Supervisory Control of Interacting Discrete Event Systems

Sherif Abdelwahed
sherif.abdelwahed@vanderbilt.edu
Institute for Software Integrated Systems
Vanderbilt University
Nashville, TN, USA

W. M. Wonham
wonham@control.toronto.edu
Department of Electrical Engineering
University of Toronto
Toronto, Ontario, Canada

Abstract

In this paper we present a theory for decentralized supervisory control of a general class of multiprocess discrete event systems referred to as interacting discrete event systems (IDES). An IDES is a discrete event system that consists of a number of components working concurrently and possibly restricted by a well-defined interaction specification. We extend the notion of decentralized supervisory control to this class of systems by adding a supervisor for the interaction specification. Conditions that guarantee the optimality of the decentralized supervision of this class of systems are established and a complexity analysis of the proposed procedure is presented.

1 Introduction

Discrete event systems (DES) are systems with finite state space where transitions are driven by discrete events. Examples of discrete event systems include communication networks, database systems, traffic networks, digital circuits and manufacturing systems. A control theory of a general class of discrete event systems was initiated by Ramadge and Wonham [5]. Control-theoretic concepts such as controllability and observability have been formalized in the DES setting.

Most practical real-life discrete event systems consist of a large number of components that operate concurrently. The composite system size grows exponentially with the number of components - a phenomenon known as the state space explosion problem. For such class of systems a decentralized control mechanism appears more suitable than a centralized one. In [4], a decentralized control scheme is developed for discrete event systems in which the specification is implemented using a set of local supervisors acting concurrently. Each local supervisor observes and controls a subset of the system events. Necessary and sufficient conditions for the optimality of the decentralized control scheme are presented in [3].

In [7], the decentralized control of concurrent discrete event systems is investigated. In this work, the system is given as a set of concurrent subsystems. In the proposed decentralized scheme a set of local supervisors is constructed for the system components. It is shown that when the given specification is "separable", i.e., decomposable with respect to the system structure, the decentralized scheme can achieve the optimal behaviour under the supremal centralized supervisor.

In this paper we extend the decentralized supervisory control scheme to a general class of multiprocess discrete event systems with well-defined interaction between the system components. This class will be referred to as interacting discrete event systems (IDES). Interacting discrete event systems are introduced in [1, 2]. An interacting discrete event system consists of a set of DES components working concurrently and an interaction specification that restricts the concurrent behaviour of these components. Many standard forms of interaction such as the parallel, serial, refinement, and interleaving compositions can be simulated using a certain class of "abstract" interaction specifications [1].

Using the IDES model, the decentralized control scheme can be applied to a more general class of systems where the interaction between the system components is considered a variable in the model, allowing different forms of interaction, rather than a fixed one (parallel composition). Also, it is shown that any DES can be converted to an IDES that generates the same language. Therefore, the proposed decentralized scheme for interacting discrete event systems is not limited to separable (decomposable) languages.

The paper is organized as follows. Section 2 introduces basic facts and notation used in this paper. In Section 3 we introduce the language based model for interacting discrete event systems. A decentralized supervisory control scheme for interacting discrete event systems is introduced in Section 4. Sufficient conditions for the optimality of the decentralized supervision are given. In section 5 we present a brief complexity analysis of the decentralized supervisory synthesis procedures.

Proofs of the propositions and theorems established in this paper can be found in [1].

2 Preliminaries and Notation

Let Σ be an alphabet representing the events in the process under consideration. A *string* or word is a sequence of events. We will write Σ^+ for the set of all nonempty finite strings with events in Σ, and $\Sigma^* = \Sigma^+ \cup \{\epsilon\}$, where $\epsilon \notin \Sigma$ denotes the empty string. A *language* over the alphabet Σ is any subset of Σ^*. The set of languages over Σ will be denoted $\mathcal{L}(\Sigma)$. A string $s' \in \Sigma^*$ is a *prefix* of $s \in \Sigma^*$, denoted $s' \leq s$, if there exists $u \in \Sigma^*$ such that $s'u = s$. The *prefix closure* of a language $H \subseteq \Sigma^*$, denoted $\overline{H}$, is the set of all strings in Σ^* that are prefixes of strings in H. The *complement* of a language $L \subseteq \Sigma^*$ is defined as $\Sigma^* - L$ and is denoted L^c.

