Automatica 49 (2013) 2786–2794

Contents lists available at ScienceDirect

Automatica

journal homepage: www.elsevier.com/locate/automatica

Brief paper

Supervision localization of timed discrete-event systems☆

Renyuan Zhang [a], Kai Cai [b], Yongmei Gan [a,1], Zhaoan Wang [a], W.M. Wonham [b]

[a] School of Electrical Engineering, Xi'an Jiaotong University, Xi'an, Shaanxi 710049, China

[b] Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON M5S 3G4, Canada

ARTICLE INFO

Article history:
Received 19 September 2012
Received in revised form
7 May 2013
Accepted 13 May 2013
Available online 17 June 2013

Keywords:
Supervisor localization
Supervisory control
Automata
Timed discrete-event systems
Real-time systems

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 preempting" 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.

1. Introduction

Recently we developed a top-down approach, called *supervisor localization* (Cai & Wonham, 2010a,b) to the distributed control of untimed discrete-event systems (DES) in the Ramadge–Wonham (RW) supervisory control framework (Ramadge & Wonham, 1987; Wonham, 2012). 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.

In this paper we extend the supervisor localization theory to a class of *real-time* DES and address distributed control problems therein. Many time-critical applications can be modeled as real-time DES, such as communication channels, sensor networks, scheduling and resource management (Leung, Lee, & Son, 2007). Typical timing features include communication delays and operational hard deadlines. The correctness and optimality of real-time DES depend not only on the system's logical behavior, but also on the times at which various actions are executed. Moreover, rapid advances in embedded, mobile computation and communication technologies (Leung et al., 2007, Part III) have enabled distributed implementation of control algorithms. These developments jointly motivate this study of supervisor localization for real-time DES.

A variety of real-time DES models and approaches are available. Notable works include Brave and Heymann's "clock automata" (Brave & Heymann, 1988), Ostroff's "timed transition models" (Ostroff, 1990), Brandin and Wonham's timed DES (TDES) (Brandin & Wonham, 1994), and Cofer and Garg's model based on "timed Petri nets" (Cofer & Garg, 1996). 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 permissive 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 (Brandin & Wonham, 1994), (Wonham, 2012, Chapter 9). While it may be possible to develop supervisor

☆ This work was supported in part by the State Key Laboratory of Electrical Insulation and Power Equipment (China) and by the Natural Sciences and Engineering Research Council (Canada), Grant no. 7399. Part of the material in this paper was presented in Proc. American Control Conference 2013. This paper was recommended for publication in revised form by Associate Editor Jan Komenda under the direction of Editor Ian R. Petersen.

E-mail addresses: r.yuan.zhang@gmail.com (R. Zhang), kai.cai@scg.utoronto.ca (K. Cai), ymgan@mail.xjtu.edu.cn (Y. Gan), zawang@mail.xjtu.edu.cn (Z. Wang), wonham@control.utoronto.ca (W.M. Wonham).

[1] Tel.: +86 29 8266 6241; fax: +86 29 8266 5223.

0005-1098/$ – see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.automatica.2013.05.015

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 Cai and Wonham (2010a) and Cai and Wonham (2010b). In this timed localization, a novel feature is "event forcing" as a 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 preemptor", "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 (Brandin & Wonham, 1994).

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 DES

This section reviews the TDES model proposed by Brandin and Wonham (1994), and Wonham (2012, Chapter 9). First consider the untimed DES model

$$\mathbf{G}_{\mathrm{act}} = (A, \Sigma_{\mathrm{act}}, \delta_{\mathrm{act}}, a_0, A_m). \tag{1}$$

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

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

$$t_{\sigma 0} := \begin{cases} u_\sigma & \text{if } \sigma \in \Sigma_{\mathrm{spe}}, \\ l_\sigma & \text{if } \sigma \in \Sigma_{\mathrm{rem}}. \end{cases} \tag{2}$$

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

$$T_\sigma := \begin{cases} [0, u_\sigma] & \text{if } \sigma \in \Sigma_{\mathrm{spe}}, \\ [0, l_\sigma] & \text{if } \sigma \in \Sigma_{\mathrm{rem}}. \end{cases} \tag{3}$$

Based on (1)–(3), the TDES model $\mathbf{G}$ is given by

$$\mathbf{G} := (Q, \Sigma, \delta, q_0, Q_m), \tag{4}$$

where $Q := A \times \prod\{T_\sigma | \sigma \in \Sigma_{\mathrm{act}}\}$ ($\prod$ denotes *Cartesian product*) is the finite set of *states*, a state $q \in Q$ being of the form $q = (a, \{t_\sigma | \sigma \in \Sigma_{\mathrm{act}}\})$ (i.e. a $(1 + |\Sigma_{\mathrm{act}}|)$-tuple); $\Sigma := \Sigma_{\mathrm{act}} \dot\cup \{tick\}$ is the finite set of events; $\delta : Q \times \Sigma \to Q$ is the (partial) *state transition function*; $q_0 = (a_0, \{t_{\sigma 0} | \sigma \in \Sigma_{\mathrm{act}}\})$ ($t_{\sigma 0}$ as in (2)) is the *initial state*; and $Q_m \subseteq A_m \times \prod\{T_\sigma | \sigma \in \Sigma_{\mathrm{act}}\}$ is the set of *marker states*. Starting from q_0, TDES $\mathbf{G}$ executes state transitions in accordance with its transition function δ. Let $q = (a, \{t_\alpha | \alpha \in \Sigma_{\mathrm{act}}\}) \in Q$ and $\sigma \in \Sigma_{\mathrm{act}}$; δ is defined at (q, σ), written as $\delta(q, \sigma)!$, if δ_{act} of $\mathbf{G}_{\mathrm{act}}$ is defined at (a, σ) (i.e. $\delta_{\mathrm{act}}(a, \sigma)!$) and timer t_σ satisfies (i) $0 \leq t_\sigma \leq u_\sigma - l_\sigma$ when $\sigma \in \Sigma_{\mathrm{spe}}$, and (ii) $t_\sigma = 0$ when $\sigma \in \Sigma_{\mathrm{rem}}$. The new state $q' = \delta(q, \sigma)$ is given by $q' = (\delta_{\mathrm{act}}(a, \sigma), \{t'_\alpha | \alpha \in \Sigma_{\mathrm{act}}\})$, where t'_σ is set to be its default value $t_{\sigma 0}$ as in (2); for other timers t_α, $\alpha \neq \sigma$, the reader is referred to detailed updating rules given in Brandin and Wonham (1994) and Wonham (2012). On the other hand, $\delta(q, tick)!$ if no timer of a prospective event is zero, and $q' = \delta(q, tick) = (a, \{t'_\alpha | \alpha \in \Sigma_{\mathrm{act}}\})$, i.e. there is no change in the activity component a of q, while the rules for updating timers are again referred to Brandin and Wonham (1994) and Wonham (2012).

