Correctness Verification of Generalized Algebraic Deadlock Avoidance Policies through Mathematical Programming

Spyros Reveliotis, Elzbieta Roszkowska and Jin Young Choi

Abstract— **Generalized algebraic deadlock avoidance policies (DAPs) for sequential resource allocation systems (RAS) have recently been proposed as an interesting extension of the class of algebraic DAPs, that maintains the analytical representation and computational simplicity of the latter, while it guarantees completeness with respect to the maximally permissive DAP. The original work of [1] that introduced these policies also provided a design methodology for them, but this methodology is limited by the fact that it necessitates the deployment of the entire state space of the considered RAS. Hence, this paper seeks the development of an alternative computational tool that can support the synthesis of correct generalized algebraic DAPs while controlling the underlying computational complexity. More specifically, the presented correctness verification test possesses the convenient form of a Mixed Integer Programming formulation that employs a number of variables and constraints polynomially related to the size of the underlying RAS, and it can be readily solved through canned optimization software. Furthermore, since generalized algebraic DAPs do not admit a convenient representation in the Petri Net modeling framework, an additional contribution of the presented results is that they effect the migration of the relevant past insights and developments with respect to simpler DAP classes, from the representational framework of Petri nets to that of the Deterministic Finite State Automata.**

Note to Practitioners—**The stable and robust operation of many contemporary technological applications, like flexibly automated manufacturing systems, industrial and urban intelligent transportation systems, and automated workflow management systems, necessitates the deployment of a control mechanism that will prevent the development of deadlock in the underlying resource allocation. Two typical requests posed in the design of these control mechanisms are (i) that they are computationally efficient, so that they are effectively implementable in real time, and (ii) that they impose the minimal restrictions that will ensure deadlock-free operation, preserving, thus, the maximum possible flexibility in the system operation. In a recent work we introduced such a control mechanism, that is known as the class of *generalized algebraic deadlock avoidance policies (DAPs)*, and possesses the desirable properties of (i) admitting a computationally efficient representation and (ii) encompassing the minimally restrictive deadlock avoidance policy for any given system configuration. However, currently we lack a computationally efficient methodology for designing generalized DAPs for any given resource allocation system (RAS). The results presented in this paper seek to cover this need by developing a computationally efficient mechanism for assessing the ability of tentative generalized DAPs to establish the deadlock-free operation of some given RAS. The availability of such a tool will eventually enable the optimized design of generalized DAPs through its embedding in a search-based scheme borrowed from the area of combinatorial optimization; this last issue is part of our current investigations.**

Index Terms—**Deadlock Avoidance, Resource Allocation Systems, Supervisory Control, Discrete Event Systems, Mathematical Programming**

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

E. Roszkowska is with the Institute of Computer Engineering, Control and Robotics, Wroclaw University of Technology, Poland, ekr@pwr.wroc.pl

J. Y. Choi is with the Division of Industrial and Information Systems Engineering, Ajou University, Korea, choijy@ajou.ac.kr

I. Introduction

The problem of deadlock avoidance in sequential resource allocation systems (RAS) is well established and extensively studied in the relevant literature. Generally speaking, the problem concerns the coordinated allocation of a finite set of reusable resources to a set of concurrently executing processes so that circular waiting situations – i.e., situations where a subset of processes wait upon each other for the release of the necessary resources for their advancement – are avoided and each process can proceed to its successful completion. Past work has formally characterized the problem by means of a number of modelling frameworks provided by qualitative Discrete Event Systems (DES) theory [2] – e.g., finite state automata, Petri nets (PN), and various other graph theoretic models – and it has also provided a number of methodologies for the synthesis of the necessary deadlock avoidance policies (DAP's) for various sub-classes of these systems; we refer the reader to [3], [4] for a systematic and comprehensive exposition of all the currently available results.

A DAP class that has received particular attention among those past results is that known as *algebraic*, since the relevant policies seek to ensure the deadlock-freedom of the underlying RAS by confining its operation in a subspace that satisfies a properly chosen set of linear inequalities. This closed-form expression of the policy-defining logic is very convenient during the real-time implementation of the policy, especially in the case that the number of the policy-defining inequalities is a polynomial function of the size of the controlled RAS. Furthermore, the works of [5], [6] have shown that, under a PN-based representation, each of the policy-defining inequalities can be conveniently represented by the super-imposition of a *"monitor"* place on the PN modeling the structure of the original RAS. This representation subsequently facilitates a host of policy analysis and synthesis techniques based on results coming from the structural analysis of the PNs modeling the considered RAS class. More specifically, the current results on algebraic DAPs encompass

1. detailed policies for a number of RAS sub-classes of very practical significance, e.g., [7], [8], [9], [10];
2. a generic methodology for testing the correctness of tentative algebraic DAPs for any given RAS, based on PN structural analysis, e.g., [11], [12], [13];
3. an emerging algebraic theory for interpreting the functionality of algebraic DAPs, e.g., [14], [15]; and
4. a methodology for designing maximally permissive al-

gebraic DAPs through PN reachability analysis and the theory of regions, e.g., [16], [17], [18].

Yet, an inherent limitation of algebraic DAPs is that, due to the linear structure of the employed inequalities, they cannot admit non-convex sub-spaces, and therefore, they might be unable to represent the *maximally permissive* DAP for some given RAS. Motivated by this observation, the work of [1] introduced the class of *generalized algebraic DAPs*, which maintains the analytical representation and computational simplicity of algebraic DAPs, and at the same time, it guarantees completeness with respect to the maximally permissive DAP. The results of [1] also include a design methodology for generalized algebraic DAPs, that essentially constitutes an extension of the results of [18] to this new class of policies (c.f. item (4) in the above list). However, it is well known that design methodologies that are based on the theory of regions necessitate the deployment of the entire reachability space of the underlying plant, and therefore, they are limited by a very high computational complexity. Hence, the work presented in this paper seeks the development of an alternative computational tool that can support the synthesis of correct generalized algebraic DAPs while controlling the underlying computational complexity. More specifically, the presented correctness verification test possesses the convenient form of a Mixed Integer Programming formulation that employs a number of variables and constraints polynomially related to the size of the underlying RAS, and it can be readily solved through canned optimization software. The developed formulation essentially constitutes an extension to the class of generalized algebraic DAPs of similar past results derived for the class of algebraic DAPs (c.f. item (2) in the above list). At the same time, the presented results effect the migration of the earlier developments from the representational framework of Petri nets to that of the Deterministic Finite State Automata, a requirement stipulated by the fact that generalized algebraic DAPs do not admit a convenient representation in the PN modeling framework. In fact, we believe that the translation of the past results in this new representational framework is a significant contribution in itself, since (a) it enables a more profound understanding of the past developments, and (b) it renders them more accessible to the practitioner (by circumventing a set of technical concepts and results that pertain to PN structural analysis).