An *automaton* is a 5-tuple structure $A = (Q, \Sigma, \delta, q_o, Q_m)$, where Q is a finite set of states, Σ is a finite nonempty set of events, $\delta : Q \times \Sigma \to Q$ is a (partial) *transition function*, $q_o \in Q$ is the *initial state*, and $Q_m \subseteq Q$ is a nonempty set of *marker states*. If $\delta(q, \sigma)$ is defined, then we say that σ is eligible at q in A. This can also be expressed by the map $Elig_A : Q \to Pwr(\Sigma)$ which assigns to each state in A the set of eligible events. The map δ is extended to strings in the usual way. For a language $L \in \mathcal{L}(\Sigma)$, we will write $A(L)$ to denote the minimal automaton that generates L.

We will extend the above notation to handle multiprocess systems. Let I be the index set of a collection of processes. An alphabet vector over I is a set $\{\Sigma_i | i \in I\}$ of alphabets. In the following we will use bold letters to distinguish vector quantities. Let $\boldsymbol{\Sigma} = \{\Sigma_i | i \in I\}$ be an alphabet vector. The union of all alphabets in $\boldsymbol{\Sigma}$ will be denoted $\alpha(\boldsymbol{\Sigma})$ or simply Σ if no confusion arises. We will write Σ_s or $\alpha_s(\boldsymbol{\Sigma})$ for the set of shared (synchronous) events in $\boldsymbol{\Sigma}$, namely $\Sigma_s = \bigcup_{i \neq j} (\Sigma_i \cap \Sigma_j)$. A multi-process environment will be referred to as a *process space*. A process space is uniquely defined by its alphabet vector, hence both terms designate the same thing.

A language vector over $\boldsymbol{\Sigma}$ is a set $\boldsymbol{L} = \{L_i \subseteq \Sigma_i^* | i \in I\}$. The set of all language vectors over $\boldsymbol{\Sigma}$ is denoted $\mathfrak{L}(\boldsymbol{\Sigma})$. The language L_i is called the ith component of $\boldsymbol{L}$. Similarly a string vector is a set $\boldsymbol{s} = \{s_i \in \Sigma_i^* | i \in I\}$. For a map $F : \mathcal{L}(\Sigma) \to \mathcal{L}(\Sigma)$, we will write $F(\boldsymbol{L})$ for the vector language $\{F(L_i) \mid i \in I\}$. The decomposition (vector projection) map $\boldsymbol{P_\Sigma} : \mathcal{L}(\Sigma) \to \mathfrak{L}(\boldsymbol{\Sigma})$ associates each language $L \in \mathcal{L}(\Sigma)$ with the language vector $\{P_i L \mid i \in I\}$ where $P_i : \Sigma^* \to \Sigma_i^*$ is the natural projection map that erases all events other than those of the ith component of $\boldsymbol{\Sigma}$. On the other hand, the composition (synchronous product) map $B_{\boldsymbol{\Sigma}} : \mathfrak{L}(\boldsymbol{\Sigma}) \to \mathcal{L}(\Sigma)$ associates each language vector with its synchronous product $\|\boldsymbol{L}$. To simplify notation we will write $\boldsymbol{P}B_{\boldsymbol{\Sigma}}$ to denote the composition $\boldsymbol{P_\Sigma} \circ B_{\boldsymbol{\Sigma}} : \mathfrak{L}(\boldsymbol{\Sigma}) \to \mathfrak{L}(\boldsymbol{\Sigma})$, and $B\boldsymbol{P_\Sigma}$ for the composition $B_{\boldsymbol{\Sigma}} \circ \boldsymbol{P_\Sigma} : \mathcal{L}(\Sigma) \to \mathcal{L}(\Sigma)$.

