Chapter 6

Deadlock Avoidance Policies for Automated Manufacturing Systems Using Finite State Automata

Spyros Reveliotis and Ahmed Nazeem[1]

[1]S. Reveliotis is with the School of Industrial & Systems Engineering, Georgia Institute of Technology, email: spyros@isye.gatech.edu. A. Nazeem is with United Airlines, email: ahmed.nazeem@united.com. This work was partially supported by NSF grant CMMI-0928231.

Abstract

This chapter considers the problem of deadlock avoidance in flexibly automated manufacturing systems, one of the most prevalent supervisory control problems that challenges the effective deployment of these environments. The problem is addressed through the modeling abstraction of the (sequential) resource allocation system (RAS), and the pursued analysis uses concepts and results from the formal modeling framework of finite state automata (FSA). A notion of optimality is defined through the notion of maximal permissiveness, but the computation of the optimal DAP is shown to be NP-Hard. Hence, the last part of the chapter discusses some approaches that have been developed by the relevant research community in its effort to deal with this negative complexity result.

6.1 Introduction

Contemporary manufacturing is characterized by (i) an ever increasing emphasis on automation and by (ii) a quest for broader product lines and more customized product offerings [22]. Indeed, since the beginning of the industrial revolution, automation has been at the core of the mass-production concept and practices, enabling activities and tasks that frequently transcend the human element in terms of capability, consistency, durability and speed [21]. More recently, automation has also been advocated as a key enabler for controlling the overall production cost, especially in economies where labor cost is particularly high. Hence, there is an increasing tendency to confine the presence of the human element on the production shop-floor to supervisory and maintenance tasks only, while the actual production activity is taken over by "intelligent" hardware, materialized in the form of numerically controlled (NC) machines,

robots, and other material handling equipment. Clearly, enabling the reliable deployment of such an extensive mode of automation is a pretty challenging task. Yet, in the recent years, things have been further complicated, and the aforementioned challenges have been further exacerbated, by the second manufacturing trend mentioned above, i.e., the emphasis that is placed by the global markets upon choice and customization. This trend implies a need to support the production of a broader set of items, in smaller batch sizes, and with shorter life spans. Such demand and production patterns negate the economies of scale and the efficiencies of the dedicated production lines that have been sought by traditional manufacturing, and (re-)places the emphasis on concepts like resource sharing, flexibility, reconfigurability and concurrency [18]. But the current reality has also shown that the effective support of these concepts on the manufacturing shop-floor is highly non-trivial, and it adds an entirely new layer of complexity and challenges to the aforementioned endeavors towards the automated operation of production systems. The manufacturing world has become increasingly conscious of the fact that the successful design and deployment of flexibly automated production systems necessitates the development of a (manufacturing) "science", in the form of analytical and computational tools, that will enable the systematic design, the analysis, and the real-time management of these systems [49].

In fact, manufacturing systems have been a subject of study for decision sciences for a very long time. Characteristically, many of the performance evaluation and scheduling problems that have been extensively studied by the disciplines of Operations Management (OM) and Industrial Engineering (IE), have been motivated by manufacturing operations. However, these problems have been formulated and analyzed at a level of abstraction that presumes the presence of the human element as a facilitator and the handler of operational aspects that are not addressed by the models themselves. Abstracting away such operational details certainly simplifies the addressed problem(s), and whenever applicable, it constitutes an effective tool for managing the underlying problem complexities. On the other hand, the ongoing migration to more extensively automated operational modes that was mentioned in the previous paragraph, implies that these operational details must be (re-)introduced in the pursued modeling and analysis. An autonomous manufacturing system must be designed and analyzed not only with

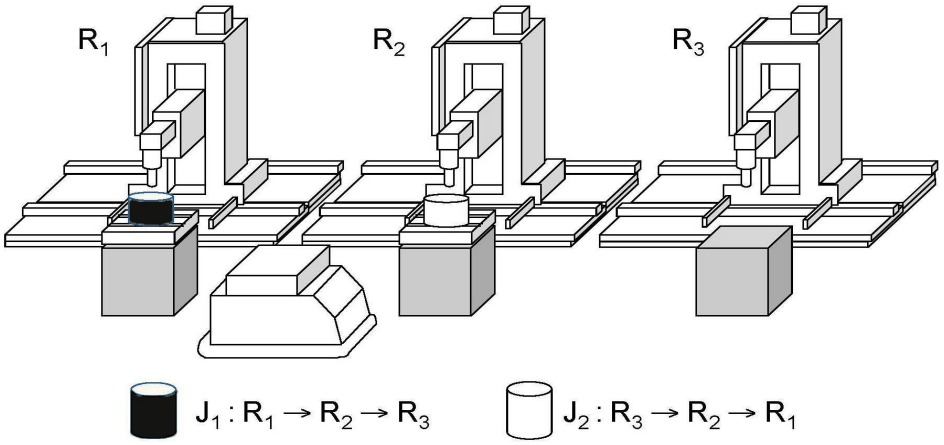


Figure 6.1: Deadlock formation in a flexibly automated manufacturing cell

a focus on operational efficiency, but also in a way that guarantees the (logical) correctness of its behavior. In fact, it can be argued that in the emerging regime of flexible automation, behavioral correctness becomes an even more crucial requirement, since it essentially implies a notion of "robustness" and "stability" that is indispensable for defining and supporting any further notions of "efficiency".

In order exemplify and concretize the positions stated in the previous paragraph, next we introduce the problem of *deadlock avoidance*, in one of its manifestations in the context of flexibly automated manufacturing systems. This problem will also be the main focus of study in the rest of this chapter. Hence, consider the flexibly automated manufacturing cell that is depicted in Figure 6.1. This cell consists of three NC machines, labeled R_1, R_2 and R_3. Each machine is able to accommodate one workpiece at its corresponding worktable, and it possesses no additional buffer. In its current configuration, this cell supports two job types. The process plan of job type J_1 consists of three operations, executed consecutively on machines R_1, R_2 and R_3. The process plan of job type J_2 also consists of three operations, but the corresponding sequence is from machine R_3 to R_2 to R_1. Jobs are transferred among the different machines and the I/O station of the cell by the depicted automated guided vehicle (AGV); this is essentially a mobile robot that moves over a pre-specified guidepath network that connects the cell workstations and its I/O station. Figure 6.1 depicts a state where an instance of job type J_1 is loaded on machine R_1 for the execution of its first processing stage, and an instance

of type J_2 is loaded on machine R_2 for the execution of its second processing stage. But then it should be clear that the further advancement of these two jobs will be permanently stalled, since no matter which of the two jobs will be picked first by the AGV for its transfer to the next station, the unloading of that job will be blocked by the job located at the destined station. Such a permanent blocking situation is characterized as a *deadlock*, and it has a very pernicious impact upon the performance of the underlying system; in automated production systems, detection and resolution of deadlocks typically necessitates external interference, while the utilization of the resources involved in a deadlock is effectively zero. It is therefore imperative that an automated production system has the capability to predict and prevent the formation of deadlock in an effective manner. In the rest of this chapter we shall discuss some of the theory and tools that are currently available for the investigation of the problem of deadlock in flexibly automated production systems, and for the design of effectively implementable control policies that will guarantee the deadlock-free operation of these environments, without compromising their operational efficiency and flexibility.

As already mentioned, deadlock formation and its avoidance constitutes a *behavioral* (or *logical*) control problem that arises in the considered production systems, and as such, it complements the more typical *performance-oriented* control problems that have been studied in the past. From a more technical standpoint, these characterizations imply that for the analysis of the deadlock avoidance problem, the system "behavior" is manifested by the executions of various events in it and by the sequence in which these events are taking place, but the exact timing of the execution of these event sequences is not very relevant. Indeed, as it can be verified in the example of the previous paragraph, the formation of deadlock depends upon the sequence of the job loadings and their advancement among their processing stages, and not on the exact timing of the occurrence of these events. The modeling, analysis and control of system behavior that is manifested as (untimed) event sequences is studied by an area of modern control theory that is known as *(Qualitative) Discrete Event Systems (DES)* theory [7]. In the DES paradigm, the events of interest define the *alphabet* that characterizes the system behavior, while the various manifestations of this behavior as sequences – or *strings* – of events constitute the *(formal) language* that is generated by the system. The theory is

complemented by additional formal models that provide a characterization of the dynamics that generate the considered behaviors, and connect these dynamics to the structural attributes of the underlying system. *Finite State Automata (FSA)* [50] and *Petri nets (PNs)* [36] are some of the most extensively used such formal models. Finally, the relevant control policies are frequently characterized as *supervisory control (SC)* policies (or simply as *supervisors*) [50].

The rest of this chapter employs the FSA modeling framework for a systematic investigation of the behavioral problem of deadlock avoidance in flexibly automated production systems.[1] Recognizing that, in the considered operational context, deadlock essentially arises from the impertinent allocation of the finite system resources to the jobs that are contesting for their acquisition,[2] we shall formulate and study the problem through the abstracting notion of the *resource allocation system (RAS)* [54]. This is a concept that captures all the relevant behavioral aspects of a flexibly automated manufacturing system, and at the same time, it enables the generalization and extension of the derived results to many other operational domains that share similar resource allocation problems. Hence, the rest of this chapter is organized as follows: Section 6.2 starts with a formal characterization of the RAS concept. That section also models the RAS behavior as an FSA, and introduces formally the concepts of deadlock and deadlock avoidance by means of the FSA modeling framework. The last part of Section 6.2 introduces a notion of optimality in the context of RAS deadlock avoidance, that is defined on the basis of maximal permissiveness. It is shown, however, that the deployment of the maximally permissive deadlock avoidance policy (DAP) is an NP-Hard problem for many practical RAS instantiations. Hence, the section concludes with a discussion of the various ways in which the relevant research community has tried to address this negative complexity result, while establishing a trade-off between computational tractability and operational efficiency for the deployed policies. Section 6.3 reports on the results of a more recent research program that has sought to deploy the maximally permissive DAP even for cases where it is known to

[1] The investigation of this problem by means of the PN modeling framework and of an alternative modeling framework that employs a graph-theoretic representation of the system structure and dynamics, is addressed respectively in Chapters 7 and 8 of this edition.

[2] As demonstrated by the example system of Figure 6.1, a key resource that necessitates careful allocation is the finite buffering capacity of the system workstations. But similar deadlock-related problems can arise from the allocation of auxiliary resources likes fixtures and processing tools. Furthermore, the allocation of the traveling segments – typically known as *"zones"* – in the guidepath network of an AGV or monorail system can also be susceptible to deadlocks.

be NP-Hard. The basic ideas that define this research program is the confinement of the most expensive part of the involved computation in an "off-line" stage, and the translation of the derived results to a more parsimonious representation of the obtained policy that will be employed in the final, "on-line" implementation of this policy. It will be shown that the obtained results are especially promising, and, in fact, they define a "paradigm" for the practical implementation of maximally permissive DES SC policies in even broader application settings. Section 6.4 concludes the chapter by summarizing its key theses and suggesting some directions for future work. Finally, Appendix A.1 provides a brief introduction of the FSA modeling framework, focusing primarily on the concepts that are necessary for the presented developments.

