2006 IEEE Conference on
Systems, Man, and Cybernetics
October 8-11, 2006, Taipei, Taiwan

Lender processes competing for shared resources: Beyond the S^{4}PR paradigm

Juan-Pablo López-Grao and José-Manuel Colom

***Abstract*— Formal models, as Petri nets, applied to the resource allocation problem have been a fruitful approach in the last years from a double perspective. Firstly, the consolidation of an abstraction process of systems leading to models structured around the concepts of processes and resources, which can be easily translated into Petri nets. Secondly, the obtention of analysis results characterizing deadlock states, as well as methods to amend the problem. Thanks to abstraction, this methods can be applied to many different application domains, although manufacturing is yet predominant. In this paper we follow the same philosophy, but extending the kind of systems that can be tackled. These extensions allow to consider nested iterations within the processes, and to hold resources in the initial state. We will show that these extensions are very relevant, from the real-world system point of view, in order to extend these techniques to a broader scope of scenarios. Nevertheless, the behaviours of the resulting models are much more complex than those of the previous restricted models, e.g., non-directedness.**

I. INTRODUCTION

Loosely speaking, a Resource Allocation System (RAS) is a discrete event system in which a finite set of resources is shared among a set of concurrent processes.

This general definition fits within a broad family of systems which range in disciplines such as manufacturing, distributed and parallel computing, game theory, networking or logistics. All of them fall in the same framework through an abstraction process that allows to construct system models bipartitioned into processes and resources.

These models are used to study the *resource allocation problem*: the procedure in serving the processes requirements for resources, according to their own resource usage policy, while accomplishing a certain goal. In quantitative terms, the concept relates to the ability of optimizing a system performance function [1]. In qualitative terms, it is widely associated to dealing with the set of partial or total *deadlocks* that may be reachable [2]: the focus of this paper.

We consider a set of processes in a deadlock state if they are indefinitely waiting for a set of resources that are already held by processes of the same set. In the context of RAS, Coffman [3] established four necessary conditions for the existence of a deadlock in general systems of processes competing for shared resources. However, these conditions are not sufficient in general, and so leave place for a wide scope of constrained subclasses of systems to be investigated in order to obtain a characterization of deadlock states. In many cases, this research is addressed by real-world systems, imposing constraints with physical meaning, and the study tries to capture the problems in terms of the discipline.

From this standpoint, it seems well-justified the success of the methodological approach on manufacturing systems, where some particular conditions (sequential processes, resources used in a conservative way, etc.) are likely to appear.

There exists abundant literature on the application of formal models to the study of the resource allocation problem. Formal models allow to focus on it avoiding unnecessary details thanks to the process of abstraction, and apply rigorous techniques to detect deadlocks and correct the system according to its expected behaviour. In particular, Petri nets have proved very useful for the modelling, analysis and synthesis of RAS, with an accent on manufacturing.

Due to the generous range of subclasses of RAS worth considering, though, different Petri net subclasses have emerged to deal with them. In general, we can classify restrictions on the model within two categories: (a) on the processes structure, and (b) on the way the processes use the resources. This paper will be devoted to the definition and study of the (P/T net) class of Systems of Processes Quarrelling over Resources (SPQR), which generalizes previous results both in (a) and (b). The authors will assume that the reader has some basic knowledge in Petri nets.

The paper is organized as follows. In section II, we will contextualize the work and illustrate its motivation through an example. In section III, the SPQR class will be introduced and related with simpler subclasses. In section IV, some structural and behavioural properties of the class will be outlined, which will be compared with those of its subclasses. In section V, we will point at some interesting liveness analysis results for the SPQR class. Finally, in section VI, we will summarize the results of the work.

II. PROBLEM DOMAIN AND MOTIVATION

A. Problem domain

As stated in section I, restrictions on Petri net models for RAS can be classified within two categories: (a) on the processes structure, and (b) on the way the processes use the resources. As far as (b) is respected, resources can be serially reusable (i.e., they are used in a conservative way by every process), or consumable (i.e., once they are consumed, they are not replaced). This work focuses on systems in which resources are serially reusable.

Regarding the processes structure, most of the current works focus on Sequential RAS (S-RAS), as opposed to Non-Sequential RAS (NS-RAS), in which assem-

J.P. López-Grao is with the Dept. of Computer Science and Systems Engineering (DIIS), Universidad de Zaragoza, Spain jpablo@unizar.es
J.M. Colom is with the Aragonese Engineering Research Institute (I3A), Universidad de Zaragoza, Spain jm@unizar.es

1-4244-0100-3/06/$20.00 ©2006 IEEE

bly/disassembly operations (fork/join operations) are allowed within the processes. Some works ([4], [5]), however, have attempted to approach NS-RAS from the Petri nets perspective, despite that finding effective solutions for them is, in general, much more complicated.

In the field of S-RAS (the scope of this paper), different Petri net models have successively emerged, frequently extending previous results and hence widening the subclass of systems that can be modelled and studied.

To the best of our knowledge, one of the first classes aimed to deal with the resource allocation problem in S-RAS is the class of Cooperating Sequential Systems (CSS) [2]. In CSS, concurrent processes share both the routing pattern and the way the resources are used in the routes (i.e., the process type is unique). These processes may compete for several resource types, allowing multiple instances of each type.

In more recent works, different process types with multiple concurrent instances are allowed, sometimes allowing alternative paths per process. In [6], the path (i.e., route) a process will follow is selected at the beginning of the process execution. Other works consider on-line routing decisions; in particular, this is dealt with in [7], where the seminal S^3PR class is introduced. However, processes in a S^3PR can use at most a single resource unit at a given state. A subclass of S^3PR, called L-S^3PR, was presented in [8], which featured some useful properties.

The mentioned restriction over resources usage is eliminated by the (more general) S^4PR class [9] (S^3PGR^2 in [10]). This allows processes to decide routings on-line and simultaneously reserve several resources belonging to distinct types. This kind of systems are named "Disjunctive-Conjunctive (OR/AND) RAS" ([11]). Nowadays, the most general class of the S^nPR family is the S^*PR class [12], in which processes are ordinary state machines with internal cycles. The authors would like to emphasize that many interesting works from different authors present and study other classes in the same vein. For a more detailed revision of these, we refer the reader to [13].

