Designing compact and maximally permissive deadlock avoidance policies for complex resource allocation systems through classification theory: the linear case

Ahmed Nazeem, Spyros Reveliotis, Yin Wang and Stéphane Lafortune

Abstract— **Most of the past research on the problem of deadlock avoidance for complex resource allocation systems (RAS) has acknowledged the fact that the computation of the maximally permissive deadlock avoidance policy (DAP) possesses super-polynomial complexity for most RAS classes, and therefore, it has resorted to solutions that trade off maximal permissiveness for computational tractability. In this work, we distinguish between the *off-line* and the *on-line* computation that is required for the effective implementation of the maximally permissive DAP, and we seek to develop representations of this policy that will require *minimal* on-line computation. The particular representation that we adopt is that of a compact *classifier* that will effect the underlying dichotomy of the reachable state space into safe and unsafe subspaces. Furthermore, in this first study of the aforementioned problem, we restrict our attention to a particular RAS class that is motivated by an ongoing project of ours called Gadara, and accepts separation of the safe and unsafe subspaces of its instantiations through a set of linear inequalities. Through a series of reductions of the derived classification problem, we are also able to attain extensive reductions in the computational complexity of the off-line task of the construction of the sought classifier. We formally establish completeness and optimality properties for the proposed design procedures. We also offer heuristics that, if necessary, can alleviate the computational effort that is necessary for the construction of the sought classifier. Finally, we demonstrate the efficacy of the developed approaches through a series of computational experiments. To the best of our knowledge, these experiments also establish the ability of the proposed methodology to effectively compute tractable implementations of the maximally permissive DAP for problem instances significantly beyond the capacity of any other approach currently available in the literature.**

I. Introduction

Deadlock avoidance for sequential resource allocation systems (RAS) is an ubiquitous problem that arises in most technological applications involving the concurrent execution of a set of processes that compete for the staged acquisition and release of some underlying set of system resources. In particular, the applied control logic must avoid the development of circular waiting patterns where a subset of these processes are waiting upon each other for the release of the resources that are needed for their further advancement, a situation characterized as "deadlock" in the relevant terminology. From a methodological standpoint, the problem can be characterized in the classical Ramadge & Wonham Supervisory Control (R&W SC) framework [1], [2] in a straightforward manner, by (i) expressing the underlying resource allocation dynamics through a Finite State Automaton (FSA) and (ii) requesting the confinement of the RAS behavior to the subspace of this FSA that is defined by its maximal strongly connected component that contains the system state $\mathbf{s}_0$ where the RAS is idle and empty of any jobs.[1] In fact, such a characterization of the problem and its solution establishes also a notion of optimality for the considered problem, since the resulting policy prevents effectively the formation of deadlock while retaining the maximum possible behavioral latitude for the underlying RAS.

However, the direct implementation of the solution approach outlined in the previous paragraph, is seriously impeded by the fact that it necessitates the "real-time" – or the "on-line" – assessment of the co-accessibility of any given RAS state to the empty state $\mathbf{s}_0$, a property that is otherwise known as the state *"safety"*, in the relevant terminology; for most RAS classes, the assessment of state safety is an NP-complete problem [3], [4], [5]. Hence, the research community has tried to circumvent the limitations imposed by this negative result either (a) by compromising for sub-optimal – i.e., non-maximally permissive – solutions that are based on polynomially assessed properties of the relevant RAS states (e.g., [6], [7], [8], [9]), or (b) by adopting alternative, more compact representations of the considered RAS dynamics and hoping that the compactness of these alternative representations, combined with further structural properties and insights revealed by them, will also lead, at least in most practical cases, to fairly compact characterizations of the target policy and to more efficient approaches for its derivation. A modeling framework that seems to hold particular promise along this second line of research, and therefore, has been explored more persistently in the past, is that of Petri nets (PN) [10]. In particular, the attribution of the non-liveness of the RAS-modeling PNs to the formation of some structural objects known as "empty – or, more generally, deadly marked – siphons", has led to the development of a multitude of efforts that seek to characterize the maximally permissive deadlock avoidance policy (DAP) by imposing the minimum possible amount of control that will prevent the formation of such deadly marked siphons. However, a significant complication for these approaches arises from the fact that the maximally permissive DAP might not admit a PN-based representation, and therefore, their practical potential and applicability is not fully explored and understood yet. Another

A. Nazeem and S. Reveliotis are with the School of Industrial & Systems Engineering, Georgia Institute of Technology, email: {anazeem@,spyros@isye.}gatech.edu

Yin Wang is with Hewlett-Packard Labs, email: yin.wang@hp.com

Stéphane Lafortune is with the Department of Electrical Engineering and Computer Science, University of Michigan, email: stephane@eecs.umich.edu

The first two authors were partially supported by NSF grants CMMI-0619978 and CMMI-0928231. The research of the fourth author was supported in part by NSF grants CCF-0819882 and CNS-0930081, and by an award from HP Labs' Innovation Research Program.

[1] All technical concepts are defined more systematically in subsequent parts of this manuscript.

prominent approach pursued within the context of the PN modeling framework is that of the "theory of regions" [11] and its derivatives. The key idea behind the theory of regions, as implemented in the considered problem context, is to first compute the maximally permissive DAP using the standard R&W SC representations and methods mentioned in the opening paragraph, and subsequently encode this policy to a PN model. This approach is also limited by the aforementioned potential inability to express the maximally permissive DAP as a PN. Furthermore, even in its feasible cases, practical experience has shown that it is very demanding from a computational standpoint and it results in PN representations of the maximally permissive DAP that are much larger than the PN modeling the original RAS. The reader can find an extensive coverage of all these past developments in [12], [13], [14], [15], and the references cited therein.

Motivated by the above remarks, in this work we pursue an alternative approach to the design of the maximally permissive DAP for any given RAS. Our work presents some conceptual similarity to the approach of the theory of regions outlined in the previous paragraph, in that we also organize the overall computation of the optimal DAP into two stages, with the first stage obtaining this policy in the R&W SC framework, and the second stage trying to express the obtained result in a more compact form. However, instead of explicitly relying this compression to concepts and results coming from the PN modeling framework, we perceive the optimal DAP as a "dichotomy" of the RAS state space and we essentially seek a compact *"classifier"* that will effect this dichotomy. In this way we are able to tap upon concepts, insights and results that are coming directly from the relevant classification theory. Indeed, as it is revealed in the rest of this document, the methods pursued in this work open new ways for thinking about the considered problem that effectively complement all the previously used approaches. This new line of reasoning subsequently results into new fundamental insights and connects the overall analysis to very classical, and yet very powerful, representation frameworks and techniques. More specifically, the proposed approach first selects a particular representation for the sought classifier that is able to provide effective and computationally efficient classification for the RAS class under consideration, and it defines explicitly the classifier design problem as a minimization problem over a certain parameter space that results from the adopted representation. The computational tractability of the posed minimization problem is facilitated by additional properties of the considered dichotomy that enable an effective compression of the information to be considered explicitly during the classifier construction process, in terms of the size and the dimensionality of the involved data sets. Furthermore, the treatment of the classifier design problem as an explicit optimization problem also enables the development of heuristics that can effectively balance the structural optimality of the sought classifier and the computational complexity that is involved in its development, and of analytical bounds that characterize the potential sub-optimality that is incurred by the use of these heuristics. From a more practical standpoint, the methodology pursued in this paper has allowed the effective and efficient implementation of the maximally permissive DAP for very large-scale RAS, with sizes and underlying state spaces way beyond of those addressed in the current literature.

The rest of this document presents a systematic investigation of the ideas outlined in the previous paragraph, for a particular RAS class where the aforementioned dichotomy of the safe and unsafe subspaces can be effected through a set of linear inequalities; we shall refer to a classifier with such a structure as a "linear classifier". Our intention is to *effectively compute linear classifiers implementing the maximally permissive DAP for any given RAS from the considered class, while minimizing the number of the inequalities involved*; in this way, the on-line complexity of the implemented control scheme will be minimized. In the considered class, the classifier design problem takes the form of a Mixed Integer Programming (MIP) formulation that is rendered tractable through a series of problem reductions that effect the data compression described in the previous paragraph. The computational results that accompany these analytical developments clearly establish that, within the scope of the considered RAS class, the proposed approach can compute effectively the maximally permissive DAPs for RAS configurations that are perceived as extremely large (and therefore, intractable) by the standards of the current literature. At the same time, the optimal classifiers obtained by those formulations have turned out to be impressively compact, utilizing only a small number of inequalities.

The practical motivation and justification of the key ideas underlying the developments presented in this work, as well as the particular RAS class that is the focus of these developments and its accompanying technical specifications, have been provided by the authors' experiences in the context of an ongoing project, called "Gadara", which has been undertaken in collaboration with HP Labs. The goal of the Gadara project is to develop a software tool that will automatically take a multi-threaded deadlock-prone program as input and instrument it with appropriate control logic so that the resultant code is deadlock-free. Hence, the entire problem addressed by Gadara reduces to the design and the deployment of a DAP that will manage the allocation of the "locks" shared by the concurrently executing processes (i.e., threads) in the context of the considered program [16], [17], [18], [19]. Furthermore, in the Gadara context, the deployed DAP must ensure the safe and live execution of the underlying processes in a way that (a) imposes the minimal necessary restriction in the execution of these processes, and (b) causes the minimum possible computational "overhead" for the underlying operating system. Requirement (a) implies that the maximal permissiveness of the applied control logic is of paramount importance in the considered application context. Requirement (b) introduces the additional notion of *"structural minimality"* for the derived supervisors that is pursued in this work.

In light of the above positioning of the paper objectives and contributions, the rest of the manuscript is organized as follows: Section II introduces the RAS class to be considered in this work, defines formally the corresponding deadlock avoidance problem, and establishes some of its properties that are crucial for the development of the main results of the paper. The main results themselves are presented in Sections III–V. More specifically, Section III first provides a formal definition of the classification problem to be considered in this work, and subsequently,

it proceeds to its reduction to an equivalent classification problem with a much smaller input set in terms of the explicitly considered state vectors and their dimensionality. Section IV addresses the synthesis of a linear classifier for the reduced classification problem, by formulating and solving this problem as a mixed integer program. On the other hand, Section V introduces the heuristic approach to the synthesis of the sought classifier, that was mentioned in the previous paragraphs. Section VI reports a series of computational experiments that demonstrate the feasibility of the methodological approaches proposed in the manuscript, and reveal the extensive computational gains that are obtained by them. Finally, Section VII concludes the paper by summarizing its main contributions and suggesting possible extensions of the presented results. Closing this introductory section, we also notice, for completeness, that an abridged version of the results of this manuscript can be found in [20]; that manuscript contains a preliminary version of the material of Sections II–IV and VI, and it omits all the technical proofs.

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

The considered RAS class We begin the more technical discussion of the paper developments, by providing a formal characterization of the RAS class to be considered in this work. This class can be perceived as a specialization of the generic RAS modeling framework of [13], that is obtained through the introduction of some additional assumptions for the underlying RAS structure. Hence, next we first review the RAS concept, as defined in [13], and subsequently we discuss the specific assumptions that will define the RAS sub-class considered in this work. Also, in the rest of this manuscript the notation $\mathbb{Z}$, $\mathbb{Z}_0^+$ and $\mathbb{Z}^+$ will respectively denote the set of integers, non-negative integers and strictly positive integers.

Definition 1: [13] A *(sequential) resource allocation system (RAS)* is defined as a 4-tuple $\Phi =< \mathcal{R}, C, \mathcal{P}, \mathcal{A} >$[2] where:

1. $\mathcal{R} = \{R_1, \ldots, R_m\}$ is the set of the system *resource types*.
2. $C : \mathcal{R} \rightarrow \mathbb{Z}^+$ is the system *capacity* function, with $C(R_i) \equiv C_i$ characterizing the number of identical units from resource type R_i that are available in the system. Resources are considered to be *reusable*, i.e., they are engaged by the various processes according to an allocation/de-allocation cycle, and each such cycle does not affect their functional status or subsequent availability.
3. $\mathcal{P} = \{J_1, \ldots, J_n\}$ is the set of the system *process types* supported by the considered system configuration. Each process type J_j is a composite element itself; in particular, $J_j = < \mathcal{S}_j, \mathcal{G}_j >$, where:
   (a) $\mathcal{S}_j = \{\Xi_{j1}, \ldots, \Xi_{j,l(j)}\}$ is the set of *processing stages* involved in the definition of process type J_j, and
   (b) $\mathcal{G}_j$ is a data structure that defines the sequential logic over the set of processing stages $\mathcal{S}_j$, that governs the execution of any process instance of type J_j.
4. $\mathcal{A} : \bigcup_{j=1}^{n} \mathcal{S}_j \rightarrow \prod_{i=1}^{m} \{0, \ldots, C_i\}$ is the *resource allocation function*, which associates every processing stage Ξ_{jk} with a *resource allocation request* $\mathcal{A}(j,k) \equiv \mathcal{A}_{jk}$. More specifically, each $\mathcal{A}_{jk}$ is an m-dimensional vector, with its i-th component indicating the number of resource units of resource type R_i necessary to support the execution of stage Ξ_{jk}. Obviously, in a well-defined RAS, $\mathcal{A}_{jk}(i) \leq C_i$, $\forall j,k,i$. Also, it is assumed that $\mathcal{A}_{jk} \neq \mathbf{0}$, i.e., every processing stage requires at least one resource unit for its execution.

