2018 IEEE Conference on Decision and Control (CDC)
Miami Beach, FL, USA, Dec. 17-19, 2018

Maximal linear deadlock avoidance policies for complex resource allocation systems

Michael Ibrahim and Spyros Reveliotis
School of Industrial & Systems Engineering
Georgia Institute of Technology
{mibrahim37@,spyros@isye}.gatech.edu

Abstract— **The problem of maximally permissive deadlock avoidance for complex resource allocation systems (RAS) is a well-defined problem in the corresponding controls literature. In some of the prevailing approaches to this problem, the sought supervisor – also known as the maximally permissive deadlock avoidance policy (DAP) – is perceived as a classifier, and its design boils down to the development of an efficient representation of the classification logic that it effects on the underlying RAS states. A popular such representation is the "linear classifier", where the admissibility of any given RAS state is resolved based on its ability to satisfy a given set of linear inequalities. However, linear classifiers cannot provide effective representation of the maximally permissive DAP for all RAS instantiations. Hence, this paper provides a methodology for synthesizing linear DAPs for any given RAS instance that might not be maximally permissive in the original sense of this term, but observe a more relaxed notion of "maximality". The presented developments formally define this new DAP class, and provide effective computational algorithms for the synthesis of a maximal linear DAP for any given RAS instance.**

I. Introduction

Complex resource allocation systems (RAS) [1] is a class of discrete event systems (DES) [2] that has received extensive attention in the corresponding literature. The supervisory control problem of deadlock avoidance that underlies the operation of these systems, seeks to coordinate the sequential allocation of a finite set of reusable resources to a set of concurrently executing processes so that all these processes are able to receive the requested resources with finite delays, and eventually complete and exit the system [1]. Furthermore, there is an additional request for *maximal permissiveness* for the corresponding supervisory control policies; i.e., these policies should ensure the aforementioned capability of all activated processes to run successfully to their completion, while imposing the minimum possible restriction to the original behavior that is generated by the uncontrolled system. Such a maximally permissive supervisor – also, known as a maximally permissive deadlock avoidance policy (DAP) – is well-defined and unique for the RAS instantiations studied in [1]. But it is also true that computing the maximally permissive DAP is an NP-hard problem for almost all RAS classes of interest [3].

Nevertheless, recognizing that the sought DAPs act as classifiers that dichotomize the underlying RAS state space into admissible and inadmissible subspaces, the corresponding research community has developed methodology that enables the off-line synthesis of representations for these policies that are very parsimonious, and therefore amenable for real-time control [1]. Among these DAP representations, one of the most interesting and tractable, in terms of, both, analysis and implementation, is that of a "linear" classifier [4], [5], [6]. In this case, the policy admissibility of any given state is resolved based on the ability of this state to satisfy a given set of linear inequalities. In the following, we shall refer to DAPs that admit such a linear representation of their state-acceptance logic as "linear" DAPs.

But as established in [7], [8], linear representation of the maximally permissive DAP is not a viable option for all RAS instantiations of practical interest. To circumvent this limitation, the works of [9], [10], [11], [12] have proposed additional representations for the sought classifiers that either employ nonlinear discriminant functions of the RAS state, or they constitute "non-parametric" classification schemes that rely on the efficient storage and processing of explicit information about the structure of the underlying state space. These alternative representations have been shown to be complete, i.e., they will always provide an effective representation of the target DAP.

Yet, in spite of the aforementioned developments, in many application contexts, DAPs that admit linear representation are still a most desirable solution, due to the analyzability of these policies, and their easy integrability into broader decision-making frameworks. And, in fact, the literature avails of methodology that can synthesize correct linear (but not necessarily maximally permissive) DAPs for a large spectrum of RAS classes of practical interest. Some characteristic examples of this methodology can be found in [13], [14], [15], [16], [17], [18], while a more comprehensive treatment of these methods is provided in Chapter 6 of [1]. But the existing theory does not allow for an explicit characterization and/or control of the extent of the sub-optimality of the DAPs that are derived by it with respect to the maximally permissive DAP.

Motivated by the last remark in the previous paragraph, in this work we seek to develop a method for the systematic development of linear DAPs that are appropriate for the major RAS classes defined in [1], and observe a "maximality" requirement in terms of their permissiveness (even though they might be less permissive than the maximally permissive DAP). We proceed to these developments by providing a complete characterization of the target RAS class, and of the notion of "maximality" that is observed by our policy design. We also detail all the necessary algorithms for the computation of a maximal linear DAP for any given instance from the considered RAS class. These algorithms build upon and extend the corresponding algorithms that are provided in [4], [6] for the computation of a linear representation of the maximally permissive DAP, whenever this last DAP admits a linear representation. Furthermore, they will return the maximally permissive DAP itself, whenever this policy admits a linear representation. Finally, since these new algoritrhms utilize techniques similar to those used in [4], [6] as their building blocks, they inherit the applicability potential that has already been established in the literature for those previous methods.

In view of the above positioning of the paper content and its intended contribution, the rest of it is organized as follows: The next section provides a formal characterization of the RAS class of interest in this work, and of the corresponding supervisory control problem of deadlock avoidance. This section also overviews the existing results on the computation of a parsimonious linear representation of the maximally permissive DAP, and the conditions for the existence of such a linear representation. Subsequently, Section III introduces the new class of the maximal linear DAPs, motivating the rationale for the definition of these policies and establishing their well-posedness for any given instance from the considered RAS class. On the other hand, Section IV addresses the more practical issue of computing the maximal linear DAPs for any given RAS instance. Finally, Section V concludes the paper and provides some directions for potential future work.

