

# COMMUNICATION DELAYS IN CONNECTIONS OF INPUT/OUTPUT DISCRETE EVENT PROCESSES

S. Balemi

Automatic Control Laboratory, Swiss Federal Institute of Technology (ETH), 8092 Zürich, Switzerland

## Abstract

This paper proposes an input/output interpretation of supervision of discrete event systems. New concepts must be introduced in order to describe the behavior of the closed-loop system of plant and supervisor subject to communication delays. This paper further characterizes the class of supervisors allowed under delays for the general case and for additional assumptions on the plant language.

## 1. Introduction

Discrete Event Processes (DEP) model a large variety of practical systems, *e.g.* manufacturing systems and computer networks. The behavior of these systems is naturally described by records (called *traces*) of certain qualitative changes (*events*) in the system.

Ramadge and Wonham [1] introduced a language-theoretical approach on control of DEP by considering the possible traces that can be executed by a system to be strings on an alphabet of symbols representing the events. The set of all such strings is called a *language*, and represents the possible behavior of the system. We denote the alphabet of symbols by  $\Sigma$ . The symbol  $\Sigma^*$  stands for the set of all finite sequences (or strings) over  $\Sigma$ . For a language  $L \subseteq \Sigma^*$ ,  $\bar{L}$  denotes the set of prefixes of strings in  $L$ . We say a language  $L$  is *prefix-closed* if  $L = \bar{L}$ . The string  $\text{proj}[\Sigma'](s)$  is obtained by removing from  $s$  all symbols not in the alphabet  $\Sigma'$ . The expression  $\text{proj}[\Sigma'](L)$  denotes the obvious extension to languages. Its converse, for a given alphabet  $\Sigma' \subseteq \Sigma$  and a language  $L' \subseteq (\Sigma')^*$  is  $\text{proj}^{-1}[\Sigma, \Sigma'](L') = \sup\{L \subseteq \Sigma^* \mid \text{proj}[\Sigma'](L) = L'\}$ . The notation “sup” indicates the supremum with respect to set inclusion.

### 1.1. Process Model and Process Composition

A process is a triple  $P = (\Sigma, L_P, M_P)$  composed of two languages  $L_P$  and  $M_P$  of finite strings over  $\Sigma$ ;  $L_P$  is prefix-closed and represents all partial traces of  $P$  while  $M_P \subseteq L_P$ , is a set of distinguished traces, the *marked language*. Often  $M_P$  marks the set of successfully *completed* traces. An important operator on processes is the synchronous composition. The synchronous composition  $P_1 \parallel P_2$  of two processes  $P_1 = (\Sigma_1, L_{P_1}, M_{P_1})$  and  $P_2 = (\Sigma_2, L_{P_2}, M_{P_2})$  is defined by  $P_1 \parallel P_2 = (\Sigma, L_{P_1 \parallel P_2}, M_{P_1 \parallel P_2})$  where  $\Sigma = \Sigma_1 \cup \Sigma_2$  and

$$\begin{aligned} L_{P_1 \parallel P_2} &= \text{proj}^{-1}[\Sigma, \Sigma_1](L_1) \cap \text{proj}^{-1}[\Sigma, \Sigma_2](L_2) \\ M_{P_1 \parallel P_2} &= \text{proj}^{-1}[\Sigma, \Sigma_1](M_1) \cap \text{proj}^{-1}[\Sigma, \Sigma_2](M_2). \end{aligned}$$

In a synchronous composition, the processes must always execute simultaneously a common event (in  $\Sigma_1 \cap \Sigma_2$ ), while the other events can happen independently in each process. The synchronous composition is commutative and associative.

### 1.2. Supervisory Control Theory

The basic problem of supervisory control introduced by Ramadge and Wonham consists in modifying the open-loop behavior of a plant by eliminating traces from the plant behavior. For a plant process  $P = (\Sigma, L_P, M_P)$ , this is achieved by full synchronization of  $P$  with another process  $S = (\Sigma, L_S, M_S)$  called supervisor. In the synthesis procedure, we are free to choose the supervisor while the plant process is given and cannot be altered. The composition of  $P$  with  $S$  is then  $P \parallel S = (\Sigma, L_P^c, M_P^c)$ .  $L_P^c$  is called the closed-loop language,  $M_P^c$  the marked closed-loop language and  $P \parallel S$  the *supervised* plant process.

In this paper we assume an input/output nature of the plant and the supervisor. The alphabet  $\Sigma$  of the process plant  $P$  is divided into two disjoint sets:  $\Sigma_S$ , the commands and  $\Sigma_P$ , the responses. The plant receives commands as inputs and produces responses as outputs. Symmetrically, the supervisor accepts responses as inputs and produces commands as outputs. The synchronous composition of  $P$  and  $S$  can be regarded as the closed-loop connection of two input/output processes (see figure 1). The supervisor reads at the input the responses produced by the plant, whereas the plant reads at the input the commands produced by the supervisor.