In the light of the above introduction, the rest of the paper is organized as follows: Section II provides the necessary background for the systematic presentation of the paper contributions. More specifically, this section provides a formal characterization of the RAS class considered in this work, the underlying deadlock avoidance problem, the class of generalized algebraic DAPs as a particular solution to this problem, and a generic DAP correctness criterion, that constitutes the basis for the main developments of the paper. These developments are presented in Section III, which introduces the problem of the correctness verification of generalized algebraic DAPs, and presents a solution to it that is based on the aforementioned criterion for DAP correctness and Mathematical (Mixed Integer) Programming. The closing part of the section also discusses some interesting properties of the derived test in terms of its computational complexity and its relationship to similar past results for simpler DAP classes. Section IV demonstrates the application of the proposed method on an elucidating example, and finally, Section V concludes the paper by highlighting the significance and applicability of the presented contributions, and suggesting directions for future work.

II. Disjunctive / Conjunctive Resource Allocation Systems and Generalized Algebraic DAPs

We begin the discussion of the results presented in this manuscript by providing a rigorous characterization of the considered RAS and their underlying dynamics. For expository purposes, we confine the subsequent discussion to the rather prototypical class of *Disjunctive / Conjunctive (D/C-)* RAS; however, our results can be easily extended to the more complex RAS classes presented in the taxonomy of [3].

A. D/C-RAS specification

For the purposes of this work, a *Disjunctive / Conjunctive Resource Allocation System (D/C-RAS)* is formally defined by a 4-tuple $\Phi =< \mathcal{R}, C, \mathcal{P}, \mathcal{D} >$ where: (i) $\mathcal{R} = \{R_1, \ldots, R_m\}$ is the set of the system *resource types*. (ii) $C : \mathcal{R} \rightarrow Z^+$ – 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 considered to be *reusable*, i.e., each allocation cycle does not affect their functional status or subsequent availability, and therefore, $C(R_i) \equiv C_i$ constitutes a system *invariant* for each i. (iii) $\mathcal{P} = \{\Pi_1, \ldots, \Pi_n\}$ denotes the set of the system *process types* supported by the considered system configuration. Each process type Π_j is a composite element itself, in particular, $\Pi_j =< \mathcal{S}_j, \mathcal{G}_j >$, where: (a) $\mathcal{S}_j = \{\Xi_{j1}, \ldots, \Xi_{j,l(j)}\}$ 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, V_j, being bijectively related to the set $\mathcal{S}_j$. Let $V_j^{\nearrow}$ (resp., $V_j^{\searrow}$) denote the set of *source* (resp., *sink*) nodes of $\mathcal{G}_j$. Then, any *path* from some node $v_s \in V_j^{\nearrow}$ to some node $v_f \in V_j^{\searrow}$ defines a *process plan* for process type Π_j. (iv) $\mathcal{D} : \bigcup_{j=1}^{n} \mathcal{S}_j \rightarrow \prod_{i=1}^{m} \{0, \ldots, C_i\}$ is the *resource allocation function* associating every processing stage Ξ_{jk} with a *resource allocation request* $\mathcal{D}(j,k) \equiv D_{jk}$. More specifically, each D_{jk} is an m-dimensional vector, with its i-th component indicating the number of resource units of resource type R_i necessary to support the execution of stage Ξ_{jk}. Obviously, in a well-defined RAS, $D_{jk}(i) \leq C_i, \ \forall j,k,i$. Furthermore, the resource set D_{jk}, required for the execution of a particular processing stage Ξ_{jk}, is allocated exclusively and non-preemptively to each process instance, and it is released by it only upon the allocation of the resources required for the execution of the subsequent stage. Finally,

$|\Phi| \equiv |\mathcal{R}| + |\bigcup_{j=1}^{n} \mathcal{S}_j| + \sum_{i=1}^{m} C_i$ will be referred to as the *size* of Φ.

B. Feasible behavior of the D/C-RAS

The behavior generated by a D/C-RAS with respect to the underlying resource allocation function can be formally modeled by a *deterministic finite state automaton (DFSA)* [2]. In order to define this automaton, it is convenient to renumber the processing stages of the D/C-RAS with the bijective map $q \equiv \{q(j,k) = k + \sum_{i=1}^{j-1} l(i) : j = 1, \ldots, n;\ k = 1, \ldots, l(j)\}$ and to denote the maximal value, $\sum_{i=1}^{n} l(i)$, of this map, by Q. Then, we have:

Definition 1: The DFSA $G(\Phi) = (S, E, \delta, s_0, S_M)$ abstracting the feasible dynamics of a D/C-RAS $\Phi =< \mathcal{R}, C, \mathcal{P}, \mathcal{D} >$ is defined as follows:

1. The *state set* S consists of vectors s such that each of the components $s(q)$, $q = 1, \ldots, Q$, of s corresponds to the respective processing stage Ξ_q and indicates the number of process instances executing this stage. Hence, S consists of all the vectors $s \in (Z_0^+)^Q$ that further satisfy

$$\forall i = 1, \ldots, m, \quad \sum_{q=1}^{Q} s(q)\, D_q(i) \leq C_i \tag{1}$$

where, according to the adopted notation, $D_q(i)$ denotes the allocation request for resource R_i that is posed by stage Ξ_q.

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

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

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

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

3. For each pair (s, e_{rp}) consider the vector s', with its components $s'(q)$, $q = 1, \ldots, Q$, given by:

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

Then, the *state transition function* $\delta : S \times E \to S$ of the automaton is a partial function, defined only over the subset of $S \times E$ for which the aforementioned vector $s' \in S$, with $\delta(s, e) \equiv s'$.

4. The *initial state* of the automaton corresponds to $s_0 = \mathbf{0}$, which characterizes the situation where the system is empty of any process instances.