6.2 Sequential Resource Allocation Systems and the Problem of Deadlock Avoidance

6.2.1 Sequential Resource Allocation Systems

For the purposes of this work, a sequential *resource allocation system (RAS)* will be defined as follows:

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

1. $\mathcal{R} = \{R_1, \ldots, R_m\}$ *is the set of the system* resource types*.*

2. $C : \mathcal{R} \to \mathbb{Z}^+$ *– the set of strictly positive integers – is the system* capacity *function, characterizing the number of identical units from each resource type available in the system. Resources are assumed to be* reusable*, i.e., each allocation cycle does not affect their functional status or subsequent availability, and therefore,* $C(R_i) \equiv C_i$ *constitutes a system* invariant *for each i.*

3. $\mathcal{P} = \{\Pi_1, \ldots, \Pi_n\}$ *denotes the set of the system* process types *supported by the considered system configuration. Each process type* Π_j *is a composite element itself, in particular,* $\Pi_j =< \Delta_j, \mathcal{G}_j >$*, 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* acyclic digraph *with its node set,* Q_j*, being bijectively related to the set* Δ_j*. Let* $Q_j^{\nearrow}$ *(resp.,* $Q_j^{\searrow}$*) denote the set of* source *(resp.,* sink*) nodes of* $\mathcal{G}_j$*. Then, any* path *from some node* $q_s \in Q_j^{\nearrow}$ *to some node* $q_f \in Q_j^{\searrow}$ *defines a* process plan *for process type* Π_j*. Also, in the following, we shall let* $\Delta \equiv \bigcup_{j=1}^{n} \Delta_j$ *and* $\xi \equiv |\Delta|$.

4. $D : \Delta \to \prod_{i=1}^{m} \{0, \ldots, C_i\}$ *is the* resource allocation function *associating every processing stage* Ξ_{jk} *with the* resource allocation vector $D(\Xi_{ij})$ *required for its execution; it is further assumed that* $D(\Xi_{ij}) \neq \mathbf{0}$, $\forall, i, j$*. At any point in time, the system contains a certain number of (possibly zero) instances of each process type that execute one of the corresponding processing stages. A process instance executing a non-terminal stage* $\Xi_{ij} \in Q_i \backslash Q_i^{\searrow}$*, must first be allocated the resource differential* $(D(\Xi_{i,j+1}) - D(\Xi_{ij}))^{+}$ *in order to advance to (some of) its next stage(s)* $\Xi_{i,j+1}$*, and only then will it release the resource units* $|(D(\Xi_{i,j+1}) - D(\Xi_{ij}))^{-}|$*, that are not needed anymore. The considered resource allocation protocol further requires that no resource type* $R_i \in \mathcal{R}$ *be over-allocated with respect to its capacity* C_i *at any point in time.*

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

A more general definition of the RAS concept is provided in [54]. The basic differences between Definition 1 and the RAS definition of [54] can be summarized as follows: First of all, the complete definition of a RAS, according to [54], involves an additional component that characterizes the time-based – or *quantitative* – dynamics of the RAS; but, as explained in the introductory section, this component is not relevant in the modeling and analysis to be pursued in the following developments, and therefore, it is omitted. Furthermore, the process-defining logic supported by Definition 1 encompasses the operational feature of routing flexibility, but it excludes the possibility of merging and splitting operations, that would correspond respectively to assembly and disassembly operations in a manufacturing setting. More generally, one can classify the various instantiations of the RAS concept into a taxonomy that is defined on the basis of (i) the structural characteristics of the sequential logic that defines the process

routes, and (ii) the complexity of the resource allocation function as expressed by the resource requests that are posed by the various processing stages. Then, the main RAS classes that are identified and supported by the RAS definition of [54] are provided in Table 6.1. The reader should notice that the RAS defined in Definition 1 above, essentially corresponds to the *Disjunctive/Conjunctive (D/C-)* RAS class in the taxonomy of Table 6.1. The discussion and the characterizations that are provided in the rest of this section are immediately extensible to the more complex classes of that taxonomy. On the other hand, the extension of the results of Section 6.3 to the more complex RAS classes of Table 6.1 would necessitate some further care.

Table 6.1: A RAS taxonomy

Based on the structure of the Process Sequential Logic	**Based on the structure of the Resource 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	
Complex: A combination of the above behaviors	**Conjunctive:** An arbitrary number of units from different resources

6.2.2 Modeling the dynamics of the considered RAS as a Finite State Automaton

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

1. The *state set S* consists of ξ-dimensional vectors $\mathbf{s}$. The components $\mathbf{s}[l]$, $l = 1, \ldots, \xi$, of $\mathbf{s}$ are in one-to-one correspondence with the RAS processing stages, and they indicate the number of process instances executing the corresponding stage in the considered RAS

state. Hence, S consists of all the vectors $\mathbf{s} \in (\mathbb{Z}_0^+)^{\xi}$ that further satisfy

$$\forall i = 1,\ldots,m,\ \sum_{l=1}^{\xi} \mathbf{s}[l] \cdot D(\Xi_l)[i] \leq C_i \tag{6.1}$$

where, according to the adopted notation, $D(\Xi_l)[i]$ denotes the allocation request for resource R_i that is posed by stage Ξ_l.[3]

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

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

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

   (c) $E^{\searrow} = \{e_{rp} : \Xi_r \in \bigcup_{j=1}^{n} Q_j^{\searrow},\ p = 0\}$, 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 $\mathbf{s}'[l]$ of the resulting state $\mathbf{s}'$ are given by:

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

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

4. The *feasible event function* $\Gamma : S \to 2^E$ collects, for each state $\mathbf{s} \in S$, the set of events $e \in E$ for which the transition function $f(\mathbf{s}, e)$ is defined (i.e., the resulting state $\mathbf{s}'$ belongs in S). [4]

[3] Following standard practice in DES literature (cf., for instance, the relevant definition in page 8 of [7]), in the rest of this document we will frequently use the terms "space" and "subspace" in order to refer to set S and its various subsets considered in this work. We want to emphasize, however, that S and its various subsets involved in this work are *not* vector spaces in the sense that this term is used in linear algebra since they are not closed to vector addition and scalar multiplication.

[4] The reader should notice that in the considered DFSA, the definition of Γ is immediately induced from the information conveyed by the definition of the state transition function in item #3 above, and therefore, this element is introduced only for completeness and consistency with the general definition of the DFSA concept that is provided in the appendix.

5. The *initial state* $\mathbf{s}_0 = \mathbf{0}$, i.e., the state vector with all its components equal to zero, which corresponds to the situation when the system is empty of any process instances.

6. 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 $S \times E^*$; i.e., for any $\mathbf{s} \in S$ and the empty event string ε,

$$\hat{f}(\mathbf{s}, \varepsilon) = \mathbf{s} \tag{6.2}$$

while for any $\mathbf{s} \in S$, $\sigma \in E^*$ and $e \in E$,

$$\hat{f}(\mathbf{s}, \sigma e) = f(\hat{f}(\mathbf{s}, \sigma), e) \tag{6.3}$$

In Equation 6.3 it is implicitly assumed that $\hat{f}(\mathbf{s}, \sigma e)$ is undefined if any of the one-step transitions that are involved in the right-hand-side recursion are undefined.

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:

$$S_r \equiv \{\mathbf{s} \in S : \exists \sigma \in L(G) \text{ s.t. } \hat{f}(\mathbf{s}_0, \sigma) = \mathbf{s}\} \tag{6.4}$$

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

$$S_s \equiv \{\mathbf{s} \in S : \exists \sigma \in E^* \text{ s.t. } \hat{f}(\mathbf{s}, \sigma) = \mathbf{s}_0\} \tag{6.5}$$

In the following, we shall denote the complements of S_r and S_s with respect to S by $S_{\bar{r}}$ and $S_{\bar{s}}$, respectively, 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.

6.2.3 The target behavior of $G(\Phi)$, the maximally permissive DAP, and its complexity

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

$$\begin{aligned} L_m(G) &\equiv \{\sigma \in L(G) : \hat{f}(\mathbf{s}_0, \sigma) \in S_M\} \\ &= \{\sigma \in L(G) : \hat{f}(\mathbf{s}_0, \sigma) = \mathbf{s}_0\} \end{aligned} \tag{6.6}$$

Equation 6.6, when combined with all the previous definitions, further implies that the set of states that are accessible under $L_m(G)$ is exactly equal to S_{rs}. Hence, we have the following definition of the *maximally permissive deadlock avoidance policy (DAP)* for the considered RAS:

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

In other words, starting from the initial state $\mathbf{s}_0$, a maximally permissive DAP must allow / enable 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 further implies 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}.

As a more concrete example of the various concepts and observations that were introduced in the above discussion, Figure 6.2 depicts the state transition diagram (STD) corresponding to the reachable state space for the RAS that models the buffer allocation taking place in the manufacturing cell of Figure 6.1.[5] In the depicted STD, the resource allocation state

[5]It should be easy to see that in the resource allocation dynamics taking place in the manufacturing cell of Figure 6.1, the AGV acts as the enabling mechanism for the job transfers among the cell workstations and its allocation will be deadlock-free as long as the allocation of the buffering capacity of the system workstation remains deadlock-free. Hence, the allocation of this resource can be safely ignored in the analysis of the resource allocation dynamics of the considered system.

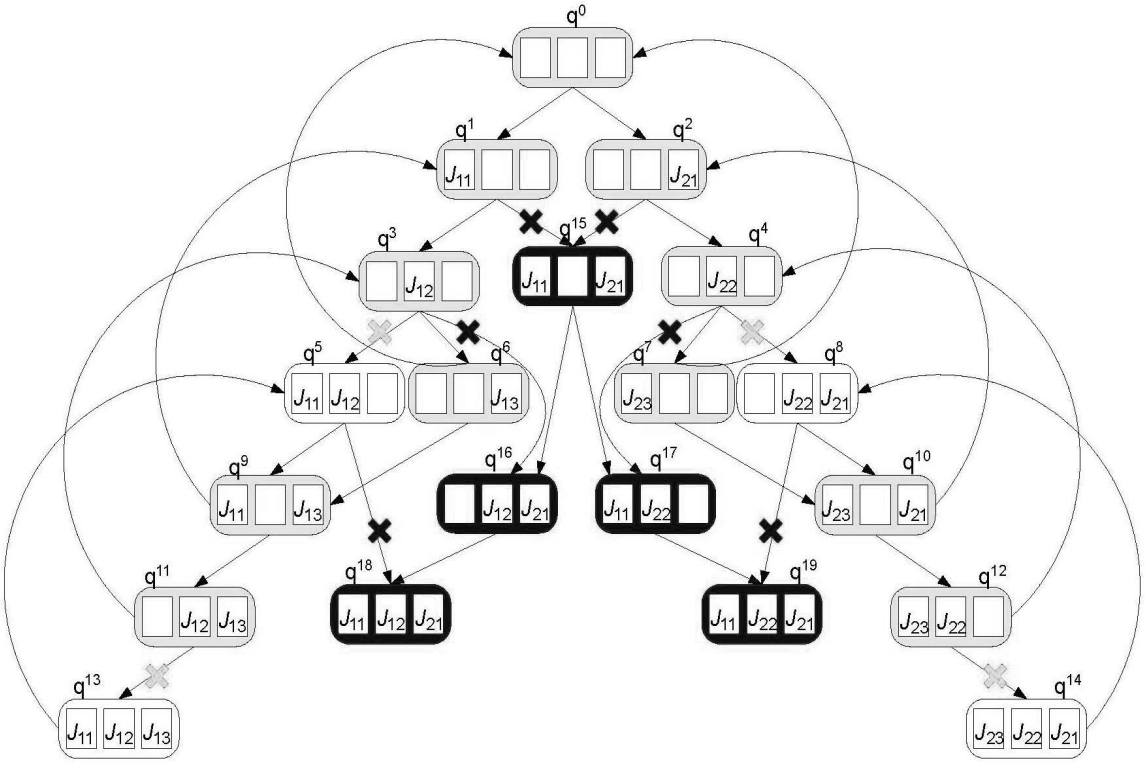


Figure 6.2: The reachable state space and the maximally permissive DAP for the RAS modeling the buffer allocation in the manufacturing cell of Figure 6.1. The figure also depicts, by gray states, the admissible state space of an implementation of the RUN DAP [55] on the considered RAS.

corresponding to each of its nodes is represented graphically, with the three internal rectangles representing the working tables of the three workstations; a non-empty content of these rectangles indicates the processing stage of the job instance that is currently executed on the corresponding machine. Unsafe states are depicted as black states; the remaining states are safe. Clearly, in the depicted case, the maximally permissive DAP can be implemented by an one-step-lookahead scheme that identifies and prevents transitions into the black states; these transitions are annotated by the black crosses in the figure.

