Distributed supervisory control of discrete-event systems with communication delay

Renyuan Zhang, Kai Cai, Yongmei Gan, and W. M. Wonham

Noname manuscript No.
(will be inserted by the editor)

Distributed Supervisory Control of Discrete-Event Systems with Communication Delay

Renyuan Zhang · Kai Cai · Yongmei Gan · W.M. Wonham

the date of receipt and acceptance should be inserted later

Abstract This paper identifies a property of delay-robustness in distributed supervisory control of discrete-event systems (DES) with communication delays. In previous work a distributed supervisory control problem has been investigated on the assumption that inter-agent communications take place with negligible delay. From an applications viewpoint it is desirable to relax this constraint and identify communicating distributed controllers which are delay-robust, namely logically equivalent to their delay-free counterparts. For this we introduce inter-agent channels modeled as 2-state automata, compute the overall system behavior, and present an effective computational test for delay-robustness. From the test it typically results that the given delay-free distributed control is delay-robust with respect to certain communicated events, but not for all, thus distinguishing events which are not delay-critical from those that are. The approach is illustrated by a workcell model with three communicating agents.

Keywords Discrete-Event Systems · Distributed Supervisory Control · Communication Delay · Delay-robustness

1 Introduction

Distributed control is pervasive in engineering practice, either by geographical necessity or to circumvent the complexity of centralized (also called 'monolithic')

R. Zhang,
Dept. Traffic and Control Engineering, Northwestern Polytechnical University, China
E-mail: ryzhang@nwpu.edu.cn

K. Cai
Urban Research Plaza, Osaka City University, Japan
E-mail: kai.cai@info.eng.osaka-cu.ac.jp

Y. Gan
School of Electrical Engineering, Xi'an Jiaotong University
E-mail: ymgan@mail.xjtu.edu.cn

W.M. Wonham
System Control Group, Dept. Electrical and Computer Engineering, University of Toronto
E-mail: wonham@control.utoronto.ca

control. Existing work on distributed supervisory control of discrete-event systems (DES) has focused on synthesis of local controllers for individual agents (plant components) such that the resulting controlled behavior is identical with that achieved by global supervision [1–6]. In these contributions, it is assumed that agents make independent observations and decisions, with instantaneous inter-agent communication. While simplifying the design of distributed control, this assumption may be unrealistic in practice, where controllers are linked by a physical network subject to delays. Hence, to model and appraise these delays is essential for the correct implementation of control strategies.

The communication problem in distributed control of multi-agent DES has been discussed by several researchers. Kalyon et al. [7] propose a framework for the control of distributed systems modeled as communicating finite state machines with reliable unbounded FIFO channels. They formulate a distributed state avoidance control problem, and show that the existence of a solution for the problem is undecidable. Lin [8] investigates supervisory control of networked discrete-event systems which features communication delays and data losses in observation and control. He assumes that the communication between a supervisor and the plant is via a shared network and communication delays are bounded. Darondeau and Ricker [9] propose to synthesize distributed control starting from a monolithic supervisor (in the DES sense) which can be represented as a distributed Petri net; local nets are linked by message passing to effect token transfer required by transitions joining places that have been distributed to distinct locations. PN distributability is admitted somewhat to constrain generality; but the exact relation of this approach to our own remains open to future research.

Research on communication problems in decentralized/modular supervisory control has also been reported in recent years. Taking delays into consideration, Yeddes et al. [10] propose a 3-state data transmission model, representing delays by timed events with lower and finite upper time bounds; these events are incorporated into the plant and specification automata, and the time bounds further restricted by a supervisor synthesis procedure; maximal permissiveness and nonblocking, however, are not guaranteed. In [11] Barrett and Lafortune propose an information structure model for analysis and synthesis of decentralized supervisory control, applicable in principle to the case of communication delays, but they assume that such delays are absent. For a limited class of specifications, Tripakis [12] formulates certain problems in decentralized control with bounded or unbounded communication delay, modeling the system with communication by automata with state output map. In this model the existence of controllers in case of unbounded delay is undecidable. In our paper, by contrast, we address the question: does a given controller have the property of delay-robustness (as we define it) or not? This question is indeed decidable, and we provide an effective test to answer it. Schmidt et al. [13] consider a heterarchical (hierarchical/decentralized) architecture requiring communication of shared events among modules of the hierarchy. A communication model is developed in which delay may affect system operation unless suitable transmission deadlines are met. If so, correct operation of the distributed supervisors is achieved if the network is sufficiently fast. In [14] correct heterarchical operation is achieved subject to a condition of "communication consistency", by which the occurrence of low-level events is restricted by the feasibility of high-level events. Xu and Kumar [15] consider monolithic supervisory control with bounded communication delay d (measured by event count) between plant

and controller; a condition is derived for equality of controlled behaviors under delay d or with zero delay respectively; verification is exponential in d. Hiraishi [16] proposes an automaton formalism for communication with delay in decentralized control, and concludes semi-decidability of the controller design problem in the case of k-bounded delay and in case an observability condition holds for state-transition cycles. Ricker and Caillaud [17] consider decentralized control (with *a priori* given individual observable event subsets) in the case where co-observability fails and therefore inter-supervisor communication is needed for correct global supervision. The issue is when, what, and to whom a given local supervisor should communicate; a solution is proposed to the protocol design problem. In our paper this question does not arise because, with supervisor localization, we already declare who communicates what to whom, and the problem is then to analyze our existing ideal (instantaneous) communication scheme to see if it is still correct in the presence of delay.

Thus we consider distributed control with separately modeled communication channels having unknown unbounded delay, imposed on an existing distributed architecture known to be optimal and nonblocking for zero delay. In this paper and its conference precursor [18], we start from the DES distributed control scheme called 'supervisor localization' reported in [5,6], which describes a systematic top-down approach to design distributed controllers which collectively achieve global optimal and nonblocking supervision. Briefly, we first synthesize a monolithic supervisor, or alternatively a set of decentralized supervisors, assuming zero delay; then we apply supervisor localization to decompose each synthesized supervisor into local controllers for individual plant components, in this process determining the set of events that need to be communicated. Next, and central to the present paper, we propose a channel model for event communication, and design a test to verify for which events the system is delay-robust (as we define it below).

The initial control problem is the standard 'Ramadge-Wonham' (RW) problem [19–21]. Here the plant (DES to be controlled) is modeled as the synchronous product of several DES agents (plant components), say $\mathbf{AGENT}_1$, $\mathbf{AGENT}_2$, ..., that are independent, in the sense that their alphabets Σ_1, Σ_2, ..., are pairwise disjoint. In a logical sense these agents are linked by specifications $\mathbf{SPEC}_1$, $\mathbf{SPEC}_2$, ..., each of which (typically) restricts the behavior of an appropriate subset of the $\mathbf{AGENT}_i$ and is therefore modeled over the union of the corresponding subfamily of the Σ_i. For each $\mathbf{SPEC}_j$, a 'decentralized' supervisory controller $\mathbf{SUP}_j$ is computed in the same way as for a 'monolithic' supervisor [19]; it guarantees optimal (i.e. maximally permissive) and nonblocking behavior of the relevant subfamily (the 'control scope' of $\mathbf{SPEC}_j$) of the $\mathbf{AGENT}_i$. In general it will turn out that the synchronous product of all the $\mathbf{SUP}_j$ is blocking (e.g. may cause deadlock in the overall controlled behavior); in that case one or more additional 'coordinators' must be adjoined to suitably restrict the decentralized controlled behavior (see [6] for an example). Techniques for coordinator design are available in the literature (e.g. [22–25]) and in this paper we take them for granted. On achieving satisfactory decentralized control we finally 'localize' each decentralized supervisor, including the coordinator(s), if any, to the agents that fall within its control scope; the algorithm that achieves this is detailed in [5], and we shall refer to it as *Localize*. The result of *Localize* is that each $\mathbf{AGENT}_i$ is equipped with local controllers, one for each of the $\mathbf{SPEC}_j$ whose scope it falls within; in that sense $\mathbf{AGENT}_i$ is now 'intelligent' and semi-autonomous, with controlled behav-

ior $\mathbf{SUPLOC}_i$, say, while the synchronous product behavior of all the $\mathbf{SUPLOC}_i$ is provably that of the monolithic supervisor for the RW problem we began with. Autonomy of the $\mathbf{SUPLOC}_i$ is qualified, in that normally the transition structure of each $\mathbf{SUPLOC}_i$ will include events from various other $\mathbf{AGENT}_k$ with $k \neq i$. The implementation of our distributed control therefore requires instantaneous communication by $\mathbf{AGENT}_k$ of 'communication' events (when they occur, in its private alphabet Σ_k) to $\mathbf{SUPLOC}_i$ so the latter can properly update its state. Think of a group of motorists maneuvering through a congested intersection without benefit of external traffic control, each instead depending solely on signals from (mostly) neighboring vehicles and on commonly accepted protocols. In our DES model each $\mathbf{SUPLOC}_i$ can disable only its private controllable events, in Σ_i, but the logic of disablement may well depend on observation of critical events from certain other $\mathbf{AGENT}_k$, as remarked above. It is clear that if these communications are subject to indefinite time delay, then control may become disrupted and the collective behavior logically unacceptable. Our first aim is to devise a test to distinguish the latter case from the 'benign' situation where delay is tolerable, in the sense that 'logical' behavior is unaffected, even though in some practical sense behavior might be degraded, for instance severely slowed down[1]. This investigation would provide practitioners with useful information to implement distributed supervisors by communication channels: 'fast' channels must be assigned for communication of 'delay-critical' events, while 'slow' channels suffice for 'delay-robust' events.

In Sect. 3, we introduce the model of our communication channel. As will be seen, there is an implicit constraint that a channeled event (i.e. a communication event transmitted by a channel with indefinite delay) can occur and be transmitted only when its channel is available. This is similar to the mechanism of "synchronous elastic circuits" or "latency insensitive systems" (e.g. [28]); see Remark 2 below for details. As a consequence, an uncontrollable channeled event may or may not be blocked by its channel, the former case being undesirable. Our second aim is to distinguish these two cases; when an uncontrollable event is indeed blocked, we discuss how long it can be delayed.

We proceed to a formal review of distributed control by supervisor localization on the assumption of instantaneous inter-agent communication. Then we introduce inter-agent communication with delay, modeled by a separate logical channel for each delayed communication event (i.e. channeled event). As our main result, both a definition and a computational test are provided for 'delay-robustness' of the channeled distributed system with respect to an arbitrary subset of communication events. In addition, we employ the standard algorithm for checking controllability to identify whether or not an uncontrollable channeled event is blocked by its channel. These issues are illustrated by a workcell model with three communicating agents. Finally we present conclusions and suggestions for future work.

[1] Similar issues are addressed in the literature on 'delay-insensitive' asynchronous networks; for the definition see [26] and for a useful summary [27].

2 Preliminaries

2.1 Notation

Following [21] we recall various standard concepts and notation. Consider a system $\mathbf{G}$ of n component DES $\mathbf{G}_i = (Q_i, \Sigma_i, \eta_i, q_{i0}, Q_{im})$, $i \in N := \{1, 2, ..., n\}$, where Q_i is the (finite) state set, Σ_i is the (finite) set of event labels, $\eta_i : Q_i \times \Sigma_i \to Q_i$ is the transition (partial) function, q_{i0} is the initial state, and $Q_{im} \subseteq Q$ is the set of marker states. Each event set Σ_i is partitioned as the disjoint union $\Sigma_i = \Sigma_{ic} \cup \Sigma_{iu}$ where Σ_{ic} (resp. Σ_{iu}) is the subset of controllable (resp. uncontrollable) events for $\mathbf{G}_i$; the full event set for $\mathbf{G}$ is the union $\Sigma = \cup\{\Sigma_i | i \in N\}$.

Let Σ_i^* denote the set of all finite strings of elements in Σ_i, including the empty string ϵ, and as usual extend the transition function η_i to $Q_i \times \Sigma_i^*$, by defining $\eta_i(q_i, \epsilon) = q_i$, $\eta_i(q_i, s\sigma) = \eta_i(\eta_i(q_i, s), \sigma)$ for all $q_i \in Q_i$, $s \in \Sigma_i^*$ and $\sigma \in \Sigma_i$. We write $\eta_i(q_{i0}, s)!$ to mean that $\eta_i(q_{i0}, s)$ is defined. The *prefix closure* of a language L over Σ^* is defined as $\overline{L} = \{s \in \Sigma^* | su \in L \text{ for some } u \in \Sigma^*\}$. The *closed behavior* and *marked behavior* of $\mathbf{G}_i$ are defined respectively by $L(\mathbf{G}_i) = \{s \in \Sigma_i^* | \eta_i(q_{i0}, s)!\}$ and $L_m(\mathbf{G}_i) = \{s \in L(\mathbf{G}_i) | \eta_i(q_{i0}, s) \in Q_{im}\}$.

As in [5, 6] we assume that the $\mathbf{G}_i$ are *a priori* independent, in the sense that their alphabets Σ_i are pairwise disjoint. The system $\mathbf{G}$ representing their combined behavior is defined to be their synchronous product $\mathbf{G} = (Q, \Sigma, \eta, q_0, Q_m) = Sync(\mathbf{G}_1, ..., \mathbf{G}_n)$[2]. The closed behavior and marked behavior of $\mathbf{G}$ are $L(\mathbf{G}) = ||\{L(\mathbf{G}_i) | i \in N\}$ and $L_m(\mathbf{G}) = ||\{L_m(\mathbf{G}_i) | i \in N\}$ where $||$ denotes synchronous product of languages. Assume each $\mathbf{G}_i$ is trim (i.e. reachable and coreachable); then by independence, $\mathbf{G}$ is trim, i.e., $\overline{L_m(\mathbf{G})} = L(\mathbf{G})$.

Let $\Sigma_o \subseteq \Sigma$ be a subset of events thought of as 'observable'. We refer the reader to [21] for the formal definition of natural projection $P : \Sigma^* \to \Sigma_o^*$, DES isomorphism, $\mathbf{G}$-controllability, and the supremal quasi-congruence relation. Simply stated, natural projection P on a string $s \in \Sigma^*$ erases all the occurrences of $\sigma \in \Sigma$ in s such that $\sigma \notin \Sigma_o$, namely $P\sigma = \epsilon$ (the empty string); P is implemented as $Project(\mathbf{G}, Null[\Sigma - \Sigma_o])$, which returns a (state-minimal) DES $\mathbf{PG}$ over Σ_o such that $L_m(\mathbf{PG}) = PL_m(\mathbf{G})$ and $L(\mathbf{PG}) = PL(\mathbf{G})$. Two DES are isomorphic if they are identical up to relabeling of states; $\mathbf{G}$-controllability is the property required for a sublanguage of $L_m(\mathbf{G})$ to be synthesizable by a supervisory controller; while projection modulo supremal quasi-congruence produces a (possibly nondeterministic) abstraction (reduced version) of a DES $\mathbf{G}$, denoted $Supqc(\mathbf{G}, Null[\Sigma - \Sigma_o])$, which preserves observable transitions and the 'observer' property [29,30]. As detailed in [21] these operations are available in a software implementation [31] and will be referred to here as needed.