[2]The complete definition of a RAS, according to [13], involves an additional component that characterizes the time-based – or *quantitative* – dynamics of the RAS, but this component is not relevant in the modeling and analysis to be pursued in the following developments, and therefore, it is omitted.

For complexity considerations, we also define the quantity $|\Phi| \equiv |\mathcal{R}| + |\bigcup_{j=1}^{n} \mathcal{S}_j| + \sum_{i=1}^{m} C_i$ as the *"size"* of RAS Φ. Furthermore, for notational convenience, in the following we shall set $\xi \equiv \sum_{j=1}^{n} |\mathcal{S}_j|$; i.e., ξ denotes the number of distinct processing stages supported by the considered RAS, across the entire set of its process types. Finally, in some of the subsequent developments, the various processing stages Ξ_{jk}, $j = 1, \ldots, n$, $k = 1, \ldots, l(j)$, will be considered in the context of a total ordering imposed on the set $\bigcup_{j=1}^{n} \mathcal{S}_j$; in that case, the processing stages themselves and their corresponding attributes will be indexed by a single index q that runs over the set $\{1, \ldots, \xi\}$ and indicates the position of the processing stage in the considered total order.

It is clear from the above that Definition 1 encompasses an entire taxonomy of RAS, obtained by the further specification of the underlying process structure (cf. item 3(b)), the capacities of the various resource types (cf. item 2), and the applied resource allocation protocol (cf. item 4). The RAS class to be considered in this work is defined by the following three assumptions that address each of the aforementioned items:

Assumption 1: In the considered RAS, the data structure $\mathcal{G}_j$ that defines the sequential logic of process type J_j, $j = 1, \ldots, n$, corresponds to a connected digraph $(\mathcal{V}_j, \mathcal{E}_j)$, where the graph node set $\mathcal{V}_j$ is in one-to-one correspondence with the processing stage set, $\mathcal{S}_j$. Furthermore, there are two subsets $\mathcal{V}_j^{\nearrow}$ and $\mathcal{V}_j^{\searrow}$ of $\mathcal{V}_j$ respectively defining the sets of initiating and terminating processing stages for process type J_j. The connectivity of digraph $\mathcal{G}_j$ is such that every node $v \in \mathcal{V}_j$ is accessible from the node set $\mathcal{V}_j^{\nearrow}$ and co-accessible to the node set $\mathcal{V}_j^{\searrow}$. Finally, any directed path of $\mathcal{G}_j$ leading from a node of $\mathcal{V}_j^{\nearrow}$ to a node of $\mathcal{V}_j^{\searrow}$ constitutes a complete execution sequence – or a "route" – for process type J_j.

Assumption 2: In the considered RAS, $C_i = 1$, $\forall i = 1, \ldots, m$.

Assumption 3: In the considered RAS, the resource allocation requests $\mathcal{A}_{jk}$, $j = 1, \ldots, n$, $k = 1, \ldots, l(j)$, are "conjunctive", i.e., a processing stage Ξ_{jk} can request an arbitrary nonempty subset of the system resources for its execution. Furthermore, a process instance executing processing stage Ξ_{jk} will be able to advance to a successor processing stage Ξ_{jq}, only after it is allocated the resource differential $(A_{jq} - A_{jk})^+$; and it is only upon this advancement that the process will release the resource units $|(A_{jq} - A_{jk})^-|$, that are not needed anymore.

In the following, we shall refer to the RAS sub-class that is defined by Assumptions 1–3 as the (class of) "Gadara" RAS. We want to point out that the directed paths that express the various process routes, according to Assumption 1 are not necessarily simple; i.e., processes of a Gadara RAS can present cycling through a number of processing stages, during their execution.

On the other hand, Assumption 2 implies that the various instantiations, Φ, of the Gadara RAS are completely defined by the specification of the resource set $\mathcal{R}$, the set of process types $\mathcal{P}$, and the resource allocation function $\mathcal{A}$, i.e., for these RAS, we can set $\Phi =< \mathcal{R}, \mathcal{P}, \mathcal{A} >$. Finally, as it will be revealed by the subsequent developments, the proposed methodology for the synthesis of maximally permissive and compact DAPs can be applied even to the broader RAS class that is obtained by the removal of Assumption 2; this RAS class is an extension of the Disjunctive / Conjunctive (D/C-) RAS defined in [13], that further allows for internal cycling in the process routes. Successful implementation of the presented approach to any instance Φ of this broader RAS class will return a representation of the corresponding maximally permissive DAP as a minimal set of linear inequalities. However, in the D/C-RAS context, there is no guarantee that the underlying reachable safe and unsafe subspaces will be linearly separable, and therefore, the MIP formulation of Section IV or the heuristic of Section V might fail to return a classifier.

Modeling the dynamics of the Gadara RAS as a Finite State Automaton Next we discuss how the dynamics of the Gadara RAS $\Phi =< \mathcal{R}, \mathcal{P}, \mathcal{A} >$, introduced in the previous paragraph, can be formally described by a *Deterministic Finite State Automaton (DFSA)* [2]. This FSA will be denoted as $G(\Phi) = (S, E, f, \mathbf{s}_0, S_M)$, and it is defined as follows:[3]

1. The *state set* S consists of ξ-dimensional vectors $\mathbf{s}$. The components $\mathbf{s}[q]$, $q = 1, \ldots, \xi$, 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 RAS state modeled by $\mathbf{s}$. Hence, S consists of all the vectors $\mathbf{s} \in (\mathbb{Z}_0^+)^\xi$ that further satisfy

$$\forall i = 1, \ldots, m, \ \sum_{q=1}^{\xi} \mathbf{s}[q] \cdot \mathcal{A}(\Xi_q)[i] \leq 1 \tag{1}$$

where, according to the adopted notation, $\mathcal{A}(\Xi_q)[i]$ denotes the allocation request for resource R_i that is posed by stage Ξ_q, and the unit element appearing in the right-hand-side of the equation expresses the unit resource capacities that define the class of Gadara RAS.

2. The *event set* E is the union of the disjoint event sets $E^{\nearrow}$, $\bar{E}$ and $E^{\searrow}$, where:

(a) $E^{\nearrow} = \{e_{rp} : r = 0, \ \Xi_p \in \bigcup_{j=1}^{n} \mathcal{V}_j^{\nearrow}\}$, i.e., event e_{rp} represents the *loading* of a new process instance that starts from stage Ξ_p.

(b) $\bar{E} = \{e_{rp} : \exists j \in 1, \ldots, n$ s.t. Ξ_p is a successor of Ξ_r in digraph $\mathcal{G}_j\}$, i.e., e_{rp} represents the *advancement* of a process instance executing stage Ξ_r to a successor stage Ξ_p.

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

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

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

Furthermore, $f(\mathbf{s}, e_{rp})$ is a *partial* function defined only if the resulting state $\mathbf{s}' \in S$.

4. The *initial state* $\mathbf{s}_0$ is set equal to $\mathbf{0}$, which corresponds to the situation when the system is empty of any process instances.

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

Let $\hat{f}$ 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, the *reachable subspace* of $G(\Phi)$ is the subset S_r of S defined as follows:

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

We also define the *safe subspace* of $G(\Phi)$, S_s, by:

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

S_s contains those states of S that are co-accessible to the marked state $\mathbf{s}_0$. 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 structure of the maximally permissive LES The desired – or "target" – behavior of RAS Φ is expressed by the *marked language* $L_m(G)$, which is defined by means of the marked state $\mathbf{s}_0$, as follows:

$$L_m(G) \ \equiv \ \{\sigma \in L(G) : \hat{f}(\mathbf{s}_0, \sigma) = \mathbf{s}_0\} \tag{4}$$

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, starting from state $\mathbf{s}_0$, a *maximally permissive deadlock avoidance policy* must allow a system-enabled transition to a next state $\mathbf{s}$ if and only if (*iff*) $\mathbf{s}$ belongs to S_s. This characterization of the maximally permissive DAP ensures its uniqueness for any given RAS instantiation. It also implies that the policy can be effectively implemented through any mechanism that recognizes and rejects the unsafe states that are accessible through one-step transitions from S_{rs}.

Example We demonstrate the various concepts introduced above through a particular instance of the Gadara RAS that is presented in Table I. This RAS will also provide an expository base for all the technical concepts, methods and issues addressed in this document. To facilitate the presentation in the limited space of this document, the considered RAS is very small and simple in terms of its structure, yet its behavioral study gives rise to all aspects and issues that will be our major focus in the subsequent developments. More specifically, the Gadara RAS considered in Table I consists of three resource types R_1, R_2 and R_3, each of unit capacity, and two process types J_1 and J_2. The sequential logic corresponding to each of these two processes has a simple linear structure involving three stages. Finally, the resource allocation function, $\mathcal{A}$, of this RAS can also be derived from the information on the process routes provided in the table.

[3] We also notice, for completeness, that an alternative formal characterization of the resource allocation dynamics investigated by the Gadara project, that is based on the Petri net modeling framework, can be found in [18].

TABLE I
The Gadara RAS considered in the provided example

Resource Types	$\{R_1, R_2, R_3\}$
Resource Capacities	$C(R_1) = C(R_2) = C(R_3) = 1$
Process Types	$\{J_1, J_2\}$
Process Routes	$\mathcal{G}_1: \ R_1 \rightarrow R_2 \rightarrow R_3$
	$\mathcal{G}_2: \ R_3 \rightarrow R_2 \rightarrow R_1$

Figure 1 depicts the reachable state space S_r for the RAS of Table I and its partition to the safe subspace S_{rs}, consisting of the white states in the figure, and the unsafe subspace $S_{r\bar{s}}$, consisting of the red states. As it can be seen in the figure, the considered RAS has 20 reachable states, 15 of which are safe and 5 are unsafe. Figure 1 also depicts the transitions that must be disabled by the maximal permissive DAP, under the assumption that the system starts its operation from the empty state, denoted by state q^0 in the figure. Finally, we notice that, while Figure 1 adopts a graphical encoding of the various states of the RAS of Table I, the state vector $\mathbf{s}$ characterizing the resource allocation of this RAS according to the definition of the DFSA $G(\Phi)$ provided in the previous paragraphs, is a 6-dimensional vector; the first three components of $\mathbf{s}$ correspond to the sequence of the three processing stages of job type J_1 and the last three components correspond to the sequence of the three processing stages of job type J_2. Furthermore, the unit capacity of the three resources R_1, R_2 and R_3 imply that the state vector $\mathbf{s}$ will be binary.

Some monotonicities observed by the state safety and unsafety concepts In this paragraph, we establish that the subspaces S_{rs} and $S_{r\bar{s}}$ present some additional structure that will become useful in the design of the target classifier. The next definition will facilitate the formal statement and development of the relevant results.

Definition 2: Let $\mathbf{x}, \mathbf{x}'$ denote two vectors in the vector space $(\mathbb{Z}_0^+)^\xi$. Then, the (partial) ordering relationship "$\leq$" imposed on $(\mathbb{Z}_0^+)^\xi$ is defined by:

$$\mathbf{x} \leq \mathbf{x}' \iff (\forall i = 1, \ldots \xi, \ \mathbf{x}[i] \leq \mathbf{x}'[i]) \tag{5}$$

Furthermore, $\mathbf{x} < \mathbf{x}'$ (resp. $\mathbf{x} > \mathbf{x}'$) will denote the fact that $\mathbf{x} \leq \mathbf{x}'$ (resp. $\mathbf{x} \geq \mathbf{x}'$) and there is at least a pair of components $\mathbf{x}[i]$, $\mathbf{x}'[i]$ for which the corresponding inequality is strict.

It should be clear from the discussion provided in the previous paragraphs that the ability of the activated processes in a given RAS state $\mathbf{s} \in S$ to proceed to completion, depends on the existence of a sequence $< \mathbf{s}^{(0)} \equiv \mathbf{s}, e^{(1)}, \mathbf{s}^{(1)}, e^{(2)}, \mathbf{s}^{(2)}, \ldots, \mathbf{s}^{(n-1)}, e^{(n)}, \mathbf{s}^{(n)} \equiv \mathbf{s}_0 >$, such that at every state $\mathbf{s}^{(i)}$, $i = 0, 1 \ldots, n-1$, the free (or "slack") resource capacities at that state enable the job advancement corresponding to event $e^{(i+1)}$. Furthermore, if such a terminating sequence exists for a given state $\mathbf{s}$, then the event feasibility condition defined by Equation 1 implies that this sequence will also provide a terminating sequence for every other state $\mathbf{s}' \leq \mathbf{s}$. On the other hand, if state $\mathbf{s}$ possesses no terminating sequences, then it can be safely inferred that no such terminating sequences will exist for any other state $\mathbf{s} \leq \mathbf{s}'$ (since, otherwise, there should also exist a terminating sequence for $\mathbf{s}$, according to the previous remark). The next proposition provides a formal statement to the above observations; these results are well known in the literature, and therefore, their formal proof is omitted.[4]

Proposition 1: Let $\mathbf{s}$, $\mathbf{s}'$ denote two states of a given Gadara RAS Φ. Then,

1. $\mathbf{s} \in S_s \ \wedge \ \mathbf{s}' \leq \mathbf{s} \implies \mathbf{s}' \in S_s$
2. $\mathbf{s} \in S_u \ \wedge \ \mathbf{s} \leq \mathbf{s}' \implies \mathbf{s}' \in S_u$

In light of Proposition 1, next we also define the concepts of *maximal safe state* and *minimal unsafe state*, that will play an important role in the subsequent developments:

Definition 3: Given a Gadara RAS $\Phi = (\mathcal{R}, \mathcal{P}, \mathcal{A})$,

1. a reachable safe state $\mathbf{s} \in S_{rs}$ is *maximal iff* $\neg\exists$ a reachable safe state $\mathbf{s}' \in S_{rs}$ such that $\mathbf{s}' > \mathbf{s}$;
2. a reachable unsafe state $\mathbf{s} \in S_{r\bar{s}}$ is *minimal iff* $\neg\exists$ a reachable unsafe state $\mathbf{s}' \in S_{r\bar{s}}$ such that $\mathbf{s}' < \mathbf{s}$.

