Policy Mixtures: A Novel Approach for Enhancing the Operational Flexibility of Resource Allocation Systems with Alternate Routings

Jonghun Park and Spyros A. Reveliotis

Abstract— **Liveness-enforcing supervisors (LESs) guarantee deadlock-free operations in resource allocation systems (RASs). Given the growing emphasis on higher resource utilization and operational flexibility, it is greatly desirable to obtain the optimal LES that can provide the maximum behavioral latitude. In general, however, computation of the maximally permissive LES is computationally intractable, and accordingly, the class of algebraic LESs has been proposed in the literature as a viable suboptimal alternative for various RAS classes. The objective of this work is to further enhance the permissiveness of algebraic LESs by taking advantage of the routing flexibility inherent in contemporary RASs. Specifically, we consider a class of algebraic LES that allows supervisor parameterizations based on the system routing flexibility, and propose a method to effectively *mix* the algebraic LESs obtained from the policy instantiation through a set of different parameter values. The resulting policy mixture, to be called *dynamic algebraic LES*, provides increased permissiveness as it allows flexible on-line switching among different resource reservation schemes, corresponding to different process routings. The proposed method for constructing the dynamic algebraic LES is presented in the context of the class of Conjunctive / Disjunctive (CD)-RASs, which subsumes the major RAS classes currently studied in the literature.**

Keywords— **Discrete Event Systems, Supervisory Control, Resource Allocation Systems, Liveness, Deadlock, Routing Flexibility, Policy Mixtures**

I. Introduction

Sequential resource allocation is a pertinent abstraction for the qualitative modeling of the behavioral dynamics of many contemporary technological applications. Generally speaking, a sequential resource allocation system (RAS) is defined by a finite set of resources, and a set of processes, executing in stages, with each process stage requiring the exclusive allocation of a certain resource subset for its successful completion. A key requirement for the behavioral correctness of these systems is that they remain live, i.e., that they effectively avoid deadlocking situations where a set of executing processes is permanently blocked due to the fact that each process in this set requires for its further advancement the allocation of some resource(s) held by another process in this set. In [1] it was shown that the problem of deadlock avoidance in sequential RAS translates to the synthesis of a supervisor that will constrain the system behavior in a strongly connected component of its underlying state space, that also contains the system initial empty state. Given the current emphasis on increased resource utilization and operational flexibility, the objective is to obtain the maximally permissive liveness-enforcing supervisor (LES), that will provide the maximum behavioral latitude. However, the implementation of the maximally permissive LES is a computationally intractable problem for most real-life applications [2]. Hence, a large body of the past results has been centered around the development of suboptimal but polynomially computable LESs, that seek to assure the system liveness while maintaining considerable flexibility, through pertinent exploitation of the available information on the system structure and dynamics [3], [1], [4], [5], [6].

In particular, *algebraic LESs* have been proposed in the past as mathematically elegant and computationally efficient solutions to the problem of LES synthesis for various RAS classes, able to provide a viable trade-off between computational tractability and operational efficiency [1], [5], [7], [6], [8]. A formal definition of this class of policies is provided in the next section, but at this point we notice that their algebraic character stems from the fact that they attain the system liveness by constraining the maximum number of jobs that can simultaneously execute various subsets of the RAS process stages. As a result, they can be naturally expressed by a system of linear inequalities to be observed by the RAS state. If the number of the stage sets defining the policy constraints is polynomially bounded with respect to (w.r.t.) the RAS size, the resulting policy is polynomially implementable through one-step-lookahead.

Given the suboptimality of the algebraic LES, past research has also proposed a number of ways for enhancing their operational flexibility/permissiveness. Hence, it has been shown that when there exists a set of algebraic LESs for a given RAS, one can define the *disjunctive* supervisor [8] that admits a RAS state if and only if (iff) the state is admitted by some LES in the set. This new supervisor leads also to live system behavior, while it is more permissive than the originally available policies, since it admits the union of the state subspaces admitted by each of the constituent supervisors. More recently, the works presented in [7], [8] attempted to enhance the permissiveness of any given algebraic LES by relaxing the right-hand-side (rhs) of its defining linear constraints, and provided a search method for identifying a maximal rhs vector that can retain the system liveness. Another relevant line of research can be found in [9], where a two-stage heuristic for applying an algebraic LES to RAS allowing for alternative resource allocations was presented.

The work presented in this paper seeks to complement the aforementioned theoretical developments, and provide a computationally efficient method to further enhance the permissiveness of algebraic LESs, by taking advantage of the routing flexibility inherent in contemporary RASs. Specifically, this paper starts from the observation that certain algebraic LES classes, when applied on RAS with routing flexibility, can have many different instantiations parameterized by the process routes supported by the eventual LES implementation, and proposes a way to effectively *mix* (a subset of) the resulting policy instantiations. The obtained supervisor, to be called *dynamic algebraic LES*, constitutes a *policy mixture*, since it allows the concurrent execution in the RAS of various process instances observing different LES instantiations, and their flexible on-line switching among the different resource reservation schemes defining the constituent LESs. As it was mentioned above, the net result is the enhancement of the underlying operational permissiveness / flexibility. In fact, it is shown that the proposed policy mixture is more powerful than supervisor disjunction. The computational procedure for constructing the dynamic algebraic LES and the proof for its correctness are presented in the context of the class of Conjunctive / Disjunctive Resource Allocation Systems (CD-RASs) [8], which subsumes the major RAS classes currently studied in the literature [10], [4], [7], [9], [8]. Nevertheless, we remark that the idea presented in this paper can be applied to more general RAS classes; one such application in the context of an RAS that extends the CD-RAS with reworks and uncontrollable events, can be found in [11].

The rest of the paper is organized as follows: Section II provides the preliminary results necessary for the subsequent analysis. Section III develops the main results of the paper. Finally, Section IV concludes the paper and provides some discussion on potential future research.