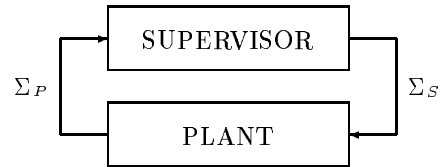


Figure 1: Symmetric feedback loop of plant and supervisor

An important assumption is that the plant can arbitrarily force the communication of responses (the events in  $\Sigma_P$ ) toward the input of the supervisor, and that the supervisor will always be able to accept these responses. This is expressed by the condition

$$P \parallel S = P \parallel (\Sigma, L_S, \Sigma_P^*, M_S) \quad (1)$$

that a supervisor has to satisfy for a particular plant  $P$ . We say, the supervisor is *complete* for the plant.

The input/output nature of the connection of plant and supervisor leads to the similar assumption that the plant cannot prevent the occurrence of commands produced by the supervisor and that any command from the supervisor must be accepted by the plant. A supervisor satisfies this additional assumption if the condition

$$(\Sigma, L_P, \Sigma_S^*, M_P) \parallel S = P \parallel S \quad (2)$$

holds. Equation (2) can be seen as a *dual* to equation (1); the plant is then said to be *complete* for the supervisor. The composition of a supervisor and a plant satisfying equations (1) and (2) is called *well-posed* [2].

In addition, the class of supervisors is further restricted to those also guaranteeing termination of a marked string. For this, Ramadge and Wonham [3] introduced the concept of non-blockingness. A process  $P$  is non-blocking if

$$L_P = \overline{M_P}. \quad (3)$$

We are now ready to present the input/output supervisory synthesis problem [2, 4].

### Input/Output Supervisory Synthesis Problem:

Given a plant  $P$  and a specification language  $L_{\text{spec}} \subseteq \Sigma_P^*$  for the closed-loop behavior, find a supervisor  $S$  such that

1.  $\emptyset \subset \text{proj}[\Sigma_P](M_S^c) \subseteq L_{\text{spec}}$ ,
2. The connection of  $P$  and  $S$  is well-posed
3.  $S \parallel P$  is non-blocking.

The reader could object that a more general specification would be  $M_P^c \subseteq L'_{\text{spec}}$  for some  $L'_{\text{spec}} \subseteq \Sigma^*$ . However, we believe it is not a relevant restriction in an input/output semantics. In fact, commands represent the request of some desired change, and we impose no constraints on simple requests. In order to model *e.g.* the limited availability of some resource, additional responses can be embedded in the language of the plant at modeling time. Moreover, the restriction to this class of specifications will be of importance later in dealing with communication delays.

**Controllability:** In order to characterize the supervisors solving the input/output supervisory synthesis problem, we use the concept of controllability introduced by Ramadge and Wonham [1]. A language  $K \subseteq \Sigma^*$  is called  $[L_P, \Sigma_P]$ -controllable if the inclusion  $\overline{K} \cdot \Sigma_P \cap L_P \subseteq \overline{K}$  holds. The class  $\mathcal{C}(L)$  of  $[L_P, \Sigma_P]$ -controllable sublanguages of a language  $L$  is closed under language union; therefore there exists a *supremal* element of  $\mathcal{C}(L)$ , denoted by  $\text{sup } \mathcal{C}(L)$ . In [2] it is shown that the input/output supervisory synthesis problem has a solution if and only if for the language

$$M_S = \text{sup } \mathcal{C}(M_P \cap \text{proj}^{-1}[\Sigma, \Sigma_P](L_{\text{spec}}))$$

$\text{proj}[\Sigma_P](M_S)$  is non-empty. If the problem has a solution, the particular supervisor  $S_{\text{sup}} = (\Sigma, L_S, M_S)$  with  $L_S = \overline{M_S}$  is a solution. Moreover, it is the least restrictive supervisor in the sense that it does not prevent any trace that is allowed by any other supervisor solving the problem.

We note that technically speaking, the input/output supervisory synthesis problem stated above without the completeness condition on the plant corresponds to the supervisory control problem introduced by Ramadge and Wonham [1]. The condition for existence of a solution is the same. Moreover, as shown in [4], any supervisor  $S$  solving the supervisory control problem of Ramadge and Wonham and satisfying  $L_S \subseteq \overline{M_P}$  (and therefore the additional condition (2)) is also a solution to the input/output supervisory synthesis problem. Finally, if  $S$  solves the supervisory control problem but  $L_S \not\subseteq \overline{M_P}$ , the supervisor  $S' = S \parallel P$  is a solution to the input/output supervisory synthesis problem.

## 2. Supervisor-Plant Connection with Delays

Until now we have assumed that an event message sent by the plant instantaneously reaches the supervisor and vice-versa. In general, however, the connection between the plant and the supervisor is affected by delays. A message sent by a process needs some time to reach the other process.

