Real-Time Management of Complex Resource Allocation Systems: Necessity, Achievements and Further Challenges ⋆,⋆⋆

Spyros Reveliotis *

* *School of Industrial & Systems Engineering, Georgia Institute of Technology, Atlanta, GA, 30332, USA (e-mail: spyros@isye.gatech.edu).*

Abstract: Many contemporary applications, ranging from flexibly automated production systems, to automated material handling and intelligent transportation systems, to internet-based workflow management systems, and more recently, to the massively parallelized software systems that emerge in the context of the novel multi-core computing architectures, can be perceived as a set of finite resources that support a number of concurrently running processes; these processes execute in a staged manner and vie for the allocation of various subsets of the system resources. To effectively support and manage the extensive levels of concurrency and operational flexibility that are contemplated for these environments, and the ensuing complexity, there is a substantial need for formal models and tools that will enable the modeling, analysis and eventually the control of the aforementioned resource allocation function so that the resulting dynamics are, both, behaviorally correct and operationally efficient. This write-up overviews a research program that seeks to address the aforementioned need by using the unifying abstraction of the resource allocation system (RAS) and supporting modeling frameworks, like automata, Petri nets, and Markov reward and decision processes, borrowed from the area of Discrete Event Systems (DES) theory. The presented results take advantage of the special structure that exists in the considered RAS classes, and are further characterized by, both, their analytical rigor and computational tractability. The write-up also highlights the further challenges that must be addressed for the successful completion and promotion of the pursued framework.

Keywords: Resource Allocation Systems, Discrete Event Systems, Supervisory Control, Deadlock Avoidance, (Stochastic) Scheduling

1. INTRODUCTION

The efficient and expedient allocation of a finite set of resources to a number of contesting processes is a ubiquitous problem, arising in various operational settings of our contemporary technological civilization. Indeed, cost-effectiveness and responsiveness are predominant concepts in modern corporate strategy and typical requirements for many everyday functions. In the context of the resource allocation functions considered in this work, cost-effectiveness is based on the ability to "make the most" – i.e., maintain a high utilization – of the engaged resources. On the other hand, the posed requirement for high responsiveness traditionally has implied the ability to fill an arising demand or to support an emerging service need in a timely manner. But in the current competitive markets, responsiveness also implies the ability to provide a broad range of diversified products and services, each of them appealing to a different market segment. And this trend for extensive diversification has more lately evolved to the mass-customization practices that have been adopted by certain industries, like the automotive and computer manufacturers, and are contemplated by many more. The effective support of all the aforestated requirements necessitates the further deployment, at the operational level, of high levels of concurrency and flexibility; i.e., a need to support the simultaneous execution, on the same set of resources, of a broad set of diversified workflows, while each of these workflows takes place at a low or moderate rate, and evolves continually into new operational patterns.

A concrete manifestation of the aforementioned trend is provided by the proverbial concept of the "flexible manufacturing system" (FMS), that has been extensively discussed in the context of various industrial settings for many years. The prototypical FMS consists of a set of numerically controlled workstations interconnected by an automated material handling system, and with the workflow taking place in the entire facility being integrated and coordinated by a computerized controller. More recently, this trend has been extended to the service sector through the concept of the "workflow management system", i.e., a computerized tool that supports the definition, the enactment and the coordination of the execution of business

⋆ This work has been partially supported by NSF grant ECCS-1405156.

⋆⋆For its most part, this write-up constitutes an extended abstract of the material that is presented in Reveliotis (2015). We refer to that work for a more systematic and extensive coverage of the presented material, and for the corresponding bibliography.

workflows, by monitoring their progress and assigning the necessary resources to them; these resources can range from data files, to supporting processing software, to printing and communication means, and even the humans that might be necessary for the support and the authorization of certain steps. It is currently believed that various routine transactions in the banking sector, in the supply chain management and logistics services, in insurance claim processing, and even in the broader health-care sector, can be rendered more responsive and efficient through their mechanization by the successful deployment of a workflow management system.

In the transportation sector, the aforementioned resource allocation paradigm manifests itself in any set-up where a set of concurrently traveling vehicles have to negotiate the necessary traveling space in the context of some "zoning scheme" that ensures collision-free operation. Trains being successively allocated a sequence of segments of the underlying railway network is a typical example of such a "zone"-based operation. Also, in the industrial sector, similar zoning schemes have been implemented in the operation of unit-load automated guided vehicle (AGV) and monorail-based material-handling systems, and in the operation of the hoist and the crane systems that support the material-handling operations at many ports and freight terminals.

Finally, another domain where the aforementioned resource allocation functions (and problems) are very prominent, is in the software and the computational platforms that control the aforementioned operations as well as any other operation that is supported by our modern technological civilization. Indeed, since its early days, our modern computing technology has employed extensive levels of (actual or virtual) concurrency, where a number of software threads run in parallel, each of them tasked with a particular role and function. These threads need to share the limited resources of the underlying computer platform (CPUs, registers, I/O devices, files, etc.) in a way that provides to each of these processes exclusive access and engagement of these resources; the corresponding coordination is attained through the use of a set of tokens, that are typically known as "mutually exclusive (mutex) locks" or "semaphores", and constitute essentially a "pass" for accessing the corresponding resource.