5. The set of *marked states*, S_M, is defined by the singleton $S_M = \{s_0\}$, which expresses the intention to execute complete process runs. ◇

The requirement that $s' \in S$ in item (3) above, essentially breaks down to the following two requirements:

1. $\forall q = 1, \ldots, Q,\ \ s'(q) \geq 0$
2. $\forall i = 1, \ldots, m,\ \ \sum_{q=1}^{Q} s'(q)\, D_q(i) \leq C_i$

We shall refer to the first of these requirements as *process-feasibility* and to the second one as *resource-feasibility*. Furthermore, in order to capture state transitions resulting from strings of events $\sigma \in E^*$, we extend inductively the domain of function δ from $S \times E$ to $S \times E^*$ in the following manner:

1. $\forall s \in S,\ \delta(s, \epsilon) = s$, and
2. $\forall s \in S, \forall u \in E^*, \forall e \in E,\ \delta(s, ue) = \delta(\delta(s, u), e)$.

The symbol ϵ used in item (1) above, denotes the *empty string*. On the other hand, in the interpretation of item (2), it is implicitly assumed that the involved single-step transitions correspond to feasible events: i.e., only the state-event pairs for which the original function δ is defined are considered; otherwise, the extended version of δ is undefined on the corresponding state-string pair. Then, we shall say that state s' is *accessible* from state s, or that state s is *coaccessible* to state s', *iff* there exists $\sigma \in E^*$ such that $s' = \delta(s, \sigma)$. In particular, the set of states which is accessible from s_0 will be denoted by S_r and it will be referred to as the set of *accessible* or *reachable* states, and the set of states which are coaccessible to s_0 will be denoted by S_s and it will be referred to as the set of *coaccessible* or *safe* states.

The DFSA-based model of the RAS behavior can be expressed graphically by the *State Transition Diagram* $STD(G)$, i.e., a digraph, whose *nodes* correspond to the state set S, and *edges* correspond to the feasible state transitions.

C. Admissible behavior of the D/C-RAS and Deadlock Avoidance Policies

A major concern in the logical control of (D/C-)RAS is the establishment of *deadlock-free* or *non-blocking* behavior. *Deadlocks* constitute RAS states where there is a set of process instances such that each of its processes, in order to advance, requests the allocation of resources currently held by some other process(es) in the considered set. Their development results from (i) the fact that processes will hold upon their allocated resources in a non-preemptive manner and (ii) the arbitrary structure of the process routes that can give rise to cyclical patterns of resource requests among the various executing processes. In the FSA-based modelling of the RAS operation, the development of deadlocks is manifested by the formation of *strongly connected components* in the system reachable space, S_r, which, however, are not *co-accessible*, i.e., the empty state, s_0, is not reachable from them through any sequence of feasible transitions. The elimination of such problematic behavior from the system dynamics is attained by the superimposition on the system behavior of a *deadlock avoidance policy (DAP)*. A formal definition of this concept, adequate for the needs of the subsequent developments, is as follows:

Definition 2: For the D/C-RAS model $G(\Phi) = (S, E, \delta, s_0, S_M)$, consider a *logical* function $\Delta : S \to \{true, false\}$ and the restriction δ_Δ of the transition function δ, i.e., the function that generates the same values as

δ, but it is only defined for pairs (s, σ) s.t. $s' = \delta(s, \sigma)$ is defined and $\Delta(s') = true$. Function Δ is a *correct* DAP for $G(\Phi)$ *iff* (a) $\Delta(s_0)$=*true* and (b) in the restricted DFSA $G_\Delta = (S, E, \delta_\Delta, s_0, S_M)$, each accessible state s is also coaccessible. $\diamond$

In the above definition, the *accessibility* and *coaccessibility* of state s in system G_Δ are understood in the standard way, i.e., as the respective existence of a string σ s.t. $s = \delta_\Delta(s_0, \sigma)$ and of a string σ' s.t. $s_0 = \delta_\Delta(s, \sigma')$.[1] In plain terms, Definition 2 implies that the state evolution in a RAS $G(\Phi)$ that is supervised by a DAP Δ is guarded by two rules: an event e can occur in state s if the tentative state transition is both *feasible* (i.e., $s' = \delta(s, e)$ is defined) and *admissible* (i.e., $\Delta(s') = true$). Clearly, the least restrictive correct DAP, Δ^*, bounds the state transition function δ to δ_{Δ^*} so that the state transition diagram $STD(G_{\Delta^*})$ is the *maximal* strongly connected component of $STD(G)$ that contains s_0. Such a policy is effectively computable through the *one-step lookahead* scheme that admits a tentative resource allocation if and only if the resulting state s' of $G(\Phi)$ is coaccessible to s_0, or *safe*. However, the corresponding state safety problem is NP-complete [19]. In the light of this result, the research community has sought the development of sub-optimal DAP's that are implementable in polynomial complexity with respect to the underlying RAS size, and yet, efficient, i.e., they manage to admit a large part of the safe states [3]. A typical approach to the design of these policies is the identification of a property $\mathcal{H}(s),\ s \in S$, such that (i) the complexity of testing $\mathcal{H}()$ on the RAS states is polynomial with respect to the RAS size, and (ii) the logical function $\Delta_{H(s)} \equiv I_{\{\mathcal{H}(s)\}}$ defines a correct DAP, according to Definition 2. Next, we introduce the particular class of generalized algebraic DAPs, which is the focus of attention to this work.

D. Generalized Algebraic DAPs

As mentioned in the introduction, the class of generalized algebraic DAPs was recently introduced in [1] as a DAP subclass that combines analytical representation and computational simplicity with the ability to recognize *non-convex* subspaces and therefore, to preserve completeness with respect to the maximally permissive DAP, Δ^*. A formal characterization of these policies is as follows:

Definition 3: Consider a DFSA $G(\Phi) = (S, E, \delta, s_0, S_M)$ and a logical function $\Delta_H(s)$, $s \in S$, such that:

1. The parameter H is a quadruple

$$H =< A, b, \pi, \theta > \tag{2}$$

where A is a real-valued matrix of dimensionality $K \times Q$; b is a real-valued K-dimensional vector; π is a real-valued K-dimensional vector; and θ is a real-valued scalar.

[1]We notice, for completeness, that since $S_M = \{s_0\}$, DAP correctness can also be interpreted as the requirement for *reversibility* of the restricted system G_Δ.

2. $\Delta_H(s) = true$ *iff*

$$\sum_{i=1}^{K} \pi(i) \cdot I_{\{A(i,\cdot)\cdot s \le b(i)\}} \le \theta \tag{3}$$

where $I_{\{A(i,\cdot)\cdot s \le b(i)\}}$ denotes the indicator variable that is priced to one if the inequality $A(i,\cdot) \cdot s \le b(i)$ is satisfied, and to zero, otherwise.