3 Interacting Discrete Event Systems

Let $\boldsymbol{\Sigma}$ be an alphabet vector with index set I. An *interacting discrete event system* over $\boldsymbol{\Sigma}$ is a pair $\mathcal{L} = (\boldsymbol{L}, K)$ where $\boldsymbol{L}$ is a language vector in $\mathfrak{L}(\boldsymbol{\Sigma})$ and K is a language in $\mathcal{L}(\Sigma)$. The language K will be referred to as the *interaction specification language* or simply the *interaction language* of the IDES $\mathcal{L}$. We will use calligraphic letters to denote IDES structures. Also we will write L_i to denote the ith component of $\mathcal{L}$. The language generated by $\mathcal{L}$ is given by

$$B_{\boldsymbol{\Sigma}}(\mathcal{L}) = \|\boldsymbol{L} \cap K$$

Therefore, the IDES structure consists of a set of components, represented by the language vector $\boldsymbol{L}$, running concurrently, and a language K that synchronizes with the composite behaviour of these components. Note that the $B_{\boldsymbol{\Sigma}}$ operation is overloaded as it is defined for both language vectors and IDES, consistently, given that vector languages are a subclass of IDES where $K = \Sigma^*$.

Based on the setting of the IDES model it is possible to decompose any single process (flat) DES to an IDES structure that generates the same behaviour. This can be done by compensating the information lost in the projection operation, that is, by adding necessary information to the composite behaviour $B\boldsymbol{P_\Sigma}(L)$ such that the overall behaviour of the structure is equal to L. It is easy to see that, for any language L such a compensator depends on L (is a function of L) and must contain L. The set of $\boldsymbol{\Sigma}$-compensators for L, denoted $\underline{C}_{\boldsymbol{\Sigma}}(L)$, is defined as follows

$$\underline{C}_{\boldsymbol{\Sigma}}(L) = \{K \in \mathcal{L}(\Sigma) \mid L = B\boldsymbol{P_\Sigma}(L) \cap K\}$$

The set $\underline{C}_{\boldsymbol{\Sigma}}(L)$ is not empty as it contains L. It is easy to verify that the set $\underline{C}_{\boldsymbol{\Sigma}}(L)$ is closed under union and intersection and hence has a supremal and infimal element. The infimal element of $\underline{C}_{\boldsymbol{\Sigma}}(L)$ is L. We will write $\check{C}_{\boldsymbol{\Sigma}}(L)$ to denote the infimal element of the set $\underline{C}_{\boldsymbol{\Sigma}}(L)$ and $\hat{C}_{\boldsymbol{\Sigma}}(L)$ to denote its supremal element.

Proposition 3.1

$$\hat{C}_{\boldsymbol{\Sigma}}(L) = L \cup B\boldsymbol{P_\Sigma}(L)^c$$

$\square$

It is easy to see that any language K such that $L \subseteq K \subseteq \hat{C}_{\boldsymbol{\Sigma}}(L)$ is a $\boldsymbol{\Sigma}$-compensator for L. In general a compensator K for L may be blocking by construction in the sense that the intersection of K with $B\boldsymbol{P_\Sigma}(L)$

may produce blocking states from which the system cannot reach any marker state. We write $\underline{C}^o_{\boldsymbol{\Sigma}}(L)$ to denote the set of *nonblocking compensators* for L. Formally, the set $\underline{C}^o_{\boldsymbol{\Sigma}}(L)$ is defined as follows:

$$\underline{C}^o_{\boldsymbol{\Sigma}}(L) = \{K \in C_{\boldsymbol{\Sigma}}(L) \mid \overline{B_{\boldsymbol{\Sigma}}(\boldsymbol{P}_{\boldsymbol{\Sigma}}(L), K)} = B_{\boldsymbol{\Sigma}}((\boldsymbol{P}_{\boldsymbol{\Sigma}}(\overline{L}), \overline{K}))\}$$

