agents. This study extends our previous work on supervisor localization for untimed DES, in that monolithic timed control action typically includes not only disabling action as in the untimed case, but also "clock preemtting" action which enforces prescribed temporal behavior. The latter action is executed by a class of special events, called "forcible" events; accordingly, we localize monolithic preemptive action with respect to these events. We demonstrate the new features of timed supervisor localization with a manufacturing cell case study, and discuss a distributed control implementation.

Key words: Supervisory control; Supervisor localization; Timed discrete-event systems.

1 Introduction

Recently we developed a top-down approach, called supervisor localization\cite{3, 4} to the distributed control of untimed discrete-event systems (DES) in the Ramadge-Wonham (RW) supervisory control framework\cite{10, 14}. We view the plant to be controlled as comprised of independent asynchronous agents which are coupled implicitly through logical control specifications. To make the agents smart and semi-autonomous, our localization algorithm allocates external supervisory control action to individual agents as their internal control strategies, while preserving the optimality (maximal permissiveness) and nonblocking properties of the overall monolithic (global) controlled behavior. Under the localization scheme, each agent controls only its own events, although it may very well need to observe events originating in other (typically neighboring) agents.

A variety of real-time DES models and approaches are available. Notable works include Brave and Heymann's "clock automata"\cite{2}, Ostroff's "timed transition models"\cite{9}, Brandin and Wonham's timed DES (TDES)\cite{1}, Cassandras's "timed state automata"\cite{5}, Wong-Toi and Hoffman's model based on "timed automata"\cite{12}, and Cofer and Garg's model based on "timed Petri nets"\cite{6}. We adopt Brandin and Wonham's TDES (or BW model) as the framework for developing a timed supervisor localization theory for two reasons. First, the BW model is a direct extension from the RW framework (where our untimed localization theory is based), retaining the central concepts of controllability, and maximally permiss-
sive nonblocking supervision. This feature facilitates developing a timed counterpart of supervisor localization. Second, the BW model captures a variety of timing issues in a useful range of real-time discrete-event control problems [1], [14, Chapter 9]. While it may be possible to develop supervisor localization in an alternative framework, as a preliminary step into real-time supervisor localization we choose the BW model for its close relation with previous work.

The principal contribution of this paper is the development of a timed supervisor localization theory in the BW TDES framework, which extends the untimed counterpart in [3, 4]. In this timed localization, a novel feature is “event forcing” as means of control, in addition to the usual “event disabling”. Specifically, “forcible” events are present in the BW model as events that can be relied on, when subject to some temporal specification, to “preempt the tick of the clock”, as explained further in Section 2. Correspondingly, in localizing the monolithic supervisor’s control action, we localize not only its disabling action as in the untimed case, but also its preemptive action with respect to individual forcible events. Central to the latter are several new ideas: “local preemtior”, “preemption consistency relation”, and “preemption cover”. We will prove that localized disabling and preemptive behaviors collectively achieve the same global optimal and nonblocking controlled behavior as the monolithic supervisor does. The proof relies on the new preemption concepts and also controllability for TDES. Moreover, the derived local controllers typically have much smaller state size than the monolithic supervisor, and hence their disabling and preemptive logics are often more transparent. We demonstrate this empirical result by a case study of a manufacturing cell taken from [1].

The paper is organized as follows. Section 2 provides a review of the BW TDES framework. Section 3 formulates the timed supervisor localization problem, and Section 4 presents a constructive solution procedure. Section 5 studies a manufacturing cell example; and finally, Section 6 draws conclusions.

2 Preliminaries on Timed Discrete-Event Systems

This section reviews the TDES model proposed by Brandin and Wonham [1], [14, Chapter 9]. First consider the untimed DES model

$$G_{act} = (A, \Sigma_{act}, \delta_{act}, a_0, A_m).$$  

(1)

Here $A$ is the finite set of $\textit{activities}$, $\Sigma_{act}$ is the finite set of $\textit{events}$, $\delta_{act} : A \times \Sigma_{act} \rightarrow A$ is the (partial) $\textit{activity transition function}$, $a_0 \in A$ is the $\textit{initial activity}$, and $A_m \subseteq A$ is the set of $\textit{marker activities}$. Let $\mathbb{N}$ denote the set of natural numbers $\{0, 1, 2, \ldots\}$. We introduce $\textit{time}$ into $G_{act}$ by assigning to each event $\sigma \in \Sigma_{act}$ a $\textit{lower time bound}$ $l_\sigma \in \mathbb{N}$ and an $\textit{upper time bound}$ $u_\sigma \in \mathbb{N} \cup \{\infty\}$, such that $l_\sigma \leq u_\sigma$; typically, $l_\sigma$ represents a delay in communication or in control enforcement, while $u_\sigma$ is often a hard deadline imposed by legal specification or physical necessity. With these assigned time bounds, the event set $\Sigma_{act}$ is partitioned into two subsets: $\Sigma_{act} = \Sigma_{spe} \cup \Sigma_{rem}$ ($\cup$ denotes $\textit{disjoint union}$) with $\Sigma_{spe} := \{ \sigma \in \Sigma_{act} | u_\sigma \in \mathbb{N} \}$ and $\Sigma_{rem} := \{ \sigma \in \Sigma_{act} | u_\sigma = \infty \}$; here “spe” denotes “prospective”, i.e. $\sigma$ will occur within some prospective time (with a finite upper bound), while “rem” denotes “remote”, i.e. $\sigma$ will occur at some $\textit{indefinite}$ time (with no upper bound), or possibly will never occur at all.

A distinguished event, written $\textit{tick}$, is introduced which represents “tick of the global clock”. Attach to each event $\sigma \in \Sigma_{act}$ a (countdown) $\textit{timer}$ $t_\sigma \in \mathbb{N}$, whose default value $t_{\sigma 0}$ is set to be

$$t_{\sigma 0} := \begin{cases} u_\sigma & \text{if } \sigma \in \Sigma_{spe}, \\ l_\sigma & \text{if } \sigma \in \Sigma_{rem}. \end{cases}$$

(2)

When timer $t_\sigma > 0$, it decreases by 1 (counting down) if event $\textit{tick}$ occurs; and when $t_\sigma = 0$, event $\sigma$ must occur (resp. may occur) if $\sigma \in \Sigma_{spe}$ (resp. if $\sigma \in \Sigma_{rem}$). Note that while $\textit{tick}$ is a global event, each timer $t_\sigma$ is local (with respect to the event $\sigma$). Also define the $\textit{timer interval}$ $T_\sigma$ by

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

(3)

Thus $t_\sigma \in T_\sigma$.