For all of the mentioned classes, except for S^*PR, siphon-based liveness characterizations are known. Due to their structural nature, they open a door to an efficient detection and correction of deadlocks, implementing controllers (usually by the addition of places) that restrain the behaviour of the net and avoid the bad markings to be reached.

In this work, it is our goal to cover broader S-RAS classes than those seized by the net classes previously introduced. The generalization is useful in the context of manufacturing but especially in other scenarios where the following elements are more frequent: (1) Internal iterations (e.g., recirculating circuits in manufacturing, nested loops in software); (2) Initial states in which there are resources that are already allocated.

In the following, we will make clear that the approach followed to date does not work with these systems. In this sense, there does not exists an analogous non-liveness characterization, and their inherent properties are much more complex. In particular, we would like to stress the fact that siphons do not longer work, in general, with the class being introduced.

B. Motivation. An example

Before introducing the new class, let us firstly illustrate how it captures more complex systems precluded by its predecessors. Suppose we own an on-line web search engine on the Internet, called Foolgle. This engine reads, processes and serves the information stored in a large centralized relational database, serving concurrently a number of users. The database stores information about thousands of sites on the Internet.

Additionally, we have a cluster with three nodes, each running a web crawler (also, *web spider* or *robot*). Each one of these robots is aimed to constantly surf the web, visiting and parsing every site, and subsequently update our centralized database, keeping it up-to-date.

These web crawlers are executed concurrently and have a cyclic behaviour through five phases: the initial, idle state, where maintenance tasks are executed (stage 0), heartbeat checking (stage 1), page retrieval (stage 2), HTML parsing with URL extraction (stage 3) and database update (stage 4). The metainformation in the database is indexed and dated, so that the job batches are well-ordered and no redundant work is taken.

As a system design principle, we want our set of concurrent robots to surf the web relentless and fairly. To ensure, it will be enforced that, once the three nodes are up and running, at most one of the robots will be in its idle state (stage 0).

Besides, only one robot can write in the database at the same time (i.e., at most one robot can be in stage 4). To synchronize the robots a Distributed Lock Manager (DLM) will be deployed. The system architecture is conceptualized in figure 1.

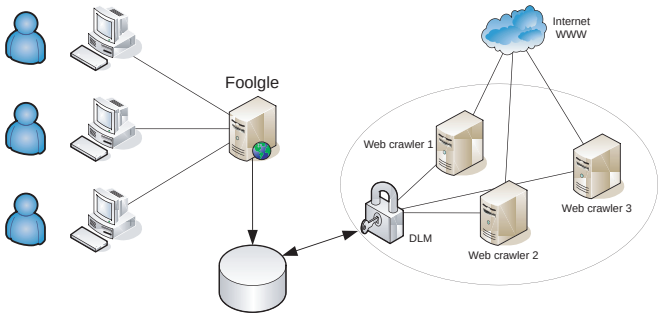

Fig. 1. The Foolgle system architecture

Hence we have three concurrent processes synchronized by a set of locks in the DLM, which can be abstracted as a set of virtual resources. Obviously, we would like to design the system so that we can make sure it is live. But, unfortunately, the methodological framework yielded by the S^4PR class proves insufficient.

It is not difficult to infer a reason why the S^4PR class is too limited to accomplish our goal. Every live S^4PR net is reversible [14], while our system should be live but non

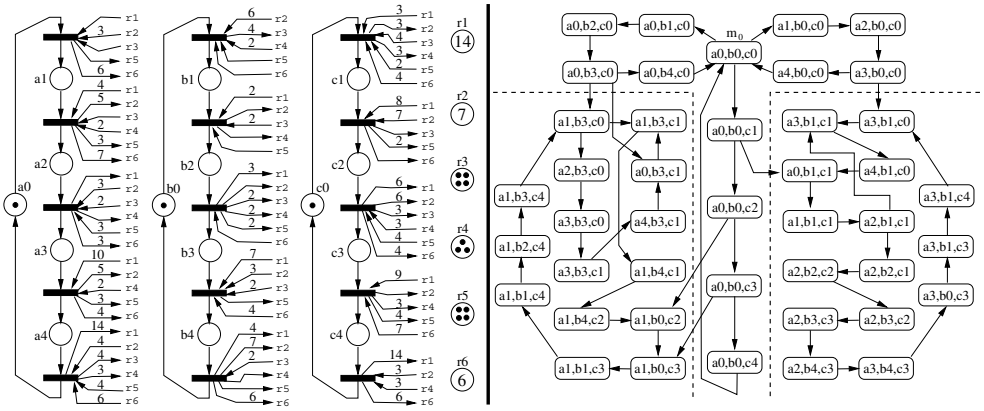

Fig. 2. The Foolgle net. A live and bounded SPQR that does not verify the directedness property. Note that $a0, b0, c0 \in P_R$.

reversible: to respect the general design principle referred some paragraphs above, the initial state (in which every robot is in its idle state) should be avoided.

The net in figure 2, however, is live but non reversible. It features three processes and nine resources ($r1$ to $r6$, which are the locks in the DLM, as well as $a0$, $b0$ and $c0$). It does not belong to the S^{4}PR class, but to one more general instead: the SPQR class. We shall introduce it in section III.

Remarkably, the system in figure 2 has some other notable properties: not only it is not reversible but it has no home states. On the contrary, it has two different terminal live regimes (one of which is fair [15]). Hence we may select in real-time the desired one; e.g. in function of the net traffic.

It is worth stressing that, in certain scenarios, a critical feature is that there must always exist an active process; i.e. at least one process out of its initial, idle state. Meanwhile, the system must be live. The reader may think of security control processes, or high-availability systems. Modelling this kind of systems goes beyond the S^{4}PR class. Nevertheless, although the modelling power of the SPQR class is higher, its analytical complexity also is (and, in particular, the complexity of liveness analysis).