II. The considered RAS class and the corresponding problem of deadlock avoidance

Disjunctive-Conjunctive RAS: The main ideas that define the methodology to be presented in this work, are applicable to the entire spectrum of RAS classes that are defined in [1]. But for better clarity and specificity, in the rest of this paper we focus primarily on the class of Disjunctive-Conjunctive (D/C-) RAS. This is a pretty broad RAS class that allows for (i) an arbitrary structure of the resource requests that are posed by the different processing stages, and also for (ii) the presence of routing flexibility in the supported process plans. A formal definition of the D/C-RAS class is as follows:

Definition 1: A *Disjunctive-Conjunctive (D/C-) Resource Allocation System (RAS)* is a 4-tuple $\Phi = \langle \mathcal{R}, C, \mathcal{P}, D \rangle$, where:

1) $\mathcal{R} = \{R_1, \ldots, R_m\}$ is the set of the system *resource types*.
2) $C : \mathcal{R} \rightarrow \mathbb{Z}^+$ – the set of strictly positive integers – is the system *capacity* function, characterizing the number of identical units from each resource type available in the system. Resources are assumed to be *reusable*, i.e., each allocation cycle does not affect their functional status or subsequent availability, and therefore, $C(R_i) \equiv C_i$ constitutes a system *invariant* for each i.
3) $\mathcal{P} = \{\Pi_1, \ldots, \Pi_n\}$ denotes the set of the system *process types* supported by the considered system configuration. Each process type Π_j is a composite element itself, in particular, $\Pi_j =< \Theta_j, \mathcal{G}_j >$, where: (a) $\Theta_j = \{\theta_{j1}, \ldots, \theta_{j,l_j}\}$ denotes the set of *processing stages* involved in the definition of process type Π_j, and (b) $\mathcal{G}_j$ is an *acyclic digraph* with its node set, Q_j, being bijectively related to the set Θ_j. Denoting by $Q_j^{\nearrow}$ (resp., $Q_j^{\searrow}$) the set of *source* (resp., *sink*) nodes of $\mathcal{G}_j$, the available *process plans* for process type Π_j are represented by the *paths* leading from some node $q_s \in Q_j^{\nearrow}$ to some node $q_f \in Q_j^{\searrow}$ in digraph $\mathcal{G}_j$. Also, in the following, we shall set $\Theta \equiv \bigcup_{j=1}^{n} \Theta_j$ and $\xi \equiv |\Theta|$.
4) $D : \Theta \rightarrow \prod_{i=1}^{m}\{0, \ldots, C_i\}$ is the *resource allocation function* associating every processing stage θ_{jk} with the *resource allocation vector* $D(\theta_{ij})$ required for its execution; it is further assumed that $D(\theta_{ij}) \neq \mathbf{0}$, $\forall, i, j$. At any point in time, the system contains a certain number of (possibly zero) instances of each process type that execute one of the corresponding processing stages. A process instance executing a non-terminal stage $\theta_{ij} \in Q_i \backslash Q_i^{\searrow}$, must first be allocated the resource differential $(D(\theta_{i,j+1}) - D(\theta_{ij}))^+$ in order to advance to (some of) its next stage(s) $\theta_{i,j+1}$, and only then will it release the resource units $|(D(\theta_{i,j+1}) - D(\theta_{ij}))^-|$, that are not needed anymore. The considered resource allocation protocol further requires that no resource type $R_i \in \mathcal{R}$ be over-allocated with respect to its capacity C_i at any point in time.

Finally, for purposes of complexity considerations, we define the *size* $|\Phi|$ of RAS Φ by $|\Phi| \equiv |\mathcal{R}| + \xi + \sum_{i=1}^{m} C_i$.

Modeling the D/C-RAS dynamics as a Finite State Automaton: The dynamics of the RAS $\Phi = \langle \mathcal{R}, C, \mathcal{P}, D \rangle$ that was described in the previous paragraph, can be further formalized by a *Deterministic Finite State Automaton (DFSA)* $G(\Phi) = \langle S, E, f, \mathbf{s}_0, S_M \rangle$, that is defined as follows:

1) The *state set* S consists of ξ-dimensional vectors $\mathbf{s}$. The components $\mathbf{s}[l]$, $l = 1, \ldots, \xi$, of $\mathbf{s}$ are in one-to-one correspondence with the RAS processing stages, and they indicate the number of process instances executing the corresponding stage in the considered RAS state. Hence, S consists of all the vectors $\mathbf{s} \in (\mathbb{Z}_0^+)^{\xi}$ that further satisfy

$$\forall i = 1, \ldots, m, \quad \sum_{l=1}^{\xi} \mathbf{s}[l] \cdot D(\theta_l)[i] \leq C_i \qquad (1)$$

where, according to the adopted notation, $D(\theta_l)[i]$ denotes the allocation request for resource R_i that is posed by stage θ_l.[1]

2) The *event set* E is the union of the disjoint event sets $E^{\nearrow}$, $\bar{E}$ and $E^{\searrow}$, where:
   a) $E^{\nearrow} = \{e_{rp} : r = 0, \ \theta_p \in \bigcup_{j=1}^{n} Q_j^{\nearrow}\}$, i.e., event e_{rp} represents the *loading* of a new process instance that starts from stage θ_p.
   b) $\bar{E} = \{e_{rp} : \exists j \in 1, \ldots, n$ s.t. θ_p is a successor of θ_r in graph $\mathcal{G}_j\}$, i.e., e_{rp} represents the *advancement* of a process instance executing stage θ_r to a successor stage

[1]Following standard practice in DES literature (cf., for instance, the relevant definition in page 8 of [2]), in the rest of this document we will frequently use the terms "space" and "subspace" in order to refer to the state set S and its various subsets considered in this work.