II. Preliminary Results

This section reviews some past results that are necessary for the developments in Section III. Specifically, the first part of the section introduces the S^3PGR^2 net, originally presented in [6], that effectively models the dynamics of CD-RAS, by allowing for conjunctive resource acquisitions and flexible routings. The rest of the section defines the notion of algebraic LES for the S^3PGR^2 net, and introduces the G-RUN LES as a canonical example of algebraic LES for the subsequent analysis. In what follows, we assume that the reader is familiar with the basic results of Petri net structural analysis [12].

CD-RAS and their Petri net modeling The CD-RAS is defined by a set of *resource types* $\mathcal{R} = \{R_i, i = 1, \ldots, m\}$, and a set of *process types* $\mathcal{J} = \{J_j, j = 1, \ldots, n\}$. Every resource type R_i is further characterized by its *capacity* $C_i \in Z^+$, where Z^+ is the set of positive integers. Processing requirements of process type J_j are defined by a set of *stages*, partially ordered through a set of precedence constraints. Each process stage p is associated with a *conjunctive* resource allocation requirement, expressed by an m-dimensional vector $(a_{ip})^{i=1,\ldots,m}$, where $a_{ip} \in \{0\} \cup Z^+, i = 1, \ldots, m$, indicates how many units of resource R_i are required to support the stage execution.

The resource allocation dynamics taking place in CD-RAS can be modeled by a marked Petri net [12]. Specifically, the process flow of each process type J_j is represented by a net structure known as *Simple Sequential Process (S^2P)* [10]. This net is formally defined by an ordinary strongly connected state machine $\mathcal{N}_j = (P_{S_j} \cup \{p_{0_j}\}, T_j, W_j)$ such that (i) $P_{S_j} \neq \emptyset$, $p_{0_j} \notin P_{S_j}$, and (ii) every circuit of $\mathcal{N}_j$ contains $\{p_{0_j}\}$. In the S^2P net, each place $p \in P_{S_j}$ corresponds to a process stage of J_j, and place p_{0_j} is characterized as the *idle place*, since its marking represents the instances of process type J_j waiting to be loaded to the RAS. The S^3PGR^2 *(System of Simple Sequential Processes with General Resource Requirements)* net modeling the CD-RAS is completed by interconnecting the S^2P nets through a set of *resource places*, P_R, which model the availability of the various resource types. Formally, it is defined as follows:

Definition 1: A *well-marked* S^3PGR^2 net is a marked PN $\mathcal{N} = (P, T, W, M_0)$ such that
1. $P = P_S \cup P_0 \cup P_R$, where $P_S = \bigcup_{i=1}^n P_{S_i}$ s.t. $P_{S_i} \cap P_{S_j} = \emptyset, \forall i \neq j$, $P_0 = \bigcup_{i=1}^n \{p_{0_i}\}$ s.t. $P_0 \cap P_S = \emptyset$, and $P_R = \{r_1, \ldots, r_m\}$ s.t. $(P_S \cup P_0) \cap P_R = \emptyset$.
2. $T = \bigcup_{i=1}^n T_i$.
3. $W = W_S \cup W_R$, where $W_S : ((P_S \cup P_0) \times T) \cup (T \times (P_S \cup P_0)) \rightarrow \{0, 1\}$ s.t. $\forall j \neq i, ((P_{S_j} \cup P_{0_j}) \times T_i) \cup (T_i \times (P_{S_j} \cup P_{0_j})) \rightarrow \{0\}$, and $W_R : (P_R \times T) \cup (T \times P_R) \rightarrow \{0\} \cup Z^+$.
4. $\forall i$, $i = 1, \ldots, n$, the subnet $\mathcal{N}_i$ generated by $P_{S_i} \cup \{p_{0_i}\} \cup T_i$ is an S^2P net.
5. $\forall r \in P_R$, $\exists$ a unique minimal p-semiflow y_r s.t. $\|y_r\| \cap P_R = \{r\}$, $\|y_r\| \cap P_0 = \emptyset$, $\|y_r\| \cap P_S \neq \emptyset$, and $y_r(r) = 1$. Furthermore, $P_S = \bigcup_{r \in P_R}(\|y_r\| - P_R)$.
6. $\mathcal{N}$ is pure and strongly connected.
7. $\forall p \in P_S$, $M_0(p) = 0$; $\forall r \in P_R$, $M_0(r) \geq max_{p \in \|y_r\|} y_r(p)$; and $\forall p_{0_i} \in P_0$, $M_0(p_{0_i}) \geq 1$.

Example 1 Figure 1 shows an example S^3PGR^2 net. It models a CD-RAS consisting of three resource types R_1, R_2, and R_3 with capacities $C_1 = C_2 = 4$, and $C_3 = 3$, and supporting two process types J_1 and J_2. The flow of process type J_1 is defined by the set of places $\{p_{10}, p_1, p_2, p_3\}$ that are strictly ordered through the flow relations. Hence, J_1 does not present any routing flexibility. Process type J_2 allows for alternate routings, and its flow is defined by the set of partially ordered places $\{p_{20}, p_4, p_5, p_6, p_7, p_8, p_9\}$. That is, a process of type J_2 can take any of the routes defined by the sequence of places $< p_4, p_5, p_7 >$, $< p_4, p_5, p_8 >$, $< p_4, p_6, p_8 >$ and $< p_4, p_6, p_9 >$, for its successful completion. The resource allocation requirements for the process places are defined as follows: $(a_{ip_1})^{i=1,2,3} = (2, 0, 0)$, $(a_{ip_2})^{i=1,2,3} = (0, 1, 0)$, $(a_{ip_3})^{i=1,2,3} = (1, 0, 2)$, $(a_{ip_4})^{i=1,2,3} = (1, 0, 1)$, $(a_{ip_5})^{i=1,2,3} = (0, 2, 0)$, $(a_{ip_6})^{i=1,2,3} = (0, 0, 1)$, $(a_{ip_7})^{i=1,2,3} = (0, 1, 0)$, $(a_{ip_8})^{i=1,2,3} = (0, 0, 2)$, $(a_{ip_9})^{i=1,2,3} = (0, 1, 1)$. Finally, assuming that places are ordered in the flow matrix $\mathbf{\Theta}$ according to the sequence $< p_{10}, p_1, p_2, p_3, p_{20}, p_4, p_5, p_6, p_7, p_8, p_9, r_1, r_2, r_3 >$, $M_0 = (4, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 4, 4, 3)^T$, and $M = (0, 0, 4, 0, 1, 0, 0, 3, 0, 0, 0, 4, 0, 0)^T$ is a reachable marking from M_0, at which no transitions are enabled. Therefore, the net is not live. $\diamond$