Figure 6.2 also demonstrates that the main cause of unsafety in the considered RAS dynamics is the formation of *(partial) deadlock* among a (sub-)set of the running processes. A formal definition of this last concept is as follows:

Definition 3 *A* (partial) deadlock *is a RAS state where a (sub-)set of processes is entangled in a circular waiting pattern with each process in this set requiring, in order to advance to its next processing stage, some resource units that are held by other processes in the set.*

In the STD of Figure 6.2, deadlock states are those corresponding to nodes $q_{16} - q_{19}$. On the other hand, the reader should notice that node q_{15} corresponds to an unsafe RAS state that does not contain a deadlock. For the RAS class considered in this chapter, deadlock detection – i.e., assessing whether a given RAS state contains a deadlock or not – is polynomially decidable w.r.t. the RAS size $|\Phi|$. An efficient deadlock detection algorithm can be found in ([54], Chpt. 2). But assessing the safety of a given state from the considered RAS class is an NP-complete problem. More specifically, we have the following result:

Theorem 1 *[53] Assessing the safety of any given RAS state is NP-complete in the strong sense, even for the subclass of Linear, Single-Unit RAS where every resource has unit capacity.*

Theorem 1 implies that the computation of the maximally permissive DAP is, in general, an NP-Hard undertaking. In the next subsection we overview the main approaches that have been employed by the research community in its efforts to cope with this non-polynomial complexity of the maximally permissive DAP. Subsequently, Section 6.3 focuses on a particular approach that has emerged very recently and seems to hold considerable promise for the effective implementation of the maximally DAP in practical RAS instantiations, even in cases where it is known to be NP-Hard.

6.2.4 Dealing with the NP-Hardness of the maximally permissive DAP: an overview of the past approaches

The research community has tried to circumvent the limitations arising from the negative result of Theorem 1 by pursuing the following directions:

(I) A first direction concerns the identification of "special structure" that allows the deployment of the maximally permissive DAP through algorithms of polynomial complexity w.r.t. the RAS size $|\Phi|$ of Definition 1. Some indicative results can be found in [2, 16, 27, 56, 65]. Furthermore, some of the identified special structure has a very practical, straightforward applicability to the RAS deadlock avoidance problem as manifested in the operational context of the flexibly automated manufacturing systems. It has been found, for instance, that in a Disjunctive, Single-Unit RAS where every resource has at least two units of capacity, the class

of unsafe states is identical to the class of deadlock states, and therefore, the problem of state safety is polynomially decidable in this particular RAS class [56]. When translated in the manufacturing context, this result implies that deadlock-free buffer allocation can be effected in a maximally permissive manner, as long as every workstation has (at least) a two-slot buffer and each job constitutes an atomic entity that requires a single buffer-slot for its accommodation. For a further generalization of this result and an extensive discussion of its implications for the buffer allocation in contemporary manufacturing systems, the reader is referred to [27].

(II) A second direction concerns the development of sub-optimal – i.e., non-maximally permissive – solutions that are based on state properties that are polynomially decidable w.r.t. the RAS size $|\Phi|$. Under these policies, a tentative RAS transition is allowed only if the resultant state satisfies the policy-defining property, which acts as a surrogate characterization of safety. An important challenge in the design of these policies is the specification of this surrogate property so that the reachable subspace satisfying it (i) contains state $\mathbf{s}_0$, and (ii) it constitutes a strongly connected component of the STD representing the dynamics of the underlying RAS; otherwise, the system will experience *policy-induced deadlocks*. Specific policies implementing this general idea can be found in [4, 15, 23, 25, 26, 47, 55]. Furthermore, Figure 6.2 exemplifies this class of policies by depicting, in gray, the subspace that is admitted by an instantiation of the RUN DAP [55] on the corresponding RAS. This policy is clearly sub-optimal since it fails to recognize the safety of states q_5, q_8, q_{13} and q_{14}. On the other hand, the policy is correct since, for every gray node in the depicted STD, there is a directed path to node q_0 that is policy-admissible (i.e., these paths visit only gray nodes).

(III) A third direction seeks to employ alternative, more compact, representations of the considered RAS dynamics in the hope 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 line of research, and therefore, has been explored more persistently in the past, is that of Petri nets [36]. Some indicative works of this approach are those of [14, 24, 29, 47, 59]. Since the RAS structural analysis and the design of effective DAPs by means of the

PN modeling framework is the topic of the next chapter in this edition, we forego any further discussion on this research direction. We notice, however, that the PN-based approaches for the realization of the maximally permissive DAP essentially seek to express this DAP as a set of linear inequalities to be imposed upon the RAS state. But as it is established in the next section, such a representation of the maximally permissive DAP will not be always possible in the considered RAS class. Hence, these approaches are inherently limited in their ability to effectively represent and compute the maximally permissive DAP.[6]

(IV) Another prominent approach that has been developed primarily in the context of PN modeling, but essentially spans, both, the FSA and the PN-based representations, is that of the "theory of regions" [3] and its derivatives. The key problem addressed by the theory of regions is the conversion of a system modeled originally as an FSA into a Petri net where (i) each distinct event is modeled by a single/unique transition and (ii) the reachability graph of the synthesized Petri net is isomorphic to the original FSA. In [19, 61], it is proposed to use the theory of regions to develop a PN-based representation of the maximally permissive DAP. This approach first computes an FSA-based representation of the target DAP, using enumerative FSA-based approaches, borrowed from DES SC theory [7, 50]; the obtained policy is subsequently encoded to a more compact PN model through the theory of regions. The approaches in [19, 61] can find an optimal supervisor if such a supervisor exists. But these approaches are also limited by the aforementioned potential inability to express the maximally permissive DAP as a PN. Furthermore, even in their feasible cases, practical experience has shown that these methods are very demanding from a computational standpoint, and they result in PN representations of the maximally permissive DAP that are much larger than the PN modeling the original RAS.

(V) A more recent approach that is reminiscent of the theory of regions, adopts an adequately compact representation for the underlying RAS in the form of an Extended Finite State Automaton (EFSA) [8]. An EFSA, as presented in [8], is an extension of the ordinary FSA that associates each transition with a "guard" (conditional) formula and "action" functions, including different variables. In this kind of automaton, a transition is enabled *iff* the associated

[6]For a particular RAS class where the PN modeling framework is provably capable of supporting the representation and computation of the maximally permissive DAP, the reader is referred to [30, 31, 32].

guard is satisfied. Furthermore, when a transition fires, the associated action functions are applied, updating the corresponding variables. In [34, 35], the sought DAP is synthesized in the FSA modeling framework using Binary Decision Diagrams (BDDs) in an effort to alleviate the relevant computational effort, and then it is encoded as guarding conditions in the EFSA. However, the scalability and the size of the thus synthesized DAP are problems that yet need to be addressed for the efficient implementation of this approach.

More extensive and comprehensive treatments of many of the results and methodologies that were outlined in the previous paragraphs, can be found in [28, 51, 54, 66]. We conclude our discussion on the key concepts and elements that underlie the current theory on the problem of deadlock avoidance in sequential RAS and the corresponding literature, by noting that the computation of the RAS state space and of an FSA-based representation of the maximally permissive DAP can be facilitated by a number of computational tools that have been developed by the research community of DES SC theory: DESUMA [57], SUPREMICA [1], and TCT [17] are some of them. Yet, it is well-recognized in the field that the enumeration and the representation of the underlying state space suffer from scalability issues even when symbolic techniques are employed to encode the automaton transition function [20, 60]. In recent years, some further attention has been shed upon this issue. For instance, the state space representation based on BDDs [6, 9, 33, 48], the data decision diagrams [12], the hierarchical set decision diagrams [13, 58], the stubborn sets [62], and the sleep-set methods for reduced state space generation [63] are some examples of this endeavor. Nevertheless, it can be safely argued that the deployment of a pertinent and adequate FSA-based representation of a DES dynamics remains a challenging task, and it can always benefit from the special structure that might be present in the considered class of DES. In the next section, we discuss a particular research program that seeks to capitalize upon the aforementioned capabilities, and to complement and extend them in ways that can render the implementation of the maximally permissive DAP effectively computable for a broad range of RAS instances, in spite of the established NP-Hardness for this policy.

6.3 Efficient Implementations of the Optimal DAP through Classification Theory

6.3.1 Preamble

The discussion of Section 6.2 has revealed a number of challenges regarding the deployment of the maximally permissive DAP for the considered RAS class of Definition 1. More specifically, the natural deployment of this policy through the on-line resolution of the state-safety problem in the context of an one-step-lookahead control scheme, is challenged by the NP-completeness of this problem. Furthermore, alternative approaches that try to pre-compute a representation of the target policy in an "off-line" mode and eventually employ this representation for its "on-line" execution, have relied on representations that either are not scalable, and therefore, not amenable to "on-line" parsing, or they are not able to provide an effective representation of the target policy for every instance from the considered RAS class; in the second case, we shall say that the adopted representation is *"incomplete"* (w.r.t. the target policy). In this section we present a research program that has managed to address successfully the aforementioned limitations, leading to effective and efficient implementations of the maximally permissive DAP for pretty sizeable RAS instances. Similar to some of the methods discussed in Section 6.2.4, the approach pursued in this program is decomposed to (i) a first stage where an FSA-based representation of the optimal DAP is computed by standard DES-SC algorithms, and (ii) a second stage where the result of the first stage is "translated" to another, more compact representation to be used during the "on-line" implementation of the policy. However, the alternative representations that are employed for the "on-line" policy implementation are selected so that they are complete, and, as it will be revealed by the subsequent parts of this section, typically they are very compact, as well. In addition, their development can be based on the application of a very rich methodology borrowed from classical optimization theory.

These new representations of the maximally permissive DAP stem from the very basic realization that, during its on-line implementation, the policy acts as a *"classifier"* that dichotomizes the RAS state space S_r into its safe and unsafe subspaces. Once this dichotomy has been characterized through standard SC theory, then, one can resort to concepts and results

borrowed from classification theory [46] for the design of the relevant classifier. Hence, the methods that are pursued in the presented research program open new ways for thinking about the considered problem, that complement effectively 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. The proposed approach selects a particular representation for the sought classifier that is able to provide effective and computationally efficient classification for the RAS class under consideration. In fact, both, *parametric* and *non-parametric* representations for the sought classifier have been developed. Roughly speaking, in the considered research program, a parametric classifier is defined by a set of linear inequalities and/or boolean functions, whereas a non-parametric classifier is defined by a pertinent data structure that stores the information needed for the classification. For parametric classifiers, the classifier-design problem is defined as a *minimization problem* over a certain parameter space that results from the adopted representation. 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 involved in its development, and of analytical bounds that characterize the potential sub-optimality incurred by the use of these heuristics. On the other hand, the design problem for the non-parametric classifier is defined as the *compression* of the stored information in a way that (i) makes the classifier adequately compact, and (ii) facilitates the on-line processing of this information.

