Algebraic Deadlock Avoidance Policies for Sequential Resource Allocation Systems*

Spyros Reveliotis
School of Industrial & Systems Engineering
Georgia Institute of Technology
spyros@isye.gatech.edu

Abstract

As many contemporary technological applications move to operational modes of more extensive and flexible automation, there is a rising need to design and control the underlying resource allocation not only for efficiency, but also for logical correctness and internal consistency. The material presented in this chapter offers a unifying and comprehensive treatment of a class of policies that have been proposed as an effective and efficient solution to this emerging class of logical control problems.

1 Introduction

This chapter deals with the problem of managing the resource allocation that takes place in various contemporary technological applications, including flexibly automated production systems, automated railway and/or monorail transportation systems, electronic workflow management systems, and business transaction supporting systems. A distinguishing trait of all the aforementioned applications is that they seek to limit the role of the human element to remote high-level supervision, while placing the burden of the real-time monitoring and coordination of the ongoing activity upon a computerized control system. This development is justified by a number of technical, economic and safety considerations, and it is facilitated by the advent of modern computing and sensing technologies. At the same time, the effective support of such an extensively automated operational mode poses new challenges to the designers and supervisors of these systems. A particularly challenging task in the emerging regime is the synthesis of the *control logic* that will manage the allocation of the resources of the aforementioned systems to the various running processes in a way that guarantees the orderly and expedient execution of all these processes, while preserving the operational *flexibility* sought by these environments.

The applications depicted in Figures 1 and 2 exemplify the aforementioned problem and they highlight the currently prevailing practice. Figure 1 depicts a small robotic cell[1] with three processing stations, W_1, W_2, and W_3. Each of these stations can accommodate only one part at a time, and collectively they support the production of two different part types, J_1 and J_2, whose processing routes are annotated in the figure. It should be clear to the reader that the state depicted in Figure 1 is a problematic state, since the two depicted jobs mutually block each other. Furthermore, this blockage will persist until it is realized and resolved, probably only through human intervention, by unloading one of the two jobs, a

*Some of the material presented in this chapter originally appeared in J. Park and S. Reveliotis, *"Deadlock Avoidance in Sequential Resource Allocation Systems with Multiple Resource Acquisitions and Flexible Routings"*, IEEE Trans. on Automatic Control, vol. 46, no.10, pgs 1572-1583 (©[2001] IEEE), S. Reveliotis, *"Implicit Siphon Control and its Role in the Liveness Enforcing Supervision of Sequential Resource Allocation Systems"*, IEEE Trans. on SMC – Part A, vol. 37, no. 3, pgs 319-328 (©[2007] IEEE), S. Reveliotis, *"Real-Time Management of Resource Allocation Systems: A Discrete Event Systems Approach"*, (©[2005] Springer), S. Reveliotis and J. Y. Choi, *"Designing Reversibility-Enforcing Supervisors of Polynomial Complexity for Bounded Petri nets through the Theory of Regions"*, LNCS 4024, pgs 322-341 (©[2006] Springer).

[1]The depicted configuration is very similar in its basic topology to the *cluster tools* used extensively in the contemporary semiconductor manufacturing industry.

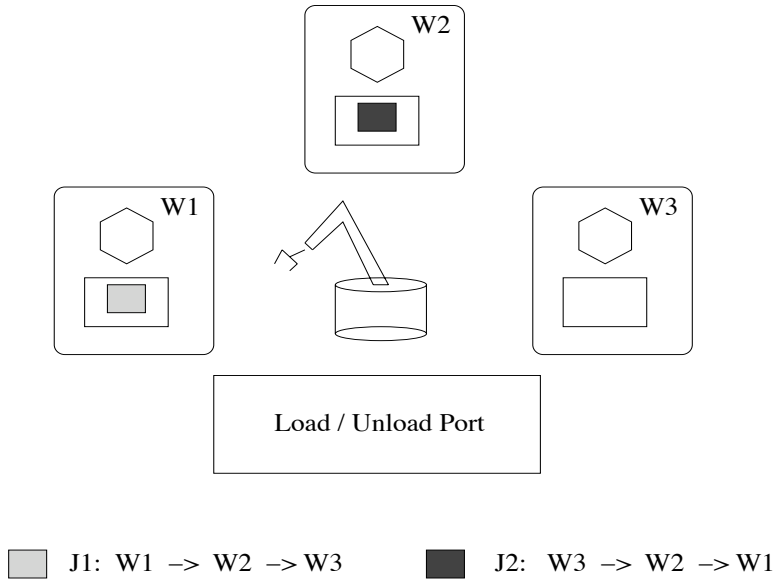

Figure 1: A manufacturing system deadlock

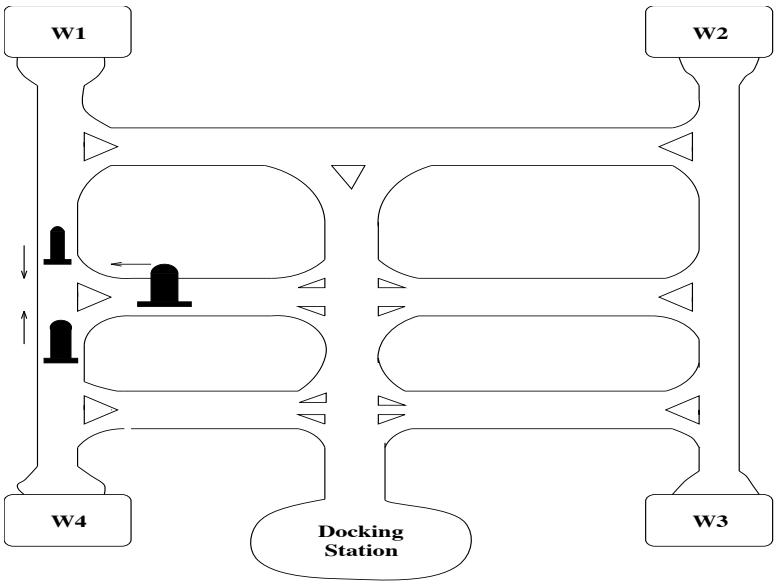

Figure 2: An AGV system deadlock

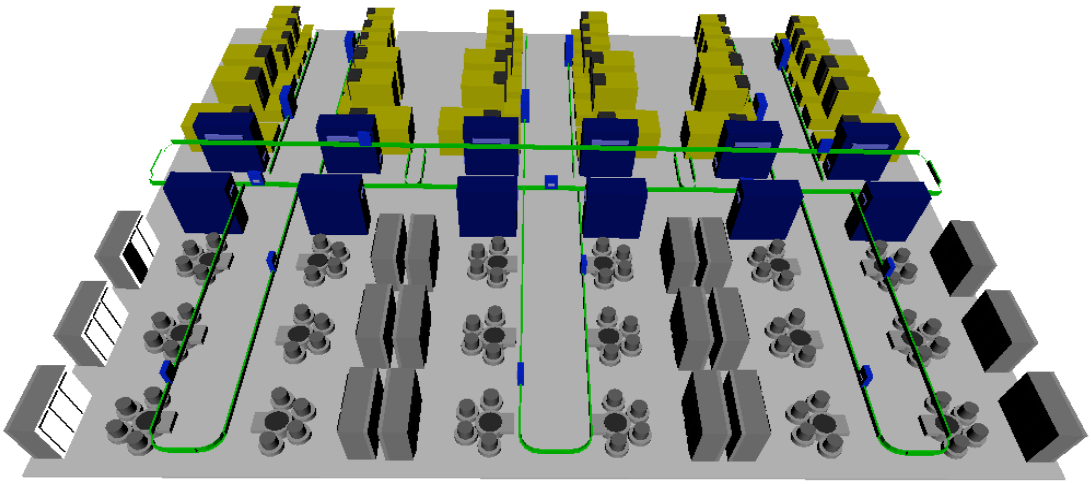

Figure 3: A tandem AGV system

rather costly operation in the considered setting. Similarly, Figure 2 depicts a zone-controlled automated guidance vehicle (AGV) system, where three vehicles block permanently the advancement of each other at a junction of the guidepath network.

The situations depicted in Figures 1 and 2 are known respectively as a manufacturing and an AGV *deadlock*. In operational contexts relying on manual labor, such deadlock problems have been typically addressed through last-minute improvisation. However, in a fully automated context, the resolution of these problems must be part of the overall design process. In lack of a systematic methodology to address these issues, past engineering practice has resorted to rather simplistic approaches that provide a robust solution to the problem, but only at the expense of the system operational flexibility, efficiency and productivity. Hence, in order to prevent the occurrence of the manufacturing deadlock mentioned above, most contemporary cells are operated in a *"batching"* mode, that separates the production of the supported part types. By preventing the concurrent production of the supported parts, the system will always be operated in a unidirectional flow that is free of any deadlocking problems. However, such a solution is a substantial departure from the notion of flexible automation and its advertised advantages. In a similar spirit, most contemporary AGV systems are designed according to the *"tandem"* configuration depicted in Figure 3, where the entire guidepath network is decomposed to a number of unidirectional loops, interfacing at a number of strategically preselected points. While managing to avoid deadlock, tandem AGV systems have to experience expensive "hand-off" procedures at the loop junctions, and the vehicle filing at any single loop implies that the pacing in that loop is determined by the slowest vehicle.

The effective and systematic resolution of the aforementioned deadlock problems must be based on a detailed study of the event sequences that take place during the system operation. Hence, the analysis and resolution of these problems necessitates a methodological framework that places the emphasis on the analysis and shaping of these event sequences, and it is substantially different from those that have been traditionally applied to performance-oriented control.[2] Such a methodological framework has been provided by an area of modern control theory known as *qualitative* or *logical analysis and control of Discrete Event Systems (DES)* [3], and the last 15 years have seen the emergence of a substantial body of results on the aforementioned deadlock problems that are based on representations and analytical tools coming from this area. The fundamental abstraction that underlies the development of these results is

[2]While it is true that event timing can provide a mechanism for enforcing event sequences, such an approach will tend to be very brittle in the face of the stochasticity characterizing the operation of the considered applications.

the *sequential resource allocation system (RAS)*, formally defined as follows:

Definition 1 *[17] A* sequential resource allocation system (RAS) *is defined as a 5-tuple* $\Phi =< \mathcal{R}, C, \mathcal{P}, \mathcal{A}, \mathcal{T} >$, *where:*

1. $\mathcal{R} = \{R_1, \ldots, R_m\}$ *is the set of the system* resource types.
2. $C : \mathcal{R} \rightarrow Z^+$ *– the set of strictly positive integers*[3] *– 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.
3. $\mathcal{P} = \{\Pi_1, \ldots, \Pi_n\}$ *denotes the set of the system* process types *supported by the considered system configuration. Each process type* Π_j *is a composite element itself, in particular,* $\Pi_j =< \mathcal{S}_j, \mathcal{G}_j >$, *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) $\mathcal{G}_j$ *represents some data structure communicating some sequential logic that applies to the execution of any process instance of type* Π_j.
4. $\mathcal{A} : \bigcup_{j=1}^{n} \mathcal{S}_j \rightarrow \prod_{i=1}^{m} \{0, \ldots, C_i\}$ *is the* resource allocation function *associating every processing stage* Ξ_{jk} *with a* resource allocation request $\mathcal{A}(j,k) \equiv A_{jk}$. *More specifically, each* A_{jk} *is an* m*-dimensional vector, with its* i*-th component indicating the number of resource units of resource type* R_i *necessary to support the execution of stage* Ξ_{jk}. *Obviously, in a well-defined RAS,* $A_{jk}(i) \leq C_i, \ \forall j, k, i$.
5. $\mathcal{T} : \bigcup_{j=1}^{n} \mathcal{S}_j \rightarrow \mathcal{D}$ *is the* timing function*, corresponding to each processing stage* Ξ_{jk} *a distribution* D_{jk} *that characterizes the statistics of the processing time* t_{jk}*, experienced during the execution of stage* Ξ_{jk}.

The above characterization of the considered RAS is further qualified by the following conditions that detail their operation and facilitate the subsequent analysis:

Condition 1 Under expedient resource allocation, every activated process instance will terminate in a *finite* number of processing steps.

Condition 2 Every processing stage $\Xi_{jk} \in \mathcal{S}_j$ can be realized by at least one execution sequence supported by $\mathcal{G}_j$.

Condition 3 A process instance j_j advances from stage Ξ_{jk} to a successor stage $\Xi_{j,k+1}$ only upon being allocated the entire set of resources implied by the resource allocation request $A_{j,k+1}$. The allocation of all these resources takes place simultaneously, and it is only at this point that the process instance j_j releases the resources allocated to it for the execution of processing stage Ξ_{jk}.

Condition 4 The only way in which two distinct activated process instances can interact with each other, is through their potential contest for some of the system resources.

Condition 1 excludes those pathological situations in which an executing process can entangle itself in an infinite loop. In well-designed applications, a process will not be allowed to run within the system indefinitely. From a representational standpoint, the satisfaction of this assumption allows the modeling of the process-defining logic through an *acyclic* data structure. Condition 2 essentially ensures that the process representation does not introduce redundant processing stages. Condition 3 introduces the *"hold-while-waiting"* effect in the considered resource allocation which is at the base of the considered

[3]Also, in this document, Z_0^+ will denote the set of nonnegative integers, Z will denote the set of all integers, and $\Re$ will denote the set of reals.

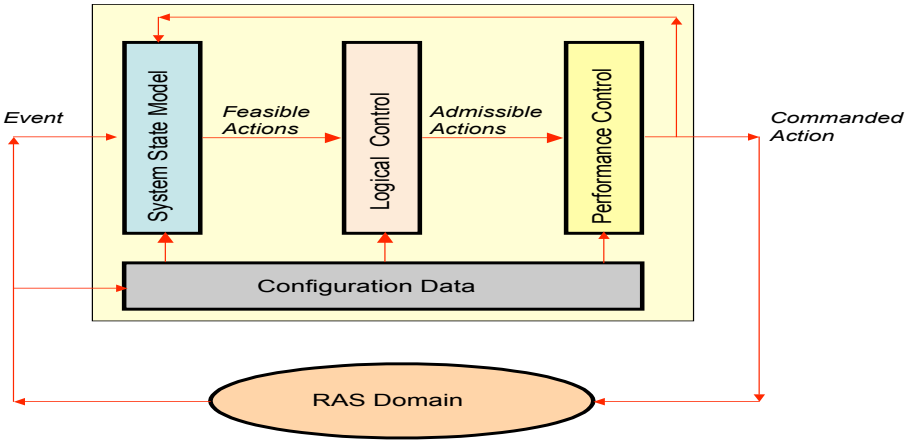

Figure 4: An event-driven framework for the RAS Supervisory Control problem

deadlock problems.[4] Condition 4 applies primarily to complex process flows that involve parallelization, and implies that the logic coordinating the execution of the various process threads does not "confound" enacted sub-processes belonging to different process instantiations. Finally, in order to facilitate the subsequent discussion on the complexity of the posed problems and the proposed solutions, we also introduce the quantity $|\Phi| \equiv |\mathcal{R}| + |\bigcup_{j=1}^{n} \mathcal{S}_j| + \sum_{i=1}^{m} C_i$, which defines the *size* of the RAS Φ.

The problem of the real-time management of the resource allocation taking place in the considered RAS can be effectively addressed through the supervisory control (SC) framework depicted in Figure 4. As indicated in Figure 4, the proposed controller is *event-driven*, i.e., the control actions commanded to the underlying RAS can be perceived as the controller responses to the various events taking place in the RAS domain and communicated to the controller through a monitoring function. Hence, the entire control function evolves in a number of *cycles*, with each cycle being triggered by a RAS event communicated to the controller. Conceptually, each cycle consists of three major phases: (i) In Phase I, the controller updates a representation of the RAS *state* so that it represents the RAS status after the occurrence of the communicated event. This representation, combined with the system knowledge about the running RAS configuration, encodes the entire set of *feasible actions* that could be executed by the RAS as a response to the occurring event. (ii) In Phase II, the controller applies a *logical control policy* in order to filter out from the set of feasible actions identified in Phase I, the set of *admissible actions*, i.e., this set of actions that satisfy some logical specification for the RAS behavior, like deadlock freedom. (iii) Finally, in Phase III, the set of admissible actions is provided to the *performance-oriented component* of the RAS supervisor in order to select the one that will be communicated eventually to the RAS environment, in a way that observes some performance considerations. In addition to this basic functionality, the RAS controller should be able to respond to the various contingencies taking place in the RAS domain, by (i) appropriately updating the RAS configuration database, and (ii) revising

[4] As demonstrated by the examples presented in Figures 1 and 2, this "hold-while-waiting" effect frequently results from the need to physically buffer the various process instances at any single point in time, i.e., parts processed in a flexibly automated production system or vehicles in an AGV network are physical entities and they always need to be accommodated somewhere during their sojourn through the system. It must be noticed, however, that, while providing the necessary specificity for the underlying resource allocation dynamics, the aforestated assumptions do not compromise the modeling power of our framework, since one can capture any additional resource allocation dynamics by augmenting the specification of process Π_j. For example, one can model the fact that, at some particular process stage Ξ_{jk}, process Π_j might release (some of) its currently allocated resources before advancing to stage $\Xi_{j.k+1}$, by introducing to the process specification an intermediate process stage Ξ_{jq}, with resource allocation request A_{jq} equal to A_{jk} minus the deallocated resource set.

the logical and performance-oriented control logic in order to apply in the emerging RAS configuration. This last function will be collectively characterized as *(re-)configuration management*.