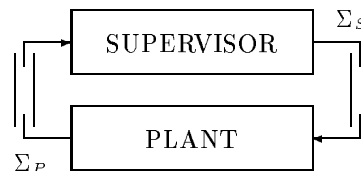


Figure 2: Closed-loop connection of processes  $P$  and  $S$  with communication delays.

In our model we introduce communication delays by considering the two processes as being connected by two communication channels, one from the output of the plant  $P$  to the input of the supervisor  $S$  and one from the output of the supervisor  $S$  to the input of the plant  $P$  (see figure 2). The messages put by the process into the respective channel appear at the exit with some delay. However, the relative order of the transmitted messages is unaffected: they exit the channel in the same order as they were put in. The channels can therefore be seen as FIFO buffers with a given buffering length.

### 2.1. Processes with Prefix-Closed Languages

We first restrict our attention to processes whose marked languages are prefix-closed, and therefore to processes of the form  $P = (\Sigma, L_P, L_P)$ , respectively  $S = (\Sigma, L_S, L_S)$ . We also assume that the channels have infinite buffering length: an infinite number of messages can be put into a channel without any exit at the channel output.

In order to characterize the behavior of the two processes in the closed-loop, we must first choose a suitable behavior representation. One possibility is register all the messages that go into and come from the communication channels. A command  $\sigma_S$  put into the channel connecting the supervisor to the plant will eventually be registered exiting at the other end of the channel. A set of sequences of the form

$$\sigma_1^{\text{enter}} \cdot \sigma_2^{\text{enter}} \cdot \sigma_1^{\text{exit}} \cdot \sigma_3^{\text{enter}} \cdot \sigma_2^{\text{exit}} \cdot \dots$$

therefore uniquely determines the behavior of the closed-loop system. This set of sequences can be seen as the language of a new process of the form  $X = (\Sigma^{\text{exit}} \cup \Sigma^{\text{enter}}, L_X, L_X)$ , where all the events ( $\Sigma^{\text{exit}}$  and  $\Sigma^{\text{enter}}$ , the set of event messages exiting resp. entering the channels) represent only outputs (no external input can be given to this new process). It would be thus possible to formally introduce a “delayed composition” of two processes  $S$  and  $P$  resulting in a new process  $X$ . We abstain from that however, and we restrict our attention to a simpler description.

The description we choose uses the view of two observers registering locally the inputs and outputs of the plant and of the supervisor respectively. This choice is suggested by the fact that the supervisor has to influence the behavior of the plant only with the knowledge of the events that it has processed so far, and that it has to infer from that knowledge

the possible behavior of the plant. We note that in opposition to the case of delay-free communication, the behavior of the plant and the behavior of the supervisor are not identical. Then, we will have to define two local closed-loop languages,  $L_P^c$  for the plant and  $L_S^c$  for the supervisor.

A key concept that we will use to describe the behavior of the supervisor and of the plant in the closed-loop subject to communication delays is the delay of a language.

**Definition 1 (Delay of a Language)**

The delay of a language  $L \subseteq \Sigma^*$  with respect to the subalphabet  $\Sigma' \subseteq \Sigma$ , denoted by  $\text{delay}[\Sigma'](L)$ , is defined as the smallest superlanguage of  $L$  such that for all  $s, t \in \Sigma^*$  and  $\sigma' \in \Sigma'$  and  $\sigma \in \Sigma \setminus \Sigma'$  the following conditions are satisfied

1.  $L \subseteq \text{delay}[\Sigma'](L)$
2.  $s.\sigma'.\sigma.t \in \text{delay}[\Sigma'](L) \Rightarrow s.\sigma.\sigma'.t \in \text{delay}[\Sigma'](L)$

For an interpretation of the delay of a language, we consider an observer sending inputs in  $\Sigma \setminus \Sigma'$  to a process and at the same time observing the responses  $\Sigma'$  coming through a channel at the output of the process (see figure 3).



Figure 3: Process with communication channel at output.

We assume that if an input given to the process can not be accepted, it is not registered, and no further input will be possible. Clearly, the responses will be observed at the channel exit with some delay due to the transmission in the channel. The effect is to shift to the right the position of the elements from  $\Sigma'$  in the strings contained in the language of the process.

Moreover, the delay operator allows us to characterize the strings that the processes  $P$  and  $S$  in the closed-loop connection subject to communication delays have processed at a given time. Let us consider a particular instant: the plant  $P$  has processed string  $s_P$ , and the supervisor  $S$  has processed string  $s_S$ . Then,  $s_P$  and  $s_S$  must satisfy the following relation.

$$\begin{aligned} s_S &\in L_S \cap \overline{\text{delay}[\Sigma_P](s_P.\Sigma_P^*)} \\ s_P &\in L_P \cap \overline{\text{delay}[\Sigma_S](s_S.\Sigma_S^*)}. \end{aligned} \tag{4}$$

The terms  $\Sigma_P^*$  and  $\Sigma_S^*$  take into account the event messages that are present at that particular instant in the channels.