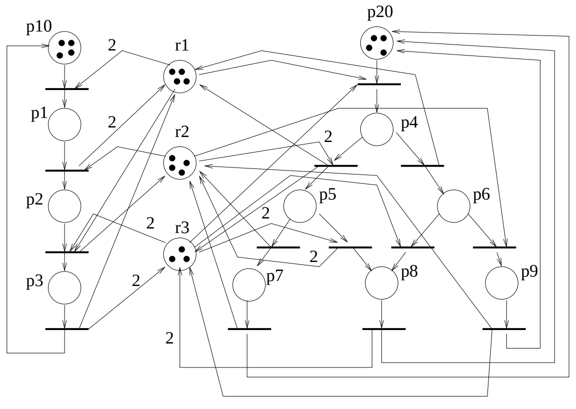

Fig. 1. An example S^3PGR^2 net

Algebraic LESs Given an S^3PGR^2 net, $\mathcal{N} = (P_0 \cup P_S \cup P_R, T, W, M_0)$, an algebraic LES is defined by the following system of linear inequalities:

$$\mathbf{A} \cdot M_S \leq \mathbf{f} \tag{1}$$

In Equation (1), $\mathbf{A} = [\bar{\alpha}_{ip}]_{p \in P_S}^{i=1,\ldots,N}$ is an $N \times |P_S|$ matrix such that $\bar{\alpha}_{ip} \geq 0, i = 1, \ldots, N$, $\forall p \in P_S$, and N is a polynomial number with respect to the RAS size;[1] M_S is a vector representing the system resource allocation state, obtained by projecting the net marking M on the subspace defined by P_S; $\mathbf{f} = (f_i)_{i=1,\ldots,N}$ is an N-dimensional vector of positive integers. As a control law, Equation (1) implies that the RAS state represented by vector M_S is *admissible* iff Equation (1) is satisfied.

The control logic imposed by an algebraic LES can be embedded to the behavior of the original, uncontrolled S^3PGR^2 net through the superposition of a *control subnet* constructed as follows: (i) Each constraint in Equation (1) is represented by a control place w_i, with initial marking $M_0(w_i) = f_i$, $i = 1, \ldots, N$; let $P_W = \{w_1, w_2, \ldots, w_N\}$. (ii) The control places $w_i \in P_W$ are connected to the net transitions as follows: First, we define $\bar{\alpha}_{ip} = 0$, $\forall p \in P_0$. Also, $\forall t \in T$, let $\{p\} \equiv {}^\bullet t \cap (P_S \cup P_0)$, and $\{q\} \equiv t^\bullet \cap (P_S \cup P_0)$. Then, $\forall w_i \in P_W$, $W(w_i, t) = \bar{\alpha}_{iq} - \bar{\alpha}_{ip}$, if $\bar{\alpha}_{iq} - \bar{\alpha}_{ip} > 0$; $W(t, w_i) = \bar{\alpha}_{ip} - \bar{\alpha}_{iq}$, if $\bar{\alpha}_{iq} - \bar{\alpha}_{ip} < 0$; $W(w_i, t) = W(t, w_i) = 0$, otherwise. The resulting net, $\mathcal{N} = (P_0 \cup P_S \cup P_R \cup P_W, T, W, M_0)$, is called CS^3PGR^2 (*Controlled* S^3PGR^2).

It is easy to see that, in the CS^3PGR^2 net, the control places defined by P_W play the role of additional *control resources*, since they control the firing of their output transitions (representing the original resource allocation events) through the availability of the tokens circulated through them, in the same way that the non-/availability of tokens in the original resource places disables/enables the firing of the net transitions. Based on this observation, we obtain the nice closure property that the CS^3PGR^2 net belongs to the class of S^3PGR^2 nets. Furthermore, we remark that the size of CS^3PGR^2 net is polynomially related to that of the original, uncontrolled S^3PGR^2 net.

[1]The RAS size is defined by the number of resource types and the number of distinct process stages.

The G-RUN LES for S^3PGR^2 nets G-RUN is a class of algebraic LESs that characterizes the set of **A** matrices leading to correct LES implementations, for any given CD-RAS, as an integral polytope [8]. To formally define this class of policies, consider a well-marked S^3PGR^2 net $\mathcal{N} = (P_0 \cup P_S \cup P_R, T, W, M_0)$, and let $o_i = o(R_i), i = 1, \ldots, m$, where $o() : \mathcal{R} \rightarrow \{1, \ldots, m\}$ is any partial order imposed on resource set $\mathcal{R}$. Given a place $p \in P_S$, a *neighborhood* N_p of p is defined by $N_p \subseteq p^{\bullet\bullet} \cap P_S$. Furthermore, a function $g() : \{p \in P_S \mid p^{\bullet\bullet} \cap P_0 = \emptyset\} \rightarrow \{N_p \mid p \in P_S \wedge p^{\bullet\bullet} \cap P_0 = \emptyset\}$ determines uniquely a *community* Ψ of $\mathcal{N}$, defined by $\Psi = \{(p, q) \mid p \in P_S \wedge p^{\bullet\bullet} \cap P_0 = \emptyset \wedge q \in N_p\}$. Let the set of all communities of $\mathcal{N}$ be denoted by $\mathcal{C}_{\mathcal{N}}$, and $\rho_p^{min} = \min\{o_i \mid a_{ip} > 0, i = 1, \ldots, m\}, \forall p \in P_S$. Then, an algebraic LES, $(\mathbf{A} = [\bar{\alpha}_{ip}]_{p \in P_S}^{i=1,\ldots,N}, \mathbf{f} = (f_i)_{i=1,\ldots,N})$, is a G-RUN LES iff $N = m$; $f_i = C_i, \ i = 1, \ldots, m$; and $\exists \ o() : \mathcal{R} \rightarrow \{1, \ldots, m\}$ and some community $\Psi \in \mathcal{C}_{\mathcal{N}}$, s.t.