III. THE SPQR CLASS

A. Purpose

The SPQR class is aimed to provide a consistent and general framework for the study of the resource allocation problem in S-RAS. First, this implies the generalization of previous, well-known models as well as the assumption of the analytical results developed for the S^{n}PR family, which can be easily mapped into the new class. Second, it also implies the enrichment of the existing expressive and analytical power by the addition of some new elements. In particular, we are interested in addressing the following types of systems:

- Systems in which there exist nested internal circuits within the control flow of the processes. This is typical in, e.g., software systems (iterative processes) or manufacturing systems (recirculating circuits) among others.
- Systems in which there are resources which are already allocated in the initial state.
- Open systems in which it is known the processes structure, but not the number of concurrent instances.
- Systems in which the number of resources is variable; despite the fact that processes use them in a conservative way (i.e., a process neither creates nor destroys resources in the system after completing its execution).

These enhancements can be captured by the new class definition, but they also raise unexpected properties and interesting questions regarding liveness analysis that will be studied in the following sections.

B. Definition

An SPQR is, in rough words, an S^{4}PR [9] without idle places, extended in a way such that processes can hold some resources in the initial state, or even when they are inactive. From a structural point of view, this means that every resource place of the net induces a unique p-flow Y_r, instead of a minimal p-semiflow.

Similarly to S^{4}PRs, the resource usage per process is conservative. Nonetheless, the idle place was an absolute minimum with relation to the resource usage state of the process. On the contrary, an SPQR may lack an absolute minimum, and this can severely complicate liveness analysis.

Due to the absence of the idle place, a marked SPQR can also be unbounded. The following is the formal definition of the SPQR class:

Definition 1: Let $I_{\mathcal{N}}$ be a finite set of indices. A *System of Processes Quarrelling over Resources (SPQR)* is a connected generalized pure P/T net, $\mathcal{N} = \langle P, T, C \rangle$, where:

1) $P = P_S \cup P_R$ where:
   a) $P_S = \cup_{i \in I_{\mathcal{N}}} P_i$, where $P_i \neq \emptyset$, $P_i \cap P_j = \emptyset$, for all $i, j \in I_{\mathcal{N}}$, $i \neq j$. [*process places*]
   b) $P_R \neq \emptyset$, $P_S \cap P_R = \emptyset$. [*resource places*]
2) $T = \cup_{i \in I_{\mathcal{N}}} T_i$, where $T_i \neq \emptyset$, $T_i \cap T_j = \emptyset$, for all $i, j \in I_{\mathcal{N}}$, $i \neq j$.
3) For each $i \in I_{\mathcal{N}}$, the subnet generated by restricting $\mathcal{N}$ to $\langle P_i, T_i \rangle$ is a connected acyclic state machine.
4) For each $r \in P_R$, exists a unique p-flow $Y_r \in \mathbb{Z}^{|P|}$ such that $\{r\} = ||Y_r|| \cap P_R$, $P_S \cap ||Y_r|| \neq \emptyset$ and $Y_r[r] = 1$.

Definition 2: A *Plain Lender Process* (PLP) is an SPQR in which $|I_{\mathcal{N}}| = 1$.

The reader can easily check that the composition of a non-empty set of PLPs by fusion of the common resource places is always an SPQR, assuming that the various sets P_i (T_i) are disjoint. Equivalently, any SPQR can be seen as the composition of a non-empty set of PLPs by fusion of some shared resource places.

C. On the processes structure

First, we will start with two definitions that will prove useful. Holder places were already defined for the S^{4}PR class. The following definition is consistent with that one:

Definition 3: Let $\mathcal{N}$ be a SPQR, $p \in P_S$, and $r \in P_R$:

- p is a *holder place* of r, $p \in \mathcal{H}(r)$, iff $Y_r[p] > 0$.
- p is a *lender place* of r, $p \in \mathcal{L}(r)$, iff $Y_r[p] < 0$.

Definition 4: Let $\mathcal{N}$ be a PLP, $\mathcal{N} = \langle P, T, C\rangle$. We define the binary relation $\leq_T$ in T by:

$$\leq_T = \{(t,t') \in T^2 \mid (t = t') \vee (\exists t'' \in {}^\bullet({}^\bullet t' \cap P_S),\ t \leq_T t'')\}$$

The relation $\leq_T$ is a precedence relation, i.e., $t \leq_T t'$ iff there is a directed path from t to t'. Since the state machine generated by $\langle P_S, T_S\rangle$ is acyclic, then $\leq_T$ induces a partial ordering on T. The pair $(T, \leq_T)$ is a poset [16].

Definition 5: A *Plain Borrower Process (PBP)* is a PLP such that, for every $r \in P_R$, there is no lender place of r in it, i.e., $Y_r[p] \geq 0$ for every process place p in the PBP.

The latter induce a special structure in PBPs. In particular, arcs from P_R always precede arcs to P_R in such a way that no resource instance is released before taken. This holds for every possible directed path in $\langle P_S, T_S\rangle$ [16]. Note that the verb "precede" refers to the partial order $\leq_T$.

For that reason, these processes are called *borrower* (PBP), in contrast to the (more general) *lender* processes (PLP), in which the above property may not hold. Figure 3 depicts an SPQR with one lender and one borrower process.

For notational convenience, we will use b-SPQR to denote those SPQRs in which every process is a PBP. Note that, in a b-SPQR, every p-flow Y_r (see point 4 in definition 1) is a minimal p-semiflow, i.e., $Y_r \in \mathbb{N}^{|P|}$ for every $r \in P_R$. Liveness analysis is more affordable for marked b-SPQRs, as will be illustrated in section V.

D. Relation to other subclasses

Proposition 6: Let $\mathcal{N} = \langle P_0 \cup P_S \cup P_R, T, C\rangle$ be a S^{4}PR net [9]. Then $\mathcal{N}' = \langle P_S \cup P'_R, T, C\rangle$, where $P'_R = P_0 \cup P_R$, is a b-SPQR.

