

## A Novel Performance Evaluation Method for DES

OZGUR KAYMAKCI AND SALMAN KURTULAN

*Department of Control Engineering  
Faculty of Electrics and Electronics Engineering  
Istanbul Technical University  
34469 Maslak, Turkey  
E-mail: kaymakcio@itu.edu.tr*

When a system includes too many possible blockings, selecting the minimal restrictive nonblocking solution as supervisor sometimes constitutes a conservative solution. Then relaxing the nonblocking condition becomes an inevitable fact to improve the overall performance. On the other hand, selecting the complete satisfying solution as supervisor may cause serious system failures due to some of these possible blockings. Then a supervisor linking these two solutions is needed. Therefore in this paper the balance between blocking and success is investigated employing an optimization approach. Firstly, we introduce a new performance measure which depends on numeric values obtained from strings that correspond to blocking and success. The proposed formulation captures the fundamental trade-off motivated by the classical optimization approach. Besides a new algorithm that explores the best result according to this performance measure is introduced.

**Keywords:** discrete event systems, blocking, regular languages, performance evaluation, supervisory control theory, blocking supervisor

### 1. INTRODUCTION

Recently in computer and communication technologies, new systems developed so as to mainly work in asynchronous mode and to show different dynamic behaviors depending the execution order of particular events. Called Discrete Event Systems in the literature, these systems also include such practical systems as computer systems, communication systems, automated manufacturing systems, traffic systems and transportation systems [1].

We can not model discrete event systems by employing ordinary differential equations, this being their most important difference from time-varying dynamic systems. Rather discrete event systems evolve in time in the form of events occurring at possibly irregular time intervals. Therefore, we often model them as regular languages represented by deterministic finite state automata. In addition, in order to examine the behavior of the system, we modify the behavior by a control action. Ramadge and Wonham's supervisory control theory is a powerful tool to build variety of modifications in the model through a supervisor. However, the supervisor can not usually mark all the desired behaviors of the system, which are built according to nonblocking and controllability constraints. Thus blocking becomes a crucial task in discrete event systems. Industrial examples give support to this observation too. We observe that supervisory control problems with blocking are generally more widespread than those without blocking, and

blocking supervisors are preferred to nonblocking ones in several applications. In such an application, database concurrency control, the protocols coupled with deadlock detection schemes are most popular since deadlock prevention and avoidance are impractical.

In modeling a distributed system with discrete event system formalism, we have to include a large number of uncontrollable events in order to take into account the failure of a large number of remote operations. Similarly, if we include system failures in the system model in automated manufacturing systems, the general performance of nonblocking solutions are usually insufficient. In similar cases nonblocking solutions may end up being too conservative. Therefore, one might risk some possible blocking occurrences in order not to constrain the system's behavior. Taking this risk may pay off if there is a serious increase in the performance of the system. Hence one has to investigate the performance of the system by introducing the benefits and losses of possible blockings to the system.

Several researchers have worked on blocking in discrete event systems. Chen and Lafortune, for example, dealt with blocking in discrete event systems and introduced two operators on a set of languages. In this way they optimized the performance measure according to set inclusion [2]. Also Levis and Gurel analyzed deadlock on a Petri net model of a large class of re-entrant flow lines systems and gave necessary and sufficient conditions for deadlock [3].

On the other hand, some researchers have proposed methods for the optimal control of DES with different assumptions. Passino and Antsaklis proposed a cost function on the set of transitions and presented a control objective that restricts the plant behavior. This enabled them to represent the optimal control of a discrete event system as a trajectory following problem with optimal cost [4]. Kumar and Garg proposed a cost function with payoff and control costs, and solved optimal control problem by transforming it into a combinatorial one. Then, they solved it by network flow algorithm. Kumar and Garg assumed that a certain state may be visited only once, so the corresponding transitions are controlled only once [5]. Sengupta and Lafortune defined control and event costs to find the optimal nonblocking supervisor, and solved the problem by dynamic programming [6]. Surana and Ray constructed a signed measure, defined on state transitions, over discrete event systems [7]. Fu and Ray also used the same signed measure to formulate an unconstrained optimal control policy. The policy is obtained by selectively disabling controllable events to maximize the measure [8]. All these studies contributed to our knowledge of how to enhance the performance of discrete event systems. But, none was interested in the blocking case.

This work aims to fill this important gap in the literature. Recently we introduced a new metric space in [9] and discussed the results obtained considering this space. Then we solved the problem when only deadlock occurs in [10]. Here these results are explained and the problem is generalized to not only optimal blocking but also maximally permissive supervisor. The paper is organized in 5 sections. Section 2 briefly gives the preliminaries. Section 3 describes the motivation and gives the problem formulation. Then at section 4 optimization algorithm and an example is presented. Finally the paper is summarized and concluded in section 5. In order to ease the reading of the paper, properties and some of the proofs are given in Appendixes A and B, respectively.

## 2. PRELIMINARIES

Following the definitions given in [1] let  $G = (X, \Sigma, f, x_0, X_m)$  be the deterministic finite state machine (DFSM) that represents the discrete event dynamics of a physical plant, where  $X = \{x_0, x_1, x_2, \dots, x_n\}$  is the set of states with  $x_0$  being the initial state,  $\Sigma$  is the finite set of events,  $f: X \times \Sigma \rightarrow X$  is the state transition function and  $X_m \subseteq X$  is the set of marked states representing a completion of a given task or operation. Then the behavior of the system  $G$  can be described by a prefix-closed language  $L(G)$ , which is defined as  $L(G) = \{s \in \Sigma^* \mid f^*(x_0, s) \in X\}$ , where  $\Sigma^*$  denotes the set of all finite concatenations of events that belong to  $\Sigma$ , including the zero length string  $\varepsilon$ . The language  $L_m(G)$  corresponds to the marked behavior of the deterministic finite state machine  $G$ . For detailed information refer to [1].

For a string  $s \in \Sigma^*$ ,  $\bar{s}$  denotes the prefixes of  $s$  and is defined as  $\bar{s} = \{s_p \in \Sigma^* : \exists t \in \Sigma^* (s_p t = s)\}$ . Extending this definition to languages, we get prefix closure of a language  $L$  denoted as  $\bar{L}$ . Here the length of a string is symbolized with  $\| \cdot \|$ . The projection function  $p_j(s)$  represents the prefix of string  $s$  of length  $j$  [6]. Also, the function  $q_j(s)$  gives the  $j$ th event of string  $s$ . For example let  $s = e_1 e_3 e_4 e_1$  be a string over  $\Sigma$ , then  $p_2(s) = e_1 e_3$  and  $q_2(s) = e_3$ .

If  $L(G) \neq \overline{L_m(G)}$  then the DFSM  $G$  is said to be blocking. Two types of blocking can occur; these are deadlock and livelock, respectively. At deadlock,  $G$  can reach a state  $x_i \notin X_m$  where  $f(x, e) = \emptyset$  for  $\forall e \in \Sigma$ . At livelock, the DFSM can reach a set of unmarked states that form a strongly connected group of states, but with no transition out of this set [11]. Also we partition the event set into two disjoint sets  $\Sigma = \Sigma_c \cup \Sigma_{uc}$ , where  $\Sigma_c$  and  $\Sigma_{uc}$  represent the set of controllable events and the set of uncontrollable events respectively. The controllable events set include the events that can be prevented from happening, or disabled, by supervisor. Sending a message in a database system or enabling a relay at a manufacturing system is expressed with these kind of events. On the other hand the uncontrollable events set include the events that can not be prevented from happening. A sensor at a manufacturing system or an error signal is expressed with uncontrollable events. Here the structure of the system determines if an event is controllable or not. For example, as we can not force a sensor to not detect the product at a manufacturing system, we have to symbolize the sensors with uncontrollable events in the model for consistency. Also we assume that all the events are observable in this work.