A systematic exposition of most of the existing results concerning the design and deployment of the RAS control function depicted in Figure 4 can be found in [17, 28]. The relevant theory provides: (i) formal characterizations of the underlying RAS dynamics in the context of *Finite State Automata (FSA)* [7] and *Petri net (PN)* [11] modeling frameworks, as well as some more *ad hoc* representations; (ii) a characterization of the *optimal* logical control problem for the considered RAS; (iii) a rigorous study of the complexity of the aforementioned optimal control problem, that establishes its *NP-hardness* [6] but also identifies important practical cases that admit optimal solutions of polynomial complexity with respect to the size $|\Phi|$ of the underlying RAS; (iv) efficient solutions for the remaining cases that trade off optimality for computational tractability; (v) a formal characterization of the notions of *robustness* and *re-configurability* for the considered operational context; and (vi) some preliminary results regarding the effective integration of the logical and the performance-oriented control.

The material presented in this chapter complements the aforementioned results by providing a unifying treatment of a particular class of RAS logical control policies known as *algebraic*. More specifically, the chapter aggregates and systematizes a number of results regarding this class of policies that emerged during the last three years, partly in response to problems and thoughts generated during the development of [17]. Collectively, the presented results offer (a) a thorough characterization of the considered class of policies, (b) effective computational tools for their design and implementation on any given RAS instance, and (c) an insightful explanation of the mechanisms underlying the policy effectiveness and computational tractability. From an organizational standpoint, the chapter will evolve as follows: Section 2 provides a systematic characterization of the RAS dynamics by means of the PN modeling framework. Subsequently, Section 3 introduces the class of algebraic logical control policies and it establishes that they also admit an effective representation within the PN modeling framework. Section 4 provides an analytical characterization of the entire set of algebraic logical control policies that can ensure the deadlock-free operation of any given RAS. Beyond its theoretical interest, this characterization enables also the introduction of a notion of optimality within the scope of the considered policies. In principle, such an optimized implementation is effectively computable through the presented developments. However, from a more practical standpoint, this computation is limited by a very high complexity. Hence, Section 5 offers an additional approach that can enable the synthesis of algebraic logical control policies for any given RAS while drastically mitigating the complexity problems arising from the previous approach. Section 6 offers some interesting and fundamental insights regarding the mechanism that facilitates the functionality of the considered policies. Finally, Section 7 concludes the chapter and suggests some directions for future developments. Throughout the following discussion, the emphasis is placed on the systematic and accessible presentation of the key results and their implications. Therefore, we have frequently omitted the detailed technical arguments underlying the relevant derivations; these arguments can be traced in the provided citations.

2 A PN-based representation of the considered RAS

As mentioned in the introductory section, Petri nets [11] have been one of the primary modeling frameworks employed for the analysis and control of the RAS dynamics considered in this work. In this section,we define the PN subclass that characterizes the RAS behavior encompassed by Definition 1, presuming that the reader is already familiar with the basic PN concepts.[5]The subsequent discussion proceeds in three steps: (i) first we introduce a PN model that expresses the execution logic of any single process instance; (ii) subsequently, this model is augmented with resource places in order to represent the dynamics of the associated resource allocation; and finally (iii) the complete RAS model is obtained by merging the various subnets developed in step (ii) through their common resource places.

PN-based modeling of the RAS process types In the PN modeling framework, the process type $\Pi_j =< \mathcal{S}_j, \mathcal{G}_j >$, introduced in item 3 of Definition 1, will be represented by the concept of the *process*

[5]A primer on the key PN concepts employed in this work is provided in the Appendix; for a more extensive discussion, the interested reader is referred to [11] .

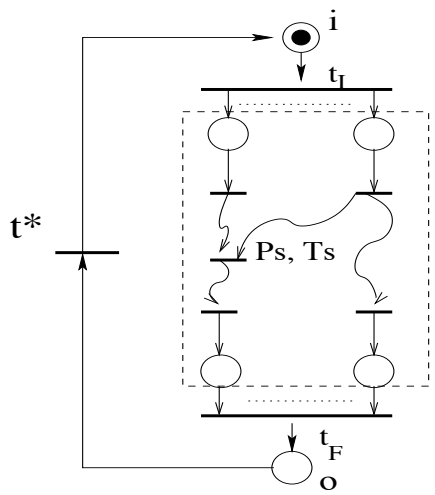

Figure 5: The process net structure of Definition 2

subnet, formally defined as follows:

Definition 2 *A* process (sub-)net *is an ordinary Petri net* $\mathcal{N}_P = (P, T, W, M_0)$ *such that:*

i. $P = P_S \cup \{i,\ o\}$ *with* $P_S \neq \emptyset$*;*

ii. $T = T_S \cup \{t_I,\ t_F,\ t^*\}$*;*

iii. $i^\bullet = \{t_I\}$*;* ${}^\bullet i = \{t^*\}$*;*

iv. $o^\bullet = \{t^*\}$*;* ${}^\bullet o = \{t_F\}$*;*

v. $t_I^\bullet \subseteq P_S$*;* ${}^\bullet t_I = \{i\}$*;*

vi. $t_F^\bullet = \{o\}$*;* ${}^\bullet t_F \subseteq P_S$*;*

vii. $(t^*)^\bullet = \{i\}$*;* ${}^\bullet(t^*) = \{o\}$*;*

viii. the underlying digraph is strongly connected*;*

ix. $M_0(i) > 0\ \wedge\ M_0(p) = 0,\ \forall p \in P \backslash \{i\}$*;*

x. $\forall M \in R(\mathcal{N}_P, M_0),\ M(i) + M(o) = M_0(i) \Longrightarrow M(p) = 0,\ \forall p \in P_S$.

The PN-based process representation introduced by Definition 2 is depicted in Figure 5. Process instances waiting to initiate processing are represented by tokens in place i, while the initiation of a process instance is modelled by the firing of transition t_I. Similarly, tokens in place o represent completed process instances, while the event of a process completion is modelled by the firing of transition t_F. Transition t^* allows the token re-circulation – i.e., the token transfer from place o to place i – in order to model *repetitive* process execution. Finally, the part of the net between transitions t_I and t_F, that involves the process places P_S, models the sequential logic defining the considered process type. In particular, places $p \in P_S$ correspond to the various processing stages $\Xi_{jk} \in \mathcal{S}_j$, while the net connectivity among these places concretizes component $\mathcal{G}_j$ of Π_j (c.f. item (3b) of Definition 1). As it can be seen in Definition 2, this part of the process subnet can be quite arbitrary. However, in order to capture the requirements posed by Conditions 1, 2 and 4 in Section 1, we further qualify the considered process subnets through the following three assumptions:

Assumption 1 *The process subnets considered in this work are assumed to be* acyclic*, i.e., the removal of transition* t^* *from them renders them acyclic digraphs.*

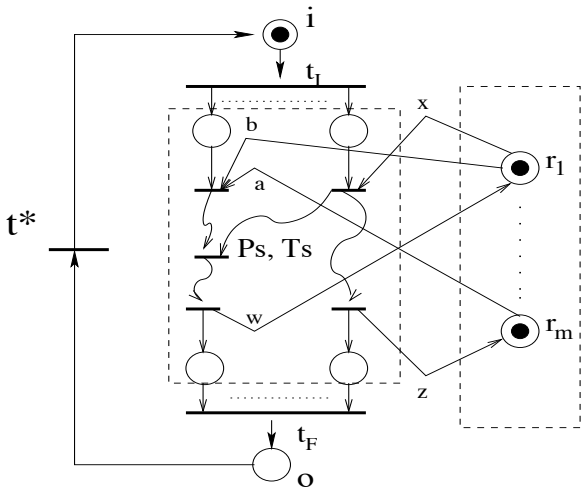

Figure 6: The resource-augmented process net

Assumption 2 *The process subnets considered in this work are assumed to be* quasi-live *for* $M_0(i) = 1$.

Assumption 3 *The process subnets considered in this work are assumed to be* strongly reversible*, i.e., their initial marking* M_0 *can be reached from any marking* $M \in R(\mathcal{N}_P, M_0)$*, through a firing sequence that does not contain transition* t_I.

Assumption 1 is introduced in order to satisfy Condition 1 of Section 1. Assumption 2 pertains to the satisfaction of Condition 2, by essentially stipulating that, in the considered process subnets, every transition models a meaningful event that can actually occur during the execution of some process instance, and therefore, it is not redundant. Assumption 3 pertains to the satisfaction of Condition 4, as it essentially stipulates that, at any point in time and under expedient resource allocation, all *active* process instances can advance to completion. Since the main focus of this work is on the analysis and control of the resource allocation function taking place in the considered environments, we forego the further study of the process subnets themselves and the investigation of the structural and behavioral properties implied by Definition 2 and Assumptions 1-3. The interested reader can find some relevant discussion and results in [24, 25, 26, 27, 8].

PN-based modeling of the resource allocation function The modeling of the resource allocation associated with the process stage Ξ_{jk} corresponding to any place $p \in P_S$, necessitates the augmentation of the process subnet $\mathcal{N}_P$, defined above, with a set of *resource* places $P_R = \{r_l,\ l = 1, \ldots, m\}$, of initial marking $M_0(r_l) = C_l,\ l = 1, \ldots, m$, and with the corresponding flow sub-matrix, Θ_{P_R}, expressing the allocation and de-allocation of the various resources to the process instances as they advance through their processing stages, according to the protocol stipulated by Condition 3 in Section 1. The resulting net will be called the *resource-augmented process (sub-)net* and it will be denoted by $\overline{\mathcal{N}_P}$. Its basic structure is depicted in Figure 6. Notice that the characterization of transitions t^*, t_I and t_F provided in the above discussion, implies that $(t^*)^\bullet \cap P_R = {}^\bullet(t^*) \cap P_R = (t_I)^\bullet \cap P_R = {}^\bullet(t_F) \cap P_R = \emptyset$. On the other hand, the *reusable* nature of the system resources presumed in this work, is modelled by the following assumption regarding the resource-augmented process net $\overline{\mathcal{N}_P}$:

Assumption 4 *Let* $\overline{\mathcal{N}_P} = (P_S \cup \{i,\ o\} \cup P_R, T, W, M_0)$ *denote a resource-augmented process (sub-)net. Then,* $\forall l \in \{1, \ldots, |P_R|\}$*, there exists a p-semiflow* y_{r_l} *s.t.: (i)* $y_{r_l}(r_l) = 1$*; (ii)* $y_{r_l}(r_j) = 0,\ \forall j \neq l$*; (iii)* $y_{r_l}(i) = y_{r_l}(o) = 0$*; (iv)* $\forall p \in P_S,\ y_{r_l}(p) =$ *number of units from resource* R_l *required for the execution of the processing stage modelled by place* p.

Furthermore, the following assumption extends the requirement for *quasi-liveness* of the process net $\mathcal{N}_P$, introduced by Assumption 2, to the resource-augmented process net $\overline{\mathcal{N}_P}$:

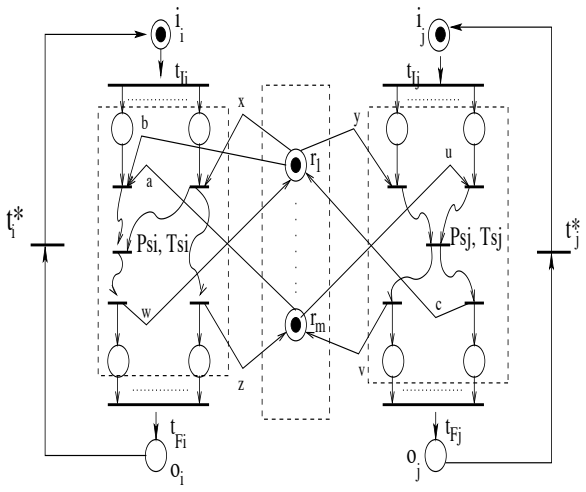

Figure 7: The process-resource net structure considered in this work

Assumption 5 *The resource-augmented process subnets considered in this work are assumed to be* quasi-live *for* $M_0(i) = 1$ *and* $M_0(r_l) = C_l,\ \forall l \in \{1, \ldots, |P_R|\}$.

In general, assessing the quasi-liveness of a resource-augmented process net is an NP-hard problem [22, 17]. However, the main source of this complexity is the presence of synchronizing transitions in the underlying process net, and therefore, the problem remains polynomial in the quite frequent case that this process net is a state machine. In such a case, assessing the quasi-liveness of the resource-augmented process net is tantamount to validating that for every resource R_i, $C_i \geq \max_{j,k}\{A_{jk}(i)\}$, or equivalently in the PN formalism, that for all $r_l \in P_R$ and for all $p \in P_S$, $M_0(r_l) \geq y_{r_l}(p)$, where y_{r_l} are the p-semiflows introduced in Assumption 4. Some interesting and quite powerful computational tests for assessing quasi-liveness for the remaining cases can be found in [8, 16].

The complete RAS model: Process-resource nets The complete PN-based model, $\mathcal{N} = (P, T, W, M_0)$, of any given instance from the RAS class considered in this work is obtained by *merging* the resource-augmented process nets $\overline{\mathcal{N}_{P_j}} = (P_j, T_j, W_j, M_{0_j}),\ j = 1, \ldots, n$, modeling its constituent process types, through their common resource places. The resulting PN class is characterized as the class of *process-resource nets with acyclic, quasi-live and strongly reversible process subnets*, and its basic structure is depicted in Figure 7. Let $P_S = \bigcup_j P_{S_j}$; $I = \bigcup_j \{i_j\}$; $O = \bigcup_j \{o_j\}$, and $P = \bigcup_j P_j = P_S \cup I \cup O \cup P_R$. Then, the re-usable nature of the resource allocation taking place in the entire process-resource net is characterized by a p-semiflow y_{r_l} for each resource type $R_l,\ l = 1, \ldots, m$, defined by: (i) $y_{r_l}(r_l) = 1$; (ii) $y_{r_l}(r_j) = 0,\ \forall j \neq l$; (iii) $y_{r_l}(i_j) = y_{r_l}(o_j) = 0,\ \forall j$; (iv) $\forall p \in P_S,\ y_{r_l}(p) = y_{r_l}^{(j*)}(p)$, where $\overline{\mathcal{N}_{P_{j*}}}$ denotes the resource-augmented process subnet containing place p, and $y_{r_l}^{(j*)}()$ denotes the corresponding p-semiflow for resource R_l. Furthermore, it is easy to see that Assumption 5, regarding the quasi-liveness of the constituent resource-augmented process subnets $\overline{\mathcal{N}_{P_j}}$, implies also the quasi-liveness of the entire process-resource net $\mathcal{N}$. Finally, in the PN modeling framework, the size of a RAS Φ, modelled by a net $\mathcal{N} = (P_S \cup I \cup O \cup P_R, T, W, M_0)$, is defined as $|\Phi| = |\mathcal{N}| \equiv |P_R| + |P_S| + \sum_{r \in P_R} M_0(r)$.