Then, Δ_H is a *generalized algebraic DAP* for RAS Φ *iff* it is a correct DAP for it, according to Definition 2. $\diamond$

Notice that by setting $\pi(i) = -1, \forall i$, and $\theta = -K$, the condition expressed by Equation 3 is equivalent to the requirement $A \cdot s \le b$, and therefore, the class of generalized algebraic DAPs subsumes the original, simpler class of algebraic DAPs [3]. Also, under the reasonable assumption that all the elements of the quadruple H are *rational*, one can obtain another quadruple H' such that all the elements of H' are integer-valued and $\Delta_H(s) = \Delta_{H'}(s)$. Hence, the correctness analysis of these policies can be pursued by considering only integer-valued quadruples H. This observation is exploited in the next section, where we study the development of a mathematical programming formulation that assesses the capability of any tentative quadruple H to define a correct generalized algebraic DAP Δ_H for some given RAS Φ.

E. A criterion for DAP correctness verification

We conclude this section by (re-)stating a criterion for DAP correctness verification that was originally developed in [8] and will be instrumental in the development of the results presented in the subsequent parts of this paper.

Proposition 1: A tentative DAP Δ for a given D/C-RAS Φ satisfies item (b) of Definition 2, if for every state $s \in S\backslash\{s_0\}$ with $\Delta_H(s)$=*true*, there exists a *feasible* and *admissible* event $e_{rp} \in \bar{E} \cup E^{\searrow}$. $\diamond$

In order to see the validity of Proposition 1, first notice that the presumed acyclicity of the graphs $\mathcal{G}_j$, that encode the available process plans for each process Π_j, $j = 1, \ldots, n$, implies that all these process plans are executable in a finite number of steps. But then, starting from a policy-admissible state $s \in S\backslash\{s_0\}$, one can establish the existence of a feasible and admissible event sequence $\sigma \in E^*$ that completes all the remaining workload in that state, by repetitive invocation of the condition implied by the above proposition.

III. Correctness analysis of tentative generalized algebraic DAP's

A. The considered problem and the basic methodology underlying the proposed solution

In this section we consider the following problem: Given a D/C-RAS $\Phi =< \mathcal{R}, C, \mathcal{P}, \mathcal{D} >$ and an *integer-valued* quadruple $H =< A, b, \pi, \theta >$ that defines a *tentative* generalized algebraic DAP, Δ_H, for RAS Φ, we want to assess the correctness of Δ_H for Φ, i.e., according to Definition 2, we want to assess whether

1. $\Delta_H(s_0) = true$, and

2. in the DFSA G_{Δ_H}, modeling the controlled system behavior, every accessible state s is also coaccessible.

Item (1) in the above list can be immediately resolved through the test suggested by item (2) in Definition 3. Next we focus on the more involved issue of assessing item (2), which will be resolved through the criterion established by Proposition 1. More specifically, in the following, we translate the criterion of Proposition 1 into a mathematical programming (MP) formulation that is able to verify the correctness of some tentative generalized algebraic DAP Δ_H for a given D/C-RAS Φ through the pricing of its optimal objective value. The derivation of this formulation will proceed in two steps: First, we shall develop an MP test that will assess the existence of a *feasible* and *admissible* event $e_{rp} \in \bar{E} \cup E^{\searrow}$, in any given state $s \in S$. In the second step, we shall extend the test derived in Step one to the test suggested by Proposition 1, by turning the parameter s into a variable that takes values over the space $\{s \in S \backslash \{s_0\} : \Delta_H(s) = true\}$.

B. Testing the existence of a feasible and admissible non-loading event in a given state $s \in S$

Consider a state $s = [s(i), i = 1, \ldots, Q] \in S$, an event $e_{rp} \in E$, $r \neq 0$, and the tentative next-state s_{rp} given by:

$$s_{rp}(i) = \begin{cases} s(i) - 1 & \text{if} \quad i = r \\ s(i) + 1 & \text{if} \quad i = p \\ s(i) & \text{otherwise} \end{cases} \tag{4}$$

The feasibility of state s_{rp} with respect to the resource-capacity constraints can be analytically characterized as follows:

$\forall j \in 1, \ldots, m$,

$$C_j - \sum_{q=1}^{Q} s_{rp}(q)\, D_q(j) \;\geq\; (1 - y_j^{rp})\, L1_j^{rp} \tag{5}$$

$$C_j - \sum_{q=1}^{Q} s_{rp}(q)\, D_q(j) \;\leq\; y_j^{rp}\, U1_j^{rp} - 1 \tag{6}$$

$$f_{rp}^1 \;\leq\; y_j^{rp} \tag{7}$$

$$f_{rp}^1 + (m-1) \;\geq\; \sum_{j=1}^{m} y_j^{rp} \tag{8}$$

$$y_j^{rp}, f_{rp}^1 \;\in\; \{0,1\} \tag{9}$$

The parameter $L1_j^{rp}$ (resp., $U1_j^{rp}$), $j = 1, \ldots, m$, denotes a lower (resp., upper) bound for the quantity $C_j - \sum_{q=1}^{Q} s_{rp}(q)\, D_q(j)$ (resp., $C_j - \sum_{q=1}^{Q} s_{rp}(q)\, D_q(j) + 1$) taken over all the states s_{rp} that can be obtained by (4).[2] Hence, constraints (5-6) enforce the pricing of the binary variable y_j^{rp} so that $y_j^{rp} = 1$ if $\sum_{q=1}^{Q} s_{rp}(q)\, D_q(j) \leq C_j$, and $y_j^{rp} = 0$ otherwise. Consequently, constraints (7-8) imply that the binary variable $f_{rp}^1 = 1$ *iff* $y_j^{rp} = 1$ for all $j \in 1, \ldots, m$, i.e., *iff* state s^{rp} is resource-feasible.

[2]The necessary methodology for obtaining these bounds is provided at the end of this section.