$$\bar{\alpha}_{ip} \geq \bar{\alpha}_{iq} \qquad \forall (p,q) \in \Psi, \forall R_i \in \mathcal{R} \quad \text{s.t. } o_i \geq \rho_p^{min} \tag{2}$$

$$\bar{\alpha}_{ip} = 0 \qquad \forall p \in P_S, \forall R_i \in \mathcal{R} \quad \text{s.t. } o_i < \rho_p^{min} \tag{3}$$

$$C_i \geq \bar{\alpha}_{ip} \geq a_{ip} \qquad \forall p \in P_S, \forall R_i \in \mathcal{R} \tag{4}$$

$$\bar{\alpha}_{ip} \in \{0\} \cup Z^+ \qquad \forall p \in P_S, \forall R_i \in \mathcal{R} \tag{5}$$

$\diamond$

We notice, for those familiar with the original definitions of RUN LES for sequential RAS with linear process flows, presented in [1], [13], that the notion of *community* Ψ facilitates the implementation of the original policy-defining logic in the broader class of RAS presenting routing flexibility, by arbitrarily selecting for each stage $p \in P_S$ with $p^{\bullet\bullet} \cap P_0 = \emptyset$, a number of immediate successor stages, $q \in N_p \equiv g(p)$, and requiring that the policy-defining conditions are satisfied w.r.t. each pair $(p, q) \in \Psi$. This arbitrary definition of the community set, Ψ, through the arbitrary definition of function $g()$ during the policy implementation, introduces an additional parameterization of the G-RUN LES[2] that constitutes the basis for the definition of the *dynamic* G-RUN in Section III. The correctness of the G-RUN LES was established in [11], where it was shown that the CS^3PGR^2 net implementing a G-RUN LES is live. Furthermore, in [11], [8] it was also shown that, given an S^3PGR^2 net $\mathcal{N}$ with some pre-selected partial resource ordering $o_i, i = 1, \ldots, m$, and community Ψ, an operationally efficient G-RUN implementation can be computed via a linear program [14] defined by the following formulation:

$$\min G(\mathbf{A}; \mathcal{N}, o, \Psi) = \sum_{i=1,\ldots,m} \sum_{p \in P_S} \bar{\alpha}_{ip} \tag{6}$$
$$\text{s.t. Equations (2) – (4)}$$

Example 2 As an example of the G-RUN LES implementation, consider the S^3PGR^2 net introduced in Example 1. Taking the arbitrary partial resource ordering $o_1 = 2, o_2 = 1, o_3 = 3$, and community $\Psi_1 = \{(p_1, p_2), (p_2, p_3), (p_4, p_5), (p_5, p_7), (p_6, p_8)\}$, we solve the linear program defined by equations (2)-(4), and (6). The resulting G-RUN LES is expressed by Equation (7).

$$\begin{bmatrix} 2 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 2 & 0 & 1 & 0 & 1 \\ 2 & 2 & 2 & 1 & 0 & 2 & 0 & 2 & 1 \end{bmatrix} \cdot M_S \quad \leq \quad \begin{bmatrix} 4 \\ 4 \\ 3 \end{bmatrix} \tag{7}$$

[2]i.e., beyond the parameterization of the original RUN LES w.r.t. the employed resource ordering

Next, with the same resource ordering, we consider another community, defined by $\Psi_2 = \{(p_1, p_2), (p_2, p_3), (p_4, p_5), (p_5, p_8), (p_6, p_9)\}$. After solving the corresponding linear program, we obtain a different G-RUN LES realization, defined in Equation (8).

$$\begin{bmatrix} 2 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 2 & 0 & 1 & 0 & 1 \\ 2 & 2 & 2 & 2 & 2 & 1 & 0 & 2 & 1 \end{bmatrix} \cdot M_S \quad \leq \quad \begin{bmatrix} 4 \\ 4 \\ 3 \end{bmatrix} \tag{8}$$

$\diamond$

III. Dynamic Algebraic LES, Policy Mixtures, and the Dynamic G-RUN

Dynamic Algebraic LES Under the resource allocation-based interpretation of the control sub-net implementing the super-imposition of an algebraic LES on a given S^3PGR^2 net, $\mathcal{N} = (P_0 \cup P_S \cup P_R, T, W, M_0)$, each column $\mathbf{a}_p$ of the LES-defining matrix **A** represents a *policy-induced* resource allocation requirement w.r.t. the control resources $w_i, \ i = 1, \ldots, N$, associated with the place $p \in P_S$. This set of fictitious (control) resources is requested for the execution of the corresponding process stage in addition to the original resource allocation requirement w.r.t. the actual resource set $\mathcal{R}$, and it is uniquely defined for each place $p \in P_S$. In the following, we shall refer to the combination of the actual and the policy-induced resource allocation requirement associated with any place $p \in P_S$ as the *effective* resource allocation requirement for p. The class of the *dynamic algebraic* LES relaxes the uniqueness assumption for the supervisor-induced resource allocation, by allowing alternative policy-induced resource allocation requests for each process stage; an input transition of the corresponding place $p \in P_S$ can be enabled if any of its alternative effective resource allocation requests can be satisfied.