2.2 Distributed Control without Communication Delay

Next we summarize the distributed control theory (assuming zero communication delay) reported in [5,6]. First suppose $\mathbf{G}$ is to be controlled to satisfy a specification language $L_m(\mathbf{SPEC}) \subseteq \Sigma^*$ represented by a DES $\mathbf{SPEC}$. Denote by $K \subseteq \Sigma^*$ the

[2] We may safely assume that the implementation *Sync* of synchronous product is always associative and commutative; for more on this technicality see [21], Sect. 3.3.

supremal controllable sublanguage of $L_m(\mathbf{G}) \cap L_m(\mathbf{SPEC})$(for details see [21]). Assume K is represented by the DES **SUP**, i.e. **SUP** has closed and marked behavior

$$L(\mathbf{SUP}) = \overline{K}, \quad L_m(\mathbf{SUP}) = K. \tag{1}$$

Since $\mathbf{G} = Sync(\mathbf{G}_1, ..., \mathbf{G}_n)$ is the synchronous product of independent components we seek to implement **SUP** in distributed fashion by 'localizing' **SUP** to each $\mathbf{G}_i$ as proposed in [5, 6]. For this we bring in a family of local controllers $\mathbf{LOC} = \{\mathbf{LOC}_i | i \in N\}$, one for each $\mathbf{G}_i$, and define $L(\mathbf{LOC}) = \|\{L(\mathbf{LOC}_i) | i \in N\}$ and $L_m(\mathbf{LOC}) = \|\{L_m(\mathbf{LOC}_i) | i \in N\}$. It is shown in [5, 6] that

$$L(\mathbf{G}) \cap L(\mathbf{LOC}) = L(\mathbf{SUP}) \tag{2a}$$

$$L_m(\mathbf{G}) \cap L_m(\mathbf{LOC}) = L_m(\mathbf{SUP}) \tag{2b}$$

Here, the supervisory action of **SUP** is fully distributed among the set of local controllers, each acting independently and asynchronously, except for being synchronized through 'communication' events. Generally, each local controller has a much smaller state set than **SUP** and a smaller event subset of Σ, containing just the events of its corresponding plant component, together with those communication events from other components that are essential to make correct control decisions. We remark that if the system and its supervisor are large scale, we first synthesize a set of decentralized supervisors to achieve global optimality and nonblocking, and then apply supervisor localization to decompose each decentralized supervisor in the set (as in [6]).

3 Distributed Control with Communication Delay

Cai and Wonham [5] discuss a boundary case of optimal distributed control that is *fully-localizable* where inter-agent communication is not needed, namely the alphabet of each local controller $\mathbf{LOC}_i$ is simply Σ_i, so that $\mathbf{LOC}_i$ observes only events in its own agent $\mathbf{G}_i$. In this case, no issue of delay will arise. The more general and usual case is that inter-agent communication is imperative.

For simplicity assume temporarily that the system **G** consists of two components $\mathbf{G}_1$ and $\mathbf{G}_2$, and let the monolithic supervisor **SUP** (in (1)) be given. By localization we compute local controllers $\mathbf{LOC}_1$ with event set $\Sigma_{\mathbf{LOC}_1}$ and $\mathbf{LOC}_2$ with event set $\Sigma_{\mathbf{LOC}_2}$; then the local controlled behaviors are represented by

$$\mathbf{SUP}_1 = Sync(\mathbf{G}_1, \mathbf{LOC}_1) \tag{3}$$

$$\mathbf{SUP}_2 = Sync(\mathbf{G}_2, \mathbf{LOC}_2). \tag{4}$$

Let $\mathbf{LOCSUP} = Sync(\mathbf{SUP}_1, \mathbf{SUP}_2)$. By the localization theory of [5, 6] we know that $L(\mathbf{LOCSUP}) = L(\mathbf{SUP})$ and $L_m(\mathbf{LOCSUP}) = L_m(\mathbf{SUP})$, namely, the synchronized behavior of $\mathbf{SUP}_1$ and $\mathbf{SUP}_2$ agrees with that of the monolithic control **SUP** (in (1)).

In the general localization theory (instantaneous) inter-agent communication is both possible and necessary, so the alphabet $\Sigma_{\mathbf{LOC}_1}$ of $\mathbf{LOC}_1$ (resp. $\Sigma_{\mathbf{LOC}_2}$ of $\mathbf{LOC}_2$) will include elements (*communication events*) from Σ_2 (resp. Σ_1) as well as events from its 'private' alphabet Σ_1 (resp. Σ_2). Let $\Sigma_{com,1}$ (resp. $\Sigma_{com,2}$) represent the set of communication events from Σ_2 (resp. Σ_1), i.e $\Sigma_{com,1} = \Sigma_{\mathbf{LOC}_1} - \Sigma_1$

(resp. $\Sigma_{com,2} = \Sigma_{\mathbf{LOC}_2} - \Sigma_2$); then the set of communication events in **LOCSUP** (i.e. **SUP**) is

$$\Sigma_{com} = \Sigma_{com,1} \cup \Sigma_{com,2}. \tag{5}$$

We say that a communication event in $\Sigma_{com,1}$ is *imported* from $\mathbf{G}_2$ by $\mathbf{LOC}_1$ (resp. $\Sigma_{com,2}$, $\mathbf{G}_1$ and $\mathbf{LOC}_2$).

Remark 1 For every state x of each controller $\mathbf{LOC}_i$ ($i \in N$), and each communication event σ in $\mathbf{LOC}_i$ but imported from some other component $\mathbf{G}_j$ ($j \neq i$), if σ is not defined at x, we add a σ-selfloop, i.e. transition (x, σ, x) to $\mathbf{LOC}_i$. Now, σ is defined at every state of $\mathbf{LOC}_i$. With this modification, the new local controllers $\mathbf{LOC}_i$ are also control equivalent to **SUP** (because $\mathbf{LOC}_i$ does not disable events σ from other components $\mathbf{G}_j$ and σ will be disabled by $\mathbf{LOC}_j$ if and only if it is disabled by **SUP**) and the definition of σ at every state of $\mathbf{LOC}_i$ is consistent with the assumption that $\mathbf{LOC}_i$ may receive σ after indefinite communication delay.

Next we model the way selected communication events are imported with indefinite time delay; we call such events *channeled events*. Let Σ_{ch} represent the set of channeled events; then $\Sigma_{ch} \subseteq \Sigma_{com}$ (Σ_{com} is defined in (5)). For example assume that communication event r in Σ_2 is transmitted to $\mathbf{LOC}_1$ from $\mathbf{G}_2$ via a channel modeled as the (2-state) DES $\mathbf{CH}(2, r, 1)$ in Fig. 1[3]; then r is a channeled event. In the transition structure of $\mathbf{LOC}_1$, hence also of $\mathbf{SUP}_1$, we replace every instance of event r with a new event r', the 'output' of $\mathbf{CH}(2, r, 1)$ corresponding to input r (we call r' the *signal event* of r); call these modified models $\mathbf{LOC}'_1$, $\mathbf{SUP}'_1$. Thus if and when r happens to occur (in $\mathbf{G}_2$) $\mathbf{CH}(2, r, 1)$ is driven by synchronization from its initial state 0 into state 1; on the eventual (and spontaneous) execution of event r' in $\mathbf{SUP}'_1$, which resets $\mathbf{CH}(2, r, 1)$ to state 0, the execution of r' will be forced by synchronization in $\mathbf{LOC}'_1$. In the standard untimed model of DES employed here, the 'time delay' between an occurrence of r and r' is unspecified and can be considered unbounded; indeed, nothing in our model so far implies that r' will cause an actual state change (as opposed to selfloop) because, subsequent to the occurrence of r in $\mathbf{G}_2$, $\mathbf{SUP}'_1$ might conceivably move to states (by events other than r') where r' is a selfloop and its occurrence will not cause a state change in $\mathbf{SUP}'_1$. As a convention, the control status of r' (controllable or uncontrollable) is taken to be that of r. Suppose in particular that r in Σ_2 is controllable. Since $\mathbf{LOC}_1$ has 'control authority' only over controllable events in its private alphabet Σ_1, $\mathbf{LOC}'_1$ never attempts to disable r' directly; r' can only be disabled implicitly by the 'upstream' disablement by $\mathbf{LOC}_2$ of r.

In general $\mathbf{LOC}'_1$ 'knows' that r has occurred in $\mathbf{G}_2$ only when it executes r'; meanwhile, other events may have occurred in $\mathbf{G}_2$. The only constraint placed on events in $\mathbf{G}_2$ is that r cannot occur again until r' has finally reset $\mathbf{CH}(2, r, 1)$ and the communication cycle is ready to repeat. In other words, event r will be delayed in re-occurring until the channel used to transmit event r again becomes

[3] Communications among local supervisors can be modeled in different ways, e.g. [11,12,32]. In our model channel capacity (for each separate channeled event) is exactly 1 (event), imposing the constraint that a given labeled event cannot be retransmitted unless its previous instance has been received and acknowledged by the intended recipient (see footnote 4); this constraint may not be appropriate in all applications. We adopt this model because its structure is reasonable, simple, and renders the distributed control problem (with unbounded communication delay) tractable.

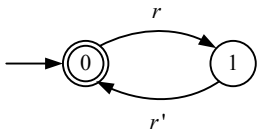

Fig. 1 Communication channel $\mathbf{CH}(2, r, 1)$, from agent $\mathbf{G}_2$ to local controller $\mathbf{LOC}_1$ with channeled event r (in the transition diagram of a DES, the circle with $\rightarrow$ represents the initial state and a double circle represents a marker state). One may think of the delay of r' as being the *sum* of the delay of (forward) event transmission plus the delay of (backward) acknowledgement, i.e. two delays lumped into one. Note that when event r is communicated to multiple local controllers, we employ separate channels with distinct signal events, as illustrated in Fig. 8 below.

available. If event r is controllable, it can be disabled or delayed by the local controller $\mathbf{LOC}_2$;[4] but if event r is uncontrollable, the constraint placed on $\mathbf{G}_2$ will require that r' should reset $\mathbf{CH}(2, r, 1)$ before r is enabled to occur again, possibly in violation of the intended meaning of 'uncontrollable'. This issue will be discussed in Sect. 3.3. The channel $\mathbf{CH}(2, r, 1)$ is not considered a control device, but rather an intrinsic component of the physical system being modeled; it will be 'hard-wired' into the model by synchronous product with $\mathbf{G}_1$ and $\mathbf{G}_2$.

Remark 2 We note that our model of communication channel (Fig. 1) is similar to the mechanism of "synchronous elastic circuits" or "latency insensitive systems" (e.g. [28]). A synchronous elastic circuit is one whose behavior does not change despite latencies (i.e. delays) of communication channels. One method of building synchronous elastic circuits is "synchronous elastic flow" [28], where the idea of "back pressure" is used in a similar way to the "signal events" we use in our model of communication delay.

Continuing with this special case we consider the joint behavior of $\mathbf{G}_1$, $\mathbf{G}_2$ and $\mathbf{CH}(2, r, 1)$ under control of $\mathbf{LOC}'_1$ and $\mathbf{LOC}_2$, namely

$$\begin{aligned} \mathbf{SUP}' :&= Sync(\mathbf{G}_1, \mathbf{LOC}'_1, \mathbf{CH}(2, r, 1), \mathbf{G}_2, \mathbf{LOC}_2) \\ &= Sync(\mathbf{SUP}'_1, \mathbf{CH}(2, r, 1), \mathbf{SUP}_2) \end{aligned} \tag{6}$$

defined over the alphabet $\Sigma_1 \cup \{r'\} \cup \Sigma_2$. We refer to $\mathbf{SUP}'$ as the *channeled behavior* of $\mathbf{SUP}$ (in (1)) with r being the channeled event (i.e. $\Sigma_{ch} = \{r\}$).

3.1 Delay-robustness and Delay-criticality

In this subsection we formalize the definition and present an effective computational test for delay-robustness.

Of principal interest is whether or not the communication delay between successive occurrences of r and r' is tolerable in the intuitive sense indicated above.

[4] Our model implicitly assumes that the sender (i.e. $\mathbf{LOC}_2$) may observe which of the two states $\mathbf{CH}(2, r, 1)$ is at. If $\mathbf{CH}(2, r, 1)$ is at state 1 (the channel is not available), $\mathbf{LOC}_2$ disables r; otherwise r is enabled. In a more fine-grained model we may set $r' = r'_{21}r'_{12}$ where r'_{21} signals to $\mathbf{LOC}'_\mathbf{1}$ the occurrence of r in $\mathbf{G_2}$, while r'_{12} represents an acknowledgement to $\mathbf{LOC_2}$ that r'_{21} has occurred in $\mathbf{SUP}'_\mathbf{1}$. We prove in [33] that these two channel models are equivalent as far as the unbounded delay-robust property is concerned.

Let Σ_{sig} be the set of new events introduced by the communication channels, in which each element is the signal event of an event in Σ_{ch}, i.e.

$$\Sigma_{sig} = \{\sigma' | \sigma \in \Sigma_{ch}, \sigma' \text{ is the signal event of } \sigma\}. \tag{7}$$

In $\mathbf{SUP}'$ (in (6)), $\Sigma_{ch} = \{r\}$ and $\Sigma_{sig} = \{r'\}$. Then the event set of $\mathbf{SUP}'$ will be $\Sigma' = \Sigma \cup \Sigma_{sig} = \Sigma \cup \{r'\}$. Let $P : \Sigma'^* \to \Sigma^*$ be the natural projection of Σ'^* onto Σ^* [21], i.e. P maps r' to ϵ (empty string).

To define whether or not $\mathbf{SUP}'$ with alphabet Σ' has the same behavior as $\mathbf{SUP}$, when viewed through P, we require that

1. anything $\mathbf{SUP}$ can do is the P-projection of something $\mathbf{SUP}'$ can do ($\mathbf{SUP}'$ is 'complete'); and

2. no P-projection of anything $\mathbf{SUP}'$ can do is disallowed by $\mathbf{SUP}$ ($\mathbf{SUP}'$ is 'correct').