Based on (1)-(3), the TDES model $G$ is given by

$$G := (Q, \Sigma, \delta, q_0, Q_m),$$

(4)

where $Q := A \times \prod \{T_\sigma | \sigma \in \Sigma_{act}\}$ ($\prod$ denotes $\textit{Cartesian product}$) is the finite set of states, a state $q \in Q$ being of the form $q = (a, \{t_\sigma | \sigma \in \Sigma_{act}\})$ (i.e. a $(1 + |\Sigma_{act}|)$-tuple); $\Sigma := \Sigma_{act} \cup \{\textit{tick}\}$ is the finite set of events; $\delta : Q \times \Sigma \rightarrow Q$ is the (partial) $\textit{state transition function}$, $q_0 = (a_0, \{t_{\sigma 0} | \sigma \in \Sigma_{act}\})$ is the $\textit{initial state}$; and $Q_m \subseteq A \times \prod \{T_\sigma | \sigma \in \Sigma_{act}\}$ is the set of $\textit{marker states}$. Starting from $q_0$, TDES $G$ executes state transitions in accordance with its transition function $\delta$. Let $q = (a, \{t_\alpha | \alpha \in \Sigma_{act}\}) \in Q$ and $\sigma \in \Sigma_{act}$; $\delta$ is defined at $(q, \sigma)$, written $\delta(q, \sigma)$, if $\delta_{act}$ of $G_{act}$ is defined at $(a, \sigma)$ (i.e. $\delta_{act}(a, \sigma)$) and timer $t_\sigma$ satisfies (i) $0 \leq t_\sigma \leq u_\sigma - l_\sigma$ when $\sigma \in \Sigma_{spe}$, and (ii) $t_\sigma = 0$ when $\sigma \in \Sigma_{rem}$. The new state $q'$ is $\delta(q, \sigma)$ is given by $q' = (\delta_{act}(a, \sigma), t'_\sigma | \in \alpha \in \Sigma_{act})$.

1 An upper bound for the state size $|Q|$ is $|A| \times \prod_{\sigma \in \Sigma_{act}} t_{\sigma 0}$ (here $\prod$ denotes $\textit{scalar multiplication}$), which in practice can be much larger than its untimed counterpart $|A|$. 


F is set to be its default value $t_{a0}$ as in (2); for other timers $t_{a}$, $a \neq \sigma$, the reader is referred to detailed updating rules given in [1, 14]. On the other hand, $\delta(q, \text{tick})$ if no timer of a prospective event is zero, and $q' = \delta(q, \text{tick}) = (a, \{t'_{a} | a \in \Sigma_{act}\})$, i.e. there is no change in the activity component $a$ of $q$, while the rules for updating timers are again referred to [1, 14].

Let $\Sigma^*$ be the set of all finite strings of elements in $\Sigma = \Sigma_{act} \cup \{\text{tick}\}$, including the empty string $\epsilon$. For $\Sigma' \subseteq \Sigma$, the natural projection $P : \Sigma^* \rightarrow \Sigma^*$ is defined according to

$$P(\epsilon) = \epsilon, \quad \epsilon \text{ is the empty string;}$$

$$P(\sigma) = \begin{cases} \epsilon, & \text{if } \sigma \notin \Sigma' \\ \sigma, & \text{if } \sigma \in \Sigma' \end{cases} \quad (5)$$

$$P(s\sigma) = P(s)P(\sigma), \quad s \in \Sigma^*, \sigma \in \Sigma.$$  

In the usual way, $P$ is extended to $P : \text{Pwr}(\Sigma^*) \rightarrow \text{Pwr}(\Sigma^*)$, where $\text{Pwr}(\cdot)$ denotes powerset. Write $P^{-1} : \text{Pwr}(\Sigma^*) \rightarrow \text{Pwr}(\Sigma^*)$ for the inverse-image function of $P$.

We introduce the languages generated by TDES $G$ in (4). The transition function $\delta$ is extended to $\tilde{\delta} : Q \times \Sigma^* \rightarrow Q$ in the usual way. The closed behavior of $G$ is the language

$$L(G) := \{s \in \Sigma^*|\tilde{\delta}(q_0, s)\} \quad (6)$$

and the marked behavior is

$$L_m(G) := \{s \in L(G)|\tilde{\delta}(q_0, s) \in Q_m\} \subseteq L(G). \quad (7)$$

We say that $G$ is nonblocking if the prefix closure ([14])

$L_m(G) = L(G).$

To use TDES $G$ in (4) for supervisory control, it is necessary to specify certain transitions that can be controlled by an external supervisor. First, as in the untimed theory [14], we need a subset of events that may be disabled. Since disabling an event usually requires preventing that event indefinitely from occurring, only remote events belong to this category. Thus let a new subset $\Sigma_{hib} \subseteq \Sigma_{rem}$ denote the prohibitive events; the supervisor is allowed to disable any prohibitive event. Next, and specific to TDES, we bring in another category of events which can preempt event tick. Note that tick may not be disabled, as much as no control technology can stop the global clock indefinitely. On this basis let a new subset $\Sigma_{for} \subseteq \Sigma_{act}$ denote the forcible events; a forcible event is one that preempts event tick; if, at a state $q$ of $G$, tick is defined and so are one or more forcible events, then tick can be effectively erased from the current list of defined events (contrast with indefinite erasure). There is no particular postulate $a$ priori between $\Sigma_{for}$ and any of $\Sigma_{hib}$, $\Sigma_{rem}$ or $\Sigma_{spe}$; in particular, a remote event may be both forcible and prohibitive. It is now convenient to define the controllable event set

$$\Sigma_c := \Sigma_{hib} \cup \{\text{tick}\}. \quad (8)$$

Here designating both $\Sigma_{hib}$ and tick controllable is to simplify terminology. We emphasize that events in $\Sigma_{hib}$ can be disabled indefinitely, while tick may be preempted only by events in $\Sigma_{for}$. The uncontrollable event set $\Sigma_u$ is

$$\Sigma_u := \Sigma - \Sigma_c = \Sigma_{spe} \cup (\Sigma_{rem} - \Sigma_{hib}). \quad (9)$$

We introduce the notion of controllability as follows. For a string $s \in L(G)$, define

$$\text{Elig}_G(s) := \{\sigma \in \Sigma|s\sigma \in L(G)\} \quad (10)$$

to be the subset of events ‘eligible’ to occur (i.e. defined) at the state $q = \delta(q_0, s)$. Consider an arbitrary language $F \subseteq L(G)$ and a string $s \in F$; similarly define the eligible event subset

$$\text{Elig}_F(s) := \{\sigma \in \Sigma|s\sigma \in F\}; \quad (11)$$

We say $F$ is controllable with respect to $G$ in (4) if, for all $s \in F$,

$$\text{Elig}_F(s) \supseteq \begin{cases} \text{Elig}_G(s) \cap (\Sigma_u \cup \{\text{tick}\}) \\ \text{Elig}_G(s) \cap \Sigma_u \\ \text{Elig}_G(s) \cap \Sigma_{for} \end{cases}$$

if $\text{Elig}_F(s) \cap \Sigma_{for} = \emptyset$, \quad (12)

Thus $F$ controllable means that an event $\sigma$ is eligible to occur in $F$ if (i) $\sigma$ is currently eligible in $G$, and (ii) either $\sigma$ is uncontrollable or $\sigma = \text{tick}$ when there is no forcible event currently eligible in $F$. Recall that in the untimed supervisory control theory [10, 14], $F$ controllable means that the occurrence of an uncontrollable event in $G$ will not cause a string $s \in F$ to exit from $F$; the difference in TDES is that, the special event tick (formally controllable) can be preempted only by a forcible event when the forcible event is eligible to occur.