In a similar spirit, the characterization of the process feasibility of the state s_{rp} – i.e., the assessment of the non-negativity of the different components of s_{rp} – can be performed as follows:

$\forall q \in 1, \ldots, Q$,

$$s_{rp}(q) \;\geq\; (1 - z_q^{rp})\, L2_q^{rp} \tag{10}$$

$$s_{rp}(q) \;\leq\; z_q^{rp}\, U2_q^{rp} - 1 \tag{11}$$

$$f_{rp}^2 \;\leq\; z_q^{rp} \tag{12}$$

$$f_{rp}^2 + (Q-1) \;\geq\; \sum_{q=1}^{Q} z_q^{rp} \tag{13}$$

$$z_q^{rp}, f_{rp}^2 \;\in\; \{0,1\} \tag{14}$$

The parameter $L2_q^{rp}$ (resp., $U2_q^{rp}$), $q = 1, \ldots, Q$, denotes a lower (resp., upper) bound for the value of $s_{rp}(q)$ (resp., $s_{rp}(q) + 1$) over all states s_{rp} that can be obtained by (4). Hence, constraints (10-11) enforce the pricing of the binary variable z_q^{rp} so that $z_q^{rp} = 1$ *iff* $s_{rp}(q) \geq 0$. But then, constraints (12-13) price the binary variable f_{rp}^2 to 1 *iff* $s_{rp} \geq \mathbf{0}$.

Finally, the admissibility of state s_{rp} with respect to Δ_H, $H = (A, b, \pi, \theta)$, can be characterized as follows:

$\forall i \in 1, \ldots, K$,

$$b_i - A(i,\cdot) \cdot s_{rp} \;\geq\; (1 - x_i^{rp})\, L3_i^{rp} \tag{15}$$

$$b_i - A(i,\cdot) \cdot s_{rp} \;\leq\; x_i^{rp}\, U3_i^{rp} - 1 \tag{16}$$

$$x_i^{rp} \;\in\; \{0,1\} \tag{17}$$

and

$$\theta - \sum_{i=1}^{K} \pi(i)\, x_i^{rp} \;\geq\; (1 - f_{rp}^3)\, L^{rp} \tag{18}$$

$$\theta - \sum_{i=1}^{K} \pi(i)\, x_i^{rp} \;\leq\; f_{rp}^3\, U^{rp} - 1 \tag{19}$$

$$f_{rp}^3 \;\in\; \{0,1\} \tag{20}$$

The parameter $L3_i^{rp}$ (resp., $U3_i^{rp}$), $i = 1, \ldots, K$, denotes a lower (resp., upper) bound for the quantity $b_i - A(i,\cdot) \cdot s_{rp}$ (resp., $b_i - A(i,\cdot) \cdot s_{rp} + 1$) over all the states s_{rp} that can be obtained by (4). Furthermore, the parameter L^{rp} (resp., U^{rp}) denotes a lower (resp., upper) bound for the quantity $\theta - \sum_{i=1}^{K} \pi(i)\, x_i^{rp}$ (resp., $\theta - \sum_{i=1}^{K} \pi(i)\, x_i^{rp} + 1$). Then, constraints (15-17) enforce the pricing of the binary variable x_i^{rp} so that $x_i^{rp} = 1$ *iff* $A(i,\cdot) \cdot s_{rp} \leq b_i$, i.e., the pricing of x_i^{rp} materializes the corresponding indicator variable I in Equation (3). Consequently, by constraints (18-19), the binary variable $f_{rp}^3 = 1$ *iff* state s_{rp} is admissible with respect to Δ_H.

Based on the above, the plausibility of the event e_{rp} at state s of the automaton $G_{\Delta_H}(\Phi)$ can be characterized as follows:

$$f_{rp} \;\leq\; f_{rp}^k, \quad \forall k = 1, 2, 3 \tag{21}$$

$$f_{rp} + 2 \;\geq\; \sum_{k=1}^{3} f_{rp}^k \tag{22}$$

$$0 \;\leq\; f_{rp} \;\leq\; 1 \tag{23}$$

Constraints (21-23) set $f_{rp}=0$ if $f^k_{rp}=0$ for any $k=1,2,3$. On the other hand, they set $f_{rp}=1$ if $f^1_{rp}=f^2_{rp}=f^3_{rp}=1$, that is, if state s_{rp} is both feasible and admissible. Thus, $f_{rp}=1$ denotes the fact that the event $e_{rp}\in E$, $r\neq 0$ is executable at state s in $G_{\Delta_H}(\Phi)$, and we have established the following proposition:

Proposition 2: Given a state $s\in S$ of a D/C-RAS Φ, and an integer-valued quadruple $H=(A,b,\pi,\theta)$, consider the values of the binary variables f_{rp} that are obtained from the unique solution of the systems of Equations (4-23) defined for each event $e_{rp}\in E$, $r\neq 0$. Then, there exists an event $e_{rp}\in E$, $r\neq 0$, that is resource and process-feasible in Φ, and also admissible with respect to policy Δ_H, *iff*

$$\sum_{e_{rp}\in E,\ r\neq 0} f_{rp} > 0 \tag{24}$$

C. Characterizing the living space for state variable s

As already mentioned, the correctness test of Proposition 1 implies that the test of Proposition 2 must be satisfied by every state $s\in S\backslash\{s_0\}$ such that $\Delta_H(s)$=*true*. Hence, when s is considered as a Q-dimensional variable vector, it must satisfy the following constraints:

First of all, the process feasibility of s implies that

$$\forall q=1,\ldots,Q,\quad s(q)\in Z^+_0 \tag{25}$$

Furthermore, the resource feasibility of s implies that

$$\forall j=1,\ldots,m,\quad \sum_{q=1}^{Q} s(q)\,D_q(j) \ \leq\ C_j \tag{26}$$

The requirement that $s\neq s_0$ can be enforced by

$$\sum_{q=1}^{Q} s(q)\geq 1 \tag{27}$$

Finally, the *admissibility* of state s with respect to Δ_H, $H=(A,b,\pi,\theta)$, can be enforced as follows:

$\forall i=1,\ldots,K$,

$$b_i - A(i,\cdot)\cdot s \ \geq\ (1-x^s_i)L_i \tag{28}$$

$$b_i - A(i,\cdot)\cdot s \ \leq\ x^s_i\,U_i - 1 \tag{29}$$

$$x^s_i \ \in\ \{0,1\} \tag{30}$$

$$\sum_{i=1}^{K}\pi(i)\,x^s_i \ \leq\ \theta \tag{31}$$

The parameters L_i (resp., U_i), $i=1,\ldots,K$, denotes a lower (resp., upper) bound for the quantity $b_i-A(i,\cdot)\cdot s$, $s\in S$ (resp., $b_i-A(i,\cdot)\cdot s+1$, $s\in S$). Thus, constraints (28-30) enforce the pricing of the binary variable x^s_i so that $x^s_i=1$ if $A(i,\cdot)\cdot s\leq b_i$, and $x^s_i=0$ otherwise. Consequently, the pricing of x^s_i materializes the corresponding indicator variable I in Equation (3), and therefore, constraint (31) enforces the admissibility of state s by policy Δ_H.