For completeness we need at least the inclusions

$$PL(\mathbf{SUP}') \supseteq L(\mathbf{SUP}) \tag{8}$$

$$PL_m(\mathbf{SUP}') \supseteq L_m(\mathbf{SUP}) \tag{9}$$

In addition, however, we need the following *observer property* of P with respect to $\mathbf{SUP}'$ and $\mathbf{SUP}$. Suppose $\mathbf{SUP}'$ executes string $s \in L(\mathbf{SUP}')$, which will be viewed as $Ps \in L(\mathbf{SUP})$. As $\mathbf{SUP}$ is nonblocking, there exists $w \in \Sigma^*$ such that $(Ps)w \in L_m(\mathbf{SUP})$. For any such w 'chosen' by $\mathbf{SUP}$, completeness should require the ability of $\mathbf{SUP}'$ to provide a string $v \in \Sigma'^*$ with the property $Pv = w$ and $sv \in L_m(\mathbf{SUP}')$. Succinctly (cf. [21, 30])

$$\begin{aligned}(\forall s \in \Sigma'^*)(\forall w \in \Sigma^*)\ & s \in L(\mathbf{SUP}')\ \&\ (Ps)w \in L_m(\mathbf{SUP}) \\ & \Rightarrow (\exists v \in \Sigma'^*)\ Pv = w\ \&\ sv \in L_m(\mathbf{SUP}').\end{aligned} \tag{10}$$

Remark 3 In ([21], Chapt. 6), P is defined to be an $L_m(\mathbf{SUP}')$-*observer* if

$$\begin{aligned}(\forall s \in \Sigma'^*)(\forall w \in \Sigma^*)\ & s \in L(\mathbf{SUP}')\ \&\ (Ps)w \in PL_m(\mathbf{SUP}') \\ & \Rightarrow (\exists v \in \Sigma'^*)\ Pv = w\ \&\ sv \in L_m(\mathbf{SUP}').\end{aligned}$$

It is clear that when $PL_m(\mathbf{SUP}') = L_m(\mathbf{SUP})$, the observer property of P with respect to $\mathbf{SUP}'$ and $\mathbf{SUP}$ is identical with the $L_m(\mathbf{SUP}')$-observer property of P.

Briefly, we define $\mathbf{SUP}'$ to be *complete* relative to $\mathbf{SUP}$ if (8), (9) and (10) hold.

Dually, but more simply, we say that $\mathbf{SUP}'$ is *correct* relative to $\mathbf{SUP}$ if

$$PL(\mathbf{SUP}') \subseteq L(\mathbf{SUP}) \tag{11}$$

$$PL_m(\mathbf{SUP}') \subseteq L_m(\mathbf{SUP}) \tag{12}$$

To summarize, we make the following definition.

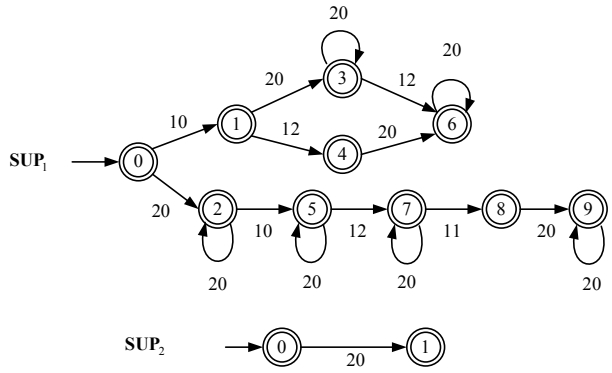

Fig. 2 Example 1: $\mathbf{SUP}_1$ and $\mathbf{SUP}_2$

Definition 1 For given $\mathbf{SUP}'$ in (6) and $\Sigma_{ch} = \{r\}$, **SUP** (in (1)) is *delay-robust* relative to Σ_{ch} provided $\mathbf{SUP}'$ is complete and correct relative to **SUP**, namely, conditions (8)-(12) hold, or explicitly

$$PL(\mathbf{SUP}') = L(\mathbf{SUP}) \tag{13}$$

$$PL_m(\mathbf{SUP}') = L_m(\mathbf{SUP}) \tag{14}$$

P has the observer property (10) with respect to $\mathbf{SUP}'$ and **SUP**. (10bis)

We stress that in Definition 1 (and its generalizations later) the natural projection P is fixed by the choice of channeled events and structure of the communication model. If the definition happens to fail (for instance if the observer property fails), the only cure in the present framework is to alter the set of channeled events, in the worst case reducing it to the empty set, that is, declaring that all communication events must be transmitted without delay.

The following example shows why the observer property is really needed; for if (13) and (14) hold, but (10) fails, $\mathbf{SUP}'$ may have behavior which is distinguishable from that of **SUP**.

Example 1 Let $\mathbf{SUP}_1$ and $\mathbf{SUP}_2$ be the generators shown in Fig. 2; assume event 20 in $\mathbf{SUP}_2$ is exported to $\mathbf{SUP}_1$, i.e., $r = 20$ and $r' = 120$; $\mathbf{SUP}_1'$ is obtained by replacing 20 in $\mathbf{SUP}_1$ by 120, and $\mathbf{SUP}'$ is obtained by (6). By inspection of Fig. 3, (13) and (14) are verified to hold. However, we can see that (10) fails. Let $s = 20.10.120.12 \in L(\mathbf{SUP}')$; then $Ps = 20.10.12$. Now $(Ps).11 = 20.10.12.11 \in L_m(\mathbf{SUP})$; but there does not exist a string v such that $Pv = 11$ and $sv \in L_m(\mathbf{SUP}')$. Thus, **SUP** can execute 11 after Ps, but $\mathbf{SUP}'$ can only execute ϵ after s. This means that $\mathbf{SUP}'$ has behavior distinguishable from that of **SUP**.

Since **SUP** is a nonblocking supervisor, delay-robustness of **SUP** also requires that $\mathbf{SUP}'$ be nonblocking, i.e.

$$\overline{L_m(\mathbf{SUP}')} = L(\mathbf{SUP}'), \tag{15}$$

as can easily be derived from (10),(13) and (14). The following example shows that when delay-robustness fails, transmission delay of r can lead to blocking in $\mathbf{SUP}'$.

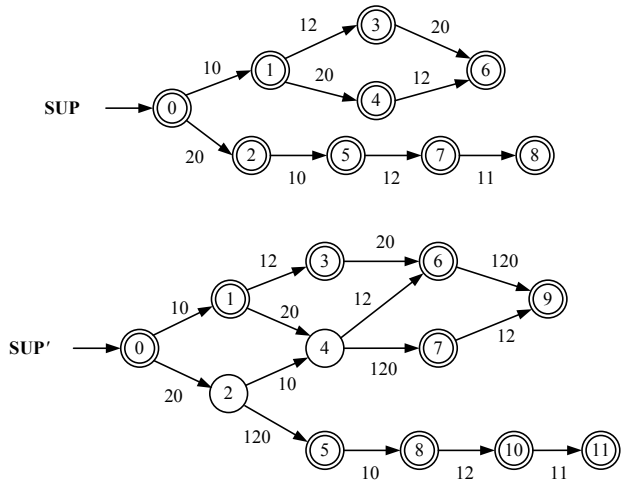

Fig. 3 Example 1: **SUP** and **SUP′**

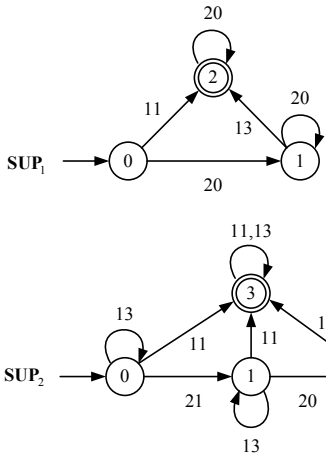

Fig. 4 Example 2: $\mathbf{SUP}_1$ and $\mathbf{SUP}_2$

Example 2 Let $\mathbf{SUP}_1$ and $\mathbf{SUP}_2$ be the generators shown in Fig. 4, and assume event 20 in $\mathbf{SUP}_2$ is exported to $\mathbf{SUP}_1$, i.e., $r = 20$ and $r' = 120$; $\mathbf{SUP}_1'$ is obtained by replacing 20 in $\mathbf{SUP}_1$ by 120. Then **SUP** is nonblocking, but $\mathbf{SUP}'$ obtained by (6) is blocking, as shown in Fig. 5. Note that delay-robustness fails because (13) fails. Indeed, string $21.20.11 \in L(\mathbf{SUP}')$ but $P(21.20.11) = 21.20.11 \notin L(\mathbf{SUP})$. To see why $\mathbf{SUP}'$ is blocking, start from the initial state, and suppose events 21 and 20 have occurred in $\mathbf{SUP}_2$ but that $\mathbf{SUP}_1'$ has not executed the corresponding event 120. Then $\mathbf{SUP}_1'$ may execute event 11, which is immediately observed by $\mathbf{SUP}_2$; however, if 11 occurs, $\mathbf{SUP}_1'$ and $\mathbf{SUP}_2$ cannot accomplish their task synchronously; hence the system blocks.

Given **SUP**, Σ_{ch}, Σ_{sig} and $\mathbf{SUP}'$, we wish to verify whether or not **SUP** is delay-robust relative to Σ_{ch}. For this we need the concept of "supremal quasi-congruence" [21, 29] and the operator *Supqc* [21, Sect. 6.7] which projects a given

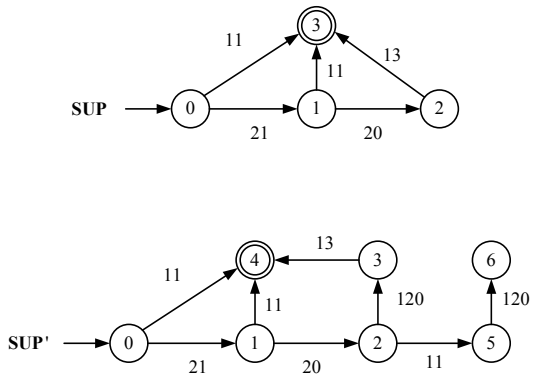

Fig. 5 Example 2: **SUP** and **SUP′**

DES over the alphabet Σ' to **QCDES**, the corresponding quotient **DES** over $\Sigma^* = P(\Sigma'^*)$. We denote the counterpart computing procedure by

$$\mathbf{QCDES} = Supqc(\mathbf{DES}, Null[])$$

where $Null[]$ is the event subset $\Sigma' - \Sigma$ that P maps to the empty string ϵ; for details see [21][5]. Let $\mathbf{QCDES} = (Z, \Sigma, \zeta, z_0, Z_m)$. In general **QCDES** will be nondeterministic with transition function $\zeta : Z \times \Sigma^* \to Pwr(Z)$ and include silent $(\epsilon-)$ transitions. If no silent or nondeterministic transitions happen to appear in **QCDES**, the latter is said to be 'structurally deterministic'. Formally, **QCDES** is *structurally deterministic* if, for all $z \in Z$ and $s \in \Sigma^*$, we have

$$\zeta(z, s) \neq \emptyset \Rightarrow |\zeta(z, s)| = 1.$$

It is known that structural determinism of **QCDES** is equivalent to the condition that P is an $L_m(\mathbf{DES})$-observer (cf. [29], and [21], Theorem 6.7.1).

Given minimal-state deterministic generators **A** and **B** over the same alphabet, we write $\mathbf{A} \subseteq \mathbf{B}$ iff $L_m(\mathbf{A}) \subseteq L_m(\mathbf{B})$ and $L(\mathbf{A}) \subseteq L(\mathbf{B})$; and $\mathbf{A} \approx \mathbf{B}$ to mean both $(\mathbf{A} \subseteq \mathbf{B})$ and $(\mathbf{B} \subseteq \mathbf{A})$, i.e. **A** and **B** are isomorphic. Clearly, "$\approx$" is transitive.

Now let $\mathbf{SUP} = (X, \Sigma, \xi, x_0, X_m)$ (in (1)), $\mathbf{SUP}' = (Y, \Sigma', \eta, y_0, Y_m)$ (in (6)),

$$\mathbf{PSUP}' = Project(\mathbf{SUP}', Null[r']) \tag{16}$$

$$\mathbf{QCSUP}' = Supqc(\mathbf{SUP}', Null[r']). \tag{17}$$

Write $\mathbf{QCSUP}' = (\overline{Y}, \Sigma, \overline{\eta}, \overline{y}_0, \overline{Y}_m)$.

The following theorem provides an effective test for whether or not the communication delay is tolerable, i.e., **SUP** is delay-robust.

Theorem 1 **SUP** *is delay-robust relative to* Σ_{ch} $(= \{r\})$ *if and only if* **QCSUP′** *is structurally deterministic, and isomorphic to* **SUP**.

[5] This procedure can also be phrased in terms of 'bisimulation equivalence' [34], as explained in [29]. We remark that the algorithm for $Supqc(\mathbf{DES}, \cdot)$ in [21], Sect. 6.7, can be estimated to have time complexity $O(kn^4)$ where (k, n) is the (alphabet, state) size of **DES**. We note that [35] reports an algorithm with quadratic time complexity for verifying the observer property alone.

As indicated above, $\mathbf{QCSUP}'$ can be computed by *Supqc* and isomorphism of DES can be verified by *Isomorph*.[6] Hence, Theorem 1 provides an effective computational criterion for delay-robustness. Before Theorem 1 is proved, a special relation between $\mathbf{QCSUP}'$ and $\mathbf{PSUP}'$ must be established; a proof is in Appendix A.

Proposition 1 *If* $\mathbf{QCSUP}'$ *is structurally deterministic, then it is a canonical (minimal-state) generator for* $PL_m(\mathbf{SUP}')$.

Proof of Theorem 1. (If) From Proposition 1, $\mathbf{QCSUP}'$ is a minimal state generator of $PL_m(\mathbf{SUP}')$. So, $\mathbf{QCSUP}' \approx \mathbf{PSUP}'$. As $\mathbf{QCSUP}'$ is isomorphic to $\mathbf{SUP}$, $\mathbf{QCSUP}' \approx \mathbf{SUP}$. Hence, $\mathbf{SUP} \approx \mathbf{PSUP}'$, i.e. (13) and (14) both hold. For (10), since $\mathbf{QCSUP}'$ is structurally deterministic [21, Theorem 6.7.1], P is an $L_m(\mathbf{SUP}')$-observer; by Remark 3 and (14), P has the observer property with respect to $\mathbf{SUP}'$ and $\mathbf{SUP}$. Thus by Definition 1, $\mathbf{SUP}$ is delay-robust relative to Σ_{ch}.