Also, in the sequel, the set of maximal reachable safe states will be denoted by $\bar{S}_{rs}$, and the set of minimal reachable unsafe states will be denoted by $\bar{S}_{r\bar{s}}$.

An additional implication of Proposition 1 that will be useful in the subsequent developments is stated in the following corollary:

Corollary 1: Consider a Gadara RAS $\Phi = (\mathcal{R}, \mathcal{P}, \mathcal{A})$, and let $Conv(S_{rs})$ denote the convex hull of the points corresponding to the states in its reachable and safe subspace, S_{rs}. Also, let $\mathbf{x} \in Conv(S_{rs})$. Then, $Conv(S_{rs})$ also contains any other vector $\mathbf{x}'$ such that $\mathbf{0} \leq \mathbf{x}' \leq \mathbf{x}$.

Proof: To establish the result of Corollary 1, it suffices to show that the state set S_{rs} contains all possible chains of integer vectors between the origin $\mathbf{0}$ and its maximal elements $\mathbf{s} \in \bar{S}_{rs}$.

For this, first we argue that all possible chains of integer vectors between the origin $\mathbf{0}$ and the elements of $\bar{S}_{rs}$ belong in S_r. Indeed, consider a state $\mathbf{s}$ belonging in an integer vector chain from state $\mathbf{0}(\equiv \mathbf{s_0})$ to an element $\mathbf{s}' \in \bar{S}_{rs}$. Since $\mathbf{s}' \in \bar{S}_{rs}$, it is a reachable state, and therefore, there exists an event sequence σ' such that $\mathbf{s}' = \hat{f}(\mathbf{s}_0, \sigma')$. Furthermore, from the specification of the states $\mathbf{s}$ and $\mathbf{s}'$ it also holds that $\mathbf{s} \leq \mathbf{s}'$, and therefore, the process instances contained in $\mathbf{s}$ is a subset of the process instances contained in $\mathbf{s}'$. But since, in the context of the Gadara RAS, the various process instances do not interact with each other except for the sharing of the system resources, it is easy to see that the reachability of state $\mathbf{s}'$ implies also the reachability of state $\mathbf{s}$; a corresponding event sequence σ can be obtained from the event sequence σ' mentioned above by removing from it all the events concerning process instances not belonging in $\mathbf{s}$.

The fact that the considered state $\mathbf{s}$ belongs also in S_s, and therefore in S_{rs}, results from Proposition 1 and the aforestated fact that $\exists \mathbf{s}' \in \bar{S}_{rs}$ s.t. $\mathbf{s} \leq \mathbf{s}'$. □

The binary nature of the state space of the Gadara RAS and its implications For any Gadara RAS $\Phi =< \mathcal{R}, \mathcal{P}, \mathcal{A} >$, Equation 1 when combined with the non-zero nature of the resource allocation requests $\mathcal{A}_{jk}$ further imply that the RAS state vector $\mathbf{s}$ is of a binary nature; i.e., the state space S of the DFSA $G(\Phi)$, as well as any other subspace of S, consist of a number of extreme points on the ξ-dimensional hypercube $\mathcal{C}$ defined by

$$\mathcal{C} \equiv \{(x_1, x_2, \ldots, x_\xi): \ 0 \leq x_i \leq 1, \ \forall i = 1, \ldots, \xi\} \tag{6}$$

[4]We notice, for completeness, that a formal proof for these results can be obtained, for instance, through the analytical characterization of state safety that is presented in [7], [21].

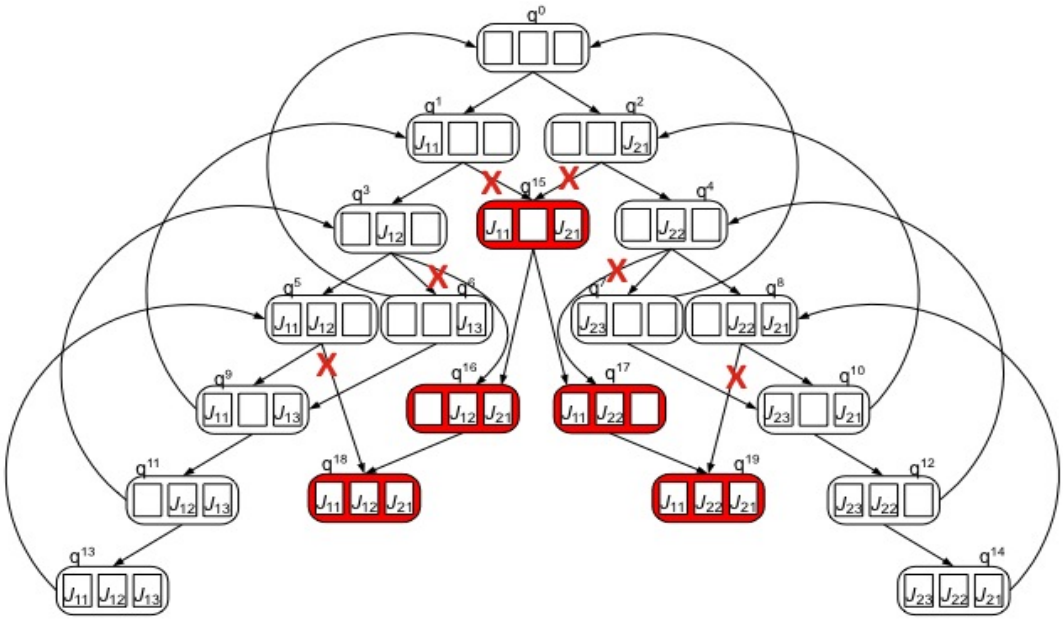

Fig. 1. The finite state automaton modeling the reachable sub-space of the RAS defined in Table I. In the adopted state representation, the three inner cells indicate the status of resources R_1, R_2 and R_3, in this order. Empty cells imply a free resource. Cells corresponding to allocated resources are annotated by the processing stage of the process instance that holds the corresponding resource. States depicted in red are the unsafe states that must be eliminated from the system behavior through the employment of a deadlock avoidance policy. In particular, the maximally permissive DAP must recognize and block the transitions indicated by the red crossings in the figure.

Next we show that every extreme point of $\mathcal{C}$ can be effectively separated from the rest by a single linear inequality.

Proposition 2: Consider the hypercube $\mathcal{C}$ defined by Equation 6, and let $\mathbf{x} = (x_1, x_2, \ldots, x_\xi)$ denote one of its extreme points. Then, point $\mathbf{x}$ can be separated from the remaining extreme points of $\mathcal{C}$ by the linear inequality $\mathbf{a}^T \cdot \mathbf{x} \leq b$ where

$$\mathbf{a}[i] := \begin{cases} 1, & \text{if } \mathbf{x}[i] = 1 \\ -1, & \text{if } \mathbf{x}[i] = 0 \end{cases} \quad ; \quad b := \sum_{i=1}^{\xi} \mathbf{x}[i] - 1 \qquad (7)$$

Proof: Under the assignments of Equation 7, $\mathbf{a}^T \cdot \mathbf{x} = \sum_{i=1}^{\xi} \mathbf{x}[i] > b$. On the other hand, the misalignment of the unit components of any other extreme point $\mathbf{x}'$ with the positive elements of vector $\mathbf{a}$ implies that $\mathbf{a}^T \cdot \mathbf{x}' \leq b$. □

The practical implications of Propostion 2 for the DAP design problems that are addressed in the rest of this document, are stated in the following corollary:

Corollary 2: Consider two sets X and $\hat{X}$ that consist of binary vectors from some ξ-dimensional space, and further assume that $\hat{X} \subset X$. Then, there exists a system of linear inequalities $\{\mathbf{a}_i^T \cdot \mathbf{x} \leq b_i,\ i = 1, \ldots, \nu\}$ that is satisfied by every $\mathbf{x} \in \hat{X}$ and it is violated by every $\mathbf{x} \in X \backslash \hat{X}$, and therefore, it can function as a linear classifier for $\hat{X}$ and $X \backslash \hat{X}$.

Proof: Consider the separating inequalities that are implied by Proposition 2 for each $\mathbf{x} \in X \backslash \hat{X}$. Then, the system of linear inequalities that is defined by the conjunction of all these inequalities is satisfied by every vector $\mathbf{x} \in \hat{X}$. At the same time, its construction implies that it is violated by any vector $\mathbf{x} \in X \backslash \hat{X}$.

III. The Classifier Design Problem in the Context of the Gadara RAS and its Simplification

The classification problem considered in this work This section considers the problem of synthesizing an effective and compact classifier that will separate the reachable safe subspace S_{rs} from the reachable unsafe subspace $S_{r\bar{s}}$, for any given RAS Φ that belongs to the class of Gadara RAS. Corollary 2 of the previous section guarantees the existence of a set of linear inequalities that will perform the requested separation. Our objective is to find the minimum number of linear inequalities that achieves the separation. The next definition provides a formal characterization of the concepts that are necessary for the exact positioning of our problem:

Definition 4: Consider two vector sets G and H from an ξ-dimensional vector space V.

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

$$\forall \mathbf{g} \in G:\ \forall i \in \{1, \cdots, k\},\ \mathbf{A}(i,.) \cdot \mathbf{g} \leq b_i \quad \wedge$$
$$\forall \mathbf{h} \in H : \exists i_s \in \{1, \cdots, k\},\ \mathbf{A}(i_s,.) \cdot \mathbf{h} > b_{i_s} \qquad (8)$$

2. A linear classifier – or separator – for vector sets G and H is *minimal*, iff it uses the minimum possible number of linear

inequalities that can separate these two sets.

Hence, the problem addressed in this section can be succinctly stated as follows:

Definition 5: **– The considered classification problem** Given a Gadara RAS Φ, construct a minimal linear separator for the vector sets corresponding to the sub-spaces S_{rs} and $S_{r\bar{s}}$, i.e., the reachable safe and the reachable unsafe states of the considered RAS Φ.

A major difficulty for the systematic construction of the aforementioned classifier for any practical instantiation of the Gadara RAS, is the huge cardinality of the sets S_{rs} and $S_{r\bar{s}}$. In the following, we utilize the properties of the state space, S, of the Gadara RAS, that were established in Section II, in order to extract two subsets of the sets S_{rs} and $S_{r\bar{s}}$ that are of significantly reduced cardinality compared to their respective supersets, and when used as input to the classifier design process, they result to a set of linear inequalities that still classifies correctly the original sets S_{rs} and $S_{r\bar{s}}$.

Thinning the set $S_{r\bar{s}}$ by focusing on its "boundary" to the reachable and safe subspace As observed in the characterization of the maximally permissive DAP in Section II, the effective implementation of this policy for any given RAS Φ is equivalent to the recognition and the blockage of transitions from the safe to the unsafe region of the underlying state space S. In the context of the state classification problem addressed in this section, this remark further implies that the sought classifier needs to discriminate successfully only between the set S_{rs} and the subset of the set $S_{r\bar{s}}$ that contains all the states $\mathbf{s} \in S_{r\bar{s}}$ that are reachable from some state $\mathbf{s}' \in S_{rs}$ in a single transition. In the following, we shall denote this subset of $S_{r\bar{s}}$ by $S^b_{r\bar{s}}$ and we shall refer to its elements as the *"boundary"* reachable unsafe states. Table II exemplifies the concept of the set of boundary reachable unsafe states, $S^b_{r\bar{s}}$, by applying it to the Gadara RAS of Table I. For this particular RAS it is easy to see, through the state transition diagram (STD) provided in Figure 1, that $S^b_{r\bar{s}} = S_{r\bar{s}}$.

Thinning the sets S_{rs} and $S^b_{r\bar{s}}$ by respectively focusing on their maximal and minimal elements The extraction of the "boundary" reachable unsafe states, $\mathbf{s} \in S^b_{r\bar{s}}$, from the broader set $S_{r\bar{s}}$, is the first step of the set "thinning" process that is proposed in this document. The next two propositions establish that it is possible to obtain a minimal linear classifier for the entire sets S_{rs} and $S^b_{r\bar{s}}$, by focusing the classifier design process only on the maximal elements of the first set, $\bar{S}_{rs}$, and the minimal elements of the second set, $\bar{S}^b_{r\bar{s}}$.

Proposition 3: Any linear separator $(\mathbf{A}, \mathbf{b})$ for the sets $\bar{S}_{rs}$ and $\bar{S}^b_{r\bar{s}}$ with non-negative coefficients a_{ij} and b_i, is also an effective separator for the entire sets S_{rs} and $S^b_{r\bar{s}}$.

Proof: Let $\mathbf{s} \in S_{rs}$ be an arbitrary non-maximal safe state vector, and $\mathbf{s}^* \in \bar{S}_{rs}$ be a maximal safe state vector such that $\mathbf{s}^* > \mathbf{s}$.

Also, let $\mathbf{u} \in S^b_{r\bar{s}}$ be an arbitrary non-minimal boundary unsafe state vector, and $\mathbf{u}^* \in \bar{S}^b_{r\bar{s}}$ be a minimal boundary unsafe state vector such that $\mathbf{u}^* < \mathbf{u}$.