**2.2. Closed-Loop Languages**

We define the closed-loop languages  $L_P^c$  and  $L_S^c$  as the set of all elements  $s_P$  resp.  $s_S$  of pairs  $(s_P, s_S)$  satisfying (4).

We are now interested in generalizing our concept of well-posedness presented previously to include communication delays. A connection of processes  $P$  and  $S$  with communication delays is well-posed if the connection of the corresponding processes  $P'$  and  $S'$  shown in figure 4 is well-posed.

The following theorem from [5] gives conditions on the open-loop languages of the plant and the supervisor for the connection to be well-posed.

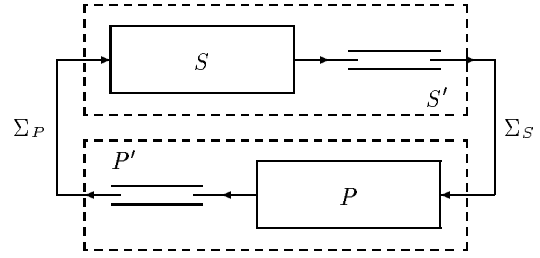


Figure 4: Equivalence of well-posedness of process connections with and without communication delays

**Theorem 1:** A composition of processes  $P$  and  $S$  subject to communication delays is well-posed if and only if

$$L_S \supseteq L_S.\Sigma_P^* \cap \text{delay}[\Sigma_P](L_P.\Sigma_S^*) \tag{5}$$

$$L_P \supseteq L_P.\Sigma_S^* \cap \text{delay}[\Sigma_S](L_S.\Sigma_P^*). \tag{6}$$

We remark that the condition on well-posedness given by equations (5) and (6) is similar to the conditions (1) and (2) characterizing well-posedness for a delay-free composition. In fact, equations (1) and (2) can be reformulated as

$$L_S \supseteq L_S.\Sigma_P^* \cap L_P \quad \text{and} \quad L_P \supseteq L_P.\Sigma_S^* \cap L_S.$$

The closed-loop languages are given as the result of the two implicit equations (4). The next theorem provides an explicit expression for  $L_P^c$  and  $L_S^c$ .

**Theorem 2:** The closed-loop languages resulting from the well-posed composition of two processes  $P$  and  $S$  subject to communication delays are given by

$$L_P^c = \overline{L_P \cap \text{delay}[\Sigma_S](L_S)} \tag{7}$$

$$L_S^c = \overline{L_S \cap \text{delay}[\Sigma_P](L_P)}. \tag{8}$$

In the sequel we will only consider supervisors such that the connection with the given plant is well-posed. Then, we are interested in knowing if the union of the languages of two such supervisors is the language of a supervisor that has a well-posed connection with the given plant. This property for supervisors would be very useful, as it allows to define an optimal, i.e. a “largest” supervisor.

**Proposition 1:** The class of languages  $L_S$  of supervisors  $S = (\Sigma, L_S, L_S)$  such that the delayed composition of  $S$  with a plant  $P = (\Sigma, L_P, L_P)$  is well-posed is closed under union.

The proposition implies directly that the class  $\mathcal{W}[L_P, \Sigma_S](L)$  of sublanguages of  $L$  for supervisors guaranteeing well-posedness of the connection with a plant having language  $L_P$  is also closed under union of languages. Therefore, there exists a supremal element of  $\mathcal{W}[L_P, \Sigma_S](L)$ , which we denote by  $\text{sup}\mathcal{W}[L_P, \Sigma_S](L)$ .

We are now ready to state the unmarked supervisor synthesis problem with delays.

**Unmarked Sup. Synthesis Problem with Delays:**

Given a plant  $P = (\Sigma, L_P, L_P)$  and a prefix-closed specification language  $L'_{\text{spec}} = \overline{L_{\text{spec}}} \subseteq \Sigma_P^*$  for the plant behavior in the closed-loop subject to communication delays. Find a supervisor  $S = (\Sigma, L_S, L_S)$  such that

1.  $L_S \subseteq L_P$
2.  $\emptyset \subset \text{proj}[\Sigma_P](L_P^c) \subseteq L_{\text{spec}}^c$ ,
3. the connection of  $S$  and  $P$  is well-posed.

Note that we have no condition for non-blockingness as the languages of both plant and supervisor are prefix-closed. Also note that we have the new constraint  $L_S \subseteq L_P$ . This assumption is justified by the wish to have with the supervisor the best possible image of the behavior of the plant.

While it is difficult to give conditions on the language of a supervisor that enforces a global specification, this is very easy if the specification is local, as stated in the next proposition.

**Proposition 2:** Consider a language  $L_{\text{spec}} \subseteq \Sigma_P^*$ , a plant  $P = (\Sigma, L_P, L_P)$  and a supervisor  $S = (\Sigma, L_S, L_S)$  such that the connection of  $P$  and  $S$  is well-posed. If  $\text{proj}[\Sigma_P](L_S) \subseteq L_{\text{spec}}$  then  $\text{proj}[\Sigma_P](L_P^c) \subseteq L_{\text{spec}}^c$ .