(Only if) By Remark 3, conditions (10) and (14) imply that P is an $L_m(\mathbf{SUP}')$-observer; thus $\mathbf{QCSUP}'$ is deterministic [21]. By Proposition 1, $\mathbf{QCSUP}' \approx \mathbf{PSUP}'$. Equations (13) and (14) say that $\mathbf{PSUP}' \approx \mathbf{SUP}$. Hence $\mathbf{QCSUP}' \approx \mathbf{SUP}$. Finally, we conclude that $\mathbf{QCSUP}'$ is isomorphic to $\mathbf{SUP}$. □

We have now obtained an effective tool to determine whether or not $\mathbf{SUP}$ is delay-robust relative to $\Sigma_{ch} = \{r\}$. If $\mathbf{SUP}$ is not delay-robust relative to r, we say that r is *delay-critical* for $\mathbf{SUP}$. In that case, communication of r (with delay, as r') could result in violation of a specification. If r is delay-critical, and if such violation is inadmissible, then r must be transmitted instantaneously to the agent (in this case, $\mathbf{LOC}_1$) that imports it – where "instantaneous" must be quantified on the application-determined time scale.

3.2 Delay-robustness for Multiple Events

In this subsection, we consider delay-robustness for multiple events. First, we adopt the result of Theorem 1 as the basis of a new (though equivalent) definition and extend delay-robustness naturally to multiple events. Then we prove that delay-robustness for a set R_2 (of multiple events) implies that delay-robustness holds for any subset of R_2.

Definition 2 Let $R_2 \subseteq \Sigma_2$ be a subset of events r imported from $\mathbf{G}_2$ by $\mathbf{LOC}_1$ via their corresponding channels $\mathbf{CH}(2, r, 1)$ (i.e. $\Sigma_{ch} = R_2$), and let $\mathbf{SUP}_1$ be modified to $\mathbf{SUP}_1'$ by replacing each r by its transmitted version r' as before. Let

$$\mathbf{SUP}' := Sync(\mathbf{SUP}_1', \{\mathbf{CH}(2, r, 1) | r \in R_2\}, \mathbf{SUP}_2).$$

Then $\mathbf{SUP}$ is *delay-robust relative to the event subset* R_2 provided $Supqc(\mathbf{SUP}', Null[\{r' | r \in R_2\}])$ is isomorphic to $\mathbf{SUP}$.

[6] For language equality *Isomorph* should be applied to minimal (Nerode) state DES; see e.g. [21] Sect. 3.7.

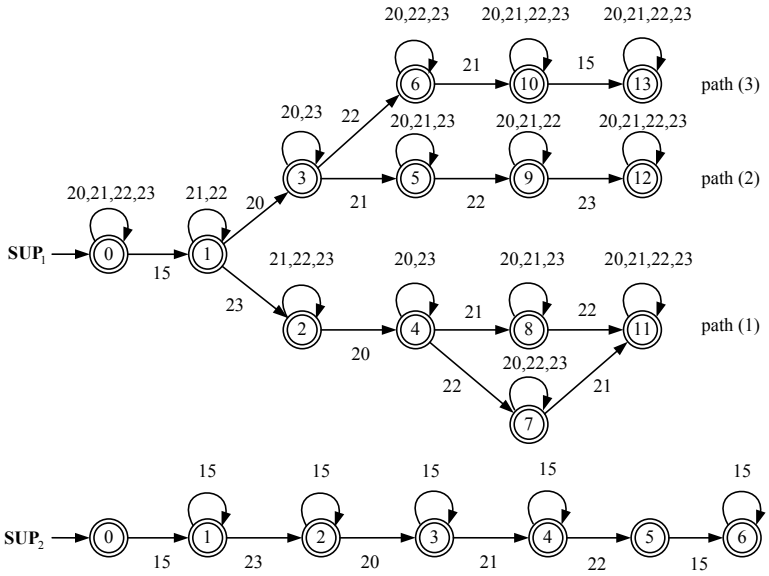

Fig. 6 Example 3: $\mathbf{SUP}_1$ and $\mathbf{SUP}_2$

Note that the property of **SUP** described in Definition 2 is stricter than in Definition 1: that **SUP** is delay-robust with respect to each event $r \in R_2$ taken separately does not imply that **SUP** is delay-robust with respect to R_2 as a subset; however, that **SUP** is delay-robust with respect to R_2 does imply that **SUP** is delay-robust with respect to each separate event $r \in R_2$. The former statement will be confirmed by Example 3 and the latter by Theorem 2.

Example 3 In this example **SUP** is delay-robust with respect to events 21 and 23 separately, but is not delay-robust with respect to the event set $\{21, 23\}$. Let $\mathbf{SUP}_1$ and $\mathbf{SUP}_2$ be the generators shown in Fig. 6, where events 20,21,22,23 in $\mathbf{SUP}_2$ are exported to $\mathbf{SUP}_1$ and event 15 in $\mathbf{SUP}_1$ is exported to $\mathbf{SUP}_2$. Let events 21 and 23 be transmitted by communication channel $\mathbf{CH}(2, 21, 1)$ (with signal event 121) and $\mathbf{CH}(2, 23, 1)$ (with signal event 123) respectively. Let $\mathbf{ASUP}'_1$ (resp. $\mathbf{BSUP}'_1$) be obtained by replacing 21 (resp. 23) in $\mathbf{SUP}_1$ by 121 (resp. 123) and $\mathbf{XSUP}'_1$ be obtained by simultaneously replacing 21 and 23 in $\mathbf{SUP}_1$ by 121 and 123. Let

$$\begin{aligned}
\mathbf{SUP} &= Sync(\mathbf{SUP}_1, \mathbf{SUP}_2) \\
\mathbf{ASUP}' &= Sync(\mathbf{ASUP}'_1, \mathbf{CH}(2, 21, 1), \mathbf{SUP}_2) \\
\mathbf{BSUP}' &= Sync(\mathbf{BSUP}'_1, \mathbf{CH}(2, 23, 1), \mathbf{SUP}_2) \\
\mathbf{XSUP}' &= Sync(\mathbf{XSUP}'_1, \mathbf{CH}(2, 21, 1), \mathbf{CH}(2, 23, 1), \mathbf{SUP}_2),
\end{aligned}$$

as shown in Fig. 7. One can verify that both $Supqc(\mathbf{ASUP}', Null[121])$ and $Supqc(\mathbf{BSUP}', Null[123])$ are isomorphic to **SUP**, i.e. **SUP** is delay-robust with respect to 21 and 23 separately. However, **SUP** is not delay-robust with respect to the event set $\{21, 23\}$. Take

$$s = 15.23.20.123.21.22.121.15.$$

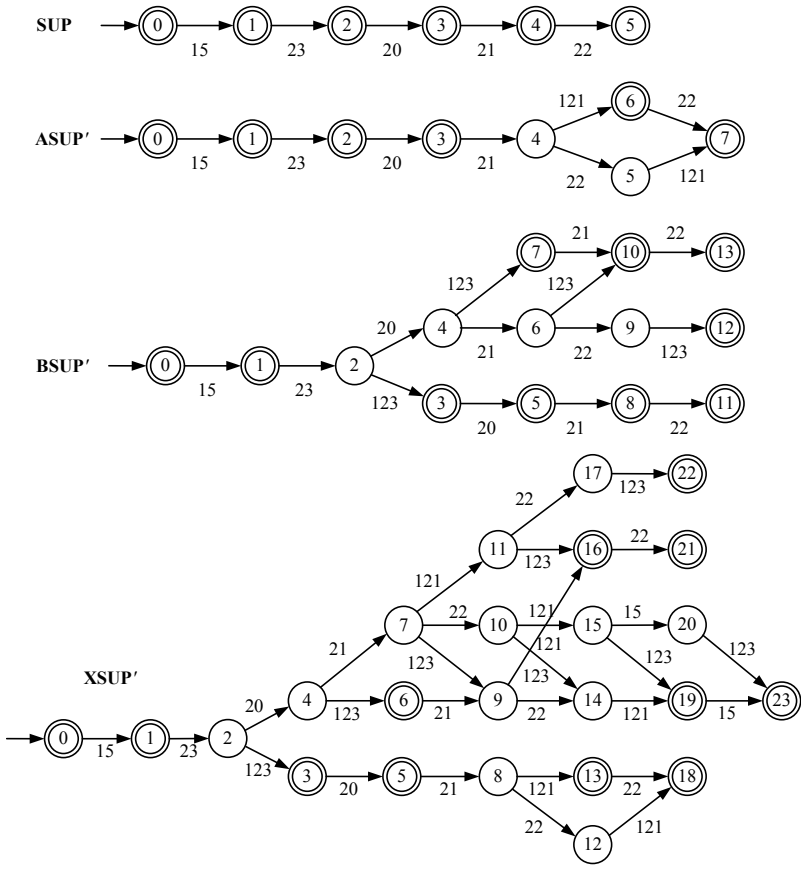

Fig. 7 Example 3: **SUP**, **ASUP′**, **BSUP′** and **XSUP′**

As in Fig. 7, $s \in L(\mathbf{XSUP'})$, but by projecting out 121 and 123,

$$Ps = 15.23.20.21.22.15 \notin L(\mathbf{SUP}),$$

which implies that $PL(\mathbf{XSUP'}) \not\subseteq L(\mathbf{SUP})$ (where P is the natural projection which projects 121 and 123 to the empty string ϵ).

Intuitively, one sees from Fig. 6 that $\mathbf{SUP}_1$ at its state 1 has three paths to choose from: paths (1) and (2) are 'safe', but path (3) is 'dangerous' (because event 15 will occur, which violates **SUP**'s behavior). Which path $\mathbf{SUP}_1$ chooses depends on the events imported from $\mathbf{SUP}_2$. If event 21 alone is delayed, $\mathbf{SUP}_1$ can choose only path (1); if event 23 alone is delayed, $\mathbf{SUP}_1$ can choose either path (1) or (2); thus delaying 21 and 23 individually leads only to 'safe' paths. If, however, events 21 and 23 are both delayed, $\mathbf{SUP}_1$ can choose any of the three paths including the 'dangerous' path (3).

Before addressing delay-robustness for event subsets, we extend our definition to the general case with n agents $\mathbf{G}_j$ ($j \in N = \{1, 2, ..., n\}$), each with local controller $\mathbf{LOC}_j$ which imports channeled events $\Sigma_{ch}(i, j) \subseteq \Sigma_i$ from $\mathbf{G}_i$ ($i \in$

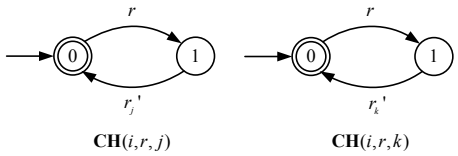

Fig. 8 $\mathbf{CH}(i,r,j)$ and $\mathbf{CH}(i,r,k)$, with distinct signal events r'_j and r'_k

$I_j \subset N$). For this configuration we employ binary channels as before, one for each $r \in \Sigma_{ch}(i,j)$. Thus an event $r \in \Sigma_i$ that is channeled to both $\mathbf{LOC}_j$ and $\mathbf{LOC}_k$ will employ separate channels $\mathbf{CH}(i,r,j)$ and $\mathbf{CH}(i,r,k)$. Here the channels $\mathbf{CH}(i,r,j)$ and $\mathbf{CH}(i,r,k)$ are distinct (see Fig. 8): we use different signal events r'_j and r'_k corresponding to r in $\mathbf{CH}(i,r,j)$ and $\mathbf{CH}(i,r,k)$, respectively; in this way, the channeled event r may be received by $\mathbf{LOC}_j$ and $\mathbf{LOC}_k$ in either order and with unspecified delays. Of course r might also be communicated (but with zero delay) from $\mathbf{G}_i$ to other local controllers $\mathbf{LOC}_l$ with $l \neq j,k$.

For this architecture, Definition 2 is generalized in the obvious way. For each $j \in N$ we compute $\mathbf{SUP}'_j$ by relabeling each event r that appears in $\mathbf{SUP}_i$, such that $r \in \Sigma_{ch}(i,j)$ $(i \in I_j)$, by its channeled output r'. Since $\Sigma_{ch}(i,j) \subseteq \Sigma_i$ and the Σ_i are pairwise disjoint, this relabeling is unambiguous. Then we compute

$$\mathbf{SUP}' = Sync(\mathbf{SUP}'_j, \mathbf{CH}(i,r,j) \mid r \in \Sigma_{ch}(i,j), i \in I_j, j \in N) \tag{18}$$

Note that if for some j, $I_j = \emptyset$, i.e. $\mathbf{LOC}_j$ imports no channeled events from other agents $\mathbf{G}_i$, $i \neq j$, then $\mathbf{SUP}'_j = \mathbf{SUP}_j$.

With $\mathbf{SUP} = Sync(\mathbf{SUP}_j \mid j \in N)$, we have the following definition.

Definition 3 $\mathbf{SUP}$ is *delay-robust for distributed control of n agents by localization* provided the projected channeled behavior

$$Supqc(\mathbf{SUP}', Null\{r' | r \in \Sigma_{ch}(i,j), i \in I_j, j \in N\}) \tag{19}$$

is deterministic, and isomorphic with $\mathbf{SUP}$.

The justification of this definition is merely a repetition of the argument for two agents based on the conditions (13), (14) and (10bis). Once the obvious generalization of $\mathbf{SUP}'$ has been framed, as above, the basic conditions just referenced are fully defined as well, and require no formal change. The final result in terms of *Supqc* is derived exactly as before.

We note that to verify delay-robustness in Definition 3 we need to compute $\mathbf{SUP}'$ as in (18). The computation may be expensive when there is a large number of communication channels. Nevertheless $\mathbf{SUP}'$ is implemented in a purely distributed fashion: distributed supervisors and communication channels. We shall investigate the computational issue of $\mathbf{SUP}'$ in our future work, one promising approach being to use *State Tree Structures* [36]. We also note in passing that all the above results can be extended to decentralized controllers; for details see Appendix B.