θ_p.

c) $E^{\searrow} = \{e_{rp} : \theta_r \in \bigcup_{j=1}^{n} Q_j^{\searrow},\ p = 0\}$, i.e, e_{rp} represents the *unloading* of a finished process instance after executing its last stage θ_r.

3) The *state transition function* $f : S \times E \rightarrow S$ is defined by $\mathbf{s}' = f(\mathbf{s}, e_{rp})$, where the components $\mathbf{s}'[l]$ of the resulting state $\mathbf{s}'$ are given by:

$$\mathbf{s}'[l] = \begin{cases} \mathbf{s}[l] - 1 & \text{if} \quad l = r \\ \mathbf{s}[l] + 1 & \text{if} \quad l = p \\ \mathbf{s}[l] & \text{otherwise} \end{cases}$$

We also notice that $f(\mathbf{s}, e_{rp})$ is a *partial* function, defined only if the resulting state $\mathbf{s}' \in S$. For any state $\mathbf{s} \in S$, the event set $\Gamma(\mathbf{s}) \subseteq E$ for which $f(\mathbf{s}, e)$ is defined, constitutes the set of *feasible events* at $\mathbf{s}$.

4) The *initial state* $\mathbf{s}_0 = \mathbf{0}$, i.e., the state vector with all its components equal to zero. This initial state represents the situation where the system is empty of any process instances.

5) The *set of marked states* S_M is the singleton $\{\mathbf{s}_0\}$. This specification of S_M expresses the requirement for complete process runs.

Letting $\hat{f}$ denote the natural extension of the state transition function f to $S \times E^*$, the behavior of RAS Φ is modeled by the *language $L(G)$ generated* by DFSA $G(\Phi)$, i.e., by all strings $\sigma \in E^*$ such that $\hat{f}(\mathbf{s}_0, \sigma)$ is defined. Furthermore, we define the *reachable subspace* S_r of $G(\Phi)$ by

$$S_r \equiv \{\mathbf{s} \in S : \exists \sigma \in L(G) \text{ s.t. } \hat{f}(\mathbf{s}_0, \sigma) = \mathbf{s}\} \qquad (2)$$

and its *safe subspace* S_s by

$$S_s \equiv \{\mathbf{s} \in S : \exists \sigma \in E^* \text{ s.t. } \hat{f}(\mathbf{s}, \sigma) = \mathbf{s}_0\} \qquad (3)$$

Also, in the following, we shall denote the complements of S_r and S_s with respect to S by $S_{\bar{r}}$ and $S_{\bar{s}}$, and we shall refer to them as the *unreachable* and *unsafe* subspaces. Finally, S_{xy}, $x \in \{r, \bar{r}\}$, $y \in \{s, \bar{s}\}$, will denote the intersection of the corresponding sets S_x and S_y.

The target behavior of $G(\Phi)$ and the maximally permissive DAP: The desired – or "target" – behavior of RAS Φ is expressed by the *marked language $L_m(G)$*, which is defined by means of the set of marked states S_M, as follows:

$$\begin{aligned} L_m(G) &\equiv \{\sigma \in L(G) : \hat{f}(\mathbf{s}_0, \sigma) \in S_M\} \\ &= \{\sigma \in L(G) : \hat{f}(\mathbf{s}_0, \sigma) = \mathbf{s}_0\} \end{aligned} \qquad (4)$$

Equation 4, when combined with all the previous definitions, further implies that the set of states that are accessible under $L_m(G)$ is exactly equal to S_{rs}. Hence, we have the following definition of the *maximally permissive deadlock avoidance policy (DAP)* Δ^* for the considered RAS:

Definition 2: The *maximally permissive deadlock avoidance policy (DAP)* Δ^* for any instantiation Φ from the RAS class of Definition 1 is a supervisory control policy that, at every state $\mathbf{s} \in S_{rs}$, admits a feasible transition $\mathbf{s}' = f(\mathbf{s}, e_{rp})$ of the underlying DFSA $G(\Phi)$ if and only if $\mathbf{s}' \in S_s$. □

The reader should also notice that the above characterization of the policy Δ^* further implies that, for any given RAS instance Φ, this policy is unique.

The maximally permissive DAP as a classifier: According to Definition 2, the maximally permissive DAP Δ^* can be effectively implemented through any mechanism that recognizes and rejects the unsafe states that are accessible through one-step transitions from S_{rs}. In the following, we shall refer to these particular unsafe states as "boundary" unsafe states, and we shall perceive the policy Δ^* as a classifier that distinguishes effectively between reachable safe states and boundary unsafe states.