Example 1 Figure 8 depicts the process-resource net modeling the resource allocation taking place in the robotic cell of Figure 1. Each of the resource places $r_i,\ i = 1, 2, 3$, models the unit buffering capacity of the corresponding workstation in Figure 1, while the processing of an instance of part type $J_j,\ j = 1, 2$, is modeled by the path $< t_{j0}, p_{j1}, t_{j1}, p_{j2}, t_{j2}, p_{j3}, t_{j3} >$. Furthermore, in the depicted net we have also adopted the common practice of aggregating the path $< i, t^*, o >$ of the underlying process nets to a single place p_0, that is called the process *idle place*. Hence, the state depicted in Figure 8 corresponds to the initial RAS state, where the system is idle and empty of any processes. Finally, notice that we have set $M_0(i_j) = 3$, $j = 1, 2$, so that these values do not constrain artificially the system loading. More

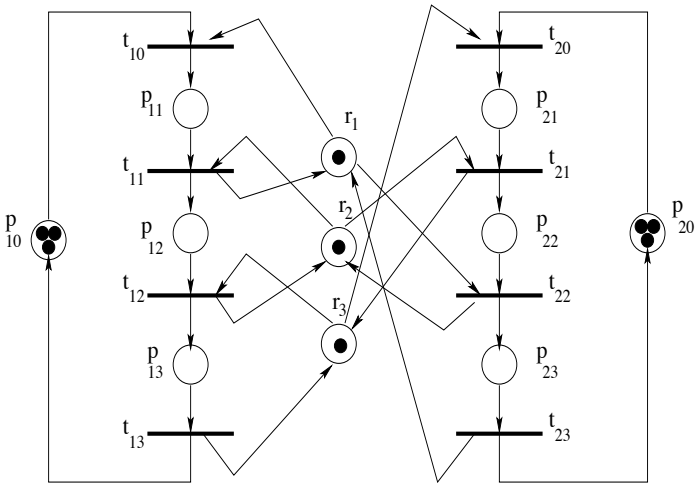

Figure 8: The process-resource net modeling the resource allocation taking place in the robotic cell of Figure 1

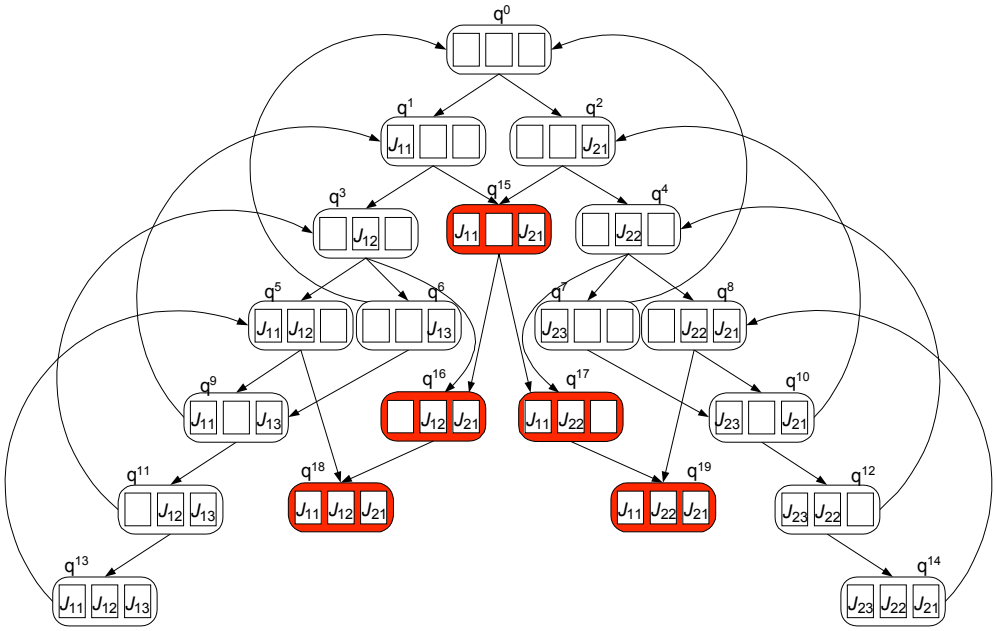

Figure 9: The reachability space of the process-resource net depicted in Figure 8; each depicted state indicates the occupancy of the workstation buffers in the original cell, while unsafe states are depicted in red

generally, in the proposed PN-based RAS modeling, $M_0(i_j)$ must be set to a value that is an *upper bound* to the maximum number of process instances from process type Π_j that can be simultaneously loaded in the system. Such an upper bound will always exist due to the finiteness of the system resources.

RAS deadlock and deadlock avoidance In the PN-based modeling framework, the formation of RAS deadlock is expressed by the lack of reversibility of the corresponding process-resource net. Furthermore, in the underlying reachability space, $R(\mathcal{N}, M_0)$, this lack of reversibility is graphically represented by the formation of *strongly connected components* that are not *co-reachable*, i.e., the initial marking M_0, is not reachable from them through any sequence of feasible transitions. As a case in point, Figure 9 depicts the reachability graph of the process-resource net of Figure 8, where it can be clearly seen that four reachable net markings are not co-accessible.

In the light of these characterizations, a *correct Deadlock Avoidance Policy (DAP)*, Δ, must try to restrict the system operation to a *strongly connected component of the underlying reachability space, $R(\mathcal{N}, M_0)$, which contains the initial marking* M_0. The RAS subspace that is reachable under – or *admissible* by – some DAP Δ will be denoted by $R_\Delta(\mathcal{N}, M_0)$. Given a RAS configuration, an applied DAP is characterized as *optimal*, if the corresponding admissible subspace is the *maximal* strongly connected component of $R(\mathcal{N}, M_0)$ which contains the initial marking M_0. The set of markings admitted by the optimal DAP, Δ^*, is characterized as the (set of *reachable*) *safe* markings, and it will denoted by $R_s(\mathcal{N}, M_0)$. The complement of $R_s(\mathcal{N}, M_0)$ with respect to $R(\mathcal{N}, M_0)$ is denoted by $R_u(\mathcal{N}, M_0)$, and it constitutes the *(reachable) unsafe* markings.

The finiteness of the reachability space $R(\mathcal{N}, M_0)$ for the considered RAS class implies that the optimal DAP, Δ^*, is well-defined, and it is effectively computable through an *one-step lookahead* scheme that admits a tentative resource allocation if and only if (*iff*) the resulting marking is safe. However, the underlying safety problem is NP-complete, in general [1]. 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 $R_s(\mathcal{N}, M_0)$. This idea has been formalized by the concept of *Polynomial Kernel (PK-)* DAP [17]. From an implementational standpoint, a typical approach to the design of PK-DAP's is the identification of a property $\mathcal{H}(M),\ M \in R(\mathcal{N}, M_0)$, such that (i) the complexity of testing $\mathcal{H}()$ on the RAS markings is polynomial with respect to the RAS size, (ii) $\mathcal{H}(M_0) = \texttt{TRUE}$, and (iii) the subspace $\{M \in R(\mathcal{N}, M_0) : \mathcal{H}(M) = \texttt{TRUE}\}$ is strongly connected. The reader is referred to [17, 28] for a broad set of results regarding the design of PK-DAPs for various subclasses of the RAS class introduced in Definition 1, and also, for the identification of special RAS structure for which the optimal DAP Δ^* is implementable with polynomial complexity with respect to the corresponding RAS size, $|\Phi|$. In the rest of this chapter we focus on a particular subclass of PK-DAPs that is known as *algebraic*, and we present a series of results regarding the analysis and the design of these policies.

3 Algebraic DAPs and their PN-based representation

Definition of algebraic PK-DAPs *Algebraic* PK-DAP's are defined as the particular class of PK-DAP's where the property $\mathcal{H}(M)$ constitutes a system of linear inequalities on the RAS marking M that is polynomially sized with respect to the RAS size $|\Phi|$. Furthermore, since the marking of the resource places $r \in P_R$ and of the process idle places, $p_{j0},\ j = 1, \ldots, n$, is determined by the marking of the remaining places $p \in P_S$, which expresses the state of the active process instances, the aforementioned inequalities will constrain explicitly only the restriction of M to its components corresponding to places $p \in P_S$, M_S. Hence, a typical algebraic PK-DAP will have the form:

$$A \cdot M_S \leq b \tag{1}$$

where A is a nonnegative integer matrix with K rows, b is a K-dimensional nonnegative integer vector, and K is polynomially related to the RAS size $|\Phi|$.

A PN-based representation of algebraic PK-DAPs The constraints expressed by Equation 1 can be enforced in the PN-based representation of the RAS dynamics through the super-imposition on

the original process-resource net of a controlling subnet that is readily constructed through the theory of *control-place invariants* presented in [10]. According to [10], each of the inequalities

$$A(k,\cdot)\cdot M_S \leq b(k), \quad k=1,\ldots,K \tag{2}$$

can be imposed on the net behavior by superimposing on the original net structure a *control* place $p_c(k)$. The connectivity of place $p_c(k)$ to the rest of the network is determined by the flow matrix

$$\theta_{p_c(k)} = -A(k,\cdot)\cdot\Theta_S \tag{3}$$

where Θ_S denotes the flow sub-matrix of the uncontrolled network $\mathcal{N}=(P,T,W,M_0)$ corresponding to places $p\in P_S$. The initial marking of place $p_c(k)$ is set to

$$M_0(p_c(k)) = b(k) \tag{4}$$

The resulting controller imposes Constraint 2 on the original system behavior by establishing the place invariant

$$A(k,\cdot)\cdot M_S + M(p_c(k)) = b(k) \tag{5}$$

Example 2 Figure 10 depicts the implementation on the process-resource net of Figure 8 of an algebraic PK-DAP, Δ, that is expressed by the following set of inequalities on the marking M_S:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} M(p_{11}) \\ M(p_{12}) \\ M(p_{13}) \\ M(p_{21}) \\ M(p_{22}) \\ M(p_{23}) \end{bmatrix} \leq \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \tag{6}$$

The reader can verify that the connectivity and the initial marking of the monitor places w_1, w_2 and w_3, that enforce each of the three constraints of Equation 6, satisfies the requirements of Equations 3 and 4, as well as Equation 5. The correctness of policy Δ for the considered process-resource net is manifested by Figure 11, that depicts the policy-admissible subspace, $R_\Delta(\mathcal{N},M_0)$, and it is formally proven in [20].

Monitor places as fictitious resources Equation 5, when interpreted in the light of Assumption 4 of Section 2, implies that the control places $p_c(k)$, implementing each of the constraints in the policy-defining Equation 1, essentially play the role of *fictitious* new resources in the dynamics of the net $\mathcal{N}^c$, that models the controlled system behavior.[6] As a result, the controlled net $\mathcal{N}^c$ remains in the class of process-resource nets that satisfy Assumptions 1 and 3. Let $P_C \equiv \bigcup_k \{p_c(k)\}$. If it can be shown that the net $\mathcal{N}^c$ satisfies also Assumption 5 with respect to the extended "resource" set $P_R \cup P_C$, then it can be inferred that $\mathcal{N}^c$ belongs to the class of process-resource nets with acyclic, quasi-live and strongly reversible process subnets, and therefore, all the analytical insights and results regarding the logical behavior of the uncontrolled RAS extend to their controlled counterparts. This remark will be especially useful in Section 5 where we derive correctness tests for algebraic PK-DAPs based on the structural properties of process-resource nets.

4 An analytical characterization of the class of algebraic DAPs

This section provides an analytical characterization for the entire set of algebraic PK-DAPs, Δ, that employ (up to) K constraints and can establish deadlock-free operation for some given process-resource net $\mathcal{N}$. Beyond its inherent theoretical interest, such a characterization enables also an "optimal" selection of the implemented policy Δ from the considered set of policies. The subsequent discussion addresses the aforementioned problem by considering the more general problem of characterizing the

[6] This effect is manifested also in Figure 10.

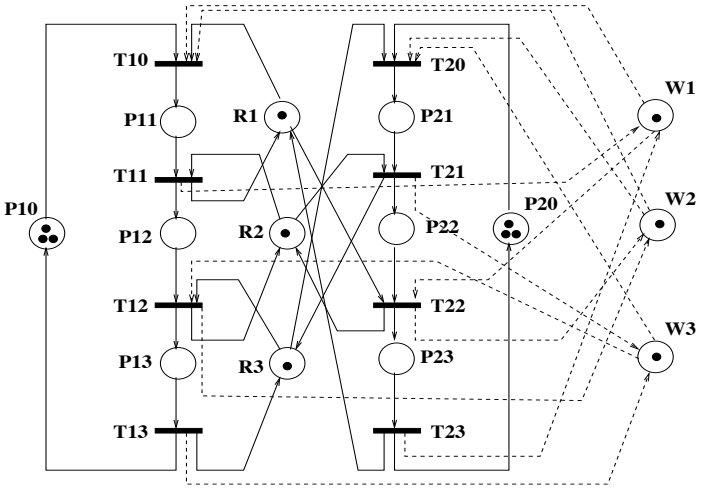

Figure 10: The process-resource net of Figure 8 under the control of the algebraic DAP of Equation 6

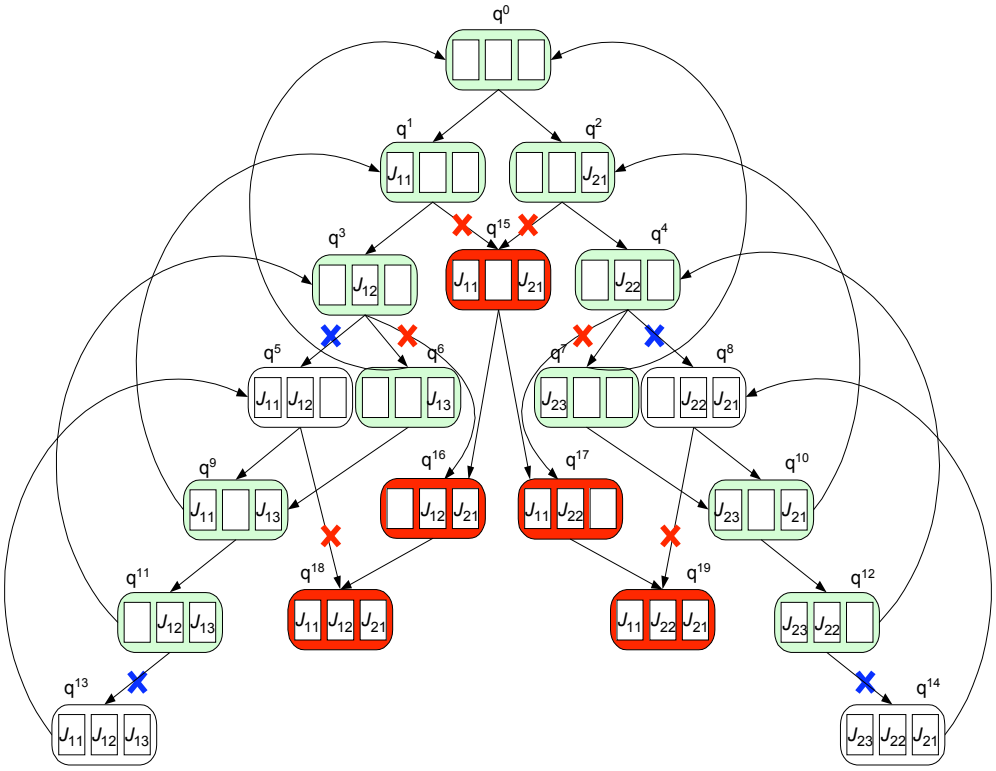

Figure 11: The policy-admissible space for the controlled net depicted in Figure 10: the admissible states are depicted in green

class of algebraic supervisors that employ (upto) K constraints and enforce the reversibility of some bounded PN $\mathcal{N}$; a solution to this extended problem can be easily customized to the notion of algebraic PK-DAPs for process-resource nets by restricting appropriately some elements of the policy-defining matrix A in the derived formulation. Hence, a formal statement of the problem considered in this section is as follows:

A formal statement of the considered problem Given a non-reversible, bounded PN $\mathcal{N}$, identify a set of constraints

$$A \cdot M \leq b \tag{7}$$

such that

i. when imposed on the plant net $\mathcal{N}$, they will incur the reversibility of the controlled system;

ii. the cardinality of the imposed constraint set must not exceed a pre-specified parameter K;

iii. the matrix A and the vector b satisfy the constraint

$$\forall i, j,\ A(i,j) \in \{0, 1, \ldots, \bar{A}(i,j)\} \text{ and } \forall i,\ b(i) \in \{0, 1, \ldots, \bar{b}(i)\}, \tag{8}$$

where $\bar{A}(i,j)$ and $\bar{b}(i)$ are *finitely* valued, externally provided parameters;

iv. and finally, assuming that every reachable marking $M_i \in R(\mathcal{N}, M_0)$ of $\mathcal{N}$ is associated with some value w_i, the developed supervisor will maximize the total value of the admissible markings, over the set of supervisors satisfying the previous three requirements.