In the foregoing notation now suppose that $\mathbf{SUP}$ is known to be delay-robust for a set of binary channels $\mathbf{CH}(i,r,j)$ with $i \in I_j$, $j \in N$, and r in some subset $\Sigma_{ch}(i,j) \subseteq \Sigma_i$. We shall prove that $\mathbf{SUP}$ *remains delay-robust when any one*

of these channels is replaced by the ideal channel with zero transmission delay. As a corollary, delay-robustness is preserved if the given set $\Sigma_{ch}(i,j)$ of channeled events from $\mathbf{G}_i$ to $\mathbf{LOC}_j$ is replaced by any subset. Focussing attention on $\mathbf{SUP}_1 = Sync(\mathbf{G}_1, \mathbf{LOC}_1)$, consider its environment $E = \{\mathbf{SUP}_2, \ldots, \mathbf{SUP}_n\}$ with $\mathbf{SUP}_E := Sync\{\mathbf{SUP}_i \mid i = 2, \ldots, n\}$. We assume that E is augmented to a channeled version E' (say) having internal channels $\mathbf{CH}(i, r_{ij}, j)$ $(i, j = 2, ..., n, i \neq j, r_{ij} \in \Sigma_{ch}(i,j))$, together with outgoing external channels $\mathbf{CH}(j, r_{j1}, 1)$ to $\mathbf{LOC}_1$ and incoming external channels $\mathbf{CH}(1, r_{1i}, i)$ from $\mathbf{G}_1$. Denote the totality of E's internal channels, along with those from $\mathbf{G}_1$, by $\mathbf{CH}_E$. Write $\mathbf{SUP}_{E'} := Sync(\mathbf{SUP}'_2, ..., \mathbf{SUP}'_n, \mathbf{CH}_E)$ where $\mathbf{SUP}'_j$ is $\mathbf{SUP}_j$ with any event $r \in \Sigma_{ch}(i,j)$ replaced by r' $(i = 1, ..., n; j = 2, ..., n; i \neq j)$ as prescribed before. For the alphabet of $\mathbf{SUP}_{E'}$ we have

$$\Sigma_{E'} = \cup\{\Sigma_i \mid i = 2, ..., n\} \ \cup \ \{r' \mid r \in \Sigma_{ch}(i,j); i = 1, ..., n, j = 2, ..., n, i \neq j\}.$$

Similarly let $\mathbf{SUP}'_1$ denote $\mathbf{SUP}_1$ with channeled events $r_{j1} \in \Sigma_{ch}(j,1)$ $(j = 2, ..., n)$ replaced by r'_{j1}, and let Σ'_1 denote the corresponding alphabet. By assumption the alphabets Σ_i $(i = 1, ..., n)$ are pairwise disjoint, hence the $\Sigma_{ch}(j,1)$ $(j = 2, ..., n)$ together with Σ_1 are pairwise disjoint. Write

$$\Sigma_{ch}(E,1) = \cup\{\Sigma_{ch}(j,1) \mid j = 2, ..., n\}.$$

For clarity assume $\Sigma_{ch}(E,1) = \{\alpha, \beta\}$; the extension to more than two events will be evident. Thus α, β are the channeled events imported to $\mathbf{LOC}_1$ from its environment $\mathbf{SUP}_E$ (actually $\mathbf{SUP}_{E'}$), and appear in $\mathbf{SUP}'_1$ as α', β'. We can therefore write $\mathbf{SUP}'$ in (19) in more detail as

$$\mathbf{SUP}' = Sync(\mathbf{SUP}'_1, \mathbf{CH}(E, \alpha, 1), \mathbf{CH}(E, \beta, 1), \mathbf{SUP}_{E'}).$$

Notice that α, β belong to $\Sigma_E := \Sigma_2 \cup \cdots \cup \Sigma_n$ but not Σ_1, whereas α', β' appear in $\mathbf{SUP}'_1$ and the two channels but not in $\mathbf{SUP}_{E'}$.

Now denote by $\mathbf{SUP}''$ the structure $\mathbf{SUP}'$ but with the channel $\mathbf{CH}(E, \alpha, 1)$ replaced by one with zero delay (and so eliminated from the channel formalism). Thus

$$\mathbf{SUP}'' = Sync(\mathbf{SUP}''_1, \mathbf{CH}(E, \beta, 1), \mathbf{SUP}_{E'})$$

where $\mathbf{SUP}''_1$ is $\mathbf{SUP}_1$ with β replaced by β' (but α left unchanged). We shall prove the following result.

Theorem 2 *If* $\mathbf{SUP}$ *is delay-robust with respect to the channel structure of* $\mathbf{SUP}'$*, then it remains so with respect to that of* $\mathbf{SUP}''$*.*

The assertion is almost obvious from the intuition that the statement for $\mathbf{SUP}''$ should be derivable by "taking the limit" at which $\mathbf{CH}(E, \alpha, 1)$ operates with zero delay, namely by replacing the communication event α, when unchanneled, with the zero-delay channeled version $\alpha.\alpha'$, and finally projecting out α'. A proof is given in Appendix C.

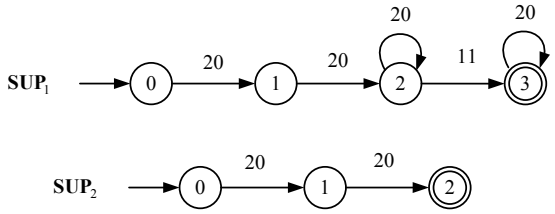

Fig. 9 Example 4: $\mathbf{SUP}_1$ and $\mathbf{SUP}_2$

3.3 Blocking of Uncontrollable Events

The foregoing discussion of delay robustness covers channeled events in general, regardless of their control status, and is adequate if all channeled events happen to be controllable. In the case of uncontrollable channeled events, however, we must additionally examine whether channel delay violates the conventional modeling assumption that uncontrollable events may occur spontaneously at states where they are enabled and should not be subject to external disablement.

In our simplified model the transmission of r from $\mathbf{G}_2$ to $\mathbf{LOC}_1$ is completed (by event r') with indefinite (unbounded) delay. A constraint imposed on $\mathbf{SUP}'$ by the channel $\mathbf{CH}(2, r, 1)$ is that r cannot occur again until r' has reset $\mathbf{CH}(2, r, 1)$ and the communication cycle is ready to repeat. If r is controllable its re-occurrence can be disabled and hence delayed until after the occurrence of r' corresponding to the previous occurrence of r. If, however, r is uncontrollable, then once it is re-enabled (by entrance of $\mathbf{SUP}_2$ to a state where r is defined) its re-occurrence cannot be externally delayed, according to the usual modeling assumption on uncontrollable events. In this sense the introduction of $\mathbf{CH}(2, r, 1)$ could conceivably conflict with the intention of the original DES model. To address this issue we examine whether or not communication delay of an uncontrollable event might violate a modeling assumption.

Example 4 For illustration, let $\mathbf{SUP}_1$ and $\mathbf{SUP}_2$ be the generators shown in Fig. 9. Assume event 20 in $\mathbf{SUP}_2$ is exported to $\mathbf{SUP}_1$, i.e., $r = 20$ and $r' = 120$; $\mathbf{SUP}_1'$ is obtained by replacing 20 in $\mathbf{SUP}_1$ by 120. As shown in Fig. 10, $\mathbf{SUP}' = Sync(\mathbf{SUP}_1', \mathbf{CH}(2, 20, 1), \mathbf{SUP}_2)$ is easily verified to be delay-robust with respect to event 20. Define $\mathbf{NSUP} = Sync(\mathbf{SUP}_1', \mathbf{SUP}_2)$. Let $s = 20$; then $s.20 \in L(\mathbf{NSUP})$, but $s.20 \notin L(\mathbf{SUP}')$. Since $\mathbf{SUP}' = Sync(\mathbf{NSUP}, \mathbf{CH}(2, 20, 1))$, event 20 is blocked by $\mathbf{CH}(2, 20, 1)$.

This example shows a case where the reoccurrence of an uncontrollable event is 'blocked' by its channel, which demonstrates that communication delay of an uncontrollable event really violates the modeling assumption that uncontrollable events cannot be disabled by any external agent. Now let

$$\mathbf{NSUP} = Sync(\mathbf{SUP}_1', \mathbf{SUP}_2); \tag{20}$$

then according to (6)

$$\mathbf{SUP}' = Sync(\mathbf{NSUP}, \mathbf{CH}(2, r, 1)). \tag{21}$$

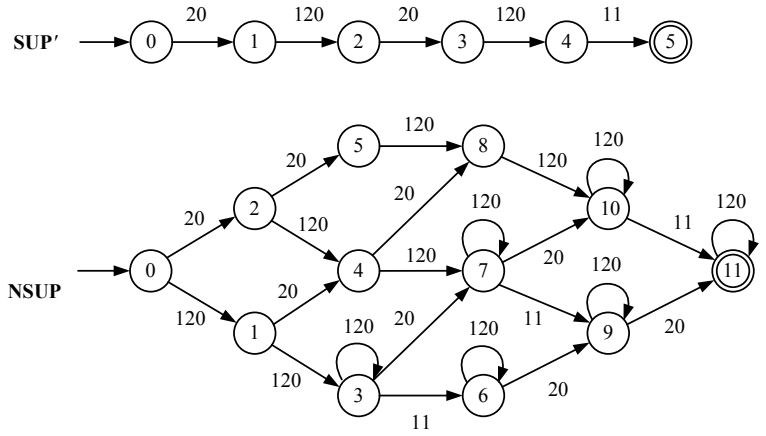

Fig. 10 Example 4: $\mathbf{SUP}'$ and **NSUP**

As before, write $\Sigma' = \Sigma \cup \{r'\}$ for the alphabet of $\mathbf{SUP}'$, let $P : \Sigma'^* \to \Sigma^*$ be the natural projection of Σ'^* to Σ^*, and define the new natural projection $P_r : \Sigma'^* \to \{r, r'\}^*$. Now, for given **NSUP** and $\mathbf{SUP}'$ as in (20) and (21), and $r \in \Sigma_u$, if there exists $s \in L(\mathbf{SUP}')$ such that $sr \in L(\mathbf{NSUP})$, but $sr \notin L(\mathbf{SUP}')$, then we say that r is *blocked* by $\mathbf{CH}(2, r, 1)$.

To check whether or not r is blocked by $\mathbf{CH}(2, r, 1)$, we check if $P_r^{-1}L(\mathbf{CH}(2, r, 1))$ is **NSUP**-controllable with respect to event r, i.e.

$$P_r^{-1}L(\mathbf{CH}(2, r, 1))r \cap L(\mathbf{NSUP}) \subseteq P_r^{-1}L(\mathbf{CH}(2, r, 1)).$$

For this, we employ the standard algorithm that checks controllability [21]; the algorithm has complexity $O(mn)$ where m and n represent the state numbers of $\mathbf{CH}(2, r, 1)$ and **NSUP**, respectively.[7]

To summarize, for an uncontrollable event r, if **SUP** is delay-robust (by Theorem 1) and r is never blocked by $\mathbf{CH}(2, r, 1)$ (by the controllability checking algorithm), then **SUP** is said to be 'unbounded' delay-robust with respect to r. Otherwise, there exists $s \in L(\mathbf{SUP}')$ such that $sr \in L(\mathbf{NSUP})$, but $sr \notin L(\mathbf{SUP}')$. Thus r is blocked by the channel, which could violate the modeling assumption that an uncontrollable event should never be prohibited or delayed by an external agent. However, if the occurrence of r' is executed by $\mathbf{LOC}_1$ before the next occurrence of r, the controllers may still achieve global optimal nonblocking supervision. In this case, we say that **SUP** is 'bounded' delay-robust with respect to r.[8]

We illustrate the foregoing results by an example adapted from [21].

[7] For the case described in Section 3.2 of transmitting multiple events by separate channels, we use the same method to check if each event r is blocked. Specifically, we check if $P_r^{-1}L(\mathbf{CH}(i, r, j))$ is **NSUP**-controllable with respect to r, where **NSUP** denotes the behavior of the system excluding $\mathbf{CH}(i, r, j)$.

[8] One way to determine a delay bound in terms of number of event occurrences is to find the shortest path between two consecutive occurrences of event r in **SUP**. A more detailed study of this issue is left for future research.

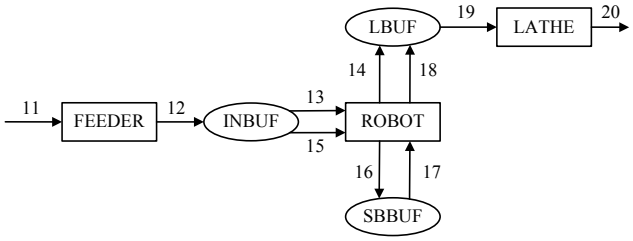

Fig. 11 WORKCELL

Table 1 Physical interpretation of events

Event label	Physical interpretation
11	**FEEDER** imports new part from infinite source
12	**FEEDER** loads new part in **INBUF**
13	**ROBOT** takes part from **INBUF** for loading into **LBUF**
14	**ROBOT** loads part from **INBUF** into **LBUF**
15	**ROBOT** takes part from **INBUF** for loading into **SBBUF**
16	**ROBOT** loads part from **INBUF** into **SBBUF**
17	**ROBOT** takes part from **SBBUF** for loading into **LBUF**
18	**ROBOT** loads part from **SBBUF** into **LBUF**
19	**LATHE** loads part from **LBUF** and starts working
20	**LATHE** exports finished part and returns to idle

4 Example - WORKCELL

4.1 Model Description and Controller Design

WORKCELL consists of **ROBOT**, **LATHE** and **FEEDER**, with three buffers, **INBUF**, **LBUF** and **SBBUF**, connected as in Fig. 11. Labeled arrows denote synchronization on shared transitions (events) in the corresponding component DES.

WORKCELL operates as follows: **FEEDER** acquires a new part from an infinite source (event 11) then stores it (event 12) in a 2-slot buffer **INBUF**. **ROBOT** takes a new part from **INBUF** (event 13) and stores it (event 14) in a 1-slot buffer **LBUF**; if **LBUF** is already full, **ROBOT** may instead take a new part from **INBUF** (event 15) and store it (event 16) in a 1-slot 'stand-by' buffer **SBBUF**. If **LBUF** is empty and there's already a part in **SBBUF**, **ROBOT** first unloads the part in **SBBUF** (event 17) and loads it in **LBUF** (event 18). If **LATHE** is idle and there exists a part in **LBUF**, **LATHE** takes that part and starts working on it (event 19), and when finished exports it and returns to idle (event 20). Event labels accord with [31]: odd-(resp. even-) numbered events are controllable (resp. uncontrollable). The physical interpretations of events are displayed in Table 1.

The specifications to be enforced are: 1) $\mathbf{SPEC}_1$ says that a buffer must not overflow or underflow; 2) $\mathbf{SPEC}_2$ says that **ROBOT** can load **SBBUF** (event sequence 15.16) only when **LBUF** is already full; 3) $\mathbf{SPEC}_3$ says that **ROBOT** can load **LBUF** directly from **INBUF** (event sequence 13.14) only when **SBBUF** is empty; otherwise it must load from **SBBUF** (event sequence 17.18). The DES models of plant components and specifications are shown in Figs. 12 and 13.