D. The main result

In order to state the main result of this work, consider the Mixed Integer Programming (MIP) formulation $\mathcal{F}$ with its objective function defined by the minimization of the left-hand-side of Equation 24, and its constraint set consisting of Equations (25-31) and also one set of Equations (4-23) for each event $e_{rp}\in E$, $r\neq 0$. Also, let $O^*(\mathcal{F})$ denote the optimal objective value of this formulation. Then, we have the following theorem:

Theorem 1: Given a D/C-RAS $\Phi=<\mathcal{R},C,\mathcal{P},\mathcal{D}>$ and an integer-valued quadruple $H=<A,b,\pi,\theta>$, the policy Δ_H defined according to Definition 3 constitutes a *correct* generalized DAP for Φ if (a) $\Delta_H(s_0)$=*true* and (b) $O^*(\mathcal{F})>0$, where $O^*(\mathcal{F})$ is defined as in the previous paragraph.

Proof: The definition of f_{rp} through constraints (4-23) implies that

$$\sum_{e_{rp}\in E,\ r\neq 0} f_{rp}\geq 0$$

Furthermore, the fact that $O^*(\mathcal{F})>0$ combined with (a) the pricing of variable s through constraints (25-31) and (b) Proposition 2, imply that, for every state $s\in S\backslash\{s_0\}$ such that $\Delta_H(s_0)$=*true*, there exists at least one event $e_{rp}\in \bar{E}\cup E^{\searrow}$ that is feasible and admissible. But then, the correctness of Δ_H is immediately inferred by Proposition 1. $\diamond$

E. Discussion

The following observations elaborate and clarify the previous developments:

Observation 1: In agreement with the notation used in the previous sections, we let $|\mathcal{R}|$ denote the number of the system resource types, Q denote the number of distinct processing stages, K denote the row dimensionality of the matrix A that defines the generalized algebraic DAP under consideration, and $|\bar{E}\cup E^{\searrow}|$ denote the number of the job advancing and unloading events. Then, it is easy to verify that the formulation $\mathcal{F}$ considered in Theorem 1 involves $(|\mathcal{R}|+Q+K+3)\cdot|\bar{E}\cup E^{\searrow}|+K$ binary variables, Q nonnegative integer variables, $Q\cdot|\bar{E}\cup E^{\searrow}|$ real variables, and $|\bar{E}\cup E^{\searrow}|$ nonnengative real variables, in $|\mathcal{R}|+2K+2+|\bar{E}\cup E^{\searrow}|(4Q+3|\mathcal{R}|+2K+9)$ constraints. Hence, the *size* of this formulation, in terms of variables and constraints, is defined by second degree polynomials in the variables that define the size of the underlying system, $|\Phi|$, and the complexity / size of the applied DAP. On the other hand, the integral nature of many of the involved variables implies that the computational complexity of the proposed test remains *super-polynomial* with respect to $|\Phi|$. Collective past experience with similar formulations obtained for algebraic DAPs in the PN modeling framework (e.g., [11], [20], [16], [21]) and the capabilities of the existing computational platforms for Mixed Integer Programming, provide substantial evidence that the correctness test of Theorem 1 will remain a viable proposition for most practical cases. Furthermore, the involved computation can be substantially alleviated when realizing that the key concern

is not the exact calculation of the optimum value of the considered MP formulation but the assessment of whether this optimum value is equal to or greater than zero. Hence, assuming that the proposed formulation is solved through some variant of the *Branch & Bound* method [22], one can fathom any search path with a corresponding lower bound for the optimal solution greater than zero. Similarly, the entire computation can be terminated as soon as a feasible solution of zero objective value is identified.

Observation 2: The test of Theorem 1 is only a *sufficient* test for the correctness of policy Δ_H, since the living space of the state variable s defined by constraints (25-31) can contain states that are not accessible under Δ_H, while $O^*(\mathcal{F})$ is priced to zero by some of these states. It is possible, however, to extend the test of Theorem 1 to a *necessary and sufficient* condition for correctness, by tightening the living space of s in a way that excludes the aforementioned problematic states, and still maintain a polynomially sized MIP formulation with respect to $|\Phi|$, in terms of the engaged variables and constraints. Such an extension can be achieved by modifying ideas and techniques that were developed originally in [23] for the class of algebraic DAP's.

Observation 3: When restricted to the case of algebraic DAPs, the test of Theorem 1 is equivalent, in terms of its discriminative power, to the corresponding tests developed in [12], [3]. This can be realized by first noticing that the presence of states $s \neq s_0$ failing the test of Proposition 2, in the DFSA modeling framework, is equivalent to the presence of markings containing resource-induced deadly marked siphons, in the PN-modeling framework. Furthermore, since the PNs modeling the (D/C-)RAS behavior under the control of algebraic DAPs are consistent and conservative, the set of markings satisfying the state equation is equivalent to the set of markings satisfying the net semi-flows [24]. But in a well-defined process-resource net, these semi-flows essentially express the resource-feasibility and policy-admissibility requirements, and therefore, the set of DFSA states characterized by constraints (25-31) is equivalent, in the PN modeling framework, to the set of markings, $M \neq M_0$, that satisfy the state equation.

Observation 4: In order to support the complete implementation of the test of Theorem 1, it remains to discuss how to obtain the upper and lower bounds appearing in different parts of the involved formulation $\mathcal{F}$. Here we highlight this computation by discussing the case of $L1_j^{rp}$; the other bounds can be obtained through similar reasoning. Since, by its definition, $L1_j^{rp}$ must provide a lower bound to the left-hand-side of Equation 5 for any value of the state variable s, it can be priced by solving the following MIP formulation:

$$L1_j^{rp} \equiv \min\{C_j - \sum_{q=1}^{Q} s_{rp}(q)\, D_q(j)\}$$

s.t.