From a computational standpoint, the effective resolution of the classifier-design problems that are described in the previous paragraph is further facilitated by some particular structure that is present in the considered dichotomy, and enables the design of the sought classifiers while considering explicitly only a (very) small subset of the underlying RAS state space. A succinct characterization of this structure is given by the following proposition:

Proposition 1 *Consider the (partial) ordering relationship "$\leq$" imposed on the state space S of a given RAS Φ that is defined as follows:*

$$\forall \mathbf{s}, \mathbf{s}' \in S, \;\; \mathbf{s} \leq \mathbf{s}' \;\; \Longleftrightarrow \;\; (\forall l = 1, \ldots \xi, \;\; \mathbf{s}[l] \leq \mathbf{s}'[l]) \tag{6.7}$$

Then,

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

2. $\mathbf{s} \in S_{\bar{s}} \wedge \mathbf{s} \leq \mathbf{s}' \implies \mathbf{s}' \in S_{\bar{s}}$

The validity of Proposition 1 becomes obvious upon the realization that a feasible terminating event sequence for state $\mathbf{s}'$ can be constructed from a terminating event sequence $< \mathbf{s}^{(0)} \equiv \mathbf{s}, e^{(1)}, \mathbf{s}^{(1)}, e^{(2)}, \mathbf{s}^{(2)}, \ldots, \mathbf{s}^{(n-1)}, e^{(n)}, \mathbf{s}^{(n)} \equiv \mathbf{s}_0 >$ for state $\mathbf{s}$ through the removal of those events that correspond to the advancement of job instances that are present in $\mathbf{s}$ but not in $\mathbf{s}'$. From a more practical standpoint, Proposition 1 implies that the necessary information for the classification of any given RAS state $\mathbf{s}$ as safe or unsafe, can be essentially reduced to a characterization of the maximal safe and the minimal unsafe states. These two sets of states are formally defined as follows:

Definition 4 *Let $\mathbf{s} < \mathbf{s}'$ (resp. $\mathbf{s} > \mathbf{s}'$) denote the fact that $\mathbf{s} \leq \mathbf{s}'$ (resp. $\mathbf{s} \geq \mathbf{s}'$) and there is at least a pair of components $\mathbf{s}[l],\ \mathbf{s}'[l]$ for which the corresponding inequality is strict. Then, given a RAS $\Phi = \langle \mathcal{R}, C, \mathcal{P}, D \rangle$,*

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 $\mathbf{s}' < \mathbf{s}$.*

Furthermore, 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 $\bar{S}_{r\bar{s}}$.

In the light of Proposition 1 and Definition 4, the proposed approach of deploying the maximally permissive DAP as a compact classifier effecting the RAS state space dichotomy into its safe and unsafe subspaces, can be pertinently organized according to the workflow that is depicted in Figure 6.3. In the depicted workflow, the tasks that are highlighted in grey color are those to be supported by the "off-line" computation. In the rest of this section,

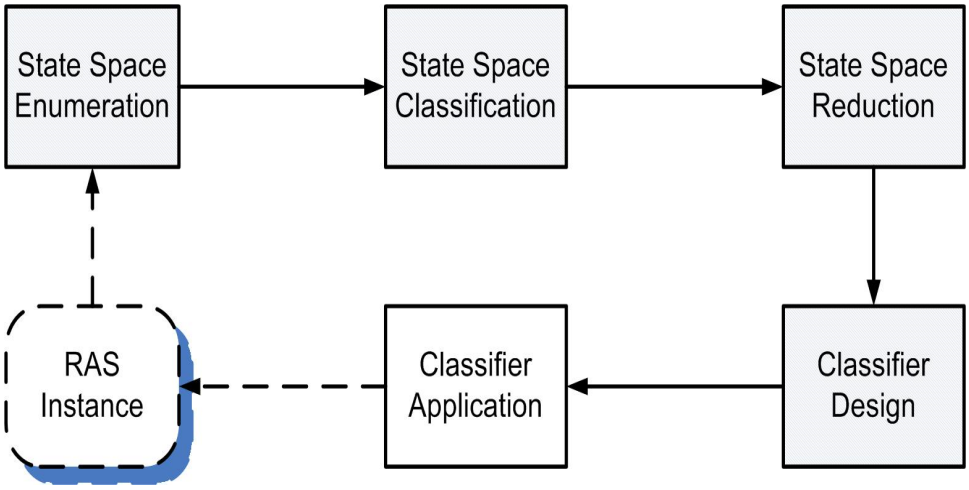


Figure 6.3: Deploying the maximally permissive DAP as a classifier of the RAS state space into its safe and unsafe subspaces.

we outline detailed realizations of this workflow for the construction of, both, parametric and non-parametric classifiers that provide a compact representation to the target DAP.

An important issue, that is at the core of all the realizations of the workflow of Figure 6.3 that are considered in the sequel, is the selection of the particular *structure* that will be employed for the sought classifier. For the case of parametric classifiers, the simplest classification structure is that provided by a system of linear inequalities; such a classifier will be referred to as a *linear* classifier in the sequel. Besides its structural simplicity, a linear classifier is also easier to understand and analyze from the standpoint of structural minimality, and it is also compatible with additional representations of the maximally permissive DAP that have been pursued in the past. More specifically, as remarked in Section 6.2.4, a representation of the maximally permissive DAP as a set of linear inequalities renders it implementable in the PN modeling framework, and endows upon it all the analytical and computational advantages that are possessed by this framework. On the other hand, the discussion of Section 6.2.4 also revealed that the representation of the maximally permissive DAP as a set of linear inequalities is not always feasible in the context of the considered RAS class, and therefore, there is a need for more powerful classification structures, that will be able to provide a complete solution for the classifier-design problem arising in the considered program.

Motivated by the above remarks, the rest of this section is organized as follows: Subsection 6.3.2 first formalizes the notions of the linear classifier and of its structural

minimality, and subsequently it outlines a methodology for the design of a structurally minimal classifier of the maximally permissive DAP. However, as previously explained, this methodology is applicable only in the case that the maximally permissive DAP admits a linear representation. Hence, Subsection 6.3.3 addresses more systematically the conditions under which the maximally permissive DAP admits a linear representation. Subsection 6.3.4 builds upon the developments of the previous two subsections in order to define complete classifiers for the more general case. Finally, Subsection 6.3.5 addresses the design of efficient, non-parametric classifiers for the target policy. Most of the material presented in the rest of this section comes from [37] and a series of publications that were derived from it.

6.3.2 Designing a Linear Classifier for the Optimal DAP

We start with a formalization of the notion of the linear classifier that is employed in this work.

Definition 5 *Consider two vector sets G and H from a ξ-dimensional vector space V.*

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

$$(\forall \mathbf{g} \in G : \ \forall i \in \{1, \cdots, k\}, \ \mathbf{a}_i^T \cdot \mathbf{g} \leq b_i) \quad \wedge$$
$$(\forall \mathbf{h} \in H : \exists i \in \{1, \cdots, k\}, \ \mathbf{a}_i^T \cdot \mathbf{h} > b_i) \tag{6.8}$$

2. *A linear classifier – or separator – for vector sets G and H is* (structurally) minimal*, iff it employs the minimum possible number of linear inequalities that can separate these two sets.*

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

Definition 6 – optimal linear-classifier design *Given a RAS instance Φ that admits a representation of its maximally permissive DAP as a linear classifier of the corresponding sets S_{rs} and $S_{r\bar{s}}$, construct a structurally minimal linear separator for these two sets.* [7]

[7] Definitions 5 and 6 imply an asymmetry for the role of the sets S_{rs} and $S_{r\bar{s}}$ in the specification of the (optimal) linear-classifier design problem. As explained in the following, this asymmetry enables the representation of the obtained classifier in the Petri net formalism.

A major difficulty for the effective construction of the sought classifier is the large (typically huge) cardinality of the sets S_{rs} and $S_{r\bar{s}}$, for any practical instance of the considered RAS classes. In the following, we discuss how the monotonicity property of the pursued dichotomy that was identified in Section 6.3.1, can enable the reduction of the aforestated linear classification problem to another linear classification problem that is defined on a pair of vector sets with a smaller cardinality and dimensionality than their original counterparts. This reduction does not compromise either the effectiveness or the minimality of the derived classifier; a complete formal treatment of these developments is presented in [44], and they are summarized next in a series of "technical results".

Technical Result 1 – 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 6.2, 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. Therefore, the sought classifier needs to separate effectively only the set of reachable safe states S_{rs} and the subset of the reachable unsafe states $S^b_{r\bar{s}} \subseteq S_{r\bar{s}}$ that are reachable from some state $\mathbf{s}' \in S_{rs}$ in a single transition.*

Technical Result 2 – Thinning the sets S_{rs} and $S^b_{r\bar{s}}$ by respectively focusing on their maximal and minimal elements *We remind the reader that $\bar{S}_{rs}$ denotes the maximal elements of the set S_{rs} according to the ordering relationship introduced in Proposition 1. Also, let $\bar{S}^b_{r\bar{s}}$ denote the minimal elements of the set $S^b_{r\bar{s}}$ according to the same relationship. Then, any linear separator $\{(\mathbf{a}_i, b_i),\ i = 1,\ldots,k\}$ for the sets $\bar{S}_{rs}$ and $\bar{S}^b_{r\bar{s}}$ with non-negative parameters $\mathbf{a}_i$ and b_i, $\forall i$, is also an effective separator for the entire sets S_{rs} and $S^b_{r\bar{s}}$. Furthermore, for RAS instances Φ that admit linear separation of their sets S_{rs} and $S^b_{r\bar{s}}$, the set of minimal linear separators for these two sets will always contain a separator $\{(\mathbf{a}_i, b_i),\ i = 1,\ldots,k\}$ with non-negative parameters $\mathbf{a}_i$ and b_i, $\forall i$,[8] and therefore, the restriction of the classification problem of Definition 6 to separators with non-negative parameters compromises neither the feasibility of the problem nor the optimality of the derived solution.*

[8]This result is a consequence of the monotonicity of state safety that is established in Proposition 1 and of the fact that the empty state $\mathbf{s}_0 = \mathbf{0}$ is, by definition, a safe state; its detailed development can be found in [44, 37].