Proof: Let $\mathcal{N}_i$, $i \in I_{\mathcal{N}}$, be a process net in $\mathcal{N}$ [9]. The subnet generated by restricting $\mathcal{N}_i$ to $\langle P_i, T_i\rangle$ becomes a connected *acyclic* state machine by eliminating p_{0_i} from P_i, since by definition every cycle contains p_{0_i}. Note that p_{0_i} is not eliminated, but 'moved' to P_R. Moreover, there exists a unique p-semiflow Y_{0_i}, $||Y_{0_i}|| = P_{S_i} \cup \{p_{0_i}\}$. Since $p_{0_i} \in P'_R$ and $P_{S_i} \cap P'_R = \emptyset$, Y_{0_i} holds the condition 4 in definition 1. Hence every process net in $\mathcal{N}$ is a PBP in $\mathcal{N}'$. ■

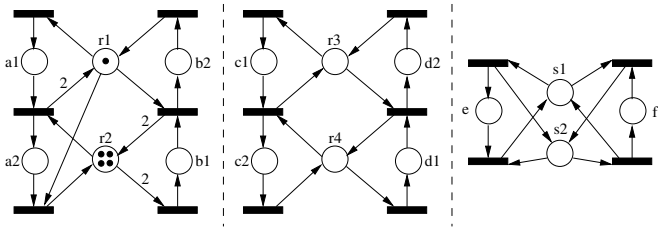

Fig. 3. On the left, a marked SPQR with two PLPs and two resources (r_1 and r_2). Only the PLP on the right is also a PBP. The net is non-live.

Fig. 4. In the middle, a b-SPQR. The net is structurally bounded. However it is non-live for every possible initial marking.

Fig. 5. On the right, an SPQR with two PLPs (none is borrower). The net is not structurally bounded (e and f are unbounded iff $m_0[s1]+m_0[s2] \neq 0$).

However, the inverse of proposition 6 is not true: not every b-SPQR is an S^{4}PR. A net in the b-SPQR class may not be structurally bounded (SB), while every S^{4}PR is SB. Nonetheless, there exists a transformation rule that transforms a non-SB b-SPQR into a SB b-SPQR preserving the liveness property. This will be shown in section V. It must be stressed that the class of SB b-SPQR subsumes, but it is not equal to, the class of S^{4}PR nets.

Obviously, S^{3}PRs [7] and L-S^{3}PRs can be [8] also redefined as SB b-SPQRs, since they are children of the S^{4}PR class. As far as we know, similar redefinitions in terms of the SPQR class can be applied to any previously defined Petri net model for S-RAS, except for S*PR nets, since the SPQR class does not directly deal with internal cycles.

Nevertheless, the reader might agree that in many real-world problems confining the syntax of processes in the S*PR class (which are state machines) to processes with nested loops may not be over-restrictive. Indeed this assumption applies rather well in software systems, as long as structured programming rules are followed. We refer the reader to the theorem of Böhm et Jacopini and related works [17] where it is demonstrated how "spaghetti" processes (using go-tos, exceptions) can always be transformed into structured processes.

This led us to the definition of the S^{5}PR class, which is in essence an extension of the S^{4}PR with the allowance of nested loops. It is always possible to transform an S^{5}PR into an equivalent SPQR, preserving its behaviour with relation to liveness. Every entry place of a loop is transformed into a lender place, adding a resource place that allows/disallows the execution of the circuit (i.e., loop). Then this circuit is no longer modelled as an internal circuit, but as a new PLP which is only executable (i.e., firable) when the resource place gets marked. This transformation is illustrated in figure 6, being p^i_k the entry place, $r_{i'}$ the new resource place, and $\mathcal{N}_{i'}$ the corresponding PLP for the loop.

IV. GLOBAL PROPERTIES OF SPQR NETS

For a better understanding of the inherent limitations and possibilities of the SPQR class, we will study here some structural and behavioural properties of these systems, comparing them with those of previous subclasses.

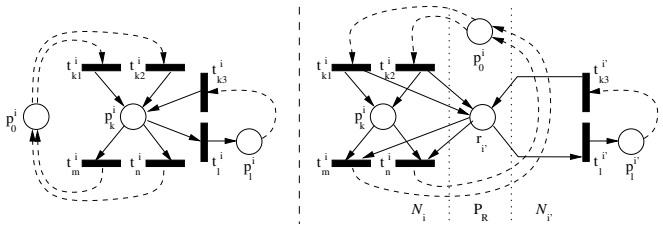

Fig. 6. Transformation rule from S^5PR (left) to SPQR (right). Dashed arcs indicate that part of the net is omitted. Other arcs from/to P_R are also omitted.

A. Structural properties

1) Conservativeness and structural boundness: A well-known general result of Petri nets is that conservativeness (CV) implies structural boundedness (SB) [18]. However, the inverse may not be true. In SPQR nets, nonetheless, both properties are equivalent. This is due to the fact that SPQRs are consistent by construction, and consistency plus SB implies CV [19]. For that reason, we will refer to SB SPQRs (or SB b-SPQRs) for nets in which every place of the net is covered by a p-semiflow. It must be noted, though, that the set of minimal p-semiflows may not be the various p-flows Y_r (see point 4, definition 1). This is only true for SB b-SPQRs.

It is easily inferrable from the above property that an SPQR $\mathcal{N}$ is SB iff every PLP in $\mathcal{N}$ is SB. Again, a PLP is SB if every process place is holder of a resource place, but this condition is only necessary, not sufficient, e.g., in figure 5, place e is a holder place of $s1$, while place f is a holder place of $s2$. However, both places e and f are unbounded if $s1$ or $s2$ are initially marked (i.e., non-empty). Note that $s1$ and $s2$ are SB. This result differs from S^4PRs, in which every net of the class is CV (and hence SB).

2) Structural liveness and repetitiveness: Another interesting difference is that resource places are no longer structurally implicit places. In the S^4PR class, this property was derived from the fact that every place was covered by a p-semiflow Y_r plus the existence of the *idle place*, which induced an additional p-semiflow which covered every process place. Here both conditions disappear.