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

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

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

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

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

We introduce the languages generated by TDES $\mathbf{G}$ in (4). The transition function δ is extended to $\delta : Q \times \Sigma^* \to Q$ in the usual way. The *closed behavior* of $\mathbf{G}$ is the language $L(\mathbf{G}) := \{s \in \Sigma^* | \delta(q_0, s)!\}$, and the *marked behavior* is $L_m(\mathbf{G}) := \{s \in L(\mathbf{G}) | \delta(q_0, s) \in Q_m\}$. We say that $\mathbf{G}$ is *nonblocking* if the *prefix closure* (Wonham, 2012) $\bar{L}_m(\mathbf{G})$ satisfies $\bar{L}_m(\mathbf{G}) = L(\mathbf{G})$.

To use TDES $\mathbf{G}$ 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 (Wonham, 2012), 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 $\Sigma_{\mathrm{hib}} \subseteq \Sigma_{\mathrm{rem}}$ denote the *prohibitible* events; the supervisor is allowed to disable any prohibitible 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, inasmuch as no control technology can stop the global clock indefinitely. On this basis let a new subset $\Sigma_{\mathrm{for}} \subseteq \Sigma_{\mathrm{act}}$ denote the *forcible* events; a forcible event is one that preempts event *tick*: if, at a state q of $\mathbf{G}$, *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).[2] There is no particular relation

[2] One may also think of forcible events as being able to occur so fast that they can occur between *ticks*. For a more general use of forcible events, see Golaszewski and Ramadge (1987).

postulated *a priori* between Σ_{for} and any of Σ_{hib}, Σ_{rem} or Σ_{spe}; in particular, a remote event may be both forcible and prohibitible. It is now convenient to define the *controllable* event set $\Sigma_c := \Sigma_{\text{hib}} \dot\cup \{tick\}$. Here designating both Σ_{hib} and *tick* controllable is to simplify terminology. We emphasize that events in Σ_{hib} can be disabled indefinitely, while *tick* may be preempted only by events in Σ_{for}. The *uncontrollable* event set Σ_u is $\Sigma_u := \Sigma - \Sigma_c = \Sigma_{\text{spe}} \dot\cup (\Sigma_{\text{rem}} - \Sigma_{\text{hib}})$.

We introduce the notion of controllability as follows. For a string $s \in L(\mathbf{G})$, define $\text{Elig}_{\mathbf{G}}(s) := \{\sigma \in \Sigma | s\sigma \in L(\mathbf{G})\}$ to be the subset of events 'eligible' to occur (i.e. defined) at the state $q = \delta(q_0, s)$. Consider an arbitrary language $F \subseteq L(\mathbf{G})$ and a string $s \in \overline{F}$; similarly define the eligible event subset $\text{Elig}_F(s) := \{\sigma \in \Sigma | s\sigma \in \overline{F}\}$. We say F is *controllable* wrt. $\mathbf{G}$ in (4) if, for all $s \in \overline{F}$,

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

Thus F controllable means that an event σ is eligible to occur in F if (i) σ is currently eligible in $\mathbf{G}$, and (ii) either σ is uncontrollable or $\sigma = tick$ when there is no forcible event currently eligible in F. Recall that in the untimed supervisory control theory (Ramadge & Wonham, 1987; Wonham, 2012), F controllable means that the occurrence of an uncontrollable event in $\mathbf{G}$ will not cause a string $s \in \overline{F}$ to exit from $\overline{F}$; 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 $\mathcal{C}(F)$ the set of all controllable sublanguages of F. Then $\mathcal{C}(F)$ is nonempty, closed under arbitrary set unions, and thus contains a unique supremal element denoted by $\sup\mathcal{C}(F)$ (Brandin & Wonham, 1994; Wonham, 2012). Now consider a specification language $E \subseteq \Sigma^*$ imposed on the timed behavior of $\mathbf{G}$; E may represent a logical and/or temporal requirement. Let[3] $\mathbf{SUP} = (X, \Sigma, \xi, x_0, X_m)$ be the corresponding *monolithic supervisor* that is optimal (i.e., maximally permissive) and nonblocking in the following sense: **SUP**'s marked language $L_m(\mathbf{SUP})$ is

$$L_m(\mathbf{SUP}) = \sup\mathcal{C}(E \cap L_m(\mathbf{G})) \subseteq L_m(\mathbf{G}) \tag{7}$$

and moreover its closed language $L(\mathbf{SUP})$ is $L(\mathbf{SUP}) = \overline{L_m(\mathbf{SUP})}$. We note that in order to achieve optimal and nonblocking supervision, **SUP** should correctly disable prohibitible events and preempt *tick* via forcible events.

3. Formulation of localization problem

Let TDES $\mathbf{G}$ in (4) be the plant to be controlled, and E be a specification language. Synthesize as in (7) the monolithic optimal and nonblocking supervisor **SUP**; throughout the paper we assume that $\overline{L_m(\mathbf{SUP})} \neq \emptyset$. Supervisor **SUP**'s control action includes (i) disabling prohibitible events in Σ_{hib} and (ii) preempting *tick* via forcible events in Σ_{for}. This section formulates the localization of **SUP**'s control action with respect to each prohibitible event as well as to each forcible event; an illustration of localization is provided in Fig. 1. Compared to Cai and Wonham (2010a), the present *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 Cai and Wonham (2010a); on the other hand, the localization of forcible events' preemptive action is specific to TDES, and we introduce below the new concept "local preemptor". Further, we will discuss applying supervisor localization to the distributed control of multi-agent TDES.

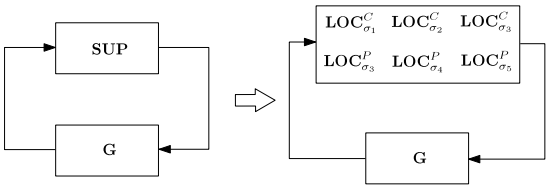


Fig. 1. Supervisor localization example for illustration: let $\Sigma_{\text{hib}} = \{\sigma_1, \sigma_2, \sigma_3\}$, $\Sigma_{\text{for}} = \{\sigma_3, \sigma_4, \sigma_5\}$; note $\sigma_3 \in \Sigma_{\text{hib}} \cap \Sigma_{\text{for}}$. Localization of **SUP**'s control action includes two parts: (i) localizing its disabling action into three *local controllers* $\mathbf{LOC}^C_{\sigma_i}$, $i = 1, 2, 3$, and (ii) localizing its preemptive action into three *local preemptors* $\mathbf{LOC}^P_{\sigma_j}$, $j = 3, 4, 5$.