The above discussion regarding the increased levels of efficiency / cost-effectiveness, responsiveness, concurrency and flexibility that are requested for many contemporary operations, and the accompanying examples, also render pretty clear that these requirements lead to operations that are characterized by a high level of operational complexity. And this complexity translates to some challenging scheduling problems for the underlying resource allocation functions. In many cases, including all of the aforementioned examples, things are further complicated by the extensive levels of automation and autonomy that is requested by the considered operations. The need for automation can arise from technological and/or feasibility considerations (as in the case of the multithreaded software mentioned above, and in the operations taking place in the modern semiconductor fabs that must be isolated from the polluting effect of the human element), or from considerations concerning the operational and financial efficiency of the underlying operation (as in the aforementioned workflow management systems and the driverless transportation systems). In either case, the removal of the human element from the underlying processes implies that the deployed controllers, and especially the resource allocation functions involved, must be not only efficient but also correct and robust to logical problems and errors that, in more traditional settings, have been addressed by human intervention and improvisation. A typical such logical problem in the context of the considered resource allocation functions is the formation of (partial) deadlock, i.e., situations where a set of the concurrently executing processes are entangled in a circular waiting pattern, each of them waiting for some of the other processes to release resources that are necessary for its advancement. Hence, any such deadlock is a pernicious situation that stalls the further advancement of the processes involved and drives to zero the utilization of all the resources that have been allocated to these processes. At the same time, it should be obvious that deadlock is a natural consequence of the concurrency and the flexibility, and, finally, the arbitrary structure of the corresponding resource allocation function that is implied by the first two requirements; therefore, it constitutes a ubiquitous problem for the operational environments discussed in the previous paragraphs.

The bottom line of all the above discussion is that, in the context of the considered automated operations, the underlying resource allocation functions must be controlled for operational efficiency, cost effectiveness and responsiveness, and also for correctness and robustness to certain problems of a more qualitative or "logical" nature, like the formation of partial deadlock. In the rest of this write-up we present the results of a research program that has sought to provide a systematic and rigorous solution to this challenging resource allocation problem by employing and extending results from modern control theory.

2. RESOURCE ALLOCATION SYSTEMS AND THE PROPOSED CONTROL FRAMEWORK

The presented research program has sought to address the control problem that was outlined in the introductory section, in a systematic and rigorous manner, by (i) abstracting the considered operations through the notion of a (sequential) Resource Allocation System (RAS), and (ii) employing and extending results coming from the controls area of Discrete Event Systems (DES). In this section, first we introduce the formal notion of the sequential RAS, and subsequently we outline the DES-based control framework that has been proposed for these RAS. We also present a RAS taxonomy that has been instrumental in the investigation of the relevant control problems. These problems, themselves, and the currently available results for them, as well as the remaining open challenges, will be addressed in subsequent sections.

Sequential Resource Allocation Systems. A sequential RAS is formally defined by a quintuple $\Phi = \langle \mathcal{R}, C, \mathcal{P}, \mathcal{A}, \mathcal{D} \rangle$, where: (i) $\mathcal{R} = \{R_1, \ldots, R_m\}$ is the set of the system *resource types*. (ii) $C : \mathcal{R} \rightarrow \mathbb{Z}^+$ – the set of strictly positive integers – is the system *capacity* function, characterizing the number of identical units from each resource type available in the system. Resources are assumed to be

reusable, i.e., each allocation cycle does not affect their functional status or subsequent availability, and therefore, $C(R_i) \equiv C_i$ constitutes a system *invariant* for each i. (iii) $\mathcal{P} = \{\Pi_1, \ldots, \Pi_n\}$ denotes the set of the system *process types* supported by the considered system configuration. Each process type Π_j is a composite element itself, in particular, $\Pi_j =< \Delta_j, \mathcal{G}_j >$, where: (a) $\Delta_j = \{\Xi_{j1}, \ldots, \Xi_{j,l_j}\}$ denotes the set of *processing stages* involved in the definition of process type Π_j, and (b) $\mathcal{G}_j$ is an additional data structure that encodes the sequential logic that integrates the set of the processing stages Δ_j into a set of potential process flows. (iv) $\mathcal{A} : \Delta \rightarrow \prod_{i=1}^{m}\{0, \ldots, C_i\}$ is the *resource allocation function* associating every processing stage Ξ_{jk} with the *resource allocation vector* $\mathcal{A}(\Xi_{jk}) \neq \mathbf{0}$ required for its execution. (v) $\mathcal{D}$ is a function mapping each processing stage Ξ_{jk} in $\Delta \equiv \bigcup_{j=1}^{n} \Delta_j$ to a distribution with positive support that characterizes the *"processing time"* of the corresponding processing stage. Finally, we also set $\xi \equiv |\Delta|$, and for purposes of complexity considerations, we define the *size* $|\Phi|$ of RAS Φ by $|\Phi| \equiv |\mathcal{R}| + \xi + \sum_{i=1}^{m} C_i$.