Whether or not $F$ is controllable, we denote by $C(F)$ the set of all controllable sublanguages of $F$. Then $C(F)$ is nonempty, closed under arbitrary set unions, and thus contains a unique supremal element denoted by $\text{sup}C(F)$ [1, 14]. Now consider a specification language $E \subseteq \Sigma^*$ imposed on the timed behavior of $G$; $E$ may represent
First, let $\alpha \in \Sigma_{for}$ be an arbitrary forcible event. We say that $\text{LOC}^P_\alpha = (Y_\alpha, \Sigma_\alpha, \zeta_\alpha, y_0, Y_m, \alpha) \subseteq \Sigma$, is a local preemper (for $\alpha$) if $\alpha$ is defined at every state of $\text{LOC}^P_\alpha$ where event $\text{tick}$ is preempted. Let $P_\alpha : \Sigma^* \rightarrow \Sigma^*$ be the natural projection as in (5). Then in terms of language, the above condition means that for every $s \in \Sigma^*$ there holds

\[
s.\text{tick} \in L(G) \& s \in P_\alpha^{-1}L(\text{LOC}^P_\alpha) \& \\
\text{s.tick} \notin P_\alpha^{-1}L(\text{LOC}^P_\alpha) \Rightarrow s.\alpha \in L(G) \cap P_\alpha^{-1}L(\text{LOC}^P_\alpha).
\]

Notation $s.\text{tick}$ means that event $\text{tick}$ occurs after string $s$, and will be used henceforth. The left side of the above implication means that event $\text{tick}$ is preempted in $\text{LOC}^P_\alpha$ after string $s$ (after $s$ event $\text{tick}$ is defined in $L(G)$ but not in $\text{LOC}^P_\alpha$), and the right side says that forcible event $\alpha$ is defined in $\text{LOC}^P_\alpha$ (and in $L(G)$) after $s$. That is, forcible event $\alpha$ acts to preempt $\text{tick}$. The event set $\Sigma_\alpha$ of $\text{LOC}^P_\alpha$ in general satisfies $\{\alpha, \text{tick}\} \subseteq \Sigma_\alpha \subseteq \Sigma$; in typical cases, however, both subset containments are strict, as will be illustrated in Section 5. Also, for simplicity we assume the lower and upper time bounds of events in $\Sigma_\alpha$ coincide with the bounds on the corresponding events in $\Sigma$ (this is, in fact, guaranteed by the localization procedure presented below in Section 4). It is worth emphasizing that $\Sigma_\alpha$ (precisely defined below) is not fixed a priori, but will be systematically determined, as part of our localization result, to ensure correct preemptive action.

Next, let $\beta \in \Sigma_{hib}$ be an arbitrary prohibitible event. We say that $\text{LOC}^C_\beta = (Y_\beta, \Sigma_\beta, \zeta_\beta, Y_m, \beta)$, $\Sigma_\beta \subseteq \Sigma$, is a local controller (for $\beta$) if $\text{LOC}^C_\beta$ can disable only event $\beta$. Let $P_\beta : \Sigma^* \rightarrow \Sigma^*$ be the natural projection as in (5). Then in terms of language, the above condition means that for all $s \in \Sigma^*$ and $\sigma \in \Sigma$, there holds (cf. [3])

\[
s.\sigma \in L(G) \& s \in P_\alpha^{-1}L(\text{LOC}^C_\beta) \& \\
\text{s.tick} \notin P_\alpha^{-1}L(\text{LOC}^C_\beta) \Rightarrow s.\sigma \in L(G) \cap P_\alpha^{-1}L(\text{LOC}^C_\beta).
\]

The event set $\Sigma_\beta$ of $\text{LOC}^C_\beta$ in general satisfies $\{\beta\} \subseteq \Sigma_\beta \subseteq \Sigma$. Like $\Sigma_\alpha$ above, $\Sigma_\beta$ will be generated as part of our localization result to guarantee correct disabling action; again, the events in $\Sigma_\beta$ are assumed to have the same lower and upper time bounds as the corresponding events in $\Sigma$.

Now we formulate the Supervisor Localization Problem of TDES: Construct a set of local preemptors $\{\text{LOC}^C_\alpha | \alpha \in \Sigma_{for}\}$

Supervisor localization is an extension from untimed DES to TDES. As will be seen below, the treatment of prohibitible events is the timed counterpart of the treatment of controllable events in [3]; on the other hand, localization of forcible events' preemptive action is specific to TDES, and we introduce below the new concept “local preemper”. Further, we will discuss applying supervisor localization to the distributed control of multi-agent TDES.

---

3 SUP need not be a (strict) TDES as defined in (4). It can be any automaton whose event set contains tick; we refer to such automata as generalized TDES.

4 LOC^C_\alpha is a generalized TDES; we further explain this below in Section 4.

5 Event set $\Sigma_\beta$ need not contain event tick, since LOC^C_\beta’s disabling action may be purely logical and irrelevant to time.
\[ \Sigma_{for} \) and a set of local controllers \( \{ \text{LOC}_\beta^\alpha \mid \beta \in \Sigma_{hib} \} \), with

\[
\begin{align*}
L(\text{LOC}) &:= \left( \bigcap_{\alpha \in \Sigma_{for}} P^{-1}_\alpha L(\text{LOC}_\alpha^\beta) \right) \\
& \quad \bigcap \left( \bigcap_{\beta \in \Sigma_{hib}} P^{-1}_\beta L(\text{LOC}_\beta^\alpha) \right)
\end{align*}
\]

such that \( \text{LOC} \) is control equivalent to \( \text{SUP} \) (with respect to \( G \)) in the following sense:

\[
\begin{align*}
L(G) \cap L(\text{LOC}) &= L(\text{SUP}), \\
L_m(G) \cap L_m(\text{LOC}) &= L_m(\text{SUP}).
\end{align*}
\]

For the sake of easy implementation and comprehensibility, it would be desired in practice that the state sizes of local preemptors/controllers be very much less than that of their parent monolithic supervisor. Inasmuch as this property is neither precise to state nor always achievable, it is omitted from the above formal problem statement; in applications, nevertheless, it should be kept in mind.

Using a set of local preemptors and local controllers that is control equivalent to \( \text{SUP} \), we can build an optimal and nonblocking distributed control architecture for a multi-agent TDES plant. Let the plant \( G \) with event set \( \Sigma \) be composed of \( n \) component TDES (or agents) \( G_k \) with \( \Sigma_k \) for each agent \( k \in [1, n] \). According to (4), \( \Sigma_k = \Sigma_{act,k} \cup \{ \text{tick} \} \) (event \( \text{tick} \) is shared by all agents); thus \( \Sigma = \bigcup_{k=1}^n \Sigma_{act,k} \cup \{ \text{tick} \} \). In addition to \( \text{tick} \), we also allow the \( \Sigma_{act,k} \) to share events. Now let \( \Sigma_{for,k}, \Sigma_{hib,k} \subseteq \Sigma_k \) be the forcible event set and prohibitive event set, respectively, of agent \( G_k \); then \( \Sigma_{for} = \bigcup_{k=1}^n \Sigma_{for,k} \) and \( \Sigma_{hib} = \bigcup_{k=1}^n \Sigma_{hib,k} \). For each forcible event \( \alpha \in \Sigma_{for} \) there is a local preemptor \( \text{LOC}^\alpha \); and for each prohibitive event \( \beta \in \Sigma_{hib} \) there is a local controller \( \text{LOC}^\beta \). These local preemptors/controllers need to be allocated among individual agents, for each agent may have multiple forcible/prohibitive events. A convenient allocation is to let each local controller/preemptor be owned by exactly one agent; an example is displayed in Fig. 2.