As a consequence, SPQRs are not, in general, structurally live (SL); even despite they are always consistent. S^4PRs were SL because every resource place could be made implicit taking an initial marking higher enough. Doing so, the system became a set of isolated marked strongly connected state machines, hence being live. However, even SB b-SPQRs may be structurally non-live, as figure 4 depicts (this net can always be deadlocked by emptying places $c2$ and $d2$ and subsequently emptying $r3$ and $r4$ by firing the first transition of each process). This raises interesting questions regarding liveness synthesis that, in some subclasses, are yet to be fully addressed.

3) Structural directedness: The structural directedness (SD) property [20] states that for every pair of potentially reachable markings of a live system there is always a common successor marking. SD is, indeed, a stronger, structural version of the directedness property [21], which we will study among the behavioural properties, and holds for classes such as equal conflict (EQ) systems [20]. SD is very interesting from the standpoint of liveness analysis since it implies the absence of killing spurious solutions or, in other words, of live systems with potentially reachable markings being non-live.

Unfortunately, SD is only satisfied for the L-S^3PR class, as proven in [22], but none of the higher classes do, including the S^3PR class. The lack of this property hardens liveness analysis, due to the emergence of killing spurious solutions, and is obviously extensible to the more general SPQR class; although the RS of live SB SPQRs is not always directed either, as shown in figure 2 and discussed later.

B. Behavioural properties

1) On deadlock-freeness and liveness: One of the characteristic properties of the S^nPR family (including the L-S^3PR class) is that deadlock freeness does not imply liveness [22]. In this sense, they are trickier than other well-known Petri net classes such as strongly connected free choice systems [23], bounded strongly connected EQ systems [20] or CCS [2], where both properties are equivalent.

On the contrary, liveness is not even monotonic with respect to the initial marking (neither to the marking of the process places, nor that of the resource places), with the L-S^3PR as the unique exception ([22]). In fact, for S^3PRs and S^4PRs, there is a discontinuity zone between the point where the resource places are empty enough so that no transition is ever firable (all the lower markings imply a deadlock), and the point where every resource place is implicit (higher markings in them imply liveness). The markings within these bounds switch discontinuously between liveness and non-liveness. Of course, the location of those points also depends on the marking of the process places. However, SPQRs and b-SPQRs are not (in general) SL, not even being SB, and this implies that there may no longer exist an upper liveness region (e.g., the net in figure 4).

2) Reversibility: In S^4PRs with an acceptable initial marking, reversibility is a necessary and sufficient condition for liveness [14]. However, the class considered here (SPQR) is more general so, in particular, reversibility is not necessary for liveness. The figure 2 depicts a system live but not reversible. Besides, reversibility is neither sufficient for liveness, provided that it is no longer required that all the t-components are firable in isolation from m_0.

3) Directedness: The directedness property [21] states that, for every pair of reachable markings in a live system, there is always a common successor marking. Although the directedness property obviously holds for the S^4PR class (reversibility equals liveness), it is not verified, in general, for the SPQR class, as figure 2 reflects. The RS of the net has two terminal strongly connected components, being the net live.

For bounded marked nets, the directedness property is equivalent to the existence of home states [21]. Hence, live S^4PR nets have home states; indeed, every reachable marking

is a home state, including the initial marking m_0, since the net is reversible[1]. This is very useful for determining if the net is non-live, since the death of the system can be reduced to: "Is m_0 unreachable from some reachable marking?". Even more, if it is reachable we can systematically construct a path that leads to m_0, and the length of this path is not higher than the size of the net, due to the structure of the S^{4}PR class.

The "bad" news here is that, since the directness property is not held by the SPQR class, not even if the net is SB, we cannot ensure (in general) that a home state will exist, whether the net is live or not. Again, figure 2 is a good example of this kind of behaviour. This is a severe problem for determining non-liveness in an efficient way, as will be patent in the following.

A table in section VI summarizes and compares the main properties of each class.

V. LIVENESS ANALYSIS FOR SPQR NETS

In the following, we will reference some interesting new results for liveness analysis in SPQR nets. Unfortunately, we cannot include detailed proofs of the theorems here due to extension limits. If interested, the reader will find them in detail at [16]. However, we consider these results are worth being referenced here since they highlight some difficulties that did not exist with previous classes.

In particular, we provide a characterization of liveness for b-SPQRs, which generalizes previous results on the simpler S^{4}PR class. We also point at either necessary or sufficient conditions for non-liveness in SB SPQRs with lender processes (siphons do not longer work so no structural characterization could be provided). Non-SB SPQRs with lender processes are yet open to future work.

A. Basics

First, it is worth stressing that the results presented below are valid for any non-negative initial marking. On the contrary, previous results were only valid for marked nets with an *acceptable initial marking* (in which every state machine should be firable in isolation). This restriction has been removed, thanks to the introduction of a new behavioural property, which we call *strong reproducibility*. Fortunately, there exists a structural characterization for this property in SPQR nets under certain conditions.

Definition 7: Let $\langle \mathcal{N}, m_0 \rangle$, $\mathcal{N} = \langle P, T, W \rangle$, be any marked consistent P/T net. $\langle \mathcal{N}, m_0 \rangle$ is *strongly reproducible* iff for every transition $t \in T$ exists a firing sequence σ, $t \in \sigma$, such that $m_0[\sigma\rangle m_0$.

We can alternatively say that m_0 is strongly reproducible in $\mathcal{N}$. The idea of *strong reproducibility* is inspired in that of reproducibility, as expressed by Lautenbach [24].

Note that strong reproducibility is a behavioural property and, in general, is neither necessary nor sufficient for reversibility (and the same applies with respect to liveness). Nonetheless, there is an structural necessary condition that captures this property for the SPQR class:

Lemma 8: [16] Let $I_{\mathcal{N}}$ be a finite set of indices, and $\langle \mathcal{N}, m_0 \rangle$ be a marked SPQR, $\mathcal{N} = \langle P, T, C \rangle$. Assuming that $Y_r[\emptyset] = 0$, then $\langle \mathcal{N}, m_0 \rangle$ is strongly reproducible if $\forall i \in I_{\mathcal{N}}$. $\exists t \in T_i$ such that:

- $m_0[t\rangle$, i.e., t is enabled at m_0, and
- $\forall r \in P_R$. $m_0[r] \geq max_{p \in P_i \cup \{\emptyset\}}(Y_r[p] - Y_r[{}^\bullet t \cap P_i])$.