Also, this set is not empty for any language L as it contains L itself. It is easy to verify that this set is closed under union and hence contains a supremal element. We will denote this supremal element by $\hat{C}^o_{\boldsymbol{\Sigma}}(L)$ for a given language L. Therefore, it is always possible to generate an optimal non-blocking IDES model that generates the language L. This is based on the following result.

Proposition 3.2

$$\hat{C}^o_{\boldsymbol{\Sigma}}(L) = L \cup \left[L\Sigma \cap [\overline{B\boldsymbol{P}_{\boldsymbol{\Sigma}}(L)}]^c \right] \Sigma^*$$

$\square$

Based on this result, the supremal non-blocking compensator $\hat{C}^o_{\boldsymbol{\Sigma}}(L)$ can be obtained by adding to L any string that extends a string in L without being a prefix of $B\boldsymbol{P}_{\boldsymbol{\Sigma}}(L)$. This can be implemented by computing the synchronous product of $A(L)$ and $A(B\boldsymbol{P}_{\boldsymbol{\Sigma}}(L))$; at every state (q_1, q_2) in this product, where q_1 is a state in $A(L)$ and q_2 is a state in $A(B\boldsymbol{P}_{\boldsymbol{\Sigma}}(L))$, we add a set of transitions with events $\Sigma - (Elig_{A(L)}(q_1) \cup Elig_{A(B\boldsymbol{P}_{\boldsymbol{\Sigma}}(L))}(q_2))$ leading to a terminal state with self-loop Σ^*. This ensures that $\hat{C}^o_{\boldsymbol{\Sigma}}(L)$ extends L to the maximal extent without allowing any prefix of $B\boldsymbol{P}_{\boldsymbol{\Sigma}}(L)$ to exist in $\hat{C}^o_{\boldsymbol{\Sigma}}(L)$ outside the prefix of L.

4 Supervisory control of IDES

The supervisory control problem for IDES is stated formally as follows. Given a system $\mathcal{L} = (\boldsymbol{L}, K)$ over a process space $\boldsymbol{\Sigma}$ generating a language $L = B_{\boldsymbol{\Sigma}}(\mathcal{L})$, and a specification language S represented by an IDES structure, $\mathcal{S} = (\boldsymbol{S}, R)$ with $S = B_{\boldsymbol{\Sigma}}(\mathcal{S})$, construct an IDES supervisor $\mathcal{V} = (\boldsymbol{V}, T)$, such that the supervised system $\mathcal{V}/\mathcal{L}$ satisfies $B_{\boldsymbol{\Sigma}}(\mathcal{V}/\mathcal{L}) \subseteq S$. The form of supervision we propose here is a component-wise supervision where each component of the supervisor controls the corresponding component of the system, and the supervisor's interaction specification controls the system's interaction specification. Formally this translates into defining $B_{\boldsymbol{\Sigma}}(\mathcal{V}/\mathcal{L})$ as follows

$$B_{\boldsymbol{\Sigma}}(\mathcal{V}/\mathcal{L}) = \|_{i \in I}(V_i/L_i) \cap (T/K)$$

where V_i is a supervisor for L_i with respect to the specification S_i, and in general (X/Y) denotes the restriction of the language Y by the supervisor represented by