---

6 Composition of multiple TDES involves first taking synchronous product of the untimed DES, and then unifying the time bounds of shared events [1, 14].

7 Note that each \( G_k \) may contain multiple forcible and/or prohibitive events.

---

Fig. 2. Example of distributed control by allocating local preemptors/controllers. Continuing the example in Fig. 1, let plant \( G \) be composed of three agents \( G_k \) with event sets \( \Sigma_k, k \in [1, 3] \). Suppose \( \sigma_1, \sigma_2 \in \Sigma_1, \sigma_3, \sigma_4, \sigma_5 \in \Sigma_2 \), and \( \sigma_3, \sigma_4, \sigma_5 \in \Sigma_3 \); thus \( G_1 \) and \( G_2 \) share event \( \sigma_2 \), and \( G_2 \) and \( G_3 \) share event \( \sigma_3 \). Then a convenient allocation is displayed, where each local controller/preemptor is owned by exactly one agent. The allocation creates a distributed control architecture for the multi-agent plant, in which each agent acts semi-autonomously while interacting with other agents through communication of shared events.

Choosing this or (obvious) alternative ways of allocation would be case-dependent.

### 4 Procedure of Supervisor Localization

We solve the Supervisor Localization Problem of TDES by developing a localization procedure for the supervisor’s preemptive and disabling action, respectively. The procedure extends the untimed counterpart in [3]. In particular, localizing the supervisor’s preemption of event \( \text{tick} \) with respect to each individual forcible event is novel in the current TDES setup, for which we introduce below two new ideas “preemption consistency relation” and “preemption cover”.

Given a TDES plant \( G = (Q, \Sigma, \delta, q_0, Q_m) \) (as in (4)) and a corresponding monolithic supervisor \( \text{SUP} = (X, \Sigma, \xi, x_0, X_m) \) (as in (13)) with respect to an imposed specification, we present the localization of \( \text{SUP} \)’s preemptive and disabling action in the sequel.

#### 4.1 Localization of Preemptive Action

Fix an arbitrary forcible event \( \alpha \in \Sigma_{for} \). First define \( E_{\text{tick}} : X \to \{ 1, 0 \} \) according to

\[ E_{\text{tick}}(x) = 1 \text{ iff } \xi(x, \text{tick})! \tag{17} \]

Thus \( E_{\text{tick}}(x) = 1 \) means that \( \text{tick} \) is defined at state \( x \) in \( \text{SUP} \). Next define \( F_\alpha : X \to \{ 1, 0 \} \) according to
Definition 2

Let \( R^P \) be the preemption relation on states. Recall that \( R^P \) is reflexive, symmetric, and transitive.

Thus a pair of states \((x, x')\) in \( \text{SUP} \) is not preemption consistent with respect to \( \alpha \) only when \( \text{tick} \) is defined at \( x \) but is preempted by \( \alpha \) at \( x' \), or vice versa. Otherwise, \( x \) and \( x' \) are preemption consistent, i.e. \((x, x') \in R^P \).

A preemption cover \( C^P_{\alpha} \) lumps states of \( \text{SUP} \) into (possibly overlapping) cells \( X_i, i \in I \). According to (i) all states that reside in a cell \( X_i \) must be pairwise preemption consistent; and (ii) for every event \( \sigma \in \Sigma \), all states that can be reached from any states in \( X_i \) by a one-step transition \( \sigma \) must be covered by the same cell \( X_j \).

Having defined a preemption cover \( C^P_{\alpha} \) on \( X \), we construct, below, a local preemperer \( \text{LOC}^P_{\alpha} = (Y_{\alpha}, \Sigma, \zeta_{\alpha}, y_{0,\alpha}, Y_{m,\alpha}) \) for the forcible event \( \alpha \) to preempt \( \text{tick} \).

(Step 1) The state set is \( Y_{\alpha} := I \), with each state \( y \in Y_{\alpha} \) being a cell \( X_i \) of the cover \( C^P_{\alpha} \). In particular, the initial state \( y_{0,\alpha} \) is a cell \( X_{i0} \) where \( x_0 \) belongs, i.e. \((x_0, x) \in R^P \), and the marker set \( Y_{m,\alpha} := \{i \in I | X_i \cap X_m \neq \emptyset\} \).

(Step 2) For the event set \( \Sigma_{\alpha} \), define the transition function \( \zeta_{\alpha} : I \times \Sigma \rightarrow I \) over the entire event set \( \Sigma \) by \( \zeta_{\alpha}(i, \sigma) = j \) if

\[
(\exists x \in X_i)(x, \sigma) \in X_j \quad \text{and} \quad (\forall x' \in X_i)[(x', \sigma) \Rightarrow (x', \sigma) \in X_j].
\]

Choose \( \Sigma_{\alpha} \) to be the union of \( \{\alpha, \text{tick}\} \) with other events which are not selfloop transitions of \( \zeta_{\alpha} \), i.e.

\[
\Sigma_{\alpha} := \{\alpha, \text{tick}\} \cup \{\sigma : \Sigma - \{\alpha, \text{tick}\} \mid (\exists i, j \in I) i \neq j \Rightarrow \zeta_{\alpha}(i, \sigma) = j\}.
\]

Intuitively, only those non-selfloop transitions may affect decisions on \( \text{tick} \) preemption, and thus the events that are only selfloops may be removed. Note that \( \{\alpha, \text{tick}\} \subseteq \Sigma_{\alpha} \subseteq \Sigma \).

(Step 3) Define the transition function \( \zeta_{\alpha} \) to be the restriction of \( \zeta_{\alpha} \) to \( \Sigma_{\alpha} \); namely \( \zeta_{\alpha} := \zeta_{\alpha}|_{\Sigma_{\alpha}} : I \times \Sigma_{\alpha} \rightarrow I \) according to \( \zeta_{\alpha}(i, \sigma) = \zeta_{\alpha}(i, \sigma) \) for every \( i \in I \) and \( \sigma \in \Sigma_{\alpha} \).

We note that \( \text{LOC}^P_{\alpha} \) thus constructed is not a TDES as defined in (4), for its states do not contain timer information. \( \text{LOC}^P_{\alpha} \) is indeed a generalized TDES because...
its event set $\Sigma_\alpha$ contains tick. We will be concerned only
with its behavior, namely its closed and marked lan-
guages. Also note that, owing to possible overlapping of
cells in the cover $C_\alpha^\beta$, the choices of $y_{0,\alpha}$ and $\zeta_\alpha$
may not be unique, and consequently $ LOC_\alpha^\beta$ may not be unique.
In that case we pick an arbitrary instance of $ LOC_\alpha^\beta$. If
$ C_\alpha^\beta$ happens to be a preemption congruence, however,
then $ LOC_\alpha^\beta$ is unique.