Finally, let $\mathbf{A}(i,.)$ be the vector of coefficients for the i-th hyperplane separating the sets $\bar{S}_{rs}$ and $\bar{S}^b_{r\bar{s}}$, $i = 1, \ldots, k$. Then, according to the stated assumptions, $\mathbf{A}(i,.) \geq \mathbf{0}\ \forall i \in \{1, \cdots, k\}$. Also, according to the definition of linear separation: $\forall \mathbf{s}' \in \bar{S}_{rs} :\ \forall i \in \{1, \cdots, k\},\ \mathbf{A}(i,.) \cdot \mathbf{s}' \leq b_i$ and $\forall \mathbf{u}' \in \bar{S}^b_{r\bar{s}} :\ \exists i_{u'} \in \{1, \cdots, k\}$ such that $\mathbf{A}(i_{u'},.) \cdot \mathbf{u}' > b_{i_{u'}}$. But then, the non-negativity of $\mathbf{A}(i,.)$, combined with the presumed relations of $\mathbf{s}$ to $\mathbf{s}^*$ and of $\mathbf{u}$ to $\mathbf{u}^*$, further imply that:

- $\forall i :\ \mathbf{A}(i,.) \cdot \mathbf{s} \leq \mathbf{A}(i,.) \cdot \mathbf{s}^* \leq b_i$
- $\exists i_{u^*} :\ \mathbf{A}(i_{u^*},.) \cdot \mathbf{u} \geq \mathbf{A}(i_{u^*},.) \cdot \mathbf{u}^* > b_{i_{u^*}}$

Since states $\mathbf{s}$ and $\mathbf{u}$ were arbitrarily chosen, we can infer that $\forall \mathbf{s} \in S_{rs} :\ \forall i \in \{1, \cdots, k\},\ \mathbf{A}(i,.) \cdot \mathbf{s} \leq b_i$ and $\forall \mathbf{u} \in S^b_{r\bar{s}} :\ \exists i_u \in \{1, \cdots, k\},\ \mathbf{A}(i_u,.) \cdot \mathbf{s} > b_{i_u}$, which means that the separator $(\mathbf{A}, \mathbf{b})$ is also an effective separator for S_{rs} and $S^b_{r\bar{s}}$. □

Proposition 4: The set of minimal linear separators for the classification problem of Definition 5 will always contain a separator $(\mathbf{A}, \mathbf{b})$ with non-negative coefficients a_{ij} and b_i.

Proof: Suppose that the minimum number of linear inequalities with free coefficients needed to seperate S_{rs} from $S_{r\bar{s}}$ is k, and such a minimal separator is represented by the set of hyperplanes $H = \{h_i \equiv (\mathbf{a}_i, b_i), i \in \{1, \cdots, k\}\}$. We need to show that there exists a linear separator with non-negative coefficients that achieves the separation of these two sets, and the number of inequalities employed by this new separator is also k.

If H happens to satisfy the posed non-negativity requirement, then there is nothing left to be proved. Otherwise, consider an inequality $h_i = (\mathbf{a}_i, b_i)$ employed in H that violates the posed non-negativity requirement. First, we notice that from the definition of minimal linear separation provided in the previous paragraphs, we can infer that

$$\forall\, \mathbf{x} \in S_{rs} :\ \mathbf{a}_i \cdot \mathbf{x} \leq b_i \ \ \wedge \ \ \exists\, \mathbf{y} \in S_{r\bar{s}} :\ \mathbf{a}_i \cdot \mathbf{y} > b_i \tag{9}$$

The first component in the proposition of Equation 9 is a direct application of the definition of linear separation, while the second component is due to the minimality of separation (if this component was not valid, h_i can be removed from H without any consequences for the correctness of the classification process, and therefore, separator H is not minimal). Let $S^{(i)}_{r\bar{s}} \equiv \{\mathbf{y} : \mathbf{y} \in S_{r\bar{s}} \ \wedge \ \mathbf{a}_i \cdot \mathbf{y} > b_i\}$ and also set $|S_{rs}| \equiv m_s$ and $|S^{(i)}_{r\bar{s}}| \equiv m_{u_i}$. Then, we have that

$$\forall \mathbf{x} \in S_{rs} :\ \mathbf{a}_i \cdot \mathbf{x} \leq b_i \ \ \wedge \ \ \forall \mathbf{y} \in S^{(i)}_{r\bar{s}} :\ \mathbf{a}_i \cdot \mathbf{y} > b_i \tag{10}$$

Equation 10, when combined with the finiteness of the set $S^{(i)}_{r\bar{s}}$, further implies that $\exists\, \hat{\delta} > 0$ such that $\forall \delta \in (0, \hat{\delta}]$:

$$\begin{bmatrix} \mathbf{x_1}^T & -1 \\ \mathbf{x_2}^T & -1 \\ \cdots & \cdots \\ \mathbf{x_{m_s}}^T & -1 \\ -\mathbf{y_1}^T & 1 \\ -\mathbf{y_2}^T & 1 \\ \cdots & \cdots \\ -\mathbf{y_{m_{u_i}}}^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{a}_i^T \\ b_i \end{bmatrix} \leq \begin{bmatrix} 0 \\ 0 \\ \cdots \\ 0 \\ -\delta \\ -\delta \\ \cdots \\ -\delta \end{bmatrix} \tag{11}$$

where operator 'T' implies transposition.

Also, notice that since (i) all inequalities employed by separator H are of the '$\leq$' type (according to Definition 4), and (ii) the initial state $\mathbf{s_0} \equiv \mathbf{0}$ satisfies these inequalities (being a safe state), the right-hand-side coefficient b_i of the considered inequality $h_i = (\mathbf{a}_i, b_i)$ must be non-negative, i.e.,

$$b_i \geq 0 \tag{12}$$

Then, Equations 11 and 12, together with Farkas' lemma ([22], pg. 165), imply that, $\forall \delta \in (0, \hat{\delta}]$, the following system of inequalities in the variable vector $\mathbf{p}$ is infeasible:

$$\mathbf{p} \geq \mathbf{0} \wedge \mathbf{p}^T \begin{bmatrix} \mathbf{x_1}^T \\ \mathbf{x_2}^T \\ \cdots \\ \mathbf{x_{m_s}}^T \\ -\mathbf{y_1}^T \\ -\mathbf{y_2}^T \\ \cdots \\ -\mathbf{y_{m_{u_i}}}^T \end{bmatrix} = \mathbf{0} \wedge \mathbf{p}^T \begin{bmatrix} -1 \\ -1 \\ \cdots \\ -1 \\ 1 \\ 1 \\ \cdots \\ 1 \end{bmatrix} \geq 0 \wedge \mathbf{p}^T \begin{bmatrix} 0 \\ 0 \\ \cdots \\ 0 \\ -\delta \\ -\delta \\ \cdots \\ -\delta \end{bmatrix} < 0 \quad (13)$$

Next we shall use the infeasibility of Equation 13 in order to show the existence of another hyperplane $h'_i = (\mathbf{a}'_i, b'_i)$ with non-negative coefficients which separates S_{rs} and $S^{(i)}_{r\bar{s}}$. If such a hyperplane exists, it can replace the original hyperplane $h_i = (\mathbf{a}_i, b_i)$ in separator H while maintaining a successful separation of sets S_{rs} and $S_{r\bar{s}}$. Furthermore, by invoking the same argument for every inequality of H that violates the non-negativity requirement, we can obtain a linear separator H' that (i) employs the same number of inequalities, k, and (ii) it has non-negative coefficients for all the inequalities involved; hence, this is the sought separator.

In a spirit similar to that underlying the development of Equation 11 above, the existence of the aforementioned hyperplane $h'_i = (\mathbf{a}'_i, b'_i)$ is equivalent to the existence of a $\hat{\hat{\delta}} > 0$ such that for every $\delta \in (0, \hat{\hat{\delta}}]$, the following system of equations in $\mathbf{a'_i}$ and b'_i is feasible:

$$\begin{bmatrix} \mathbf{x_1}^T & -1 \\ \mathbf{x_2}^T & -1 \\ \cdots & \cdots \\ \mathbf{x_{m_s}}^T & -1 \\ -\mathbf{y_1}^T & 1 \\ -\mathbf{y_2}^T & 1 \\ \cdots & \cdots \\ -\mathbf{y_{m_{u_i}}}^T & 1 \end{bmatrix} \begin{bmatrix} (\mathbf{a}'_i)^T \\ b'_i \end{bmatrix} \leq \begin{bmatrix} 0 \\ 0 \\ \cdots \\ 0 \\ -\delta \\ -\delta \\ \cdots \\ -\delta \end{bmatrix} \wedge \begin{bmatrix} (\mathbf{a}'_i)^T \\ b'_i \end{bmatrix} \geq \mathbf{0} \quad (14)$$

We proceed to establish the existence of the aforementioned $\hat{\hat{\delta}}$ and the feasibility of the systems of inequalities expressed by Equation 14 through contradiction. For this, we assume that the system of inequalities represented by Equation 14 is infeasible for any $\delta \to 0^+$. Then, Farkas' lemma [22] implies that the following system of inequalities in the variable vector $\mathbf{p}$ must be feasible:

$$\mathbf{p} \geq \mathbf{0} \wedge \mathbf{p}^T \begin{bmatrix} \mathbf{x_1}^T & -1 \\ \mathbf{x_2}^T & -1 \\ \cdots & \cdots \\ \mathbf{x_{m_s}}^T & -1 \\ -\mathbf{y_1}^T & 1 \\ -\mathbf{y_2}^T & 1 \\ \cdots & \cdots \\ -\mathbf{y_{m_{u_i}}}^T & 1 \end{bmatrix} \geq \mathbf{0} \wedge \mathbf{p}^T \begin{bmatrix} 0 \\ 0 \\ \cdots \\ 0 \\ -\delta \\ -\delta \\ \cdots \\ -\delta \end{bmatrix} < 0 \quad (15)$$

Selecting $\delta \in (0, \hat{\delta})$, and considering the infeasiblity of Equation 13, Equation 15 reduces to

$$\mathbf{p} \geq \mathbf{0} \wedge \mathbf{p}^T \begin{bmatrix} \mathbf{x_1}^T \\ \mathbf{x_2}^T \\ \cdots \\ \mathbf{x_{m_s}}^T \\ -\mathbf{y_1}^T \\ -\mathbf{y_2}^T \\ \cdots \\ -\mathbf{y_{m_{u_i}}}^T \end{bmatrix} > \mathbf{0} \wedge \mathbf{p}^T \begin{bmatrix} -1 \\ -1 \\ \cdots \\ -1 \\ 1 \\ 1 \\ \cdots \\ 1 \end{bmatrix} \geq 0 \wedge \mathbf{p}^T \begin{bmatrix} 0 \\ 0 \\ \cdots \\ 0 \\ -\delta \\ -\delta \\ \cdots \\ -\delta \end{bmatrix} < 0 \quad (16)$$

where the notation '$>$' in the second component of the above statement implies that at least one of the inequalities involved in that component is strict.

For the rest of the proof, let $p_j \equiv \mathbf{p}[j],\ j = 1, \ldots, m_s$, and $\tilde{p}_j \equiv \mathbf{p}[m_s + j],\ j = 1, \ldots, m_{u_i}$. Then, the feasibility of Equation 16 can be restated as follows:

$$\begin{aligned} \exists \mathbf{p}: \quad & \mathbf{p} \geq \mathbf{0} & \wedge \quad (17) \\ & \textstyle\sum_{j=1}^{m_s} p_j \mathbf{x_j} - \sum_{j=1}^{m_{u_i}} \tilde{p}_j \mathbf{y_j} > \mathbf{0} & \wedge \quad (18) \\ & \textstyle -\sum_{j=1}^{m_s} p_j + \sum_{j=1}^{m_{u_i}} \tilde{p}_j \geq 0 & \wedge \quad (19) \\ & \textstyle -\delta \sum_{j=1}^{m_{u_i}} \tilde{p}_j < 0 & (20) \end{aligned}$$

Let $l_1 \equiv \sum_{j=1}^{m_s} p_j$ and $l_2 \equiv \sum_{j=1}^{m_{u_i}} \tilde{p}_j$. Equation 20 implies that

$$l_2 > 0 \quad (21)$$

Furthermore, the combination of Equations 17, 18 and 21 also implies that

$$l_1 > 0 \quad (22)$$

Finally, notice that Equation 19 implies that

$$l_2 \geq l_1 \quad (23)$$

But then, Equation 18 can be rewritten as

$$\sum_{j=1}^{m_s} \frac{p_j}{l_1} \mathbf{x_j} > \frac{l_2}{l_1} \sum_{j=1}^{m_{u_i}} \frac{\tilde{p}_j}{l_2} \mathbf{y_j} \quad (24)$$

and when combined with Equation 23, it further implies that

$$\sum_{j=1}^{m_s} \frac{p_j}{l_1}\mathbf{x_j} \;>\; \sum_{j=1}^{m_{u_i}} \frac{\tilde{p}_j}{l_2}\mathbf{y_j} \tag{25}$$

Next we show that the inequality of Equation 25 contradicts the working assumptions. Therefore, we can conclude that $\exists\ \delta > 0$ such that the system of inequalities in Equation 14 is feasible, and our proof is completed. To see the infeasibility of Equation 25, first notice that by the definition of the vectors $\mathbf{x_j},\ j = 1,\ldots,m_s$, and of the scalar quantity l_1, the vector $\sum_{j=1}^{m_s} \frac{p_j}{l_1}\mathbf{x_j}$ belongs in the convex hull of S_{rs}. Furthermore, the non-negativity of the vector $\sum_{j=1}^{m_{u_i}} \frac{\tilde{p}_j}{l_2}\mathbf{y_j}$, combined with Equation 25 and Corollary 1, imply that $\sum_{j=1}^{m_{u_i}} \frac{\tilde{p}_j}{l_2}\mathbf{y_j} \in Conv(S_{rs})$, as well. But this cannot be possible since, by the definition of l_2, the vector $\sum_{j=1}^{m_{u_i}} \frac{\tilde{p}_j}{l_2}\mathbf{y_j}$ belongs also in the convex hull of the set of points $\{\mathbf{y}_j,\ j = 1,\ldots,m_{u_i}\}$, and this set is linearly separable from S_{rs} (by the hyperplane $h_i = (\mathbf{a}_i, b_i)$, considered at the beginning of the proof). □

We exemplify the set-"thinning" step established by Propositions 3 and 4, through the middle part of Table II, which provides the sets $\bar{S}_{rs}$ and $\bar{S}^b_{r\bar{s}}$ that are obtained by the application of this step to the sets of safe states, S_{rs}, and boundary unsafe states, $S^b_{r\bar{s}}$, corresponding to the Gadara RAS of Table I.