The control of DES for the first time was explicitly introduced in the work of Wonham and Ramadge [12]. The existence of a supervisor is guaranteed if the desired language  $K$  satisfies the Controllability Condition defined as  $\bar{K} \Sigma_{uc} \cap L(G) \subseteq \bar{K}$ . But the given language can not be always controllable with respect to  $\Sigma_{uc}$ . Then the idea of obtaining the maximum admissible part of a given language occurs. Here the ‘‘maximal’’ means in terms of set inclusion. This maximum part of  $K$  is Supremal Controllable Sublanguage and denoted with  $K^{\uparrow C}$ . Efficient algorithms are given in the literature to compute supremal controllable sublanguage [13, 14]. With a similar approach, violating the controllability condition, one finds the smallest prefix-closed and controllable language containing  $K$ . This language is known as Infimal Prefix-Closed Controllable Superlanguage of  $K$  and denoted with  $K^{\downarrow C}$ . Likewise algorithms also exist for its computation [11].

### 3. MOTIVATION AND PROBLEM FORMULATION

When we examine a discrete event system within the corresponding admissible language, too many blockings may be realized. We then usually select minimally restrictive nonblocking solution (MRNBS) as the supervisor, even though it is inadequate due to its restrictive behaviour [1, 14]. Preventing all uncontrollable events that lead to blocking, MRNBS can provide a conservative result. As a result, we can generate only a small part of admissible marked language. As such this strategy may constrain the behaviour of the system considerably. On the other hand, the completely satisfying solution (CSS), which is studied in [11], generates all the admissible marked strings, but its price adds too many blockings to the supervised language. Therefore, neither of these two results can be acceptable in many discrete event system problems. Here the success of the supervisor is related to the generated admissible marked language. Then it will be of interest to relax the nonblocking requirement and consider the synthesis of blocking supervisors. What motivates us to consider blocking solutions is that by allowing a certain amount of blocking, we can increase the part of admissible marked language that can be achieved under control. But, then, “how many blockings do arise?” This work aims to find a formal answer to this question, and solves the problem for acyclic case.

In the literature blocking supervisors were first studied in [2]. If possible, the proposed solution may improve the performance either by reducing the supervised language’s blocking without affecting its achievement or by increasing the supervised language’s achievement without affecting its blocking. But, the literature did not concern with improving the performance by reducing the blockings and achievements at the same time.

Furthermore, the generated strings on the system have different meanings in practice. For example, some strings, which symbolize critic tasks, have high importance compared to other strings. Thus the differences between the strings have to be taken into consideration before selecting the best supervisor.

Now we will present the above discussion in a formal setting. For a given automata  $G$  and the admissible supervisor  $S$ , the resulting closed loop system is symbolized with  $S/G$ . The admissible language and admissible marked language are  $L_a$  and  $L_{am}$ , respectively. Then the specifications on the controlled language are stated as follows:  $L_m(S/G) := L(S/G) \cap L_m(G)$ ,  $L_m(S/G) \subseteq L_{am}$ ,  $L(S/G) \subseteq L_a = \bar{L}_a \subseteq L(G)$ ,  $L_{am}^{\downarrow C} \subseteq L_a$ ,  $L_{am} = L_a \cap L_m(G) = L_a \cap L_{am}$ .

The MRNBS and CSS are achieved by  $L(S/G) = \overline{L_{am}^{\uparrow C}}$  and  $L(S/G) = L_{am}^{\downarrow C}$  respectively. Then any meaningful solution  $S$  should satisfy the following:  $\overline{L_{am}^{\uparrow C}} \subseteq L(S/G) \subseteq L_{am}^{\downarrow C}$ .

Therefore we define the class of all admissible solutions is  $L_{cand} := \{\Gamma: (\overline{L_{am}^{\uparrow C}} \subseteq \Gamma \subseteq L_{am}^{\downarrow C}) \wedge (\Gamma = \bar{\Gamma}) \wedge (\bar{\Gamma} \sum_{uc} \cap L(G) \subseteq \bar{\Gamma})\}$ . For an admissible solution, the strings that drive the supervised system to blocking can be represented with blocking measure set which is defined by  $BM(L(S/G)) := L(S/G) \setminus \overline{L_m(S/G)}$ . Likely, the admissible marked strings that are not allowed by the supervised system are denoted with blocking measure set,  $SM^C(L(S/G)) := L_{am} \setminus L_m(S/G)$ . For detailed information refer to [2]. At this point there is no difference between the strings generated by the system. So at first, we express the cost of a generated string and cost of a language, respectively and then we formalize a new per-

formance measure, which gives an opportunity to discriminate the languages. Here only the definitions will be given. For detailed information refer to [10].

**Definition 1** The cost of generated string is denoted as  $c_i$  where  $s \in \Sigma^*$  then  $c_i(s): \Sigma^* \rightarrow \mathfrak{R}^+$ .  $\square$

Here the costs of strings have to be given by a system expert's vision. Note that, this is beyond the scope of this paper.

**Definition 2** The cost of a language is defined as  $\beta(L): \Sigma^* \rightarrow \mathfrak{R}^+$  where  $L \in \Sigma^*$  such that  $L = \{s_1, s_2, s_3, \dots, s_n\}$

$$\text{Then } \beta(L) := \begin{cases} \sum_{i=1}^n c_s(s_i) & L \neq \emptyset \\ 0 & \text{otherwise} \end{cases}. \quad \square$$

**Definition 3** The distance function  $d: \Sigma^* \times \Sigma^* \rightarrow \mathfrak{R}^+ + \{0\}$  is defined in terms of the language cost as:

$$d(L_1, L_2) := \beta(\{L_1 \setminus L_2\} \cup \{L_2 \setminus L_1\}). \quad \square$$

**Proposition 1** The set  $\Sigma^*$  and the distance function  $d$  form a metric space  $(2^{\Sigma^*}, d)$ .

The proof of this proposition is also at [10]. Also for detailed information on metric spaces refer to [15].

**Definition 4** The non-satisfying measure and blocking measure for supervised language  $L(S/G) \in L_{cand}$  are defined, respectively

$$\begin{aligned} \widehat{SM}^C(L(S/G)) &:= d(L_{am}, L_m(S/G)), \\ \widehat{BM}(L(S/G)) &:= d(L(S/G), \overline{L_m(S/G)}). \end{aligned} \quad \square$$

As we mentioned the cost of a string is related with its meaning in the overall process. If it expresses a very dangerous blocking or an essential work, its cost has to be greater than other ones. The blocking measure defines the total cost of how much the supervised system can be affected by the blockings and gives a unique numeric value according to the possible blockings in the system. Similarly, with the non-satisfying measure the failure of supervisor S is expressed and a positive value is introduced.

**Definition 5** For  $L(S/G) \in L_{cand}$ , the performance measure of the language is defined as:  $\hat{J}(L(S/G)) := \widehat{SM}^C(L(S/G)) + \widehat{BM}(L(S/G))$ .  $\square$

The performance of blocking supervisors holds on two concepts: blocking and fail-

ure. Then the defined performance measure captures the fundamental concept, as it is given over sum of blocking measure and non-satisfying measure. Here it is clear that a trade-off between blocking and failure usually occurs usually while selecting the blocking supervisor. For example, when we examine the MRNBS and CSS, we can easily see the trade-off between blocking and failure. As the MRNBS does not include a blocking string due to its structure, the blocking measure of it is equal to zero. But its non-satisfying measure score is the highest in the admissible solutions set. On the other hand, as the CSS generates all the admissible marked strings, its non-satisfying measure equals to zero but it has a high blocking measure. Also the loss factor of the language can be shown with a function like that  $F(L) := \beta(L \setminus \overline{L_m}) - \beta(L_m)$ . This has no determining effect on selecting the language, only gives an opinion about it. But this function will be used in the proposed algorithm to determine the priority of the blocking strings.