By the same procedure, we generate a set of local pre-
emptors $ LOC_\alpha^\beta$, one for each forcible event $\alpha \in \Sigma_{for}$.
We will verify below that these generated preemptions
collectively achieve the same preemptive action of event
tick as the monolithic supervisor $ SUP$ does.

### 4.2 Localization of Disabling Action

Next, we turn to the localization of $ SUP$’s disabling
action, which is analogous to the treatment in [3]. Fix
an arbitrary prohibitible event $\beta \in \Sigma_{hib}$. First define
$E_\beta : X \to \{1, 0\}$ according to

$$E_\beta(x) = 1 \text{ iff } \xi(x, \beta)!$$

So $E_\beta(x) = 1$ means that $\beta$ is defined at state x in $ SUP$.
Next define $D_\beta : X \to \{1, 0\}$ according to $ D_\beta(x) = 1 \text{ iff} \n$$\neg \xi(x, \beta)!$. & $(\exists s \in \Sigma^\star) (\xi(x_0, s) = x \& \delta(q_0, s)!)$

Thus $D_\beta(x) = 1$ means that $\beta$ must be disabled at x
(i.e. $\beta$ is disabled at x in $ SUP$ but is defined at some
state in the plant $G$ corresponding to x via string s). In
addition, define $M : X \to \{1, 0\}$ according to

$$M(x) = 1 \text{ iff } x \in X_m.$$ 

Thus $M(x) = 1$ means that state x is marked in $ SUP$.
Finally define $T : X \to \{1, 0\}$ according to

$$T(x) = 1 \text{ iff } (\exists s \in \Sigma^\star) \xi(x_0, s) = x \& \delta(q_0, s)! \in Q_m$$

So $T(x) = 1$ means that some state, corresponding to x via
s, is marked in $G$. Note that for each x $\in X$, it follows
from $L_m(SUP) \subseteq L_m(G)$ that $T(x) = 0 \Rightarrow M(x) = 0$
and $M(x) = 1 \Rightarrow T(x) = 1$ [3].

Based on (23)-(26), we define the following binary relation
$R_\beta^C \subseteq X \times X$, called control consistency with respect
to prohibitible event $\beta$ (cf. [3]), according to $(x, x') \in R_\beta^C$ if

$$E_\beta(x) \cdot D_\beta(x') = 0 = E_\beta(x') \cdot D_\beta(x),$$

$$T(x) = T(x') \Rightarrow M(x) = M(x').$$

Thus a pair of states $(x, x')$ in $ SUP$ satisfies $(x, x') \in R_\beta^C$ if

(i) event $\beta$ is defined at one state, but not disabled at
the other; and (ii) x and $x'$ are both marked or both un-
marked in $ SUP$, provided both are marked or unmarked
in $G$. It is easily verified that $R_\beta^C$ is generally not trans-
itive [3], thus not an equivalence relation. Now let $I$ be
some index set, and $C_\beta^G = \{X_i \subseteq X | i \in I\}$ a cover on
$X$. Similar to Definition 2, we define $C_\beta^G$ to be a control
cover with respect to $\beta$ if

$$(i) (\forall i \in I, \forall x, x' \in X_i) (x, x') \in R_\beta^C$$

$$(ii) (\forall i \in I, \forall \sigma \in \Sigma) \left[ (\exists x \in X_i) \xi(x, \sigma)! \Rightarrow \xi(x', \sigma) \in X_j \right].$$

Note that the only difference between control cover and
preemption cover in Definition 2 is the binary relation
(control consistency $R_\beta^C$ or preemption consistency $R_\beta^P$)
used in condition (i).

With the control cover $C_\beta^G$ on $X$, we construct by the
same steps (Step1 - Step3), above, a local controller
$LOC_\beta^C = (Y_\beta, \Sigma_\beta, \zeta_\beta, y_0, Y_m, \beta)$ for prohibitible event
$\beta$. Here, the choice of event set $\Sigma_\beta$ is (cf. (22))

$$\Sigma_\beta := \{\beta\} \cup \{\sigma \in \Sigma \setminus \{-\beta\} | (\exists i, j \in I)$$

$$i \neq j \& \zeta_\beta(i, \sigma) = j\}.$$  

$\Sigma_\beta$ need not contain event tick, as noted in Footnote 5.
As before, owing to possible overlapping of cells in the
control cover $C_\beta^G$, a local controller $ LOC_\beta^C$ need not be
unique. If, however, $C_\beta^G$ happens to be a control congruence
(i.e. $C_\beta^C$ is a partition on $X$), then $ LOC_\beta^C$ is unique.
In the same way, we generate a set of local controllers
$ LOC_\beta^C$, one for each prohibitible event $\beta \in \Sigma_{hib}$. We will
verify that the collective disabling action of these local
controllers is identical to that of the monolithic supervi-
sor $ SUP$.

Finally, notice that an event $\beta$ may be both prohibitible
and forcible. In that case, $\beta$ will be equipped with both a
local controller which exercises disabling action specific
to $\beta$, and a local preemtor which implements preemp-
tion of event tick via $\beta$. It appears that here a conflict
could arise: $\beta$’s local preemtor intends to use $\beta$ to pre-
empt tick, but $\beta$ is disabled by its local controller. How-
ever, since $\beta$’s local preemtor and controller are both
derived from $ SUP$ which is proved to contain no such
conflict [1, 14], the conflict indeed cannot arise between
$\beta$’s local preemtor and controller. Our main result
below confirms this fact.
4.3 Main Result

Here is the main result of this section, which states that the local preempts and controllers generated by the proposed localization procedure collectively achieve the monolithic optimal and nonblocking supervision.

**Theorem 3** The set of local preemptors \( \{ \text{LOC}^p_\alpha \mid \alpha \in \Sigma_{for} \} \) and the set of local controllers \( \{ \text{LOC}^c_\beta \mid \beta \in \Sigma_{hib} \} \) constructed above solve the Supervisor Localization Problem; that is,

\[
L(G) \cap L(\text{LOC}) = L(\text{SUP}),
\]

\[
L_m(G) \cap L_m(\text{LOC}) = L_m(\text{SUP}).
\]

where \( L(\text{LOC}) \) and \( L_m(\text{LOC}) \) are as defined in (15) and (16), respectively.

Theorem 3 extends the untimed supervisor localization result in [3] to the TDES setup, where not only the disabling action but also the tick-preemptive action of the monolithic supervisor needs to be localized. Thus supervisor localization in TDES generates a set of local controllers, one for each individual prohibitible event, as well as a set of local preempts, one for each individual forcible event. The proof of Theorem 3, below, relies on the concepts of TDES controllability, control cover, as well as preemption cover.

Since for every preemption cover (resp. control cover), the presented procedure constructs a local preemt (resp. preemption cover), Theorem 3 asserts that every set of preemption and control covers together generates a solution to the Supervisor Localization Problem. In particular, a set of state-minimal local preemptors (resp. local controllers), possibly non-unique, can in principle be defined from a set of suitable preemption covers (resp. control covers). The minimal state problem, however, is known to be NP-hard [11]. In [3] we proposed, nevertheless, a polynomial-time algorithm which computes congruences instead of covers; and empirical evidence was given that significant state size reduction can often be achieved. That localization algorithm (see [3, Section III-B]) for untimed DES can easily be adapted in the current TDES case, the only modification being to use the new definitions of preemption and control consistency given in Sections 4.1 and 4.2.