Converting the separation problem of $\bar{S}_{rs}$ and $\bar{S}^b_{r\bar{s}}$ to an equivalent separation problem of reduced dimensionality The "thinning" of the original state sets S_{rs} and $S_{r\bar{s}}$ to their state subsets $\bar{S}_{rs}$ and $\bar{S}^b_{r\bar{s}}$, that was discussed in the previous paragraphs, can have the additional effect that one or more of the components of the vectors belonging in the "thinned" sets $\bar{S}_{rs}$ and $\bar{S}^b_{r\bar{s}}$ is constant. More specifically, in our numerical experimentation we have consistently encountered a situation where many components of the vectors included in the set $\bar{S}^b_{r\bar{s}}$ are identically zero. Next, we show that the removal of these components from further consideration, through the orthogonal projection of the vector sets $\bar{S}_{rs}$ and $\bar{S}^b_{r\bar{s}}$ to the subspace defined by the remaining components, retains all the information that is necessary for the development of a minimal linear separator that will separate effectively the original state subsets S_{rs} and $S_{r\bar{s}}$. To formalize the subsequent discussion, let V denote the ξ-dimensional vector space supporting the vector sets $\bar{S}_{rs}$ and $\bar{S}^b_{r\bar{s}}$, I denote the set of coordinates of space V, I_0 denote the set of coordinates that are identically zero in the vectors of $\bar{S}^b_{r\bar{s}}$, and therefore, they will be removed by the proposed projection P, $I_P \equiv I \backslash I_0$, and V_P denote the $|I_P|$-dimensional sub-space supporting the projection P. Also, let $\Gamma : \mathbb{N} \to \mathbb{N}$ be a bijection that maps the elements of the coordinate set I_P to the coordinates of subspace V_P. Finally, let $P(\bar{S}_{rs})$ and $P(\bar{S}^b_{r\bar{s}})$ denote respectively the images of the sets $\bar{S}_{rs}$ and $\bar{S}^b_{r\bar{s}}$ under P. Then, we have the following proposition:

Proposition 5: There exists a set of k hyperplanes, Q, that separates the projected sets $P(\bar{S}_{rs})$ and $P(\bar{S}^b_{r\bar{s}})$ in subspace V_P, iff there exists a set of k hyperplanes, H, that separates the sets $\bar{S}_{rs}$ and $\bar{S}^b_{r\bar{s}}$ in the original space V.

Proof: First we show that the existence of a linear separator Q with k inequalities for the projected sets $P(\bar{S}_{rs})$ and $P(\bar{S}^b_{r\bar{s}})$ in subspace V_P, implies the existence of a separator H with the same number of inequalities that separates the sets $\bar{S}_{rs}$ and $\bar{S}^b_{r\bar{s}}$ in the original space V. Let $q_i = (\mathbf{a}_{q_i}, b_{q_i})$ be an arbitrary hyperplane of Q, and denote by $P(\bar{S}^b_{r\bar{s}})^{(i)}$ the set of points in $P(\bar{S}^b_{r\bar{s}})$ separated by q_i; i.e.,

$$\forall \mathbf{x} \in P(\bar{S}_{rs}) :\ \mathbf{a}_{q_i}\cdot\mathbf{x} \;\le\; b_{q_i} \;\wedge\; \forall \mathbf{y} \in P(\bar{S}^b_{r\bar{s}})^{(i)} :\ \mathbf{a}_{q_i}\cdot\mathbf{y} > b_{q_i} \tag{26}$$

Also, let $(\bar{S}^b_{r\bar{s}})^{(i)} \subseteq \bar{S}^b_{r\bar{s}}$ be the set of states in $\bar{S}^b_{r\bar{s}}$ with their projection being in the set $P(\bar{S}^b_{r\bar{s}})^{(i)}$. To prove our case, it suffices to show that there exists a hyperplane h_i in the original space V that separates $(\bar{S}^b_{r\bar{s}})^{(i)}$ from $\bar{S}_{rs}$. This hyperplane $h_i = (\mathbf{a}_{h_i}, b_{h_i})$ can be constructed as follows:

$$b_{h_i} := b_{q_i} \;\wedge\; \forall j \in I_0 :\ \mathbf{a}_{h_i}[j] := 0 \;\wedge\; \forall j \in I_P :\ \mathbf{a}_{h_i}[j] := \mathbf{a}_{q_i}[\Gamma(j)] \tag{27}$$

Indeed, we can see that $\forall \mathbf{x} \in \bar{S}_{rs}$,

$$\mathbf{a}_{h_i}\cdot\mathbf{x} = \sum_{j\in I} \mathbf{a}_{h_i}[j]\cdot\mathbf{x}[j] = \sum_{j\in I_p} \mathbf{a}_{h_i}[j]\cdot\mathbf{x}[j] = \mathbf{a}_{q_i}\cdot\mathbf{x}_p \;\le\; b_{q_i} = b_{h_i} \tag{28}$$

where $\mathbf{x}_p$ is the image of $\mathbf{x}$ in subspace V_P. Similarly we have $\forall \mathbf{y} \in (\bar{S}^b_{r\bar{s}})^{(i)}$,

$$\mathbf{a}_{h_i}\cdot\mathbf{y} = \sum_{j\in I} \mathbf{a}_{h_i}[j]\cdot\mathbf{y}[j] = \sum_{j\in I_p} \mathbf{a}_{h_i}[j]\cdot\mathbf{y}[j] = \mathbf{a}_{q_i}\cdot\mathbf{y}_p > b_{q_i} = b_{h_i} \tag{29}$$

where $\mathbf{y}_p$ is the image of $\mathbf{y}$ in subspace V_P. Therefore h_i separates $(\bar{S}^b_{r\bar{s}})^{(i)}$ from $\bar{S}_{rs}$ and the forward part of Proposition 5 is proved.

Next, we show the validity of the reverse part of the proposition, i.e., that the existence of a linear separator H with k inequalities that separates the sets $\bar{S}_{rs}$ and $\bar{S}^b_{r\bar{s}}$ in the original space V, implies the existence of a linear separator Q with the same number of inequalities that separates the projected sets $P(\bar{S}_{rs})$ and $P(\bar{S}^b_{r\bar{s}})$ in subspace V_P. To prove this result, we first notice that Proposition 4 implies that if there exists a linear separator H with k inequalities that separates the sets $\bar{S}_{rs}$ and $\bar{S}^b_{r\bar{s}}$ in the original space V, then there also exists a linear separator H' with k inequalities and non-negative coefficients that separates the same two sets in V. So, in the following we shall focus on such a separator H'. The rest of the proof proceeds similarly to the proof provided for the forward part of the proposition, but it also relies on the fact that

$$\forall s \in \bar{S}^b_{r\bar{s}},\ \forall j \in I_0 :\ \mathbf{s}[j] = 0 \tag{30}$$

Let $h_i = (\mathbf{a}_{h_i}, b_{h_i}) \in H'$ and denote by $(\bar{S}^b_{r\bar{s}})^{(i)}$ the set of points in $(\bar{S}^b_{r\bar{s}})$ separated by h_i. Then,

$$\forall \mathbf{x} \in \bar{S}_{rs} :\ \mathbf{a}_{h_i}\cdot\mathbf{x} \;\le\; b_{h_i} \;\wedge\; \forall \mathbf{y} \in (\bar{S}^b_{r\bar{s}})^{(i)} :\ \mathbf{a}_{h_i}\cdot\mathbf{y} > b_{h_i} \tag{31}$$

Also, let $P(\bar{S}^b_{r\bar{s}})^{(i)} \subseteq P(\bar{S}^b_{r\bar{s}})$ be the projection of $(\bar{S}^b_{r\bar{s}})^{(i)}$ on subspace V_P. We need to show that there exists a hyperplane q_i in subspace V_P which separates $P(\bar{S}^b_{r\bar{s}})^{(i)}$ from $P(\bar{S}_{rs})$. This hyperplane $q_i = (\mathbf{a}_{q_i}, b_{q_i})$ can be constructed as follows:

$$b_{q_i} := b_{h_i} \;\wedge\; \forall j \in I_P :\ \mathbf{a}_{q_i}[\Gamma(j)] = \mathbf{a}_{h_i}[j] \tag{32}$$

Then, we can see that

$$\forall \mathbf{x}_p \in P(\bar{S}_{rs}) :\ \mathbf{a}_{q_i}\cdot\mathbf{x}_p = \sum_{j\in I_P} \mathbf{a}_{h_i}[j]\cdot\mathbf{x}[j] \;\le\; \mathbf{a}_{h_i}\cdot\mathbf{x} \;\le\; b_{h_i} = b_{q_i} \tag{33}$$

where $\mathbf{x}$ is an element in the pre-image of $\mathbf{x}_p$ in the original space V, and the first inequality holds true because of the non-negativity of the coefficients of the separator H'. Similarly, we can see that

$$\forall \mathbf{y}_p \in P(\bar{S}^b_{r\bar{s}})^{(i)} : \ \mathbf{a}_{q_i} \cdot \mathbf{y}_p = \sum_{j \in I_p} \mathbf{a}_{h_i}[j] \cdot \mathbf{y}[j] = \mathbf{a}_{h_i} \cdot \mathbf{y} \ > \ b_{h_i} = b_{q_i} \tag{34}$$

where $\mathbf{y}$ is the pre-image of $\mathbf{y_p}$ in the original space V, and the second equality above holds true because of Equation 30. Therefore, q_i separates $P(\bar{S}^b_{r\bar{s}})^{(i)}$ from $P(\bar{S}_{rs})$, and the proof is complete. □

As remarked at the beginning of this paragraph, the practical implication of Proposition 5 is that we can construct a minimal linear separator H for the sets $\bar{S}^b_{r\bar{s}}$ and $\bar{S}_{rs}$ by first developing a linear separator Q for the projected sets $P(\bar{S}^b_{r\bar{s}})$ and $P(\bar{S}_{rs})$, and subsequently constructing H from Q through Equation 27, which is repeated here for emphasis and convenience:

$$\begin{aligned} & b_{h_i} := b_{q_i} \ \wedge \\ \forall j \in I_0 : \ \mathbf{a}_{h_i}[j] := 0 \ \ \wedge \ \ \forall j \in I_P : \ \mathbf{a}_{h_i}[j] := \mathbf{a}_{q_i}[\Gamma(j)] \end{aligned} \tag{35}$$

Furthermore, the second part of the proof of Proposition 5 guarantees the we can have a *minimal* linear separator Q for the projected sets $P(\bar{S}^b_{r\bar{s}})$ from $P(\bar{S}_{rs})$ with *non-negative* coefficients.

Some further simplifications We close this section by discussing some further simplifications for the problem of the construction of the linear separator Q mentioned above, that result from the structure of the set $P(\bar{S}_{rs})$, i.e., the projection of the set $\bar{S}_{rs}$ to subspace V_P. The first of these simplifications has to do with the fact that the coordinates removed from the elements of set $\bar{S}_{rs}$ by the introduced projection P can have a value of either zero or one. Hence, projection P is not bijective for the set $\bar{S}_{rs}$, and this fact generates some redundancy in the data structure representing the projected set $P(\bar{S}_{rs})$, that must be systematically identified and removed. The second simplification arises from the fact that the removal of the coordinates in I_0 can introduce some dominance among the elements of $P(\bar{S}_{rs})$ with respect to the ordering '$\leq$'. To alleviate the computational complexity of the construction of the target separator Q, any non-maximal elements in $P(\bar{S}_{rs})$ should be identified and removed from this set; the resulting set will be denoted by $\overline{P(\bar{S}_{rs})}$.[5] The last part of Table II presents also the various vector sets that are obtained from the application of the projection P discussed in this paragraph on the relevant state sets for the RAS of Table I, and the ensuing "thinning" of the vector set $P(\bar{S}_{rs})$ that results from the projection of the set of maximal reachable and safe states, $\bar{S}_{rs}$.