Let us suppose that we have already a supervisor  $S = (\Sigma, L_S, L_S)$  and that its connection with the plant  $P$  is well-posed, but that a given local specification  $L_{\text{spec}}$  on the closed-loop language of the plant  $L_P^c$  is not satisfied.

From the previous proposition we know that a supervisor  $S'$ , further restricting the possible commands in  $\Sigma_S$  sent to  $P$  until  $\text{proj}[\Sigma_P](L_S') \subseteq L_{\text{spec}}$  holds, guarantees the enforcement of the specification  $L_{\text{spec}}$  on the plant. As we restrict only the commands, the language  $L_S'$  is  $[L_S, \Sigma_S]$ -controllable. Then we hope that this new supervisor  $S'$  has also a well-posed composition with  $P$ . The next proposition shows this.

**Proposition 3:** Consider  $P = (\Sigma, L_P, L_P)$  and  $S = (\Sigma, L_S, L_S)$  such that their composition is well-posed. The composition of  $P$  with a supervisor  $S' = (\Sigma, L_S', L_S')$  with  $L_S' \subseteq L_S \subseteq L_P$  is well-posed if and only if the language  $L_S'$  is  $[L_S, \Sigma_S]$ -controllable.

This proposition allows us to state the condition for existence of a solution of the unmarked supervisor synthesis problem with delays.

**Theorem 3:** The unmarked supervisor synthesis problem with delays has a solution if and only if for the language

$$L_S = \sup \mathcal{C}(\sup \mathcal{W}[L_P, \Sigma_S](L_P) \cap \text{proj}^{-1}[\Sigma, \Sigma_P](L_{\text{spec}}^c)).$$

we have  $\emptyset \subset \text{proj}[\Sigma_P](L_S)$ . If a solution exists, the supervisor with language  $L_S$  is a solution.

We note, however, that the computation of the language  $\sup \mathcal{W}[L_P, \Sigma_S](L_P)$  is not easy, as the conditions for well-posedness are expressed in terms of non-regular languages also for a plant with regular language. We are therefore interested in special cases of plants for which the computation of a supervisor can be performed using known algorithms. For this we introduce the concept of self-well-posed languages.

### 2.3. Plants with Self-Well-Posed Languages

An important class of plants is that of plants of the form  $P = (\Sigma, L_P, L_P)$  for which the composition with a supervisor with the same language  $L_P$  is well-posed. We can see

this in figure 5 (note the slight abuse of notation: both the supervisor and the plant are denoted by the symbol  $P$ , although the output sets are different). The languages of plants in this class are called *self-well-posed*.

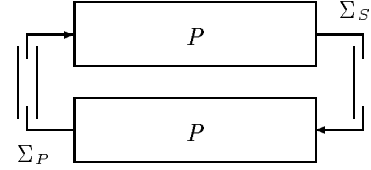


Figure 5: The language of a plant having well-posed composition with a supervisor having the plant's language is self-well-posed.

### Definition 2 (Self-Well-Posed Language)

A prefix-closed language  $L$  is self-well-posed with respect to the partition  $\Sigma = \Sigma_P \dot{\cup} \Sigma_S$  if the following inclusions hold:

$$L \supseteq L \cdot \Sigma_S^* \cap \text{delay}[\Sigma_S](L \cdot \Sigma_P^*) \quad (9)$$

$$L \supseteq L \cdot \Sigma_P^* \cap \text{delay}[\Sigma_P](L \cdot \Sigma_S^*). \quad (10)$$

Suppose the language  $L_P$  of plant  $P = (\Sigma, L_P, L_P)$  is self-well-posed. We have the following corollary to theorem 3.

**Corollary 1:** The unmarked supervisor synthesis problem with delays for a plant with self-well-posed language  $L_P$  has a solution if and only if for the language

$$L_S = \sup \mathcal{C}(L_P \cap \text{proj}^{-1}[\Sigma, \Sigma_P](L_{\text{spec}}^c)).$$

we have  $\emptyset \subset \text{proj}[\Sigma_P](L_S)$ . If a solution exists, the supervisor with language  $L_S$  is a solution.

The condition for the existence of a solution to the unmarked supervisor synthesis problem with delays is therefore the same as for the input/output supervisor synthesis problem (where all languages are prefix-closed). We note that the class of self-well-posed languages is not closed under union. In order to see this, consider  $\sigma \in \Sigma \setminus \Sigma_P$ ,  $\sigma' \in \Sigma_P$ ,  $L_1 = \{\sigma\}$ ,  $L_2 = \{\sigma'\}$ . Both  $L_1$  and  $L_2$  are self-well-posed, but  $L_1 \cup L_2$  is not, as the conditions stated in the definition of a self-well-posed language require  $\{\sigma, \sigma'\} \in L_1 \cup L_2$ .