In the sequel, a PN supervisor that is defined by Equation 7 for some pricing of matrix A and vector b, will be referred to as the supervisor $\mathcal{S}(A, b)$.

Overview of the proposed solution Next, we provide a *Mixed Integer Programming (MIP)* formulation for the aforestated problem. The objective function of this formulation will express the optimality requirement stated in item (iv) above. Requirement (ii) will be captured by the structure of the decision variables of the presented formulation, while requirements (i) and (iii) will be explicitly encoded in its constraints. More specifically, given a pricing of the matrix A and the rhs vector b, the constraint set must check whether this pricing abides to requirement (iii) and it must also assess the ability of this pricing to satisfy requirement (i), i.e., establish the reversibility of the controlled system. This last requirement further implies that all the markings $M \in R(\mathcal{N}, M_0)$ that remain reachable under the policy constraints, are also co-reachable under these constraints. Hence, the constraint set of the proposed formulation must be able to assess the reachability and co-reachability of the markings $M \in R(\mathcal{N}, M_0)$ under the net supervision by any tentative constraint set, $A \cdot M \leq b$, and it must also be able to validate that all reachable markings are also co-reachable. The rest of this section proceeds to the detailed derivation of a formulation that possesses the aforementioned qualities.

Characterizing the net transition firing under supervision by $\mathcal{S}(A, b)$ In order to be able to assess the reachability and co-reachability of the various markings $M \in R(\mathcal{N}, M_0)$ under supervision by supervisor $\mathcal{S}(A, b)$, it is necessary to characterize how the various transitions, $t \in T$, of the plant net $\mathcal{N}$, retain their fireability in the controlled system. Next, we introduce a set of variables and constraints that will achieve this purpose. The main issue to be addressed is whether a transition t that was fireable in some marking $M_i \in R(\mathcal{N}, M_0)$, leading to another marking $M_j \in R(\mathcal{N}, M_0)$, will remain fireable under supervision by $\mathcal{S}(A, b)$. For this to be true, t must be enabled at M_i by all the monitor places, $p_c(k)$, $k = 1, \ldots, K$, that implement the supervisor $\mathcal{S}(A, b)$. Testing whether transition t is enabled at marking M_i by a monitor place $p_c(k)$ can be done through the employment of a *binary* variable z_{ij}^k, that will be priced to one, if this condition is true, and to zero, otherwise. A set of constraints that will enforce the pricing of z_{ij}^k according to the aforementioned scheme is the following:

$$M_0(p_c(k)) + \sum_{(u,v)\in\xi(i)} \Theta(p_c(k), t(u,v)) + \Theta(p_c(k), t(i,j)) + (z_{ij}^k - 1)L_{ij}^k \geq 0 \tag{9}$$

$$M_0(p_c(k)) + \sum_{(u,v)\in\xi(i)} \Theta(p_c(k), t(u,v)) + \Theta(p_c(k), t(i,j)) - z_{ij}^k U_{ij}^k \leq -1 \tag{10}$$

The parameter $\xi(i)$ appearing in Equations 9 and 10 denotes any path in $R(\mathcal{N}, M_0)$ leading from M_0 to M_i. (u,v) denotes an edge of $\xi(i)$ leading from node M_u to node M_v, and $t(u,v)$ denotes its labeling transition. L_{ij}^k denotes a lower bound for the quantity $M_0(p_c(k)) + \sum_{(u,v)\in\xi(i)} \Theta(p_c(k), t(u,v)) + \Theta(p_c(k), t(i,j))$, and U_{ij}^k denotes an upper bound for the quantity $M_0(p_c(k)) + \sum_{(u,v)\in\xi(i)} \Theta(p_c(k), t(u,v)) + \Theta(p_c(k), t(i,j)) + 1$. Then, it is clear that when $M_0(p_c(k)) + \sum_{(u,v)\in\xi(i)} \Theta(p_c(k), t(u,v)) + \Theta(p_c(k), t(i,j)) \geq 0$ – i.e., when transition $t(i,j)$ is enabled by monitor place $p_c(k)$ in marking M_i – the above set of constraints is satisfied by setting $z_{ij}^k = 1$. On the other hand, when $M_0(p_c(k)) + \sum_{(u,v)\in\xi(i)} \Theta(p_c(k), t(u,v)) + \Theta(p_c(k), t(i,j)) < 0$ the above constraint set is satisfied by setting $z_{ij}^k = 0$.

It remains to connect the variables $M_0(p_c(k))$ and $\Theta(p_c(k), \cdot)$ to the primary problem variables, A, b, and explain how to compute the bounds L_{ij}^k and U_{ij}^k employed in the above equations. Connecting $M_0(p_c(k))$ and $\Theta(p_c(k), \cdot)$ to the variables A, b can be done straightforwardly through Equations 3 and 4; the corresponding substitutions respectively transform Equations 9 and 10 to:

$$b(k) - \sum_{(u,v)\in\xi(i)} A(k,\cdot)\cdot\Theta(\cdot, t(u,v)) - A(k,\cdot)\cdot\Theta(\cdot, t(i,j)) + (z_{ij}^k - 1)L_{ij}^k \geq 0 \tag{11}$$

$$b(k) - \sum_{(u,v)\in\xi(i)} A(k,\cdot)\cdot\Theta(\cdot, t(u,v)) - A(k,\cdot)\cdot\Theta(\cdot, t(i,j)) - z_{ij}^k U_{ij}^k \leq -1 \tag{12}$$

Finally, it should be clear from the structure of Constraints 11 and 12 that the bound L_{ij}^k (resp., U_{ij}^k), defined above, can be obtained by minimizing (resp., maximizing) the quantity $b(k) - \sum_{(u,v)\in\xi(i)} A(k,\cdot)\cdot\Theta(\cdot, t(u,v)) - A(k,\cdot)\cdot\Theta(\cdot, t(i,j))$ over the space defined by the admissible ranges of the involved variables $A(k,\cdot)$ and $b(k)$ (c.f., item (iii) in the formal problem statement provided at the beginning of this section).

Once variables z_{ij}^k have been properly priced for all k, the feasibility of $M_i \xrightarrow{t(i,j)} M_j$ can be assessed by introducing another *real* variable, z_{ij}, that is priced according to the following constraints:

$$z_{ij} \leq z_{ij}^k, \quad \forall k \in \{1, \ldots, K\} \tag{13}$$

$$z_{ij} \geq \sum_{k=1}^{K} z_{ij}^k - K + 1 \tag{14}$$

$$0 \leq z_{ij} \leq 1 \tag{15}$$

To understand the pricing logic behind Constraints 13–15, first notice that Constraint 15 restricts the variable z_{ij} within the interval $[0, 1]$. Then, Constraint 13 sets it to zero, as long as any of the variables z_{ij}^k is priced to zero – and therefore, the corresponding monitor place $p_c(k)$ disables $t(i,j)$. On the other hand, when all variables z_{ij}^k are priced to one, Constraint 14 forces variable z_{ij} to its extreme value of one.

Characterizing the reachability of the markings $M_i \in R(\mathcal{N}, M_0)$ under supervision by $\mathcal{S}(A,b)$ The availability of the variables z_{ij}, defined above, subsequently enables the characterization of the reachability of the various markings $M_i \in R(\mathcal{N}, M_0)$ under supervision by the supervisor $\mathcal{S}(A,b)$. This can be done by introducing the *real* variables y_i^l, $0 \leq i \leq |R(\mathcal{N}, M_0)|$, $0 \leq l \leq \bar{l}$, and pricing them so that $y_i^l = 1$ indicates that marking M_i is reachable from the initial marking M_0 under supervision by $\mathcal{S}(A,b)$ and the minimum length of any transition sequence leading from M_0 to M_i is l; if M_i is not reachable from M_0 under supervision by $\mathcal{S}(A,b)$, y_i^l should be set to zero, for all l. Clearly, in order to satisfy this definition of y_i^l, $\bar{l}$ must be set to the length of the maximum path in $\mathcal{G}(\mathcal{N}, M_0)$ that starts from M_0 and contains no cycles. Then, a set of constraints that achieves the pricing of y_i^l described above, is as follows:

$$y_i^0 = \begin{cases} 1, & i = 0 \\ 0, & i \neq 0 \end{cases} \tag{16}$$

$$0 \leq y_i^l, \quad \forall i \in \{1, \ldots, |R(\mathcal{N}, M_0)|\}, \; l \in \{1, \ldots, \bar{l}\} \tag{17}$$

$$\sum_{l=0}^{\bar{l}} y_i^l \leq 1 \tag{18}$$

$$\delta_{ji}^l \leq y_j^{l-1}, \quad \forall j : (M_j, M_i) \in \mathcal{G}(\mathcal{N}, M_0) \tag{19}$$

$$\delta_{ji}^l \leq z_{ji}, \quad \forall j : (M_j, M_i) \in \mathcal{G}(\mathcal{N}, M_0) \tag{20}$$

$$y_i^l \leq \sum_j \delta_{ji}^l \tag{21}$$

$$y_i^l \geq y_j^{l-1} + z_{ji} - 1 - \sum_{q=0}^{l-1} y_i^q, \quad \forall j : (M_j, M_i) \in \mathcal{G}(\mathcal{N}, M_0) \tag{22}$$

Constraint 16 expresses the fact that marking M_0 is reachable from itself in zero steps, under supervision by $\mathcal{S}(A, b)$, and this is the only marking in $R(\mathcal{N}, M_0)$ possessing this property. Constraint 17 states the nonnegative real nature of variables $y_i^l, \; i > 0, \; l > 0$, while Constraint 18 expresses the fact that, according to the pricing scheme discussed above, only one of the variables $y_i^l, \; 0 \leq l \leq \bar{l}$, can be priced to one. Constraints 19, 20 and 21 express the fact that, under supervision by $\mathcal{S}(A, b)$, there is a minimal path from marking M_0 to marking M_i of length l, only if there is a minimal path of length $l-1$ from M_0 to some marking M_j such that (i) $(M_j, M_i) \in \mathcal{G}(\mathcal{N}, M_0)$ and (ii) this transition remains feasible under $\mathcal{S}(A, b)$. In particular, variables δ_{ji}^l is a set of auxiliary *real* variables that are used to force y_i^l to zero every time that the aforestated condition is violated for all the markings $M_j \in R(\mathcal{N}, M_0)$ such that $(M_j, M_i) \in \mathcal{G}(\mathcal{N}, M_0)$. On the other hand, Constraint 22 tends to price variable y_i^l to one every time that there exists a marking M_j such that (i) $(M_j, M_i) \in \mathcal{G}(\mathcal{N}, M_0)$, (ii) this transition remains feasible under $\mathcal{S}(A, b)$, and (iii) M_j is reachable from M_0 under supervision by $\mathcal{S}(A, b)$ through a minimal path of length $l-1$; however, this pricing is enforced only when the quantity $\sum_{q=0}^{l-1} y_i^q$ appearing in the right-hand-side of this constraint is equal to zero – i.e., only when the marking M_i cannot be reached from the initial marking M_0 through a path of smaller length.

Characterizing the co-reachability of the markings $M_i \in R(\mathcal{N}, M_0)$ under supervision by $\mathcal{S}(A, b)$ Clearly, the co-reachability of a marking $M_i \in R(\mathcal{N}, M_0)$ is equivalent to the reachability of the same marking in the graph $\mathcal{G}^R(\mathcal{N}, M_0)$, obtained from $\mathcal{G}(\mathcal{N}, M_0)$ by reversing all its arcs. In the light of this observation, the set of constraints characterizing the co-reachability of the markings $M_i \in R(\mathcal{N}, M_0)$, under supervision by supervisor $\mathcal{S}(A, b)$, can be obtained through a straightforward modification of the constraint set 16–22, characterizing the reachability of these markings. More specifically, let ψ_i^l be a *real* variable that will be priced to one, if $M_i \in R(\mathcal{N}, M_0)$ is co-reachable under supervision by $\mathcal{S}(A, b)$, and a minimal transition sequence leading from M_i to M_0 has a length equal to l; otherwise, ψ_i^l should be priced to zero. By following a logic similar to that employed in the previous paragraph for the pricing of variables y_i^l, we obtain the following set of constraints for the pricing of variables ψ_i^l:

$$\psi_i^0 = \begin{cases} 1, & i = 0 \\ 0, & i \neq 0 \end{cases} \tag{23}$$

$$0 \leq \psi_i^l, \quad \forall i \in \{1, \ldots, |R(\mathcal{N}, M_0)|\}, \; l \in \{1, \ldots, \tilde{l}\} \tag{24}$$

$$\sum_{l=0}^{\tilde{l}} \psi_i^l \leq 1 \tag{25}$$

$$\eta_{ij}^{l} \leq \psi_{j}^{l-1}, \quad \forall j : (M_i, M_j) \in \mathcal{G}(\mathcal{N}, M_0) \tag{26}$$

$$\eta_{ij}^{l} \leq z_{ij}, \quad \forall j : (M_i, M_j) \in \mathcal{G}(\mathcal{N}, M_0) \tag{27}$$

$$\psi_{i}^{l} \leq \sum_{j} \eta_{ij}^{l} \tag{28}$$

$$\psi_{i}^{l} \geq \psi_{j}^{l-1} + z_{ij} - 1 - \sum_{q=0}^{l-1} \psi_{i}^{q}, \quad \forall j : (M_i, M_j) \in \mathcal{G}(\mathcal{N}, M_0) \tag{29}$$

The parameter $\tilde{l}$, appearing in Equations 24 and 25, denotes the length of the maximum path in $\mathcal{G}^{R}(\mathcal{N}, M_0)$ that leads from node M_0 to node M_i and contains no cycles, and the auxiliary variables η_{ij}^{l}, that appear in Constraints 26 and 27, play a role identical to that played by variables δ_{ji}^{l} in Constraints 19 and 20.

Characterizing the closure of the sub-space that is reachable and co-reachable under supervision by $\mathcal{S}(A, b)$ Let x_i be a *real* variable that will be priced to one when the marking $M_i \in R(\mathcal{N}, M_0)$ is reachable and co-reachable under supervision by $\mathcal{S}(A, b)$, and it will be priced to zero, otherwise. Then, in the light of the above characterizations of reachability and co-reachability, the desired pricing of x_i can be enforced by the following constraints:

$$x_i \leq \sum_{l=0}^{\bar{l}} y_{i}^{l} \tag{30}$$

$$x_i \leq \sum_{l=0}^{\tilde{l}} \psi_{i}^{l} \tag{31}$$

$$x_i \geq \sum_{l=0}^{\bar{l}} y_{i}^{l} + \sum_{l=0}^{\tilde{l}} \psi_{i}^{l} - 1 \tag{32}$$

$$0 \leq x_i \leq 1 \tag{33}$$

Constraint 33 restricts x_i in the interval $[0, 1]$. Then, Constraints 30 and 31 force it to zero, when marking M_i is not reachable or co-reachable. On the other hand, if M_i is both reachable and co-reachable, Constraint 32 forces x_i to its extreme value of one.

Finally, the availability of variables x_i allows us to express the requirement for closure of the sub-space of $R(\mathcal{N}, M_0)$ that is reachable and co-reachable under supervision by $\mathcal{S}(A, b)$, through the following constraint:

$$(1 - x_i) + x_j \geq z_{ij}, \quad \forall i, j : (M_i, M_j) \in \mathcal{G}(\mathcal{N}, M_0) \tag{34}$$

When $x_i = 1$ and $x_j = 0$ – i.e., when x_i belongs to the target space of markings that are reachable and co-reachable under supervision by $\mathcal{S}(A, b)$, but x_j does not belong to this set – Constraint 34 forces variable z_{ij} to zero – i.e., it requires that the corresponding transition $M_i[t(i,j)\rangle M_j$ is disabled by $\mathcal{S}(A, b)$. In any other case, the left-hand-side of Constraint 34 is greater than or equal to one, and therefore, the constraint becomes inactive.

The objective function of the proposed formulation The objective function of the considered formulation is straightforwardly expressed as follows:

$$\max \sum_{i} w_i x_i \tag{35}$$

The correctness of the proposed formulation The next theorem states the correctness of the derived formulation. A formal proof of this result can be based on concepts and arguments coming from the *theory of regions* [2], a theory that concerns the design of PNs from a specification of their reachability space; the reader is referred to [19] for the relevant details.