**Definition 6**  $L_1, \dots, L_n \in L_{cand}$  and  $\hat{J}(L_1) = \dots = \hat{J}(L_n)$  then if  $\widehat{SM}^C(L_k) < \widehat{SM}^C(L_i)$  always holds for  $i = 1, \dots, n$  then  $L_k$  is the maximally permissive language.  $\square$

It is clear that the supervisor that generates the maximally permissive language is the maximally permissive supervisor. Here the maximally permissive supervisor generates the admissible marked language as much as possible.

**Definition 7** Let the uncontrolled behavior of the system be  $L(G)$  and the admissible solutions set  $L_{cand}$  be non-empty set. The performance measure of a supervised system is  $\hat{J}[L(S/G)]$ , where  $L(S/G) \in L_{cand}$ . Then the optimal blocking supervisor problem is defined as

$$\arg\left\{ \min_{L(S/G) \in L_{cand}} \hat{J}(L(S/G)) \right\}. \quad \square$$

As the performance measure of depends on blockings and failure, the best solution will be the language that generates all the admissible marked strings with having no blockings. But this solution may not be feasible according to controllability and admissibility conditions. Then we have to select a blocking supervisor from the admissible solutions set. Also this solution has the minimum performance measure. At the same time we expect from the supervisor to generate the admissible marked strings as much as possible. Then the optimality of the solution is not enough but also it has to be the maximally permissive solution. Also properties derived from the definitions are given in Appendix A.

## 4. OPTIMIZATION ALGORITHM AND AN EXAMPLE

### 4.1 Theoretical Background of Optimization Algorithm

In this section, the theoretical foundations of an optimal blocking supervisor will be presented and the algorithm to derive an optimal blocking supervisor will be introduced. For  $L(S/G) \in L_{cand}$ , the blocking measure set and non-satisfying measure set will be a set as  $BM(S/G) = \{s_1, \dots, s_m\}$ ,  $SM^C(L(S/G)) = \{t_1, \dots, t_n\}$ , respectively. As  $L(S/G)$  is a regular language, the blocking measure set and non-satisfying measure set have to be two finite

sets. Here the problem is solved for a proper  $L_{cand}$  set such that  $\forall L_1, L_2 \in L_{cand}$  holds the  $(L_1 \cap \overline{L_{2m}}) \setminus \overline{L_{1m}} = \emptyset$  condition.

**Definition 8** For  $L \in L_{cand}$  and  $s_i \in BM(L(S/G))$ , there exists a transformation  $T: L_{cand} \rightarrow L_{cand}$  such that

$$T(L, \alpha_i) := \begin{cases} [(L \setminus s_i)^{\uparrow C} \cap L_{am}]^{\downarrow C} & \text{if } \hat{J}([(L \setminus s_i)^{\uparrow C} \cap L_{am}]^{\downarrow C}) < \hat{J}(L) \\ L & \text{otherwise} \end{cases} \quad \square$$

To obtain the transformation result,  $((L \setminus s_i)^{\uparrow C} \cap L_{am})^{\downarrow C}$  is obtained at first and its performance is calculated. Then, if its performance is smaller than  $L$ , it is accepted. If not,  $L$  remains. Here the transformation is bounded by a strict performance improvement so the content of Definition 8 is a new concept as it is related to performance change. Also  $L_{cand}$  is closed under  $T$  due to Property 3 given in Appendix A then the transformation  $T$  is from  $L_{cand}$  set to  $L_{cand}$  set. If the criteria over performance is fulfilled, the removed strings from the language  $L$  is  $L \setminus ((L \setminus s_i)^{\uparrow C} \cap L_{am})^{\downarrow C}$ .  $r(L, s_i)$  will be used instead of  $L \setminus ((L \setminus s_i)^{\uparrow C} \cap L_{am})^{\downarrow C}$  from now on.

**Remark 1:** When blocking in discrete event systems is examined, every blocking has a different influence on the system and the effect of each blocking on the system is usually dissimilar in practice. Also some of these blockings can be acceptable and some of them must be prevented. The risky blockings have to be removed from the supervised language with an appropriate method. More strings can be removed from the supervised language due to admissibility and controllability conditions during this removing operation. These strings can also influence the system performance. On the other hand, the removed strings from the language due to different blockings either have no common elements or one group enclose the other one. These results can be shown with two expressions  $r(L, s_i) \cap r(L, s_j) = \emptyset$  and  $r(L, s_i) \subseteq r(L, s_j)$ , respectively. Here  $L$  is a member of  $L_{cand}$  set and  $s_i, s_j \in BM(L)$ .

**Proposition 2** Let  $L_1, L_2 \in L_{cand}$  be two languages such that  $L_1 \subset L_2$  and  $(L_1 \cap \overline{L_{2m}}) \setminus \overline{L_{1m}} = \emptyset$  then  $\hat{J}(L_1) = \hat{J}(L_2) - F(L_2 \setminus L_1)$  always hold.

The proof of this proposition is given in Appendix B. This proposition expresses the performance relation of two languages over loss factor. Here  $\hat{J}(L_1)$  and  $\hat{J}(L_2)$  implies the performance of  $L_1$  and  $L_2$ , respectively. At the same time  $F(L_2 \setminus L_1)$  gives the loss factor of  $L_2 \setminus L_1$ .

**Remark 2:** Let  $L \in L_{cand}$  be a proper language and  $s_i \in BM(L)$ . Then as  $((L \setminus s_i)^{\uparrow C} \cap L_{am})^{\downarrow C} \subset L$  so the Proposition 2 can be applied to these two languages. If  $F(r(L, s_i)) \leq 0$  then  $T(L, s_i) = L$  because  $\hat{J}(((L \setminus s_i)^{\uparrow C} \cap L_{am})^{\downarrow C}) \geq \hat{J}(L)$  due to Proposition 2. If  $F(r(L, s_i)) > 0$ ,  $\hat{J}(((L \setminus s_i)^{\uparrow C} \cap L_{am})^{\downarrow C}) < \hat{J}(L)$  then is accepted as transformation result. Here the  $(L_1 \cap \overline{L_{2m}}) \setminus \overline{L_{1m}} = \emptyset$  condition at the Proposition 2 implies that no new blocking strings occur after the  $((L \setminus s_i)^{\uparrow C} \cap L_{am})^{\downarrow C}$  operation. In other words  $BM(L_1) \subseteq BM(L_2)$  always holds.

**Proposition 3** If  $L \in L_{cand}$ ,  $(L \cap L_{am})^{\downarrow C} = L$  and  $s_i, s_k \in BM(L)$  then  $\{[(L \setminus s_i)^{\uparrow C} \cap L_{am}]^{\downarrow C} \setminus s_k]^{\uparrow C} \cap L_{am}\}^{\downarrow C} = \{[(L \setminus s_k)^{\uparrow C} \cap L_{am}]^{\downarrow C} \setminus s_i]^{\uparrow C} \cap L_{am}\}^{\downarrow C}$ .