As discussed in the introductory section, methodology for the effective development of such a classifier is provided in [4], [9], [10], [11], [12], [8], [6]. A result that has proven very useful in the development of the corresponding theory, is the following "monotonicity" property that is exhibited by the RAS state safety:

Proposition 1: Consider the partial order "$\leq$" that is defined on the state space S of any given RAS Φ through the following comparison of the state components:

$$\forall \mathbf{s}, \mathbf{s}' \in S, \;\; \mathbf{s} \leq \mathbf{s}' \iff (\forall l = 1, \ldots \xi, \;\; \mathbf{s}[l] \leq \mathbf{s}'[l]) \qquad (5)$$

Then,

1) $\mathbf{s} \in S_s \;\wedge\; \mathbf{s}' \leq \mathbf{s} \implies \mathbf{s}' \in S_s$
2) $\mathbf{s} \in S_{\bar{s}} \;\wedge\; \mathbf{s} \leq \mathbf{s}' \implies \mathbf{s}' \in S_{\bar{s}}$

□

In [4] it is shown that, thanks to Proposition 1, it is possible to develop a classifier that will distinguish correctly between (a) the states of the reachable and safe subspace S_{rs}, and (b) the boundary unsafe states, by focusing only on the correct classification of the maximal elements of the set S_{rs} and the minimal boundary unsafe states. Furthermore, additional efficiencies in this endeavor, and in the on-line computational complexity of the developed classifier, can be obtained by identifying and removing from the classified vectors any components corresponding to processing stages that do not impact the safety of the system state (e.g., like the terminal processing stages of any process type Π_j). The reader is referred to Chapter 4 of [1] for a concise and comprehensive exposition of the corresponding theory on the effective and efficient synthesis of the sought classifiers.

Linear representation of the maximally permissive policy Δ^*: As remarked in the introductory section, a desirable representation of the classification logic that is effected by the maximally permissive DAP Δ^* is that of a linear classifier. This last concept has been formally defined in [4] as follows:

Definition 3: Consider two vector sets G and H from a ξ-dimensional vector space V.

1) We shall say that sets G and H are *linearly separated* by a set of k linear inequalities $\{(\mathbf{a}_i, b_i) : i = 1, \cdots, k\}$ if and only if (*iff*)

$$\begin{aligned} &(\forall \mathbf{g} \in G : \; \forall i \in \{1, \cdots, k\}, \; \mathbf{a}_i^T \cdot \mathbf{g} \leq b_i) \quad \wedge \\ &\quad (\forall \mathbf{h} \in H : \exists i \in \{1, \cdots, k\}, \; \mathbf{a}_i^T \cdot \mathbf{h} > b_i) \end{aligned} \qquad (6)$$

2) A linear classifier – or separator – for vector sets G and H is *structurally minimal*, *iff* it employs the minimum

possible number of linear inequalities that can separate these two sets.

□

In the case of the classification that is effected by the DAP Δ^*, the roles of the sets G and H in Definition 3 are played, respectively, by the sets $\bar{S}_{rs}$ and $\bar{S}^b_{r\bar{s}}$ that contain the maximal reachable safe states and the minimal boundary unsafe states. In this case, Proposition 1 implies the following additional result for the sought classifiers [4]:

Proposition 2: If the maximally permissive DAP Δ^* of a given D/C-RAS Φ admits a representation as a linear classifier of Definition 3, then, there exists such a linear classifier with *nonnegative* parameters $(\mathbf{a}_i, b_i)$ for all the involved inequalities. □

The astute reader will also notice that Definition 3 implies an asymmetry for the role of the sets $\bar{S}_{rs}$ and $\bar{S}^b_{r\bar{s}}$ in the design of the sought (linear) classifier. This asymmetry is dictated by the further implementation of the resulting classifier through some popular modeling frameworks for the (controlled) RAS dynamics, and especially, the modeling framework of Petri nets (PNs) [19]. In the PN modeling framework, each of the inequalities implementing a linear classifier that (a) presents the structure described in Definition 3, and (b) satisfies the additional "non-negativity" condition of Proposition 2, can be enforced on the RAS-modeling PN through the addition of a single place that is known as the corresponding "monitor" place [20], [21]. More importantly, the resulting PN, that represents the behavior of the controlled RAS, belongs to the same class with the PNs that model the uncontrolled RAS behavior, and therefore, it is amenable to the same analysis and design methods that are available for the "plant" (i.e., the RAS-modeling) PN.

On the other hand, it is also well known that Δ^* might not admit a linear representation along the lines of Definition 3 [7], [8].[2] Such a case is provided in Figure 1, where it can be seen that the lack of a linear representation for the corresponding DAP Δ^* is due to the inclusion of elements of the set $\bar{S}^b_{r\bar{s}}$ in the convex hull of S_{rs}. As remarked in the introductory section, this problem has been addressed through the development of additional representations for the classification logic that is effected by the target policy Δ^*. However, these representations are not amenable to the PN-based implementation of Δ^* that was discussed in the previous paragraph, and to the various analytical and computational possibilities and efficiencies that result from such an implementation. One such possibility that is of particular interest in an ongoing research program of ours is the inclusion of the logic of the employed DAP into some linear programming formulations that seek to complement the preventive control of deadlock avoidance with scheduling capability, and are known as "fluid relaxations" of the underlying RAS dynamics.[3]

[2]Also, some interesting related work concerning the limitations of the aforementioned structure of "monitor" places to provide effective representation of the maximally permissive supervisor that ensures deadlock-free and/or live operation for various PN classes, can be found in [22], [23].

[3]Some representative works that employ LP-based "fluid relaxations" as instruments for computing efficient scheduling policies for complex resource allocation, can be found in [24], [25], [26], [27].

