On the linear separability of the safe and unsafe state subsets of Single-Unit Resource Allocation Systems

Spyros Reveliotis

Abstract— **The main purpose of this correspondence is to demonstrate, through a counter-example, that, contrary to what was published recently in [1], the states belonging to the safe and unsafe subspaces of a single-unit resource allocation system (SU-RAS) might not be linearly separable. In addition, we identify a particular sub-class of SU-RAS for which the aforementioned property is guaranteed.**

I. Introduction

The problem of deadlock avoidance in sequential resource allocation systems (RAS) is well-established in the current control literature [2], [3]. In its basic definition, a sequential RAS Φ consists of a set of resource types, $\mathcal{R} = \{R_1, \ldots, R_m\}$, that are shared by a set of concurrently executing process types, $\mathcal{J} = \{J_1, \ldots, J_n\}$. Resources are reusable and each resource type R_i, $i = 1, \ldots, m$, is available in C_i identical units that define the resource capacity. On the other hand, each process type J_j, $j = 1, \ldots, n$, is further defined by a set of stages $\mathcal{S}_j = \{\Xi_{j1}, \ldots, \Xi_{j,l(j)}\}$, and some (appropriately coded) sequential logic that describes all the possible ways that an instance j_j of process type J_j can execute through the stages of $\mathcal{S}_j$. Furthermore, the proper execution of any stage Ξ_{jk}, $j = 1, \ldots, n$, $k = 1, \ldots, l(j)$, by a process instance j_j requires the exclusive allocation to j_j of a (non-empty) subset of resources $\mathcal{A}(\Xi_{jk})$; this set is typically represented as an m-dimensional vector with its i-th component indicating the stage requirement with respect to (w.r.t.) resource type R_i. Finally, it is assumed that a process instance j_j seeking to advance from a stage Ξ_{jk} to a successor stage $\Xi_{jk'}$ must first secure the resource differential $[\mathcal{A}(\Xi_{jk'}) - \mathcal{A}(\Xi_{jk})]^+$, and only then it will release any previously held resources that are not needed for the execution of the new stage $\Xi_{jk'}$.[1] This "hold-while-waiting" resource allocation protocol, when combined with the arbitrary structure of the resource allocation sequences involved, can give rise to deadlock, i.e., situations where a subset of the active process instances cannot advance any further in their potential process routes because each of them requests resources for its further advancement that are held by some other process in the set.

The investigation of the resource allocation dynamics that were described in the previous paragraph, and the corresponding problem of deadlock formation and avoidance, can be further formalized using concepts and results borrowed from qualitative Discrete Event Systems (DES) theory [4]. In particular, both modeling frameworks of Finite State Automata (FSA) [4] and Petri nets (PNs) [4] have been used extensively for the modeling of sequential RAS and the study of the deadlock avoidance problem [2]. In the FSA modeling framework, the RAS state is modeled by a ξ-dimensional vector s, where $\xi = \sum_{j=1}^{n} |\mathcal{S}_j|$ and the components of s are in one-to-one correspondence with the RAS processing stages; component $s[k] \equiv s_k$, $k = 1, \ldots, \xi$, corresponds to the processing stage $\Xi(s_k)$, and its value indicates the number of process instances that are executing this processing stage in state s. The event set of this FSA is defined by the loading and unloading of the various process instances, as well as their advancement among their various processing stages. The automaton is initiated at the state $s_0 = \mathbf{0}$ that represents the system empty state, and the same state is also the unique marked state of this automaton, a fact the represents the requirement for successful completion of each initiated process instance.

Let the reachable state space of a RAS Φ, when started from its initial state s_0, be denoted by S_r. Also, the subspace of Φ consisting by states that are co-reachable to state s_0 will be denoted by S_s and referred to as the set of "safe" states. Under this FSA representation of the RAS dynamics, maximally permissive deadlock avoidance implies the confinement of RAS Φ in its reachable and co-reachable subspace $S_{rs} \equiv S_r \cap S_s$. The complement of S_{rs} w.r.t. S_r is denoted by $S_{r\bar{s}}$ and it is referred to as the (set of reachable and) "unsafe" states.