The proof of this proposition is also given in Appendix B. This result can also be extended to a general form with more than two strings in any order.

$$\{[(L \setminus s_1)^{\uparrow C} \cap L_{am}]^{\downarrow C} \dots \setminus s_m]^{\uparrow C} \cap L_{am}\}^{\downarrow C} = \{[(L \setminus s_m)^{\uparrow C} \cap L_{am}]^{\downarrow C} \dots \setminus s_1]^{\uparrow C} \cap L_{am}\}^{\downarrow C}$$

Proposition 3 means that a change in the order of applying  $BM(L)$ 's elements does not cause any difference on the final language. But as the transformation  $T$ 's result is obtained with an "if...else" condition, a change at the order of application may cause not to reach the same final language. Then the performance of these final solutions can be different. Then a formal method for obtaining the language having minimal performance measure is needed.

**Lemma 1** Let  $L \in L_{cand}$  be a language and  $s_i, s_j \in BM(L)$  be two possible blocking strings such that  $F(r(L, s_i)) > F(r(L, s_j))$  and  $(L \cap L_{am})^{\downarrow C} = L$  then  $\hat{J}(T(T(L, s_i), s_j)) \leq \hat{J}(T(T(L, s_j), s_i))$  always holds.

Proof of Lemma 1 is given in Appendix B. This result can easily be expanded to more than two strings. Here Lemma 1 expresses that if we order the blocking strings appropriately according to their loss factors, the transformation applied according to this order always reaches the smallest performance measure.

#### 4.2 Optimization Algorithm

Here we propose an algorithm to obtain the maximally permissive and also optimal blocking supervisor. Here in the algorithm  $BM(L_{temp\ i})\{i\}$  is used to symbolize the  $i$ th element of  $BM(L_{temp\ i})$ .

**Table 1. The algorithm.**

<p><b>Step 1:</b> Calculate <math>L_{IS} = L_{am}^{\downarrow C}</math> and <math>L_{temp\ 1} = L_{IS}</math></p> <p><b>Step 2:</b> Find <math>BM(L_{temp\ 1}) = \{s_1, s_2, s_3, \dots, s_n\}</math> For <math>\forall s_i \in BM(L_{temp\ 1})</math> calculate <math>F(r(L_{temp\ 1}, s_i))</math> and reorganize the <math>BM(L_{temp\ 1})</math> set according to its loss factor values in a descending order.</p> <p><b>Step 3:</b> For <math>i = 1</math> to <math>n</math>  <math>A = BM(L_{temp\ 1})(i)</math>  <math>L_x = T(L_{temp\ 1}, A)</math>  <math>L_{temp\ i+1} = L_x</math>  End  <math>L_{rst} = L_{temp\ i+1}</math></p> <p><b>Step 4:</b> If <math>L_{rst} \neq L_{temp\ 1}</math>  <math>L_{temp\ 1} = L_{rst}</math>  Goto step 2  else  <math>L_{FS} = L_{rst}</math></p>
---

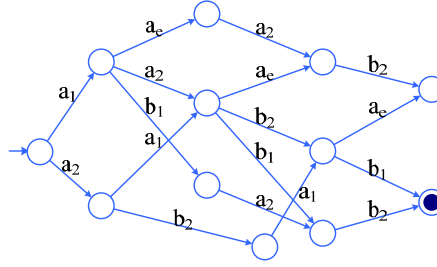
The structure of the algorithm is not complex. At step 1, the CSS is accepted as initial language and afterwards in step 2 the blocking strings set of it is found. For  $\forall s_i \in BM(L)$ ,  $P(r(L, s_i))$  value is calculated and the blocking strings set is reorganized according to these values in a descending order. At step 3, the transformation  $T$  is applied in turn. Finally at step 4  $L_{rst}$  is compared with  $L_{temp\ 1}$ . If they are equal,  $L_{rst}$  is accepted as final language else  $L_{rst}$  is appointed as the new initial language and the algorithms at steps 2 and 3 are applied again in turn in order. This algorithm is applied recursively until the final solution is obtained.

**Remark 3:** As we want to generate admissible marked strings as many as possible, we have to take into consideration all the admissible marked strings. So the CSS has to be selected as the initial language. Then according to the algorithm, the blockings of the CSS are examined and some of them are removed in a formal way at step 3. Here the execution sequence of these blockings is arranged by the rule at step 2. Due to Lemma 1, applying the blocking strings in this ordered sequence guarantees that the performance of final solution is always smaller or equal to the other execution order final solutions. On the other hand selecting the CSS as initial language makes the final solution to be maximally permissive supervisor because all the admissible marked strings are taken into consideration. As a result the obtained final solution will be maximally permissive and also optimal blocking supervisor. We can compute every cycle in polynomial complexity and if we assume that one blocking is removed from the language at each cycle then we need of  $n(n + 1)$  iteration steps at worst where  $n$  refers the number of the elements in  $BM(L_{am}^{\downarrow C})$ .

Here the algorithm will be explained over a database management system. A database management system must ensure that a database remains consistent despite the fact that many users may be doing retrievals and updates concurrently. The sequence of read and write operations of each user is called a transaction. When many transactions are being executed concurrently, the database management systems must schedule the interleaving of the operations of the transactions in order to preserve the consistency of the data. This is called concurrency control problem. The interesting aspect of concurrency control is that it is not necessary to completely prevent any interleaving of the events of individual transactions in order to obtain the correct result. Since the most popular concurrency control protocols are coupled with deadlock detection schemes, deadlock prevention and avoidance is impractical. Then a number of blockings can be allowed to generate as many admissible schedules as possible, since allowing more interleavings of the events from different transactions means less waiting on the part of the users who are submitting these transactions and thus better performance.

### 4.3 Example

Consider two transactions  $T_1 = a_1b_1$  and  $T_2 = a_2b_2$  such that  $\Sigma = \{a_1, b_1, a_2, b_2\}$ . Also in order to simplify the notation, we have omitted specifying if a given operation is write or read. At the same time we assumed that an error can be occurred during the transaction  $T_1$  with having a meaning of not concluding the transaction by some reason. This error is included to model by uncontrollable event  $a_e$ . The uncontrolled concurrent execution of  $T_1$  and  $T_2$  is modeled by automaton  $G_1$  depicted in Fig. 1. From the theory of

Fig. 1. Generator  $G_1$ .

database concurrency control, it can be shown that the only admissible marked strings are those where  $a_1$  precedes  $a_2$  if and only if  $b_1$  precedes  $b_1$ . Then admissible marked language will be as follows  $L_{am} = \{a_1b_1a_2b_2, a_2b_2a_1b_1, a_1a_2b_1b_2, a_2a_1b_2b_1\}$ . Also the uncontrollable event set is  $\Sigma_{uc} = \{a_e\}$ . Let the cost of some generated strings be as follows:  $c_s(a_1a_2b_1b_2) = 5$ ,  $c_s(a_1b_1a_2b_2) = 7.25$ ,  $c_s(a_1a_e) = 2$ ,  $c_s(a_1a_2a_e) = 9$ ,  $c_s(a_2a_2b_2b_1) = 8.5$ ,  $c_s(a_2b_2a_1b_1) = 15.5$ ,  $c_s(a_2a_1a_e) = 13.33$ ,  $c_s(a_2a_1b_2a_e) = 8$ ,  $c_s(a_2b_2a_1a_e) = 9.75$ .

**Step 1:**  $L_{IS} = \{\overline{a_1b_1a_2b_2}, \overline{a_1a_2b_1b_2}, \overline{a_2a_1b_2b_1}, \overline{a_2b_2a_1b_1}, a_1a_2a_e, a_1a_e, a_2a_1a_e, a_2a_1b_2a_e, a_2b_2a_1a_e\}$ ,  $L_{temp\ 1} = L_{IS}$ .