Figure 2 summarizes the data "thinning" process described in the previous paragraphs, providing a flowchart of the entire workflow that is necessary for the development of a minimal linear separator for the safe and unsafe subspaces of a Gadara RAS. Also, as already discussed, Table II demonstrates the results that are obtained by the application of this "thinning" process to the sets S_{rs} and $S_{r\bar{s}}$ corresponding to the safe and unsafe subspaces of the Gadara RAS of Table I. The detailed algorithms that will support the pre-processing stages that are depicted in Figure 2 and precede the actual construction of the separator Q, are rather straightforward and well-established in the relevant literature. For example, the first stage depicted in Figure 2, involving the state space enumeration, can be easily supported by any "search"-type algorithm [23] that starts from the initial state and processes one state at a time, "reaching out" to new states according to the logic established in Section II. Similarly, the state classification involved in the second stage of the depicted flowchart can be easily performed by applying on the state transition diagram obtained from the first stage, any algorithm that assesses state co-accessibility w.r.t. the initial state s_0 [2]. Also, the remaining steps involve elementary operations on the extracted sets and their vector elements. Therefore, all these algorithms are omitted for the sake of brevity. The next two sections focus on the support of the last step depicted in Figure 2, i.e., the construction of the linear separator Q from the input sets $\overline{P(\bar{S}_{rs})}$ and $P(\bar{S}^b_{r\bar{s}})$.

TABLE II

The various sets obtained by the application of the data "thinning" process of Figure 2 to the sets S_{rs} and $S_{r\bar{s}}$ corresponding to the reachable safe and unsafe subspaces of the Gadara RAS of Table I.

$S^b_{r\bar{s}} \ = \ S_{r\bar{s}}$
$\bar{S}_{rs} \ = \ \{q^{13} \equiv [1\ 1\ 1\ 0\ 0\ 0]^T,\ q^{14} \equiv [0\ 0\ 0\ 1\ 1\ 1]^T\}$
$\bar{S}^b_{r\bar{s}} \ = \ \{q^{15} \equiv [1\ 0\ 0\ 1\ 0\ 0]^T,\ q^{16} \equiv [0\ 1\ 0\ 1\ 0\ 0]^T,$
$\quad q^{17} \equiv [1\ 0\ 0\ 0\ 1\ 0]^T\}$
$I_0 \ = \ \{3,\ 6\}$
$P(\bar{S}^b_{r\bar{s}}) \ = \ \{[1\ 0\ 1\ 0]^T,\ [0\ 1\ 10]^T,\ [1\ 0\ 0\ 1]^T\}$
$P(\bar{S}_{rs}) \ = \ \{[1\ 1\ 0\ 0]^T,\ [0\ 0\ 1\ 1]^T\} \ = \ \overline{P(\bar{S}_{rs})}$

IV. Synthesizing classifier Q through Mathematical Programming

The MIP formulation In this section we provide a Mixed Integer Programming (MIP) formulation [24] for the construction of the separator Q, that was specified in Section III. We remind the reader that the primary inputs to this final stage of the proposed design process, are:

- the elements $\mathbf{x}_i$ of the projected safe state set $\overline{P(\bar{S}_{rs})}$, and
- the elements $\mathbf{y}_i$ of the projected unsafe state set $P(\bar{S}^b_{r\bar{s}})$.

In the subsequent discussion, we shall set $m_s \equiv |\overline{P(\bar{S}_{rs})}|$, $m_u \equiv |P(\bar{S}^b_{r\bar{s}})|$, and $n \equiv |I_P|$, i.e., n denotes the dimensionality of the subspace V_P supporting the vectors $\mathbf{x}_i$ and $\mathbf{y}_i$. Some additional inputs that parameterize this last stage of our design process and provide additional controls to it, are as follows:

- The parameter w which provides an upper bound for the "size" of – i.e., the number of inequalities employed by – the sought separator. Such an upper bound is readily obtained as $w = m_u$ from Proposition 2 and Corollary 2 of Section II when combined with the results of Proposition 4 and Equation 35 of Section III. A tighter value for w can be effectively computed through the heuristic discussed in the next section.
- A strictly positive parameter ε that controls the minimum distance of the points $\mathbf{y}_i$ from the separating hyperplanes and

[5] The fact that this additional "thinning" of the set $P(\bar{S}_{rs})$ does not compromise the effectiveness of the obtained separator Q with respect to the separation of the sets $P(\bar{S}_{rs})$ and $P(\bar{S}^b_{r\bar{s}})$, can be argued on the basis of the non-negativity of the coefficients of the target separator Q; cf. Proposition 3.

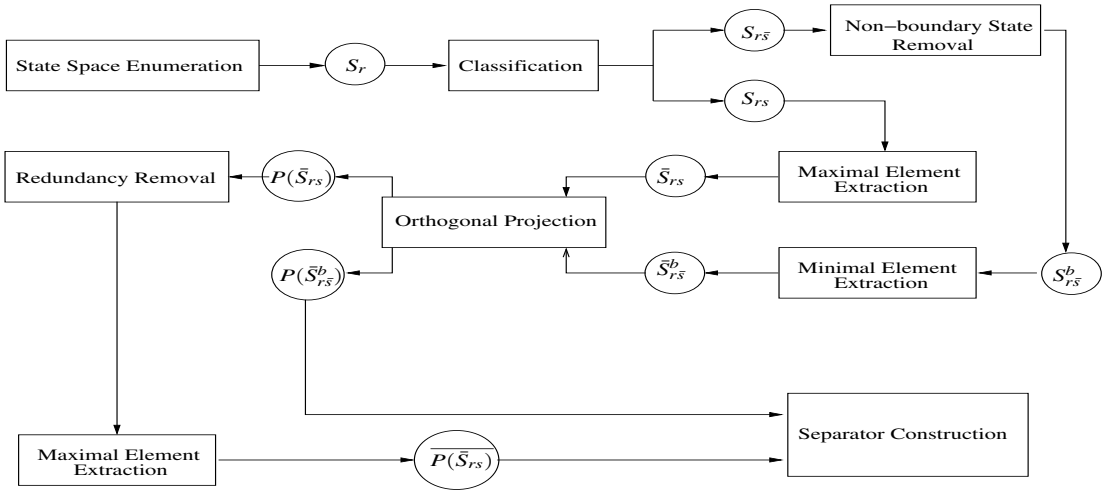

Fig. 2. The proposed workflow for the construction of a minimal linear separator for the reachable safe and unsafe subspaces of a Gadara RAS.

should be priced sufficiently close to zero. This parameter can be perceived as a "degree of separation" that is enforced between the two sets $\overline{P(\bar{S}_{rs})}$ and $P(\bar{S}^b_{r\bar{s}})$. In order to guarantee that the employed value of ε is not unnecessarily large to the extent that it compromises the minimality of the derived solution, one should re-solve the proposed formulation for a sequence $\{\varepsilon_i\}$ such that $\varepsilon_i \rightarrow 0^+$, and consider the stability of the size of the obtained supervisors.

- A strictly positive parameter M that is useful for the modelling of the separation logic in the proposed MIP formulation and must take sufficiently large values. This is the notorious "big-M" parameter that appears in many MIP formulations. A more detailed discussion on its role in the proposed formulation, as well as on its appropriate pricing, is provided in later parts of this section.

The variables employed by the proposed formulation are as follows:

- z_l, $l = 1,\ldots,w$, is a binary variable that is priced to one if the l-th inequality is used for separation and to zero otherwise.
- $(\mathbf{A}[l,.],\mathbf{b}[l])$, $l = 1,\ldots,w$, are the coefficients to be employed by the l-th separating linear inequality.
- δ_{il}, $i = 1,\ldots,m_u$, $l = 1,\ldots,w$, is a binary variable that must be priced to one if the state $\mathbf{y}_i$ and the hyperplane $(\mathbf{A}[l,.],\mathbf{b}[l])$ satisfy the inequality $\mathbf{A}[l,.]\cdot\mathbf{y}_i \leq \mathbf{b}[l]+\varepsilon$.

Finally, the formulation itself takes the following form:

$$\min \sum_{l=1}^{w} z_l \tag{36}$$

$$\forall i \in \{1,\cdots,m_s\},\ \forall l \in \{1,\cdots,w\}:\ \mathbf{A}[l,.]\cdot\mathbf{x}_i - \mathbf{b}[l]\ \leq\ 0 \tag{37}$$

$$\forall i \in \{1,\cdots,m_u\},\ \forall l \in \{1,\cdots,w\}:\ \mathbf{A}[l,.]\cdot\mathbf{y}_i - \mathbf{b}[l] + M\cdot\delta_{il} \geq\ \varepsilon \tag{38}$$

$$\forall i \in \{1,\cdots,m_u\}:\ \sum_{l=1}^{w} \delta_{il}\ \leq\ w-1 \tag{39}$$

$$\forall l \in \{1,\cdots,w\},\ \forall j \in \{1,\cdots,n\}:\ 0\ \leq\ \mathbf{A}[l,j]\ \leq\ z_l \tag{40}$$

$$\forall i \in \{1,\cdots,m_u\},\ \forall l \in \{1,\cdots,w\}:\ \delta_{il} \in \{0,1\} \tag{41}$$

$$\forall l \in \{1,\cdots,w\}:\ z_l \in \{0,1\} \tag{42}$$

The validity of the above MIP formulation as a construction tool for the sought separator Q can be established as follows: First, the reader should notice that Equation 36 defines the objective of the formulation as the minimization of the number of hyperplanes that will be actively used for the pursued separation. Also, Equations 41 and 42 respectively enforce the binary nature of the variables δ_{il} and z_l. On the other hand, the constraints of Equation 40 enforce (i) the non-negativity of the matrix $\mathbf{A}$ in the returned solution, and also (ii) the requirement that the coefficients of inactive inequalities should be set to zero. An additional implication of the constraints expressed by the right inequality in Equation 40, is the restriction of all the elements of matrix $\mathbf{A}$ to values no greater than one. This effect does not compromise the generality of the obtained solution, since the values of the intercepts $\mathbf{b}[l]$, $l = 1,\ldots,w$, are not restricted by this constraint.[6] On the other hand, we shall see in the following discussion that the restriction of the elements of matrix $\mathbf{A}$ in the interval $[0,1]$ enables the effective resolution of some other aspects of the formulation.

The separation logic is primarily expressed by the constraints of Equations 37–39. More specifically, the constraints of Equation 37 force every point corresponding to a safe vector $\mathbf{x}_i$, $i = 1,\ldots,m_s$, to lie below each separating hyperplane. When combined with the non-negativity of vectors $\mathbf{x}_i$ and of matrix $\mathbf{A}$, this constraint also enforces the non-negativity of the intercepts $\mathbf{b}[l]$. On the other hand, the constraints of Equations 38 and 39 require that every point corresponding to an unsafe vector $\mathbf{y}_i$, $i = 1,\ldots,m_u$, lies above of at least one of the separating hyperplanes. To understand the detailed mechanism that enforces this requirement, first notice that for any vector $\mathbf{y_i}$ and inequality $(\mathbf{A}[l,.],\mathbf{b}[l])$ such that $\mathbf{A}[l,.]\cdot\mathbf{y}_i \geq \mathbf{b}[l]+\varepsilon$, Equation 38 is

[6] In other words, every derived inequality $(\mathbf{A}[l,.],\mathbf{b}[l])$ can have its coefficients normalized by the maximum element of the row vector $\mathbf{A}[l,.]$ and this normalization will not affect its informational content.

satisfied irrespective of the value of the binary variable δ_{il} (provided that $M \geq 0$). In the opposite case, the corresponding variable δ_{il} must be set to one, in order to attain the satisfaction of Equation 38 (provided that the non-negative parameter M is sufficiently large). But at the same time, Equation 39 requires that, for every point $\mathbf{y}_i$, at least one of the corresponding variables δ_{il}, $l = 1, \ldots, w$, must be equal to zero. Therefore, in any feasible solution of the proposed formulation, every point $\mathbf{y}_i$ must be above at least one of the hyperplanes $(\mathbf{A}[l,.], \mathbf{b}[l])$.

The above discussion also reveals the condition for the proper pricing of the parameter M:

$$M \geq \sup\{[\mathbf{A}[l,.] \cdot \mathbf{y}_i - \mathbf{b}[l] - \varepsilon]^-\} \tag{43}$$

where $[a]^- \equiv |\min\{0, a\}|$ and the supremum is taken over all pairs of vectors $\mathbf{y_i}$ and inequalities $(\mathbf{A}[l,.], \mathbf{b}[l])$ in any viable solution. An upper bound for the quantity on the right-hand-side of Equation 43 can be obtained by setting $\mathbf{A}[l,.] \cdot \mathbf{y}_i = 0$, and considering an upper bound for $\sup\{\mathbf{b}[l] + \varepsilon\}$, where the latter is taken over all the inequalities that can constitute a viable component of the sought separator Q. To obtain this last upper bound, we recall that the minimality of Q implies that, for every constituent inequality $(\mathbf{A}[l,.], \mathbf{b}[l])$, there exists at least one point $\mathbf{y}_j$ such that $\mathbf{A}[l,.] \cdot \mathbf{y}_i \geq \mathbf{b}[l] + \varepsilon$. But then, Equations 40 and 42, together with the binary nature of vectors $\mathbf{y}_i$, further imply that $\sup\{\mathbf{b}[l] + \varepsilon\} \leq n$, where n denotes the dimensionality of the supporting hyperspace V_P. Hence, M can be set equal to n during the solution of the considered formulation.

The next theorem provides a formal statement of the validity of the MIP formulation of Equations 36–42 as a classifier design tool for the classification problem considered in this work, and it is an immediate implication of all the above discussion provided in this section and of the developments of Section III.