the language X. Typically such restriction is achieved by total synchronization (intersection). The IDES supervision setting is depicted in Figure 1.
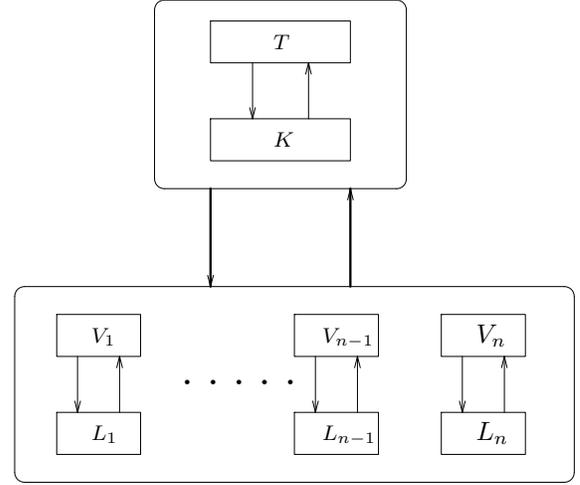

Figure 1: The IDES supervision scheme

For a language S representing a specification and a system with language L, we will write $\mathcal{C}_L(S)$ to denote the set of all sublanguages of S that are controllable with respect to L^1. This set contains a supremal element denoted $\sup \mathcal{C}_L(S)$. This element, if not empty, corresponds to the optimal supervisor for L with respect to the specification S. The blocking issue will not be addressed in this paper. Therefore, all languages are assumed prefix closed.

Proposition 4.1 Let $S_1, S_2, L \in \mathcal{L}(\Sigma)$ be prefix closed languages. Then

$$\sup \mathcal{C}_L(S_1 \cap S_2) = \sup \mathcal{C}_L(S_1) \cap \sup \mathcal{C}_L(S_1)$$

$\square$

The above proposition is a special case of a more general result in [8]. Optimal supervisors are implemented by the language $\sup \mathcal{C}_L(S \cap L)$. Based on the above Proposition we can write $\sup \mathcal{C}_L(S) \cap L = \sup \mathcal{C}_L(S \cap L)$. To simplify notation, this language will be denoted as $\sup \widetilde{\mathcal{C}}_L(S)$. Given this notation, the optimality of IDES supervision can be expressed by the following condition:

$$\begin{aligned} \sup \widetilde{\mathcal{C}}_L(S) &= \| \sup \widetilde{\mathcal{C}}_{L_i}(S_i) \cap \sup \widetilde{\mathcal{C}}_K(R) \\ &= \bigcap_{i \in I} P_i^{-1} \sup \widetilde{\mathcal{C}}_{L_i}(S_i) \cap \sup \widetilde{\mathcal{C}}_K(R) \end{aligned}$$

---

[1] For a detailed introduction to supervisory control theory we refer the reader to [8]. We adopt the notation used in this reference.

Note that the optimality of IDES supervision implies its validity. Conditions for the optimal IDES supervision will be explored hereafter.

Proposition 4.2

$$S = \overline{S} \text{ and } L = \overline{L} \implies \sup\widetilde{\mathcal{C}}_L(S) = \overline{\sup\widetilde{\mathcal{C}}_L(S)}$$

$\square$

Based on the above result, the optimal supervisor is guaranteed to be prefix closed when the system and the specification are prefix closed . The next result shows that the $\sup\widetilde{\mathcal{C}}$ operation is invariant with respect to inverse projection.

Proposition 4.3

$$P_i^{-1} \sup\widetilde{\mathcal{C}}_{L_i}(S_i) = \sup\widetilde{\mathcal{C}}_{P_i^{-1}L_i}(P_i^{-1}S_i)$$

$\square$

The next step is to find conditions to ensure that the synthesis of local supervisors for the system components can be done optimally using only information about these components without the need of information from other components of the system.

Proposition 4.4 Let $\mathcal{L} = (\boldsymbol{L}, K)$ be an IDES such that K is controllable with respect to $B_{\boldsymbol{\Sigma}}(\boldsymbol{L})$ and $\Sigma_s \subseteq \Sigma_c$. Let $L = B_{\boldsymbol{\Sigma}}(\mathcal{L})$. Then for all $i \in I$,

$$\sup\mathcal{C}_L(P_i^{-1}S_i \cap P_i^{-1}L_i) \cap L = \\ \sup\mathcal{C}_{P_i^{-1}L_i}(P_i^{-1}S_i \cap P_i^{-1}L_i) \cap L$$

$\square$

The conditions in the above proposition are sufficient to guarantee the optimality of supervision at the components level. The controllability conditions for the interaction part of the model are examined next.