Closedness under union, however, is true for the particular class of sublanguages of a language  $L$  which are also  $[L_P, \Sigma_S]$ -controllable.

**Proposition 4:** The class  $\mathcal{SC}[L_P, \Sigma_S](L)$  of self-well-posed and  $[L_P, \Sigma_S]$ -controllable sublanguages of  $L$  is closed under language union. There exists a supremal element of this class, denoted by  $\sup \mathcal{SC}[L_P, \Sigma_S](L)$

Suppose that the plant has not a self-well-posed language. While theorem 3 provided a necessary and sufficient condition for the existence of a solution of the unmarked supervisor synthesis problem with delays, the above proposition suggests an alternative *sufficient* characterization of the solution. Compute  $\sup \mathcal{SC}[L_P, \Sigma_S](L_P)$  and use this as the

language of a new fictitious plant. Then compute the supervisor for the new plant. The  $[L_P, \Sigma_S]$ -controllability of the above language guarantees that the closed-loop behavior is identical for the connection of the supervisor with both the new and the original plant. Another method is the direct computation of the supervisor for the original plant using the class of self-well-posed and  $[L_P, \Sigma_S]$ -controllable languages. We state the two solution methods in the next theorem.

**Theorem 4:** *The unmarked supervisor synthesis problem with delays has a solution if for the language*

$$\begin{aligned} L_S &= \sup \mathcal{C}(\sup \mathcal{SC}[L_P, \Sigma_S](L_P) \cap \text{proj}^{-1}[\Sigma, \Sigma_P](L'_{\text{spec}})) \\ &= \sup \mathcal{SC}[L_P, \Sigma_S](L_P \cap \text{proj}^{-1}[\Sigma, \Sigma_P](L'_{\text{spec}})) \end{aligned}$$

*we have  $\emptyset \subset \text{proj}[\Sigma_P](L_S)$ . If a solution is proven to exist, the supervisor with language  $L_S$  is a solution.*

Note that the language  $L_S$  described above does not yield in general the “largest” supervisor as the inclusion

$$L_S \subseteq \sup \mathcal{C}(\sup \mathcal{W}[L_P, \Sigma_S](L_P) \cap \text{proj}^{-1}[\Sigma, \Sigma_P](L'_{\text{spec}})).$$

is strict in general. Moreover, if  $\text{proj}[\Sigma_P](L_S)$  is empty, a solution to the synthesis problem may still exist.

#### 2.4. Memoryless Languages

We have now characterized the solution to the unmarked supervisor synthesis problem with delays by means of self-well-posed languages. The general computation of self-well-posed (and controllable) sublanguages of a given language is almost as difficult as the computation of the optimal solution given by theorem 3. However, if the language of the plant satisfies a property called memorylessness, the computation of the optimal solution can be efficiently performed using the concept of self-well-posed languages.

##### Definition 3 (Memoryless Language)

*A prefix-closed language  $L$  is memoryless with respect to the partition  $\Sigma = \Sigma_P \cup \Sigma_S$  if for every  $s, s' \in L$  such that  $s \in \text{delay}[\Sigma_P](\{s'\})$  and an arbitrary string  $t \in \Sigma^*$*

$$s.t \in L \Leftrightarrow s'.t \in L \quad (11)$$

N.B.: The definition is symmetric, *i.e.* the definition is equivalent to the one using  $s \in \text{delay}[\Sigma_S](\{s'\})$  instead of  $s \in \text{delay}[\Sigma_P](\{s'\})$ .

Memorylessness implies that the permutation of commands and responses does not affect the future evolution of the system. This property is satisfied for most real-world systems, as responses are in general the reaction to commands previously given. For a memoryless language we can state the next proposition.

**Proposition 5:** *A prefix-closed memoryless language  $L$  is self-well-posed if and only if for  $s \in L, t_P \in \Sigma_P^*$  and  $t_S \in \Sigma_S^*$*

$$\begin{aligned} s.t_P \in L, s.t_S \in L &\Rightarrow \quad (12) \\ s.\text{delay}[\Sigma_S](\{t_S.t_P\}) &= s.\text{delay}[\Sigma_P](\{t_P.t_S\}) \subseteq L \end{aligned}$$

From proposition 4 we know that if the language  $L_P$  is memoryless we can find  $\sup \mathcal{SC}(L_P, \Sigma_S)(L_P)$  by finding the

largest sublanguage  $L'_P$  of  $L_P$  satisfying  $L'_P.\Sigma_P \cap L_P \subseteq L'_P$  and equation (12). Therefore, the computation of  $\sup \mathcal{SC}(L_P, \Sigma_S)(L_P)$  can be performed with an algorithm of polynomial complexity for a memoryless language  $L_P$ .

A very important property of a memoryless language  $L_P$  is the fact that the language  $\sup \mathcal{W}(L_P, \Sigma_S)(L_P)$  is also self-well-posed, *i.e.* the language  $\sup \mathcal{W}(L_P, \Sigma_S)(L_P)$  is also the supremal self-well-posed and  $[L_P, \Sigma_S]$ -controllable sublanguage of  $L_P$ . We state this in the next theorem.