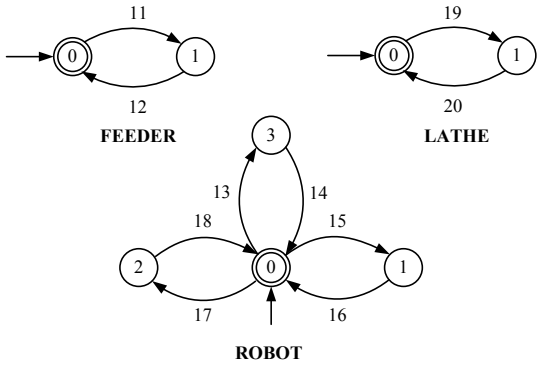

Fig. 12 Plant models to be controlled

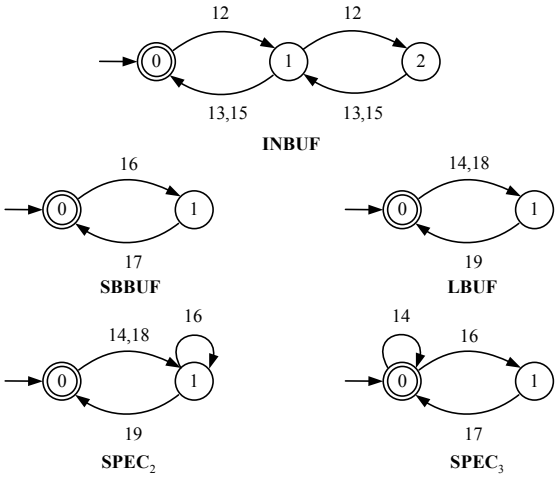

Fig. 13 Model of Specifications

We first compute the monolithic supervisor by a standard method (e.g. [21, 31]). The behavior of **WORKCELL** is the synchronous product of **FEEDER**, **ROBOT**, and **LATHE**. As $\mathbf{SPEC}_1$ is automatically incorporated in the buffer models, the total specification **SPEC** is the synchronous product of **INBUF**, **LBUF**, **SBBUF**, $\mathbf{SPEC}_2$, and $\mathbf{SPEC}_3$. The monolithic supervisor is $\mathbf{SUPER} = \mathit{Supcon}(\mathbf{WORKCELL}, \mathbf{SPEC})$ with (state, transition) count (70, 153).

Next by use of procedure *Localize* [21, 31], we compute the localization of **SUPER** (in the sense of [5, 6]) to each of the three **WORKCELL** agents, to obtain local controllers **FEEDERLOC**, **ROBOTLOC** and **LATHELOC**, as shown in Fig. 14. The local controlled behaviors are

$$\begin{aligned}\mathbf{FEEDERSUP} &= \mathit{Sync}(\mathbf{FEEDER}, \mathbf{FEEDERLOC}),\\ \mathbf{ROBOTSUP} &= \mathit{Sync}(\mathbf{ROBOT}, \mathbf{ROBOTLOC}),\\ \mathbf{LATHESUP} &= \mathit{Sync}(\mathbf{LATHE}, \mathbf{LATHELOC}).\end{aligned}$$

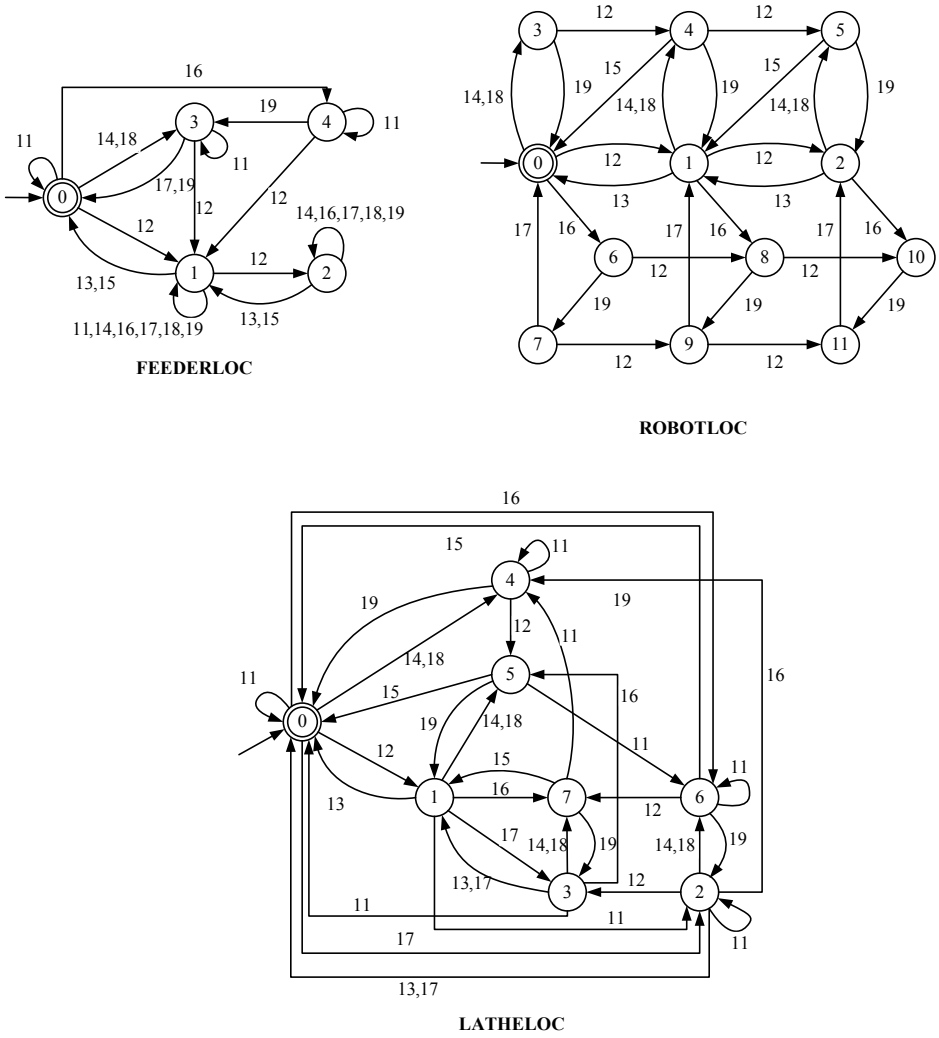

Fig. 14 Local Controller for each component. According to Remark 1, for every state x of each controller, and each communication event σ imported from some other component, if σ is not defined at x, we add a σ-selfloop. Let $*(x)$ be the set of selfloops to be adjoined at state x. In **FEEDERLOC**, $*(0) = \{13, 15, 17, 19\}$, $*(3) = \{13, 14, 15, 16, 18\}$, $*(4) = \{13, 14, 15, 16, 17, 18\}$; in **ROBOTLOC**, $*(0) = \{19\}$, $*(1) = \{19\}$, $*(2) = \{12, 19\}$, $*(5) = \{12\}$, $*(7) = \{19\}$,$*(9) = \{19\}$,$*(10) = \{12\}$, $*(11) = \{19\}$; in **LATHELOC**, $*(0) = \{13, 15\}$, $*(1) = \{12, 15\}$, $*(2) = \{15\}$, $*(3) = \{12, 15\}$, $*(4) = \{13, 14, 15, 16, 17, 18\}$,$*(5) = \{12, 13, 14, 16, 17, 18\}$, $*(6) = \{13, 14, 16, 17, 18\}$,$*(7) = \{12, 13, 14, 16, 17, 18\}$.

From the transition structures shown in Fig. 14, we see that **FEEDERLOC** (**FEEDERSUP**) must import events 13, 14, 15, 16, 17 and 18 from **ROBOT**, and 19 from **LATHE**; **ROBOTLOC** (**ROBOTSUP**) must import events 12 from **FEEDER**, and 19 from **LATHE**; and **LATHELOC** (**LATHESUP**) must import events 11 and 12 from **FEEDER**, and 13, 14, 15, 16, 17 and 18 from **ROBOT**.

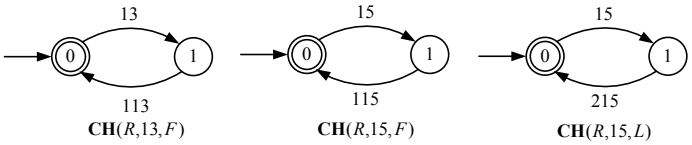

Fig. 15 $\mathbf{CH}(R, 13, F)$, $\mathbf{CH}(R, 15, F)$, and $\mathbf{CH}(R, 15, L)$

4.2 Illustrative Cases

Based on the computed local controllers, we illustrate our new verification tools with the following cases.

Case 1 – Single Event

In this case, we show that **SUPER** is delay-robust relative to event 13, but is not delay-robust relative to event 19.

(1) Taking **FEEDERLOC** for example, build a channel $\mathbf{CH}(R, 13, F)$, as shown in Fig. 15, using a new event label 113 to represent the corresponding channel output; use 113 to replace 13 in **FEEDERSUP** to obtain $\mathbf{FEEDERSUP}'$, over the alphabet {11,12,113,14,15,16,17,18,19}.

Now compute the channeled behavior $\mathbf{SUPER}'$ according to

$$\mathbf{SUPER}' = Sync(\mathbf{FEEDERSUP}', \mathbf{CH}(R, 13, F), \mathbf{ROBOTSUP}, \mathbf{LATHESUP})$$

over the augmented alphabet {11, ..., 20, 113} and with (state, transition) count (124, 302). Next, to check delay-robustness we project $\mathbf{SUPER}'$ modulo supremal quasi-congruence with nulled event 113, to get, say,

$$\mathbf{QCSUPER}' := Supqc(\mathbf{SUPER}', Null[113]) \\ (deterministic, with\ size\ (70, 153))$$

Finally we verify that $\mathbf{QCSUPER}'$ is isomorphic to **SUPER**, and conclude that **SUPER** is delay-robust with respect to the channeled communication of event 13 from **ROBOT** to **FEEDERLOC**. As a physical interpretation, consider the case where events 11, 12, 11, 12, 13 have occurred sequentially (i.e. there exist two parts in **INBUF** and **ROBOT** has taken a part from **INBUF**) and $\mathbf{FEEDERSUP}'$ has not executed the occurrence 113 of event 13. On the one hand, if $\mathbf{FEEDERSUP}'$ executes event 113 (i.e. it acknowledges the occurrence of event 13), it will enable event 11 legally (according to **SUPER**). On the other hand, if $\mathbf{FEEDERSUP}'$ does not execute event 113, then **ROBOT** will load the part into **LBUF** and take another part from **INBUF** (execute event 15). So $\mathbf{FEEDERSUP}'$ can enable event 11 again, which is also legal according to **SUPER**. Hence, in this case, the channeled system $\mathbf{SUPER}'$ can run 'correctly'(no extra behavior violates the specification) and can 'complete' the given task (with the help of **SBBUF**), i.e. the communication delay of event 13 is tolerable with respect to **SUPER**.

(2) By the same method, one can verify that event 19 channeled to **ROBOTLOC** is delay-critical with respect to **SUPER**. By tracking the working process, we show that indefinite communication delay of event 19 may result in violation of $\mathbf{SPEC}_2$. Consider the following case: events 11,12,11,12,13,14,19 have occurred sequentially, i.e. there exists one part in **INBUF**, **ROBOT** has loaded a part in **LBUF** and

LATHE has taken the part from **LBUF** (i.e. **LBUF** is now empty). Since the transmission of event 19 is delayed unboundedly, if **ROBOT** doesn't 'know' that **LATHE** has taken the part from **LBUF**, it may take a new part from **INBUF** (event 15) and load it into **SBBUF** (event 16) according to $\mathbf{ROBOTSUP}'$, i.e. the event sequence 11.12.11.12.13.14.19.15.16 occurs in **WORKCELL** with communication delay, violating $\mathbf{SPEC}_2$. Hence event 19 is delay-critical.

Case 2 – Multiple Events (Channels)

(1) We show that **SUPER** is delay-robust relative to the event set {13, 15}, with 13 and 15 both channeled to **FEEDERLOC**.

Consider the channel $\mathbf{CH}(R, 15, F)$ displayed in Fig. 15, using the signal event 115 to represent the corresponding channel output. Use labels 113, 115 to replace 13, 15 in **FEEDERSUP** to obtain $\mathbf{FEEDERSUP}'$, over the alphabet {11,12,113, 14, 115,16,17,18,19}.

We compute the channeled behavior $\mathbf{SUPER}'$ according to

$$\mathbf{SUPER}' = Sync(\mathbf{FEEDERSUP}', \mathbf{CH}(R, 13, F), \mathbf{CH}(R, 15, F), \mathbf{ROBOTSUP}, \mathbf{LATHESUP}),$$

over the augmented alphabet {11, ..., 20, 113, 115} and with (state, transition) count (180, 470). Next, to check delay-robustness we project $\mathbf{SUPER}'$ modulo supremal quasi-congruence with nulled events 113, 115, to get

$$\mathbf{QCSUPER}' := Supqc(\mathbf{SUPER}', Null[113, 115]) \quad (deterministic, with\ size\ (70, 153))$$

$\mathbf{QCSUPER}'$ turns out to be isomorphic to **SUPER**, and we conclude that **SUPER** is delay-robust with respect to the channeled communication of events 13, 15 from **ROBOT** to **FEEDERLOC**. Briefly, the reason is that $\mathbf{FEEDERSUP}'$ will enable event 11 after it executes event 113 or 115, and **ROBOT** will remain idle if no more parts are loaded into the system (i.e. event 11 cannot occur again).

(2) We verify that **SUPER** is delay-robust with respect event 15 *provided* it is channeled only to **FEEDERLOC**; however, it must be communicated to **LATHELOC** without delay. To verify this, we have two separate channels, $\mathbf{CH}(R, 15, F)$ and $\mathbf{CH}(R, 15, L)$, with distinct signal events 115 and 215 (see Fig. 15). Taking the two channels separately, by Definition 1 and the same method as above for event 13, we verify that **SUPER** is delay-robust when 15 is communicated to **FEEDERLOC** by $\mathbf{CH}(R, 15, F)$, but delay-critical to **LATHELOC** by $\mathbf{CH}(R, 15, L)$. Moreover, by Definition 3 and the procedure in Sect. 3.2, we verify that **SUPER** is delay-critical when 15 is communicated to both **FEEDERLOC** and **LATHELOC**.