Given the IDES specification $\mathcal{S} = (\boldsymbol{S}, R)$, to achieve optimal IDES supervision a condition is needed such that information about the interaction language K is enough to calculate the language $\sup\mathcal{C}_L(R) \cap K$. This language represents the optimal supervisor for the system's interaction language with respect to the interaction language of the specification.

Proposition 4.5 Assume $R \cap K$ is controllable with respect to K. Then

$$\sup\mathcal{C}_L(R \cap K) = \sup\mathcal{C}_K(R \cap K) = R \cap K$$

$\square$

In the above Proposition, the condition that $R \cap K$ is controllable with respect to K is sufficient but not necessary in general. For instance, when $K = L$, the implication of the proposition holds trivially independent of R. Based on the above Proposition, optimal supervision for the system's interaction language K can be achieved if a compensator R for the specification is found such that $K \cap R$ is controllable with respect to K. In general, the set of compensators for the specification S is any language R such that $S \subseteq R \subseteq \hat{C}_{\boldsymbol{\Sigma}}(S)$. Based on that we can state the following result.

Proposition 4.6 There exists an IDES model $(\boldsymbol{S}, R)$ for S such that R is prefix closed and $K \cap R$ is controllable with respect to K if

$$S \subseteq \sup\mathcal{C}_K(\hat{C}_{\boldsymbol{\Sigma}}^o(S))$$

$\square$

Note that if S is prefix closed, then the language $\hat{C}_{\boldsymbol{\Sigma}}^o(S)$ is also prefix closed, and therefore $\sup\mathcal{C}_K(\hat{C}_{\boldsymbol{\Sigma}}^o(S))$ is prefix closed.

It is worthwhile to consider limit cases of K for the above Proposition. When $K = L$, the supervisor of the system interactions is always optimal regardless of the specification. However, this is also the most computationally complex case. The other limit is more interesting, namely when $K = \Sigma^*$ indicating total parallelism of the system components. Clearly here $K \cap R = R$ for any compensator R for the specification. It is straightforward to check that in this case, the above condition holds if and only if $\overline{R}\Sigma_u \subseteq \overline{R}$, namely all uncontrollable events are enabled at any state in the automaton of the language R.

The results established so far can be used to obtain the main result of this paper regarding the optimality of the IDES supervision.

Theorem 4.1 Let $\mathcal{L} = (\boldsymbol{L}, K)$ be an IDES over a process space $\boldsymbol{\Sigma}$ with $L = L(\mathcal{L})$, and let $S \in \mathcal{L}(\Sigma)$. Assume that,

1. $\Sigma_s \subseteq \Sigma_c$
2. K is controllable w.r.t $B_{\boldsymbol{\Sigma}}(\boldsymbol{L})$
3. $S \subseteq \sup\mathcal{C}_K(\hat{C}_{\boldsymbol{\Sigma}}^o(S))$.

Then

$$\sup\widetilde{\mathcal{C}}_L(S) = \| \sup\widetilde{\mathcal{C}}_{L_i}(S_i) \cap \sup\widetilde{\mathcal{C}}_K(R)$$

$\square$

Note that based on the above Theorem, the efficiency of the decentralized supervisory synthesis depends,

among other factors, on the interaction language of the system, as well as that of the specification. Roughly speaking, the chance of finding an optimal IDES supervisor increases by restricting the interaction language of the system and relaxing that of the specification.