**Step 2:**  $BM(L_{temp\ 1}) = \{a_1a_e, a_1a_2a_e, a_2a_1a_e, a_2a_1b_2a_e, a_2b_2a_1a_e\}$ . The cost factors of blocking strings are as follows,  $F(r(L_{temp\ 1}, a_1a_e)) = -1.25$ ,  $F(r(L_{temp\ 1}, a_1a_2a_e)) = 4$ ,  $F(r(L_{temp\ 1}, a_2a_1a_e)) = 12.58$ ,  $F(r(L_{temp\ 1}, a_2a_1b_2a_e)) = 12.58$ ,  $F(r(L_{temp\ 1}, a_2b_2a_1a_e)) = -5.75$ .

The reordered blocking measure set is  $BM(L_{temp\ 1}) = \{a_2a_1b_2a_e, a_2a_1a_e, a_1a_2a_e, a_1a_e, a_2b_2a_1a_e\}$ .

**Step 3:** For  $i = 1$ ,  $A = BM(L_{temp\ 1})\{1\} = \overline{a_2a_1b_2a_e}$ . As  $\hat{J}[(L_{temp\ 1} \setminus A)^{\uparrow C} \cap L_{am}]^{\downarrow C} < \hat{J}[L_{temp\ 1}] \Rightarrow L_{temp\ 2} = T(L_{temp\ 1}, A) = \{\overline{a_1b_1a_2b_2}, \overline{a_1a_2b_1b_2}, \overline{a_2b_2a_1b_1}, a_1a_2a_e, a_1a_e, a_2b_2a_1a_e\}$ .

For  $i = 2$ ,  $A = a_2a_1a_e$ . As  $A = a_2a_1a_e \notin L_{temp\ 2} \Rightarrow ((L_{temp\ 2} \setminus A)^{\uparrow C} \cap L_{am})^{\downarrow C} = L_{temp\ 2}$  so  $L_{temp\ 3} = L_{temp\ 2}$ .

If the transformation is applied for the elements of  $BM(L_{temp\ 1})$  similarly,  $L_{rst}$  is obtained as  $\{\overline{a_1b_1a_2b_2}, \overline{a_2b_2a_1b_1}, a_1a_e, a_2b_2a_1a_e\}$ .

**Step 4:** As  $L_{rst} \neq L_{temp\ 1}$  then  $L_{temp\ 1}$  is taken as  $L_{rst}$ , steps 2 and 3 will be applied again.

**Step 2:**  $L_{temp\ 1} = \{\overline{a_1b_1a_2b_2}, \overline{a_2b_2a_1b_1}, a_1a_e, a_2b_2a_1a_e\}$ ,  $BM(L_{temp\ 1}) = \{a_1a_e, a_2b_2a_1a_e\}$ . As  $F(r(L_{temp\ 1}, a_1a_e)) = -5.25$  and  $F(r(L_{temp\ 1}, a_2b_2a_1a_e)) = -5.75$ , the reordered set is  $BM(L_{temp\ 1}) = \{a_1a_e, a_2b_2a_1a_e\}$ .

**Step 3:** If the transformation is applied for the elements of  $BM(L_{temp\ 1})$  in given order,  $L_{rst}$  is obtained as  $\{\overline{a_1b_1a_2b_2}, \overline{a_2b_2a_1b_1}, a_1a_e, a_2b_2a_1a_e\}$ .

**Step 4:** As  $L_{rst} = L_{temp 1}$  then  $L_{FS} = L_{rst} = L_{rst} = \overline{\{a_1b_1a_2b_2, a_2b_2a_1b_1, a_1a_e, a_2b_2a_1a_e\}}$ ,  $\widehat{SM}^C(L_{FS}) = \beta(a_1a_2b_1b_2, a_2a_1b_2b_1) = 13.5$ ,  $\widehat{BM}(L_{FS}) = \beta(a_1a_e, a_2b_2a_1a_e) = 11.75$  so  $\widehat{J}(L_{FS}) = \widehat{SM}^C(L_{FS}) + \widehat{BM}(L_{FS}) = 25.25$ .

Also  $\widehat{SM}(L_{IS}) = 0$ ,  $\widehat{BM}(L_{IS}) = \beta\{a_1a_2a_e, a_1a_e, a_2a_1a_e, a_2a_1b_2a_e, a_2b_2a_1a_e\} = 42.08$  so  $\widehat{J}(L_{IS}) = 42.08$ . Here it is seen that  $\widehat{J}(L_{IS}) < \widehat{J}(L_{FS})$ . The performance of the final language is better than initial language. On the other hand selecting the CSS as initial language provides the final solution to be maximally permissive supervisor.

## 5. CONCLUSION

When we focus on industrial examples, supervisory control problem with blocking arises more generally than the supervisory control problems of nonblocking and also blocking supervisors are preferred to nonblocking ones in several applications. Database concurrency control is a good example for this case. So the topic of blocking in supervisory control of DES is studied with a different approach and also an explanation for a better understanding of blocking is given in this paper. In this context, we offer a new metric space and propose a new performance measure that depends on numeric values obtained from blockings and failure. Then we present a new transformation according to this performance measure and construct a new optimization algorithm to minimize it. Here the proposed algorithm also gives the maximally permissive solution. Also the uniqueness of the solution is explained. Finally the proposed algorithm is explained over database system with having two transactions and the performance of it is compared. Future work will be concerned with relaxing the assumptions on languages.

## REFERENCES

1. C. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*, Kluwer Academic Publishers, Massachusetts, USA, 1999,
2. E. Chen and S. Lafortune, "Dealing with blocking in supervisory control of discrete-event systems," *IEEE Transactions on Automatic Control*, Vol. 36, 1991, pp. 724-735.
3. F. L. Lewis, *et al.*, "Analysis of deadlock and circular waits using a matrix model for flexible manufacturing systems," *Automatica*, Vol. 34, 1998, pp. 1083-1100.
4. K. Passino and P. J. Antsaklis, "On optimal control of discrete event systems," in *Proceedings of the 28th IEEE Decision and Control Conference*, 1989, pp. 2713-2718.
5. R. Kumar and V. K. Garg, "Optimal supervisory control of discrete event dynamical systems," *SIAM Journal of Control and Optimization*, Vol. 33, 1995, pp. 419-439.
6. R. Sengupta and S. Lafortune, "An optimal control theory for discrete event systems," *SIAM Journal of Control and Optimization*, Vol. 26, 1998, pp. 448-541.
7. A. Ray, A. Surana, and S. Phoha, "A language measure for supervisory control," *Applied Mathematics Letters*, Vol. 16, 2003, pp. 985-991.
8. A. Ray, J. Fu, and C. M. Lagoa, "Optimal supervisory control of finite state automata," *International Journal of Control*, Vol. 77, 2004, pp. 1083-1100.

9. O. Kaymakci and S. Kurtulan, "A metric space approach for a class of discrete event systems," in *Proceedings of the 12th Mediterranean Conference on Control and Automation*, 2004.
10. O. Kaymakci, S. Kurtulan, and L. Goren, "Improving the behaviour of supervisor under blocking," in *Proceedings of the 16th IFAC World Congress*, 2005.
11. S. Lafortune and E. Chen, "The Infimal closed controllable superlanguage and its application in supervisory control," *IEEE Transactions on Automatic Control*, Vol. 35, 1990, pp. 398-405.
12. P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event systems," *SIAM Journal Control and Optimization*, Vol. 25, 1987, pp. 206-230.
13. P. J. Ramadge and W. M. Wonham, "Modular feedback logic for discrete event systems," *SIAM Journal Control and Optimization*, Vol. 25, 1987, pp. 1202-1218.
14. W. M. Wonham and P. J. Ramadge, "On the supremal controllable sublanguage of a given language," *SIAM Journal of Control and Optimization*, Vol. 25, 1987, pp. 637-659.
15. E. Kreyszig, *Introductory Functional Analysis with Applications*, John Wiley & Sons, New York, 1989.