First, let $\alpha \in \Sigma_{\text{for}}$ be an arbitrary forcible event. We say that $\mathbf{LOC}^P_\alpha = (Y_\alpha, \Sigma_\alpha, \zeta_\alpha, y_{0,\alpha}, Y_{m,\alpha})$,[4] $\Sigma_\alpha \subseteq \Sigma$, is a *local preemptor* (for α) if α is defined at every state of $\mathbf{LOC}^P_\alpha$ where event *tick* is preempted. Let $P_\alpha : \Sigma^* \to \Sigma^*_\alpha$ be the natural projection as in (5). Then in terms of language, the above condition means that for every $s \in \Sigma^*$ there holds

$$s.tick \in L(\mathbf{G}) \quad \text{and} \quad s \in P_\alpha^{-1}L(\mathbf{LOC}^P_\alpha) \quad \text{and}$$
$$s.tick \notin P_\alpha^{-1}L(\mathbf{LOC}^P_\alpha) \Rightarrow s\alpha \in L(\mathbf{G}) \cap P_\alpha^{-1}L(\mathbf{LOC}^P_\alpha).$$

Notation $s.tick$ means that event *tick* occurs after string s and will be used henceforth. The left side of the above implication means that event *tick* is preempted in $\mathbf{LOC}^P_\alpha$ after string s (after s event *tick* is defined in $L(\mathbf{G})$ but not in $\mathbf{LOC}^P_\alpha$), and the right side says that forcible event α is defined in $\mathbf{LOC}^P_\alpha$ (and in $L(\mathbf{G})$) after s. That is, forcible event α acts to preempt *tick*. The event set Σ_α of $\mathbf{LOC}^P_\alpha$ in general satisfies $\{\alpha, tick\} \subseteq \Sigma_\alpha \subseteq \Sigma$; in typical cases, however, both subset containments are strict, as will be illustrated in Section 5. Also, for simplicity we assume that the lower and upper time bounds of events in Σ_α coincide with the bounds on the corresponding events in Σ (this is, in fact, guaranteed by the localization procedure presented below in Section 4). It is worth emphasizing that Σ_α (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 $\beta \in \Sigma_{\text{hib}}$ be an arbitrary prohibitible event. We say that $\mathbf{LOC}^C_\beta = (Y_\beta, \Sigma_\beta, \zeta_\beta, y_{0,\beta}, Y_{m,\beta})$, $\Sigma_\beta \subseteq \Sigma$, is a *local controller* (for β) if $\mathbf{LOC}^C_\beta$ can disable only event β. Let $P_\beta : \Sigma^* \to \Sigma^*_\beta$ 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. Cai & Wonham, 2010a)

$$s\sigma \in L(\mathbf{G}) \quad \text{and} \quad s \in P_\alpha^{-1}L(\mathbf{LOC}^C_\beta) \quad \text{and}$$
$$s\sigma \notin P_\alpha^{-1}L(\mathbf{LOC}^C_\beta) \Rightarrow \sigma = \beta.$$

The event set Σ_β of $\mathbf{LOC}^C_\beta$ in general satisfies $\{\beta\} \subseteq \Sigma_\beta \subseteq \Sigma$.[5] Like Σ_α above, Σ_β will be generated as part of our localization result to guarantee correct disabling action; again, the events in Σ_β are assumed to have the same lower and upper time bounds as the corresponding events in Σ.

Now we formulate the *Supervisor Localization Problem*. Construct a set of local preemptors $\{\mathbf{LOC}^P_\alpha | \alpha \in \Sigma_{\text{for}}\}$ and a set of local controllers $\{\mathbf{LOC}^C_\beta | \beta \in \Sigma_{\text{hib}}\}$, with

$$L(\mathbf{LOC}) := \left(\bigcap_{\alpha \in \Sigma_{\text{for}}} P_\alpha^{-1}L(\mathbf{LOC}^P_\alpha)\right) \cap \left(\bigcap_{\beta \in \Sigma_{\text{hib}}} P_\beta^{-1}L(\mathbf{LOC}^C_\beta)\right) \tag{8}$$

[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] $\mathbf{LOC}^P_\alpha$ is a generalized TDES; we further explain this below in Section 4.

[5] Event set Σ_β need not contain event *tick*, since $\mathbf{LOC}^C_\beta$'s disabling action may be purely logical and irrelevant to time.

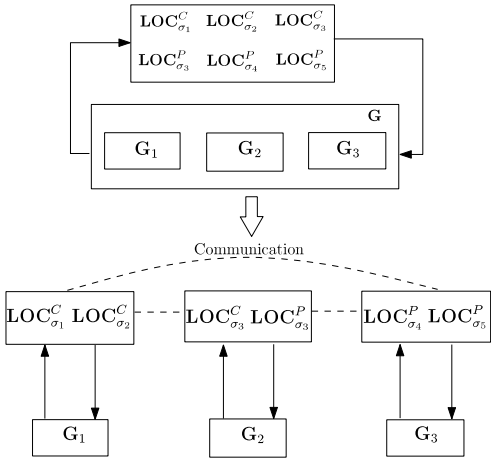


Fig. 2. Example of distributed control by allocating local preemptors/controllers. Continuing the example in Fig. 1, let plant **G** be composed of three agents $\mathbf{G}_k$ with event sets Σ_k, $k \in [1, 3]$. Suppose $\sigma_1, \sigma_2 \in \Sigma_1$, $\sigma_2, \sigma_3 \in \Sigma_2$, and $\sigma_3, \sigma_4, \sigma_5 \in \Sigma_3$; thus $\mathbf{G}_1$ and $\mathbf{G}_2$ share event σ_2, and $\mathbf{G}_2$ and $\mathbf{G}_3$ share event σ_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.

$$L_m(\mathbf{LOC}) := \left(\bigcap_{\alpha\in\Sigma_{\mathrm{for}}} P_\alpha^{-1} L_m(\mathbf{LOC}^P_\alpha)\right) \cap \left(\bigcap_{\beta\in\Sigma_{\mathrm{hib}}} P_\beta^{-1} L_m(\mathbf{LOC}^C_\beta)\right) \tag{9}$$

such that **LOC** is *control equivalent* to **SUP** (with respect to **G**) in the following sense:

$$L(\mathbf{G}) \cap L(\mathbf{LOC}) = L(\mathbf{SUP}),$$
$$L_m(\mathbf{G}) \cap L_m(\mathbf{LOC}) = L_m(\mathbf{SUP}).$$

Using a set of local preemptors and local controllers that is control equivalent to **SUP**, we can build an optimal and nonblocking *distributed control architecture* for a multi-agent TDES plant. Let the plant **G** with event set Σ be composed[6] of n component TDES (or agents) $\mathbf{G}_k$ with Σ_k ($k \in [1, n]$).[7] According to (4), $\Sigma_k = \Sigma_{act,k} \dot\cup \{tick\}$ (event *tick* is shared by all agents); thus $\Sigma = \bigcup_{k=1}^n \Sigma_{act,k} \dot\cup \{tick\}$. In addition to *tick*, we also allow the $\Sigma_{act,k}$ to share events. Now let $\Sigma_{\mathrm{for},k}$, $\Sigma_{\mathrm{hib},k} \subseteq \Sigma_k$ be the forcible event set and prohibitible event set, respectively, of agent $\mathbf{G}_k$; then $\Sigma_{\mathrm{for}} = \bigcup_{k=1}^n \Sigma_{\mathrm{for},k}$ and $\Sigma_{\mathrm{hib}} = \bigcup_{k=1}^n \Sigma_{\mathrm{hib},k}$. For each forcible event $\alpha \in \Sigma_{\mathrm{for}}$ there is a local preemptor $\mathbf{LOC}^P_\alpha$, and for each prohibitible event $\beta \in \Sigma_{\mathrm{hib}}$ there is a local controller $\mathbf{LOC}^C_\beta$. These local preemptors/controllers need to be allocated among individual agents, for each agent may have multiple forcible/prohibitible events. A convenient allocation is to let each local controller/preemptor be owned by exactly one agent; an example is displayed in Fig. 2. 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 Cai and Wonham (2010a). In particular, localizing the supervisor's preemption of event *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 $\mathbf{G} = (Q, \Sigma, \delta, q_0, Q_m)$ (as in (4)) and a corresponding monolithic supervisor $\mathbf{SUP} = (X, \Sigma, \xi, x_0, X_m)$ with respect to an imposed specification, we present the localization of **SUP**'s preemptive and disabling action in the sequel.