Example 4.1 The system in this example consists of two processes representing two machines working concurrently. The two machines can coordinate their operations using a shared event x which can be viewed as a synchronization signal. The IDES model of the system is shown below. In the diagrams the following convention is used: components are shown in rounded boxes, the interaction specification is shown in a double box, and controllable events are marked with a dash.
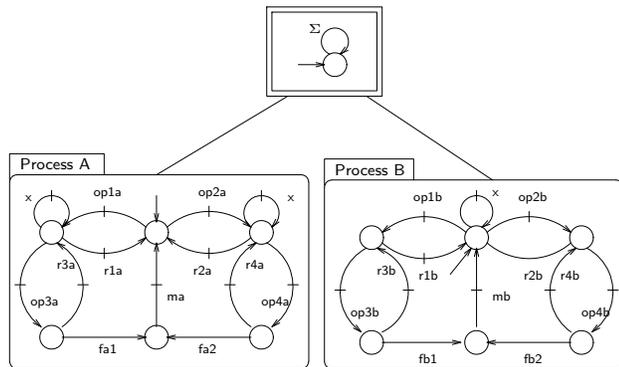

The specification of the system is shown in the figure below. The specification S is converted to the displayed IDES structure $\mathcal{S} = (\boldsymbol{P}_{\boldsymbol{\Sigma}}(S), \hat{C}^o_{\boldsymbol{\Sigma}}(S))$. To simplify the diagram, the selfloops of all remaining events at each state are omitted.
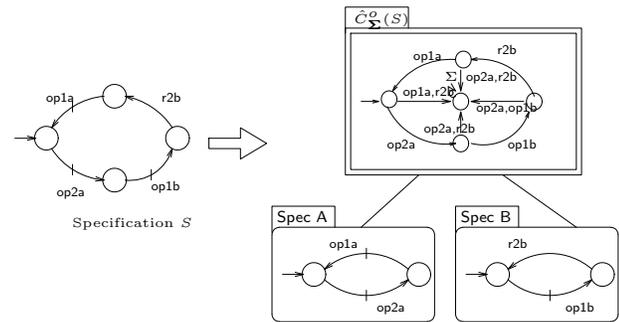
It is easy to check that the specification S is not controllable with respect to the interaction language of the system $K = \Sigma^*$. However, $\hat{C}^o_{\boldsymbol{\Sigma}}(S)$ is controllable with respect to K and therefore $\sup \mathcal{C}_K(\hat{C}^o_{\boldsymbol{\Sigma}}(S)) = \hat{C}^o_{\boldsymbol{\Sigma}}(S)$. The IDES supervisor for the specification is shown in the following figure.
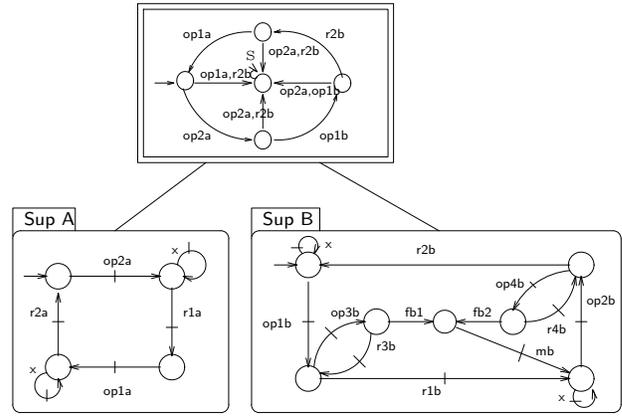
The following figure shows a new specification for the system. The new specification affects only the local specification for process B and the compensator $\hat{C}^o_{\boldsymbol{\Sigma}}(S)$. It can be checked that the new compensator is also controllable with respect to K. Therefore, the new IDES supervisor can be obtained from the one above by updating only the supervisor for process B with respect to its new specification.
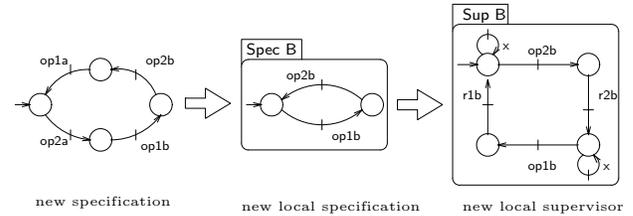
In general, under the decentralized supervision scheme, we need only recompute the solution with respect to the changed components of the system, rather than the whole system as would be needed under the direct centralized approach.