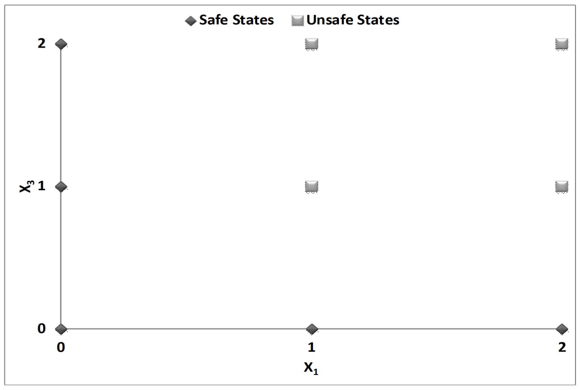

Fig. 1: Characterization of the safe and unsafe reachable states for an example D/C-RAS with two resource types, R_1 and R_2, with corresponding capacities $C(R_1) = C(R_2) = 2$, and two process types, Π_1 and Π_2, with corresponding process plans $R_1 \rightarrow 2.R_2$ and $R_2 \rightarrow 2.R_1$. Recognizing that the terminal processing stages of these two process types will never get involved in a deadlock, the information that is provided by this figure is projected on the sub-space that is defined by the state components $\mathbf{s}_1$ and $\mathbf{s}_3$, which correspond to the first processing stage of each process plan. Safe reachable states are depicted by rhombi and unsafe reachable states by squares. The reader should notice that the convex hull of the depicted safe states includes the unsafe state corresponding to point $(1, 1)$, and therefore, in this case, the reachable safe states and the boundary unsafe states of the considered system are not linearly separable.

Motivated by the above remarks and needs, in the rest of this work, we define an approximation to the maximally permissive DAP Δ^* that (i) admits a linear representation along the lines of Definition 3 and Proposition 2, and (ii) is effectively computable for every instance Φ of the considered class of D/C-RAS. Furthermore, the formal definition of these policies employs a notion of "maximality" that intends to keep their permissiveness as close as possible to the permissiveness of Δ^*. We provide a formal characterization of this "maximality" concept, and the necessary algorithms for the effective computation of the corresponding policies, for any given D/C-RAS Φ.

III. Maximal linear DAPs

The formal definition of the maximal linear DAPs and its motivation: This section introduces the new concept of the "maximal linear DAP", as it materializes in the considered class of D/C-RAS. We shall formally define this new DAP class by providing a complete set of conditions that must be satisfied by the admissible subspaces of its constituent policies. Hence, let Δ denote a tentative DAP from the considered class for some given D/C-RAS Φ, and let $S_a(\Delta) \subseteq S$ denote the corresponding policy-admissible subspace. We also define $S_{\bar{a}}(\Delta) \equiv S \setminus S_a(\Delta)$. For the dynamics of the controlled system to be well-defined, clearly we need

$$\mathbf{s}_0 \in S_a(\Delta) \tag{7}$$

Then, we can also define $S_r(\Delta)$, the reachable subspace of Φ under policy Δ, as the limit set of the following recursion:

$$S_r(\Delta)^{(0)} := \{\mathbf{s}_0\} \tag{8}$$

$$S_r(\Delta)^{(k+1)} := S_r(\Delta)^{(k)} \;\cup\; \{\mathbf{s}' \in S_a(\Delta) : \exists \mathbf{s} \in S_r(\Delta)^{(k)}, e \in \Gamma(\mathbf{s}) \text{ with } f(\mathbf{s}, e) = \mathbf{s}'\} \tag{9}$$

A primary requirement in the specification of the sought policy Δ is that it does not induce any new deadlocks or livelocks; such a DAP is characterized as "correct" in the relevant literature [1]. The correctness of Δ translates into the following requirement for the corresponding set $S_r(\Delta)$:

$$\forall \mathbf{s} \in S_r(\Delta),\ \exists e \in \Gamma(\mathbf{s}) \setminus E^{\nearrow},\ f(\mathbf{s}, e) \in S_r(\Delta) \tag{10}$$

In more natural terms, the condition of Equation 10 requires that at every state $\mathbf{s}$ that is reachable in the considered RAS under supervision by Δ, there is a policy-admissible event e that concerns the stage advancement or the unloading of an already initiated process instance. In Chapter 6 of [1] it is shown that this condition further implies that the subgraph $\mathcal{G}_r(\Delta)$ of the state transition diagram (STD) of the FSA $G(\Phi)$ that is induced by the state set $S_r(\Delta)$, contains the initial state $\mathbf{s}_0$ and it is strongly connected. Hence, state $\mathbf{s}_0$ is reachable from every state $\mathbf{s} \in S_r(\Delta)$, and therefore, there will be no deadlocks or livelocks in the operation of the controlled RAS.

Next we address the requirement that the sought policy Δ will admit a representation through the linear classifiers of Definition 3. Furthermore, for the reasons that were explained in the previous section, we also want the linear representations for our target policies Δ to satisfy the "non-negativity" property of Proposition 2.

To formally state the conditions that will help us meet these two requirements, let us denote the convex hull of any given vector set V by $conv(V)$, and also define the set $S^b_{\bar{r}}(\Delta)$ as follows:

$$S^b_{\bar{r}}(\Delta) \equiv \{\mathbf{s}' \in S_r \setminus S_a(\Delta) : \exists \mathbf{s} \in S_r(\Delta),\ e \in \Gamma(\mathbf{s}) \text{ with } f(\mathbf{s}, e) = \mathbf{s}'\} \tag{11}$$

The set $S^b_{\bar{r}}(\Delta)$ contains all the states $\mathbf{s}$ that are reachable through a single transition from $S_r(\Delta)$ but are blocked by policy Δ. Hence, this set collects all the "boundary inadmissible" states in the controlled dynamics of RAS Φ.