Technical Result 3 – Converting the separation problem of $\bar{S}_{rs}$ and $\bar{S}^b_{r\bar{s}}$ to an equivalent separation problem of reduced dimensionality *Let V denote the ξ-dimensional vector space supporting the vector sets $\bar{S}_{rs}$ and $\bar{S}^b_{r\bar{s}}$, L denote the set of coordinates of space V, and L_0 denote the set of coordinates that are identically zero in the vectors of $\bar{S}^b_{r\bar{s}}$. Also, let P denote the orthogonal projection that removes from the elements of $\bar{S}_{rs}$ and $\bar{S}^b_{r\bar{s}}$ those coordinates that correspond to the aforementioned set L_0. Set $L_P \equiv L \backslash L_0$, and let V_P denote the $|L_P|$-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 L_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, there exists a set of k hyperplanes with non-negative coefficients, W, 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. Furthermore, the separator H can be obtained from separator W as follows:*

$$\begin{aligned} & b_{h_i} := b_{w_i} \ \ \wedge \\ \forall j \in L_0 : \ \mathbf{a}_{h_i}[j] := 0 \ \ \wedge \ \ & \forall j \in L_P : \ \mathbf{a}_{h_i}[j] := \mathbf{a}_{w_i}[\Gamma(j)] \end{aligned} \tag{6.9}$$

Technical Results 1–3 imply that one can construct the sought separator H for the state sets S_{rs} and $S^b_{r\bar{s}}$ by first extracting the smaller vector sets (in terms of, both, cardinality and dimensionality) $P(\bar{S}_{rs})$ and $P(\bar{S}^b_{r\bar{s}})$, designing a linear separator W with non-negative coefficients for the last two sets, and finally obtaining H through Equation 6.9. In fact, the projected set $P(\bar{S}_{rs})$ can be further "thinned" to set $\overline{P(\bar{S}_{rs})}$ by recognizing that for the set $\bar{S}_{rs}$ the aforestated projection P can create additional dominance among the elements of $P(\bar{S}_{rs})$ according to the ordering relationship of Proposition 1; set $\overline{P(\bar{S}_{rs})}$ is obtained from set $P(\bar{S}_{rs})$ by identifying and removing the dominated elements. The entire workflow described above is depicted in the flowchart of Figure 6.4, which refines the corresponding part in the flowchart of Figure 6.3 for the particular case of linear classification.

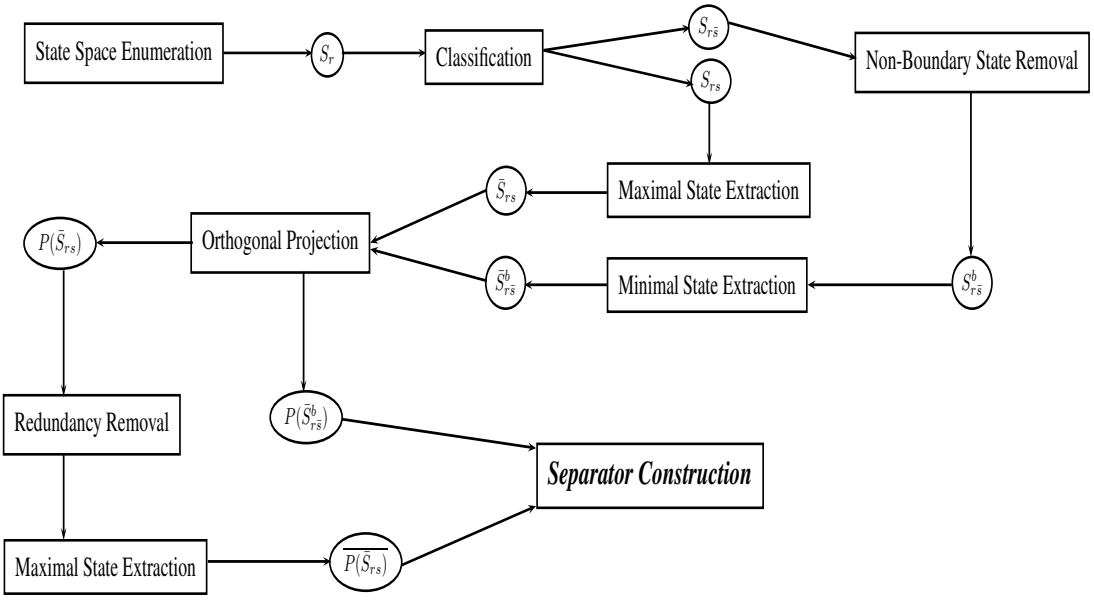


Figure 6.4: The workflow proposed in [44] for the construction of a minimal linear classifier implementing the maximally permissive DAP of any given RAS Φ (provided that this policy is amenable to such an implementation).

Detailed algorithms that can support the (off-line) execution of the various pre-processing stages in this flowchart while mitigating the computational difficulties that arise from the typically large size of the underlying state spaces, can be found in [39].[9] The last step in the workflow of Figure 6.4 – i.e., the synthesis of the sought classifier – can be formulated and solved as a mixed integer program (MIP); we refer to [44] for the relevant details. The same work also provides an efficient heuristic that can be applied in the cases that the proposed MIP is intractable; in particular, it is shown that the (structural) sub-optimality of the classifier that is returned by this heuristic is bounded by a factor of $\ln|P(\bar{S}^b_{r\bar{s}})|$.

Table 6.2 demonstrates the application of the workflow of Figure 6.4 for the optimal linear-classifier design, and concretizes further the concepts and the results that were discussed in the previous paragraphs, by applying this workflow to the design of an optimal linear classifier for the RAS that abstracts the buffer space allocation taking place in the manufacturing system of Figure 6.1. Furthermore, the application of the MIP formulation of [44] upon the sets $\overline{P(\bar{S}_{rs})}$ and $P(\bar{S}^b_{r\bar{s}})$ of Table 6.2 returned a minimal linear classifier for these two sets – and by means of Technical Result 3, for the entire sets S_{rs} and S^b_{rs} – which consists

[9] We briefly revisit this issue in Section 6.3.5.

Table 6.2: The various sets obtained by the application of the data-thinning process of Figure 6.4 to the sets S_{rs} and $S_{rs\bar{}}$ corresponding to the reachable safe and unsafe subspaces of the RAS that abstracts the buffer space allocation taking place in the manufacturing system of Figure 6.1.

$S^b_{rs\bar{}} = S_{rs\bar{}}$
$\bar{S}_{rs} = \{q^{13} \equiv [1\ 1\ 1\ 0\ 0\ 0]^T,\ q^{14} \equiv [0\ 0\ 0\ 1\ 1\ 1]^T\}$ $\bar{S}^b_{rs\bar{}} = \{q^{15} \equiv [1\ 0\ 0\ 1\ 0\ 0]^T,\ q^{16} \equiv [0\ 1\ 0\ 1\ 0\ 0]^T,$ $q^{17} \equiv [1\ 0\ 0\ 0\ 1\ 0]^T\}$
$L_0 = \{3,\ 6\};\quad L_P = \{1,\ 2,\ 4,\ 5\}$ $P(\bar{S}^b_{rs\bar{}}) = \{[1\ 0\ 1\ 0]^T,\ [0\ 1\ 1\ 0]^T,\ [1\ 0\ 0\ 1]^T\}$ $\overline{P(\bar{S}_{rs})} = P(\bar{S}_{rs}) = \{[1\ 1\ 0\ 0]^T,\ [0\ 0\ 1\ 1]^T\}$

of the following two linear inequalities:

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

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

It should be clear from Definition 5 and the example that was provided in the previous paragraph, that each linear inequality employed by the sought classifier acts as a "cover" for the subset of unsafe states that are separated by it, and therefore, the entire problem of the optimal linear-classifier design has a strong affinity to the well known set-covering problem [64]. This affinity has motivated the heuristic of [44] that was mentioned above, and it is further exploited in [11], where an alternative method for the computation of the minimal linear classifier is proposed, based on set-covering concepts and techniques that are borrowed from combinatorial optimization theory [45]. In the next section, we shall return to the identified affinity between the considered problem and the set covering problem, in order to develop the conditions that characterize the existence of a linear classifier for the representation of the maximally permissive DAP of any given RAS instance Φ.

We conclude the discussion of this section by providing, in Table 6.3, some indicative results that were obtained through a set of computational experiments on a subclass of the

Linear, Conjunctive-RAS, where every resource possesses unit capacity.[10] The presented results demonstrate very clearly the computational tractability of the considered approach (whenever applicable) and the compactness that is attained in the representation of the optimal DAP.

Table 6.3: Some computational results characterizing the compression attained by the encoding of the maximally permissive DAP as a linear classifier. The considered RAS instances have been suggested by the Gadara project [30].

state dim.	# safe states	# unsafe states	reduced state dim.	proj. max safe states	proj. min unsafe states	# lin. inqlties
61	8,696,502	71,677	7	9	5	1
75	5,699,463	268,807	23	315	43	4
107	5,696,776	1,165,958	35	533	155	9
60	3,994,272	348,576	10	22	12	2
97	3,718,540	706,177	31	418	122	7
112	2,521,030	556,743	38	643	168	8
63	1,567,434	17,579	17	163	29	3
96	1,240,726	188,189	33	467	113	8

6.3.3 Determining the Existence of a Linear Classifier

It should be clear from the discussion of the previous section that, given an instance Φ from the considered RAS class, there will exist a linear classifier for its maximally permissive DAP, of the type specified in Definition 5, *iff* every unsafe state $\mathbf{u} \in S^b_{rs\bar{s}}$ is linearly separable from the elements of the safe subspace S_{rs}. But it is well known that a given vector $\mathbf{u}$ is linearly separable from a vector set $V = \{\mathbf{v}_i, \ldots, \mathbf{v}_n\}$ *iff* $\mathbf{u}$ does not belong in the convex hull of V (i.e., $\mathbf{u}$ cannot be expressed as a convex combination of the elements of V) [45]. Things can be a little more complicated, in general, when the sought classifier must contain nonnegative parameters only. The next set of results, adapted from [10], formalizes the above remarks, and also provides a complete set of conditions for the existence of linear classifiers with nonnegative parameters. To formally state these results, let $Conv(S_{rs})$ denote the convex hull of S_{rs}. Also, define $\overline{Conv(S_{rs})} \equiv \bigcup_{\mathbf{y} \in Conv(S_{rs})} \{\mathbf{x} \in \mathbb{R}^{\xi} : \mathbf{x} \leq \mathbf{y}\}$. Then, we have the following proposition:

[10] As discussed in the next subsection, the restriction of the RAS class of Definition 1 to unit resource capacities guarantees the linear representation of the maximally permissive DAP.

Table 6.4: An example RAS that does not admit a linear representation for its optimal DAP.