For marked b-SPQR nets, this condition is necessary and sufficient when $m_0[P_S] = \mathbf{0}$. Being checkable in linear time in the size of the net, this property will prove specially convenient for liveness analysis in this subclass.

The following definition will also prove much useful. It is worth highlighting the different semantics between the concept of *m-process-enabled transition* defined for the SPQR class and the homonym used in previous works [9]. Here it is slightly changed for notational convenience:

Definition 9: Let $\langle \mathcal{N}, m \rangle$ be a marked SPQR, $\mathcal{N} = \langle P, T, C \rangle$:

- $t \in T$ is *m-process-enabled* iff ${}^\bullet t \cap P_S = \emptyset$ or $m[{}^\bullet t \cap P_S] > \mathbf{0}$. Otherwise, t is *m-process-disabled*.
- $t \in T$ is *m-resource-enabled* iff $\forall r \in {}^\bullet t \cap P_R$, $m[r] \geq Pre[r, t]$. Otherwise, t is *m-resource-disabled*.

B. Liveness analysis for b-SPQR nets

The following theorem generalizes the non-liveness characterization for the S^{4}PR class [9]:

Theorem 10: [16] Let $\langle \mathcal{N}, m_0 \rangle$ be a marked SB b-SPQR, $\mathcal{N} = \langle P, T, C \rangle$, and $T^0 = \{t \in T \mid {}^\bullet t \cap P_S = \emptyset\}$. Then $\langle \mathcal{N}, m_0 \rangle$ is non-live iff

- $\langle \mathcal{N}, m_{idle} \rangle$ is strongly reproducible, where m_{idle} is the unique solution to the linear system:
  - $m_{idle}[P_S] = \mathbf{0}$, $m_{idle}[P_R] \geq \mathbf{0}$,
  - $m_{idle} = m_0 + C \cdot x$, $x \geq \mathbf{0}$.
- and $\exists m \in RS(\mathcal{N}, m_0)$ such that the set of m-process-enabled transitions in $T \setminus T^0$ is non-empty and each one of these transitions is m-resource-disabled.

Since $\mathcal{N}$ is SB, it can be transformed into an S^{4}PR by inserting one implicit place p_{0_i} per every PBP $\mathcal{N}_i$, where $p_{0_i}{}^\bullet = \{t \mid {}^\bullet t \cap P_S = \emptyset\}$ and ${}^\bullet p_{0_i} = \{t \mid t^\bullet \cap P_S = \emptyset\}$. Then we can proceed the demonstration by using a reasoning quite similar to that introduced in [9]. In the same vein, an equivalent condition can be expressed in terms of *insufficiently marked* siphons.

The next theorem allows using the previous result for analyzing the liveness of non-SB b-SPQRs. This is based on the fact (proved in [16]) that we can safely omit the unbounded places when analyzing the liveness of a non-SB b-SPQR. Unfortunately, this does not work with general SPQRs:

Theorem 11: [16] Let $\langle \mathcal{N}, m_0 \rangle$ be a marked b-SPQR, and let P_u be the set of process places of $\mathcal{N}$ that are not covered by any p-semiflow, i.e. $P_u = P_S \setminus (\cup_{r \in P_R} \mathcal{H}(r))$. Then $\langle \mathcal{N}, m_0 \rangle$ is non-live iff $\langle N', m_0' \rangle$ is non-live, where $\mathcal{N}'$ is the SB b-SPQR generated by restricting $\mathcal{N}$ to $\langle P \setminus P_u, T \rangle$, and m_0' is the mapping of m_0 over $P \setminus P_u$.

[1] It is interesting to note that, in live L-S^{3}PR systems, a stronger condition holds: every potentially reachable marking is a home state [22].

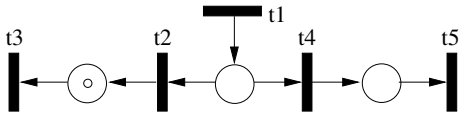

Fig. 7. A PLP net $\mathcal{N}_i$ (the arcs from/to P_R have been omitted). The white token marks the result of the state-choice function $g(i)$.

C. Liveness analysis for SB SPQR nets

In case that not every process is borrower, the previously presented results cannot be applied, since the directedness property no longer holds. Here we will point at two new necessary or sufficient conditions that can be implemented in a computationally efficient manner. Due to extension limits, they will be presented here in a pretty concise way. An example will be provided to remark the fact that siphons are no longer the key to effective liveness analysis. Some useful definitions will be introduced before.

1) Preliminaries: A state-choice function is a function that, given an SPQR, selects at most one place per PLP. This is a key concept for both the necessary and sufficient condition, and it will also enable us to introduce the concepts of g-enabled and g-router transition:

Definition 12: Let $\mathcal{N}$ be an SPQR, $\mathcal{N} = \langle P, T, C\rangle$, and let g be a partial function $g : I_{\mathcal{N}} \rightarrow P_S$. Then g is a *state-choice function* on $\mathcal{N}$ iff, for every $i \in Dom(g)$, $g(i) \in P_i$.

For practical purposes, we will assume that $g(i) = \emptyset$ when $i \notin Dom(g)$.

Any transition of an SPQR can be g-enabled (or g-disabled) and also g-router (or not). These conditions depend both on the net structure and on the state-choice function g, but not on the marking. A transition will be g-enabled iff its entry place is selected by the state-choice function. In rough words, a transition will be g-router iff its firing does *not* move the token further away from the target place selected by g. More formally:

Definition 13: Let $\mathcal{N}$ be an SPQR, $\mathcal{N} = \langle P, T, C\rangle$, and let g be a state-choice function on $\mathcal{N}$:

- $t \in T_i$ is *g-enabled* iff ${}^\bullet t \cap P_S = g(i)$. Otherwise, t is *g-disabled*.
- $t \in T_i$ is *g-router* iff:
  - $g(i) = \emptyset$, or else,
  - $t \leq_T t_g$, or else,
  - $\nexists\ t' \in T_i$ s.t. $(Pre[P_i, t] = Pre[P_i, t']) \wedge (t' \leq_T t_g)$,

  where $t_g = {}^\bullet g(i)$.