The cardinality of set S_r and its two subsets S_{rs} and $S_{r\bar{s}}$ is, in general, a super-polynomial function of the size of any parsimonious description of the structure of RAS Φ (where the latter is defined by the items that were listed in the opening paragraph) [2]. Hence, the implementation of the maximally permissive deadlock avoidance policy (DAP) through one-step-lookahead schemes that are based on the explicit enumeration and storage of the sets S_{rs} and $S_{r\bar{s}}$ is practically intractable. However, a recent line of research has established that the aforementioned one-step look-ahead schemes can be pertinently implemented by perceiving the corresponding supervisory control problem as a classification problem concerning the dichotomy of the state (vector) set S_r to its safe and unsafe subsets S_{rs} and $S_{r\bar{s}}$; in this way, the relevant control-synthesis problem boils down to the design of the necessary classifier that will provide an efficient representation of the sought dichotomy [5], [6], [7]. Among the classes of the pursued classifiers, the most conspicuous one is the class of linear classifiers, which represents the sought dichotomy through a set of linear inequalities in the RAS state [5], [7]. The popularity of this class is due to (i) the conceptual and computational simplicity of the synthesis of the target classifiers compared to the classifiers with a non-linear structure, and (ii) the possibility of representing the obtained classifiers in the PN modeling framework in a simple and efficient manner, using the theory of "monitor" places of [8], [9]. However, it has also been established that the class of linear classifiers is not a complete representation for the maximally permissive DAP of the considered RAS; i.e., there are RAS Φ for which the sets S_{rs} and $S_{r\bar{s}}$ are not separable through a set of linear inequalities [6]. Detailed analytical conditions that must be satisfied by the sets S_{rs} and $S_{r\bar{s}}$ in order to be linearly separable are provided in [10], [11].

The above discussion raises naturally the additional question of what type of conditions can be imposed on the RAS structure itself so that the maximally permissive DAP for the resulting RAS sub-class is representable as a linear classifier. Along these lines, the work of [5] has established that this property is possessed by RAS where every processing stage involves at least one resource type R_i of unit capacity (i.e., with $C_i = 1$), since,

S. Reveliotis is with the School of Industrial & Systems Engineering, Georgia Institute of Technology, email: spyros@isye.gatech.edu

[1]We notice, for completeness, that this operational assumption is not restrictive in any sense, since pure resource releases can still be modeled through the introduction of additional processing stages.

TABLE I
THE CONSIDERED SU-RAS

$\mathcal{R} = \{R_1, R_2, R_3, R_4, R_5.R_6\}$
$C_1 = C_2 = C_5 = C_6 = 1$; $C_3 = C_4 = 2$
$\mathcal{J} = \{J_1, J_2, J_3, J_4\}$
$J_1 \ : \ R_1 \rightarrow R_2 \rightarrow R_3 \rightarrow R_4 \rightarrow R_5 \rightarrow R_6$
$J_2 \ : \ R_6 \rightarrow R_5 \rightarrow R_4 \rightarrow R_3 \rightarrow R_2 \rightarrow R_1$
$J_3 \ : \ R_3 \rightarrow R_4 \rightarrow R_5 \rightarrow R_6$
$J_4 \ : \ R_4 \rightarrow R_3 \rightarrow R_2 \rightarrow R_1$

in that case, the reachable state space S_r can be represented by binary vectors only, and any set of binary vectors can be dichotomized, in any way, by a set of linear inequalities. More recently, the work of [1] has claimed a similar result for the RAS sub-class where (i) process instances execute as atomic entities (i.e., there are no process merging and splitting operations in the relevant sequential logic), (ii) there is potential routing flexibility but no looping in the corresponding execution paths, and finally, (iii) the vectors that define the resource allocation function $\mathcal{A}(\cdot)$ are m-dimensional unit vectors (i.e., every processing stage requires only a single unit from a single resource type). Because of trait (iii), this RAS sub-class has been characterized as the "Single-Unit" (SU-)RAS in the relevant literature [2]. In the PN modeling framework, SU-RAS are modeled by a particular PN sub-class that is known as the S^3PR nets, and it is the actual representation used in [1]. The main purpose of this article is to show that, contrary to the claims of [1], there are SU-RAS (or, equivalently, S^3PR nets) for which the maximally permissive DAP is not representable by a set of linear inequalities. We establish this result in the next section by detailing and analyzing such an SU-RAS. Also, in Section III we provide some further discussion that elucidates some misconceptions in the developments of [1] that led to the erroneous result. This discussion also reveals a particular subclass of SU-RAS, of practical interest, for which the claimed result in [1] actually holds true. Finally, Section IV concludes the paper.