Theorem 1 *The formulation of Equations 8,11–35 returns an optimal solution to the problem stated at the beginning of this section, provided that such a solution exists; otherwise, this formulation will be infeasible.*

Customizing the derived formulation to the design of algebraic PK-DAPs We remind the reader that, according to the definition of Section 3, algebraic PK-DAPs restrict explicitly only the projection M_S of the entire marking M of the corresponding process-resource net $\mathcal{N}$. This additional feature for the sought supervisors can be readily introduced in the formulation of Equations 8,11–35 by setting $\bar{A}(i,j) = 0$ for all the elements of matrix A corresponding to places $p \notin P_S$.[7] On the other hand, the values for the remaining elements of matrix A and those of vector b should not be artificially restricted by the choice of the corresponding upper bounds, $\bar{A}(i,j)$ and $\bar{b}(i)$; this can be attained by maintaining these bounds to sufficiently large values. The resulting formulation will be always feasible, since it contains the *trivial* policy that confines the RAS to its initial state s_0.[8] Of course, such a result should be interpreted as lack of an effective DAP in the considered policy space. If we want such a negative result to be communicated as *infeasibility* by the proposed formulation, we can add the constraint

$$\sum_{i:i\neq 0} x_i \geq 1 \tag{36}$$

Furthermore, in most practical cases, one would like to enforce the existence of at least one policy-admissible process plan for each process type $\Pi_j,\ j = 1,\ldots,n$. In such a case, letting $IN(j)$ denote the set of transitions corresponding to the initiation of an instance from process type Π_j, we can replace Constraint 36 with the following stronger requirement:

$$\forall j = 1,\ldots,n, \quad \sum_{(i,q)\in IN(j)} z_{iq} \geq 1 \tag{37}$$

Finally, the typical objective for maximal permissiveness of the resulting policy can be communicated in the developed formulation by setting $w_i = 1,\ \forall i$.

Example 3 We demonstrate the efficacy of the design methodology developed in Section 5, by applying it to the design of an algebraic DAP for the PN depicted in Figure 12. This PN models a RAS consisting of three resource types, R_1, R_2, and R_3, with respective capacities $C_1 = C_3 = 1$, and $C_2 = 2$, and supporting two process types, JT_1 and JT_2, whose process plans are respectively modelled by the paths $< t_{10}p_{11}t_{11}p_{12}t_{12}p_{13}t_{13} >$ and $< t_{20}p_{21}t_{21}p_{22}t_{22}p_{23}t_{23} >$. The reachability space, $R(\mathcal{N}, M_0)$, for the PN depicted in Figure 12 is provided in Figure 13, while the detailed characterization of the markings corresponding to the various nodes of the graph of Figure 13 can be found in Table 1.[9] Clearly, the considered net is not reversible, since the states depicted by the darker-shaded nodes in Figure 13 are not co-reachable to M_0.

Two algebraic PK-DAPs for the process-resource net of Figure 12, originally developed in [12], are respectively expressed by the constraint sets:

$$\begin{bmatrix} 1 & & & 1 & 1 & 1 \\ 1 & 1 & & 1 & 1 & \\ 1 & 1 & 1 & 1 & & \end{bmatrix} \cdot M_S \leq \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}. \tag{38}$$

[7] Although, from a computational standpoint, it is preferable to identify all these zero-valued variables and systematically remove them from the formulation.

[8] This policy is expressed by the single constraint $[1\ 1\ \ldots 1]M_s \leq 0$.

[9] Table 1 provides only M_S, i.e., the markings of the places corresponding to the various processing stages, since the markings of the remaining places can be easily obtained from the net invariants corresponding to (i) the reusability of the system resources and (ii) the circuits established by the introduction of the process idle places.

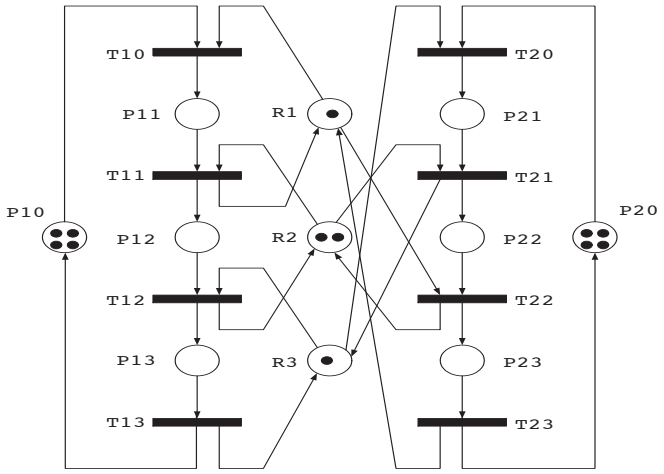

Figure 12: The process-resource net of Example 3

$$\begin{bmatrix} 1 & & & 1 & 1 & 1 \\ 1 & 1 & & 1 & 1 & \\ 1 & 1 & 1 & 1 & & \end{bmatrix} \cdot M_S \leq \begin{bmatrix} 3 \\ 2 \\ 3 \end{bmatrix}. \tag{39}$$

It is interesting to notice that the Constraint set 39 is a relaxation of the Constraint set 38 since $A_1 = A_2$ and $b_1 \leq b_2$. Therefore, the supervisor established by the Constraint set 39 is expected to be more permissive than the supervisor established by the Constraint set 38, and this is indeed reflected in Figure 13 that also depicts the sub-spaces admitted by each of these two supervisors. On the other hand, the application on the net of Figure 12 of the MIP formulation developed in the earlier parts of this section, with the number of policy constraints, K, set equal to 3, returned the following supervisor:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 3 & 0 \\ 0 & 1 & 0 & 2 & 0 & 0 \\ 2 & 2 & 0 & 2 & 3 & 0 \end{bmatrix} \cdot M_S \leq \begin{bmatrix} 6 \\ 3 \\ 8 \end{bmatrix} \tag{40}$$

The sub-space admitted by the supervisor of Equation 40 is also depicted in Figure 13. As it can be seen in this figure, the obtained supervisor manages to recognize the entire safe space of the considered process-resource net, and therefore, it expresses the optimal DAP, Δ^*. Hence, this example corroborates the efficacy and analytical power of the proposed methodology.

Complexity considerations Yet, the practical applicability of the algebraic DAP design methodology developed in this section can be severely limited from the fact that it requires the explicit enumeration of the reachability space, $R(\mathcal{N}, M_0)$, of the underlying process-resource net, $\mathcal{N}$. It is well established in the relevant literature that the size of this state space is, in general, an exponential function of the size of the underlying net, $|\mathcal{N}|$, and it grows very fast. This situation is further complicated by the fact that the presented methodology eventually boils down to the solution of a MIP formulation with a number of variables and constraints that is determined by the size of the space $R(\mathcal{N}, M_0)$. A first approach to alleviate this problem is to restrict the target behavior for the controlled net to a sub-space of $R(\mathcal{N}, M_0)$ so that the resulting formulation is computationally manageable. Another approach is to develop additional methodology that will provide correct algebraic DAPs for some given process-resource net $\mathcal{N}$, while avoiding the explicit enumeration of the reachability space $R(\mathcal{N}, M_0)$. We present such an alternative methodology in the next section.

Table 1: The markings of the reachability space depicted in Figure 13

State	p_{11}	p_{12}	p_{13}	p_{21}	p_{22}	p_{23}	State	p_{11}	p_{12}	p_{13}	p_{21}	p_{22}	p_{23}
0	0	0	0	0	0	0	24	0	1	0	1	1	0
1	1	0	0	0	0	0	25	0	1	0	0	0	1
2	0	0	0	1	0	0	26	1	0	0	0	2	0
3	0	1	0	0	0	0	27	0	0	0	1	2	0
4	1	0	0	1	0	0	28	0	0	0	0	1	1
5	0	0	0	0	1	0	29	1	1	1	0	0	0
6	1	1	0	0	0	0	30	1	2	0	1	0	0
7	0	0	1	0	0	0	31	1	0	1	0	1	0
8	0	1	0	1	0	0	32	1	1	0	1	1	0
9	1	0	0	0	1	0	33	0	0	1	0	0	1
10	0	0	0	1	1	0	34	0	1	0	1	0	1
11	0	0	0	0	0	1	35	1	0	0	1	2	0
12	0	2	0	0	0	0	36	0	0	0	1	1	1
13	1	0	1	0	0	0	37	0	2	1	0	0	0
14	1	1	0	1	0	0	38	0	1	1	0	1	0
15	0	1	0	0	1	0	39	0	1	0	0	1	1
16	1	0	0	1	1	0	40	0	0	0	0	2	1
17	0	0	0	0	2	0	41	1	2	1	0	0	0
18	0	0	0	1	0	1	42	1	1	1	0	1	0
19	1	2	0	0	0	0	43	0	1	1	0	0	1
20	0	1	1	0	0	0	44	0	0	1	0	1	1
21	0	2	0	1	0	0	45	0	1	0	1	1	1
22	1	1	0	0	1	0	46	0	0	0	1	2	1
23	0	0	1	0	1	0							

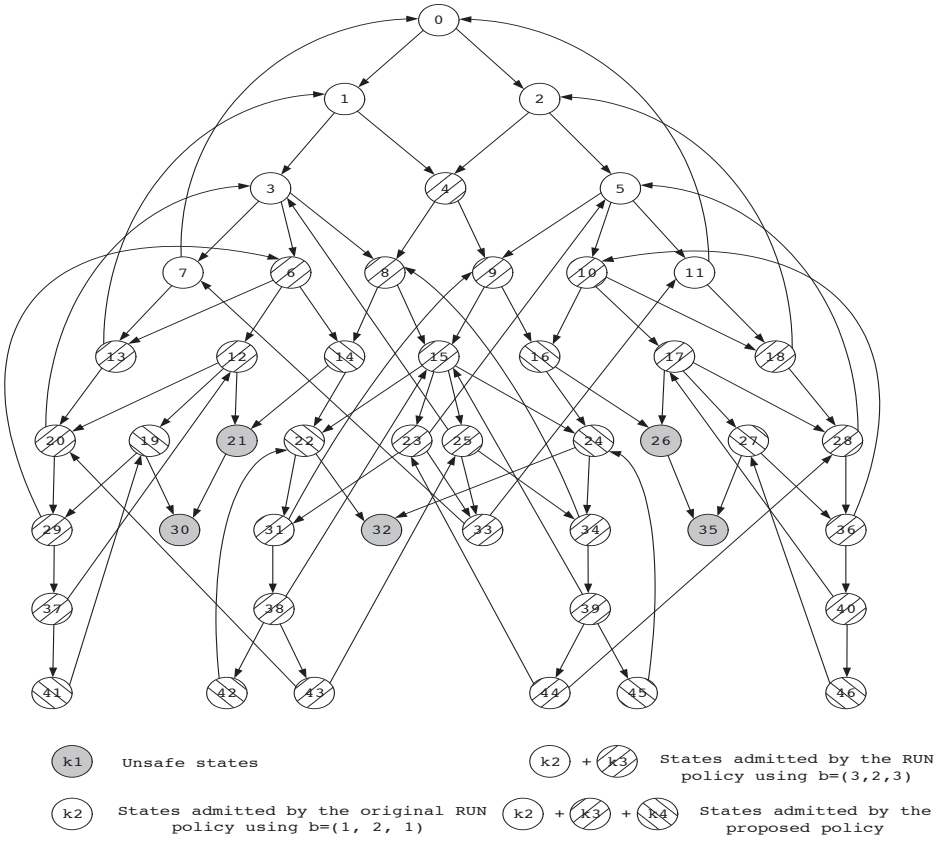

Figure 13: The reachability graph of the process-resource net of Figure 12, and a comparison of the sub-spaces admitted by the presented supervisors

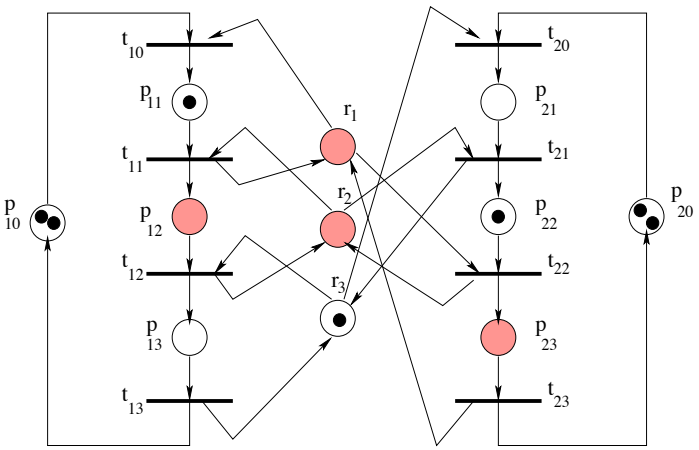

Figure 14: Interpreting the non-liveness of PT-ordinary process-resource nets with acyclic, quasi-live and strongly reversible process subnets through empty siphons

5 Analysis and design of algebraic DAPs through PN structural analysis

A structural characterization for the reversibility of process-resource nets and the correctness of algebraic DAPs The characterization of the RAS deadlock-freedom and the DAP correctness that was provided in the closing part of Section 2, and was behind the design methodology of Section 3, is *behavioral*, since it engages patterns and structure that are traceable in the reachability space of the considered process-resource net. In this section we focus on an alternative characterization for these two concepts that is *structural*, since it is based on the identification of special structure in the reachable markings of the underlying PNs. As it will be shown in the subsequent developments, the ability to confine the search for special structure in individual markings further enables the assessment of the deadlock-freedom of the underlying RAS and/or the correctness of any given algebraic DAP through an *implicit* enumeration of the underlying state space. At the basis of all the developments presented in this section is the following structural characterization for the reversibility of the process-resource nets defined in Section 2:

Theorem 2 *Let $\mathcal{N} = (P_S \cup I \cup O \cup P_R, T, W, M_0)$ be a process-resource net with acyclic, quasi-live and strongly reversible processes. Then, the following hold true:*

1. *$\mathcal{N}$ is reversible if and only if it is live.*
2. *$\mathcal{N}$ is live if and only if the space of* modified *reachable markings, $\overline{R(\mathcal{N}, M_0)}$, that is induced from $R(\mathcal{N}, M_0)$ through the projection:*

$$\overline{M}(p) = \begin{cases} M(p) & \text{if } p \notin I \cup O \\ 0 & \text{otherwise} \end{cases} \tag{41}$$

   contains no deadly marked siphon S such that (i) $S \cap P_R \neq \emptyset$ and (ii) $\forall p \in S \cap P_R$, p is a disabling place at $\overline{M}$.
3. *In the particular case that $\mathcal{N}$ is PT-ordinary, $\mathcal{N}$ is live if and only if the space of reachable markings, $R(\mathcal{N}, M_0)$, contains no empty siphons.*

A complete development of the results stated in Theorem 2 can be found in [15, 17]. Here we give an intuitive explanation for the role of deadly marked and empty siphons in the interpretation of the RAS deadlock, under the PN representation introduced in Section 2. Hence, Figure 14 depicts the empty siphon that interprets the deadlock of the robotic cell depicted in Figure 1. As indicated in Figure 14,