In figure 7, a PLP net is depicted. In this case, t_1, t_2 and t_5 are g-disabled and g-router; t_3 is g-enabled; and t_4 is g-disabled but not g-router. The only m-process-enabled transition is t_1, since all the process places are empty (note that the white token is not a real token of the net).

2) Necessary condition:

Theorem 14: [16] Let $I_{\mathcal{N}}$ be a finite set of indices, and $\langle \mathcal{N}, m_0\rangle$ be a marked SB SPQR. $\langle \mathcal{N}, m_0\rangle$ is live if exists a state-choice function g such that:

- Exists a solution to the linear system:
  - $m_{goal} \geq \mathbf{0}$,
  - $m_{goal} = m_0 + C \cdot x$, $x \geq \mathbf{0}$,
  - $||m_{goal}[P_S]|| = \cup_{i \in I_{\mathcal{N}}} g(i)$.
- $\langle \mathcal{N}, m_{goal}\rangle$ is strongly reproducible, and
- $\nexists\ m \in RS(\mathcal{N}, m_0)$ such that the set of transitions that are m-process-enabled and g-disabled is non-empty and each one of these transitions is m-resource-disabled and g-router.

In other words, m_{goal} is one of the possible markings in which only the places selected by the function g are marked. Note that this marking may not be reachable, since there exist spurious solutions of the net state equation.

In the case of SB b-SPQRs, the condition collapses into the necessary and sufficient condition introduced before, since it is enough to consider the state-choice function $g(i) = \emptyset$, for every $i \in I_{\mathcal{N}}$ (hence $m_{goal}[P_S] = m_{idle}[P_S] = \mathbf{0}$).

3) Sufficient condition:

Theorem 15: [16] Let $\langle \mathcal{N}, m_0\rangle$ be a marked SB SPQR. $\langle \mathcal{N}, m_0\rangle$ is non-live if exists a state-choice function g and a reachable marking m, $m \in RS(\mathcal{N}, m_0)$, such that the set of transitions that are m-process-enabled and g-disabled is non-empty and each one of these transitions is g-router and m'-resource-disabled, for every $m' \in \{m + C \cdot \boldsymbol{\sigma} \,|\, (\boldsymbol{\sigma} \geq \mathbf{0}) \wedge (\boldsymbol{\sigma}[T_d] = \mathbf{0})\}$, where $T_d = \{t \in T \,|\, t$ is m-process-enabled, g-disabled, and g-router$\}$.

Again, the condition collapses into the necessary and sufficient for SB b-SPQRs. For general SPQRs, this condition is necessary and sufficient if, instead of computing m' using the net state equation, we apply the condition only for every marking $m' \in RS(\mathcal{N}, m)$ that can be reached by firing transitions in $T \setminus T_d$. Obviously, the tradeoff is a higher computational complexity.

As an exciting brainstorming exercise, we leave the net in figure 8 to the reader. This net is doubly *within the gap* between the necessary and sufficient conditions: there is no known characterization of (non-)liveness. In particular, we want to stress that siphons no longer succeed in fully capturing liveness for this kind of nets.

VI. CONCLUSIONS

In this paper, a new Petri net class for S-RAS called SPQR has been introduced. The new framework colligates previous theoretical achievements for dealing with the resource allocation problem within this system category. Additionally, the generalization is enriched providing support for other S-RAS which were not supported by previous classes. This includes systems in which there are nested iterations within the processes, among others.

We have also studied the general properties of the systems that can be modelled with the class, and compared them with those of earlier members of the S^{n}PR family. In some cases, the results are surprising and clearly reveal the inherent complexity of the class. As a result, the previous siphon-based characterizations for liveness analysis are no longer valid. We provide instead a necessary as well as a sufficient condition for non-liveness in SB SPQRs. Check algorithms for both conditions can be efficiently implemented. Moreover, both

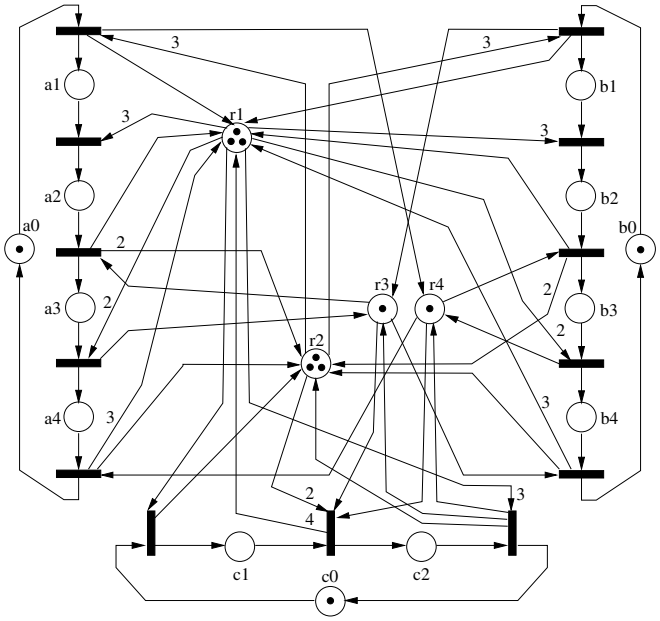

Fig. 8. An ill-fated net with no known structural decision criteria for (non-)liveness. The net is non-live for the given initial marking, but the sufficient condition does not work. If we make $m_0[r_4] = 2$, then the net is live, but the necessary condition also fails.

collapse into a necessary and sufficient condition for certain subclasses, such as b-SPQRs.