Theorem 1: The application of the formulation of Equations 36–42 to the sets $\overline{P(\bar{S}_{rs})}$ and $P(\bar{S}^b_{r\bar{s}})$ corresponding to any given Gadara RAS Φ, returns a minimal linear classifier for these two sets, and through Equation 35, a minimal linear classifier for the original sets S_{rs} and $S^b_{r\bar{s}}$.

Example The application of the formulation of Equations 36–42 to the sets $\overline{P(\bar{S}_{rs})}$ and $P(\bar{S}^b_{r\bar{s}})$ provided in Table I, while setting $\varepsilon = 0.01$, resulted in a linear classifier that is expressed by the following two inequalities:

$$\mathbf{s}[1] + \mathbf{s}[5] \leq 1.0$$

$$0.01 \cdot \mathbf{s}[1] + 0.99 \cdot \mathbf{s}[2] + \mathbf{s}[4] \leq 1.0$$

Indeed, the reader can verify that this system of inequalities is satisfied by every vector in $\overline{P(\bar{S}_{rs})}$ and it is violated by every vector in $P(\bar{S}^b_{r\bar{s}})$. But then, the previous developments in this manuscript imply that the one-step-lookahead policy defined by the above two inequalities is an effective implementation of the maximally permissive DAP for the RAS of Table I. Furthermore, by its construction, this implementation is minimal, i.e., it uses the minimum possible number of linear inequalities that can represent effectively the maximally permissive DAP for the considered RAS.

Complexity considerations It should be clear from the above discussion that the MIP formulation of Equations 36–42 involves $(m_u + 1) \cdot w$ binary variables, $(|I_P| + 1) \cdot w$ real variables and $w \cdot (m_s + m_u + |I_P|) + m_u$ technological constraints.[7] Hence, the size of this formulation, in terms of the variables and constraints involved, is polynomially related to the size of the classified sets $\overline{P(\bar{S}_{rs})}$ and $P(\bar{S}^b_{r\bar{s}})$ and the dimensionality of their supporting sub-space V_P. In Section VI we provide a set of computational results that demonstrate the tractability of the MIP formulation of Equations 36–42 for RAS instantiations from the Gadara RAS class with size comparable to those encountered in many practical applications. These results make us believe that the MIP formulation of Equations 36–42 will be an effective computational tool for the synthesis of minimal linear separators Q for a very broad range of the Gadara RAS instantiations to be encountered in "real-life" applications.

Yet, a general MIP formulation is always an expensive (non-polynomial complexity) proposition from a computational standpoint. Furthermore, m_s and m_u, i.e., the cardinalities of the sets $\overline{P(\bar{S}_{rs})}$ and $P(\bar{S}^b_{r\bar{s}})$, in general will grow super-polynomially with respect to the size of the underlying RAS Φ. Therefore, in the next section, we present a heuristic that can provide an effective trade-off between the structural minimality of the obtained separator, as measured by the number of the involved inequalities, and the computational effort for its development. We also show that the relative size of the separators returned by this heuristic to the size of any minimal separator is no higher than $\ln|P(\bar{S}^b_{r\bar{s}})|$. Therefore, the classifiers provided by this heuristic are still quite compact and computationally efficient.[8]

V. An efficient heuristic for the synthesis of classifier Q

The main idea that underlies the heuristic proposed in this section is to construct the sought separator Q one hyperplane at a time; in particular, at each iteration we consider the set of points in $P(\bar{S}^b_{r\bar{s}})$ that have not been separated yet from the set $\overline{P(\bar{S}_{rs})}$ by any of the constructed hyperplanes, and we try to identify a hyperplane that will separate the maximum possible number of these points from $\overline{P(\bar{S}_{rs})}$. Corollary 2 of Section II together with Proposition 4 of Section III imply that in the case of the Gadara RAS, which is the focus of this work, this iterative procedure will terminate in a finite number of iterations.

Next we present a MIP formulation that can support the computation of the hyperplane sought at each of the iterations described in the previous paragraph. The input data for this formulation are:

- the elements $\mathbf{x}_i$ of the projected safe state set $\overline{P(\bar{S}_{rs})}$;
- the elements $\mathbf{y}_i$ of the projected unsafe state set $P(\bar{S}^b_{r\bar{s}})$ that remain unseparated in the current iteration;
- strictly positive parameters ε and M that play a role similar to that played by the corresponding parameters in the MIP formulation of Equations 36–42.

Also, similar to the MIP formulation of Equations 36–42, we let $m_s \equiv |\overline{P(\bar{S}_{rs})}|$, $m_u \equiv |\{\mathbf{y}_i\}|$, and $n \equiv |I_P|$. The variables em-

[7] By technological constraints we mean all the formulation constraints except from those that impose the nonnegative and the binary nature of the various variables. These are the constraints that are explicitly considered in the computations performed by any solution algorithm for the MIP formulation.

[8] Actually, the heuristic of Section V can also assist the solution of the MIP formulation of Equations 36–42 itself, e.g., by providing additional constraints in the form of incumbent solutions to this formulation.

ployed by this new formulation are as follows:

- $(\mathbf{a}, b)$ are the coefficients of the generated hyperplane.
- δ_i, $i = 1, \ldots, m_u$, is a binary variable that is priced to one *iff* the state $\mathbf{y}_i$ and the generated hyperplane satisfy the inequality $\mathbf{a} \cdot \mathbf{y}_i \geq b + \varepsilon$ (i.e., point $\mathbf{y}_i$ is separated by the generated hyperplane).

The formulation itself is expressed by the following equations:

$$\max \sum_{i=1}^{m_u} \delta_i \tag{44}$$

$$\forall i \in \{1, \cdots, m_s\} : \ \mathbf{a} \cdot \mathbf{x}_i - b \leq 0 \tag{45}$$

$$\forall i \in \{1, \cdots, m_u\} : \ \mathbf{a} \cdot \mathbf{y}_i - b + (1 - \delta_i) \cdot M \geq \varepsilon \tag{46}$$

$$\forall j \in \{1, \cdots, n\} : \ 0 \leq \mathbf{a}[j] \leq 1 \tag{47}$$

$$\forall i \in \{1, \cdots, m_u\} : \ \delta_i \in \{0, 1\} \tag{48}$$

Equation 44 expresses the objective of the considered formulation as the maximization of the number of the unsafe points that will be separated by the generated hyperplane. Equation 45 forces all the safe state vectors $\mathbf{x}_i$ to lie below the generated hyperplane. On the other hand, Equation 46, in collaboration with Equation 44, enforces the correct pricing of the indicator variables δ_i, and therefore, it ensures the correct evaluation of the objective function for any given pricing of the design variables $(\mathbf{a}, b)$. The detailed mechanics of this enforcement are as follows: For a given unsafe state $\mathbf{y}_i$, if $\mathbf{a} \cdot \mathbf{y}_i - b < \varepsilon$, the corresponding variable δ_i is forced to zero. On the other hand, if $\mathbf{a} \cdot \mathbf{y}_i - b \geq \varepsilon$, Equation 46 allows δ_i to take any of its two possible values, but the objective stated in Equation 44 will force δ_i to one, in any optimal solution. Equation 47 constrains the generated hyperplanes to have non-negative coefficients, and it also enforces a normalization of these coefficients similar to the normalization performed by the formulation of Equations 36–42. Equation 48 enforces the binary nature of the variables δ_i. Finally, we notice that for reasons similar to those explained in the discussion of the formulation of Equations 36–42, the parameter M of Equation 46 can be safely set equal to n during the solution of the formulation.

The formulation of Equations 44–48 employs $(|I_P| + 1)$ real and m_u binary variables in $(m_s + m_u + |I_P|)$ technological constraints; therefore, the computational effort required for its solution is expected to be much smaller than the corresponding effort required for the solution of the exact formulation of Equations 36-42. Our computational results reveal that the relevant gains are so substantial, that the heuristic proposed in this section will tend to run much faster than the algorithm that solves the exact formulation, in spite of the fact that the formulation of Equations 44–48 is solved repeatedly in the context of this heuristic.

The complete algorithm that utilizes the formulation of Equations 44–48 for the iterative construction of a linear separator for the state sets $\overline{P(\bar{S}_{rs})}$ and $P(\bar{S}^b_{r\bar{s}})$, is provided in Figure 3. The

Input: (i) the set of safe states $\overline{P(\bar{S}_{rs})}$; (ii) the set of unsafe states $P(\bar{S}^b_{r\bar{s}})$; (iii) the parameters ε and M to be used during the solution of the MIP formulation of Equations 44–48
Output: (i) a list L containing the coefficients of the generated linear inequalities $(\mathbf{a}(l,.), b(l))$; (ii) the number of the linear inequalities in L, k'

/* Let *UnseparatedUnsafeStates* be the set of all unsafe states $\mathbf{y}_i \in P(\bar{S}^b_{r\bar{s}})$ that are not separated by any of the already generated hyperplanes; i.e., $\mathbf{a} \cdot \mathbf{y}_i - b(l) < \varepsilon$ for every hyperplane $(\mathbf{a}, b)$ in list L. */

1. $UnseparatedUnsafeStates \leftarrow P(\bar{S}^b_{r\bar{s}})$; $k' := 0$;
2. While *UnseparatedUnsafeStates* $\neq \emptyset$
   (a) $k' := k' + 1$;
   (b) Generate a hyperplane $(\mathbf{a}, b)$ by solving the MIP formulation of Equations 44-48 with input $\overline{P(\bar{S}_{rs})}$ and *UnseparatedUnsafeStates*
   (c) Add $(\mathbf{a}, b)$ to L
   (d) For each $\mathbf{y}_i \in$ *UnseparatedUnsafeStates*
   - if $(\mathbf{a} \cdot \mathbf{y}_i - b \geq \varepsilon)$
   – Remove $\mathbf{y}_i$ from *UnseparatedUnsafeStates*
3. Return L and k'

Fig. 3. A heuristic algorithm for the iterative construction of a linear separator for the state sets $\overline{P(\bar{S}_{rs})}$ and $P(\bar{S}^b_{r\bar{s}})$.

next proposition establishes an upper bound to the potential suboptimality of this algorithm.[9]

Proposition 6: The number of the separating hyperplanes obtainted by the algorithm of Figure 3 is at most $K \ln |P(\bar{S}^b_{r\bar{s}})|$, where K is the minimum number of hyperplanes that achieves separation.

Proof: In order to simplify the notation employed in this proof, we shall set $U \equiv P(\bar{S}^b_{r\bar{s}})$, i.e., the set of unsafe states fed to the algorithm of Figure 3, and we shall also set $m_u \equiv |P(\bar{S}^b_{r\bar{s}})|$. In addition, $W(h)$ will denote the elements of U that are separated from the set of safe states, $\overline{P(\bar{S}_{rs})}$, by any given hyperplane h. We claim that for each $U' \subseteq U$, there exists a hyperplane h that separates at least $|U'|/K$ states of U'; i.e.,

$$\forall U' \subseteq U, \ \exists h : \ |W(h) \cap U'| \geq |U'|/K \tag{49}$$

To see the validity of Equation 49, just notice that if it was not true, we would need more than K hyperplanes to separate U', and the same fact would be true for the separation of the superset U. But this contradicts the definition of K as the minimum number of hyperplanes that achieves the separation of U.

Next consider the execution of the algorithm of Figure 3, and let $U_i \subseteq U$ denote the set of unsafe states still not separated after i steps of the algorithm. Also, let h_{i+1} be the hyperplane

[9]The content of this result and the arguments employed in its proof are reminiscent of similar results developed for the set cover problem [25]. In fact, an interesting research extension for the results presented in this section, is a more profound exploration of the analogies between the classification problem addressed herein and the set cover problem, and of the implications of these analogies for the complexity of the considered problem and the approximation of its optimal solution.

generated at step $(i+1)$. Hence,

$$W(h_{i+1}) \cap U_i = U_i \backslash U_{i+1} \tag{50}$$

Since the considered algorithm maximizes $|W(h_{i+1}) \cap U_i|$, we can infer from Equations 49 and 50 that

$$\begin{aligned} |W(h_{i+1}) \cap U_i| = |U_i| - |U_{i+1}| \geq |U_i|/K \quad \Longrightarrow \\ |U_{i+1}| \leq |U_i|(1-1/K) \end{aligned} \tag{51}$$

So, by induction on i, we have that

$$|U_i| \ \leq \ (1-1/K)^i m_u \tag{52}$$

Choosing $i \ \geq \ K\ln(m_u)$, we see that

$$|U_i| \ \leq \ (1-1/K)^{K\ln(m_u)} m_u \ < \ e^{-\ln(m_u)} m_u \ = \ 1 \tag{53}$$

Thus, all the unsafe states in U must have been separated from the safe states in $\overline{P(\bar{S}_{rs})}$ after $K\ln(m_u)$ steps. □

Example Closing this section, we notice that the application of the heuristic of Figure 3 to the sets $\overline{P(\bar{S}_{rs})}$ and $P(\bar{S}^b_{rs})$ of Table I, while setting $\varepsilon = 0.01$, resulted in a linear classifier consisting of two inequalities, and therefore, of minimal size. For completeness, we mention that the obtained inequalities were as follows:

$$\mathbf{s}[1] + 0.01 \cdot \mathbf{s}[4] + 0.99 \cdot \mathbf{s}[5] \leq 1.0$$

$$\mathbf{s}[2] + \mathbf{s}[4] \leq 1.99$$

VI. Computational results

In this section we report the results from a series of computational experiments, in which we applied the DAP design methodology described in the earlier parts of this manuscript upon a number of randomly generated instantiations of Gadara RAS. We remind the reader that according to the class definition, all resources in a Gadara RAS possess unit capacity. On the other hand, each of the generated instances was further specified by:

- The number of resources in the system; the range of this parameter was between 1 and 14.
- The number of process types in the system; the range of this parameter was between 2 and 14. Furthermore, in the considered experiments all process types were assumed to have a simple linear structure, with the corresponding graphs $\mathcal{G}_j$ being simple paths (i.e., paths without any loops) for all j.
- The number of transitions in each process, with each transition corresponding to a single resource acquisition or a single resource release. The range of this parameter was between 2 and 14, but additional logic was applied to ensure a meaningful resource allocation sequence; hence, the eventual number of transitions appearing in every generated process differed by the originally specified number.[10] In particular, upon its initiation, a process was allocated randomly one of the system resources. At every subsequent transition, the process was either releasing one of its allocated resources, or it was acquiring one of the remaining resources. The association of any given transition with a resource release or a resource acquisition was equiprobable, except for the case where the process found itself possessing no resources; in that case the next transition was an acquisition with probability one. Similarly, the selection of the resource to be released by the process during a release transition, or the resource to be added to its current acquisitions during an acquisition transition, was determined equiprobably among the corresponding resource sets. Once the pre-specified number of transitions was determined as described above, the necessary number of release transitions was appended so that the process eventually returned all the acquired resources. Furthermore, in order to remain consistent with the RAS structure of Definition 1, all process stages resulting from the above construction that might correspond to zero resource allocation were identified and pruned from the process-defining sequence.