Case 3 – Blockingness of Uncontrollable Event

This case shows that although the occurrence of (uncontrollable) event 12 (channelled to **ROBOTLOC**) may be blocked by its channel $\mathbf{CH}(F, 12, R)$, as shown in Fig. 16, this will not violate the specifications. According to Sect 3.3, we check whether $L(\mathbf{CH}(F, 12, R))$ is controllable with respect to

$$\mathbf{NSUPER} = Sync(\mathbf{FEEDERSUP}, \mathbf{ROBOTSUP}', \mathbf{LATHESUP}).$$

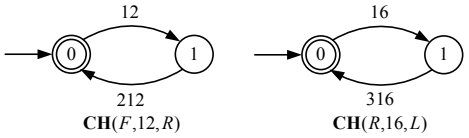

Fig. 16 $\mathbf{CH}(F, 12, R)$ and $\mathbf{CH}(R, 16, L)$

In [31], we use *Condat*, which tabulates the set of events disabled in $\mathbf{CH}(F, 12, R)$ with respect to **NSUPER**, to implement the verification of the controllability for $L(\mathbf{CH}(F, 12, R))$.[9]

By using *Condat*, it turns out that event 12 is disabled at state 1 of $L(\mathbf{CH}(F, 12, R))$. Physically, suppose 11, 12 and 11 have occurred sequentially, i.e., **FEEDER** has stored a part in **INBUF** and taken another part (event 11). After that, **FEEDER** may store the part in **INBUF** (event 12, which is uncontrollable). If **ROBOTSUP** does not acknowledge the first occurrence of 12, then $\mathbf{CH}(F, 12, R)$ is at state 1, and thus cannot transmit the next occurrence of 12. So, in the channeled system $\mathbf{SUPER}'$, event 12 is blocked by $\mathbf{CH}(F, 12, R)$. If transmission of the first 12 is completed (i.e. event 212 occurs) before the second occurrence of event 12, then event 12 will not be blocked. In **SUPER**, only event 11 occurs between two occurrences of event 12; thus we say that **SUPER** is '1-bound'-delay-robust with respect to event 12.

Case 4 – All communication events

When all communication events are subject to delay through channels (i.e. $\Sigma_{ch} = \Sigma_{com}$), it can be verified that delay-robustness of **SUPER** in the strong sense of Definition 3 fails, i.e. **SUPER** fails to be delay-robust for distributed control by localization. In fact when all the channeled events except 19 (channeled to **ROBOTLOC**) are received without delay, Case 4 is reduced to Case 1; so **SUPER** cannot be delay-robust with respect to the set of all communication events, consistently with Theorem 2 in Sect. 3.

5 Conclusions and Future Work

In this paper we have studied distributed control obtained by supervisor localization on the relaxed assumption (compared to previous literature [5, 6]) that inter-agent communication of selected 'communication events' (channeled events) may be subject to unknown time delays. For this distributed architecture we have identified a property of 'delay-robustness' which guarantees that the logical properties of our delay-free distributed control (i.e. the original DES specifications) continue to be enforced in the presence of delay, albeit with possibly degraded temporal behavior. We have shown that delay-robustness can be effectively tested with polynomial complexity, and that such tests serve to distinguish between events that are delay-critical and those that are not. The case that an uncontrollable channeled event may be blocked by its communication channel is identified by the algorithm for checking controllability. A simple workcell exemplifies the

[9] Here the alphabet of $\mathbf{CH}(F, 12, R)$ is $\{12, 212\}$; before calling *Condat*, one should add the selfloop with events in **NSUPER** but not in $\{12, 212\}$ at each state of $\mathbf{CH}(F, 12, R)$.

approach, showing how delay-robustness may depend on the subset of events subject to delay, and that a given event may be delay-critical for some choices of the delayed event subset but not for others.

With the definitions and tests reported here as basic tools, future work should include the investigation of alternative channel models and, of especial interest, global interconnection properties of a distributed system of DES which render delay-robustness more or less likely to be achieved. A quantitative approach involving timed discrete-event systems could also be an attractive extension.

Appendices

A Proof of Proposition 1

Recall that $\mathbf{SUP}' = (Y, \Sigma', \eta, y_0, Y_m)$. According to natural projection $P : \Sigma'^* \to \Sigma^*$ which maps $(\Sigma' - \Sigma)$ to ϵ, define $\eta' : Y \times \Sigma^* \to Pwr(Y)$ given by

$$\eta'(y,t) = \{\eta(y,s) | s \in \Sigma'^*, \eta(y,s)! \ \& Ps = t\}. \tag{22}$$

Let ρ be the supremal quasi-congruence on Y with respect to $\mathbf{SUP}'$, and define $P_\rho : Y \to Y/\rho = \overline{Y}$. As in ([21], Chapt. 6), $\mathbf{QCSUP}' = (\overline{Y}, \Sigma, \overline{\eta}, \overline{y}_0, \overline{Y}_m)$ is defined with $\overline{\eta} : \overline{Y} \times \Sigma^* \to Pwr(\overline{Y})$ given by

$$\overline{\eta}(\overline{y}, t) := \bigcup \{P_\rho(\eta'(y,t)) | P_\rho(y) = \overline{y}\}, \tag{23}$$

$\overline{y}_0 = P_\rho(y_0)$ and $\overline{Y}_m = P_\rho(Y_m)$.

Proof. We must prove that $\mathbf{QCSUP}'$ represents $PL_m(\mathbf{SUP}')$ and is a canonical generator.

(1) We show that $\mathbf{QCSUP}'$ represents $PL_m(\mathbf{SUP}')$, i.e,

$$L_m(\mathbf{QCSUP}') = PL_m(\mathbf{SUP}')$$

and

$$L(\mathbf{QCSUP}') = PL(\mathbf{SUP}').$$

(i) $L(\mathbf{QCSUP}') \subseteq PL(\mathbf{SUP}')$

Let $t \in L(\mathbf{QCSUP}')$. We prove by induction that $t \in PL(\mathbf{SUP}')$.

Base step: $t = \epsilon \in PL(\mathbf{SUP}')$ trivially.

Inductive step: Suppose $t \in L(\mathbf{QCSUP}')$, $t \in PL(\mathbf{SUP}')$, and $t\alpha \in L(\mathbf{QCSUP}')$; we must prove $t\alpha \in PL(\mathbf{SUP}')$.

Since $t\alpha \in L(\mathbf{QCSUP}')$, we have $\overline{\eta}(\overline{y_0}, t)!$ and $\overline{\eta}(\overline{y_0}, t\alpha)!$. So, $(\exists \overline{y} \in \overline{Y})\ \overline{y} = \overline{\eta}(\overline{y_0}, t)\ \&\ \overline{\eta}(\overline{y}, \alpha)!$. We have $\overline{y_0} = P_\rho y_0$. Since $t \in PL(\mathbf{SUP}')$, $(\exists s \in L(\mathbf{SUP}'))\ Ps = t$, i.e. $\eta(y_0, s)!$. So, $\eta(y_0, s) \in \eta'(y_0, t)$, i.e., $\eta'(y_0, t) \neq \emptyset$. Thus, $\overline{y} = P_\rho \eta'(y_0, t)$ because $\mathbf{QCSUP}'$ is deterministic. Since $\overline{\eta}(\overline{y}, \alpha)!$ and $\eta'(y_0, t) \neq \emptyset$, there exists $y \in \eta'(y_0, t)$ such that $\overline{\eta}(\overline{y}, \alpha) = P_\rho \eta'(y, \alpha)$. Hence, $\eta'(y_0, t\alpha)!$. However, according to (22)

$$\eta'(y_0, t\alpha) = \{\eta(y_0, s) | s \in \Sigma^*, \eta(y_0, s)!, Ps = t\alpha\}.$$

Thus, $(\exists s \in L(\mathbf{SUP}'))\ Ps = t\alpha$, so $t\alpha \in PL(\mathbf{SUP}')$.

(ii) $PL(\mathbf{SUP}') \subseteq L(\mathbf{QCSUP}')$

Let $t \in PL(\mathbf{SUP}')$; we show that $t \in L(\mathbf{QCSUP}')$.

Base step: $t = \epsilon \in L(\mathbf{QCSUP}')$ trivially.

Inductive step: Supposing $t \in PL(\mathbf{SUP}')$, $t \in L(\mathbf{QCSUP}')$, and $t\alpha \in PL(\mathbf{SUP}')$, we show $t\alpha \in L(\mathbf{QCSUP}')$).

Since $t \in PL(\mathbf{SUP'})$ and $t \in L(\mathbf{QCSUP'})$, $\eta'(y_0,t) \neq \emptyset$, $\overline{\eta}(\overline{y_0},t)!$; letting $\overline{y} = \overline{\eta}(\overline{y_0},t)$, then $\overline{y} = P_\rho\eta'(y_0,t)$ because $\mathbf{QCSUP'}$ is deterministic. Since $t\alpha \in PL(\mathbf{SUP'})$, there exists $s' \in L(\mathbf{SUP'})$, i.e. $\eta(y_0,s')!$ such that $Ps' = t\alpha$; thus

$$\begin{aligned}
& \bigcup\{\eta'(y',\alpha)|y' \in \eta'(y_0,t)\} \\
= & \bigcup\{\eta'(y',\alpha)|s \in \Sigma'^*, y' = \eta(y_0,s), Ps = t\} \ \text{(according to (22))} \\
= & \{\eta((\eta(y_0,s),v))|v \in \Sigma'^*, \eta(\eta(y_0,s),v)!, Ps = t, Pv = \alpha\} \\
= & \{\eta(y_0,sv)|sv \in \Sigma'^*, \eta(y_0,sv)!, P(sv) = t\alpha\} \\
\neq & \ \emptyset \ \text{(since } \eta(y_0,s')! \text{ and } Ps' = t\alpha),
\end{aligned}$$

i.e. there exists $y \in \eta'(y_0,t)$ such that $\eta'(y,\alpha)!$. Then, $P_\rho y = \overline{y}$ due to $\overline{y} = P_\rho\eta'(y_0,t)$. Hence, $\overline{\eta}(\overline{y},\alpha) = P_\rho\eta'(y,\alpha) \neq \emptyset$, i.e., $\overline{\eta}(\overline{y},\alpha)!$. So, $t\alpha \in L(\mathbf{QCSUP'})$.

(iii) $L_m(\mathbf{QCSUP'}) \subseteq PL_m(\mathbf{SUP'})$

For any $t \in \Sigma^*$, if $t \in L_m(\mathbf{QCSUP'})$, then $(\exists \overline{y} \in \overline{Y})\ \overline{y} = \overline{\eta}(\overline{y}_0,t)\ \&\ \overline{y} \in \overline{Y}_m$. By (i), we conclude that $t \in PL(\mathbf{SUP'})$. Thus, $\eta'(y_0,t) \neq \emptyset$. Because $\mathbf{QCSUP'}$ is deterministic, we know that $\overline{y} = P_\rho\eta'(y_0,t)$. So, $P_\rho\eta'(y_0,t) \in \overline{Y}_m$. Further, $\eta'(y_0,t) \cap Y_m \neq \emptyset$, i.e., there exists $s \in \Sigma'^*$ such that $\eta(y_0,s)!\ \&\ \eta(y_0,s) \in Y_m\ \&\ Ps = t$. Hence, $s \in L_m(\mathbf{SUP'})$, thus $t = Ps \in PL_m(\mathbf{SUP'})$.

(iv) $PL_m(\mathbf{SUP'}) \subseteq L_m(\mathbf{QCSUP'})$

For any $t \in \Sigma^*$, if $t \in PL_m(\mathbf{SUP'})$, then $\eta'(y_0,t)!\ \&\ \eta'(y_0,t) \cap Y_m \neq \emptyset$. By (ii), $t \in L(\mathbf{QCSUP'})$, i.e., $(\exists \overline{y} \in \overline{Y})\ \overline{\eta}(\overline{y}_0,t)!\ \&\ \overline{y} = \overline{\eta}(\overline{y}_0,t)$. Since $\mathbf{QCSUP'}$ is deterministic, $\overline{y} = P_\rho\eta'(y_0,t)$. We conclude that $P_\rho\eta'(y_0,t) \in \overline{Y}_m$ from $\eta'(y_0,t) \cap Y_m \neq \emptyset$. Hence, $\overline{y} \in \overline{Y}_m$, i.e., $t \in L_m(\mathbf{QCSUP'})$.

2. We prove that $\mathbf{QCSUP'}$ is a canonical(minimal-state) generator.

Let ν be a congruence on $\overline{Y}$ defined according to: $\overline{y} \equiv \overline{y'} \pmod{\nu}$ provided

(i) $(\forall t \in \Sigma^*)\ \overline{\eta}(\overline{y},t)! \Leftrightarrow \overline{\eta}(\overline{y'},t)!$

(ii)$(\forall t \in \Sigma^*)\ \overline{\eta}(\overline{y},t) \in \overline{Y}_m \Leftrightarrow \overline{\eta}(\overline{y'},t) \in \overline{Y}_m$.

With reference to ([21], Proposition 2.5.1), projection (mod ν) reduces $\mathbf{QCSUP'}$ to a state-minimal generator.

Define $P_\nu : \overline{Y} \to \overline{Y}/\nu$ and write $\nu \circ \rho = ker(P_\nu \circ P_\rho)$. Next we will prove that $\nu \circ \rho$ is a quasi-congruence on Y,i.e., for all $y, y' \in Y$,

$$P_\nu \circ P_\rho(y) = P_\nu \circ P_\rho(y') \Rightarrow (\forall \alpha \in \Sigma) P_\nu \circ P_\rho\eta(y,\alpha) = P_\nu \circ P_\rho\eta(y',\alpha).$$

Now

$$\begin{aligned}
& P_\nu \circ P_\rho(y) = P_\nu \circ P_\rho(y') \\
\Rightarrow\ & P_\nu(P_\rho(y)) = P_\nu(P_\rho(y')) \\
\Rightarrow\ & P_\nu(\overline{\eta}(P_\rho(y)),\alpha) = P_\nu(\overline{\eta}(P_\rho(y')),\alpha) \\
& \text{(cf. (ii) of Proposition 2.5.1 in [21])} \\
\Rightarrow\ & P_\nu(\overline{\eta}(\overline{y},\alpha)) = P_\nu(\overline{\eta}(\overline{y'},\alpha)) \\
\Rightarrow\ & P_\nu(P_\rho(\eta'(y,\alpha))) = P_\nu(P_\rho(\eta'(y',\alpha))) \\
\Rightarrow\ & P_\nu \circ P_\rho\eta'(y,\alpha) = P_\nu \circ P_\rho\eta'(y',\alpha)
\end{aligned}$$

Hence, $\nu \circ \rho$ is a quasi-congruence on Y. Obviously, $\nu \circ \rho$ is coarser than ρ. However, ρ is the supremal quasi-congruence on Y, so for any $y, y' \in Y$, if $P_\nu(P_\rho(y)) = P_\nu(P_\rho(y'))$, i.e., $(y,y') \in \nu \circ \rho$, then $(y,y') \in \rho$, which means that $P_\rho(y) = P_\rho(y')$. Hence, $\nu = \perp$ (namely all its cells are singletons).

We have shown that $\mathbf{QCSUP'}$ is a canonical generator. □

B Delay-Robustness of Decentralized Controllers

Here we show that the verification tool for delay-robustness of distributed controllers can be used without change to verify the delay-robustness of decentralized supervisors.