The following table highlights the main similarities and differences between the different members of the S^{n}PR family:

Property	**L-S^{3}PR**	**S^{3}PR**	**S^{4}PR**	**S^{5}PR**	**SPQR**
Structural					
Well-formedness	✓	✓	✓	✓	×
Structural directedness	✓	×	×	×	×
Behavioural (for an m_0 acceptable for the class)					
RS = PRS	×	×	×	×	×
DF = Liveness	×	×	×	×	×
Liveness monotonicity	✓	×	×	×	×
Directedness	✓	✓	✓	×	×
Live ⇒ Home states	✓	✓	✓	×	×
Reversible ⇒ Live	✓	✓	✓	✓	×

REFERENCES

[1] J. F. Kurose and R. Simha, "A microeconomic approach to optimal resource allocation in distributed computer systems," *IEEE Transactions on Computers*, vol. 38, no. 5, pp. 705–717, 1989.

[2] K. Lautenbach and P. S. Thiagarajan, "Analysis of a resource allocation problem using Petri nets." in *Proc. of the 1st European Conf. on Parallel and Distributed Processing*, Syre, J.C., Ed. Toulouse: Cepadues Editions, 1979, pp. 260–266.

[3] E. G. Coffman, M. Elphick, and A. Shoshani, "System deadlocks," *ACM Computing Surveys*, vol. 3, no. 2, pp. 67–78, 1971.

[4] X. Xie and M. D. Jeng, "ERCN-merged nets and their analysis using siphons." *IEEE Transactions on Robotics and Automation*, vol. 29, no. 4, pp. 692–703, 1999.

[5] J. Ezpeleta and L. Recalde, "A deadlock avoidance approach for non–sequential resource allocation systems," *IEEE Transactions on Systems, Man and Cybernetics. Part–A: Systems and Humans*, vol. 34, no. 1, 2004.

[6] M. P. Fanti, B. Maione, S. Mascolo, and B. Turchiano, "Event-based feedback control for deadlock avoidance in flexible production systems," *IEEE Transactions on Robotics and Automation*, vol. 13, no. 3, pp. 347–363, 1997.

[7] J. Ezpeleta, J. Colom, and J. Martínez, "A Petri net based deadlock prevention policy for flexible manufacturing systems," *IEEE Transactions on Robotics and Automation*, vol. 11, no. 2, pp. 173–184, 1995.

[8] J. Ezpeleta, F. García-Valles, and J. Colom, "A class of well structured Petri nets for flexible manufacturing systems," in *Proc. of the 19th Int. Conf. on Application and Theory of Petri Nets*, ser. LNCS, J. Desel and M. Silva, Eds., vol. 1420. Lisbon, Portugal: Springer–Verlag, June 1998, pp. 65–83.

[9] F. Tricas, "Deadlock analysis, prevention and avoidance in sequential resource allocation systems," Ph.D. dissertation, University of Zaragoza, Zaragoza, May 2003.

[10] J. Park and S. A. Reveliotis, "Deadlock avoidance in sequential resource allocation systems with multiple resource acquisitions and flexible routings," *IEEE Transactions on Automatic Control*, vol. 46, no. 10, pp. 1572–1583, 2001.

[11] S. A. Reveliotis, M. A. Lawley, and P. M. Ferreira, "Polynomial complexity deadlock avoidance policies for sequential resource allocation systems," *IEEE Transactions on Automatic Control*, vol. 42, no. 10, pp. 1344–1357, 1997.

[12] J. Ezpeleta, F. Tricas, F. García-Vallés, and J. Colom, "Bankers–like approaches to deadlock avoidance in concurrent systems," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 4, pp. 621–625, 2002.

[13] J. M. Colom, "The resource allocation problem in flexible manufacturing systems," in *Proc. of the 24th Int. Conf. on Applications and Theory of Petri Nets*, ser. LNCS, Van der Aalst, W. and Best, E., Ed., vol. 2679. Eindhoven, Netherlands: Springer–Verlag, June 2003, pp. 23–35.

[14] S. A. Reveliotis, "On the siphon-based characterization of liveness in sequential resource allocation systems," in *Proc. of the 24th Int. Conf. on Applications and Theory of Petri Nets*, ser. LNCS, Van der Aalst, W. and Best, E., Ed., vol. 2679. Eindhoven, Netherlands: Springer–Verlag, June 2003, pp. 241–255.

[15] T. Murata, "Petri nets: Properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541–580, 1989.

[16] J. P. López-Grao and J. M. Colom, "Liveness enforcing in Resource Allocation Systems with iterative processes: A study on the SPQR class," Dept. Informática e Ingeniería de Sistemas, Tech. Rep., 2006.

[17] D. Harel, "On folk theorems," *Communications of the ACM*, vol. 23, no. 7, pp. 379–389, 1980.

[18] M. Silva and J. Colom, "On the computation of structural synchronic invariants in P/T nets," in *Advances in Petri Nets 1988*, G. Rozenberg, Ed. Berlin: Springer–Verlag, 1988, vol. 340, pp. 386–417.

[19] M. Silva, "Introducing Petri nets," in *Practice of Petri nets in manufacturing*, F. DiCesare, G. Harhalakis, J. Proth, M. Silva, and F. Vernadat, Eds. Chapman and Hall, 1993, pp. 1–62.

[20] E. Teruel and M. Silva, "Liveness and home states in equal conflict systems," in *Proc. of the 14th Int. Conf. on Application and Theory of Petri Nets*, ser. LNCS, M. Ajmone Marsan, Ed. Springer–Verlag, 1993, vol. 691, pp. 415–432.

[21] E. Best and K. Voss, "Free choice systems have home states." *Acta Informatica 21*, pp. 89–100, 1984.

[22] F. García-Vallés, "Contributions to the structural and symbolic analysis of place/transition nets with applications to flexible manufacturing systems and asynchronous circuits," Ph.D. dissertation, University of Zaragoza, Zaragoza, April 1999.

[23] D. Hillen, "Relationship between deadlock-freeness and liveness in free-choice nets." *Newsletter*, no. 19, pp. 28–32, Feb. 1985.

[24] K. Lautenbach, "Reproducibility of the empty marking," in *Proc. of the 23rd Int. Conf. on Applications and Theory of Petri Nets*, ser. LNCS, J. Esparza and C. Lakos, Eds., vol. 2360. London, UK: Springer-Verlag, 2002, pp. 237–253.