At any point in time, the system contains a certain number of (possibly zero) instances of each process type that execute one of the corresponding processing stages; this distribution of the active process instances across the various processing stages defines a notion of *"state"* for the considered RAS. Obviously, this RAS state must respect the resource capacities; i.e., no resource type $R_i \in \mathcal{R}$ can be over-allocated w.r.t. its capacity C_i at any point in time. Furthermore, in order to model the "hold-while-waiting" effect that is exhibited in the resource allocation dynamics of the considered processes, the adopted resource allocation protocol stipulates that a process instance J_j executing a non-terminal stage Ξ_{jk} and seeking to advance to a next stage $\Xi_{jk'}$, must first be allocated the resource differential $(\mathcal{A}(\Xi_{ik'}) - \mathcal{A}(\Xi_{ik}))^+$ and only then will it release the resource units $|(\mathcal{A}(\Xi_{jk'}) - \mathcal{A}(\Xi_{jk}))^-|$, that are not needed anymore. [1] Then, in the resulting operational context, the *RAS deadlock* can be formally defined as a RAS state containing a set of active process instances, DJ, such that every instance $J_j \in DJ$, in order to advance to any of its next processing stages, requests some resources currently held by some other process instance $J_k \in DJ$.

A real-time control framework for the considered RAS. As remarked in the introductory section, an effective real-time controller for the considered RAS must ensure the attainment of some set of performance objectives typically defined w.r.t. the timed RAS behavior, while keeping the RAS away from problematic behavioral patterns like the aforementioned deadlock states. This last control requirement is frequently known as the RAS *behavioral* or *logical* control problem, because the corresponding problematic behavior can be effectively avoided by controlling only the sequencing of the relevant resource allocation events and not their exact timing. Furthermore, it is generally accepted by the relevant research community that, due to the stochasticity that is generally present in the timed dynamics of the considered RAS, any robust solution to the RAS behavioral and performance control problems

[1] This assumption is not restrictive since the release of resources that do not adhere to this protocol can be modeled by the insertion of additional processing stages in the underlying process plan.

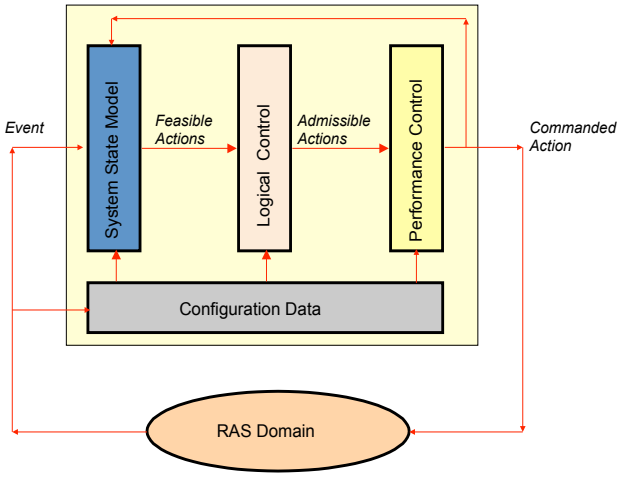

Fig. 1. An event-driven control scheme for the real-time management of the considered RAS.

should rely on some feedback control scheme and not on the open-loop execution of some precomputed plan.

Such a feedback-based controller is presented in Fig. 1. The depicted control paradigm is an *event-driven* approach, where the applied control function monitors the *events* taking place in the underlying RAS and commands a certain *action sequence* in response to these events. More specifically, the proposed controller maintains a representation of the RAS *state*, which enables it to monitor the system status and to identify the entire set of *feasible actions* that can be executed by the system at any given time. Hence, this information is instrumental for enabling the controller to determine the scope of its possible responses to a certain event. However, the controller must eliminate (*"filter out"*) all those actions that can result in problematic behavior. Such problematic behavior includes the formation of deadlock, but in the more general case, this part of the depicted control scheme will address additional specifications that might be defined, for instance, by quality concerns or some policy considerations, like those arising from a notion of "fairness" to the contesting processes. All the aforementioned concerns boil down to the systematic exclusion of certain resource allocation patterns from the RAS behavior, and, as already mentioned, the resulting problem is generally known as the *logical* or *behavioral* control problem to be addressed by the controller. The set of actions eventually accepted by the logical controller defines the space of the *"admissible"* behavior for the considered RAS. Then, the second stage of the proposed control logic must shape / bias this admissible behavior in a way that aligns best with the system *performance objectives*; this biasing is effectively achieved through the selection of the particular admissible action to be commanded upon the system, at each decision stage. The corresponding problem is known as the RAS *performance-oriented* control or *scheduling*.