II. THE PROVIDED COUNTER-EXAMPLE

Consider the SU-RAS depicted in Table I. This RAS consists of six resource types with the corresponding capacities annotated in the table, and four process types. Each process type constitutes a linear sequence of processing stages, and since the considered RAS is of the single-unit type, in Table I we characterize the corresponding resource allocation function $\mathcal{A}(\Xi_{jk})$ by simply providing the single resource type that is requested by each processing stage. The reader should also notice that the considered RAS has 20 processing stages in total, and therefore, the corresponding state vector s is a 20-dimensional vector. However, for reasons of representational economy, in the following we shall adopt a symbolic representation that represents any state $s = \{s_1, \ldots, s_{20}\}$ by $\sum_{i=1}^{20} s_i \cdot \Xi(s_i)$; under this new representation, only components with non-zero coefficients will appear in the provided sum.

Consider the state $s_1 = 1 \cdot \Xi_{11} + 1 \cdot \Xi_{12} + 1 \cdot \Xi_{21} + 1 \cdot \Xi_{22} + 2 \cdot \Xi_{31}$, and name the process instances encountered when parsing the above sum from left to right by j_1 to j_6; i.e., j_1 is the process instance executing processing stage Ξ_{11}, j_2 is the process instance executing processing stage Ξ_{12}, j_3 is the process instance executing processing stage Ξ_{21}, j_4 is the process instance executing processing stage Ξ_{22}, and eventually, j_5 and j_6 are the two process instances executing processing stage Ξ_{31}. State s_1 is clearly in the reachable state space S_r of the considered RAS. Next, we shall show that s_1 is also in the safe subspace S_{rs}. For this, it suffices to provide an event sequence that completes successfully all the active process instances. Such an event sequence can be constructed as follows: First, advance one of the process instances executing stage Ξ_{31}, let's say j_5, to its next processing stage, Ξ_{32}; this advancement is possible since resource R_4 has no processes allocated to it in state s_1. Let the resulting state be denoted by s_1^1. In state s_1^1, each of the multi-capacity resources R_3 and R_4 has only one of its two units allocated. This enables the advancement of process instances j_3 and j_4 respectively to their processing stages Ξ_{23} and Ξ_{24}, and the release of the unit-capacity resources R_5 and R_6; let the corresponding state be denoted by s_1^2. Since resources R_5 and R_6 are free in state s_1^2, process instances j_5 and j_6 can advance to their completion, one at a time; let the resulting state be denoted by s_1^3. In state s_1^3, each of the resources R_3 to R_6 has at least one unit of free capacity; hence, process instances j_2 and j_1 can advance to their completion, one at a time. Let the resulting state be denoted by s_1^4. In state s_1^4, resources R_1 and R_2 have been released, and therefore, process instances j_3 and j_4 can also advance to completion.

Using the symmetries of the considered RAS, it is also easy to see that state $s_2 = 1 \cdot \Xi_{11} + 1 \cdot \Xi_{12} + 1 \cdot \Xi_{21} + 1 \cdot \Xi_{22} + 2 \cdot \Xi_{41}$ belongs in S_{rs}. On the other hand, state $s_3 = 1 \cdot \Xi_{11} + 1 \cdot \Xi_{12} + 1 \cdot \Xi_{21} + 1 \cdot \Xi_{22} + 1 \cdot \Xi_{31} + 1 \cdot \Xi_{41}$ is in $S_{r\bar{s}}$. This result can be verified, for instance, through an exhaustive search over the set of event sequences that originate from state s_3 and they do not involve the initiation of any new process instances.[2] But from the standpoint of the standard vector-based state representation, $s_3 = \frac{1}{2}s_1 + \frac{1}{2}s_2$. Hence, state s_3 belongs to the convex hull of the states in S_{rs}, and it is not linearly separable from them [10], [11].

III. DISCUSSION

The counter-example of Section II establishes the fallacy of the results that are claimed in [1]. On the other hand, the discussion that is provided in this section has as a first objective to provide a brief explanation of the erroneous logic in the developments of [1] that has led to the aforementioned problem. But the following arguments will also reveal an entire sub-class of SU-RAS that is guaranteed, indeed, to admit linear separability of the corresponding sets S_{rs} and $S_{r\bar{s}}$, and thus, a representation of the corresponding maximally permissive DAP by a linear classifier.