4.1. Localization of preemptive action

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

$$E_{tick}(x) = 1 \quad \text{iff} \quad \xi(x, tick)!. \tag{10}$$

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

$$\xi(x, \alpha)! \quad \text{and} \quad \neg\xi(x, tick)! \quad \text{and} \quad (\exists s \in \Sigma^*) \Big(\xi(x_0, s) = x \text{ and } \delta(q_0, s.tick)!\Big). \tag{11}$$

So $F_\alpha(x) = 1$ means that forcible event α is defined at state x (i.e. $\xi(x, \alpha)!$), which effectively preempts the occurrence of event *tick* (i.e. *tick* is not defined at x in **SUP** but is defined at some state in the plant **G** corresponding to x via string s). It should be noted that at state x, α need not be the only forcible event that preempts *tick*, for there can be other forcible events, say α', defined at x. In that case, by (11) $F_{\alpha'}(x) = 1$ as well.

Based on the preemption information captured by E_{tick} and F_α above, we define the following binary relation $\mathcal{R}^P_\alpha$ (for α) on X, called 'preemption consistency'. This relation determines if two states of **SUP** have consistent preemptive action with respect to the forcible event α.

Definition 1. Let $\mathcal{R}^P_\alpha \subseteq X \times X$. We say that $\mathcal{R}^P_\alpha$ is a *preemption consistency relation* with respect to $\alpha \in \Sigma_{\mathrm{for}}$ if for every $x, x' \in X$, $(x, x') \in \mathcal{R}^P_\alpha$ iff

$$E_{tick}(x) \cdot F_\alpha(x') = 0 = E_{tick}(x') \cdot F_\alpha(x). \tag{12}$$

Thus a pair of states (x, x') in **SUP** is *not* preemption consistent with respect to α only when *tick* is defined at x but is preempted by α at x', or vice versa. Otherwise, x and x' are preemption consistent, i.e. $(x, x') \in \mathcal{R}^P_\alpha$. It is easily verified that $\mathcal{R}^P_\alpha$ is reflexive and symmetric, but not transitive; an illustration is provided in Fig. 3. Hence $\mathcal{R}^P_\alpha$ is not an equivalence relation. This fact leads to the following definition of a *preemption cover*. Recall that a *cover* on a set X is a family of nonempty subsets (or *cells*) of X whose union is X.

Definition 2. Let I be some index set, and $\mathcal{C}^P_\alpha = \{X_i \subseteq X | i \in I\}$ a cover on X. We say that $\mathcal{C}^P_\alpha$ is a *preemption cover* with respect to α if

(i) $(\forall i \in I, \forall x, x' \in X_i)(x, x') \in \mathcal{R}^P_\alpha$,

(ii) $(\forall i \in I, \forall \sigma \in \Sigma)\Big[(\exists x \in X_i)\xi(x, \sigma)!$
$\Rightarrow \big((\exists j \in I)(\forall x' \in X_i)\xi(x', \sigma)! \Rightarrow \xi(x', \sigma) \in X_j\big)\Big]$. (13)

[6] Composition of multiple TDES involves first taking *synchronous product* of the untimed DES and then unifying the time bounds of shared events (Brandin & Wonham, 1994; Wonham, 2012).

[7] Note that each $\mathbf{G}_k$ may contain multiple forcible and/or prohibitible events.

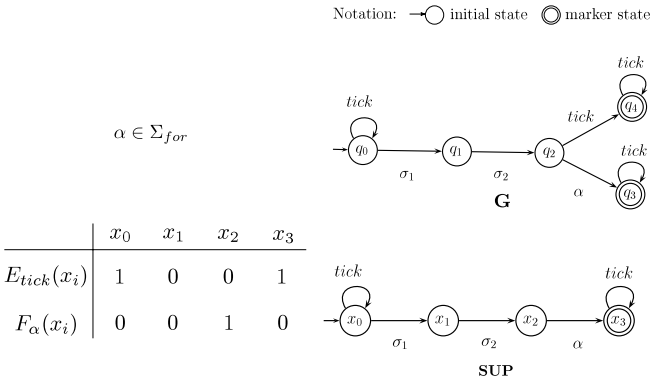


Fig. 3. Preemption consistency relation is not transitive: $(x_0, x_1) \in \mathcal{R}^P_\alpha$, $(x_1, x_2) \in \mathcal{R}^P_\alpha$, but $(x_0, x_2) \notin \mathcal{R}^P_\alpha$.

A preemption cover $\mathcal{C}^P_\alpha$ lumps states of **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 σ must be covered by the same cell X_j. Inductively, two states x, x' belong to a common cell of $\mathcal{C}^P_\alpha$ if and only if x and x' are preemption consistent, and two future states, say y and y', that can be reached respectively from x and x' by a given string are again preemption consistent. We say that a preemption cover $\mathcal{C}^P_\alpha$ is a *preemption congruence* if $\mathcal{C}^P_\alpha$ happens to be a *partition* on X, namely its cells are pairwise disjoint.

Having defined a preemption cover $\mathcal{C}^P_\alpha$ on X, we construct, below, a local preemptor $\mathbf{LOC}^P_\alpha = (Y_\alpha, \Sigma_\alpha, \zeta_\alpha, y_{0,\alpha}, Y_{m,\alpha})$ for the forcible event α to preempt *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 $\mathcal{C}^P_\alpha$. In particular, the initial state $y_{0,\alpha}$ is a cell X_{i0} where x_0 belongs, i.e. $x_0 \in X_{i0}$, and the marker state set $Y_{m,\alpha} := \{i \in I | X_i \cap X_m \neq \emptyset\}$.

(*Step* 2) For the event set Σ_α, define the transition function $\zeta'_\alpha : I \times \Sigma \to I$ over the entire event set Σ by $\zeta'_\alpha(i, \sigma) = j$ if

$$(\exists x \in X_i)\xi(x, \sigma) \in X_j \quad \text{and} \quad (\forall x' \in X_i)\big[\xi(x', \sigma)! \Rightarrow \xi(x', \sigma) \in X_j\big]. \tag{14}$$

Choose Σ_α to be the union of $\{\alpha, tick\}$ with other events which are *not* selfloop transitions of ζ'_α, i.e.

$$\Sigma_\alpha := \{\alpha, tick\} \dot\cup \{\sigma \in \Sigma - \{\alpha, tick\} \mid (\exists i, j \in I)\ i \neq j \text{ and } \zeta'_\alpha(i, \sigma) = j\}. \tag{15}$$

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

(*Step* 3) Define the transition function ζ_α to be the restriction of ζ'_α to Σ_α; namely $\zeta_\alpha : I \times \Sigma_\alpha \to I$ according to $\zeta_\alpha(i, \sigma) = \zeta'_\alpha(i, \sigma)$ for every $i \in I$ and $\sigma \in \Sigma_\alpha$.