The effective deployment of the RAS control scheme that is described in the previous paragraph necessitates a pertinent formal characterization of the RAS state, and the reference of the RAS logical and performance-oriented control problems to appropriate formal modeling frameworks

Table 1. A RAS taxonomy

Based on the Process Sequential Logic	**Based on the Requirement Vectors**
Linear: Each process is defined by a linear sequence of stages	**Single-Unit:** Each stage requires a single unit from a single resource
Disjunctive: A number of alternative process plans encoded by an acyclic digraph	**Single-Type:** Each stage requires an arbitrary number of units, but all from a single resource
Merge-Split: Each process is a fork-join network	**Conjunctive:** Stages require different resources at arbitrary levels
Complex: A combination of the above behaviors	

that will enable a rigorous analysis of the corresponding RAS dynamics and the effective synthesis of the necessary policies. At a basic level, these capabilities have been conveniently provided to the developing RAS theory by the areas of *qualitative* and *quantitative analysis of Discrete Event Systems (DES)*. Generally speaking, DES theory is a field of modern control theory investigating the behavior of dynamical systems that evolve their state discontinuously over time, in response to the occurrence of certain critical, instantaneous events. In this general setting, *qualitative DES theory* uses formal linguistic frameworks borrowed from theoretical computer science, augmented with control-theoretic concepts and techniques, in order to analyze and control the event sequences that are generated and observed by the underlying DES dynamics. On the other hand, *quantitative DES theory* analyzes and controls the timed DES dynamics, using models and tools that are borrowed from (stochastic) OR and simulation theory. However, as it will be further revealed in the following, while enabling a formal positioning of the RAS behavioral and scheduling problems, the practical computational capabilities of the corresponding DES frameworks are severely limited by a very high representational and computational complexity. But in the subsequent parts of this write-up we shall demonstrate how the relevant research community has leveraged the representational and analytical capabilities that are provided by DES theory in order to develop effective *practical* solutions to the two aforementioned control problems that arise in the RAS operational context. These solutions result from the pertinent exploitation of the special structure that exists in the considered problems, and the corresponding developments have substantially enriched and extended the capabilities of the supporting DES theory itself.

A RAS taxonomy. We close the discussion on the basic RAS model and the induced control problems, by presenting a RAS taxonomy that has been instrumental for the systematic investigation of these problems, especially the RAS behavioral control problem of deadlock avoidance. The main RAS classes recognized by this taxonomy are defined by (i) the structure that is supported for the process sequential logic, and (ii) the structure of the resource allocation requests that are posed by the various processing stages; the most prominent RAS classes w.r.t. these two classification attributes are presented in Table 1. Furthermore, more recent developments have revealed the significance of some additional RAS attributes when it comes to the analytical characterization of the qualitative RAS dynamics and their control for deadlock avoidance. These new attributes include (iii) the absence of resources with *non-unit* capacities, (iv) the presence of *cycling* in the sequential logic of the RAS process types, and (v) the presence of RAS dynamics of an *uncontrollable* nature; this last feature can be further differentiated into (a) uncontrollability w.r.t. the exact timing of a certain resource allocation and (b) uncontrollability of the branching decisions of some underlying process that possesses alternate routings. We shall use this taxonomy in the next sections, especially when we discuss the existing theory on the RAS deadlock avoidance problem.

3. THE RAS LOGICAL CONTROL PROBLEM

The most straightforward way to represent formally the behavior of a given RAS Φ, and the corresponding logical control problem of deadlock avoidance and liveness-enforcing supervision, is by means of a finite state automaton (FSA), $G(\Phi)$. The state s of this FSA is a vector of dimensionality ξ – i.e., the number of the distinct processing stages of the underlying RAS Φ – where each component indicates the number of active process instances that execute the corresponding processing stage. The state space S consists of all those vectors that define a feasible resource allocation w.r.t. the RAS resource allocation function $\mathcal{A}$ and the resource capacities C_i. The events that evolve the RAS state comprise (i) the "loading" events that initiate new process instances, setting them to their first processing stage, (ii) the "advancing" events, that advance an active process instance from its current processing stage to a successor processing stage, and (iii) the "unloading" events that terminate those process instances that have completed their last processing stage and remove them from the system. The initial state of this automaton corresponds to the state $\mathbf{0}$ where the system is idle and empty of any jobs. State $\mathbf{0}$ is also the only marked state of G, a fact that expresses the desire for complete process runs. Then, the states of the FSA G that define the feasible behavior of the underlying RAS is the entire set of states, S_r, that are reachable from state $\mathbf{0}$. On the other hand, the admissible behavior is characterized by the set of states S_s that are co-reachable to state $\mathbf{0}$; in the relevant terminology, these states are also characterized as "safe", while their complement w.r.t. the state set S is the set of the "unsafe" states S_u. The state set S_u contains the set of the "deadlock" states, S_d, i.e., states that exhibit a (partial) deadlock, according to the definition of this concept in the previous section, but it can also contain states that do not exhibit such a partial deadlock but lead unavoidably to a deadlock state; this last set of states are characterized as "deadlock-free unsafe" states.

Given the above classification of the RAS state space S, the sought supervisor should restrict the RAS behavior in the subspace $S_r \cap S_s \equiv S_{rs}$, i.e., the set of reachable and safe states. This restriction ensures the ability of every activated process instance to terminate, and at the same time, it is the *minimal* required restriction of the uncontrolled RAS behavior for establishing deadlock-free operation; hence, in the relevant terminology, the corresponding supervisor is characterized as the *"maximally permissive deadlock avoidance policy" (DAP)*. A natural implementation of this policy would be an one-step-lookahead scheme that would permit a transition in automaton G on the ba-

sis of the "safety" (i.e., the co-reachability) of the resulting state. But it has been shown that assessing the safety of any given RAS state is an NP-complete problem even for the simplest class of the Linear-Single-Unit-RAS. Hence, the deployment of the maximally permissive DAP is an NP-Hard proposition. In the rest of this section we discuss how the relevant research community has coped with this negative result, developing a rich methodology that has enabled the implementation of the optimal or near-optimal DAPs for RAS instances of very large size and behavioral complexity.