Instrumental for the following arguments is the realization that the set of unsafe states in sequential RAS, including the SU-

[2]A more intuitive explanation for the unsafety of state s_3 is provided by the observation that, in this state, there are three process instances moving in each direction that is defined by the natural ordering of the system resources, while there are only two resources of double capacity. Hence, it is not possible to construct the clearing "passes" that were utilized in the case of states s_1 and s_2.

RAS class, can be separated into deadlock states and deadlock-free unsafe states. This last class of states are states that do not contain any already deadlocked processes but lead unavoidably to deadlock states; state s_3 in the example of Section II is such a deadlock-free unsafe state. In [2] it is shown that, in the considered RAS class, the recognition of a deadlock state is a task of polynomial complexity with respect to the size of the underlying RAS. On the other hand, the recognition of deadlock-free unsafe states can be a much more complex task, to the point that the safety assessment of any given RAS state is eventually an NP-complete problem [2], [12]. Some additional results that are of relevance to this discussion are as follows:

Result 1: The single-unit allocation that is associated with each processing stage of an SU-RAS implies that any deadlock state of this RAS class must have all the involved resources allocated to capacity to the deadlocked processes. Hence, to prevent the formation of such a deadlock in a maximally permissive manner, it suffices to request that the total number of the process instances executing simultaneously the processing stages Ξ_{jk} that are involved in the deadlock, must remain (strictly) below the total capacity of the corresponding resource types that are associated with these stages through the allocation function $\mathcal{A}(\Xi_{jk})$. Bu this last requirement can be expressed straightforwardly by a linear inequality.

Result 2: For any given deadlock-free unsafe state s, a control policy that prevents the accessibility of its successor (unsafe) states in a maximally permissive manner, will expose the unsafety of s by rendering it a (policy-)induced deadlock.

In the analysis of [1], the aforestated capability to prevent the formation of a potential deadlock in any SU-RAS, in a maximally permissive manner, by a single linear inequality, is the essence of Theorem 5.1. From a more technical standpoint, the proof of Theorem 5.1 in [1] establishes the aforementioned result by leveraging an established connection of deadlock formation in SU-RAS and the PN structural object of empty siphon [13].[3] On the other hand, Theorem 5.2 tries to leverage the combination of Results 1 and 2, in order to argue that Result 1 essentially extends to the maximally permissive control of all the unsafe states of an SU-RAS. But the authors failed to realize that the augmentation of the original nets with the necessary "monitor" places that enforce any linear inequalities on the underlying RAS state, leads to PN structures that do not belong to the class of the SU-RAS-modeling PNs (i.e., these new nets are not S^3PR). Hence, the scheme of incremental control synthesis that is implied by the proof of Theorem 5.2 in [1] will not work, since Result 1 will not apply to the sequence of the "monitor"-augmented nets that will be generated by this process.

In fact, these augmented nets can be shown to belong in the broader class of S^3PGR^2 nets of [14]. As it is also remarked in [1], S^3PGR^2 nets model RAS classes with more arbitrary resource requests for each processing stage than the requests allowed by SU-RAS, and for these nets, deadlock can be interpreted by siphons that are insufficiently marked to enable the firing of their output transitions, but not necessarily empty.[4] A more intuitive interpretation of this last result is that, due to the presence of requests for more than one resource units by a single process instance, a process can be in deadlock even though some of the requested resources have free capacity (but not sufficient to meet the process needs). Unfortunately, the formation of this new type of partial deadlock is not necessarily controllable, in a maximally permissive manner, by a set of linear inequalities on the RAS state.

While explaining the source of the error in the developments of [1], the above remarks also imply that for SU-RAS with no deadlock-free unsafe states, the corresponding sets S_{rs} and $S_{r\bar{s}}$ are indeed linearly separable (due to Result 1 above and/or the corresponding arguments of Theorem 5.1 in [1]).[5] An extensive discussion on the sub-class of SU-RAS with no deadlock-free unsafe states, that also provides structural characterizations for the elements of this class, can be found in Chapter 3 of [2].

IV. Conclusion

This technical note has demonstrated by means of a counter-example that a result recently claimed in [1] is not true. Furthermore, the accompanying discussion of Section III has identified the misconceptions in the developments of [1] that led to the erroneous claim, and it has also revealed some further conditions that, when imposed on the considered RAS class (essentially defining a more restricted RAS class), will render the result that is pursued in [1] actually true.

References

[1] G. Liu, D. Y. Chao, and M. Uzam, "Maximally permissive deadlock prevention via an invariant controlled method," *IJPR*, vol. 51, pp. 4431–4442, 2013.