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

By the same procedure, we generate a set of local preemptors $\mathbf{LOC}^P_\alpha$, one for each forcible event $\alpha \in \Sigma_{\text{for}}$. We will verify below that these generated preemptors 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 Cai and Wonham (2010a). Fix an arbitrary prohibitible event $\beta \in \Sigma_{\text{hib}}$. First define $E_\beta : X \to \{1, 0\}$ according to $E_\beta(x) = 1$ iff $\xi(x, \beta)!$. So $E_\beta(x) = 1$ means that β is defined at state x in **SUP**. Next define $D_\beta : X \to \{1, 0\}$ according to $D_\beta(x) = 1$ iff

$$\neg\xi(x, \beta)! \quad \text{and} \quad (\exists s \in \Sigma^*)\,(\xi(x_0, s) = x \text{ and } \delta(q_0, s\beta)!)\,. \tag{16}$$

Thus $D_\beta(x) = 1$ means that β must be disabled at x (i.e. β 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 $M(x) = 1$ 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

$$T(x) = 1 \quad \text{iff} \quad (\exists s \in \Sigma^*)\xi(x_0, s) = x \text{ and } \delta(q_0, s) \in Q_m.$$

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 $L_m(\mathbf{SUP}) \subseteq L_m(\mathbf{G})$ that $T(x) = 0 \Rightarrow M(x) = 0$ and $M(x) = 1 \Rightarrow T(x) = 1$ (Cai & Wonham, 2010a).

We now define the binary relation $\mathcal{R}^C_\beta \subseteq X \times X$, called *control consistency* with respect to prohibitible event β (cf. Cai & Wonham, 2010a), according to $(x, x') \in \mathcal{R}^C_\beta$ iff

$$\text{(i) } E_\beta(x) \cdot D_\beta(x') = 0 = E_\beta(x') \cdot D_\beta(x),$$
$$\text{(ii) } T(x) = T(x') \Rightarrow M(x) = M(x'). \tag{17}$$

Thus a pair of states (x, x') in **SUP** satisfies $(x, x') \in \mathcal{R}^C_\beta$ if (i) event β is defined at one state, but not disabled at the other, and (ii) x and x' are both marked or both unmarked in **SUP**, provided both are marked or unmarked in **G**. It is easily verified that $\mathcal{R}^C_\beta$ is generally not transitive (Cai & Wonham, 2010a), thus not an equivalence relation. Now let I be some index set, and $\mathcal{C}^C_\beta = \{X_i \subseteq X | i \in I\}$ a cover on X. Similar to Definition 2, we define $\mathcal{C}^C_\beta$ to be a *control cover* with respect to β if

$$\text{(i) } (\forall i \in I, \forall x, x' \in X_i)(x, x') \in \mathcal{R}^C_\beta,$$
$$\text{(ii) } (\forall i \in I, \forall \sigma \in \Sigma)\Big[(\exists x \in X_i)\xi(x, \sigma)! \Rightarrow \big((\exists j \in I)(\forall x' \in X_i)\xi(x', \sigma)! \Rightarrow \xi(x', \sigma) \in X_j\big)\Big]. \tag{18}$$

Note that the only difference between control cover and preemption cover in Definition 2 is the binary relation (control consistency $\mathcal{R}^C_\beta$ or preemption consistency $\mathcal{R}^P_\alpha$) used in condition (i).

With the control cover $\mathcal{C}^C_\beta$ on X, we construct by the same steps (*Step* 1)–(*Step* 3), above, a local controller $\mathbf{LOC}^C_\beta = (Y_\beta, \Sigma_\beta, \zeta_\beta, y_{0,\beta}, Y_{m,\beta})$ for prohibitible event β. Here, the choice of event set Σ_β is (cf. (15))

$$\Sigma_\beta := \{\beta\} \dot\cup \{\sigma \in \Sigma - \{\beta\} \mid (\exists i, j \in I)\ i \neq j \text{ and } \zeta'_\beta(i, \sigma) = j\}. \tag{19}$$

Σ_β need not contain event *tick*, as noted in Footnote 5. In the same way, we generate a set of local controllers $\mathbf{LOC}^C_\beta$, one for each prohibitible event $\beta \in \Sigma_{\text{hib}}$. We will verify that the collective disabling action of these local controllers is identical to that of **SUP**.

4.3. Main result

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

Theorem 3. *The set of local preemptors* $\{\mathbf{LOC}^P_\alpha | \alpha \in \Sigma_{\text{for}}\}$ *and the set of local controllers* $\{\mathbf{LOC}^C_\beta | \beta \in \Sigma_{\text{hib}}\}$ *constructed above solve the Supervisor Localization Problem; that is,*

$$L(\mathbf{G}) \cap L(\mathbf{LOC}) = L(\mathbf{SUP}), \tag{20}$$

$$L_m(\mathbf{G}) \cap L_m(\mathbf{LOC}) = L_m(\mathbf{SUP}). \tag{21}$$

where $L(\mathbf{LOC})$ *and* $L_m(\mathbf{LOC})$ *are as defined in* (8) *and* (9)*, respectively.*

Since for every preemption cover (resp. control cover), the presented procedure constructs a local preemptor (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 (Su & Wonham, 2004). In Cai and Wonham (2010a) we proposed, nevertheless, a polynomial-time localization 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 Cai & Wonham, 2010a, 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 Cai and Wonham (2010a) and Cai and Wonham (2010b) 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. Eq. (21) and the ($\supseteq$) direction of (20) may be verified analogously as in Cai and Wonham (2010a). Here we prove ($\subseteq$) in (20), which involves the TDES controllability definition, preemption consistency, and control consistency.

Proof of Theorem 3 ($\subseteq$, (20))**.** We show this by induction. First, the empty string ϵ belongs to $L(\mathbf{G})$, $L(\mathbf{LOC})$, and $L(\mathbf{SUP})$, because these languages are all nonempty. Next, suppose $s \in L(\mathbf{G}) \cap L(\mathbf{LOC})$, $s \in L(\mathbf{SUP})$, and $s\sigma \in L(\mathbf{G}) \cap L(\mathbf{LOC})$ for an arbitrary event $\sigma \in \Sigma$. It will be proved that $s\sigma \in L(\mathbf{SUP})$. Since $\Sigma = \Sigma_u \dot\cup \Sigma_c = \Sigma_u \dot\cup \{tick\} \dot\cup \Sigma_{\text{hib}}$, we consider the following three cases.

(i) Let $\sigma \in \Sigma_u$. Since $L_m(\mathbf{SUP})$ is controllable (see (6)), and $s\sigma \in L(\mathbf{G})$ (i.e. $\sigma \in \text{Elig}_{\mathbf{G}}(s)$), we have $\sigma \in \text{Elig}_{L_m(\mathbf{SUP})}(s)$. That is, $s\sigma \in \bar{L}_m(\mathbf{SUP}) = L(\mathbf{SUP})$.