As it has been the case with many other optimization problems that exhibit super-polynomial complexity, the first reaction of the research community on the deadlock avoidance problem was to identify surrogate conditions to safety that are testable with a polynomial complexity w.r.t. the underlying RAS size, $|\Phi|$, and lead to a significant coverage of the target state space S_{rs}. Such policies are known as "Polynomial Kernel" (PK-) DAPs in the relevant literature. A characteristic example of such a policy is Dijkstra's Banker's algorithm, that restricts the RAS behavior to a subspace of S_{rs} for which the existence of paths to the target state $\mathbf{0}$ is polynomially verifiable; such states are known as "ordered" states in the relevant literature. Some other classes of PK-DAPs for the considered RAS take the form of a polynomial set of linear inequalities on the RAS state s; specific examples of such policies are the Resource Upstream Neighborhood (RUN) Policy and the Resource Ordering (RO) policy that were originally developed for Linear-Single-Unit RAS, but have been extended to the case of Disjunctive-Conjunctive RAS as well. Furthermore, as we will discuss in more detail in the sequel, currently we also avail of a methodology that can assess automatically the ability of any given set of inequalities to define a correct DAP for a given RAS Φ. Finally, it should also be pointed out that the disjunction of a set of PK-DAPs for a given RAS Φ – i.e., the policy that admits a state s of Φ if it is admitted by any of the constituent policies – is another correct DAP with admissible state space equal to the union of the state spaces that are admitted by the constituent policies. Hence, such a DAP disjunction can be perceived as an attempt to reconstruct (an approximation of) the notion of RAS state safety by "patching" together a set of surrogate concepts, each contributing a particular facet of the target concept.

A second typical reaction to an NP-Hardness result, is the effort to identify "special structures" of practical interest – i.e., certain subclasses of the considered problem – that admit solutions of polynomial complexity. In the case of the deployment of the maximally permissive DAP for the considered RAS classes, such special structure has been defined effectively by two lines of analysis. The first line has sought to identify RAS structure that renders the search for a path to the marked state $\mathbf{0}$ a task of polynomial complexity w.r.t. the underlying RAS size $|\Phi|$. Generally speaking, this RAS structure enables the "easy" identification of process advancement steps that lead to a monotonic increase of the resource slackness, and therefore, can be effected by a "greedoid" type of algorithm (i.e., a search algorithm without the need for backtracking). The second line of analysis that has enabled the identification of RAS special structure admitting polynomially deployable maximally permissive DAP is based on the realization that while the assessment of RAS state safety is NP-complete, the detection of a deadlock in a RAS state is a task of polynomial complexity in many RAS classes of the taxonomy of Table 1, including the broad class of Disjunctive-Conjunctive RAS. This realization implies, in turn, that in any RAS classes with no deadlock-free unsafe states (i.e., RAS where $S_u = S_d$), unsafety is polynomially recognizable by substituting, in the aforementioned one-step-lookahead scheme that implements the maximally permissive DAP, the original tests for (un-)safety with the polynomial test for deadlock. There is a number of such RAS subclasses identified in the literature, but currently all of them are sub-classes of the Disjunctive-Single-Unit-RAS.

Substantial advances in the deployment of the maximally permissive DAP for the considered RAS have been obtained more recently through the realization that the NP-Hardness of this policy is essentially a "worst-case" result, while practical, efficient implementations of this policy can still be obtained by (i) isolating the hardest part of its computation to an off-line phase, and (ii) employing a pertinent, parsimonious representation of the final result of that first computational step that will enable the on-line / real-time instantiation of the policy with a manageable computational cost. In a basic implementation of the above idea, the off-line phase can be supported by the enumerative, set-theoretic techniques that are employed by the DES Supervisory Control (SC) theory. On the other hand, the development of a parsimonious representation of the obtained maximally permissive supervisor is based on the realization that such a supervisor essentially dissects the vector set S (or S_r) into two subsets containing, respectively, the admissible and the inadmissible states. Hence, at the end, the maximally permissive DAP acts as a "classifier" of the elements of this set. But then, the sought parsimonious representations of the maximally permissive DAP can be provided by more general results of classification theory adapted to the considered problem context. More specifically, it can be shown that a finite vector set can be dichotomized to two subsets by a disjunction of classifiers where each of these classifiers is implemented by a set of linear inequalities. In the considered problem context of the maximally permissive DAP, these inequalities present additional structure that results from the relative topology of the underlying safe and unsafe subspaces, and enables the effective design of the sought classifier while considering only the maximal safe and the minimal unsafe states under the partial ordering of these vectors that is based on a component-wise comparison. Furthermore, there are additional interesting and practical cases where the sought classifier can be expressed as a single set of linear inequalities; such a classifier has been characterized as "linear" in the relevant literature. Finally, a complementary line of research has sought to express the target DAP in a "non-parametric" manner, by simply identifying and storing the set of unsafe states that are on the "boundary" between the safe and the unsafe subspaces; i.e., those unsafe states that can be reached from the safe subspace in one transition. In this case, the aforementioned relative topology of the safe and unsafe subspaces renders the target set of the boundary unsafe states a "right-closed" set, and therefore, it can be represented compactly