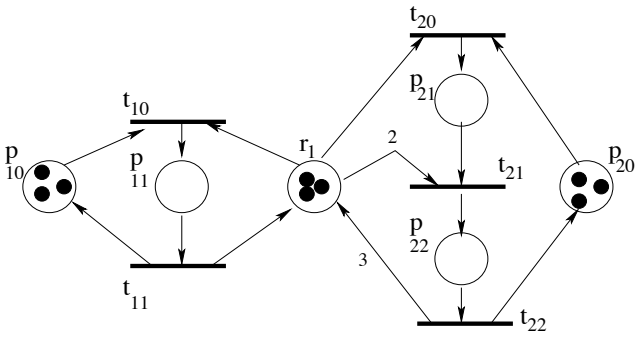

Figure 15: The considered process-resource net

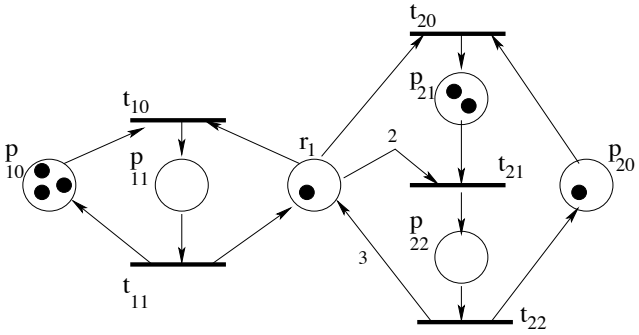

Figure 16: A net marking containing a RAS deadlock

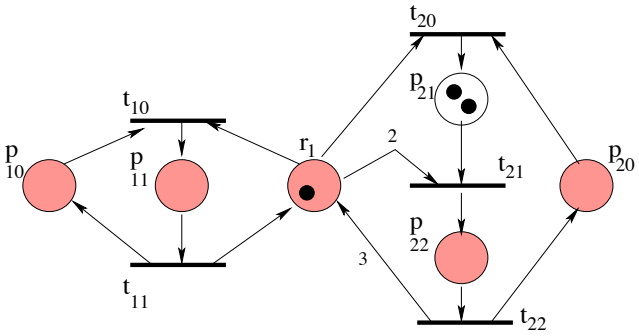

Figure 17: The resource-induced deadly marked siphon

this siphon consists of the resources involved in the considered deadlock and the stages that involve the allocation of at least one of the deadlocked resources and are currently empty; these stages are known as *"empty holders"* in the relevant terminology. Since a fundamental property of an empty siphon is that it will remain empty during the entire evolution of the net marking, the empty siphon of Figure 14 is a PN-based manifestation of the deadlock experienced in the underlying RAS. On the other hand, in the case of non-PT-ordinary PNs, the *non-uniformity* of the resource allocation requests for any single resource type, across the various processing stages, allows the existence of RAS states where certain process types can be executed repetitively, even though some of their supporting resource types are involved in a deadlock. The situation is depicted in Figures 15-17: In particular, Figure 16 depicts a reachable marking of the process-resource net depicted in Figure 15, where the active process instances corresponding to the tokens in place p_{21} are deadlocked, since their request for 2 extra units of resource R_1 cannot be met unless one of them releases its currently held resource. However, process instances executing stage p_{11} can still engage the remaining free unit of resource R_1 and successfully proceed to completion. But then, place p_{11} cannot be part of an empty siphon, even though it is an empty holder of resource R_1, that is involved in the depicted RAS deadlock. The aforementioned problem is remedied by the introduction of the concept of the *modified marking*. The modified marking of the original marking depicted in Figure 16 is depicted in Figure 17. By removing all the tokens resident in places $p_{j0},\ \forall j$, we construct a deadlock marking, in which the set S of all disabling places – depicted by shaded places in Figure 17 – will be a deadly marked siphon.[10] It is interesting to notice that the constructed siphon S is deadly marked but not empty. On the other hand, the token removal implied by the definition of the modified marking $\overline{M}$ will also generate artificially empty siphons, especially, for those process types with no active process instances in the original marking M; in Figure 17, the place set $S' = \{p_{10}, p_{11}\}$ is such an artificially constructed empty siphon. Hence, in order to infer the net non-liveness, one must focus on the particular type of deadly marked siphon characterized in item 2 of Theorem 2; these siphons are called *resource-induced* deadly marked siphons in the relevant literature.

The liveness and reversibility criteria of Theorem 2 can also provide correctness criteria for any tentative DAP for a given process-resource net, provided that the controlled net remains in the class of process-resource nets with acyclic, quasi-live and strongly reversible processes. In particular, as it was discussed in Section 3, the net $\mathcal{N}^c$ resulting from the imposition of some algebraic DAP on a given process-resource net $\mathcal{N}$ will possess this property as long as the super-imposed monitor places do not affect the quasi-liveness of the resource-augmented process subnets. While the assessment of the quasi-liveness of a resource-augmented process net can be a challenging problem in itself, there is broad set of cases for which this problem is easily resolved,[11] and furthermore, as it will be demonstrated in the example provided in the closing part of this section, in certain cases the imposed policy can be defined in a way that it will ensure the sought quasi-liveness.

The remaining part of this section establishes that the criterion stated in item 2 of Theorem 2 can be effectively assessed through a mathematical programming formulation.

Assessing the liveness and reversibility criterion of Theorem 2 through a mathematical programming formulation The starting point for the development of the sought formulation is the realization that, given a PN $\mathcal{N} = (P, T, W, M_0)$ and a marking $M \in R(\mathcal{N}, M_0)$, the maximal deadly marked siphon S in M can be computed by the algorithm of Figure 18, originally developed in [13]. In the case of *structurally bounded* nets, the algorithm of Figure 18 can be converted to a MIP formulation as follows: First, let $SB(p)$ denote a structural bound for the markings of place $p \in P$. Furthermore, let $v_p,\ z_t$ and f_{tp} be *binary indicator* variables respectively denoting the following conditions:

$$\begin{aligned} v_p = 1 &\iff \text{place } p \text{ is removed by the algorithm}, \quad \forall p \in P \\ z_t = 1 &\iff \text{transition } t \text{ is removed by the algorithm}, \quad \forall t \in T \\ f_{pt} = 1 &\iff M(p) \geq W(p,t) \ \vee \ v_p = 1, \quad \forall W(p,t) > 0 \end{aligned}$$

Then, we have the following theorem:

[10] c.f. Theorem 5 in the Appendix.
[11] c.f. the relevant discussion in Section 2

Input: A PN $\mathcal{N} = (P, T, W, M_0)$ and a marking $M \in R(\mathcal{N}, M_0)$
Output: The maximal deadly marked siphon in M, S

1. $S := P$; $\mathcal{N}' := \mathcal{N}$
2. **while** $\exists\ t \in T$ such that t is fireable in the modified net $\mathcal{N}'$ **do**
   (a) $S := S \backslash t^\bullet$
   (b) Remove t from $\mathcal{N}'$
   (c) Remove $t^\bullet$ from $\mathcal{N}'$

   endwhile
3. **Return** S

Figure 18: An algorithm for computing the maximal deadly marked siphon in a given marking M

Theorem 3 *[13, 17] Given a marking $M \in R(\mathcal{N}, M_0)$ of a structurally bounded PN $\mathcal{N} = (P, T, W, M_0)$, the maximal deadly marked siphon S contained in M is determined by:*

$$S = \{p \in P \mid v_p = 0\} \tag{42}$$

where $v_p,\ p \in P$, is obtained through the following IP formulation:

$$G(M) = \min \sum_{p \in P} v_p \tag{43}$$

s.t.

$$f_{pt} \geq \tfrac{M(p) - W(p,t) + 1}{SB(p)}, \quad \forall W(p,t) > 0 \tag{44}$$

$$f_{pt} \geq v_p, \quad \forall W(p,t) > 0 \tag{45}$$

$$z_t \geq \textstyle\sum_{p \in {}^\bullet t} f_{pt} - |{}^\bullet t| + 1, \quad \forall t \in T \tag{46}$$

$$v_p \geq z_t, \quad \forall W(t,p) > 0 \tag{47}$$

$$v_p, z_t, f_{pt} \in \{0, 1\},\ \forall p \in P,\ \forall t \in T \tag{48}$$

To understand the result of Theorem 3, first notice that Equation 46 together with Equation 44 imply that all transitions z_t fireable in marking M will have $z_t = 1$. Furthermore, Equation 47 implies that all places $p \in t^\bullet$ for some t with $z_t = 1$ will have $v_p = 1$, which implements Step (2.b) in the algorithm of Figure 18. Similarly, Equation 45 combined with Equation 46 force $z_t = 1$ for all transitions t with $v_p = 1,\ \forall p \in {}^\bullet t$. Finally, the fact that no additional place p (resp., transition t) has $v_p = 1$ (resp., $z_t = 1$), is guaranteed by the specification of the objective function in the above formulation.

In the case that the net $\mathcal{N}$ is a process-resource net, the formulation of Theorem 3 can be restricted to the computation of the maximal *resource-induced* deadly marked siphon, through the introduction of the following two constraints:

$$\textstyle\sum_{r \in P_R} v_r \leq |P_R| - 1 \tag{49}$$

$$\textstyle\sum_{t \in r^\bullet} f_{rt} - |r^\bullet| + 1 \leq v_r, \quad \forall r \in P_R \tag{50}$$

Constraint 49 enforces that the identified siphon S must contain at least one resource place, while Constraint 50 requires that all resource places included in S must be disabling. The resulting necessary and sufficient condition for the non-existence of resource-induced deadly marked siphons in a given marking M of a process-resource net is as follows:

Corollary 1 *[13, 17] A given marking M of a process-resource net $\mathcal{N}$ contains no resource-induced deadly marked siphons, if and only if the corresponding formulation of Equations 43–50 is infeasible.*

The test of Corollary 1 can be extended to a test for the non-existence of resource-induced deadly marked siphons over the entire modified reachability space, $\overline{R(\mathcal{N}, M_0)}$, of a process-resource net $\mathcal{N} = (P, T, W, M_0)$, by:

i. substituting the marking M in the MIP formulation of Theorem 3 with the modified marking $\overline{M}$;

ii. introducing an additional set of variables, M, representing the net reachable markings;

iii. adding two sets of constraints, the first one linking variables M and $\overline{M}$ according to the logic of Equation 41, and the second one ensuring that the set of feasible values for the variable vector M is equivalent to the PN reachability space $R(\mathcal{N}, M_0)$.

In the general case, the characterization of the set $R(\mathcal{N}, M_0)$ by a system of linear inequalities will involve a number of variables and constraints that is an exponential function of the net size $|\mathcal{N}|$ [23]. However, in the case of the process-resource nets considered in this work, there is such a characterization of $R(\mathcal{N}, M_0)$ that is polynomially sized with respect to $|\mathcal{N}|$, and therefore, the plan outlined above remains a viable proposition; we refer to [18] for the relevant details. Furthermore, a practical and frequently used implementation of the aforementioned plan substitutes the exact characterization of the reachability space $R(\mathcal{N}, M_0)$ by its superset that is provided by the *state equation*.[12] The resulting formulation provides a *sufficient* condition for the non-existence of resource-induced deadly marked siphons S in the entire space $\overline{R(\mathcal{N}, M_0)}$ of a given process-resource net $\mathcal{N}$, which in the light of Theorem 2, constitutes also a *sufficient* condition for the liveness and reversibility of process-resource nets with acyclic, quasi-live, and strongly reversible process subnets. The following corollary summarizes the above discussion:

Corollary 2 *Let $\mathcal{N} = (P, T, W, M_0)$ be a process-resource net with acyclic, quasi-live, and strongly reversible process subnets. If the mixed integer program defined by (i) Equations 43–50, where vector variable M is replaced by vector variable $\overline{M}$, (ii) Equations 72–73, and (iii) Equation 41, is infeasible, then $\mathcal{N}$ is live and reversible.*

Concluding this discussion, we notice that for the case of PT-ordinary process-resource nets with acyclic, quasi-live, and strongly reversible subnets, a similar but simpler liveness and reversibility sufficiency test can be obtained by focusing on the presence of empty siphons in the original net reachability space, $R(\mathcal{N}, M_0)$; we refer the reader to [5, 12] for a detailed discussion of this formulation.

Design of efficient algebraic DAPs through the criterion of Corollary 2 As it was discussed in the opening part of this section, the correctness of an algebraic DAP can be established by the criterion of Corollary 2 as long as the super-imposition of the relevant monitor places maintains the quasi-liveness of the underlying resource-augmented process sub-nets. Hence, the formulation of Corollary 2 can be embedded in a search process seeking a pair (A, b) that will render this formulation infeasible.[13] This search can be further assisted by additional insights regarding the dynamics of the underlying process-resource net. A particularly effective use of the aforementioned criterion has sought the systematic relaxation of the right-hand-side vector, b, in algebraic DAPs that have been developed through alternative approaches; we refer the reader to [12, 13] for some relevant examples. Next, we demonstrate the application of the criterion of Corollary 2 by focusing on a particular class of algebraic DAPs known as *"process-release"* policies. This class of policies seeks only to restrict the number of process instances that are loaded simultaneously into the system, rather than their access to particular segments of their process routes. Hence, they can be expressed by a single linear inequality

$$a \cdot M_S \leq b \tag{51}$$

where b defines the ceiling on the process concurrency imposed by the considered supervisor, and the elements of the row vector a are provided by the p-semiflows that characterize the flow logic of the

[12] This is Equation 72 in the Appendix.

[13] For a more concrete experience, the reader is invited to apply the criterion of Corollary 2 on the algebraic DAPs that were presented in Examples 2 and 3.

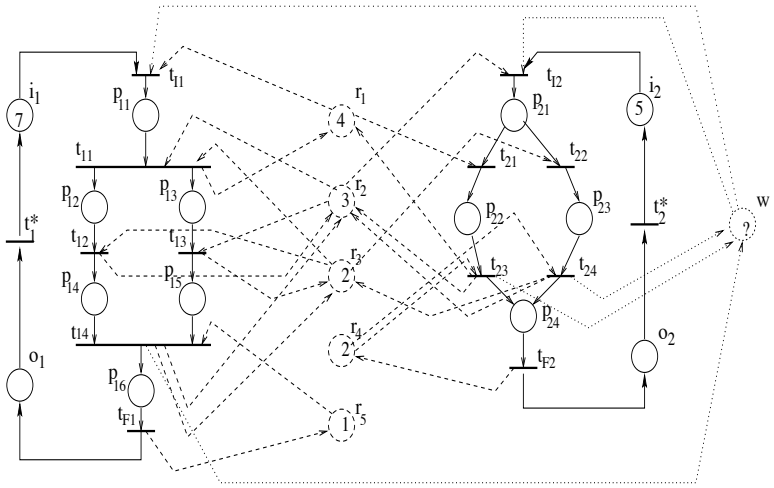

Figure 19: The process-resource net and the imposed process-release control policy for Example 4

various process types. The DAP of Equation 51 is superimposed to the original process-resource net $\mathcal{N}$ through the introduction of a single control place p_c with $p_c^\bullet \subseteq \bigcup_j \{t_{I_j}\}$. Hence, it should be obvious to the reader that the resulting controlled net $\mathcal{N}^c$ preserves the quasi-liveness of the original net $\mathcal{N}$, as long as $M_0(p_c) \geq 1$. The following example demonstrates the DAP synthesis method discussed in this section, and also, the particular concept of process-release control.

Example 4 Consider the process-resource net depicted in Figure 19. As it can be seen in the figure, the underlying RAS consists of two process types, Π_1 and Π_2, and five resource types, $R_1, \ldots, R_5$. Process type Π_1 has a flow represented by an acyclic marked graph, and it involves six processing stages, $\Xi_{11}, \ldots, \Xi_{16}$, with corresponding resource requirements: $(1,0,0,0,0)^T$, $(0,1,\ 0,0,0)^T$, $(0,0,1,0,0)^T$, $(0,0,1,0,0)^T$, $(0,1,0,0,0)^T$ and $(0,0,0,0,1)^T$. Process type Π_2 has a flow represented by an acyclic state machine, and it involves four stages, $\Xi_{21}, \ldots, \Xi_{24}$, with corresponding resource requirements: $(0,1,0,0,0)^T$, $(1,1,0,0,0)^T$, $(0,1,1,0,0)^T$ and $(0,0,0,1,0)^T$. A closer inspection of the stage resource requirements for these two processes reveals that the only resources that could be entangled in a deadlock are R_1, R_2 and R_3. Therefore, the critical sections for Π_1 and Π_2 are respectively defined by the stage sets $\{\Xi_{11}, \Xi_{12}, \Xi_{13}, \Xi_{14}, \Xi_{15}\}$ and $\{\Xi_{21}, \Xi_{22}, \Xi_{23}\}$.

Our intention is to develop an algebraic DAP that will establish the liveness and reversibility of the controlled net by restricting the number of process instances that can execute simultaneously in their critical sections identified above. Hence, the proposed supervisor constitutes a more refined implementation of the general *"process-release"* control scheme, to the particular process-resource net of Figure 19. This supervisor is super-imposed to the original process-resource net of Figure 19 by introducing the control place w, connected to the original process-resource net through the flow structure depicted by dotted lines in Figure 19.