## APPENDIX A

**Property 1** If  $L = \bar{L}$  and  $\alpha_i \in BM(L)$  then  $\overline{(L \setminus \alpha_i)^{\uparrow C}} = (L \setminus \alpha_i)^{\uparrow C}$ .

**Proof:**  $(\bar{L})^{\uparrow C} \supseteq \overline{L^{\uparrow C}}$  always holds. Also  $\overline{L^{\uparrow C}} \supseteq L^{\uparrow C}$  always holds due to prefix-closed operation. Then  $\overline{L^{\uparrow C}} \supseteq (\bar{L})^{\uparrow C}$  due to  $L = \bar{L}$ . So  $\overline{L^{\uparrow C}} = (\bar{L})^{\uparrow C}$ .

$$\text{As a result } \overline{L^{\uparrow C}} = L^{\uparrow C}. \quad (1)$$

$$\text{As } \alpha_i \in BM(L) \text{ and } L = \bar{L} \text{ then } L \setminus \alpha_i = \overline{L \setminus \alpha_i}. \quad (2)$$

According to Eqs. (1) and (2),  $\overline{(L \setminus \alpha_i)^{\uparrow C}} = (L \setminus \alpha_i)^{\uparrow C}$ . □

**Property 2** If  $L \in L_{cand}$ ,  $L = \bar{L}$  and  $\alpha_i \in BM(L)$  then  $\alpha_i \notin [(L \setminus \alpha_i)^{\uparrow C} \cap L_{am}]^{\downarrow C}$ .

**Proof:** As  $\alpha_i \in L$  and the monotony of  $\uparrow C$ ,  $(L \setminus \alpha_i)^{\uparrow C} \subseteq L \setminus \alpha_i$  always holds. (3)

At the same time  $(L \setminus \alpha_i)^{\uparrow C} \cap L_{am} \subseteq (L \setminus \alpha_i)^{\uparrow C}$  always holds. As  $(L \setminus \alpha_i)^{\uparrow C}$  is prefixed closed and controllable according to Property 1 so  $[(L \setminus \alpha_i)^{\uparrow C}]^{\downarrow C} = (L \setminus \alpha_i)^{\uparrow C}$ .

$$[(L \setminus \alpha_i)^{\uparrow C} \cap L_{am}]^{\downarrow C} \subseteq [(L \setminus \alpha_i)^{\uparrow C}]^{\downarrow C} = (L \setminus \alpha_i)^{\uparrow C}. \quad (4)$$

When the results of Eqs. (3) and (4) are joined,  $[(L \setminus \alpha_i)^{\uparrow C} \cap L_{am}]^{\downarrow C} \subseteq L \setminus \alpha_i$  is obtained. As  $\alpha_i \notin L \setminus \alpha_i$  so  $\alpha_i \notin [(L \setminus \alpha_i)^{\uparrow C} \cap L_{am}]^{\downarrow C}$ . □

**Property 3** For  $\alpha_i \in BM(L)$  and  $L \in L_{cand}$  then  $[(L \setminus \alpha_i)^{\uparrow C} \cap L_{am}]^{\downarrow C}$  is always a member of  $L_{cand}$ .

**Proof:**  $\overline{L_{am}^{\uparrow C}} \subseteq L \subseteq L_{am}^{\downarrow C}$  so  $[\overline{L_{am}^{\uparrow C}} \setminus \alpha_i]^{\uparrow C} \subseteq (L \setminus \alpha_i)^{\uparrow C} \subseteq [L_{am}^{\downarrow C} \setminus \alpha_i]^{\uparrow C}$ . (5)

For  $\forall \alpha_i \in BM(L) \Rightarrow \alpha_i \notin \overline{L_{am}^{\uparrow C}}$  then  $[\overline{L_{am}^{\uparrow C}} \setminus \alpha_i]^{\uparrow C} = \overline{L_{am}^{\uparrow C}}$ . (6)

$[L_{am}^{\downarrow C} \setminus \alpha_i]^{\uparrow C} \subseteq L_{am}^{\downarrow C}$  always holds due to monotony of supremal operation. (7)

When we integrate Eqs. (6) and (7) to Eq. (5),  $\overline{L_{am}^{\uparrow C}} \subseteq (L \setminus \alpha_i)^{\uparrow C} \subseteq L_{am}^{\downarrow C}$  is obtained.  $[\overline{L_{am}^{\uparrow C}} \cap L_{am}]^{\downarrow C} \subseteq [(L \setminus \alpha_i)^{\uparrow C} \cap L_{am}]^{\downarrow C} \subseteq [L_{am}^{\downarrow C} \cap L_{am}]^{\downarrow C}$ .

At the same time  $[\overline{L_{am}^{\uparrow C}} \cap L_{am}]^{\downarrow C} = \overline{L_{am}^{\uparrow C}}$  and  $[L_{am}^{\downarrow C} \cap L_{am}]^{\downarrow C} = L_{am}^{\downarrow C}$  then  $\overline{L_{am}^{\uparrow C}} \subseteq [(L \setminus \alpha_i)^{\uparrow C} \cap L_{am}]^{\downarrow C} \subseteq L_{am}^{\downarrow C}$ . (8)

As  $[(L \setminus \alpha_i)^{\uparrow C} \cap L_{am}]^{\downarrow C}$  is controllable and prefix-closed, according to Eq. (8)  $[(L \setminus \alpha_i)^{\uparrow C} \cap L_{am}]^{\downarrow C} \in L_{cand}$ .  $\square$

## APPENDIX B

**Proof of Proposition 2:** As  $L_1 \subset L_2$  then  $L_{1m} = L_{2m} \setminus [(L_2 \setminus L_1) \cap L_{am}]$ .

$$\begin{aligned} L_{am} \setminus L_{1m} &= L_{am} \setminus \{L_{2m} \setminus [(L_2 \setminus L_1) \cap L_{am}]\} = \{L_{am} \setminus L_{2m}\} \cup \{L_{am} \cap [(L_2 \setminus L_1) \cap L_{am}]\} \\ &= \{L_{am} \setminus L_{2m}\} \cup \{(L_2 \setminus L_1) \cap L_{am}\} \Rightarrow \beta\{L_{am} \setminus L_{1m}\} = \beta\{L_{am} \setminus L_{2m}\} + \beta\{(L_2 \setminus L_1) \cap L_{am}\} \end{aligned} \quad (9)$$

On the other hand

$$\begin{aligned} L_2 \setminus \overline{L_{2m}} &= (L_1 \cup (L_2 \setminus L_1)) \setminus \overline{[L_1 \cup (L_2 \setminus L_1)] \cap L_{am}} \\ &= \{L_1 \setminus \overline{L_{1m}} \cup [(L_2 \setminus L_1) \cap L_{am}]\} \cup \{(L_2 \setminus L_1) \setminus \overline{L_{1m}} \cup [(L_2 \setminus L_1) \cap L_{am}]\} \\ &= \{(L_1 \setminus \overline{L_{1m}}) \cap (L_1 \setminus \overline{(L_2 \setminus L_1) \cap L_{am}})\} \cup \{[(L_2 \setminus L_1) \setminus \overline{L_{1m}}] \cap [(L_2 \setminus L_1) \setminus \overline{(L_2 \setminus L_1) \cap L_{am}}]\}. \end{aligned}$$

As  $L_1 \in L_{cand}$  then  $L_1 \supseteq \overline{L_{1m}}$  so  $(L_2 \setminus L_1) \cap \overline{L_{1m}} = \emptyset \Rightarrow (L_2 \setminus L_1) \setminus \overline{L_{1m}} = (L_2 \setminus L_1)$ . Then the equation is equal to  $= \{(L_1 \setminus \overline{L_{1m}}) \cap (L_1 \setminus \overline{(L_2 \setminus L_1) \cap L_{am}})\} \cup \{(L_2 \setminus L_1) \setminus \overline{(L_2 \setminus L_1) \cap L_{am}}\}$ .