by storing only its minimal elements. Finally, the relevant literature also avails of very efficient algorithms that can detect the minimal boundary unsafe states through only a partial exploration of the underlying state space, that starts from a pertinent reconstruction of the minimal deadlocks and backtraces "intelligently" from them. Some of these exploration algorithms are also employing the power of the symbolic computation that has been developed in the recent years for the efficient representation and processing of Boolean functions with a very large number of variables, as well as the representation and processing of more general very large finite sets stored under a binary representation. As a result, currently we are in a position to develop very compact representations of the maximally permissive DAP for RAS with very large sizes $|\Phi|$, and with billions of states in the corresponding state spaces.

We close our discussion of the RAS logical control problem by overviewing another line of research that has been very active and very prominent in the relevant literature, and with substantial theoretical and practical contributions in our efforts to cope with this problem. This research line is employing the formal modeling framework of Petri nets (PNs) for representing the qualitative RAS dynamics. Compared to the FSA modeling framework, that enumerates explicitly the underlying state space S, PNs enjoy a much more compact representation of the RAS dynamics, and even more importantly, they express explicitly the linkage of those dynamics to the underlying system structure. Hence, this modeling framework is particularly amenable for pursuing a "structural analysis" of the considered problem, i.e., a line of analysis that links particular behavioral properties of the system to structural formations and properties. In the context of the deadlock avoidance, a highly celebrated structural result concerns the association of the RAS partial deadlock and non-liveness to the structural objects of "empty", or, more generally, "deadly marked siphons". More specifically, these results attribute the RAS non-liveness to the formation of such badly marked siphons in the net markings or, in certain cases, the projection of these markings to appropriately selected subspaces. A powerful feature of this theory is that it has managed to express the aforementioned condition for the RAS (non-)liveness in the form of some mixed Integer programming (MIP) formulations that are obtained from, and are polynomially related in terms of their variables and constraints to, the structure of the underlying RAS. The derived tests will either conclude the liveness for the assessed RAS, or they will return a partial deadlock in the form of the corresponding deadly marked siphon. Furthermore, in the case of DAPs that can be expressed as additional places superimposed on the structure of the RAS-modeling PN (known as "monitor" or "control" places), the aforementioned tests can also assess the ability of these policies to establish correct deadlock avoidance for the considered RAS, and therefore, they can support a "DAP synthesis" process. Of particular interest are such synthesis processes where the maximally permissive DAP is obtained iteratively through the detection and control of partial deadlocks existing in the original RAS behavior, or maybe brought about by the policy logic that is incrementally defined through these iterations; this second type of partial deadlocks are characterized as "policy-induced" deadlocks. The literature avails of specific results where such an incremental synthesis will converge to the maximally permissive DAP. Obviously, a necessary condition for such a convergence is the ability to represent the maximally permissive DAP by a set of "monitor" places, a requirement that further translates to the possibility of expressing by a linear classifier the dichotomy of the state set S that is effected by the maximally permissive DAP. For the corresponding RAS classes, the target policy can be obtained while avoiding completely an explicit (even partial) enumeration of the underlying RAS state space.[2] Finally, we also mention, for completeness, that there have been additional attempts that seek to design the maximally permissive DAP for a given RAS through the standard DES SC theory, and subsequently, rehash the obtained policy in a PN representation, in order to exploit the compactness of this modeling framework; these approaches are collectively known as the "theory of regions", but they tend to suffer from a high representational complexity for the final result and also they lack completeness (since, as discussed above, the maximally permissive DAP must admit a linear representation in order to be effectively represented as a set of "monitor" places).

4. THE RAS PERFORMANCE CONTROL PROBLEM

As already mentioned in the previous sections, the RAS performance control problem is essentially a scheduling problem that must be formulated and solved on the admissible subspace that is defined by the RAS logical control policy. From a theoretical standpoint, these scheduling problems belong to the broader class of problems that seek to schedule stochastic networks with blocking, a particular area in the scheduling theory that has not been extensively researched up to this point. To a large extent, the current lack of results for this class of problems is due to the fact that blocking introduces strong "coupling" effects among the underlying workstations, or more generally the involved resources, that render the characterization of the optimal scheduling policy a very challenging undertaking. Furthermore, the permanent blocking – i.e., the deadlocking – effects that can arise in these environments, in addition to the transient blocking, have been another important source of complexity and a very practical roadblock to the endeavors of the more traditional stochastics community w.r.t. this set of scheduling problems. On the other hand, the current availability of the results on the RAS logical control problem that were described in the previous section provides a systematic methodology to deal with these blocking and deadlocking effects and opens the corresponding set of scheduling problems to more active and systematic research. In the rest of this section, we shall briefly outline some ongoing endeavors w.r.t. this set of problems. But as already mentioned above, it can be safely argued that the research on the RAS performance control has not reached the maturity and the richness of the results that characterize its logical control counterpart.