$\square$

5 Complexity Analysis

To compare the computational complexity of the decentralized control with that of the centralized form, consider the case of an IDES $\mathcal{L} = (\boldsymbol{L}, K)$ where $\boldsymbol{L}$ contains N components each with state size n. Let S be a specification of size m. In general, the single process model of $\mathcal{L}$ will have a size of $O(n^N)^2$. As required in the decentralized scheme, the specification S is assumed decomposed into the IDES $\mathcal{S} = (\boldsymbol{S}, R)$.

In the worst case, the size of each component of the specification $S_i = P_i(S)$ is $O(2^m)$, and that of R is

---

[2]For detailed complexity analysis of the relevant operations we refer the reader to [6].

$O(2^{mN})$. However, in typical situations the size of each S_i is closer to $O(m)$. In this case the size of R can be in the range from $O(m)$ to $O(m^N)$ although more likely to be closer to the lower end based on the way the compensator is constructed. Solving the supervisory control problem using the flat model of $\mathcal{L}$ will require $O(mn^N)$ time and $O(n^N)$ space. Under the assumption that the elements of decomposed structure of the specification is within the same dimension of the specification, the decentralized supervision will require $O(mnN)$ time and $O(\max(n,m)N)$ space.

The main challenge in checking the optimality of the decentralized supervisor is to check condition 2 in Theorem 4.1, namely, that the interaction specification of the system is controllable with respect to the parallel composition of the system components. However, this check can be eliminated if the language K is always controllable with respect to any language, that is when $\overline{K}\Sigma_u \subseteq \overline{K}$. The most widely used parallel composition scheme ($K = \Sigma^*$) satisfies this condition.

6 Conclusion

In this paper we presented a decentralized supervision structure for a general class of multiprocess DES with possible interaction constraints. It is shown that under certain conditions related to the structure of process space, the architecture of the system, and the given specification, an optimal supervisor can be computed and implemented modularly.

7 Acknowledgement

References

[1]   S. Abdelwahed. *Interacting Discrete Event Systems: Modelling, Verification, and Supervisory Control.* PhD thesis, University of Toronto, 2002.

[2]   S. Abdelwahed and W.M. Wonham. Interacting discrete event systems. In *Proc. of 37th Allerton Conf. on Communication, Control and Computing*, Champaign, IL, USA, September 1999.

[3]   R. Cieslak, C. Desclaux, A. Fawaz, and P. Varaiya. Supervisory control of discrete-event processes with partial observations. *IEEE Trans. Autom. Control*, 33(3):249–260, March 1988.

[4]   F. Lin and W.M. Wonham. Decentralized control and coordination of discrete event systems with partial observation. *IEEE Trans. Autom. Control*, 35(12):1330–1337, 1990.

[5]   P.J. Ramadge and W.M. Wonham. Supervisory control of a class of discrete-event systems. *SIAM Journal on Control and Optimization*, 25:206–230, 1987.

[6]   K. Rudie. Software for the control of discrete-event systems: A complexity study. Master's thesis, Dept. of Elec. Eng., University of Toronto, 1988.

[7]   Y. Willner and M. Heymann. Supervisory control of concurrent discrete event systems. *Int. Journal of Control*, 54(5):1143–1169, 1991.

[8]   W.M. Wonham. *Notes on Control of Discrete-Event Systems.* ECE Department, University of Toronto, revised 2002.07.01. `www.control.utoronto.ca/people/profs/wonham`.