Resource Types	$\{R_1, R_2\}$
Resource Capacities	$C(R_1) = C(R_2) = 2$
Process Types	$\{J_1, J_2\}$
Process Routes	$\mathcal{G}_1: \ R_1 \rightarrow 2.R_2$
	$\mathcal{G}_2: \ R_2 \rightarrow 2.R_1$

Proposition 2 *Consider a RAS instance Φ from the RAS class of Definition 1, and the corresponding sets S_{rs} and $S^b_{r\bar{s}}$ of the underlying state space. Then, the following statements hold true:*

1. *There exists a linear classifier of the form defined by Definition 5 that can represent the maximally permissive DAP of Φ iff $Conv(S_{rs}) \cap S^b_{r\bar{s}} = \emptyset$.*

2. *There exists a linear classifier of the form defined by Definition 5 with nonnegative parameters that can represent the maximally permissive DAP of Φ iff $\overline{Conv(S_{rs})} \cap S^b_{r\bar{s}} = \emptyset$.*

3. *For the considered RAS class, it also holds that $\overline{Conv(S_{rs})} \cap (\mathtt{R}^+_0)^\xi = Conv(S_{rs})$, and therefore, the two tests in items (1) and (2) above are equivalent.*

Some further characterizations and an algorithmic construction of $\overline{Conv(S_{rs})}$ are provided in [52]. On the other hand, a simple (although computationally intensive) test for assessing the condition $\overline{Conv(S_{rs})} \cap S^b_{r\bar{s}} = \emptyset$ can be based on the assessment, for each state $\mathbf{u} \in S^b_{r\bar{s}}$, of the feasibility in $\mathbf{a}, b$ of the following system of linear inequalities :

$$\begin{aligned} \forall \mathbf{s} \in S_{rs}, \quad \mathbf{a} \cdot \mathbf{s} &\leq b \\ \mathbf{a} \cdot \mathbf{u} &\geq b+1 \\ \mathbf{a},\ b &\geq 0 \end{aligned}$$

Table 6.4 and Figure 6.5 concretize the results Proposition 2 by providing an example RAS Φ that does not admit a representation of its maximally permissive DAP as a linear classifier. More specifically, the RAS configuration defined in Table 6.4 has two resource types, R_1 and R_2, each with capacity $C(R_i) = 2$. It also has two process types, J_1 and J_2, where each

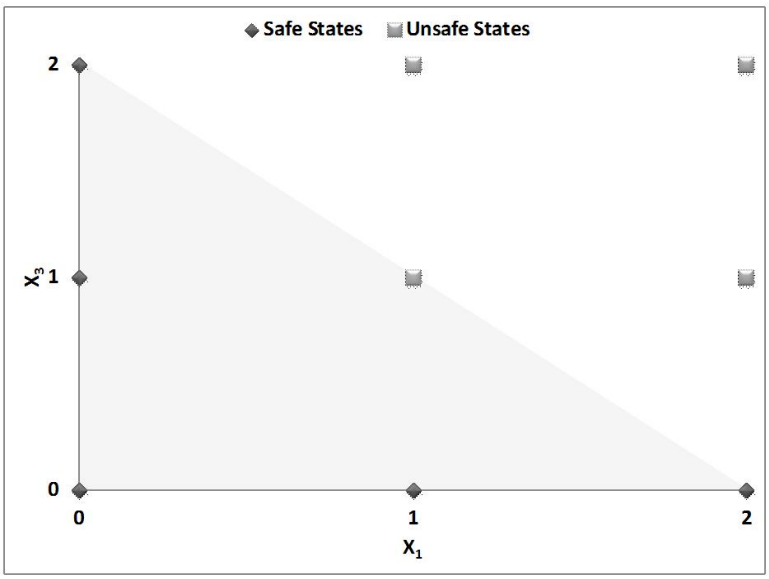


Figure 6.5: Characterization of the safe and unsafe reachable states for the example RAS of Table 6.4, in the projected sub-space defined by the state components x_1 and x_3. Safe reachable states are depicted by rhombi and unsafe reachable states by squares. Also, the shadowed area corresponds to the convex hull of the projected safe states.

process type consists of two processing stages with each stage engaging a single resource type at the amount indicated by the corresponding coefficient. The RAS state $\mathbf{s}$ is defined by the 4-dim vector $[x_1, x_2, x_3, x_4]$ corresponding to the stages Ξ_{11}, Ξ_{12}, Ξ_{21} and Ξ_{22} respectively. It can be easily checked that $S_{rs} \equiv \{[0,0,0,0],\ [1,0,0,0],\ [0,1,0,0],\ [0,0,1,0],\ [0,0,0,1],\ [2,0,0,0],\ [0,0,2,0],\ [1,1,0,0],\ [0,0,1,1]\}$, and that $S^b_{r\bar{s}} \equiv \{[1,0,1,0],\ [2,0,1,0],\ [1,0,2,0]\}$. Since process instances executing the stages Ξ_{12} and Ξ_{22} can immediately exit the system upon their completion, we can study the corresponding deadlock avoidance problem by considering only the projection of the reachable state space to the subspace defined by the components x_1 and x_3;[11] Figure 6.5 provides this characterization. It is easy to see that the boundary unsafe state corresponding to point $[1,1]$ belongs in the convex hull of the points that correspond to the RAS safe states, and therefore, it is not linearly separable from them.

We conclude the discussion on the representation of the maximally permissive DAP for the RAS instances of Definition 1 as a linear classifier, by characterizing two RAS sub-classes for which this representation is always possible. The first of them concerns RAS instantiations where every resource has a single unit of capacity. Since, according to Definition 1, every

[11] This remark is a manifestation of the dimensionality reduction that was discussed in Technical Result 3 of Section 6.3.2.

processing stage in the considered RAS requires at least one unit from some resource type, the above restriction on the RAS capacities further implies that the resulting RAS sub-class has binary state spaces. But it can be easily checked that every "corner" (more formally, extreme point) of the ξ-dimensional binary hypercube that encompasses these state spaces, can be separated from the rest by a single hyperplane. The reader is referred to [44] for a more formal statement of this result. Some further implications of it are explored in [30] and the references that are cited therein.

A second RAS sub-class for which the maximally permissive DAP is always representable as a linear classifier, is the subclass of Disjunctive, Single-Unit RAS where the set of unsafe states coincides with the set of deadlock states. We remind the reader that some specific RAS sub-classes that satisfy this property, were identified and discussed in Section 6.2.4. It can be easily seen that, in the case of Single-Unit RAS, the prevention of any (reachable) deadlock can be achieved in a maximally permissive manner through the enforcement of a linear inequality stipulating that the total number of process instances executing the processing stages that are involved in the considered deadlock remains strictly below the combined capacity of the resources that are involved in the deadlock; this last remark establishes the claim that was made at the beginning of this paragraph.[12]

6.3.4 Complete Parametric Classifiers for the Optimal DAP

In this section we seek to characterize classifier structures that are complete w.r.t. maximally permissive DAP of the entire RAS class of Definition 1; i.e., these classifiers will be able to provide an effective (and hopefully compact) representation for the RAS state space dichotomy that is effected by the maximally permissive DAP of any instance Φ from the considered RAS class. The existence of such a classifier is guaranteed by a well-known result in classification theory that establishes the effective separation of two distinct, finite vector sets from any ξ-dim space V, through a particular classification mechanism that is known as a *"(generalized) committee machine (CM)"* [46]. This classifier is defined by a 4-tuple $CM = \langle \mathbf{A}, \mathbf{b}, \pi, \theta \rangle$, where:

[12] However, the practical significance of this last result is mitigated, to a certain extent, by the availability for these RAS sub-classes of polynomial one-step-lookahead implementations of the optimal DAP that guard against the formation of deadlock through "on-line" deadlock detection algorithms; c.f., the relevant discussion in Section 6.2.4.

(i) $\mathbf{A}$ is a real-valued matrix of some dimensionality $k \times \xi$; (ii) $\mathbf{b}$ and π are real-valued k-dim vectors; and (iii) θ is a real-valued scalar. This classifier accepts a vector $\mathbf{v} \in V$ *iff*

$$\sum_{i=1}^{k} \pi[i] \cdot I_{\{\mathbf{A}[i,\cdot]\cdot\mathbf{v} \leq \mathbf{b}[i]\}} \leq \theta \tag{6.10}$$

In Equation 6.10, $I_{\{\mathbf{A}[i,\cdot]\cdot\mathbf{v} \leq \mathbf{b}[i]\}}$ is an indicator variable that is priced to one *iff* $\mathbf{A}[i,\cdot] \cdot \mathbf{v} \leq \mathbf{b}[i]$. Hence, a CM can be perceived as a two-layered structure. The first layer of this structure consists of the linear inequalities that are defined by the pair $\langle \mathbf{A}, \mathbf{b} \rangle$ and it constitutes a mapping mechanism that maps the ξ-dim input vector $\mathbf{v}$ to another k-dim binary vector I consisting of the indicator variables that appear in Equation 6.10. The second layer accepts or rejects the input vector $\mathbf{v}$ based on the satisfaction by its image vector I of the linear inequality that is defined by the pair $\langle \pi, \theta \rangle$.

However, a direct application of the above result to the classification problem under consideration would be challenged by the excessive size of the classified subspaces. Hence, we stipulate that any practical approach towards the development of the sought classifiers should retain the capability of focusing explicitly only upon the maximal safe and the minimal unsafe states of the underlying state space, that was established for the linear classifier-design problem in Section 6.3.2. Next, we present two variations of the CM classifier that provide this additional capability, while retaining their completeness w.r.t. the target policy that is considered in this work. From a more technical standpoint, both of these variations are enabled by the monotonicity property of Proposition 1, and the topology of the underlying subspaces that is implied by this property; a concrete example of this topology is provided in Figure 6.5.

The first variation is known as the *two-layer classifier*, and its basic structure is depicted in Figure 6.6. As it can be seen in this figure, the two-layer classifier retains the two-layered structure of the CM, but it replaces the original inequality in the second layer by an entire set of linear inequalities. In general, there might be a trade-off among the number of inequalities of the first and the second layer that are necessary for effecting the target classification. Hence, the aforementioned expansion of the second layer intends to allow for more compact representations of the target DAP, and a more economical computation for the "on line" classification of any given RAS state, than those that are supported by the original CM.

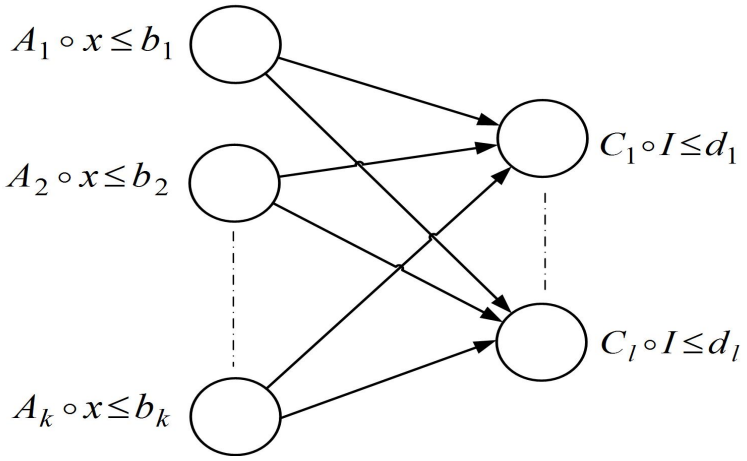


Figure 6.6: A schematic diagram of the two-layer classifier

Furthermore, this expansion, when combined with the monotonicity property of Proposition 1, enables also the restriction of the sought two-layer classifiers to the sub-class of this family that involves only nonnegative coefficients for the classifier inequalities. This last effect subsequently enables the design of the sought classifier based only on the explicit consideration of the maximal safe and the minimal unsafe states, and establishes the practicality of the approach. The reader is referred to [40] for a complete treatment of the corresponding classifier-design problem, along lines similar to those that were pursued for the design of the linear classifier in Section 6.3.2.

The second complete classifier that has been employed for the representation of the maximally permissive DAP by the research program that is considered in this section, is known as the *Boolean classifier*. As indicated by its name, this classifier substitutes the linear inequality in the second layer of the CM by a Boolean function that is defined in terms of the binary components of the image vector I that is produced by the first layer. The motivation for this substitution comes from the realization that, for RAS instances that fail the conditions of Proposition 2, the safe subspace S_{rs} can be represented by the union of a number of polytopes that do not contain any elements of $S^b_{r\bar{s}}$; Figure 6.7 depicts the generic structure of the Boolean classifier.