(ii) Let $\sigma = tick$. We will show $tick \in \text{Elig}_{L_m(\mathbf{SUP})}(s)$ to conclude that $s.tick \in \bar{L}_m(\mathbf{SUP}) = L(\mathbf{SUP})$. By the hypothesis that $s, s.tick \in L(\mathbf{LOC})$ and Eq. (8), for every forcible event $\alpha \in \Sigma_{\text{for}}$ there holds $s, s.tick \in P^{-1}_\alpha L(\mathbf{LOC}^P_\alpha)$, i.e. $P_\alpha(s), P_\alpha(s)\, tick \in L(\mathbf{LOC}^P_\alpha)$. Recall $\mathbf{LOC}^P_\alpha = (Y_\alpha, \Sigma_\alpha, \zeta_\alpha, y_{0,\alpha}, Y_{m,\alpha})$, and let $i := \zeta_\alpha(y_{0,\alpha}, P_\alpha(s))$ and $j := \zeta_\alpha(i, tick)$. By definition of ζ'_α in (14), any $\sigma \notin \Sigma_\alpha$ (defined in (15)) is only a selfloop transition of ζ'_α; hence $\zeta'_\alpha(y_{0,\alpha}, s) = i$. By (14) again, there exist $x, x' \in X_i$ and $x'' \in X_j$ such that $\xi(x_0, s) = x$ and $\xi(x', tick) = x''$ in **SUP**.

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. $(x, x') \in \mathcal{R}^P_\alpha$. Since $\xi(x', tick)!$, by (10) we have $E_{tick}(x') = 1$. Thus the requirement $E_{tick}(x') \cdot F_\alpha(x) = 0$ (Definition 1) yields that $F_\alpha(x) = 0$. The latter, by (11), gives rise to the following three cases: (Case 1) $\neg\xi(x, \alpha)!$, (Case 2) $\xi(x, tick)!$, or (Case 3) $(\neg\exists s \in \Sigma^*)\big(\xi(x_0, s) = x \text{ and } \delta(q_0, s.tick)!\big)$. First, Case 3 is impossible, because by the hypothesis that $s \in L(\mathbf{SUP})$ and $s.tick \in L(\mathbf{G})$ we have $\xi(x_0, s)!$ and $\delta(q_0, s.tick)!$. Next, Case 2 means directly $tick \in \text{Elig}_{L_m(\mathbf{SUP})}(s)$. Finally, Case 1 implies $\alpha \notin \text{Elig}_{L_m(\mathbf{SUP})}(s)$; note that this holds for all $\alpha \in \Sigma_{\text{for}}$. Hence $\text{Elig}_{L_m(\mathbf{SUP})}(s) \cap \Sigma_{\text{for}} = \emptyset$. Then by the fact that **SUP** is controllable, we derive from (6) that $tick \in \text{Elig}_{L_m(\mathbf{SUP})}(s)$.

(iii) Let $\sigma \in \Sigma_{\text{hib}}$. By the hypothesis $s, s\sigma \in L(\mathbf{LOC})$ and Eq. (8), we have $s, s\sigma \in P^{-1}_\sigma L(\mathbf{LOC}^C_\sigma)$, i.e. $P_\sigma(s), P_\sigma(s)\sigma \in L(\mathbf{LOC}^C_\sigma)$. As in (ii), let $i := \zeta_\sigma(y_{0,\sigma}, P_\sigma(s)) = \zeta'_\sigma(y_{0,\sigma}, s)$ and $j := \zeta_\sigma(i, \sigma)$. By the definition of ζ'_σ in (14), there exist $x, x' \in X_i$, $x'' \in X_j$ such that $\xi(x_0, s) = x$ and $\xi(x', \sigma) = x''$. Since x, x' belong to the same cell X_i, by the control cover definition x and x' must be control consistent, i.e. $(x, x') \in \mathcal{R}^C_\sigma$. That $\xi(x', \sigma)!$ implies that $E_\sigma(x') = 1$. Thus the requirement $E_\sigma(x') \cdot D_\sigma(x) = 0$ yields that $D_\sigma(x) = 0$. The latter, by (16), gives rise to the following two cases: (Case 1) $\xi(x, \sigma)!$, or (Case 2) $(\neg\exists s \in \Sigma^*)\xi(x_0, s) = x$ and $\delta(q_0, s\sigma)!$. Case 2 is impossible, because by the hypothesis that $s \in L(\mathbf{SUP})$ and $s\sigma \in L(\mathbf{G})$ we have $\xi(x_0, s)!$ and $\delta(q_0, s\sigma)!$. But in Case 1, $\xi(x, \sigma)!$ i.e. $s\sigma \in L(\mathbf{SUP})$. □

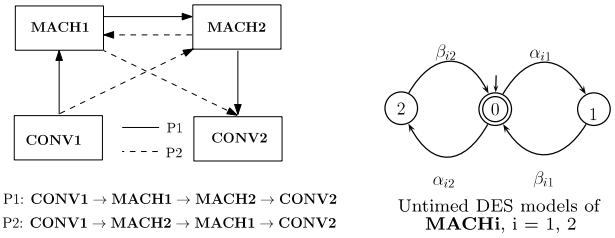


Fig. 4. Manufacturing cell.

5. Case study: manufacturing cell

We illustrate supervisor localization in TDES by studying a manufacturing cell example, taken from Brandin and Wonham (1994), Wonham (2012, Section 9.11). As displayed in Fig. 4, 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. 4. The untimed DES models of the machines are also displayed in Fig. 4; here α_{ij} ($i, j \in [1, 2]$) is the event "**MACHi** starts to work on a Pj-part", while β_{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 :

$(\alpha_{11}, 1, \infty)$ $(\beta_{11}, 3, 3)$ $(\alpha_{12}, 1, \infty)$ $(\beta_{12}, 2, 2)$

MACH2's timed events :

$(\alpha_{21}, 1, \infty)$ $(\beta_{21}, 1, 1)$ $(\alpha_{22}, 1, \infty)$ $(\beta_{22}, 4, 4)$.

So α_{ij} are remote events (upper bound ∞) and β_{ij} prospective events (finite upper bounds). Now the TDES models of the two machines can be generated (Wonham, 2012, p. 425). Their joint behavior is the synchronous product of the two TDES, which is the plant to be controlled.

To impose behavioral constraints on the two machines' joint behavior, we take the events α_{ij} to be both prohibitible and forcible, i.e. $\Sigma_{\text{hib}} = \Sigma_{\text{for}} = \{\alpha_{ij} | i, j = 1, 2\}$, and the β_{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

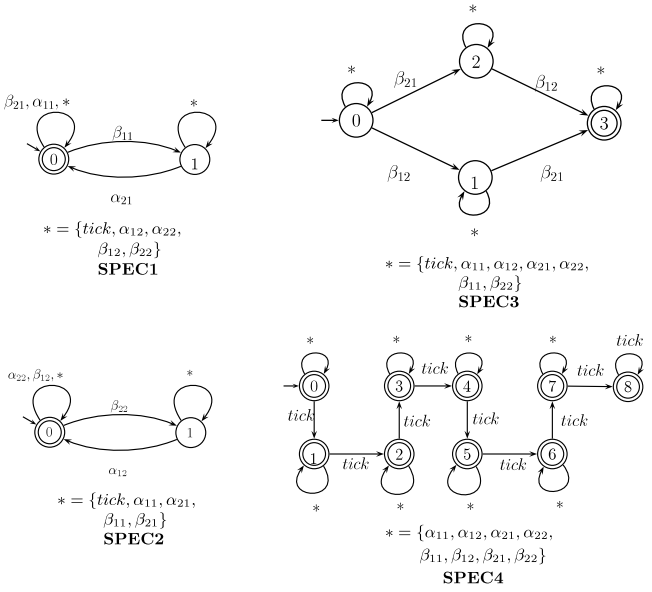


Fig. 5. Control specifications: logical and temporal. The marked state 3 of **SPEC3** corresponds to the completion of a production cycle: one P1-part and one P2-part are processed by both machines.