In more technical terms, given a S^3PGR^2 net $\mathcal{N} = (P_0 \cup P_S \cup P_R, T, W, M_0)$ with $P_R = \{r_1, \ldots, r_m\}$ and $P_S = \{p_1, \ldots, p_\sigma\}$, a dynamic algebraic LES for this net, allowing γ alternative policy-induced resource allocation requests for each process stage, is represented by a set of linear inequalities:

$$\mathcal{D} \cdot \hat{M}_S \leq \mathbf{f} \tag{9}$$

where $\mathcal{D} = [\mathbf{D}_{p_1} \ldots \mathbf{D}_{p_\sigma}]$ is an $(N) \times (\sigma \cdot \gamma)$ matrix, with $\mathbf{D}_{p_k} = [\mathbf{d}_{p_k}^1 \ldots \mathbf{d}_{p_k}^\gamma]$, $k = 1, \ldots, \sigma$, and $\mathbf{d}_{p_k}^j = (\delta_{1p_k}^j, \ldots, \delta_{Np_k}^j)^T$, $k = 1, \ldots, \sigma, j = 1, \ldots, \gamma$. That is, for each $p_k, k = 1, \ldots, \sigma$, there are γ alternative policy-induced resource allocation requests associated with it for enforcing liveness, and $\delta_{ip_k}^j, k = 1, \ldots, \sigma, j = 1, \ldots, \gamma, i = 1, \ldots, N$, represents the requested amount of the i-th control resource type for the process place p_k under the j-th alternative allocation scheme. In this representation, $\hat{M}_S = (\hat{m}_{p_1}, \ldots, \hat{m}_{p_\sigma})^T$ is a $(\sigma \cdot \gamma)$- dimensional vector representing the system resource allocation state, where $\hat{m}_{p_k} = (\mu_{p_k}^1, \ldots, \mu_{p_k}^\gamma), k = 1, \ldots, \sigma$, and $\mu_{p_k}^j, k = 1, \ldots, \sigma, j = 1, \ldots, \gamma$, indicates the number of processes (i.e. tokens) at stage (i.e. place) p_k under the j-th alternative allocation scheme. $\mathbf{f} = (f_i)_{i=1,\ldots,N}$ is an N-dimensional integral vector. The dynamic algebraic LES requires that an RAS state represented by $\hat{M}_S$ is *admissible* iff $\mathcal{D} \cdot \hat{M}_S \leq \mathbf{f}$.

In the PN-based modelling framework, the behavior of a S^3PGR^2 net, $\mathcal{N} = (P_0 \cup P_S \cup P_R, T, W, M_0)$, under the control of a dynamic algebraic LES with γ resource allocation alternatives per process stage, is represented by another S^3PGR^2 net, $\mathcal{N}^c = (P_0^c \cup P_S^c \cup P_R^c, T^c, W^c, M_0^c)$, where: (i) $P_0^c = P_0$, $P_S^c = \bigcup_{p_j \in P_S} \{p_j^i, \ i = 1, \ldots, \gamma\}$, and $P_R^c = P_R \cup \{w_i, \ i = 1, \ldots, N\}$; (ii) $M_0^c(p) = M_0(p), \ \forall p \in P_0 \cup P_R$, $M_0^c(p) = 0, \ \forall p \in P_S^c$ and $M_0^c(w_i) = f_i, \ i = 1, \ldots, N$; (iii) every transition $t \in T$ with

${}^{\bullet}t = \{p_j\} \subset P_S$ and $t^{\bullet} = \{p_k\} \subset P_S$ is represented in T^c by γ^2 transitions, $t^l,\ l = 1, \ldots, \gamma^2$, interconnecting every place $p_j^i,\ i = 1, \ldots, \gamma$ to every place $p_k^i,\ i = 1, \ldots, \gamma$; similarly every transition t connecting a place $p_j \in P_0$ (resp. P_S) to a place $p_k \in P_S$ (resp. P_0) in the original net S^3PGR^2, is represented in T^c by γ transitions connecting p_j (resp., each $p_j^i,\ i = 1, \ldots, \gamma$) to each $p_k^i,\ i = 1, \ldots, \gamma$ (resp., p_k); (iv) finally, the connectivity of T^c to the resource set P_R^c expresses the effective resource allocation taking place during the process advancement through the various stage alternatives. It is obvious from this construction, that if N and γ are polynomially related to the size of the original S^3PGR^2 net, $\mathcal{N}$, the size of the net $\mathcal{N}^c$, modelling the controlled system behavior, is polynomially related to the size of $\mathcal{N}$. Hence, *the considered dynamic algebraic LES can be implemented and executed with polynomial computational cost.*

Policy Mixtures One way to synthesize a γ-alternative dynamic algebraic LES for a given S^3PGR^2 net $\mathcal{N} = (P_0 \cup P_S \cup P_R, T, W, M_0)$, is by interleaving the policy-induced resource allocation requirements for each place $p \in P_S$, generated by γ simple algebraic LES $(\mathbf{A}_i, \mathbf{f}),\ i = 1, \ldots, \gamma$, with the same rhs vector $\mathbf{f}$. The resulting LES essentially *mixes* the defining logic of its constituent policies, since it allows a process to execute any given stage through the effective resource allocation specified by any of the constituent policies. This basic characterization of the policy mixture through the synthesis of a dynamic algebraic LES, can be further refined, by recognizing that, given two columns $\mathbf{d}_{p_k}^i$ and $\mathbf{d}_{p_k}^j$ from the resulting LES-defining matrix $\mathcal{D}$, with $\mathbf{d}_{p_k}^i > \mathbf{d}_{p_k}^j$,[3] the policy-induced resource allocation alternative $\mathbf{d}_{p_k}^i$ can be dropped from matrix $\mathcal{D}$ without compromising the correctness or the permissiveness of the resulting policy. Hence, under this more concise realization, the LES-defining matrix $\mathcal{D}$ will contain only minimal columns of matrices $\mathbf{A}_i,\ i = 1, \ldots, \gamma$, with the min operation taken on a process stage-by-process stage basis. In the following discussion, columns that have been dropped from the $\mathcal{D}$ matrix of a (policy mixture-based) dynamic algebraic LES will be represented by a column full of ∞.