Then, in analogy to the corresponding results for the maximally permissive DAP Δ^*, the aforestated requirement for a representation of the policy Δ through a linear classifier of Definition 3 with non-negative coefficients can be met by introducing the following two conditions to the policy specification:

$$\forall \mathbf{s}, \mathbf{s}' \in S_a(\Delta),\ \mathbf{s}' \le \mathbf{s}\ \wedge \mathbf{s} \in S_a(\Delta) \Longrightarrow \mathbf{s}' \in S_a(\Delta) \tag{12}$$

$$conv(S_r(\Delta)) \cap S^b_{\bar{r}}(\Delta) = \emptyset \tag{13}$$

Up to this point, we have articulated the requirements that must be satisfied by the sought DAP Δ for any given D/C-RAS Φ so that (i) it is correct, and (ii) admits a desired linear representation, as qualified by Definition 3 and the condition of Proposition 2. Policy Δ will also be a *"maximal"* (correct) linear DAP for RAS Φ, if there is no other correct linear DAP Δ' for RAS Φ with an admissible reachable subspace $S_r(\Delta')$ such that $S_r(\Delta') \supset S_r(\Delta)$.

The following definition provides a more formal expression to all the previous discussion.

Definition 4: A policy Δ is a *linear* DAP for some given D/C-RAS Φ *iff* its admissible subspace $S_a(\Delta)$ satisfies the following conditions:

Correctness: $(\mathbf{s}_0 \in S_a(\Delta))\ \wedge$

$$\big(\forall \mathbf{s} \in S_r(\Delta),\ \exists e \in \Gamma(\mathbf{s}) \setminus E^{\nearrow},\ f(\mathbf{s}, e) \in S_r(\Delta)\big)$$

Monotonicity: $\forall \mathbf{s}, \mathbf{s}' \in S_a(\Delta)$,

$$\mathbf{s}' \le \mathbf{s}\ \wedge \mathbf{s} \in S_a(\Delta) \Longrightarrow \mathbf{s}' \in S_a(\Delta)$$

Linearity: $conv(S_r(\Delta)) \cap S^b_{\bar{r}}(\Delta) = \emptyset$

Furthermore, a linear DAP Δ for a given D/C-RAS Φ is *maximal iff* there is no other linear DAP Δ' for D/C-RAS Φ with $S_r(\Delta') \supset S_r(\Delta)$. $\square$

Example: Two maximal linear DAPs for the example D/C-RAS of Figure 1, are the DAPs Δ^1 and Δ^2 that will admit a state $\mathbf{s} \in S$ if its projection on the 2-dim space that is defined by the state components $\mathbf{s}_1$ and $\mathbf{s}_3$, belongs, respectively, in the sets $S^1_a \equiv \{(0,0), (1,0), (2,0), (0,1)\}$ and $S^2_a \equiv \{(0,0), (1,0), (0,1), (0,2)\}$.

Indeed, both of these policies admit the initial state $\mathbf{s}_0$ and it can be easily checked that they do not suffer from any policy-induced deadlock or livelock. Furthermore, they satisfy the "monotonicity" requirement of Definition 4, and the corresponding state sets $S_r(\Delta^i), S^b_{\bar{r}}(\Delta^i),\ i = 1, 2$, will admit linear separation in the projected space that is defined by the state coordinates $\mathbf{s}_1$ and $\mathbf{s}_3$. Finally, these two policies are also maximal, since the only possible expansion of the corresponding sets $S_r(\Delta^i),\ i = 1, 2$, is by re-admitting the blocked pairs (0,2) and (2,0) in the corresponding sets $S^i_a,\ i = 1, 2$; but the policy that will result from any of these two augmentations is Δ^*, and we know that this policy is not linear.

Finally, the reader should also notice that $S_r(\Delta^1) \neq S_r(\Delta^2)$, and therefore, the two policies Δ^1 and Δ^2 are essentially different.

Existence but non-uniqueness of maximal linear DAPs: The closing remark in the previous example further implies that, for any given D/C-RAS Φ, the maximal linear DAPs of Definition 4 will not be unique, in general. Hence, for further reference, we shall denote the set of linear DAPs for any given D/C-RAS Φ by $\mathcal{L}(\Phi)$, and its subset that contains its maximal elements by $\bar{\mathcal{L}}(\Phi)$.

The next result is also important for the well-posedness of the considered DAP class.

Proposition 3: For any given D/C-RAS Φ, $\bar{\mathcal{L}}(\Phi) \neq \emptyset$.

Proof: For any given D/C-RAS Φ, consider the policy $\hat{\Delta}$ that admits a state $\mathbf{s} \in S$ *iff* (a) either it is the initial state $\mathbf{s}_0$, or (b) it contains only one active process instance. Then, it is easy to see that the policy $\hat{\Delta}$ is correct, and satisfies the "monotonicity" requirement of Definition 4. It is also clear that the admissibility logic of this policy can be expressed by the linear inequality

$$\sum_{i=1}^{\xi} \mathbf{s}[i] \le 1$$

Algorithm 1 The main algortihm for computing $\bar{\mathcal{L}}(\Phi)$

```
Input: DFSA $G(\Phi)$
Output: $\bar{\mathcal{L}}(\Phi)$
    /* INITIALIZE */
 1: $STORE$ := NILL;  $EXPLORE := \langle \bar{S}_{rs} \rangle$;
    /* MAIN ITERATION */
 2: while $EXPLORE \neq NILL$ do
 3:   $\bar{S}_r$ := POP($EXPLORE$);  Compute $\bar{S}^b_{\bar{r}}$;
 4:   if (($\bar{S}_r, \bar{S}^b_{\bar{r}}$) linearly separable) AND
      ($\nexists\ \bar{S}'_r \in STORE : \bar{S}'_r \supseteq \bar{S}_r$) then
 5:     Remove from $STORE$ any element sets $\bar{S}''_r$ s.t.
        $\bar{S}''_r \subset \bar{S}_r$;
 6:     Enter $\bar{S}_r$ in $STORE$;
 7:   else
 8:     for all s $\in \bar{S}_r$ do
 9:       $\tilde{S}_r := PRUNE(\bar{S}_r,$ s, $G(\Phi))$;
10:       if ($\nexists\ \bar{S}'_r \in STORE :\ \bar{S}'_r \supseteq \tilde{S}_r$) then
11:         PUSH($\tilde{S}_r; EXPLORE$);
12:       end if
13:     end for
14:   end if
15: end while
    /* TERMINATE */
16: return $STORE$;
```