The employed RAS generator was encoded in Java, and it was compiled and linked by Java 1.6.0. Each generated RAS instance, Φ, was subjected to the DAP design process described in Figure 2. The construction of the linear classifier itself was performed according to, both, the exact and heuristic approaches described respectively in Sections IV and V. In the case of the exact approach, we imposed a hard limit of 30 minutes (or 1800 secs) for the solution of the MIP formulation of Equations 36–42. For instances that were not completely solved within this time budget, the solver terminated prematurely and reported the best feasible solution identified up to that point. All our computational experiments were performed on a 2.66 GHz quad-core Intel Xeon 5430 processor with 6 MB of cache memory and 32 GB RAM; however, each job was single-threaded. The algorithms involved in the preprocessing stages of the proposed methodology were encoded in C++, compiled and linked by the GNU g++ compiler under Unix, while the MIP formulations of Equations 36–42 and Equations 44–48, that are employed respectively by the exact and the heuristic approaches, were solved through ILOG CPLEX 11.1 with ILOG Concert technology using C++.

Table III reports a representative sample of the results that we obtained in our experiments.[11] The reported cases are ordered in decreasing magnitude of the corresponding $|S_{rs}|$, the number of reachable and safe states (second column in the presented table). The first column in Table III reports the dimensionality of the original state space corresponding to each listed configuration. Columns 3 – 9 report the cardinalities of the state subsets extracted through the various processing stages depicted in Figure 2, and also, the dimensionality reduction that was obtained through the projection P, discussed in the earlier parts of this section. Columns 10 and 11, entitled by k_{exact} and $k_{heuristic}$, report the number of linear inequalities in the solutions returned by the exact and the heuristic approach, respectively. Furthermore, the qualification [O] and [F] of the values reported in Column 10 indicates whether the obtained solution was optimal or just a feasible one (this would happen if the solution algorithm was terminated prematurely, upon the exhaustion of the 30 minute budget). Finally, Columns 12, 13 and 14, respectively entitled by $t_{thinning}$, t_{exact} and $t_{heuristic}$, report the amount of

[10] The particular RAS dynamics adopted in the presented experiments were chosen so that they imitate closely the lock allocation and deallocation in real computer programs.

[11] This sample has been selected from data resulting by the application of the proposed methodology on more than 500 instances of the Gadara RAS generated according to the logic described in the earlier parts of this section.

TABLE III
A sample of our experimental results

ξ	$\lvert S_{rs}\rvert$	$\lvert S_{r\bar{s}}\rvert$	$\lvert S^b_{r\bar{s}}\rvert$	$\lvert \bar{S}_{rs}\rvert$	$\lvert \bar{S}^b_{r\bar{s}}\rvert = \lvert P(\bar{S}^b_{r\bar{s}})\rvert$	$\lvert I_P\rvert$	$\lvert P(\bar{S}_{rs})\rvert$	$\lvert \overline{P(\bar{S}_{rs})}\rvert$	k_{exact}	$k_{heuristic}$	$t_{thinning}$	t_{exact}	$t_{heuristic}$
61	8,696,502	71,677	71,677	162,052	5	7	35	9	1 [O]	1	80	0	0
75	5,699,463	268,807	267,853	389,332	43	23	3,613	315	4 [O]	4	57	26	0
107	5,696,776	1,165,958	1,021,301	1,544,165	155	35	3,286	533	12 [F]	9	100	1,800	1,800
70	5,501,728	1,321,928	1,137,856	152,570	21	21	1,340	245	4 [O]	5	70	10	0
74	4,432,641	283,561	278,490	660,951	28	21	3,068	367	4 [O]	4	42	4	0
60	3,994,272	348,576	348,576	105,606	12	10	44	22	2 [O]	2	39	0	0
97	3,718,540	706,177	622,035	938,461	122	31	2,672	418	8 [F]	7	56	1,800	492
79	3,144,658	249,690	246,301	754,307	75	22	1,366	330	9 [F]	7	30	1,800	1
69	3,102,964	56,752	56,752	201,944	38	20	2,386	235	4 [O]	4	25	12	0
87	2,841,494	834,672	750,951	386,960	40	28	1,802	175	4 [O]	4	43	45	0
112	2,521,030	556,743	518,684	929,498	168	38	2,633	643	10 [F]	8	42	1,800	675
64	2,501,508	501,060	433,632	54,496	18	17	307	71	3 [O]	4	27	1	0
74	1,953,671	110,937	103,311	109,964	57	17	272	81	2 [O]	2	18	0	0
73	1,906,704	152,387	150,349	423,799	50	21	1,139	266	6 [F]	7	17	1,800	0
106	1,696,349	382,291	352,622	587,314	152	36	2,295	553	8 [F]	8	25	1,800	97
63	1,567,434	17,579	17,579	84,109	29	17	1,194	163	3 [O]	3	11	1	0
96	1,240,726	188,189	181,689	413,175	113	33	2,005	467	8 [F]	8	13	1,800	2
67	1,197,240	121,442	97,434	30,481	13	15	86	30	2 [O]	2	11	0	0
53	963,900	9,618	9,618	29,354	5	7	35	9	1 [O]	1	6	0	0
71	911,283	209,248	199,507	98,772	13	18	380	67	2 [O]	2	8	0	0

computing time (in seconds) that was required to execute (i) the pre-processing steps indicated in Figure 2, (ii) the construction of the linear classifier obtained by the exact approach, and (iii) the construction of the linear classifier obtained by the heuristic approach. The perusal of the data provided in Table III reveals very clearly

- the efficacy and the significance of the various set-"thinning" steps performed by the process depicted in Figure 2,
- the solvability of the exact MIP formulation of Section IV for very large configurations from the considered RAS class, as a result of this "thinning" process, and also,
- the capability of the heuristic algorithm of Section V to provide separators that are (i) very efficient compared to those returned by the exact approach, and (ii) obtained in computational times that are significantly shorter than the times necessary for the solution of the exact MIP formulation.

VII. Conclusions

This work has developed a novel methodology for the effective deployment of the maximally permissive DAP in the context of the complex resource allocation that takes place in many contemporary applications. Our results are enabled by (i) a careful distinction between the off-line and the on-line parts of the computation that is required for the effective characterization and deployment of the target policy, and (ii) the effective control of the complexity involved in the on-line part of the computation through the employment of pertinent representations and data structures. More specifically, under the proposed approach, the on-line implementation of the maximally permissive DAP is perceived as a classifier that effects the dichotomy of the underlying state space into the safe and unsafe subspaces, and therefore, the efficient implementation of the policy reduces to the synthesis of a classifier that can express the sought separation of the state space in a succinct and compact manner. It was shown that certain properties of the underlying state space enable extensive reductions of the information that must be explicitly considered during the classifier synthesis process, alleviating substantially the computational effort of the process itself, and enabling the structural compactness and minimality of the derived supervisor. The manuscript provided a thorough formal treatment of the above ideas in the context of Gadara RAS, a particular RAS class for which the sought classifiers can take the convenient form of a set of linear inequalities. Furthermore, extensive numerical experimentation in the context of the aforementioned RAS class confirmed the tractability of the approach and its ability to provide compact implementations of the maximally permissive DAP for RAS with a very large size and very large state spaces.

As it was pointed out in Section II, the methodology for the synthesis of maximally permissive and compact DAPs developed in this manuscript, can also be applied to the broader RAS class that is obtained from the class of Gadara RAS through the removal of Assumption 2 that was stated in that section, but in that case there is not guarantee for its completeness. More specifically, the MIP formulation of Section IV may fail to identify any feasible solutions, and the iterations performed by the heuristic of Section V may reach a point where none of the remaining unsafe states will be linearly separable from the safe ones. Hence, in our future work we shall seek the extension and the detailed implementation of the basic approach developed in this manuscript in a way that guarantees its completeness for RAS classes beyond that of the Gadara RAS. Some initial results in this direction can be found in [26].

References

[1] P. J. G. Ramadge and W. M. Wonham, "The control of discrete event systems," *Proceedings of the IEEE*, vol. 77, pp. 81–98, 1989.

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

[3] T. Araki, Y. Sugiyama, and T. Kasami, "Complexity of the deadlock avoidance problem," in *2nd IBM Symp. on Mathematical Foundations of Computer Science*. IBM, 1977, pp. 229–257.

[4] E. M. Gold, "Deadlock prediction: Easy and difficult cases," *SIAM Journal of Computing*, vol. 7, pp. 320–336, 1978.

[5] M. A. Lawley and S. A. Reveliotis, "Deadlock avoidance for sequential resource allocation systems: hard and easy cases," *Intl. Jrnl of FMS*, vol. 13, pp. 385–404, 2001.

[6] Z. A. Banaszak and B. H. Krogh, "Deadlock avoidance in flexible manufacturing systems with concurrently competing process flows," *IEEE Trans. on Robotics and Automation*, vol. 6, pp. 724–734, 1990.

[7] 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.

[8] 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.

[9] J. Ezpeleta, F. Tricas, F. Garcia-Valles, and J. M. Colom, "A Banker's solution for deadlock avoidance in FMS with flexible routing and multiresource states," *IEEE Trans. on R&A*, vol. 18, pp. 621–625, 2002.

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

[11] E. Badouel and P. Darondeau, "Theory of regions," in *LNCS 1491 – Advances in Petri Nets: Basic Models*, W. Reisig and G. Rozenberg, Eds. Springer-Verlag, 1998, pp. 529–586.

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

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

[14] S. Reveliotis, "Algebraic deadlock avoidance policies for sequential resource allocation systems," in *Facility Logistics: Approaches and Solutions to Next Generation Challenges*, M. Lahmar, Ed. Auerbach Publications, 2007, pp. 235–289.

[15] Z. Li, M. Zhou, and N. Wu, "A survey and comparison of Petri net-based deadlock prevention policies for flexible manufacturing systems," *IEEE Trans. Systems, Man and Cybernetics – Part C: Applications and Reviews*, vol. 38, pp. 173–188, 2008.

[16] Y. Wang, T. Kelly, M. Kudlur, S. Lafortune, and S. Mahlke, "Gadara: Dynamic deadlock avoidance for multithreaded programs." in *OSDI '08 – Proceedings of the 8th Usenix Symposium on Operating Systems Design and Implementation*. USENIX Association, 2008, pp. 281–294.

[17] Y. Wang, S. Lafortune, T. Kelly, M. Kudlur, and S. Mahlke, "The theory of deadlock avoidance via discrete control," in *POPL '09 – Proceedings of the 36th annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. New York, NY, USA: ACM, 2009, pp. 252–263.

[18] Y. Wang, H. Liao, S. Reveliotis, T. Kelly, S. Mahlke, and S. Lafortune, "Gadara nets: Modeling and analyzing lock allocation for deadlock avoidance in multithreaded software," in *Proceedings of the 48th IEEE Conference on Decision and Control*, Dec. 2009, pp. 4971 – 4976.

[19] T. Kelly, Y. Wang, S. Lafortune, and S. Mahlke, "Eliminating concurrency bugs with control engineering," *IEEE Computer*, vol. 42, no. 12, pp. 52–60, Dec. 2009.

[20] 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.

[21] S. A. Reveliotis, "Structural analysis & control of flexible manufacturing systems with a performance perspective," Ph.D. dissertation, University of Illinois, Urbana, IL, 1996.

[22] D. Bertsimas and J. N. Tsitsiklis, *Introduction to Linear Optimization*. Belmont, MA: Athena Scientific, 1997.

[23] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms, 2nd ed.* Boston, MA: MIT Press, 2001.

[24] L. A. Wolsey, *Integer Programming*. NY, NY: John Wiley & Sons, 1998.

[25] V. Vazirani, *Approximation Algorithms*. NY,NY: Springer, 2003.

[26] A. Nazeem and S. Reveliotis, "A practical approach to the design of maximally permissive liveness-enforcing supervisors for complex resource allocation systems," in *Proceedings of the 6th IEEE Conf. on Automation Science and Engineering*. IEEE, 2010.