Let $\mathbf{G}$ be the DES to be controlled, and $\mathbf{LOC}_1$ and $\mathbf{LOC}_2$ be two decentralized controllers, which achieve global supervision with zero-delay communication. Let Σ_i, Σ_{io} be the event set and observable event set of $\mathbf{LOC}_i$, respectively $(i = 1, 2)$. Assume event $r \in \Sigma_1 \cap (\Sigma_{2o} - \Sigma_{1o})$, which is not observed by $\mathbf{LOC}_1$, but is observed by $\mathbf{LOC}_2$. Hence, r should be transmitted to $\mathbf{LOC}_1$. We use the channel $\mathbf{CH}(2, r, 1)$, as shown in Fig. 1, to transmit r and use r' to represent that $\mathbf{LOC}_1$ receives the occurrence of r. Then, replacing r by r', we obtain $\mathbf{LOC}_1'$. Let $\mathbf{SUP} = Sync(\mathbf{G}, \mathbf{LOC}_1, \mathbf{LOC}_2)$, $\mathbf{SUP'} = Sync(\mathbf{G}, \mathbf{LOC}_1', \mathbf{CH}(2, r, 1), \mathbf{LOC}_2)$, and $\mathbf{QCSUP'} = Supqc(\mathbf{SUP'}, Null[r'])$. Finally, by Theorem 1, if $\mathbf{SUP} \approx \mathbf{QCSUP'}$, $\mathbf{SUP}$ is delay-robust with respect to r, or $\mathbf{LOC}_1$ and $\mathbf{LOC}_2$ achieve global supervision with unbounded delay communication.

C Proof of Theorem 2

The relevant natural projections are

$$P' : (\Sigma_1 \cup \{\alpha', \beta'\} \cup \Sigma_{E'})^* \to \Sigma^*$$
$$P'' : (\Sigma_1 \cup \{\beta'\} \cup \Sigma_{E'})^* \to \Sigma^*.$$

Thus P' (resp. P'') nulls $\{\alpha', \beta'\}$ (resp. $\{\beta'\}$) $\cup \{r' | r' \in \Sigma_{E'}\}$.

For the proof we assume that

$$P'L(\mathbf{SUP'}) = L(\mathbf{SUP}) \tag{24a}$$
$$P'L_m(\mathbf{SUP'}) = L_m(\mathbf{SUP}) \tag{24b}$$
$$P' \text{ has the observer property with respect to } \mathbf{SUP'} \text{ and } \mathbf{SUP}. \tag{24c}$$

It must be shown that the counterpart properties hold for P'' and $\mathbf{SUP''}$, namely

$$P''L(\mathbf{SUP''}) = L(\mathbf{SUP}) \tag{25a}$$
$$P''L_m(\mathbf{SUP''}) = L_m(\mathbf{SUP}) \tag{25b}$$
$$P'' \text{ has the observer property with respect to } \mathbf{SUP''} \text{ and } \mathbf{SUP}. \tag{25c}$$

We need the following lemmas.

Lemma 1 *(α' insertion) Let $s'' = x.\alpha.x.\beta.x.\beta'.x \in L(\mathbf{SUP''})$ where the (generally distinct) strings written x are free of α, β, β'. Then $s' := x.\alpha.\alpha'.x.\beta.x.\beta'.x \in L(\mathbf{SUP'})$.*

Proof. Immediate from the definition of the relevant synchronous products. □

Evidently Lemma 1 extends to multiple appearances of α, β, β' and arbitrary possible orderings of the α with respect to the β, β'; and holds with L replaced by L_m throughout.

Lemma 2 *(α' deletion) Let $t' = x.\alpha.y.\alpha'.z.\beta.z.\beta'.z \in L_m(\mathbf{SUP'})$, where the strings x, y, z are free of α, α', β, β'. Then $t'' := x.\alpha.y.z.\beta.z.\beta'.z \in L_m(\mathbf{SUP''})$.*

Proof. Recall that the synchronous products defining $L_m(\mathbf{SUP'})$ and $L_m(\mathbf{SUP''})$ differ only in that the latter omits the factor $\mathbf{CH}(E, \alpha, 1)$, and in $\mathbf{SUP}_1''$ α appears as in $\mathbf{SUP}_1$ (and not as α'). The string y is of form, say $a_1.b_1.a_2.b_2$, where $a_1, a_2 \in (\Sigma_1')^*$ and $b_1, b_2 \in \Sigma_{E'}^*$, hence by definition of synchronous product can be re-ordered as $a_1.a_2.b_1.b_2$ without affecting membership of t' in $L_m(\mathbf{SUP'})$; next $\alpha.y$ can be re-ordered in t' as $a_1.a_2.\alpha.b_1.b_2$, and then $\alpha.y.\alpha'$ can be re-ordered as $a_1.a_2.\alpha.\alpha'.b_1.b_2$, again preserving membership of t' in $L_m(\mathbf{SUP'})$. In this new ordering it is clear that deletion of α' converts t' to a string t'' in $L_m(\mathbf{SUP''})$. Reversing the ordering restores our original t'', proving the claim. □

Proof of Theorem 2. For (25a) suppose $s'' = x.\alpha.x.\beta.x.\beta'.x \in L(\mathbf{SUP''})$. By Lemma 1, $s' := x.\alpha.\alpha'.x.\beta.x.\beta'.x \in L(\mathbf{SUP'})$, so by (24a) $P'(s') \in L(\mathbf{SUP})$. Evidently $P''(s'') = P'(s')$ as required. For the reverse inclusion, if $s = x.\alpha.x.\beta.x \in L(\mathbf{SUP})$ then applying Lemma 1 to s with β we get that $s'' = x.\alpha.x.\beta.\beta'.x \in L(\mathbf{SUP''})$ and then $s = P''(s'')$, as claimed. The argument for (25b) is similar. For the observer property we have by (24c) that

$$(\forall s' \in L(\mathbf{SUP'}))(\forall v \in \Sigma^*)P'(s').v \in L_m(\mathbf{SUP}) \Rightarrow$$
$$(\exists v' \in (\Sigma')^*)s'.v' \in L_m(\mathbf{SUP'}) \;\&\; P'(v') = v$$

and must verify the counterpart (25c), namely

$$(\forall s'' \in L(\mathbf{SUP}''))(\forall v \in \Sigma^*)P''(s'').v \in L_m(\mathbf{SUP}) \Rightarrow \\ (\exists v'' \in (\Sigma'')^*)s''.v'' \in L_m(\mathbf{SUP}'') \ \&\ P''(v'') = v.$$

For the proof let $s'' \in L(\mathbf{SUP}'')$, $v \in \Sigma^*$, $P''(s'').v \in L_m(\mathbf{SUP})$. By Lemma 1 with α'-insertion we obtain $s' \in L(\mathbf{SUP}')$ such that $P'(s') = P''(s'')$, so $P'(s').v \in L_m(\mathbf{SUP})$, and by (24c) there is $v' \in (\Sigma')^*$ with $s'.v' \in L_m(\mathbf{SUP}')$ and $P'(v') = v$. Thus v' is of the form $v' = y.\alpha.y.\alpha'.y.\beta.y.\beta'.y$ (possibly with multiple α's and β's in various interleavings). Define $v'' = Q(v')$ where Q projects α' to the empty string ϵ. Then $P''(v'') = P''Q(v') = P'(v') = v$. Also, by Lemma 2, $s''.v'' = Q(s'.v') \in QL_m(\mathbf{SUP}') \subseteq L_m(\mathbf{SUP}'')$. Thus v'' has the properties required in (25c), which completes the proof. □

Acknowledgements This work was supported in part by the National Nature Science Foundation of China, Grant no. 61403308; the Fundamental Research Funds for the Central Universities, China, Grant no. 3102014JCQ01069; the Program to Disseminate Tenure Tracking System, MEXT, Japan; the Natural Sciences and Engineering Research Council, Canada, Grant no. 7399.

References

1. R. Su and J.G. Thistle. A distributed supervisor synthesis approach based on weak bisimulation. In *Proc. 8th International Workshop on Discrete-Event Systems (WODES'06)*, pages 64–69, Ann Arbor, MI, July 2006.
2. A. Mannani and P. Gohari. Decentralized supervisory control of discrete-event systems over communication networks. *IEEE Trans. on Automatic Control*, 53(2):547–559, March 2008.
3. P. Darondeau. Distributed implementation of Ramadge-Wonham supervisory control with Petri nets. In *Proc. 44th IEEE Conference on Decision and Control and 2005 European Control Conference. CDC-ECC'05*, pages 2107–2112, Seville, Spain, December 2005.
4. K. T. Seow, M. T. Pham, C. Ma, and M. Yokoo. Coordination planning: applying control synthesis methods for a class of distributed agents. *IEEE Trans. on Control Systems Technology*, 17(2):405–415, March 2009.
5. K. Cai and W. M. Wonham. Supervisor localization: a top-down approach to distributed control of discrete-event systems. *IEEE Trans. on Automatic Control*, 55(3):605–618, March 2010.
6. K. Cai and W.M. Wonham. Supervisor localization for large discrete-event systems: case study production cell. *International J. of Advanced Manufacturing Technology*, 50(9-12):1189–1202, October 2010.
7. G. Kalyon, T. Le Gall, H. Marchand, and T. Massart. Synthesis of communicating controllers for distributed systems. In *2011 50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC)*, Orlando, FL, USA, December 2011.
8. F. Lin. Control of networked discrete event systems: dealing with communication delays and losses. *SIAM J. Control and Optimization*, 52(2):1276–1298, 2014.
9. P. Darondeau and L. Ricker. Distributed control of discrete-event systems: A first step. *Transactions on Petri Nets and Other Models of Concurrency*, 6:24–45, 2012.
10. M. Yeddes, H. Alla, and R. David. On the supervisory synthesis for distributed control of discrete event dynamic systems with communication delays. In *Proc. 1999 IEEE International Symposium on Intelligent Control/Intelligent Systems and Semiotics*, pages 1–6, Cambridge, MA, September 1999.
11. G. Barrett and S. Lafortune. Decentralized supervisory control with communicating controllers. *IEEE Trans. on Automatic Control*, 45(9):1620–1638, September 2000.
12. S. Tripakis. Decentralized control of discrete-event systems with bounded or unbounded delay communication. *IEEE Trans. on Automatic Control*, 49(9):1489–1501, September 2004.
13. K. Schmidt, E.G. Schmidt, and J. Zaddach. A shared-medium communication architecture for distributed discrete event systems. In *Proc. Mediterranean Conf. on Control and Automation*, pages 1–6, Athens, Greece, 2007.

14. K. Schmidt and E.G. Schmidt. Communication of distributed discrete-event supervisors on a switched network. In *Proc. 9th Int. Workshop on Discrete Event Systems (WODES'08)*, pages 419–424, Goteborg, Sweden, 2008.
15. S. Xu and R. Kumar. Asynchronous implementation of synchronous discrete event control. In *Proc. 9th Int. Workshop on Discrete Event Systems (WODES'08)*, pages 181–186, 2008.
16. K. Hiraishi. On solvability of a decentralized supervisory control problem with communication. *IEEE Trans. on Automatic Control*, 54(3):468–480, March 2009.
17. L. Ricker and B. Caillaud. Mind the gap: expanding communication options in decentralized discrete-event control. *Automatica*, 47(11), 2011.
18. R. Zhang, K. Cai, Y. Gan, Z. Wang, and W. M. Wonham. Checking delay-robustness of distributed supervisors of discrete-event systems. In *Proc. Int. Conf. on Information Science and Control Engineering*, pages 350–355, Shenzhen, China, 2012.
19. P.J. Ramadge and W.M. Wonham. Supervisory control of a class of discrete event processes. *SIAM J. Control and Optimization*, 25(1):206–230, January 1987.
20. W. M. Wonham and P. J. Ramadge. On the supremal controllable sublanguage of a given language. *SIAM J. Control and Optimization*, 25(3):637–659, May 1987.
21. W.M. Wonham. *Supervisory Control of Discrete-Event Systems*. Systems Control Group, ECE Dept, Univ. Toronto, Toronto, ON, Canada, July 2014. Available at `http://www.control.utoronto.ca/DES`.
22. L. Feng, K. Cai, and W. M. Wonham. A structural approach to the nonblocking supervisory control of discrete-event systems. *International J. of Advanced Manufacturing Technology*, 41(11):1152–1167, 2009.
23. K. Wong and W. M. Wonham. Modular control and coordination of discrete-event systems. *Discrete Event Dynamic Systems*, 8(3):247–297, October 1998.
24. R. Hill and D. Tilbury. Modular supervisory control of discrete-event systems with abstraction and incremental hierarchical construction. In *Proc. 8th International Workshop on Discrete-Event Systems (WODES'06)*, pages 399–406, Ann Arbor, MI, July 2006.
25. R. Su, Jan H. van Schuppen, and Jacobus E. Rooda. Aggregative synthesis of distributed supervisors based on automaton abstraction. *IEEE Trans. on Automatic Control*, 55(7):1627–1640, July 2010.
26. J.Y. Udding. A formal model for defining and classifying delay-insensitive circuits and systems. *Distributed Computing*, 1:197–204, 1986.
27. H. Zhang. Delay Insensitive Networks. Master of math. thesis, Computer Science Dept., University of Waterloo, Waterloo, ON, Canada, 1997.
28. M. Kishinevsky and J. Cortadella. Synchronous elastic systems. Tutorial presented at the ASYNC08/NOCS08 in Newcsstle, UK., 2008. Available at `http://async.org.uk/async2008/async-nocs-slides/Tutorial-Monday/Mike_tutorial.pdf`.
29. K.C. Wong and W.M. Wonham. On the computation of observers in discrete-event systems. *Discrete Event Dynamic Systems*, 14(1):55–107, January 2004.
30. L. Feng and W.M. Wonham. On the computation of natural observers in discrete-event systems. *Discrete Event Dynamic Systems*, 20(1):63–102, March 2010.
31. W.M. Wonham. *Design Software: XPTCT*. Systems Control Group, ECE Dept, Univ. Toronto, Toronto, ON, Canada, July 2014. Available at `http://www.control.utoronto.ca/DES`.
32. S.-J. Park and K.-H. Cho. Decentralized supervisory control of discrete event systems with communication delays based on conjunctive and permissive decision structures. *Automatica*, 43(4):738–743, April 2007.
33. R. Zhang, K. Cai, Y. Gan, and W.M. Wonham. Distributed supervisory control of discrete-event systems with communication delay. Available at `http://arxiv.org/abs/1207.5072`.
34. R. Milner. *Communication and Concurrency*. Prentice Hall, Englewood Cliffs, NJ, 1989.
35. H.J. Bravo, A.E.C. da Cunha, P.N. Pena, R. Malik, and J.E.R. Cury. Generalised verification of observer property in discrete event systems. In *Proc. 11th International Workshop on Discrete Event Systems(WODES'12)*, pages 337–342, Guadalajara, Mexico, October 2012.
36. C. Ma and W. M. Wonham. *Nonblocking Supervisory Control of State Tree Structures*. Springer-Verlag, 2005.