In general, there is no guarantee that the dynamic algebraic LES resulting from the mixture of γ correct algebraic LES, as described above, will successfully enforce the liveness of the controlled system. Yet, next we present a particular dynamic algebraic LES, resulting from the mixing of a number of G-RUN implementations, which is guaranteed to be a correct LES for the underlying RAS. Furthermore, it is shown that this new LES class is more permissive than the LES that results from the standard disjunction of the constituent LES's (i.e., the LES under which a RAS state is accepted iff there exists at least one constituent policy accepting the state).

Dynamic G-RUN LES Given a well-marked S^3PGR^2 net, $\mathcal{N} = (P_0 \cup P_S \cup P_R, T, W, M_0)$, a dynamic G-RUN LES for it is defined by the mixture of γ different G-RUN LES, with matrices $\mathbf{A}^1, \mathbf{A}^2, \ldots, \mathbf{A}^{\gamma}$, computed based on a *fixed* resource ordering $o() : \mathcal{R} \rightarrow \{1, \ldots, m\}$, but γ different communities $\Psi_1, \Psi_2, \ldots, \Psi_{\gamma} \in \mathcal{C}_{\mathcal{N}}$. Hence, based on the above discussion on dynamic algebraic LES and policy mixtures, the *dynamic G-RUN LES* defined by the $\mathbf{A}^1, \mathbf{A}^2, \ldots, \mathbf{A}^{\gamma}$ matrices, is the dynamic algebraic LES $(\mathcal{D} = [\mathbf{d}_{p_1}^1 \ldots \mathbf{d}_{p_1}^{\gamma} \ldots \mathbf{d}_{p_\sigma}^1 \ldots \mathbf{d}_{p_\sigma}^{\gamma}], \mathbf{f} = (f_i)_{i=1,\ldots,m}))$ with $f_i = C_i,\ i = 1, \ldots, m$, and $\forall p_k \in P_S,\ \forall j = 1, \ldots, \gamma$,

$$\mathbf{d}_{p_k}^j = \begin{cases} \mathbf{a}_{p_k}^j & \text{if } \nexists \mathbf{a}_{p_k}^l \text{ s.t. } \mathbf{a}_{p_k}^l < \mathbf{a}_{p_k}^j, l = 1, \ldots, \gamma \\ (\infty) & \text{otherwise} \end{cases} \tag{10}$$

[3]Given two N-dimensional vectors $\mathbf{a}_1$ and $\mathbf{a}_2$, $\mathbf{a}_1 > \mathbf{a}_2$ iff $\forall i = 1, \ldots, N$, $\mathbf{a}_1[i] \geq \mathbf{a}_2[i]$, and $\mathbf{a}_1 \neq \mathbf{a}_2$.

Theorem 1: The PN $\mathcal{N}^c = (P_0^c \cup P_S^c \cup P_R^c, T^c, W^c, M_0^c)$ representing a dynamic G-RUN implementation on a well-marked S^3PGR^2 net, $\mathcal{N} = (P_0 \cup P_S \cup P_R, T, W, M_0)$, is live.

Proof: Since, according to the previous discussion, the net $\mathcal{N}^c$ belongs to the class of S^3PGR^2 nets, it suffices to show that the policy-induced resource allocation requirements specified by the dynamic G-RUN LES satisfy Constraints (2) - (5) for a resource ordering $o() : \mathcal{R} \rightarrow \{1, \ldots, m\}$ and a community $\Psi \in \mathcal{C}_{\mathcal{N}^c}$. For this, suppose that the considered dynamic G-RUN LES mixes γ simple G-RUN LESs, each of which is defined by matrix $\mathbf{A}^k,\ k = 1, \ldots, \gamma$. Let $o'() : \mathcal{R} \rightarrow \{1, \ldots, m\}$ and Ψ_k respectively denote the common resource ordering and the community employed in the k-th G-RUN implementation. Then, it follows that for $k = 1, \ldots, \gamma$, the triplet $< \mathbf{A}^k,\ o'(),\ \Psi_k >$ satisfies Constraints (2) - (5). Set $o() \equiv o'()$. A community $\Psi \in \mathcal{C}_{\mathcal{N}^c}$ s.t. Ψ, $o()$, and $\mathcal{D}$ satisfy Constraints (2) - (5), is constructed as follows: Without loss of generality, consider $p^k \in P_S^c$ for some $k \in \{1, \ldots, \gamma\}$ and $p \in P_S$ s.t. $p^{\bullet\bullet} \cap P_0 = \emptyset$. If $\exists q^k \in P_S^c$ (i.e., the corresponding column has not been dropped from $\mathcal{D}$) and $(p, q) \in \Psi_k$, we set $g(p^k) \equiv \{q^k\}$ in the definition of Ψ. Constraints (2) - (5) are immediately satisfied. Otherwise, all places $q^k \in P_S^c$ with $(p, q) \in \Psi_k$ have been removed during the construction of $\mathcal{N}^c$. Then, there must exist $q \in p^{\bullet\bullet} \cap P_S$ with $(p, q) \in \Psi_k$, and $d \in \{1, \ldots, \gamma\} - \{k\}$ with $q^d \in P_S^c$, s.t. $\mathbf{a}_q^k > \mathbf{a}_q^d$. We set $g(p^k) \equiv \{q^d\}$ in the definition of Ψ. Since $(\mathbf{a}_p^k, \mathbf{a}_q^k)$ satisfies Constraints (2) - (5), it follows that $(\mathbf{a}_p^k, \mathbf{a}_q^d)$ also satisfies them. $\diamond$

The combination of the next theorem and example establishes that the dynamic G-RUN LES can be more permissive than the LES resulting from the disjunction of the constituent policies.