A typical framework for tackling the scheduling problems that are discussed in the previous paragraph is that of the Markov Decision Processes (MDPs). Under an approximation of the "processing time" distributions of the var-

[2] Essentially, the considered methods substitute the explicit search of the RAS state space for partial deadlocks with an *implicit* search that is effected by the aforementioned MIP formulations.

ious processing stages by pertinently selected phase-type distributions, and assuming some additional stationarity in the behavior of the involved process types, the MDP modeling framework enables the structured modeling and analysis of all the stochasticity that might characterize the operations of the considered RAS. Furthermore, the co-reachability of the admissible states w.r.t. the initial state $\mathbf{0}$ that is established by the applied logical control policy, implies that the obtained MDP formulations belong to the class of MDPs that, in principle, are solvable by effective and fairly efficient algorithms. Things are complicated, however, by the explosive size of the underlying state space; in fact, this state space explosion implies that the sought scheduling policies are not only hard to compute, but they also have a very high (frequently prohibitive) representational complexity.

Motivated by the above remarks, a recently initiated research program has sought to confine the aforementioned scheduling problems in policy spaces that admit a more parsimonious representation. Instrumental to this endeavor has been the expression of the underlying dynamics in the modeling framework of the Generalized Stochastic Petri Nets (GSPNs). The basic PN structure of this modeling framework enables the effective representation of the qualitative RAS dynamics and the corresponding logical control (e.g., deadlock avoidance) policy. At the same time, the partitioning of the net transitions to "timed" and "untimed", according to the standard GSPN semantics, enables the explicit expression of the corresponding scheduling problem by means of the externally provided distributions that will regulate the conflicts of the enabled untimed transitions at the various net markings. These distributions are known as "random switches" in the GSPN terminology and, in the considered problem context, essentially define randomized stationary policies for the corresponding MDP formulation. An optimal set of random switches can be computed, at least in principle, through a non-linear programming (NLP) formulation that has as its primary variables the elements (i.e., the probabilities) of these random switches, and as secondary variables the stationary distributions of the corresponding policies. Furthermore, this NLP formulation can be solved by stochastic approximation algorithms and results drawn from the theory of regenerative Markov reward processes, that enable the estimation of the performance objective of the formulation and its gradients through simulation and avoid the enumeration of the underlying state spaces.

On the other hand, the association of a distinct random switch to every GSPN marking with more than one enabled untimed transition suffers by a high representational cost for the defined policies that is similar to that experienced by the aforementioned MDP formulations. But the explicit representation of the system structure and dynamics by the underlying GSPN also enables the redefinition of the employed random switches in a way that reduces the representational complexity of the resulting scheduling policies, and still renders these policies quite pertinent and practical in the context of the considered operations. A particular such redefinition that is currently explored by the considered research program seeks the replacement of the original set of random switches by a "static" version of this set which assigns the same distribution to every GSPN marking that activates the same set of untimed transitions. This restriction reduces substantially the decision variables that appear in the final NLP formulation. At the same time, it can be easily seen that static random switches can express the entire class of "static priority" policies that are frequently used in many practical settings, and therefore, this set of policies is very relevant to the realities of the current industrial practice.

Furthermore, both classes of scheduling policies, based either on dynamic or static random switches, can be further refined, and their member policies can be rendered more parsimonious, by a pertinent assessment of the actual conflicts that might exist among the enabled untimed transitions at any given marking, and the elimination of any unnecessary effort to coordinate non-conflicting transitions. The corresponding analysis can be performed on the subgraphs of the underlying state transition diagram (STD) that characterize the net transition between two tangible markings, i.e., markings that enable only timed transitions and therefore not involving any further scheduling decisions. These subgraphs are rather "local" structures in the underlying STD and therefore, the aforementioned analysis for (non-)conflict can be effectively integrated in the simulation logic that is used for the solution of the corresponding NLP formulation. Practical experience has shown that such a refinement usually results in very extensive reductions of the decision variables employed by the final NLP formulation. We should also notice that the aforementioned screening process develops at the interface of the qualitative / behavioral and the quantitative / performance-oriented dynamics of the underlying RAS, and therefore, it relies substantially upon the ability of the GSPN modeling framework to provide an integrated representation of these two types of dynamics of the considered RAS.

An additional important remark is that static random switches essentially define an aggregation scheme on the underlying RAS state space. Hence, it is plausible to attempt the enhancement of the performance of a scheduling policy that is optimal within the class of policies expressed by the static random switches, through a "refinement" process that seeks the partial disaggregation of the underlying state aggregates. Such a disaggregation scheme can be driven by information that is conveyed in the structure of the current policy, and it constitutes an effective mechanism for managing the underlying trade-off between the operational efficiency of the obtained policies and their representational and operational complexity.