Equations 4, 25-31

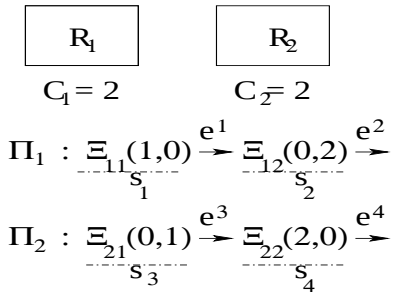

Fig. 1. The considered D/C-RAS

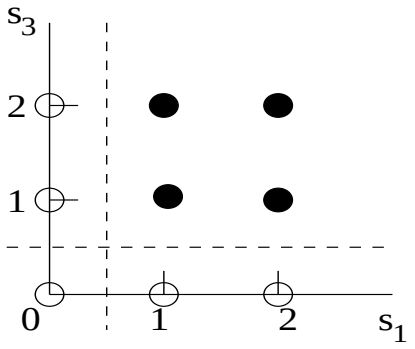

Fig. 2. Characterization of the safe and unsafe reachable states for the example D/C-RAS of Figure 1, in the projected sub-space defined by the state components s_1 and s_3; safe reachable states are depicted by white circles and unsafe reachable states by black ones.

IV. Example

This section demonstrates the methodology for DAP correctness verification that was developed in Section III, by employing it to assess the correctness of a number of generalized algebraic DAPs that are contemplated for the D/C-RAS depicted in Figure 1. As indicated in Figure 1, the considered D/C-RAS consists of two resource types, R_1 and R_2, with corresponding capacities $C_1 = C_2 = 2$, and it supports two process types: Process type Π_1 consists of a linear sequence of two processing stages, Ξ_{11} and Ξ_{12}, with corresponding resource allocation request vectors $D_{11} = (1,0)^T$ and $D_{12} = (0,2)^T$. Process type Π_2 is another linear sequence of two processing stages, Ξ_{21} and Ξ_{22}, with corresponding resource allocation request vectors $D_{21} = (0,1)^T$ and $D_{22} = (2,0)^T$. Figure 1 also annotates the mapping of the RAS stages, $\{\Xi_{11}, \Xi_{12}, \Xi_{21}, \Xi_{22}\}$, to the components of the state vector, s, that is employed by the FSA-based representation of the considered RAS, and the job advancing and unloading events that will be of interest in the subsequent formulation.

Clearly, the optimal DAP for the considered RAS must restrict only the resource allocation with respect to stages Ξ_{11} and Ξ_{21}, since any process instances executing the remaining stages Ξ_{12} and Ξ_{22} can immediately exit the system upon their completion. Hence, the admissibility of any given RAS state, $s = (s_1, s_2, s_3, s_4)^T$, can be resolved by considering only its projection to the subspace defined by its components s_1 and s_3. This projection is depicted in Figure 2, which indicates that the set of reachable and safe

TABLE I
THE POLICY-DEFINING PARAMETERS

Policy	a_1	b_1	a_2	b_2
1	2	1	2	1
2	2	3	2	1

TABLE II
THE FORMULATION PARAMETERS E^i_j

$i \setminus j$	1	2	3	4
1	-1	1	0	0
2	0	-1	0	0
3	0	0	-1	1
4	0	0	0	-1

states of the considered D/C-RAS is indeed non-convex. Figure 2 also proposes a structure for the generalized algebraic DAPs that would implement the optimal DAP, Δ^*. More specifically, such a policy can be defined by any pair of straight lines that separate the safe from the unsafe subspace, as indicated in the figure, plus a "voting" scheme that admits a state vector s if and only if it satisfies one of the two inequalities defined by the aforementioned straight lines. A particular instantiation, that corresponds to the separating lines annotated in Figure 2, is given by the inequality:

$$-1 \cdot I_{\{2s_1 \leq 1\}} - 1 \cdot I_{\{2s_3 \leq 1\}} \leq -1 \tag{32}$$

Motivated by the above observations, in the following, we employ the MIP formulation of Theorem 1, in order to assess the correctness for the D/C-RAS of Figure 1, of the two generalized algebraic DAPs defined by the following structure

$$-1 \cdot I_{\{a_1 s_1 \leq b_1\}} - 1 \cdot I_{\{a_2 s_3 \leq b_2\}} \leq -1 \tag{33}$$

and the parameterizations for $a_1,\ b_1,\ a_2$ and b_2 provided in Table I. Notice that, under the proposed parameterizations, Policy #1 is the maximally permissive DAP defined by Equation 32. On the other hand, Policy #2 is an incorrect DAP, since it admits all the reachable unsafe states with $s_1 = 1$.

The reader can verify that the customization of the MIP formulation of Theorem 1 to the considered application context, results in the formulation of Figure 3. This formulation consists of 148 constraints, involving 46 binary, 4 non-negative integer, 16 real and 4 nonnegative real variables. The parameters $E^i_j,\ i = 1, \ldots, 4,\ j = 1, \ldots, 4$, that appear in it, are readily obtained by the definition of the events e^1, e^2, e^3 and e^4 in Figure 1, and they are tabulated in Table II. The remaining bounding parameters of the L and U types, that are necessary for the complete characterization of the formulation, can be obtained as indicated in Observation 4.

The solution of the formulation of Figure 3 for the two policy parameterizations provided in Table I, through the

$$\min \sum_{i=1}^{4} f_i$$

s.t.