If  $(L_1 \cap \overline{L_{2m}}) \setminus \overline{L_{1m}} = \emptyset$  then  $L_1 \setminus \overline{(L_2 \setminus L_1) \cap L_{am}} = L_1$ . As a result,  $L_2 \setminus \overline{L_{2m}} = \{L_1 \setminus \overline{L_{1m}}\} \cup \{(L_2 \setminus L_1) \setminus \overline{(L_2 \setminus L_1) \cap L_{am}}\}$ ,  $\beta\{L_2 \setminus \overline{L_{2m}}\} = \beta\{\{L_1 \setminus \overline{L_{1m}}\} \cup \{(L_2 \setminus L_1) \setminus \overline{(L_2 \setminus L_1) \cap L_{am}}\}\}$ .

$$\begin{aligned} \text{As } \{L_1 \setminus \overline{L_{1m}}\} \cap \{(L_2 \setminus L_1) \setminus \overline{(L_2 \setminus L_1) \cap L_{am}}\} &= \emptyset, \beta\{L_2 \setminus \overline{L_{2m}}\} = \beta\{L_1 \setminus \overline{L_{1m}}\} + \beta\{(L_2 \setminus L_1) \setminus \overline{(L_2 \setminus L_1) \cap L_{am}}\}, \\ \beta\{L_1 \setminus \overline{L_{1m}}\} &= \beta\{L_2 \setminus \overline{L_{2m}}\} - \beta\{(L_2 \setminus L_1) \setminus \overline{(L_2 \setminus L_1) \cap L_{am}}\}. \end{aligned} \quad (10)$$

Then due to Eqs. (9) and (10), we obtain  $\beta\{L_1 \setminus \overline{L_{1m}}\} + \beta\{L_{am} \setminus L_{1m}\} = \beta\{L_2 \setminus \overline{L_{2m}}\} + \beta\{L_{am} \setminus L_{2m}\} - \beta\{(L_2 \setminus L_1) \setminus \overline{(L_2 \setminus L_1) \cap L_{am}}\} + \beta\{(L_2 \setminus L_1) \cap L_{am}\}$ .

As a result  $\hat{J}(L_1) = \hat{J}(L_2) - F(L_2 \setminus L_1)$  always holds.  $\square$

**Proof of Proposition 3:** For any pair of  $\alpha_i, \alpha_k \in BM(L)$ , we can write

As  $L \in L_{cand}$  and  $(L \cap L_{am})^{\downarrow C} = L$ ,  $(L \setminus \alpha_i)^{\uparrow C} \subseteq L$  and  $((L \setminus \alpha_i)^{\uparrow C} \cap L_{am})^{\uparrow C} \subseteq (L \cap L_{am})^{\downarrow C} = L$  always holds. Then  $\{((L \setminus \alpha_i)^{\uparrow C} \cap L_{am})^{\uparrow C} \setminus \alpha_i\}^{\uparrow C} \cap L_{am}^{\downarrow C} \subseteq ((L \setminus \alpha_i)^{\uparrow C} \cap L_{am})^{\uparrow C}$ .

$$\{\{((L \setminus \alpha_i)^{\uparrow C} \cap L_{am})^{\downarrow C} \setminus \alpha_k\}^{\uparrow C} \cap L_{am}^{\downarrow C} \setminus \alpha_i\}^{\uparrow C} \cap L_{am}^{\downarrow C} \subseteq \{((L \setminus \alpha_k)^{\uparrow C} \cap L_{am})^{\downarrow C} \setminus \alpha_i\}^{\uparrow C} \cap L_{am}^{\downarrow C}. \quad (11)$$

As  $\alpha_i \in BM(L)$  and  $(L \cap L_{am})^{\downarrow C} = L$  then  $\alpha_i \notin \{((L \setminus \alpha_i)^{\uparrow C} \cap L_{am})^{\downarrow C} \setminus \alpha_k\}^{\uparrow C} \cap L_{am}^{\downarrow C}$ . (12)

The left part of Eq. (11) will be reorganized according to Eq. (12) then Eq. (13) is obtained.

$$\{((L \setminus \alpha_i)^{\uparrow C} \cap L_{am})^{\downarrow C} \setminus \alpha_k\}^{\uparrow C} \cap L_{am}^{\downarrow C} \subseteq \{((L \setminus \alpha_k)^{\uparrow C} \cap L_{am})^{\downarrow C} \setminus \alpha_i\}^{\uparrow C} \cap L_{am}^{\downarrow C} \quad (13)$$

Since  $\alpha_i$  and  $\alpha_k$  are arbitrary elements then

$$\{((L \setminus \alpha_i)^{\uparrow C} \cap L_{am})^{\downarrow C} \setminus \alpha_k\}^{\uparrow C} \cap L_{am}^{\downarrow C} = \{((L \setminus \alpha_k)^{\uparrow C} \cap L_{am})^{\downarrow C} \setminus \alpha_i\}^{\uparrow C} \cap L_{am}^{\downarrow C}. \quad \square$$

**Proof of Lemma 1:**  $F(r(L, s_i)) > F(r(L, s_j))$  holds true in three ways. These are  $0 > F(r(L, s_i)) > F(r(L, s_j))$ ,  $F(r(L, s_i)) > 0 > F(r(L, s_j))$  and  $F(r(L, s_i)) > F(r(L, s_j)) > 0$ , respectively. Now the relation between  $T(T(L, s_i), s_j)$  and  $T(T(L, s_j), s_i)$  will be investigated in these three situations.

- Let  $0 > F(r(L, s_i)) > F(r(L, s_j))$ .

Also  $\hat{J}(((L \setminus s_i)^{\uparrow C} \cap L_{am})^{\downarrow C}) = \hat{J}(L) - F(r(L, s_i))$  always holds according to Proposition 2 and as  $0 > F(r(L, s_i))$  then  $\hat{J}(((L \setminus s_i)^{\uparrow C} \cap L_{am})^{\downarrow C}) > \hat{J}(L)$  so  $T(L, s_i) = L$ . Similarly as  $0 > F(r(L, s_j))$ ,  $T(L, s_j) = L$ . Finally  $T(T(L, s_i), s_j) = L$  and  $T(T(L, s_j), s_i) = L$  then

$$\Rightarrow \hat{J}[T(T(L, s_i), s_j)] = \hat{J}[T(T(L, s_j), s_i)]. \quad (14)$$

- Let  $F(r(L, s_i)) > 0 > F(r(L, s_j))$ .

Here in this situation the  $T(L, s_i) = T(L, s_j) = [(L \setminus s_i)^{\uparrow C} \cap L_{am}]^{\downarrow C}$  always holds due to  $0 < F(r(L, s_i))$  and Proposition 2.

And at the same time for the removed strings  $r(L, s_j)$  and  $r(L, s_i)$ , three possible conditions occur. These are  $r(L, s_i) \cap r(L, s_j) = \emptyset$ ,  $r(L, s_i) \supseteq r(L, s_j)$  and  $r(L, s_i) \subseteq r(L, s_j)$ , respectively. Now the results of transformations will be investigated and compared for these three conditions.