Next we seek to determine the maximal initial marking for place w that leads to live and reversible behavior for the controlled net of Figure 19, using the siphon-based analysis that was developed in this section. For this, first we determine an upper bound to the maximal number of processes that can be executed simultaneously by the considered RAS. The reader can convince herself that, based on the resource capacities and the process flows annotated in Figure 19, an upper bound for the system concurrency with respect to process type Π_1 (resp., Π_2) is 7 (resp., 5) process instances. Then, application of the MIP formulation of Corollary 2 in a binary search over the integer set $\{1, \ldots, 12\}$ reveals that the maximal initial marking for control place w leading to a correct algebraic DAP – or equivalently, the maximal number of processes that can be simultaneously loaded and let to execute uncontrollably through the system without the possibility of running into any deadlocking problems – is 6. For completeness, we mention that the deadlock marking identified by the computerized solver when the MIP formulation of Corollary 2 was solved with $M_0(w) = 7$, is: $M(i_1) = 1$; $M(p_{11}) = 4$; $M(p_{12}) = M(p_{13}) = 2$; $M(i_2) = 4$;

$M(p_{21}) = 1$; $M(r_4) = 2$; $M(r_5) = 1$; and zero for every other place.

Closing the discussion of this example, we want to point out that, while in the case of "process-release" control policies the satisfaction of the criterion of Corollary 2 presents a monotonicity with respect to the initial marking of the control place, p_c, that enables the search for the maximally permissive implementation through binary search, this property will not be true for algebraic DAPs implementing more involved control schemes. In those cases, the identification of a maximal marking leading to live and reversible behavior will necessitate a more careful search mechanism. Furthermore, in the more general case, the *structural liveness* of the controlled net with respect to markings $M_0(p_c(k)),\ k = 1, \ldots, \dim(b)$– i.e., the existence of some marking $M_0(p_c(k)),\ k = 1, \ldots, \dim(b)$, that satisfies the the liveness and reversibility condition of Corollary 2 for the resulting net $\mathcal{N}^c$ – cannot be guaranteed *a priori*. The next section offers some further insights on these issues by providing an analytical characterization of the basic mechanism that enables the control of the net siphons through a limited number of monitor places.

6 Explaining the functionality of algebraic DAPs

Implicit siphon control This section offers an analytical interpretation of the basic mechanism that enables the algebraic DAPs to control the marking of the entire set of siphons of a process-resource net with only a limited number of control places. Our discussion epitomizes the key insights and results of [14], that constitutes a more formal and extensive reference for the subsequent developments. Furthermore, in order to enhance the clarity of the presentation, in the following we shall confine our attention to PT-ordinary process-resource nets. Then, thanks to the last item of Theorem 2, we are able to focus on the empty siphons of the underlying process-resource net instead of the more elusive set of resource-induced deadly marked siphons. In particular, we shall say that a net siphon is *controlled* if it remains non-empty during the entire evolution of the net marking.

The following series of definitions introduce a number of concepts that are instrumental for the linkage of the structure of the algebraic DAPs to the control of the net siphons.

Definition 3 *Consider a marked PN $\mathcal{N} = (P, T, W, M_0)$ and a vector $v \in \Re^{|P|}$, where $\Re$ denotes the set of reals. Then, for any marking $M \in R(\mathcal{N}, M_0)$, the* generalized compound marking *generated by v, is defined by*

$$GCM(M, v) = \sum_{p \in P} v(p)M(p) = v^T M \tag{52}$$

The vector v will be called the generator *of $GCM(M, v)$ and the set of places corresponding to non-zero elements of v will be denoted by P^v. Finally, in the particular case that $v(p) \in \{0, 1\},\ \forall p \in P$, a $GCM(M, v)$ reduces to the* compound marking *of the place subset P^v.*

Definition 4 *Consider a* pure *marked PN $\mathcal{N} = (P, T, W, M_0)$ and a GCM generator $v \in \Re^{|P|}$. Then, the* net flow (vector) *of v is defined by*

$$NF(v) = v^T \Theta \tag{53}$$

where Θ denotes the flow matrix of $\mathcal{N}$.

Notice that $NF(v)$ is a $|T|$-dimensional row vector. Furthermore, the components of $NF(v)$ have the following very intuitive interpretation: *For every transition $t \in T$, $NF(v; t)$ denotes the net change of $GCM(M, v)$ resulting by the firing of transition t at M.*[14] Finally, the next definition connects the GCM and NF concepts to the concept of siphon.

Definition 5 *Consider a siphon S of a* pure *marked PN $\mathcal{N} = (P, T, W, M_0)$. The* characteristic vector *of S is a $|P|$-dimensional binary vector λ_S such that*

$$\forall p \in P, \quad \lambda_S(p) = 1 \Longleftrightarrow p \in S \tag{54}$$

[14] This becomes obvious in the light of Equation 71 in the Appendix.

Hence, the characteristic vector, λ_S, of any given siphon S, can be considered as a GCM generator with $GCM(M, \lambda_S)$ being equal to the token content of siphon S at marking M. Furthermore, the components of the corresponding net flow vector $NF(\lambda_S)$ express the net change incurred to the siphon marking by the firing of any single transition $t \in T$.

The next theorem, which constitutes the main result of this section, establishes the connection between the siphon control and the concepts introduced in the above definitions:

Theorem 4 *[14] Let S denote a siphon of a* pure *marked PN $\mathcal{N} = (P, T, W, M_0)$ such that*

$$NF(\lambda_S) = \sum_{i=1}^{n} a_i NF(v^i) \tag{55}$$

where v^i, $i = 1, \ldots, n$, are GCM generators of $\mathcal{N}$, and $a_i \in \Re$, $\forall i$. Then,

$$S \text{ is controlled in } \mathcal{N} \Longleftrightarrow \lambda_S^T M_0 + G^* > 0 \tag{56}$$

where

$$G^* = \min_{M \in R(\mathcal{N}, M_0)} (M - M_0)^T \sum_{i=1}^{n} a_i v^i \tag{57}$$

To see the validity of this theorem, consider a marking $M \in R(\mathcal{N}, M_0)$. Then, there exists a vector $z \in (Z_0^+)^{|T|}$ such that $M = M_0 + \Theta z$ (c.f. Equations 72 and 73 in the Appendix). Therefore,

$$\begin{aligned}
M(S) &= \sum_{p \in S} M(p) \\
&= \lambda_S^T M \\
&= \lambda_S^T M_0 + \lambda_S^T \Theta z \\
&= \lambda_S^T M_0 + NF(\lambda_S) z \\
&= \lambda_S^T M_0 + [\sum_i a_i NF(v^i)] z \\
&= \lambda_S^T M_0 + [\sum_i a_i (v^i)^T \Theta] z \\
&= \lambda_S^T M_0 + [\sum_i a_i v^i]^T \Theta z \\
&= \lambda_S^T M_0 + [\sum_i a_i v^i]^T (M - M_0) \\
&= \lambda_S^T M_0 + (M - M_0)^T \sum_i a_i v^i
\end{aligned} \tag{58}$$

Clearly, the right-hand-side of Equation 58 is minimized over $R(\mathcal{N}, M_0)$ by G^*, and therefore, S will be controlled if and only if the criterion of Equation 56 holds.

A siphon S controlled by means of the criterion of Theorem 4 will be characterized as an *implicitly* controlled siphon. The corresponding generator vectors v^i, $i = 1, \ldots, n$, of Equation 55, will be called the *controlling generators* of S. In order to operationalize the criterion of Theorem 4, we must provide an analytic characterization of the constraint $M \in R(\mathcal{N}, M_0)$. This can be done effectively using the theory presented in [18]. Alternatively, one can compromise for a *sufficiency* test by relaxing the requirement $M \in R(\mathcal{N}, M_0)$ in Equation 57 to that expressed by the *state equations* 72 and 73, in the Appendix. We state the resulting criterion as a corollary.

Corollary 3 *Let S denote a siphon of a* pure *marked PN $\mathcal{N} = (P, T, W, M_0)$ such that*

$$NF(\lambda_S) = \sum_{i=1}^{n} a_i NF(v^i) \tag{59}$$

where v^i, $i = 1, \ldots, n$, *are GCM generators of* $\mathcal{N}$, *and* $a_i \in \Re$, $\forall i$. *Also, let*

$$G' = \min_{(M,z)} (M - M_0)^T \sum_{i=1}^{n} a_i v^i \tag{60}$$

s.t.

$$M = M_0 + \Theta z \tag{61}$$

$$M \geq 0, \quad z \in (Z_0^+)^{|T|} \tag{62}$$

Then,

$$\lambda_S^T M_0 + G' > 0 \Longrightarrow S \textit{ is} \text{ controlled } \textit{in } \mathcal{N} \tag{63}$$

Notice that the mathematical programming (MP) formulation involved in the criterion of Corollary 3 is a Mixed Integer Program (MIP), and therefore, it can be easily addressed through commercial solvers.[15] Next we present another criterion that is weaker than the criterion of Corollary 3, but it connects the presented results to those originally derived in [9]. Furthermore, this new criterion can be simpler, from a computational standpoint.

Corollary 4 *Let* S *denote a siphon of a* pure *marked PN* $\mathcal{N} = (P, T, W, M_0)$ *such that*

$$NF(\lambda_S) = \sum_{i=1}^{n} a_i NF(v^i) \tag{64}$$

where v^i, $i = 1, \ldots, n$, *are GCM generators of* $\mathcal{N}$, *and* $a_i \in \Re$, $\forall i$. *Also, for every* $i \in \{1, \ldots, n\}$, *let* $\underline{GCM}(v^i)$ *and* $\overline{GCM}(v^i)$ *respectively denote a lower and an upper bound of* $GCM(M, v^i)$, *for all* M *such that*

$$M = M_0 + \Theta z \tag{65}$$

$$M \geq 0, \quad z \in (Z_0^+)^{|T|} \tag{66}$$

Finally, let

$$\begin{aligned} G'' \quad = \quad & \sum_{i:a_i>0} a_i[\underline{GCM}(v^i) - GCM(M_0, v^i)] + \\ & \sum_{i:a_i<0} a_i[\overline{GCM}(v^i) - GCM(M_0, v^i)] \end{aligned} \tag{67}$$

Then,

$$\lambda_S^T M_0 + G'' > 0 \Longrightarrow S \textit{ is} \text{ controlled } \textit{in } \mathcal{N} \tag{68}$$

The validity of Corollary 4 can be easily established by noticing that

$$\begin{aligned} (M - M_0)^T \sum_i a_i v^i \quad &= \quad \sum_i a_i (M^T v^i - M_0^T v^i) \\ &= \quad \sum_i a_i [GCM(M, v^i) - \\ &\quad\quad GCM(M_0, v^i)] \end{aligned} \tag{69}$$

Then, the definitions of $\underline{GCM}(v^i)$ and $\overline{GCM}(v^i)$, when combined with Equations 60-62, 65-67 and 69, imply that $G' \geq G''$ and the validity of Corollary 4 follows from Corollary 3.

[15] In fact, the integrality requirement for z can be further relaxed to $z \geq 0$, providing a test that is computationally easier, but also with diminished resolution power, compared to the test of Corollary 3.

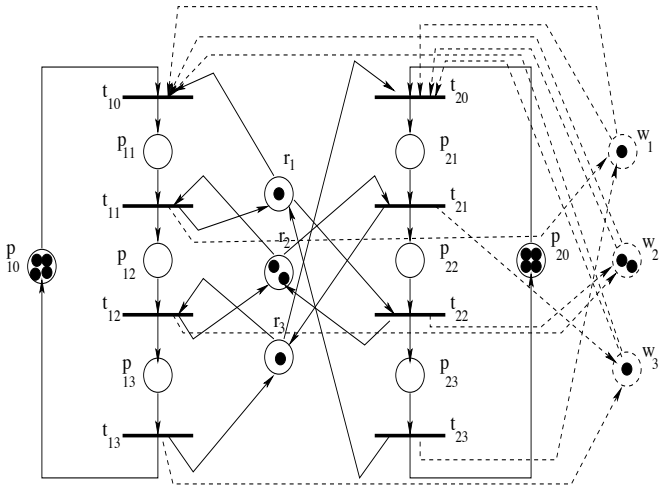

Figure 20: The nets $\mathcal{N}$ and $\mathcal{N}^c$ of Example 5

Beyond providing a sufficiency test for assessing whether a given siphon S is implicitly controlled by a set of GCM generator vectors $\{v^i : i = 1, \ldots, n\}$, the result of Corollary 4 can also provide the basis for deploying a control mechanism that will actively enforce the implicit control of siphon S by some generator set $\{v^i : i = 1, \ldots, n\}$. Under this approach, the upper and lower bounds $\overline{GCM}(v^i)$ and $\underline{GCM}(v^i)$, $i = 1, \ldots, n$, are *"design parameters"*, and their values are chosen such that they guarantee the condition of Equation 68. The selected bounds can be subsequently enforced on the behavior of the original net by the addition of appropriate *"monitor places"*, according to the theory developed in [10]. Finally, it is also known that:

Proposition 1 *[9] Given a pure marked PN $\mathcal{N} = (P, T, W, M_0)$, the* rank *of the space of net flow vectors $NF(\lambda_S)$, corresponding to the net siphons S, is bounded from above by* $\min\{|P|, |T|\}$.

Hence, the entire set of siphons, $\mathcal{S}$, of a pure marked PN $\mathcal{N} = (P, T, W, M_0)$, can be potentially controlled by a set of generators, and the resultant monitor places, that is *linearly* sized with respect to the net size $|\mathcal{N}|$. The following example demonstrates this capability and connects the above discussion to the context of process-resource nets and algebraic DAPs.

Example 5 Consider the net $\mathcal{N}$ depicted by solid lines in Figure 20, under the supervision of the algebraic DAP expressed by the following constraints:

$$\begin{bmatrix} 1 & & & 1 & 1 & 1 \\ 1 & 1 & & 1 & 1 & \\ 1 & 1 & 1 & 1 & & \end{bmatrix} \begin{bmatrix} M(p_{11}) \\ M(p_{12}) \\ M(p_{13}) \\ M(p_{21}) \\ M(p_{22}) \\ M(p_{23}) \end{bmatrix} \leq \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \tag{70}$$

The control sub-net enforcing the constraints of Equation 70 on $\mathcal{N}$ is also depicted in Figure 20, through dashed lines. The resulting controlled net, $\mathcal{N}^c$, has been shown to be live and reversible in [12]. Here we re-establish the liveness of net $\mathcal{N}^c$, and the correctness of the DAP expressed by Equation 70, by applying the siphon control criterion of Corollary 4.

The characteristic vectors of the minimal siphons in the controlled net $\mathcal{N}^c$ of Figure 20 are tabulated in Table 2. Siphons S_1–S_8 correspond to the support of p-semiflows, and therefore, they are already controlled. The net flows $NF(\lambda_{S_k})$ of the remaining uncontrolled siphons S_k, $k = 9, 10, 11$, can be expressed as linear combinations of the net flows $NF(v^l)$ corresponding to the GCM generator vectors v^l, $l = 1, \ldots, 6$, presented in Table 3; Table 4 provides the relevant coefficients a^l_k, $l = 1, \ldots, 6$, $k =$

Table 2: Example 5: The minimal siphons of the controlled net $\mathcal{N}^c$

siphon	p_{10}	p_{11}	p_{12}	p_{13}	p_{20}	p_{21}	p_{22}	p_{23}	r_1	r_2	r_3	w_1	w_2	w_3
S_1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
S_2	0	0	0	0	1	1	1	1	0	0	0	0	0	0
S_3	0	1	0	0	0	0	0	1	1	0	0	0	0	0
S_4	0	0	1	0	0	0	1	0	0	1	0	0	0	0
S_5	0	0	0	1	0	1	0	0	0	0	1	0	0	0
S_6	0	1	0	0	0	1	1	1	0	0	0	1	0	0
S_7	0	1	1	0	0	1	1	0	0	0	0	0	1	0
S_8	0	1	1	1	0	1	0	0	0	0	0	0	0	1
S_9	0	0	1	0	0	0	0	1	1	1	0	0	0	0
S_{10}	0	0	0	1	0	0	1	0	0	1	1	0	0	0
S_{11}	0	0	0	1	0	0	0	1	1	1	1	0	0	0

Table 3: Example 5: The GCM generators, v^l, employed for the expansion of the net flow vectors $NF(\lambda_{S_k}),\ k = 9, 10, 11$, and the associated bounds used in the evaluation of the criterion of Corollary 4

gen.	p_{10}	p_{11}	p_{12}	p_{13}	p_{20}	p_{21}	p_{22}	p_{23}	r_1	r_2	r_3	w_1	w_2	w_3	$\underline{GCM}$	$\overline{GCM}$
v^1	0	1	0	0	0	1	1	1	0	0	0	0	0	0	0	1
v^2	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	2
v^3	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	1
v^4	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
v^5	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1
v^6	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1

Table 4: Example 5: The coordinates for the expansions of $NF(\lambda_{S_k}),\ k = 9, 10, 11$, as linear combinations of $NF(v^l),\ l = 1, \ldots, 6$, and the obtained values for the test of Corollary 4

siphon	a^1	a^2	a^3	a^4	a^5	a^6	$\lambda_S^T M_0 + G''(S)$
S_9	0.0	-1.0	0.0	0.0	1.0	1.0	3-2=1
S_{10}	0.0	0.0	0.0	0.0	-1.0	-1.0	3-2=1
S_{11}	0.0	-1.0	0.0	0.0	0.0	0.0	4-2=2

$9, 10, 11$, for these expansions. Notice that the vector set $\{v^l,\ l = 1, \ldots, 6\}$ contains the GCM generator set $\{v^i,\ i = 1, 2, 3\}$, that is induced by the DAP constraints of Equation 70, and an additional vector set $\{v^j,\ j = 4, 5, 6\}$, selected in a way that facilitates the aforementioned expansion of the vector set $\{NF(\lambda_{S_k})\},\ k = 9, 10, 11$.