**Theorem 5:** *Given a memoryless language  $L_P$ ,*

$$\sup \mathcal{W}(L_P, \Sigma_S)(L_P) = \sup \mathcal{SC}(L_P, \Sigma_S)(L_P).$$

With help of this theorem, we can state the condition for existence of a solution of the unmarked supervisor synthesis problem when the plant has a memoryless language.

**Theorem 6:** *The unmarked supervisor synthesis problem with delays for a plant  $P$  with memoryless language  $L_P$  has a solution if and only if for the language*

$$L_S = \sup \mathcal{C}(\sup \mathcal{SC}[L_P, \Sigma_S](L_P) \cap \text{proj}^{-1}[\Sigma, \Sigma_P](L'_{\text{spec}}))$$

*we have  $\emptyset \subset \text{proj}[\Sigma_P](L_S)$ . If a solution exists, the supervisor with language  $L_S$  is a solution.*

### 3. Processes with Marked Languages

We have previously defined the concept of non-blockingness of a process. We now have to generalize this definition to include non-blockingness of a process in a composition with delays, where the process considered corresponds to the local view that an observer has when looking at the plant or at the supervisor only. A process is non-blocking in a well-posed composition of two processes if any string in its closed-loop language can always be completed to a string in its marked language. We first note that the future evolution of a closed-loop language does not only depend on the string processed so far by the process considered but also by the string in the other process. Consider the example of two processes with languages  $L_P = \{\sigma_P.\sigma_S.\sigma'_S, \sigma_P.\sigma_S.\sigma''_S\}$  and  $L_S = \{\sigma_P.\sigma_S.\sigma'_S, \sigma_S.\sigma_P.\sigma''_S\}$  where  $\sigma_P \in \Sigma_P$  and  $\sigma_S, \sigma'_S, \sigma''_S \in \Sigma_S$ . Then, after the string  $s_P = \sigma_P.\sigma_S$  and for the pair  $(\sigma_P.\sigma_S, \sigma_P.\sigma_S)$  we will have  $\sigma'_S$  as continuation of  $s_P$  in  $L_P^c$ . However, if the the current pair is  $(\sigma_P.\sigma_S, \sigma_S.\sigma_P)$ , the continuation of  $s_P$  in  $L_P^c$  will be  $\sigma''_S$ . For a process connection with communication delays, the post-language of a closed-loop language must be expressed with the help of the current strings in both processes. We denote this by

$$L_P^c/(s_P, s_S) \quad \text{and} \quad L_S^c/(s_P, s_S)$$

for the closed-loop languages of  $P$  and  $S$  respectively. We are now ready to formally express non-blockingness in the next definition.

##### Definition 4 (Non-blocking Process with Delay)

*The composition of processes  $P = (\Sigma, L_P, M_P)$  and  $S = (\Sigma, L_S, M_S)$  subject to communication delays is non-blocking for  $S$  if every string in  $L_S^c$  can be completed to a string in  $M_S$ , *i.e.* if for all strings  $s_P$  and  $s_S$  of pairs  $(s_P, s_S)$  satisfying equations (4)*

$$s_S.(L_S^c/(s_P, s_S)) \cap M_S \neq \emptyset$$

In particular, we can state the following proposition.

**Proposition 6:** *A necessary condition for non-blockingness of process  $S$  in a delayed composition is that*

$$\overline{L_S^c \cap M_S} = L_S^c. \quad (13)$$

To show that equation (13) it is not sufficient, take  $\sigma_S, \sigma'_S \in \Sigma_S, \sigma_P, \sigma'_P \in \Sigma_P$  and  $M_S = M_P = \{\sigma_S.\sigma_P.\sigma'_S, \sigma_P.\sigma_S.\sigma'_P\}$ . It is easy to show that condition (13) is satisfied as  $L_S^c = L_P^c = \overline{M_S} = \overline{M_P}$ . The pair  $(\sigma_S.\sigma_P, \sigma_P.\sigma_S)$  satisfies relation (4) but  $L_S^c/(\sigma_S.\sigma_P, \sigma_P.\sigma_S) = \emptyset$ , in contrast with the definition of non-blockingness. Notice that the condition stated in the proposition, in the case without delays, was necessary and sufficient for non-blockingness for the process  $S$ . In order to see this, consider the delay operator to have no effect. Then equation (13) becomes  $L_S \cap L_P = \overline{M_S} \cap \overline{L_P}$ . For a plant  $P$  with prefix-closed marked language  $M_P = \overline{M_P} = L_P$ , the equation is also equivalent to  $L_P \parallel S = M_P \parallel S$ , which is the condition guaranteeing that a marking (in  $S$ , as  $P$  has prefix-closed marked language) can always be reached from any string in the closed-loop language.