Finally, another important feature of the scheduling methodology that is outlined in the above paragraphs, is the ability of this methodology to integrate additional operational requirements by expressing them as "behavioral" requirements for the underlying RAS. More specifically, these requirements can be imposed on the considered RAS by the applied logical control policy, and effectively encoded by the GSPN structure that models the admissible RAS behavior. On the other hand, quite conveniently, such an approach leaves unaltered the computational procedures that will compute the final scheduling policies. As a more concrete example of this capability we mention the potential enforcement of throughput-ratio (or other similar) constraints regulating the relevant throughputs

of the various process types in any given RAS; these constraints essentially constitute "fairness" constraints for the corresponding processes and they can be encoded by superimposing additional structure on the RAS modeling GSPN.

Concluding this discussion on the performance control of the considered RAS, we want also to notice a set of further developments that pertain to this problem, although not necessarily presented by means of the corresponding terminology. Such an example is provided by the ongoing endeavors to develop deadlock-free, cyclic / repetitive schedules for the cluster tools that are used in the contemporary semiconductor fabs. These cluster tools can be modeled as Linear-Single-Unit RAS with deterministic processing times, and the aforementioned endeavors to the scheduling of these environments essentially constitute a specialization of the existing theory of max-plus algebra, that has been developed for the PN class of timed (weighted) marked graphs, to the particular structures that are encountered in the cluster tool configurations. Finally, there have been some sporadic endeavors to address some RAS scheduling problems with deterministic processing times through the generic search-based methods that are used in combinatorial optimization.

5. OPEN CHALLENGES

The previous sections have outlined the extensive progress that has been made by the existing RAS theory in its effort to support the efficient and expedient resource allocation that is necessary in the contemporary technological applications. Yet, there are important remaining challenges that must be addressed for the effective completion of this theory and its migration to the relevant practice. We conclude this write-up by highlighting some of these challenges and the opportunities that they define for the corresponding research communities.

It is evident from the previous discussion that a major remaining challenge for the existing RAS theory is the strengthening and the extension of the corresponding scheduling theory. This strengthening can be pursued at the modeling and the algorithmic level, by seeking the definition, the effective computation and the deployment of richer classes of parsimonious scheduling policies than the set of classes that was outlined in Section 4. But even more importantly, this set of scheduling problems requires a more profound understanding of the corresponding notions of "feasibility" and "optimality", from a qualitative / analytical standpoint. Characteristically, it is important to obtain a clearer understanding of the structure of the "value functions" for the MDPs that model the considered scheduling problems, and the particular factors that are most influential in the determination of the "value" of any given RAS state. The availability of such a perspective can subsequently guide the endeavors towards the development of pertinent and computationally efficient approximations of the optimal scheduling policies in the context of the burgeoning theory of the Approximate Dynamic Programming (ADP).

On the RAS logical control side, one can consider the extension of the existing theory to RAS with even more complex structure and behavior than that addressed by (most of) the existing results. The taxonomy of Section 2 offers a systematic base for organizing this extension. Perhaps, an even more important extension of the current RAS SC theory is in a direction that will enable it to address more challenging operational environments, like those that offer only partial observability of the underlying resource allocation function, or necessitate the distribution of the control function to a number of coordinating controllers due to scaling, communication or other operational constraints. In a similar vein, one can consider the problem of reactive or proactive accommodation of resource capacity losses, in a way that minimizes the experienced disruption and/or ensures a certain minimal functionality for the degraded system. All these developments can be referred to results borrowed from the existing DES theory, but it is also expected that the special and rich structure of the RAS concept will enable customized analyses and solutions for this set of problems, as well.

Another possible extension of the existing RAS theory is its enrichment in order to encompass additional operational constraints beyond the fundamental problem of deadlock avoidance. Along these lines, we already saw the potential need to observe certain "fairness" requirements across the RAS process types. This notion of "fairness" can be extended to more general "coordination" requirements among the running processes. In all these cases, the new operational requirements must be formally expressed in the employed DES-modeling frameworks of FSA and PNs, and the resulting formal models must be further analyzed for their properties w.r.t. blocking; the existing deadlock avoidance theory must be extended to this new class of systems, as well.

Finally, as the presented RAS theory grows and strengthens its methodology along the lines indicated in the previous paragraphs, additional endeavor must be expended towards the development of the human capital and of the technological and computational base that will enable the constructive migration of this theory to the future engineering practice. This endeavor certainly involves the eventual undertaking of some "pilot" large-scale applications that will highlight the technical strength of the theory and the competitive advantage that can be supported by it. But even more importantly, it must also seek the effective integration of the existing and the emerging results into the relevant engineering curricula, and the organization of these results in a series of computational platforms that will enable their robust and expedient utilization by the field engineers. In fact, this last activity can be part of a broader initiative concerning the further promotion of DES theory and of the emerging formal methods in the engineering curriculum and practice. It is expected that, collectively, all these endeavors will define a spectrum of fundamental developments and trends with profound and transformative repercussions for the related fields of control and automation engineering.

REFERENCES

Reveliotis, S. (2015). Coordinating autonomy: sequential resource allocation systems for automation. *IEEE Robotics & Automation Magazine*, June.