If  $r(L, s_i) \cap r(L, s_j) = \emptyset$  then  $T(T(L, s_i), s_j) = T(L, s_i) = [(L \setminus s_i)^{\uparrow C} \cap L_{am}]^{\downarrow C}$  and  $T(L, s_j) = L \Rightarrow T(T(L, s_j), s_i) = [(L \setminus s_i)^{\uparrow C} \cap L_{am}]^{\downarrow C}$ . As a result  $T(T(L, s_i), s_j) = T(T(L, s_j), s_i)$

$$\Rightarrow \hat{J}[T(T(L, s_i), s_j)] = \hat{J}[T(T(L, s_j), s_i)]. \quad (15)$$

If  $r(L, s_i) \supseteq r(L, s_j)$  then  $s_j \notin T(L, s_i)$  so  $T(T(L, s_i), s_j) = T(L, s_i) = [(L \setminus s_i)^{\uparrow C} \cap L_{am}]^{\downarrow C}$ . On the other hand as  $0 > F(r(L, s_j))$ ,  $T(L, s_i) = L \Rightarrow T(T(L, s_j), s_i) = [(L \setminus s_j)^{\uparrow C} \cap L_{am}]^{\downarrow C}$ . As a result  $T(T(L, s_i), s_j) = T(T(L, s_j), s_i)$

$$\Rightarrow \hat{J}[T(T(L, s_i), s_j)] = \hat{J}[T(T(L, s_j), s_i)]. \quad (16)$$

If  $r(L, s_i) \subseteq r(L, s_j)$  then  $F(r(T(L, s_i), s_j)) = F(r(L, s_j)) - F(r(L, s_i))$ . Also at the same time  $((L \setminus s_i)^{\uparrow C} \cap L_{am})^{\downarrow C} \cap \overline{L_m} \setminus ((L \setminus s_i)^{\uparrow C} \cap L_{am})^{\downarrow C} \cap L_{am} = \emptyset$ . Then as  $F(r(T(L, s_i), s_j))$  is smaller than zero so  $T(T(L, s_i), s_j) = T(L, s_i) = [(L \setminus s_i)^{\uparrow C} \cap L_{am}]^{\downarrow C}$ . On the other hand as  $0 > F(r(L, s_j))$  so  $T(L, s_j) = L \Rightarrow T(T(L, s_j), s_i) = T(L, s_i) = [(L \setminus s_i)^{\uparrow C} \cap L_{am}]^{\downarrow C}$ . So  $T(T(L, s_i), s_j) = T(T(L, s_j), s_i)$

$$\Rightarrow \hat{J}[T(T(L, s_i), s_j)] = \hat{J}[T(T(L, s_j), s_i)]. \quad (17)$$

- Let  $F(r(L, s_i)) > F(r(L, s_j)) > 0$ .

Here the  $T(L, s_i) = [(L \setminus s_i)^{\uparrow C} \cap L_{am}]^{\downarrow C}$  always holds due to  $0 > F(r(L, s_j))$ .

If  $r(L, s_i) \cap r(L, s_j) = \emptyset$  then  $T(T(L, s_i), s_j) = (((L \setminus s_i)^{\uparrow C} \cap L_{am})^{\downarrow C} \setminus s_j)^{\uparrow C} \cap L_{am})^{\downarrow C}$ . On the other hand  $T(T(L, s_j), s_i) = (((L \setminus s_j)^{\uparrow C} \cap L_{am})^{\downarrow C} \setminus s_i)^{\uparrow C} \cap L_{am})^{\downarrow C}$ . As these two results are same according to Proposition 3 then  $T(T(L, s_i), s_j) = T(T(L, s_j), s_i)$

$$\Rightarrow \hat{J}[T(T(L, s_i), s_j)] = \hat{J}[T(T(L, s_j), s_i)]. \quad (18)$$

If  $r(L, s_i) \supseteq r(L, s_j)$  then as  $s_j \notin T(L, s_i)$  so  $T(T(L, s_i), s_j) = T(L, s_i) = [(L \setminus s_i)^{\uparrow C} \cap L_{am}]^{\downarrow C}$ . On the other hand as  $F(r(L, s_j)) > 0$  then  $T(L, s_j) = [(L \setminus s_j)^{\uparrow C} \cap L_{am}]^{\downarrow C}$ . Also  $F(r(T(L, s_j), s_i)) = F(r(L, s_i)) - F(r(L, s_j))$  and  $F(r(L, s_i)) - F(r(L, s_j)) > 0$  so  $F(r(T(L, s_j), s_i)) > 0$ . As a result  $T(T(L, s_j), s_i) = (((L \setminus s_j)^{\uparrow C} \cap L_{am})^{\downarrow C} \setminus s_i)^{\uparrow C} \cap L_{am})^{\downarrow C}$ . Also in accordance with  $r(L, s_i) \supseteq r(L, s_j)$  and Proposition 3,  $T(T(L, s_i), s_j) = T(T(L, s_j), s_i)$ . So

$$\hat{J}[T(T(L, s_i), s_j)] = \hat{J}[T(T(L, s_j), s_i)]. \quad (19)$$

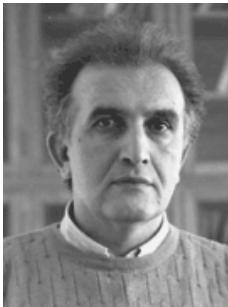
If  $r(L, s_i) \subseteq r(L, s_j)$  then  $T(L, s_i) = [(L \setminus s_i)^{\uparrow C} \cap L_{am}]^{\downarrow C}$ . Also  $F(r(T(L, s_i), s_j)) = F(r(L, s_j)) - F(r(L, s_i))$  is always smaller than zero as  $F(r(L, s_j)) > F(r(L, s_i)) > 0$ . Then  $T(T(L, s_i), s_j) = T(L, s_i) = [(L \setminus s_i)^{\uparrow C} \cap L_{am}]^{\downarrow C}$ . On the other hand  $T(L, s_j) = [(L \setminus s_j)^{\uparrow C} \cap L_{am}]^{\downarrow C}$  and  $r(L, s_i) \subseteq r(L, s_j)$  then  $s_j \notin T(L, s_j)$ . So  $T(T(L, s_j), s_i) = T(L, s_j) = [(L \setminus s_j)^{\uparrow C} \cap L_{am}]^{\downarrow C}$ . As  $F(r(L, s_i)) > F(r(L, s_j)) > 0$  then  $\hat{J}[T(L, s_i)] < \hat{J}[T(L, s_j)]$  holds. As a result

$$\hat{J}[T(T(L, s_i), s_j)] < \hat{J}[T(T(L, s_j), s_i)]. \quad (20)$$

When we join the results in Eqs. (14)-(20)  $\hat{J}[T(T(L, s_i), s_j)] \leq \hat{J}[T(T(L, s_j), s_i)]$  is obtained.  $\square$



**Ozgur Kaymakci** received his Ph.D. and M.S. degrees in Control Systems Engineering from Istanbul Technical University, Istanbul, Turkey, in 2007 and 2000, respectively, and his B.S. degree in Electrical Engineering from Istanbul Technical University, Istanbul, Turkey, in 1998. He has been a Research Assistant of Control Systems Engineering, Istanbul Technical University, Istanbul, Turkey, since 1998. His research interests include discrete event systems, industrial automation systems, motion control systems and industrial communication.



**Salman Kurtulan** received his Ph.D. and M.S. degrees in Control and Computer Systems Engineering from Istanbul Technical University, Istanbul, Turkey, in 1992 and 1985, respectively, and his B.S. degree in Electrical Engineering from Istanbul Technical University, Istanbul, Turkey, in 1982. He has been an Associate Professor of Control Systems Engineering, I.T.U., Istanbul, Turkey, since 1995. His research interests are industrial automation systems, PLC applications, electrical transportation systems and control of electrical drives.