So far we have focused on localization of the monolithic supervisor. In fact, the developed localization procedure may be applied to decompose a modular (decentralized or hierarchical) supervisor just as well. Thus when a TDES is large-scale and the monolithic supervisor not feasibly computable, we may in principle combine localization with an effective modular supervisory synthesis: first compute a set of modular supervisors which achieves the same behavior as the monolithic supervisor, and then apply localization to decompose each modular supervisor in the set. This is done in [3, 4] for large-scale untimed DES; and we aim to work out the timed counterpart in future research.

We now provide the proof of Theorem 3. Equation (31) and the (\( \preceq \)) direction of (30) may be verified analogously as in [3]. For completeness we present the verification in the Appendix. Here we prove (\( \succeq \)) in (30), which involves the TDES’s controllability definition, preemption consistency, and control consistency.

**Proof of Theorem 3.** (\( \preceq, 30 \)) We show this by induction. First, the empty string \( \epsilon \) belongs to \( L(G) \), \( L(\text{LOC}) \), and \( L(\text{SUP}) \), because these languages are all nonempty. Next, suppose \( s \in L(G) \cap L(\text{LOC}) \), \( s \in L(\text{SUP}) \), and \( s\sigma \in L(G) \cap L(\text{LOC}) \) for an arbitrary event \( \sigma \in \Sigma \). It will be proved that \( s\sigma \in L(\text{SUP}) \). Since \( \Sigma = \Sigma_u \cup \Sigma_c = \Sigma_u \cup \{ \text{tick} \} \cup \Sigma_{hib} \) (as in (9)), we consider the following three cases.

(i) Let \( \sigma \in \Sigma_u \). Since \( L_m(\text{SUP}) \) is controllable (see (12)), and \( \sigma \in L(G) \), i.e. \( \sigma \in \text{Elig}_G(s) \) by (10), we have \( \sigma \in \text{Elig}_{L_m}(\text{SUP}) \). That is, by (11) \( s \sigma \in L_m(\text{SUP}) = L(\text{SUP}) \).

(ii) Let \( \sigma = \text{tick} \). We will show \( \text{tick} \in \text{Elig}_{L_m}(\text{SUP}) \). To conclude that \( s, \text{tick} \in L_m(\text{SUP}) \), By the hypothesis that \( s, \text{tick} \in L(\text{LOC}) \) and equation (15), we have \( s, \text{tick} \in \text{Elig}(s) \). Recall \( \text{Elig}(s) = (Y, \Sigma, \zeta, y_0, a, Y_m, a) \), and let \( i := \zeta_0(y_0, a, s) \). By definition of \( \zeta_0 \) in (21), \( i \notin \Sigma_0 \) (defined in (22)) is only a selfloop transition of \( \zeta' \); hence \( \zeta'(y_0, s) = i \). By (21) again, there exist \( x, x' \in X_i \) and \( x'' \in X_i \) such that \( (x, 0, s) = x \) and \( \xi(x', \text{tick}) = x'' \) in \( \text{SUP} \). These state-transition correspondences between \( \text{LOC}^p \) and \( \text{SUP} \) are displayed in Fig. 4.

Now that \( x, x' \) belong to the same cell \( X_i \), by the preemption cover definition (Definition 2) \( x \) and \( x' \) must be preemption consistent, i.e. \( (x, x') \in \mathcal{R}_P \). Since \( \xi(x', \text{tick}) \), by (17) we have \( E_{\text{tick}}(x') = 1 \). Thus the requirement \( E_{\text{tick}}(x') \cdot F_{\alpha}(x) = 0 \) (Definition 1) yields that \( F_{\alpha}(x) = 0 \). The latter, by (18), gives rise to the following three cases: (Case 1) \( -\forall (x, a) \), (Case 2) \( \xi(x, \text{tick})' \), or (Case 3) \( -\exists s \in \Sigma' \{(x_0, s) = x \} \). First, Case 3 is impossible, because by the hypothesis that \( s \in L(\text{SUP}) \) and \( \text{tick} \in L(G) \) we have \( \xi(x_0, s) \) and \( \delta(y_0, s, \text{tick}) \).

Next, Case 2 means directly \( \text{tick} \in \text{Elig}_{m}(\text{SUP}) \); finally, Case 1 implies \( \alpha \notin \text{Elig}_{m}(\text{SUP}) \); note that this holds for all \( \alpha \in \Sigma_{for} \). Hence \( \text{Elig}_{L_m}(\text{SUP}) \subset \Sigma_{for} = \emptyset \).

Then by the fact that \( \text{SUP} \) is controllable, we derive from (12) that \( \text{tick} \in \text{Elig}_{L_m}(\text{SUP}) \).

(iii) Let \( \sigma \in \Sigma_{hib} \). By the hypothesis \( s, \sigma \in L(\text{LOC}) \) and equation (15), we have \( s, \sigma \in P_{\sigma}^{-1}L(\text{LOC}^c) \),
The following two cases: (Case 1) ξ yields that because by the hypothesis that

\[ x, x' \in X, \quad x' \in X \] such that \( x, x' \) belong to the same cell \( X_i \), by the control cover definition \( x \) and \( x' \) must be control consistent, i.e. \( (x, x') \in R_x^i \). That \( \xi(x', \sigma)! \) implies by (23) that \( E_\sigma(x') \cdot D_\sigma(x) = 0 \) yields that \( D_\sigma(x) = 0 \). The latter, by (24), gives rise to the following two cases: (Case 1) \( \xi(x, \sigma)! \), or (Case 2) \( \neg\exists s \in \Sigma^* \xi(x_0, s) = x & \delta(q_0, s)\sigma! \). Case 2 is impossible, because by the hypothesis that \( s \in L(SUP) \) and \( s.tick \in L(G) \) we have \( \xi(x_0, s)! \) and \( \delta(q_0, s.tick)! \). But in Case 1, \( \xi(x, \sigma)! \) i.e. \( s.\sigma \in L(SUP) \).

5 Case Study: Manufacturing Cell

We illustrate supervisor localization in TDES by studying a manufacturing cell example, taken from [1, 14, Section 9.11]. As displayed in Fig. 5, the cell consists of two machines, MACH1 and MACH2, an input conveyor CONV1 as an infinite source of workpieces, and output conveyor CONV2 as an infinite sink. Each machine processes two types of parts, P1 and P2. Each type of part is routed as shown in Fig. 5. The untimed DES models of the machines are also displayed in Fig. 5; here \( \alpha_{ij} \) (\( i, j \in [1, 2] \)) is the event “MACHi starts to work on a Pj-part”, while \( \beta_{ij} \) (\( i, j \in [1, 2] \)) is “MACHi finishes working on a Pj-part”. Assign lower and upper time bounds to each event, with the notation (event, lower bound, upper bound), as follows:

MACH1’s timed events:

\[ (\alpha_{11}, 1, \infty) \quad (\beta_{11}, 3, 3) \quad (\alpha_{12}, 1, \infty) \quad (\beta_{12}, 2, 2) \]

MACH2’s timed events:

\[ (\alpha_{21}, 1, \infty) \quad (\beta_{21}, 1, 1) \quad (\alpha_{22}, 1, \infty) \quad (\beta_{22}, 4, 4) \]

So \( \alpha_{ij} \) are remote events (upper bound \( \infty \)), and \( \beta_{ij} \) prospective events (finite upper bounds). Now the TDES models of the two machines can be generated [14, p.425]. Their joint behavior is the synchronous product of the two TDES, which in this example is the plant to be controlled.