Theorem 2: Given a well-marked S^3PGR^2 net, $\mathcal{N} = (P_0 \cup P_S \cup P_R, T, W, M_0)$ with $P_S = \{p_1, \ldots, p_\sigma\}$, consider γ different G-RUN LES matrices, $\mathbf{A}^1, \mathbf{A}^2, \ldots, \mathbf{A}^{\gamma}$, computed based on a fixed resource ordering $o() : \mathcal{R} \rightarrow \{1, \ldots, m\}$, but γ different communities $\Psi_1, \Psi_2, \ldots, \Psi_{\gamma} \in \mathcal{C}_{\mathcal{N}}$. The dynamic G-RUN LES $(\mathcal{D} = [\mathbf{d}_{p_1}^1 \ldots \mathbf{d}_{p_1}^{\gamma} \ldots \mathbf{d}_{p_\sigma}^1 \ldots \mathbf{d}_{p_\sigma}^{\gamma}], \mathbf{f})$, constructed by Equation (10) is at least as permissive as the supervisor disjunction of $(\mathbf{A}^1, \mathbf{f}), (\mathbf{A}^2, \mathbf{f}), \ldots, (\mathbf{A}^{\gamma}, \mathbf{f})$.

Proof: Suppose that $M_S = (m_{p_1}, \ldots, m_{p_\sigma})$ is admitted by some LES $(\mathbf{A}^j, \mathbf{f}),\ j = 1, \ldots, \gamma$. Next, we construct a marking $\hat{M}_S = (\mu_{p_1}^1, \ldots, \mu_{p_1}^{\gamma}, \ldots, \mu_{p_\sigma}^1, \ldots, \mu_{p_\sigma}^{\gamma})$ for the net $\mathcal{N}^c$ that corresponds to the same RAS state as the marking M_S, and it is admitted by the dynamic G-RUN LES under consideration. $\hat{M}_S$ is obtained as follows: (i) Initially, set $\hat{M}_S = (0)$; (ii) Subsequently, $\forall m_{p_k}, k = 1, \ldots, \sigma$, if $\mathbf{d}_{p_k}^j \neq (\infty)$, $\mu_{p_k}^j = m_{p_k}$. Otherwise, there exists $\mathbf{d}_{p_k}^l$ for some $l = 1, \ldots, \gamma$, s.t. $\mathbf{d}_{p_k}^l < \mathbf{a}_{p_k}^j$, where $\mathbf{a}_{p_k}^j$ is the column corresponding to process place $p_k \in P_S$ in the G-RUN LES matrix $\mathbf{A}^j$; we set $\mu_{p_k}^l = m_{p_k}$. It follows that (i) $\forall k = 1, \ldots, \sigma$, $\sum_{i=1}^{\gamma} \mu_{p_k}^i = m_{p_k}$ and (ii) $\mathcal{D} \cdot \hat{M}_S \leq \mathbf{A}^j \cdot M_S \leq \mathbf{f}$, i.e., marking $\hat{M}_S$ represents the same RAS state as marking M_S, and it is admitted by the dynamic G-RUN LES. $\diamond$

Example 3 Consider the two G-RUN LES implementations expressed by Equation (7) and Equation (8), obtained in Example 2 by the use of two different community sets. Based on the construction procedure defined in Equation (10), we obtain the following dynamic G-RUN LES, $\mathcal{D} \cdot \hat{M}_S \leq \mathbf{C}$, where:

$$\mathcal{D} = \begin{bmatrix} 2\,2 & 1\,1 & 1\,1 & 1\,\infty & 0\,\infty & \infty\,0 & 0\,0 & 0\,0 & 0\,0 \\ 0\,0 & 1\,1 & 0\,0 & 0\,\infty & 2\,\infty & \infty\,0 & 1\,1 & 0\,0 & 1\,1 \\ 2\,2 & 2\,2 & 2\,2 & 1\,\infty & 0\,\infty & \infty\,1 & 0\,0 & 2\,2 & 1\,1 \end{bmatrix} \tag{11}$$

The dynamic G-RUN defined by Equation (11) is more permissive than the disjunction of the G-RUN LESs defined by

Equations (7) and (8). Indeed, from Theorem 2, every state admitted by the supervisor disjunction is also admitted by the dynamic G-RUN. However, the dynamic G-RUN admits also the state $M_S = (0,0,0,0,2,2,0,0,0)^T$ which is not admitted by any of the constituent G-RUN LESs (c.f. Eqs 7 and 8). The corresponding state representation in the dynamic G-RUN implementation is $\hat{M}_S = (0,0,0,0,0,0,0,-,2,-,-,2,0,0,0,0,0,0)^T$; the dashes '-' indicate the vector elements that multiply the (∞) columns of $\mathcal{D}$, which would be dropped from the matrix in an actual, more concise implementation. $\diamond$

Enhancing the permissiveness of dynamic G-RUN LES The specialization to the G-RUN LES of the discussion on the computational complexity of the implementation of dynamic algebraic LES on any given S^3PGR^2 net, implies that a G-RUN LES will be polynomially implementable as long as the number of the constituent policies, γ, is polynomially related to the size of the underlying S^3PGR^2 net. However, if one is willing to forego the requirement for polynomial real-time execution of the resulting policy instantiation, an even more flexible implementation of a dynamic G-RUN LES can be obtained as follows: Given a S^3PGR^2 net, a partial ordering on the system resources, $o() : \mathcal{R} \rightarrow \{1,\ldots,m\}$, and γ different communities, the matrix $\mathcal{D}$, representing the resulting dynamic G-RUN LES, can be computed by solving γ linear programs and taking the minimal columns from the resulting matrices. Then, the admissibility of a state M_S is decided by the following test: A state $M_S = (m_{p_1},\ldots,m_{p_\sigma})$ is admissible iff $\exists$ a positive integer vector $\hat{M}_S = (\mu^1_{p_1},\ldots,\mu^{\gamma_1}_{p_1},\ldots,\mu^1_{p_\sigma},\ldots,\mu^{\gamma_\sigma}_{p_\sigma})$ s.t. $\mathcal{D}\cdot\hat{M}_S \leq \mathbf{f}$ and $\forall k = 1,\ldots,\sigma$, $\sum_{j=1}^{\gamma_k}\mu^j_{p_k} = m_{p_k}$. This can be tested on-line by solving a system of linear diophantine equations [14]. Notice that the solution of this system of equations essentially tries to *logically*[4] redistribute the active process instances across the various policy-induced control resource allocation requests in a way that will satisfy the constraint of Equation (9). On the other hand, the integral nature of the system variables renders it a proposition of non-polynomial complexity.