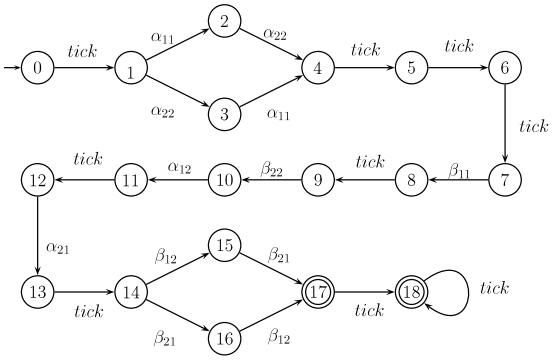


Fig. 6. Monolithic optimal and nonblocking supervisor **SUP**.

cycle. (S4) A production cycle must be completed in at most 8 time units.[8]

These four specifications are formalized as automata **SPEC1**, **SPEC2**, **SPEC3**, and **SPEC4**, respectively, as displayed in Fig. 5. 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 Brandin & Wonham, 1994, Wonham, 2012).

Now the plant to be controlled is the synchronous product of TDES **MACH1** and **MACH2** (Wonham, 2012, p. 425), and the overall control specification is the synchronous product of automata **SPEC1**–**SPEC4** in Fig. 5. We compute as in (7) the corresponding monolithic optimal and nonblocking supervisor **SUP**; the computation is done by the *supcon* command in XPTTCT (Wonham, 2008). **SUP** has 19 states and 21 transitions, as displayed in Fig. 6. 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. Each event is executed exactly once, and each forcible event preempts *tick* immediately after it becomes eligible.

We now apply supervisor localization to decompose the monolithic supervisor **SUP** into local preemptors and local controllers, respectively for each forcible event and each prohibitible event. Specifically, since $\Sigma_{\text{hib}} = \Sigma_{\text{for}} = \{\alpha_{ij} | i, j = 1, 2\}$, we will compute a local preemptor and a local controller for each α_{ij}, responsible for α_{ij}'s *tick*-preemptive action and its disabling action, respectively. This computation can be done by an algorithm adapted from Cai and Wonham (2010a) (as discussed in Section 4.3); here, however, owing to the simple (chain-like) structure of **SUP** (Fig. 6), local preemptors/controllers can be derived by inspection. We demonstrate such a derivation below, which results in a local preemptor $\mathbf{LOC}^P_{\alpha_{11}}$ for the forcible (and prohibitible) event α_{11}. Other derivations of local preemptors/controllers are similar.

To derive a local preemptor $\mathbf{LOC}^P_{\alpha_{11}}$ for event α_{11}, we find a preemption cover $\mathcal{C}^P_{\alpha_{11}}$ for α_{11} on **SUP**'s state set as follows. Initialize $\mathcal{C}^P_{\alpha_{11}}$ to be $\mathcal{C}^P_{\alpha_{11}} = \big\{[0], [1], [2], \ldots, [18]\big\}$, 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 $\mathcal{C}^P_{\alpha_{11}}$ to be a preemption cover.

(i) Cells [0] and [1] cannot be merged. Since $E_{tick}(0) = 1$ (event *tick* is defined at state 0) and $F_{\alpha_{11}}(1) = 1$ (*tick* is preempted by α_{11} at state 1), the pair of states (0, 1) is not preemption consistent, i.e. $(0, 1) \notin \mathcal{R}^P_{\alpha_{11}}$. 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 $F_{\alpha_{11}}(2) = 0, E_{tick}(2) = 0$ (*tick* is preempted at state 2, but by α_{22} not by α_{11}) and $E_{tick}(4) = 1, F_{\alpha_{11}}(4) = 0$ (event *tick* is defined at state 4). Thus $(2, 4) \in \mathcal{R}^P_{\alpha_{11}}$, 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 α_{11} at both states 1 and 3) and $E_{tick}(1) = E_{tick}(3) = 0$. Thus $(1, 3) \in \mathcal{R}^P_{\alpha_{11}}$, which satisfies requirement (i) of preemption cover. Now event α_{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 $\mathcal{C}^P_{\alpha_{11}} = \big\{[0], [1, 3], [2, 4], [5], \ldots, [18]\big\}$.

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

Having found the preemption cover $\mathcal{C}^P_{\alpha_{11}}$, we apply (*Step* 1)–(*Step* 3) in Section 4.1 to construct a local preemptor $\mathbf{LOC}^P_{\alpha_{11}}$, with transition structure displayed in Fig. 7. Note that the event set of $\mathbf{LOC}^P_{\alpha_{11}}$ is exactly $\{\alpha_{11}, tick\}$, which means that $\mathbf{LOC}^P_{\alpha_{11}}$ 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. 7 and 8. Here, for example, the event set of $\mathbf{LOC}^P_{\alpha_{12}}$ is $\{\alpha_{12}, tick, \beta_{22}\}$; so event β_{22} originating in **MACH2** has to be observed by $\mathbf{LOC}^P_{\alpha_{12}}$. We have then verified that their joint behavior (via synchronous product) is identical to the

[8] We choose "8 time units" because it is, according to Brandin and Wonham (1994) and Wonham (2012), the minimal time to complete one production cycle. Thus this temporal specification is a time-minimization requirement.

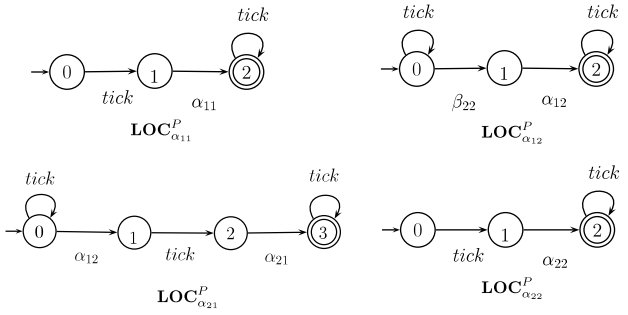


Fig. 7. Local preemptors for individual forcible events. The alphabet of each local preemptor is the set of events displayed in each automaton.

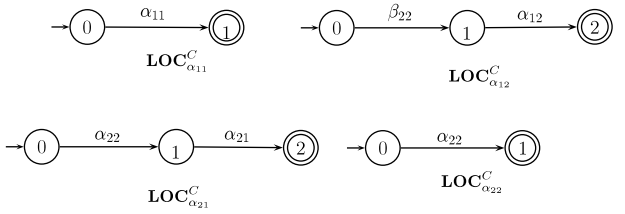


Fig. 8. Local controllers for individual prohibitible events. The alphabet of each local controller is the set of events displayed in each automaton.