To impose behavioral constraints on the two machines’ joint behavior, we take the events \( \alpha_{ij} \) to be both prohibitable and forcible, i.e. \( \Sigma_{hij} = \Sigma_{for} = \{ \alpha_{ij} | i, j = 1, 2 \} \), and the \( \beta_{ij} \) to be uncontrollable, i.e. \( \Sigma_u = \{ \beta_{ij} | i, j = 1, 2 \} \). We impose the following logical control specifications as well as a temporal specification:

(S1) A P1-part must be processed first by MACH1 and then by MACH2.

(S2) A P2-part must be processed first by MACH2 and then by MACH1.

(S3) One P1-part and one P2-part must be processed in a production cycle.

(S4) A production cycle must be completed in at most 8 time units. \footnote{Here we choose “8 time units” because it is, according to [1,}
These four specifications are formalized as automata SPEC1, SPEC2, SPEC3, and SPEC4, respectively, as displayed in Fig. 6. The temporal specification SPEC4 is simply an 8-tick sequence, with all states marked; SPEC4 forces any TDES with which it is synchronized to halt after at most 8 ticks, i.e., after 8 ticks to execute no further event whatever except event tick. Thus it extracts the marked strings (if any) which satisfy this constraint, namely the “tasks” of TDES that can be accomplished in at most 8 ticks (which turns out to be exactly one production cycle according to [1, 14]).

Now the plant to be controlled is the synchronous product of TDES MACH1 and MACH2 [14, p.425], and the overall control specification is the synchronous product of automata SPEC1-SPEC4 in Fig. 6. We compute as in (14) the corresponding monolithic optimal and nonblocking supervisor SUP, the computation is done by the supcon command in XPTTCT [13]. SUP has 19 states and 21 transitions, as displayed in Fig. 7. We see that SUP represents the behavior that the manufacturing cell accomplishes exactly one working cycle, within 8 ticks, producing one P1-part and one P2-part. Indeed, each event is executed exactly once, and each forcible event preempts tick immediately after it becomes eligible to occur.

We now apply supervisor localization to decompose the monolithic supervisor SUP into local preemptors and local controllers, respectively for each forcible event and each prohibitive event. Specifically, since \( \Sigma_{hib} = \Sigma_{for} = \{ \alpha_{ij} | i, j = 1, 2 \} \), we compute a local preemptor and a local controller for each \( \alpha_{ij} \), responsible for \( \alpha_{ij} \)'s tick-preemptive action and its disabling action, respectively. This computation can be done by an algorithm adapted from [3] (as discussed in Section 4.3): here, however, owing to the simple (chain-like) structure of SUP (Fig. 7), local preemptors/controllers can be derived by inspection. We demonstrate such a derivation below, which results in a local preemptor \( \text{LOC}^P_{\alpha_{11}} \) for the forcible (and prohibitive) event \( \alpha_{11} \). Other derivations of local preemptors/controllers are similar.

To derive a local preemptor \( \text{LOC}^P_{\alpha_{11}} \) for event \( \alpha_{11} \), we find a preemption cover \( C^P_{\alpha_{11}} \) for \( \alpha_{11} \) on SUP’s state set as follows. Initialize \( C^P_{\alpha_{11}} \) to be \( C^P_{\alpha_{11}} = \{ \{0\}, \{1\}, \{2\}, \ldots, \{18\} \} \), i.e. each cell contains exactly one state of SUP. Subsequently, we merge as many cells together as possible according to Definitions 1 and 2, while maintaining \( C^P_{\alpha_{11}} \) to be a preemption cover.

(i) Cells \([0]\) and \([1]\) cannot be merged. Since \( E_{\text{tick}}(0) = 1 \) (event tick is defined at state 0) and \( F_{\alpha_{11}}(1) = 1 \) (tick is preempted by \( \alpha_{11} \) at state 1), the pair of states \((0, 1)\) is not preemption consistent, i.e. \((0, 1) \notin R^P_{\alpha_{11}} \). Consequently, merging cells \([0]\) and \([1]\) violates requirement (i) of preemption cover (Definition 2).

(ii) Cells \([1], [3]\) and cells \([2], [4]\) can be merged. For cells \([2]\) and \([4]\), we have \( F_{\alpha_{11}}(2) = 0 \), \( E_{\text{tick}}(2) = 0 \) (tick is preempted at state 2), but by \( \alpha_{22} \) not by \( \alpha_{11} \) and \( E_{\text{tick}}(4) = 1 \), \( F_{\alpha_{11}}(4) = 0 \) (event tick is defined at state 4). Thus \((2, 4) \in R^P_{\alpha_{11}} \), which satisfies requirement (i) of preemption cover. Moreover since no common event is defined on states 2 and 4, requirement (ii) of preemption cover is trivially satisfied. Therefore cells \([2], [4]\) can be merged.

For cells \([1]\) and \([3]\), we have \( F_{\alpha_{11}}(1) = F_{\alpha_{11}}(3) = 1 \) (tick is preempted by \( \alpha_{11} \) at both states 1 and 3) and \( E_{\text{tick}}(1) = E_{\text{tick}}(3) = 0 \). Thus \((1, 3) \in R^P_{\alpha_{11}} \), which satisfies requirement (i) of preemption cover. Now event \( \alpha_{11} \) is defined at both states 1 and 3, but it leads to states 2 and 4 respectively, which have been verified to be preemption consistent. Hence, requirement (ii) of preemption cover is also satisfied, and cells \([1], [3]\) can be merged.

By merging the above two pairs of cells, we derive \( C^P_{\alpha_{11}} = \{\{0\}, \{1, 3\}, \{2, 4\}, \{5\}, \ldots, \{18\}\} \).

(iii) Cells \([2], [4], [5], \ldots, [18]\) can all be merged together. Note, indeed, that \( F_{\alpha_{11}}(\cdot) = 0 \) for all these states (no tick preemption by \( \alpha_{11} \)). On checking the preemption consistency and preemption cover definitions as above, we conclude that the final preemption cover is \( C^P_{\alpha_{11}} = \{\{0\}, \{1, 3\}, \{2, 4, 5, \ldots, 18\}\} \). It is in fact a preemption congruence.