**Theorem 7:** *Consider the well-posed composition of processes  $S = (\Sigma, \overline{M_S}, M_S)$  and  $P = (\Sigma, \overline{M_P}, M_P)$ , with  $M_S \subseteq M_P$  where  $\overline{M_P}$  is a self-well-posed and memoryless language. The process  $S$  is non-blocking in the closed-loop if and only if equation (13) is satisfied.*

We now present the marked supervisor synthesis problem with delays.

#### Supervisor Synthesis Problem with Delays:

*Given a plant  $P = (\Sigma, \overline{M_P}, M_P)$  and a specification language  $L'_{\text{spec}} \subseteq \Sigma_P^*$  for the closed-loop behavior, find a supervisor  $S = (\Sigma, \overline{M_S}, M_S)$  with  $M_S \subseteq M_P$  such that*

1.  $\emptyset \subset \text{proj}[\Sigma_P](L_P^c) \subseteq \overline{L'_{\text{spec}}}$ ,
2. *The connection with delays of  $P$  and  $S$  is well-posed,*
3. *The closed-loop is non-blocking for  $S$ ,*
4. *A marking in  $S$  will eventually be a marking in  $P$  and in the specification  $\text{proj}^{-1}[\Sigma, \Sigma_P](L'_{\text{spec}})$ .*

For the characterization of the solution we need to extend of the concept of memorylessness to marked languages.

#### Definition 5 (Memoryless Marked Language)

*A marked language  $L$  is memoryless with respect to the partition  $\Sigma = \Sigma_P \dot{\cup} \Sigma_S$  if  $\overline{L}$  is memoryless, and if for every  $s, s' \in \overline{L}$  with  $s \in \text{delay}[\Sigma_P](\{s'\})$*

$$s.t \in L \Leftrightarrow s'.t \in L \quad (14)$$

Suppose that the language  $M_P$  of the supervisor is memoryless, and moreover that  $s \in M_P$  implies  $s.\Sigma_P \cap \overline{M_P} = \emptyset$  (this condition states that if a marking is reached, no additional response can occur without a next command). Then we can state the condition for existence of a solution in the next theorem.

**Theorem 8:** *Consider a plant with memoryless language  $M_P$  satisfying*

$$s \in M_P \Rightarrow s.\Sigma_P \cap \overline{M_P} = \emptyset \quad (15)$$

*The supervisor synthesis problem with delays has a solution if and only if for the language*

$M_S = \sup \mathcal{C}(\sup \mathcal{SC}[L_P, \Sigma_S](M_P) \cap \text{proj}^{-1}[\Sigma, \Sigma_P](L'_{\text{spec}}))$   
 $\text{proj}[\Sigma_P](M_S)$  *is non-empty.  $S = (\Sigma, \overline{M_S}, M_S)$  is the least restrictive solution.*

Here, the class  $\sup \mathcal{SC}[L_P, \Sigma_S](M_P)$  is trivially extended to marked languages. It is therefore possible, given the assumptions of memorylessness and the additional assumption (15) on plant languages, to compute with a polynomial algorithm the solution of the supervisor synthesis problem with delays.

## 4. Conclusion

In this paper, an input/output perspective on supervisory control has been presented. The plant generates responses in reaction to commands and, symmetrically, the supervisor accepts the responses of the plant as inputs and produces commands for the plant. The notion of well-posedness of a connection has been presented. The effect of communication delays on the composition of a plant and a supervisor has been addressed. The languages of supervisors that guarantee well-posedness of the composition subject to delays have been characterized and conditions for non-blockingness extended to connections with communication delays have been presented. The restriction of plants to have memoryless languages allows to compute in polynomial time a supervisor solving a supervisory synthesis problem with communication delays. Future research consists in analyzing how strong is the assumption of memorylessness for languages of real systems, and possibly in looking for alternative classes of languages relaxing this assumption.

## References

- [1] P.J. Ramadge and W.M. Wonham. Supervisory control of a class of discrete event processes. *SIAM J. Control Optim.*, 25(1):206–230, January 1987.
- [2] S. Balemi and U.A. Brunner. Supervision of discrete event systems with communication delays. In *Proc. of 1992 American Control Conference*, pages 2794–2798, Chicago, IL, USA, June 1992.
- [3] W.M. Wonham and P.J. Ramadge. On the supremal controllable sublanguage of a given language. *SIAM J. Control Optim.*, 25(3):637–659, May 1987.
- [4] S. Balemi, G.J. Hoffmann, P. Gyugyi, H. Wong-Toi, and G.F. Franklin. Supervisory control of a rapid thermal multiprocessor. Technical Report # 91.17, Automatic Control Laboratory, Swiss Federal Institute of Technology (ETH), Zürich, Switzerland, November 1991. Also appears as Technical Report # ISL/GFF/91-1 at the Information Systems Laboratory of Stanford University, CA, USA.
- [5] S. Balemi. *Control of Discrete Event Systems: Theory and Application*. PhD thesis, Automatic Control Laboratory, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, May 1992.