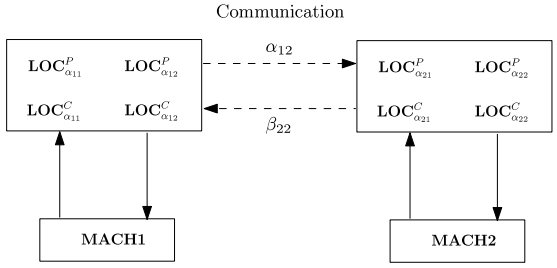


Fig. 9. Distributed control architecture.

monolithic optimal and nonblocking behavior of **SUP**, i.e. (20) and (21) 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 $\mathbf{LOC}^P_{\alpha_{11}}$ (resp. $\mathbf{LOC}^P_{\alpha_{22}}$) in Fig. 7 means that after one *tick*, forcible event α_{11} preempts event *tick* and **MACH1** starts to work on a P1-part (resp. α_{22} preempts *tick* and **MACH2** works on a P2-part). This is possible because α_{11} (resp. α_{22}) has lower time bound 1 and becomes eligible to occur after one *tick*. For another example, the local preemptor $\mathbf{LOC}^P_{\alpha_{21}}$ in Fig. 7 specifies that after the occurrence of α_{12} followed by a *tick*, forcible event α_{21} preempts *tick* and **MACH2** starts to work on a P1-part. This preemption is due to the fact that α_{21} has lower time bound 1 and becomes eligible to occur after the occurrence of β_{22} plus one *tick* (according to Fig. 6 event α_{22} first occurs in **MACH2**, which implies from the untimed model in Fig. 4 the event order $\alpha_{22} \cdot \beta_{22} \cdot \alpha_{21}$). But the occurrence of α_{12} implies that β_{22} has just occurred (see Fig. 6).

Finally, with the derived set of local preemptors and controllers, we build a distributed control architecture for this manufacturing cell; see Fig. 9. Each machine acquires those local preemptors/controllers wrt. its own distinct forcible/prohibitible events, thereby being capable of executing local preemptive/disabling actions. For these local actions to jointly achieve the same controlled behavior as the monolithic supervisor does, communicating the 'critical' events α_{12} and β_{22} between the two machines is essential. The critical events are obtained by intersecting the alphabet of one machine and the alphabets of local preemptors/controllers of the other.

6. Conclusions

We have established supervisor localization in the Brandin–Wonham timed DES framework. Under this localization scheme, each individual agent disables its own prohibitible events and preempts event *tick* via its own forcible events; overall, these local control actions collectively achieve monolithic optimal and nonblocking supervision. In future research, we aim to combine the localization approach with an effective modular supervisor synthesis to address large-scale real-time DES.

References

Brandin, B., & Wonham, W. M. (1994). Supervisory control of timed discrete-event systems. *IEEE Transactions on Automatic Control*, *39*(2), 329–342.

Brave, Y., & Heymann, M. (1988). Formulation and control of real time discrete event processes. In *Proc. 27th IEEE conf. on decision and control* (pp. 1131–1132). Austin, TX.

Cai, K., & Wonham, W. M. (2010a). Supervisor localization: a top-down approach to distributed control of discrete-event systems. *IEEE Transactions on Automatic Control*, *55*(3), 605–618.

Cai, K., & Wonham, W. M. (2010b). Supervisor localization for large discrete-event systems: case study production cell. *International Journal of Advanced Manufacturing Technology*, *50*(9–12), 1189–1202.

Cofer, D. D., & Garg, V. K. (1996). Supervisory control of real-time discrete-event systems using lattice theory. *IEEE Transactions on Automatic Control*, *41*(2), 199–209.

Golaszewski, C. H., & Ramadge, P. J. (1987). Control of discrete event processes with forced events. In *Proc. 26th IEEE conf. on decision and control* (pp. 247–251).

Leung, J., Lee, I., & Son, S. (Eds.) (2007). *Handbook of real-time and embedded systems*. Chapman & Hall/CRC.

Ostroff, J. S. (1990). Deciding properties of timed transition models. *IEEE Transactions on Parallel and Distributed Systems*, *1*(2), 170–183.

Ramadge, P. J., & Wonham, W. M. (1987). Supervisory control of a class of discrete event process. *SIAM Journal on Control and Optimization*, *25*(1), 206–230.

Su, R., & Wonham, W. M. (2004). Supervisor reduction for discrete-event systems. *Discrete Event Dynamic Systems*, *14*(1), 31–53.

Wonham, W. M. (2008). *Design software: XPTTCT*. System Control Group, ECE Dept, University of Toronto, Available at: http://www.control.utoronto.ca/DES.

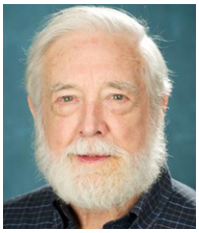
Wonham, W. M. (2012). *Supervisory control of discrete-event systems*. Systems Control Group, ECE Dept, University of Toronto, Available at: http://www.control.utoronto.ca/DES.

Renyuan Zhang received the B.Eng. degree in Electrical Engineering from Xi'an Jiaotong University, Xi'an, Shaanxi, China, in 2007, and is currently pursuing the Ph.D. degree in Electrical Engineering from Xi'an Jiaotong University. From Sept. 2011 to Dec. 2012, he studied in Department of Electrical and Computer Engineering in University of Toronto. His research interest is investigating delay-robust property in distributed control of untimed discrete-event systems and timed discrete-event systems.

Kai Cai received the B. Eng. degree in Electrical Engineering from Zhejiang University (China) in 2006, the M.A.Sc. degree in Electrical and Computer Engineering from the University of Toronto (Canada) in 2008, and the Ph.D. degree in Systems Science from Tokyo Institute of Technology (Japan) in 2011. He is currently a Postdoctoral Fellow in Systems Control, with the Department of Electrical and Computer Engineering of the University of Toronto. His research interests are distributed control of multi-agent systems and distributed control of discrete-event systems.

Yongmei Gan received the B.S. and M.S. degrees from Xi'an Technology University, Xi'an, China, in 1993 and 1996, respectively, and the Ph.D. degree from Northwestern Polytechnical University in 1999. Since 2000, she has been with the school of Electrical Engineering, Xi'an Jiaotong University, where she is currently an Associate Professor. She has authored or coauthored more than 60 technical papers in modeling and simulation of control systems, robust control, and supervisory control of discrete-event systems.

Zhaoan Wang received the B.S. and M.S. degrees from Xi'an Jiaotong University, Xi'an, China, in 1970 and 1982, respectively, and the Ph.D. degree from Osaka University, Osaka, Japan, in 1989. Since 1982, he has been with Xi'an Jiaotong University, where he is currently a Professor. He has been active in industrial and government consulting, and in university research on power systems and control theory, and has published more than 150 technical papers.



W. M. Wonham received the B. Eng. degree in engineering physics from McGill University in 1956, and the Ph.D. in control engineering from the University of Cambridge (UK) in 1961. From 1961 to 1969 he was associated with several US research groups in control. Since 1970 he has been a faculty member in Systems Control, with the Department of Electrical and Computer Engineering of the University of Toronto. Wonham's research interests have included stochastic control and filtering, geometric multivariable control, and discrete-event systems.