Table 3 also provides the bounds $\underline{GCM}(v^l)$ and $\overline{GCM}(v^l)$ used in the evaluation of G'', during the application of the criterion of Corollary 4 to the siphons $S_k,\ k = 9, 10, 11$. The values of $\underline{GCM}(v^l)$ are obtained immediately by noticing that (i) $M^T \cdot v^l \geq 0,\ \forall l$, (ii) $v^l(p) > 0 \Longrightarrow p \in P_S,\ \forall p \in P$, and (iii) $M_0(p) = 0,\ \forall p \in P_S$. The values of $\overline{GCM}(v^l)$ were obtained by solving the following MIP for each $l \in \{1, \ldots, 6\}$:

$$\overline{GCM}(v^l) = \max_{(M,z)} M^T \cdot v^l$$

s.t.

Equations 65 and 66

Finally, Table 4 provides also the values obtained for the left-hand-side of the inequality that is employed by the test of Corollary 4 (c.f. Equation 68), based on the aforementioned expansions and bounds. Since all the obtained values are strictly greater than zero, it is concluded that the net $\mathcal{N}^c$ is live, and the DAP of Equation 70 is a correct DAP for the original net $\mathcal{N}$.

The work of [12] has also established that the DAP obtained by replacing the right-hand-side of Equation 70 with the vector $[2\ 4\ 2]^T$, is another correct DAP for net $\mathcal{N}$. Interestingly, the application of the test of Corollary 4, based on the GCM generator set $\{v^l\}$ of Table 3, fails to recognize the ability of this new DAP to control the siphons S_9 and S_{10} of Table 2. On the other hand, this effect is successfully recognized by the more powerful test of Corollary 3. We leave the relevant computational details to the reader.

7 Conclusions

This chapter started with the observation that the flexible automation pursued in the context of many contemporary technological application necessitates the explicit logical analysis and control of these environments with respect to the underlying resource allocation, and subsequently it offered a unified and comprehensive treatment of the theory of algebraic deadlock avoidance policies, that provides an effective and efficient solution to the emerging logical control problems. The presented developments characterized the state of art in the relevant research area, and, hopefully, they have also revealed its maturity and vigor. At the same time, these results can function as the starting point for additional developments in the field in terms of, both, theory and application.

On the theoretical side, a novel research direction was recently developed by [21], which introduced the notion of *generalized* algebraic DAP, based on the notion of *"committee machine"* that was borrowed from pattern recognition and machine learning. The key property of generalized algebraic DAPs is that, when viewed as pattern classifiers in the underlying state space, they can recognize *non-convex* state subsets, something that is not possible with the linear structure of algebraic DAPs. The work of [21] extends the design methodology presented in Section 4 to this new class of policies, but currently, we lack a complete understanding and characterization of the properties of these policies in the spirit of Sections 5 and 6. An even more prominent open problem on the theoretical side is the development of the necessary theory for the effective and systematic integration of logical and performance-oriented control. Some preliminary thoughts and results along these lines are reported in [17, 4].

From an application standpoint, the ultimate objective of the research program underlying the results presented in this work, is the integration of the developed theory to a control architecture that will function as the next-generation *"operating system"*, able to support robust, yet highly flexible and efficient operation of the target technological applications. While this effort can be initiated and led by the relevant research community, a profound understanding of, and extensive interaction with, the target industries is of paramount importance for the successful implementation and the eventual acceptance of the final product.

Appendix: Petri nets – Basic Concepts and Definitions

A formal definition of the Petri net model is as follows:

Definition 6 *[11] A* Petri net (PN) *is defined by a quadruple* $\mathcal{N} = (P, T, W, M_0)$*, where*

- P *is the set of* places*,*
- T *is the set of* transitions*,*
- $W : (P \times T) \cup (T \times P) \rightarrow Z_0^+$ *is the* flow relation*, and*
- $M_0 : P \rightarrow Z_0^+$ *is the net* initial marking*, assigning to each place* $p \in P$*,* $M_0(p)$ tokens*.*

The first three items in Definition 6 essentially define a *weighted bipartite digraph* representing the system *structure* that governs its underlying dynamics. The last item defines the system *initial state*. A conventional graphical representation of the net structure and its marking depicts nodes corresponding to places by empty circles, nodes corresponding to transitions by bars, and the tokens located at the various places by small filled circles. The flow relation W is depicted by directed edges that link every nodal pair for which the corresponding W-value is non-zero. These edges point from the first node of the corresponding pair to the second, and they are also labeled – or, *"weighed"* – by the corresponding W-value. By convention, absence of a label for any edge implies that the corresponding W-value is equal to unity.

PN structure-related concepts and properties Given a transition $t \in T$, the set of places p for which $(p, t) > 0$ (resp., $(t, p) > 0$) is known as the set of *input* (resp., *output*) places of t. Similarly, given a place $p \in P$, the set of transitions t for which $(t, p) > 0$ (resp., $(p, t) > 0$) is known as the set of *input* (resp., *output*) transitions of p. It is customary in the PN literature to denote the set of input (resp., output) transitions of a place p by ${}^\bullet p$ (resp., $p^\bullet$). Similarly, the set of input (resp., output) places of a transition t is denoted by ${}^\bullet t$ (resp., $t^\bullet$). This notation is also generalized to any set of places or transitions, X, e.g. ${}^\bullet X = \bigcup_{x \in X} {}^\bullet x$.

The ordered set $X =< x_1 \dots x_n > \in (P \cup T)^*$ is a *path*, if and only if (*iff*) $x_{i+1} \in x_i^\bullet, i = 1, \dots, n-1$. Furthermore, a path X is characterized as a *circuit iff* $x_1 \equiv x_n$.

A PN with a flow relation W mapping onto $\{0, 1\}$ is said to be *ordinary*. If only the restriction of W to $(P \times T)$ maps on $\{0, 1\}$, the PN is said to be *PT-ordinary*. An ordinary PN such that (s.t.) $\forall t \in T$, $|t^\bullet| = |{}^\bullet t| = 1$, is characterized as a *state machine*, while an ordinary PN s.t. $\forall p \in P$, $|p^\bullet| = |{}^\bullet p| = 1$, is characterized as a *marked graph*.

A PN is said to be *pure* if $\forall (x, y) \in (P \times T) \cup (T \times P)$, $W(x, y) > 0 \Rightarrow W(y, x) = 0$. The flow relation of pure PN's can be represented by the *flow matrix* $\Theta = \Theta^+ - \Theta^-$ where $\Theta^+(p, t) = W(t, p)$ and $\Theta^-(p, t) = W(p, t)$.

PN dynamics-related concepts and properties In the PN modeling framework, the system state is represented by the net *marking* M, i.e., a function from P to Z_0^+ that assigns a *token* content to the various net places. The net marking M is initialized to marking M_0, introduced in Definition 6, and it subsequently evolves through a set of rules summarized in the concept of *transition firing*. A concise characterization of this concept has as follows: Given a marking M, a transition t is *enabled iff* for every place $p \in {}^\bullet t$, $M(p) \geq W(p, t)$, and this is denoted by $M[t\rangle$. $t \in T$ is said to be *disabled* by a place $p \in {}^\bullet t$ at M *iff* $M(p) < W(p, t)$. Furthermore, a place $p \in P$ for which there exists $t \in p^\bullet$ s.t. $M(p) < W(p, t)$ is said to be a *disabling* place at M. Given a marking M, a transition t can be *fired* only if it is enabled in M, and firing such an enabled transition t results in a new marking M', which is obtained from M by removing $W(p, t)$ tokens from each place $p \in {}^\bullet t$, and placing $W(t, p')$ tokens in each place $p' \in t^\bullet$. For pure PN's, the marking evolution incurred by the firing of a transition t can be concisely expressed by the *state equation*:

$$M' = M + \Theta \cdot \mathbf{1}_t \tag{71}$$

where $\mathbf{1}_t$ denotes the unit vector of dimensionality $|T|$ and with the unit element located at the component corresponding to transition t.

The set of markings reachable from the initial marking M_0 through any *fireable* sequence of transitions is denoted by $R(\mathcal{N}, M_0)$ and it is referred to as the net *reachability space*. In the case of pure PN's, a necessary condition for $M \in R(\mathcal{N}, M_0)$ is that the following system of equations is feasible in z:

$$M = M_0 + \Theta z \tag{72}$$

$$M \geq 0,\ z \in Z_0^+ \tag{73}$$

A PN $\mathcal{N} = (P, T, W, M_0)$ is said to be *bounded iff* all markings $M \in R(\mathcal{N}, M_0)$ are bounded. $\mathcal{N}$ is said to be *structurally bounded iff* it is bounded for any initial marking M_0. $\mathcal{N}$ is said to be *reversible iff* $M_0 \in R(\mathcal{N}, M)$, for all $M \in R(\mathcal{N}, M_0)$. A transition $t \in T$ is said to be *live iff* for all $M \in R(\mathcal{N}, M_0)$, there exists $M' \in R(\mathcal{N}, M)$ s.t. $M'[t\rangle$; non-live transitions are said to be *dead* at those markings $M \in R(\mathcal{N}, M_0)$ for which there is no $M' \in R(\mathcal{N}, M)$ s.t. $M'[t\rangle$. PN $\mathcal{N}$ is *quasi-live iff* for all $t \in T$, there exists $M \in R(\mathcal{N}, M_0)$ s.t. $M[t\rangle$; it is *weakly live iff* for all $M \in R(\mathcal{N}, M_0)$, there exists $t \in T$ s.t. $M[t\rangle$; and it is *live iff* for all $t \in T$, t is live. A marking $M \in R(\mathcal{N}, M_0)$ is a (total) *deadlock iff* every $t \in T$ is dead at M.[16]

Siphons and their role in the interpretation of the PN deadlock Of particular interest for the liveness analysis of the PN's considered in this chapter is a structural element known as *siphon*, i.e., a set of places $S \subseteq P$ such that ${}^\bullet S \subseteq S^\bullet$. A siphon S is *minimal iff* there exists no other siphon S' s.t. $S' \subset S$. A siphon S is said to be *empty* at marking M *iff* $M(S) \equiv \sum_{p \in S} M(p) = 0$. S is said to be *deadly marked* at marking M, *iff* every transition $t \in {}^\bullet S$ is disabled by some place $p \in S$. Clearly, empty siphons are deadly marked siphons. It is easy to see that, if S is a deadly marked siphon at some marking M, then (i) $\forall t \in {}^\bullet S$, t is a dead transition in M, and (ii) $\forall M' \in R(\mathcal{N}, M)$, S is deadly marked. Furthermore, the next theorem establishes a strong relationship between the notion of deadly marked siphon and that of the PN deadlock:

Theorem 5 *[17] Given a deadlock marking M of a PN $\mathcal{N} = (P, T, W, M_0)$, the set of disabling places $S \subseteq P$ in M constitutes a deadly marked siphon.*

PN semiflows PN semiflows provide an analytical characterization of various concepts of *invariance* underlying the net dynamics. Generally, there are two types, p and t-semiflows, with a *p-semiflow* formally defined as a $|P|$-dimensional vector y satisfying $y^T\Theta = 0$ and $y \geq 0$, and a *t-semiflow* formally defined as a $|T|$-dimensional vector x satisfying $\Theta x = 0$ and $x \geq 0$. In the light of Equation 72, the invariance property expressed by a p-semiflow y is that $y^T M = y^T M_0$, for all $M \in R(\mathcal{N}, M_0)$. Similarly, Equation 72 implies that for any t-semiflow x, $M = M_0 + \Theta x = M_0$.

Given a p-semiflow y (resp., t-semiflow x) its *support* is defined as $\|y\| = \{p \in P \mid y(p) > 0\}$ (resp., $\|x\| = \{t \in T \mid x(t) > 0\}$). A p-semiflow y (resp., t-semiflow x) is said to be *minimal iff* there is no p-semiflow y' (resp., t-semiflow x') s.t. $\|y'\| \subset \|y\|$ (resp., $\|x'\| \subset \|x\|$).

PN merging We conclude this introductory discussion on the PN concepts and properties by defining a merging operation for two PN's: Given two PN's $\mathcal{N}_1 = (P_1, T_1, W_1, M_{01})$ and $\mathcal{N}_2 = (P_2, T_2, W_2, M_{02})$ with $T_1 \cap T_2 = \emptyset$ and $P_1 \cap P_2 = Q \neq \emptyset$ s.t. for all $p \in Q$, $M_{01}(p) = M_{02}(p)$, the PN $\mathcal{N}$ resulting from the *merging* of the nets $\mathcal{N}_1$ and $\mathcal{N}_2$ *through the place set* Q, is defined by $\mathcal{N} = (P_1 \cup P_2, T_1 \cup T_2, W_1 \cup W_2, M_0)$ with $M_0(p) = M_{01}(p),\ \forall p \in P_1 \backslash P_2;\ M_0(p) = M_{02}(p),\ \forall p \in P_2 \backslash P_1;\ M_0(p) = M_{01}(p) = M_{02}(p),\ \forall p \in P_1 \cap P_2$.

References

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

[16] Notice that the concept of *deadlock* in the PN framework is different from the usage of this term in the RAS context. Some further elaboration on this issue is provided in Section 2.

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

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

[4] J. Y. Choi and S. A. Reveliotis. Relative value function approximation for the capacitated re-entrant line scheduling problem. *IEEE Trans. on Automation Science and Engineering*, 2:285–299, 2005.

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

[6] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Co., New York, NY, 1979.

[7] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, Reading, MA, 1979.

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

[9] 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*, 34:38–51, 2004.

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

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

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

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

[14] 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*, (to appear).

[15] S. A. Reveliotis. On the siphon-based characterization of liveness in sequential resource allocation systems. In *Applications and Theory of Perti Nets 2003*, pages 241–255, 2003.

[16] S. A. Reveliotis. Structural analysis of assembly/disassembly resource allocation systems. In *Proc. of ICRA 2003*. IEEE, 2003.

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

[18] 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 and Engineering*, 3:462–468, 2006.

[19] 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*, pages 322–341, 2006.

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

[21] S. A. Reveliotis, E. Roszkowska, and J. Y. Choi. Generalized algebraic deadlock avoidance policies for sequential resource allocation systems. In *2007 IEEE Intl. Conf. on Robotics & Automation*, pages –. IEEE, 2007.

[22] E. Roszkowska and R Wojcik. Problems of process flow feasibility in fas. In *CIM in Process and Manufacturing Industry*, pages 115–120. Pergamon Press, 1993.

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

[24] W. Van der Aalst. Structural characterizations of sound workflow nets. Technical Report Computing Science Reports 96/23, Eindhoven University of Technology, 1996.

[25] W. Van der Aalst. Verification of workflow nets. In P. Azema and G. Balbo, editors, *Lecture Notes in Computer Science, Vol. 1248*, pages 407–426. Springer Verlag, 1997.

[26] W. Van der Aalst and K. Van Hee. *Workflow Management: Models, Methods and Systems*. The MIT Press, Cambridge, MA, 2002.

[27] K. Van Hee, N. Sidorova, and M. Voorhoeve. Soundness and separability of workflow nets in the stepwise refinement approach. In W. Van der Aalst and E. Best, editors, *Lecture Notes in Computer Science, Vol. 2679*, pages 337–356. Springer Verlag, 2003.

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