Finally, it should be noted that the flexibility of the dynamic G-RUN LES can be further enhanced when it is applied in connection with the following supervisor disjunction scheme: Given γ different communities and ω different resource orderings, we first compute the dynamic G-RUN LES for each ordering, obtaining ω dynamic G-RUN LESs, $\mathcal{D}_1,\ldots,\mathcal{D}_\omega$. Then, it follows that a state represented by $\hat{M}_S$ is admissible iff $\exists\mathcal{D}_l$ s.t. $\mathcal{D}_l\cdot\hat{M}_S \leq \mathbf{f}$, for some $l = 1,\ldots,\omega$. The LES resulting from the disjunction of dynamic G-RUN LESs will be at least as permissive as any of its constituent dynamic supervisors, and it could potentially admit additional states, not admitted by any of the constituent LES.

IV. Conclusions

This paper presented a method for enhancing the permissiveness of the class of algebraic LESs that has been investigated actively in the literature as a computationally viable tool for enforcing the liveness of various RAS classes. The proposed class of LESs, named dynamic algebraic LESs, was able to provide increased permissiveness by taking better advantage of the routing flexibility inherent in contemporary RASs. Specifically, this paper considered the G-RUN LES, a class of algebraic LESs proposed in [8], that allows supervisor parameterization based on the routing flexibility, and augmented its structure so that alternative effective resource allocation requirements for each process stage can be allowed. It also provided a computationally efficient method for constructing a dynamic algebraic LES from a set of G-RUN LESs, instantiated through different routing schemes. Finally, the proposed approach was shown to be more powerful than the standard supervisor disjunction.

It is interesting to notice that any class of RASs allowing conjunctive resource allocations is closed under implementation of an algebraic LES. On the other hand, the closure property of S^3PGR^2 nets with respect to the embedding of a dynamic algebraic LES can be attributed to the fact that the S^3PGR^2 net also allows for alternate process routings. Accordingly, we believe that the motivating ideas and the correctness proof method presented in this paper can be extended to other RAS and/or supervisor classes, as long as (i) the considered RAS class allows for conjunctive resource allocation and routing flexibility, and (ii) the proposed policy mixture preserves the structure/property underlying the correctness of the constituent policies. One such application in the context of an RAS that augments the CD-RAS with the features of reworks and uncontrollable events, can be found in [11]. Future work will focus on the problem of selecting polynomially-sized sets of communities in a way that tends to maximize the operational permissiveness/flexibility of the resulting policy mixture.

Acknowledgement

The authors would like to thank the Associate Editor and the paper reviewers who, with their thorough comments, enhanced the legibility of the original manuscript.

References

[1] S. A. Reveliotis and P. M. Ferreira, "Deadlock avoidance policies for automated manufacturing cells," *IEEE Transactions on Robotics & Automation*, vol. 12, no. 6, pp. 845–857, 1996.
[2] M. Lawley and S. Reveliotis, "Deadlock avoidance for sequential resource allocation systems : Hard and easy cases," *International Journal of Flexible Manufacturing Systems*, vol. 13, pp. 385–404, 2001.
[3] Z. A. Banaszak and B. H. Krogh, "Deadlock avoidance in flexible manufacturing systems with concurrently competing process flows," *IEEE Transactions on Robotics & Automation*, vol. 6, no. 6, pp. 724–734, 1990.
[4] M. P. Fanti, B. Maione, S. Mascolo, and B. Turchiano, "Event-based feedback control for deadlock avoidance in flexible production systems," *IEEE Transactions on Robotics & Automation*, vol. 13, pp. 347–363, 1997.
[5] M. Lawley, S. Reveliotis, and P. Ferreira, "A correct and scalable deadlock avoidance policy for flexible manufacturing systems," *IEEE Transactions on Robotics & Automation*, vol. 14, no. 5, pp. 796–809, 1998.
[6] J. Park and S. A. Reveliotis, "A polynomial-complexity deadlock avoidance policy for sequential resource allocation systems with multiple resource acquisitions and flexible routings," in *Proceedings of the IEEE International Conference on Decision & Control*. IEEE, 2000.
[7] J. Park and S. A. Reveliotis, "Algebraic synthesis of efficient deadlock avoidance policies for sequential resource allocation systems," *IEEE Transactions on Robotics & Automation*, vol. 16, no. 2, pp. 190–195, 2000.
[8] J. Park and S. A. Reveliotis, "Algebraic deadlock avoidance policies for conjunctive/disjunctive resource allocation systems," in *Proc. of the IEEE International Conferenfce on R&A*. IEEE, 2001.
[9] M. A. Lawley, "Integrating flexible routing and algebraic deadlock avoidance policies in automated manufacturing systems," *Intl. Jrnl of Prod. Res.*, vol. 38, pp. 2931–2950, 2000.
[10] J. Ezpeleta, J. M. Colom, and J. Martinez, "A petri net based deadlock prevention policy for flexible manufacturing systems," *IEEE Transactions on Robotics & Automation*, vol. 11, pp. 173–184, 1995.
[11] J. Park, *Structural Analysis and Control of Resource Allocation Systems using Petri nets*, Ph.D. thesis, Georgia Institute of Technology, Atlanta, GA, 2000.
[12] W. Reisig, *Petri nets : An introduction*, Springer-Verlag, 1985.
[13] J. Park, S. A. Reveliotis, M. A. Lawley, and P. M. Ferreira, "A correction to the run dap for conjunctive ras presented in 'polynomial complexity deadlock avoidance policies for sequential resource allocation systems'," *IEEE Transactions on Automatic Control*, vol. 46, pp. 672, 2001.
[14] A. Schrijver, *Theory of Linear and Integer Programming*, John Wiley & Sons, 1998.

[4]Remember that all tokens of the markings $\mu^j_{p_k}$, $j = 1,\ldots,\gamma_k$, correspond to process instances executing the same process stage in the original RAS, under an identical allocation from the original resource set $\mathcal{R}$.