Algorithm 2 Function $PRUNE(\bar{S},\ \tilde{\text{s}},\ G(\Phi))$

```
Input: DFSA $G(\Phi)$, maximal-state set $\bar{S}$, pruned state $\tilde{\text{s}}$
Output: $PRUNE(S, G(\Phi))$
1: $\hat{S}_r := \{\text{s}_0\}$;
2: while $\hat{S} := \{\text{s} \in S \setminus (\hat{S}_r \cup \{\tilde{\text{s}}\}) : (\exists \text{s}' \in \hat{S}_r,\ e \in \Gamma(\text{s}')$ with $f(\text{s}', e) = \text{s})$ AND $(\exists \text{s}'' \in \bar{S}$ s.t. $\text{s} \leq \text{s}'')\} \neq \emptyset$ do
3:   $\hat{S}_r := \hat{S}_r \cup \hat{S}$;
4: end while
5: while $\hat{S} := \{\text{s} \in \hat{S}_r :\ \forall e \in \Gamma(s) \backslash E^{\nearrow},\ f(\text{s}, e) \notin \hat{S}_r\} \neq \emptyset$ do
6:   $\hat{S}_r := \hat{S}_r \setminus \hat{S}$;
7: end while
8: $\bar{S}_r := \{\text{s} \in \hat{S}_r : \nexists \text{s}'$ s.t. $\text{s}' > \text{s}\}$;
9: return $\bar{S}_r$;
```

Hence, the set of linear DAPs for any given D/C-RAS Φ, $\mathcal{L}(\Phi)$, is non-empty. Since this set is also finite, it will possess well-defined maximal elements, and therefore, the set $\bar{\mathcal{L}}(\Phi)$ is also non-empty. $\square$

With the notion of the maximal linear DAP well-defined, next we turn to the development of the necessary algorithms that will provide a maximal linear DAP for any given D/C-RAS Φ.

IV. Computing the maximal linear DAPs

In this section, we present a complete algorithm that will generate all the maximal linear DAPs Δ for any given D/C-RAS Φ. This algorithm essentially starts with the set of reachable and safe states, S_{rs}, that defines the reachable subspace under the maximally permissive DAP Δ^*, and seeks to detect all the maximal subsets of this set that will define correct linear DAPs. Each subset of S_{rs} that is considered by this process, is obtained from a "parent" subset by removing (i) a single maximal element of the "parent" set, and (ii) any additional states that need to be removed in order to restore the correctness of the induced DAP. The induced DAP itself is tested for membership in $\mathcal{L}(\Phi)$, through the corresponding algorithms that are available in [4], [6]. Finally, another salient point of the presented algorithm is that the aforementioned subsets of S_{rs} that are generated during the search process, are primarily represented by means of their maximal elements; in the presented pseudo-code, this fact is indicated by "barring" or "tilding" the corresponding sets.

The complete pseudo-code of the aforementioned algorithm is presented in Algorithm 1. The algorithm uses two lists, $STORE$ and $EXPLORE$, that hold, respectively, (a) the subsets of S_{rs} that correspond to linear DAPs and are maximal among the currently detected such sets, and (b) subsets of S_{rs} that have been generated as potential candidates for specifying maximal linear DAPs, but have not been assessed and further processed yet.

As it can be seen in Lines 3–14 of Algorithm 1, the processing of a set that has been extracted from the list $EXPLORE$, consists of the following steps: First this set is checked whether it defines a linear DAP.[4] If this is the case, and, furthermore, this set is not dominated by any set already in $STORE$, the set is entered in $STORE$ as the reachable subspace of a tentative maximal linear DAP. During this stage, $STORE$ is also cleared by any already stored sets that are dominated by the new entrance. If, on the other hand, the considered set does not specify a linear DAP, then it spawns a number of entries for the list $EXPLORE$. Each of these entries is generated through (i) the removal of a maximal element from the "parent" set, and (ii) the further pruning of the resulting set in order to ensure that it specifies a correct DAP. The function that performs this pruning is listed in Algorithm 2, and it constitutes a "fixed point" computation that seeks to establish the correctness condition of Definition 4. An additional element that is important for ensuring the correctness of the proposed algorithm, is that the list $EXPLORE$ is processed according to the First-In-First-Out (FIFO) policy; in this way, all sets that are stored in that list are processed after their "parent" sets.

The entire algorithm is initialized with list $STORE$ empty and list $EXPLORE$ containing the set S_{rs} (represented by its maximal elements). Hence, the algorithm will assess whether the maximally permissive DAP Δ^* is a linear DAP, and if this is the case, it will terminate without considering any other policies. In the opposite case, it will run as described in the previous paragraph, eventually returning the contents of the list $STORE$ as its output. The algorithm termination will take place when the list $EXPLORE$ becomes empty.

We also notice, for completeness, that the set dominance that is tested in certain parts of the algorithm, can be resolved

[4]As already mentioned, this test can be performed through the procedures that have been developed in [4], [6].

by means of the maximal elements that are stored in the employed representations of these sets, through the following criterion:

$$\bar{S}_r \supseteq \bar{S}'_r \iff \forall \mathbf{s}' \in \bar{S}'_r,\ \exists \mathbf{s} \in \bar{S}_r :\ \mathbf{s} \geq \mathbf{s}' \qquad (14)$$

Finally, we conclude this section with the following theorem that formally states the correctness of the presented algorithm.