$$s_1 + 2s_4 \leq 2$$
$$2s_2 + s_3 \leq 2$$
$$\sum_{i=1}^{4} s_i \geq 1$$
$$a_1 s_1 - L_1 x^s_1 \leq b_1 - L_1$$
$$a_2 s_3 - L_2 x^s_2 \leq b_2 - L_2$$
$$a_1 s_1 + U_1 x^s_1 \geq b_1 + 1$$
$$a_2 s_3 + U_2 x^s_2 \geq b_2 + 1$$
$$x^s_1 + x^s_2 \geq 1$$
$$\forall i = 1, \ldots, 4,\ \forall j = 1, \ldots, 4,\quad s^i_j - s_j = E^i_j$$
$$\forall i = 1, \ldots, 4,\quad s^i_1 + 2s^i_4 - L1^i_1 y^i_1 \leq 2 - L1^i_1$$
$$\forall i = 1, \ldots, 4,\quad 2s^i_2 + s^i_3 - L1^i_2 y^i_2 \leq 2 - L1^i_2$$
$$\forall i = 1, \ldots, 4,\quad s^i_1 + 2s^i_4 + U1^i_1 y^i_1 \geq 3$$
$$\forall i = 1, \ldots, 4,\quad 2s^i_2 + s^i_3 + U1^i_2 y^i_2 \geq 3$$
$$\forall i = 1, \ldots, 4,\ \forall j = 1, 2,\quad f^1_i - y^i_j \leq 0$$
$$\forall i = 1, \ldots, 4,\quad \sum_{j=1}^{2} y^i_j - f^1_i \leq 1$$
$$\forall i = 1, \ldots, 4,\ \forall j = 1, \ldots, 4\quad s^i_j + L2^i_j z^i_j \geq L2^i_j$$
$$\forall i = 1, \ldots, 4,\ \forall j = 1, \ldots, 4\quad U2^i_j z^i_j - s^i_j \geq 1$$
$$\forall i = 1, \ldots, 4,\ \forall j = 1, \ldots, 4\quad f^2_i - z^i_j \leq 0$$
$$\forall i = 1, \ldots, 4,\quad \sum_{j=1}^{4} z^i_j - f^2_i \leq 3$$
$$\forall i = 1, \ldots, 4,\quad a_1 s^i_1 - L3^i_1 x^i_1 \leq b_1 - L3^i_1$$
$$\forall i = 1, \ldots, 4,\quad a_2 s^i_3 - L3^i_2 x^i_2 \leq b_2 - L3^i_2$$
$$\forall i = 1, \ldots, 4,\quad a_1 s^i_1 + U3^i_1 x^i_1 \geq b_1 + 1$$
$$\forall i = 1, \ldots, 4,\quad a_2 s^i_3 + U3^i_2 x^i_2 \geq b_2 + 1$$
$$\forall i = 1, \ldots, 4,\quad \sum_{j=1}^{2} x^i_j + L^i f^3_i \geq L^i + 1$$
$$\forall i = 1, \ldots, 4,\quad \sum_{j=1}^{2} x^i_j - U^i f^3_i \leq 0$$
$$\forall i = 1, \ldots, 4,\ \forall j = 1, 2, 3,\quad f_i - f^j_i \leq 0$$
$$\forall i = 1, \ldots, 4,\quad \sum_{j=1}^{3} f^j_i - f_i \leq 2$$
$$f^j_i, x^s_i, y^i_j, z^i_j, x^i_j \in \{0, 1\};\ 0 \leq f_i \leq 1;\ s_j \in Z^+_0;\ s^i_j \in R$$

Fig. 3. The MIP formulation for the considered example

$GAMS^{©}$ solver, returned, as expected, the optimal value of 1 for the first case, and the optimal value of 0 for the second. In the second case, the s vector returned in the optimal solution was $s = (1, 0, 1, 0)^T$, which can be verified to be a deadlock state. Finally, in order to provide some more concrete feeling regarding the underlying computational requirements, we also notice that the execution of the considered formulation for the two policy instantiations on a SunOS 5.9 platform, required 10 milliseconds for Policy #1 and less than one millisecond for Policy #2. These times are consistent with the insights provided in Observation 1, since the identification of an admissible deadlock state during the correct evaluation of Policy #2 – i.e., a feasible solution with a zero objective value – results in drastic fathoming, and therefore, an expedient search process. On the other hand, verifying the correctness of Policy #1 requires a more exhaustive enumeration of the search tree underlying the Branch & Bound method. Both of these times could have been improved through a customized implementation of the Branch & Bound method according to the suggestions offered in Observation 1.

V. Conclusion

The main contribution of this paper is an analytical test that can verify the correctness of any tentative generalized algebraic DAP Δ_H for some given (D/C-)RAS Φ. The presented test possesses the convenient form of a Mixed Integer Programming formulation that employs a number of variables and constraints polynomially related to the RAS size $|\Phi|$, and it can be readily solved through canned optimization software. The developed formulation essentially constitutes an extension to the class of generalized algebraic DAPs of similar past results derived for the class of algebraic DAPs. At the same time, it effects the migration of those past results from the PN to the DFSA representational framework.

Future work will evolve in two directions: (i) One leg will explore the design of detailed, customized algorithms for the criterion of Theorem 1 and the potential development of alternative MP-based correctness verification tests for generalized algebraic DAPs that will employ a smaller number of variables and/or constraints. (ii) An additional leg will seek the embedding of the derived test(s) in an intelligent search procedure towards the identification of highly efficient generalized algebraic DAPs for any given (D/C-)RAS Φ, possibly satisfying additional design requirements.

References

[1] S. Reveliotis, E. Roszkowska, and J. Y. Choi, "Generalized algebraic deadlock avoidance policies for sequential resource allocation systems," *IEEE Trans. on Automatic Control*, vol. 52, pp. 2345–2350, 2007.

[2] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. Boston, MA: Kluwer Academic Pub., 1999.

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

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

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

[6] J. O. Moody and P. J. Antsaklis, *Supervisory Control of Discrete Event Systems using Petri nets*. Boston, MA: Kluwer Academic Pub., 1998.

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

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

[9] 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*, vol. 13, pp. 347–363, 1997.

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

[11] F. Chu and X.-L. Xie, "Deadlock analysis of petri nets using siphons and mathematical programming," *IEEE Trans. on R&A*, vol. 13, pp. 793–804, 1997.

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

[13] S. A. Reveliotis, "On the siphon-based characterization of liveness in sequential resource allocation systems," in *Applications and Theory of Petri Nets 2003*, 2003, pp. 241–255.

[14] Z. W. Li and M. C. Zhou, "Elementary siphons of Petri nets and their application to deadlock prevention in flexible manufacturing systems," *IEEE Trans. on SMC – Part A*, vol. 34, pp. 38–51, 2004.

[15] S. A. Reveliotis, "Implicit siphon control and its role in the liveness enforcing supervision of sequential resource allocation systems," *IEEE Trans. on SMC: Part A*, pp. 319–328, 2007.

[16] 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*, vol. 19, pp. 192–208, 2002.

[17] 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*, vol. 19, pp. 137–141, 2003.

[18] S. A. Reveliotis and J. Y. Choi, "Designing reversibility-enforcing supervisors of polynomial complexity for bounded Petri nets through the theory of regions," in *Proceedings of ATPN 2006*, 2006, pp. 322–341.

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

[20] J. Park and S. Reveliotis, "Algebraic synthesis of efficient deadlock avoidance policies for sequential resource allocation systems," *IEEE Trans. on R&A*, vol. 16, pp. 190–195, 2000.

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

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

[23] S. A. Reveliotis, "A necessary and sufficient condition for the liveness and reversibility of process-resource nets with acyclic, quasi-live serialisable and reversible process subnets," *IEEE Trans. on Automation Science & Engineering*, vol. 3, pp. 462–468, 2006.

[24] M. Silva, E. Teruel, and J. M. Colom, "Linear algebraic and linear programming techniques for the analysis of place/transition net systems," in *Lecture Notes in Computer Science, Vol. 1491*, W. Reisig and G. Rozenberg, Eds. Springer-Verlag, 1998, pp. 309–373.