[2] S. A. Reveliotis, *Real-time Management of Resource Allocation Systems: A Discrete Event Systems Approach*. NY, NY: Springer, 2005.

[3] M. Zhou and M. P. Fanti (editors), *Deadlock Resolution in Computer-Integrated Systems*. Singapore: Marcel Dekker, Inc., 2004.

[4] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems (2nd ed.)*. NY,NY: Springer, 2008.

[5] A. Nazeem, S. Reveliotis, Y. Wang, and S. Lafortune, "Designing maximally permissive deadlock avoidance policies for sequential resource allocation systems through classification theory: the linear case," *IEEE Trans. on Automatic Control*, vol. 56, pp. 1818–1833, 2011.

[6] A. Nazeem and S. Reveliotis, "Designing maximally permissive deadlock avoidance policies for sequential resource allocation systems through classification theory: the non-linear case," *IEEE Trans. on Automatic Control*, vol. 57, pp. 1670–1684, 2012.

[7] Y. F. Chen and Z. W. Li, "Design of a maximally permissive liveness-enforcing supervisor with a compressed supervisory structure for flexible manufacturing systems," *Automatica*, vol. 47, pp. 1028–1034, 2011.

[8] A. Giua, F. DiCesare, and M. Silva, "Generalized mutual exclusion constraints on nets with uncontrollable transitions," in *Proceedings of the 1992 IEEE Intl. Conference on Systems, Man and Cybernetics*. IEEE, 1992, pp. 974–979.

[9] K. Yamalidou, J. Moody, M. D. Lemmon, and P. J. Antsaklis, "Feedback control of petri nets based on place invariants," *Automatica*, vol. 32, pp. 15–28, 1996.

[3]We remind the reader that, in PN theory, a siphon S is a set of places such that the set of transitions that deposit tokens in some place(s) of S (known as the "input" transitions of S) is a subset of the set of transitions that need some tokens from some place(s) in S for their firing (known as the "output" transitions of S). Hence, once a siphon gets empty of tokens, none of its output transitions can fire anymore.

[4]Such siphons are characterized as "deadly marked" in the literature [14].

[5]Once the possibility of linear separation of the sets S_{rs} and $S_{r\bar{s}}$ has been established through the above argument, more parsimonious linear classifiers effecting this separation can be developed through the techniques presented in [5], [11]. On the other hand, the practical significance of the linear separability of the S_{rs} and $S_{r\bar{s}}$ sets in the context of this SU-RAS subclass is mitigated by the fact that, for this RAS class, the maximally permissive DAP is efficiently (i.e., polynomially) implementable through one-step-lookahead just for deadlock (and not for (un-)safety, as is the case for broader RAS classes that contain deadlock-free unsafe states)[15].

[10] S. Reveliotis and A. Nazeem, “Optimal linear separation of the safe and unsafe subspaces of sequential resource allocation systems as a set-covering problem: algorithmic procedures and geometric insights,” *SIAM Journal on Control and Optimization*, vol. 51, pp. 1707–1726, 2013.

[11] R. Cordone, A. Nazeem, L. Piroddi, and S. Reveliotis, “Designing optimal deadlock avoidance policies for sequential resource allocation systems through classification theory: existence results and customized algorithms,” *IEEE Trans. on Automatic Control*, vol. 58, pp. 2772–2787, 2013.

[12] S. Reveliotis and E. Roszkowska, “On the complexity of maximally permissive deadlock avoidance in multi-vehicle traffic systems,” *IEEE Trans. on Automatic Control*, vol. 55, pp. 1646–1651, 2010.

[13] J. Ezpeleta, J. M. Colom, and J. Martinez, “A Petri net based deadlock prevention policy for flexible manufacturing systems,” *IEEE Trans. on R&A*, vol. 11, pp. 173–184, 1995.

[14] J. Park and S. A. Reveliotis, “Deadlock avoidance in sequential resource allocation systems with multiple resource acquisitions and flexible routings,” *IEEE Trans. on Automatic Control*, vol. 46, pp. 1572–1583, 2001.

[15] S. A. Reveliotis, M. A. Lawley, and P. M. Ferreira, “Polynomial complexity deadlock avoidance policies for sequential resource allocation systems,” *IEEE Trans. on Automatic Control*, vol. 42, pp. 1344–1357, 1997.