Theorem 1: When applied on any given D/C-RAS Φ, Algorithm 1 will terminate in a finite number of steps, and it will return a nonempty output that is a correct enumeration (under the adopted representation) of the set $\bar{\mathcal{L}}(\Phi)$.

The space limitations imposed for this document do not allow the inclusion of the proof of the above theorem in the document itself. However, a complete proof of the theorem is contained in a technical report under the same title that can be obtained from the second author upon request.

V. Conclusions

This paper has provided a complete algorithm for enumerating all the correct DAPs of a D/C-RAS Φ that (i) admit a representation as a linear classifier[5], and (ii) are maximal in terms of their admissible subspaces. A significant part of the paper contribution is that it provided thorough definitions of the concept of the "linear DAP" for the considered RAS, and of the notion of "maximality" that can be pursued in this policy space. As discussed in the main part of the paper, the availability of these policies enables all the advantages of the underlying linear representation of the policy logic, and at the same time, it controls the sub-optimality that will result from a potential departure from the maximally permissive DAP Δ^* (in the case that the latter is not linearly representable). The paper also outlines specific aspects in our ongoing work on the real-time control of the considered RAS that have motivated our practical interest in the considered class of policies. Additional future work on the presented results can also investigate the possibility for further enhancements in the detailed implementation of the presented algorithm.

Acknowledgement

This work has been partially supported by NSF grant ECCS-1707695.

References

[1] S. Reveliotis, "Logical Control of Complex Resource Allocation Systems," *NOW Series on Foundations and Trends in Systems and Control*, vol. 4, pp. 1–224, 2017.

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

[3] 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.

[4] A. Nazeem, S. Reveliotis, Y. Wang, and S. Lafortune, "Optimal deadlock avoidance for complex resource allocation systems through classification theory," in *Proceedings of the 10th Intl. Workshop on Discrete Event Systems.* IFAC, 2010, pp. 277–284.

[5] 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.

[5]according to the definition and the usage of this concept in the past literature [4]

[6] R. Cordone and L. Piroddi, "Parsimonious monitor control of petri net models of flexible manufacturing systems," *IEEE Trans. on SMC: Systems*, vol. 43, pp. 215–221, 2013.

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

[8] 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. Autom. Control*, vol. 58, pp. 2772–2787, 2013.

[9] 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.

[10] ——, "A practical approach for maximally permissive liveness-enforcing supervision of complex resource allocation systems," *IEEE Trans. on Automation Science and Engineering*, vol. 8, pp. 766–779, 2011.

[11] ——, "Efficient enumeration of minimal unsafe states in complex resource allocation systems," *IEEE Trans. on Automation Science & Engineering*, vol. 11, pp. 111–124, 2014.

[12] Z. Fei, S. Reveliotis, S. Miremadi, and K. Akesson, "A BDD-based approach for designing maximally permissive deadlock avoidance policies for complex resource allocation systems," *IEEE Trans. on Automation Science and Engineering*, vol. 12, pp. 990–1006, 2015.

[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] S. A. Reveliotis and P. M. Ferreira, "Deadlock avoidance policies for automated manufacturing cells," *IEEE Trans. on Robotics & Automation*, vol. 12, pp. 845–857, 1996.

[15] M. Lawley, S. Reveliotis, and P. Ferreira, "A correct and scalable deadlock avoidance policy for flexible manufacturing systems," *IEEE Trans. on Robotics & Automation*, vol. 14, pp. 796–809, 1998.

[16] 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.

[17] Z. Li and M. Zhou, "Elementary siphons of Petri nets and their application to deadlock prevention in flexible manufacturing systems," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 34, no. 1, pp. 38–51, 2004.

[18] F. Tricas, F. Garcia-Valles, J. M. Colom, and J. Ezpeleta, "A Petri net structure-based deadlock prevention solution for sequential resource allocation systems," in *Proceedings of the ICRA 2005.* IEEE, 2005, pp. 271–277.

[19] T. Murata, "Petri nets: Properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, pp. 541–580, 1989.

[20] 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.

[21] M. V. Iordache and P. J. Antsaklis, *Supervisory Control of Concurrent Systems: A Petri net structural approach.* Boston, MA: Birkhäuser, 2006.

[22] M. V. Iordache, M. J. O., and P. J. Antsaklis, "Synthesis of deadlock prevention supervisors using Petri nets," *IEEE Trans. on Robotics and Automation*, vol. 18, pp. 59–68, 2002.

[23] M. V. Iordache and P. J. Antsaklis, "Design of t-liveness enforcing supervisors in petri nets," *IEEE Trans. on Automatic Control*, vol. 48, pp. 1962–1974, 2003.

[24] G. Weiss, "On the optimal draining of re-entrant fluid lines," Georgia Tech and Technion, Tech. Rep., 1994.

[25] J. G. Dai and G. Weiss, "Stability and instability of fluid models for certain re-entrant lines," *Math. of Op. Res.*, vol. 21, pp. 115–134, 1996.

[26] S. Meyn, *Control Techniques for Complex Networks.* Cambridge, UK: Cambridge University Press, 2008.

[27] M. Ibrahim and S. Reveliotis, "Throughput maximization of capacitated re-entrant lines through fluid relaxation," in *Proc. of ACC'18.* APS, 2018, pp. –.