Having found the preemption cover \( C^P_{\alpha_{11}} \), we apply (Step 1) - (Step 3) in Section 4.1 to construct a local preemptor \( \text{LOC}^P_{\alpha_{11}} \), with transition structure displayed in Fig. 8. Note that the event set of \( \text{LOC}^P_{\alpha_{11}} \) is exactly \( \{\alpha_{11}, \text{tick}\} \), which means that \( \text{LOC}^P_{\alpha_{11}} \) does not need to observe any external events in order to execute its preemptive action. Similarly, we derive other local preemptors and local controllers, all displayed in Figs. 8 and 9. Here, for example, the event set of \( \text{LOC}^P_{\alpha_{12}} \) is
Finally, with the derived set of local preemptors and controllers, we build a distributed control architec-

\{α12, tick, β22\}; so event β22 originating in MACH2 has to be observed by LOC_{α12}. We have then verified that their joint behavior (via synchronous product) is identical to the monolithic optimal and nonblocking behavior of SUP, i.e. (30) and (31) hold.

We see that each local preemptor/controller has fewer states, with a simpler structure, than the monolithic SUP; this renders each one’s preemptive/disabling action more transparent. For example, the local preemptor \(LOC_{α11}^P\) (resp. \(LOC_{α22}^P\)) in Fig. 8 means that after one tick, forcible event α11 preempts event tick and MACH1 starts to work on a P1-part (resp. α22 preempts tick and MACH2 works on a P2-part). This is possible because α11 (resp. α22) has lower time bound 1 and becomes eligible to occur after one tick. For another example, the local preemptor \(LOC_{α22}^P\) in Fig. 8 specifies that after occurrence of α12 followed by a tick, forcible event α21 preempts tick and MACH2 starts to work on a P1-part. This preemption is due to the fact that α21 has lower time bound 1 and becomes eligible to occur after occurrence of β22 plus one tick (according to Fig. 7 event α22 first occurs in MACH2, which implies from the untimed model in Fig. 5 the event order α22, β22, α21). But occurrence of α12 implies that β22 has just occurred (see Fig. 7).

For control logic, the local controller \(LOC_{α12}^C\) in Fig. 9 means that prohibitive event α12 is enabled only after occurrence of event β22, i.e. MACH1 starts to work on a P2-part only after MACH2 finishes the P2-part. On the other hand, the logic of \(LOC_{α21}^C\) is a bit subtle; it specifies that prohibitive event α21 is enabled after occurrence of event α22. At first glance, the logic seems to violate the specification SPEC1 in Fig. 6, which says that α21 should not be enabled before occurrence of β11. Observe, nevertheless, that α21 cannot become eligible to occur before occurrence of β22 which has lower (and upper) time bound 4, and event β11 in fact has already occurred when β22 occurs (see Fig. 7). Hence it is legal to enable α21 after α22.

Finally, with the derived set of local preemptors and controllers, we build a distributed control architec-

![Fig. 8. Local preemptors for individual forcible events. The alphabet of each local preemptor is the set of events displayed in each automaton.](image)

![Fig. 9. Local controllers for individual prohibitive events. The alphabet of each local controller is the set of events displayed in each automaton.](image)

![Fig. 10. Distributed control architecture for manufacturing cell.](image)

\[ \text{References} \]

[1] B. Brandin and W.M. Wonham. Supervisory control of timed discrete-event systems. IEEE Trans. Autom. Control, 39(2):329–342, 1994.

[2] Y. Brave and M. Heymann. Formulation and control of real time discrete event processes. In Proc. 27th IEEE Conf. on Decision and Control, pages 1131–1132, Austin, TX, 1988.
We complete the proof of Theorem 3, namely equation (31) and (≥) in (30).

(≥, 31) Since $L_m(SUP) \subseteq L_m(G)$, it suffices to show that $L_m(SUP) \subseteq L_m(LOC)$. That is, by (16),

$$\forall \alpha \in \Sigma_{for} \ L_m(SUP) \subseteq P^{-1}_\alpha \ L_m(LOC^\alpha),$$  
(A.1)

$$\forall \beta \in \Sigma_{hub} \ L_m(SUP) \subseteq P^{-1}_\beta \ L_m(LOC^\beta).$$  
(A.2)

We prove (A.1), and (A.2) follows similarly. Let $s = \sigma_0, \sigma_1, \ldots, \sigma_h \in L_m(SUP)$. Then $x_1 := \xi(x_0, \sigma_0), \ldots, x_{h+1} := \xi(x_h, s) \in X_m$. By the construction of $LOC^\alpha$ ($\alpha \in \Sigma_{for}$ arbitrary), in particular the transition function $\zeta'_\alpha$ over $\Sigma$ in (21), there exist $i_0, i_1, \ldots, i_{h+1}$ with ($i_0 = y_{0,\alpha}$) such that

$$x_0 \in X_{i_0} \ & \ & \zeta'_\alpha(i_0, \sigma_0) = i_1,$$
$$x_1 \in X_{i_1} \ & \ & \zeta'_\alpha(i_1, \sigma_1) = i_2,$$
$$\vdots$$
$$x_{h+1} \in X_{i_{h+1}} \ & \ & \zeta'_\alpha(i_{h+1}, \sigma_h) = i_{h+1}.$$  
(A.3)

So $\zeta'_\alpha(i_0, \sigma_0, \ldots, \sigma_h) = \zeta'_\alpha(i_0, s)!$, and belongs to $Y_{m,\alpha}$ because $X_{i_{h+1}} \cap X_m \neq \emptyset (x_{h+1} \text{ belongs to } X_m)$. Moreover since any $\sigma \not\in \Sigma_{\alpha}$ (defined in (22)) is only a selfloop transition of $\zeta'_\alpha$, we derive $\zeta_\alpha(i_0, P_\alpha(s)) \in Y_{m,\alpha}$. Hence, $P_\alpha(s) \in L_m(LOC^\alpha)$, i.e. $s \in P^{-1}_\alpha \ L_m(LOC^\alpha)$.

(≥, 30) This is an easy consequence of (≥, 31):

$$L(SUP) = \frac{L_m(SUP)}{L_m(G) \cap L_m(LOC)} \subseteq L(G) \cap L(LOC).$$

(≤, 31) Let $s \in L_m(G) \cap L_m(LOC)$; by (16), for every $\beta \in \Sigma_{hub}$, $s \in P^{-1}_\beta \ L_m(LOC^\beta)$, i.e. $P_\beta(s) \in L_m(LOC^\beta)$. Write $i := \zeta_\beta(y_{0,\beta}, P_\beta(s))$. Then there exists $x \in X_i \cap X_m$; thus $M(x) = 1$ (defined in (25)), which also implies $T(x) = 1$ (defined in (26)). On the other hand, since $L_m(G) \cap L_m(LOC) \subseteq L(G) \cap L(LOC) = L(SUP)$ (the last equality has already been shown), we have $s \in L(SUP)$. That is, $\xi(x_0, s)!$ as in (A.3) above we derive $\xi(x_0, s) \in X_i$, and by the control cover definition (28) it holds that $(x, \xi(x_0, s)) \in R^C_{\beta}$. Since $s \in L_m(G)$, i.e. $\delta(q_0, s) \in Q_m$, we have $T(\xi(x_0, s)) = 1$. Therefore by requirement (ii) of the control consistency definition (27), $M(\xi(x_0, s)) = 1$, i.e. $s \in L_m(SUP)$.

A Appendix

We complete the proof of Theorem 3, namely equation (31) and (≥) in (30).