From a more analytical standpoint, each of these polytopes is characterized as a set of linear inequalities, while their union can be expressed as a disjunctive normal form (DNF)

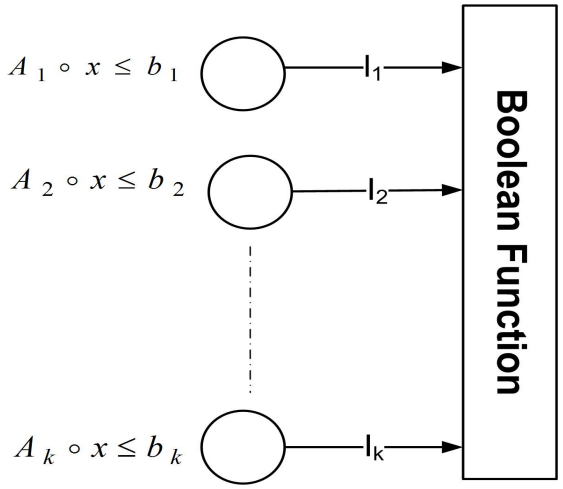


Figure 6.7: A schematic diagram of the Boolean classifier

with the constituent clauses being defined by the conjunctions of the indicator variables that correspond to the inequalities of each polytope. As in the previous cases, the restriction of the parameters of the first-layer inequalities to nonnegative values does not compromise the completeness of the classifier, and it enables its design based only on the explicit consideration of the maximal safe and minimal unsafe states of the underlying state space. A complete analysis is presented in [38], where it is also shown that the Boolean function of the second layer can be replaced by alternative forms than the one suggested above, with potential gains in the compactness of the classifier and/or the computational effort for its design. Furthermore, the work of [10] provides additional algorithms for the design of the Boolean classifiers of the DNF form that was described above. These algorithms are based on the embedding of the geometrical perspectives that underlie the specification of the aforementioned DNF form into pertinent "Branch & Bound" schemes, similar to those that are employed by combinatorial optimization theory [45].

6.3.5 Non-Parametric Classifiers for the Optimal DAP

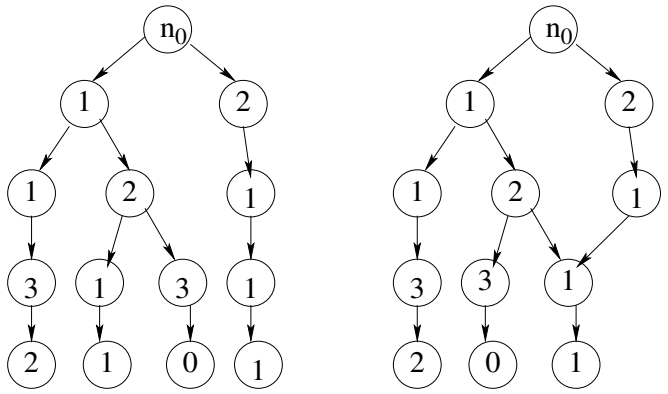
As stated in Section 6.3.1, the key thesis underlying the proposition of this classifier is that, in many practical applications of the considered RAS theory, once the FSA-based representation of the optimal DAP has been obtained, it is possible to encode the part of that representation that is necessary to resolve the underlying state safety problem, in a "data structure / mechanism" sufficiently compact, so that the problem can be effectively addressed within the time and

other resource constraints that typically arise in a real-time computation. More specifically, a straightforward implementation of the aforementioned idea might explicitly store the set $S^b_{r\bar{s}}$, and during the "on-line" stage, it will use this information to test the membership of a given state $\mathbf{s}$ in $S^b_{r\bar{s}}$. Furthermore, using the monotonicity property of Proposition 1, we can explicitly store only $\overline{S}^b_{r\bar{s}}$, i.e., the subset of the minimal elements of $S^b_{r\bar{s}}$; then, starting from the initial state $\mathbf{s}_0$, the transition to a reachable state $\mathbf{s}$ should be blocked as unsafe *iff* there exists a state $\mathbf{u}' \in \overline{S}^b_{r\bar{s}}$ such that $\mathbf{s} \geq \mathbf{u}'$. An equivalent characterization of state safety is as follows: $\mathbf{s}$ is a safe state *iff* $\forall \mathbf{u}' \in \overline{S}^b_{r\bar{s}}$, $\exists i_{u'} \in L$ s.t. $\mathbf{s}[i_{u'}] < \mathbf{u}'[i_{u'}]$. In this last characterization, L denotes the dimension set of the state vector, as defined in Technical Result 3. But from the definition of the dimension set L_0 in the same Technical Result, we can see that the sought $i_{u'} \notin L_0$; hence, the set of dimensions L_0 can be dropped from the stored data set. This last remark further implies that the set $P(\overline{S}^b_{r\bar{s}})$, that is obtained by applying the workflow depicted in Figure 6.4, contains sufficient information for the correct classification of any state vector that must be considered during the "on-line" stage.

For a more efficient storage and on-line processing of the set $P(\overline{S}^b_{r\bar{s}})$, we propose to use *(n-ary) decision diagrams* to store this set, which are also known as the *TRIE* data structure [5]. These decision diagrams are an extension of the BDD concept [6] so that they can support the storage and processing of vectors with finite but possibly non-binary support for their components. Given a set of k vectors, an (n-ary) decision diagram employed for their storage is an acyclic digraph with a dummy source node and multiple sink nodes. This digraph is characterized mainly by the following properties: (i) Other than the source node, each node stores the value of a single coordinate of some vector(s) in the given set. (ii) There exist k directed paths from the source node to the sink nodes, and the contents of the nodes on each of these paths correspond to a vector in the given set. Moreover, there is an one-to-one correspondence between these directed paths and the given vectors. (iii) The diagram employs the minimum number of nodes that support properties (i) and (ii).

From an algorithmic standpoint, the (n-ary) decision diagram for the storage of k l-dimensional vectors $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k\}$ is constructed in two stages:

(a) First, an (n-ary) *decision tree* is constructed for the storage of the given vectors as



(a) The decision tree (b) The corresponding decision diagran

Figure 6.8: A decision tree and the corresponding decision diagram storing the vector set $\{[1,2,1,1], [2,1,1,1], [1,1,3,2], [1,2,3,0]\}$.

follows: This tree has a dummy root node, n_0, of depth 0, and l layers of nodes, with depths from 1 to l that correspond to the l dimensions of vectors $\mathbf{v}_i$. Starting with node n_0 as the single node of layer 0, the tree nodes at each of the remaining layers are defined recursively as follows: The children of a node n at layer $l(n) \in \{0,\ldots,l-1\}$ correspond to all the possible values of coordinate $l(n)+1$ in the vector subset of $\{\mathbf{v}_1,\mathbf{v}_2,\ldots,\mathbf{v}_k\}$ that is obtained by fixing the first $l(n)$ coordinates at the values specified by the path from the root node n_0 to node n. The coordinate value that corresponds to each node n in layers 1 to l, according to this node generation scheme, is characterized as the "content" of n. Obviously, the nodes generated for layer l according to the previous recursion have no children, and they constitute the leaf nodes of the tree. In the resulting tree structure, every vector $\mathbf{v}_i,\ i = 1,\ldots,k$, is represented by a path to one of the leaf nodes.

(b) In the second stage, the decision tree is converted to a decision diagram by iteratively identifying and eliminating duplicate sub-graphs in the generated structure, while starting from the last layer l. Two subgraphs – or sub-diagrams – originating at given layer $i \in \{1,\ldots,l\}$ are considered duplicate if (i) they are isomorphic, and (ii) each isomorphically related pair of nodes has the same content.

Figure 6.8 exemplifies the construction of an n-ary decision diagram by depicting the decision tree and the corresponding decision diagram that store the vector set $\{[1,2,1,1],[2,1,1,1],\ [1,1,3,2],[1,2,3,0]\}$. A complete treatment of the application of the TRIE data structure for the efficient representation and deployment of the optimal DAP in the

considered RAS class can be found in [39].

We conclude the discussion of this section by noticing that all the three approaches for the representation and the deployment of the maximally permissive DAP that were discussed in it, reveal clearly that the information that is really necessary for the effective resolution of the safety problem, is contained in some subsets of the safe and unsafe subspaces consisting of extreme elements of these sets. Furthermore, Table 6.3, and additional collective experience that is reported in the various references that are cited in this section, suggest that these critical subsets are significantly (typically, by many orders of magnitude) smaller than the corresponding state spaces. Hence, it is pertinent to explore the possibility of enumerating these target sets while foregoing a complete enumeration of the underlying state space. Such an approach necessitates a systematic investigation of the notions of maximality / minimality w.r.t. the concepts of RAS state safety / unsafety. A study along the aforementioned lines has been undertaken in [41, 42], which provides an efficient algorithm for the enumeration of the minimal unsafe states, based on a backtracing scheme from the set of minimal deadlocks. Furthermore, the companion work of [43] establishes that the ability to identify the minimal unsafe states through a partial enumeration of the underlying RAS state space, localized around the minimal deadlocks, enables the effective deployment of the maximally permissive DAP even for certain RAS classes where the underlying state space is not finite, and therefore, not amenable to the classical enumerative approaches of the DES-SC theory.

6.4 Concluding Remarks

This chapter has addressed the deadlock-related problems that arise in the context of the contemporary flexibly automated manufacturing systems. Starting from the thesis that the behavioral complexity that is exhibited by the aforementioned systems necessitates a structured approach to the modeling, analysis and, eventually, the control of their behavior, the chapter has presented a formal approach for the investigation of the relevant deadlock avoidance problems that is based on (i) the abstracting notion of the resource allocation system, and (ii) the further representation of the RAS dynamics in the FSA modeling framework. This analysis has enabled

a rigorous characterization of the concepts of deadlock and deadlock avoidance policies. It has also led to a notion of optimality for the sought DAPs, in the form of maximal permissiveness. However, it was further established that the deployment of the maximally permissive DAP is an NP-Hard proposition. Hence, the second part of the chapter discussed a number of approaches that have been used by the relevant research community in its endeavor to deal with the computational challenges that arise from this negative result. The presented results demonstrate very clearly that the pertinent identification and the effective exploitation of various problem attributes that define "special structure" for it, are of paramount importance for the effective management of the underlying complexities. They also reveal the gains that can be obtained from a synergistic application of various modeling frameworks and analytical perspectives and tools on the considered problem. In this way, the results that were presented in this chapter can define a "paradigm" for the effective deployment of optimal supervision for other applications that are amenable to DES-SC theory.

At the same time, the overall development of this section also reveals a number of possibilities for the further strengthening, extension, and complementation of the presented material. Some of the most prominent ones concern the extension of the methodology of Section 6.3 to more complex classes of the RAS taxonomy of Table 6.1 than the Disjunctive/Conjuctive RAS, which is the RAS class essentially considered in this chapter. A key challenge for this extension is the identification of pertinent properties, similar to those of Proposition 1, that will enable the effective compression of the information that must be explicitly employed in the synthesis of the sought classifiers. In a similar spirit, one can seek the extension of the results of this chapter to RAS that exhibit uncontrollable and unobservable behavior; in the manufacturing context, such behavior can arise, for instance, from timing restrictions regarding the consecutive execution of certain operations, routing decisions that depend on the inherent process dynamics and are not controllable by the supervisory control logic, or merely by the lack of the necessary sensing and actuating capability that would result in a fully observable and controllable behavior. Finally, an important remaining issue is the integration of the results that were presented in this chapter, and the broader results that are currently available regarding the behavioral control of flexibly automated production systems,

with the complementary set of results that pertain to the performance modeling, analysis and control of these environments. Some further discussion on the research directions that are highlighted above, including some preliminary results and pointers to the relevant literature, can be found in [54].

Appendix A

A.1 Finite State Automata: Some Basic Concepts and Definitions

Among the class of qualitative behavioral models employed by Discrete Event System theory, the most straightforward, and, probably, the most widely used, is the *Finite State Automaton (FSA)* [7]. A formal definition of this model is as follows:

Definition 7 *[7] A* (Deterministic) Finite State Automaton (FSA) *G is a 6-tuple*

$$G = \langle E, S, f, \Gamma, s_0, S_m \rangle$$

where

- E *is a finite set, called the* event set *of the automaton;*
- S *is a finite set, called the* state set *of the automaton;*
- $f: S \times E \to S$, *is the* state transition function, *i.e.,* $\forall s \in S, \forall e \in E,\ f(s,e) = s'$ *means that there is a transition from state* s *to state* s' *that is triggered by event* e*; in general,* f *is a partial function on its domain, i.e., certain events cannot occur in state* s*;*
- $\Gamma : S \to 2^E$ *is the* feasible event function, *i.e.,* $\forall s \in S,\ \Gamma(s)$ *denotes the set of all events* e *for which* $f(s,e)$ *is defined;*
- $s_0 \in S$ *is the* initial state *of the automaton;*

- $S_m \subseteq S$ *is the set of* marked states.

The transitional structure expressed by the FSA model can be visualized by a labeled digraph $\mathcal{G}$; this directed graph is known as the *state transition diagram (STD)* of the FSA, and its node set corresponds to the state set S of the automaton, its edge set is defined by the state transition function, and its edge label set corresponds to the event set E of the automaton.

It is obvious from the above definitions that the FSA and its corresponding STD can be perceived as a complete map for the behavioral evolution of the modelled system. This effect can be formalized as follows:

Definition 8 *[7] Consider an FSA* $G = \langle E, S, f, \Gamma, s_0, S_m \rangle$ *and let* E^* *denote the set containing all the* finite-length *sequences that can be generated from E, including the empty sequence* ε.

1. *The FSA state transition function* f *is naturally extended to* $S \times E^*$ *as follows:*

$$\forall s \in S, \quad f(s, \varepsilon) \equiv s \tag{A.1}$$

$$\forall s \in S, \forall u \in E^*, \forall e \in E, \quad f(s, ue) \equiv f(f(s, u), e) \tag{A.2}$$

2. *The* language $L(G)$ generated *by G is defined by*

$$L(G) \equiv \{u \in E^* : f(s_0, u) \text{ is well-defined}\} \ [1] \tag{A.3}$$

3. *The* language $L_m(G)$ marked *by G is defined by*

$$L_m(G) \equiv \{u \in L(G) : f(s_0, u) \in S_m\} \tag{A.4}$$

In the STD context, $L(G)$ can be described as the set of all event sequences $u \in E^*$ that can be traced on any path, not necessarily simple, starting from the initial state s_0. $L_m(G)$ is used to model event sequences that correspond to the achievement of some "milestone" in the system behavior.

[1] i.e., $f(s_0, u)$ involves only transitions corresponding to feasible events

Bibliography

[1] K. Akesson, M. Fabian, H. Flordal, and R. Malik. SUPREMICA-an integrated environment for verification, synthesis and simulation of discrete event systems. In *Proceedings of the 8th International Workshop on Discrete Event Systems*, pages 384–385. IEEE, 2006.

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

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

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

[5] P. Brass. *Advanced Data Structures*. Cambridge University Press, NY,NY, 2008.

[6] R.E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, 100(8):677–691, 1986.

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

[8] Y.L. Chen and F. Lin. Modeling of discrete event systems using finite state machines with parameters. In *Proceedings of CCA'00*, 2000.

[9] G. Ciardo. Reachability set generation for Petri nets: Can brute force be smart? *Applications and Theory of Petri Nets*, pages 17–34, 2004.

[10] R. Cordone, A. Nazeem, L. Piroddi, and S. Reveliotis. Maximally permissive deadlock avoidance for sequential resource allocation systems using disjunctions of linear classifiers. In *Proceedings of CDC 2012*. IEEE, 2012.

[11] R. Cordone and L. Piroddi. Monitor optimzation in Petri net control. In *Proceedings of the 7th IEEE Conf. on Automation Science and Engineering*, pages 413–418. IEEE, 2011.

[12] J.M. Couvreur, E. Encrenaz, E. Paviot-Adet, D. Poitrenaud, and P.A. Wacrenier. Data decision diagrams for Petri net analysis. *Application and Theory of Petri Nets*, pages 129–158, 2002.

[13] J.M. Couvreur and Y. Thierry-Mieg. Hierarchical decision diagrams to exploit model structure. *Formal Techniques for Networked and Distributed Systems-FORTE*, pages 443–457, 2005.

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

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

[16] M. P. Fanti, B. Maione, S. Mascolo, and B. Turchiano. Event-based feedback control for deadlock avoidance in flexible production systems. *IEEE Trans. on Robotics and Automation*, 13:347–363, 1997.

[17] L. Feng and W.M. Wonham. TCT: A computation tool for supervisory control synthesis. In *Proceedings of the 8th International Workshop on Discrete Event Systems*, pages 388–389. IEEE, 2006.

[18] D. Gerwin. Manufacturing flexibility: A strategic perspective. *Management Science*, 39:395–410, 1993.

[19] A. Ghaffari, N. Rezg, and X. Xie. Design of a live and maximally permissive Petri net controller using the theory of regions. *IEEE Trans. on Robotics & Automation*, 19:137–141, 2003.

[20] P. Gohari and M. W. Wonham. On the complexity of the supervisory control design in the RW framework. *IEEE Trans. on SMC – Part B*, 30:643–652, 2000.

[21] M. P. Groover. *Automation, Production Systems and Computer-Aided Manufacturing*. Prentice Hall, Englewood Cliffs, N.J., 1980.

[22] J. Heizer and B. Render. *Operations Management, (10th ed.)*. Pearson, Upper Saddle River, NJ, 2010.

[23] F. S. Hsieh and S. C. Chang. Dispatching-driven deadlock avoidance controller synthesis for flexible manufacturing systems. *IEEE Trans. on Robotics and Automation*, 10:196–209, 1994.

[24] M. Jeng, X. Xie, and M. Y. Peng. Process nets with resources for manufacturing modeling and their analysis. *IEEE Trans. on Robotics & Automation*, 18:875–889, 2002.

[25] M. Lawley, S. Reveliotis, and P. Ferreira. The application and evaluation of Banker's algorithm for deadlock-free buffer space allocation in flexible manufacturing systems. *Intl. Jrnl. of Flexible Manufacturing Systems*, 10:73–100, 1998.

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

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

[28] 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*, 38:173–188, 2008.

[29] Z.W. Li and M.C. Zhou. Elementary siphons of Petri nets and their application to deadlock prevention in flexible manufacturing systems. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 34(1):38–51, 2004.

[30] H. Liao. *Modeling, Analysis and Control of a Class of Resource Allocation Systems Arising in Concurrent Software*. PhD thesis, University of Michigan, Ann Arbor, MI, 2012.

[31] H. Liao, S. Lafortune, S. Reveliotis, Wang. Y., and S. Mahlke. Optimal liveness-enforcing control of a class of Petri nets arising in multithreaded software. *IEEE Trans. Autom. Control*, (to appear).

[32] H. Liao, Y. Wang, J. Stanley, S. Lafortune, S. Reveliotis, T. Kelly, and S. Mahlke. Robust and adaptive supervisory control of discrete event systems. *IEEE Trans. Control, Systems Technology*, (to appear).

[33] A. Miner and G. Ciardo. Efficient reachability set generation and storage using decision diagrams. *Application and Theory of Petri Nets*, pages 691–691, 1999.

[34] S. Miremadi, K. Akesson, and B. Lennartson. Extraction and representation of a supervisor using guards in extended finite automata. In *Proceedings of the 9th International Workshop on Discrete Event Systems*.

[35] S. Miremadi, K. Akesson, and B. Lennartson. Symbolic computation of reduced guards in supervisory control. *IEEE Transactions on Automation Science and Engineering*, 8(4):754–765, 2011.

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

[37] A. Nazeem. *Designing parsimonious representations of the maximally permissive deadlock avoidance policy for complex resource allocation systems through classification theory*. PhD thesis, Georgia Tech, Atlanta, GA, 2012.

[38] A. Nazeem and S. Reveliotis. Designing maximally permissive deadlock avoidance policies for sequential resource allocation systems through classification theory. In *Proceedings of the 7th IEEE Conf. on Automation Science and Engineering*, pages 405–412. IEEE, 2011.

[39] A. Nazeem and S. Reveliotis. A practical approach for maximally permissive liveness-enforcing supervision of complex resource allocation systems. *IEEE Trans. on Automation Science and Engineering*, 8:766–779, 2011.

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

[41] A. Nazeem and S. Reveliotis. An efficient algorithm for the enumeration of the minimal unsafe states in complex resource allocation systems. In *Proceedings of the 8th IEEE Conf. on Automation Science and Engineering*. IEEE, 2012.

[42] A. Nazeem and S. Reveliotis. Efficient enumeration of minimal unsafe states in complex resource allocation systems. Technical Report (submitted for publication), Georgia Inst. of Technology, 2012.

[43] A. Nazeem and S. Reveliotis. Maximally permissive deadlock avoidance for resource allocation systems with R/W-locks. In *Proceedings of WODES 2012*. IFAC, 2012.

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

[45] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. Wiley, NY, NY, 1988.

[46] N. J. Nilsson. *The Mathematical Foundations of Learning Machines*. Morgan Kaufmann, San Mateo, CA, 1990.

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

[48] E. Pastor, J. Cortadella, and O. Roig. Symbolic analysis of bounded Petri nets. *IEEE Transactions on Computers*, 50(5):432–448, 2001.

[49] D. Pillai. The future of semiconductor maufacturing: Factory integration breakthrough opportunities. *IEEE Robotics & Automation Magazine*, 13-4:16–24, 2006.

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

[51] S. Reveliotis. Algebraic deadlock avoidance policies for sequential resource allocation systems. In M. Lahmar, editor, *Facility Logistics: Approaches and Solutions to Next Generation Challenges*, pages 235–289. Auerbach Publications, 2007.

[52] S. Reveliotis and A. Nazeem. Optimal linear separation of the safe and unsafe subspaces of sequential RAS as a set-covering problem: algorithmic procedures and geometric insights. *SIAM Journal on Control and Optimization*, (under revision).

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

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

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

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

[57] L. Ricker, S. Lafortune, and S. Gene. DESUMA: A tool integrating Giddes and Umdes. In *Proceedings of the 8th International Workshop on Discrete Event Systems*, pages 392–393. IEEE, 2006.

[58] Y. Thierry-Mieg, D. Poitrenaud, A. Hamez, and F. Kordon. Hierarchical set decision diagrams and regular models. *Tools and Algorithms for the Construction and Analysis of Systems*, pages 1–15, 2009.

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

[60] J. Tsitsiklis. On the control of discrete-event dynamical systems. *Mathematics of Control, Signal and Systems*, 2:95–107, 1989.

[61] M. Uzam. An optimal deadlock prevention policy for flexible manufacturing systems using Petri net models with resources and the theory of regions. *Intl. Jrnl of Advanced Manufacturing Technology*, 19:192–208, 2002.

[62] A. Valmari. Stubborn sets for reduced state space generation. *Advances in Petri Nets*, pages 491–515, 1991.

[63] K. Varpaaniemi. Efficient detection of deadlocks in Petri nets. *Helsinki University of Technology, Digital Systems Laboratory Report A*, 26.

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

[65] K. Y. Xing, B. S. Hu, and H. X. Chen. Deadlock avoidance policy for Petri net modeling of flexible manufacturing systems with shared resources. *IEEE Trans. on Aut. Control*, 41:289–295, 1996.

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