Chapter 13
Structural methods for the control of Discrete Event Dynamic Systems – The case of the Resource Allocation Problem

Juan-Pablo López-Grao and José-Manuel Colom

Abstract The study of resource allocation related aspects is a fundamental issue in the design and control of Discrete Event Dynamic Systems (DEDSs) belonging to domains ranging from multithreaded software applications to Flexible Manufacturing Systems (FMSs). The formulation of this application-driven problem in terms of Petri nets leads to a family of net models with a specific structure-based characterization. These net subclasses are derived from a specific methodology to abstract the system in order to obtain its Resource Allocation System (RAS) view, which we describe in this chapter. After that, we concentrate our efforts in the characterization of the liveness of such models. The structural causes of the non-liveness (deadlock of some processes) are also discussed. These will lay the foundations to introduce control elements which forbid all the bad states enforcing the liveness property. The methods to compute the control are based on structural techniques avoiding the construction of the reachability graph.

13.1 Introduction

Resource scarceness is a traditional scenario in many systems engineering disciplines. In such cases, available resources may be shared among concurrent processes, which must compete in order to be granted their allocation. DEDSs of this kind are named *RASs*. In this paper we revisit a family of Petri net-based deadlock handling methodologies which have been successfully applied in many application domains, such as FMSs [6], multicomputer interconnection networks [25] or multi-

Juan-Pablo López-Grao
Department of Computer Science and Systems Engineering, University of Zaragoza, Spain.
e-mail: jpablo@unizar.es

José-Manuel Colom
Aragon Institute of Engineering Research (I3A), University of Zaragoza, Spain.
e-mail: jm@unizar.es

threaded software engineering [21]. These methodologies are based in the attention to the RAS view of the system and are basically deployed in three stages.

The first stage is that of abstraction and modelling, in which physical details of the system not relevant to the Resource Allocation Problem (RAP) are discarded, obtaining a Petri net as outcome. That Petri net is processed in the analysis stage, in which potential deadlock situations due to diseased resource allocation patterns are inspected. At the third stage, that of synthesis and implementation, the Petri net is corrected in order to obtain a deadlock-free system, usually by the addition of virtual resources. This correction is finally unfolded in the real system by the correction of the processes involved and the inclusion of control mechanisms. Often structural results exist which enable powerful structure-based analysis and synthesis techniques for identifying and fixing potential or factual deadlocks [6, 22, 28]. the model over the real-world system.

Regarding the first stage of abstraction, and with a view to obtain a useful, descriptive but tractable model, it is necessary to have into account the different characteristics and physical restrictions derived from the application domain. Consequently, the syntax of the RAS models obtained for different domains can differ notably. On the following, we address a classification of the diverse models developed in the literature, and their specific features with regard to RAS modelling.

Basically, restrictions on Petri net models for RASs 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 concerned, resources can be serially reusable (i.e., they are used in a conservative way by all processes), or consumable (i.e., they are consumed and not regenerated). This work focuses on RAS with serially reusable resources.

Regarding the processes structure, most of the current works focus on Sequential RASs (S-RASs), as opposed to Non-Sequential RASs (NS-RASs), in which assembly/disassembly operations are allowed within the processes. Some works ([30, 8]), however, have attempted to approach NS-RASs from the Petri nets perspective, despite that finding effective solutions for them is, in general, much more complicated.

In the field of S-RASs (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.

One of the first classes aimed to deal with the resource allocation problem in S-RASs is the class of Cooperating Sequential Systems (CSSs) [14]. 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 [10], 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 [6], where the seminal System of Simple Sequential Processes with Resources (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 Linear S^3PR (L-S^3PR), was presented in [7], which featured some useful properties.

The mentioned restriction over resources usage is eliminated by the (more general) S^{4}PR class [27] (S^{3}PGR^2 in [22]). This allows processes to simultaneously reserve several resources belonging to distinct types. Many others [13, 30] were defined for this aim, with specific attributes for modelling different configurations.

In Section 13.2, the most general Petri net classes for RASs are reviewed and categorized. In Section 13.3, the concept of insufficiently marked siphon is introduced as an artifact to approach the liveness problem in RASs from the system structure. It will be shown, however, that this artifact falls short on characterizing liveness in the context of multithreaded control software. In Section 13.4, an iterative control policy for RASs is presented. Section 13.4 summarizes the conclusions.

13.2 Abstraction and modelling. Class definitions and relations

S^{4}PR nets are modular models composed of state machines with no internal cycles plus shared resources. One of the most interesting features of this kind of models is their composability. Two S^{4}PR nets can be composed into a new S^{4}PR model via fusion of the common resources. Since multiple resource reservation is allowed, S^{4}PR nets are not ordinary, i.e., the weight of the arcs from the resources to the state machines (or vice versa) is not necessarily equal to one, in contrast to S^{3}PR nets.

Definition 1. [28] Let I_N be a finite set of indices. An S^{4}PR is a connected generalized pure P/T net $N = \langle P, T, \mathbf{Pre}, \mathbf{Post} \rangle$ where:

1. $P = P_0 \cup P_S \cup P_R$ is a partition such that:
   a. *[idle places]* $P_0 = \bigcup_{i \in I_N} \{p_{0_i}\}$.
   b. *[process places]* $P_S = \bigcup_{i \in I_N} P_{S_i}$, where $\forall i \in I_N : P_{S_i} \neq \emptyset$, and $\forall i, j \in I_N,\ i \neq j : P_{S_i} \cap P_{S_j} = \emptyset$.
   c. *[resource places]* $P_R = \{r_1, r_2, r_3, ..., r_n\}, n > 0$.
2. $T = \bigcup_{i \in I_N} T_i$, where $\forall i \in I_N : T_i \neq \emptyset$, and $\forall i, j \in I_N, i \neq j : T_i \cap T_j = \emptyset$.
3. *[i-th process subnet]* For each $i \in I_N$ the subnet generated by $\{p_{0_i}\} \cup P_{S_i}$, T_i is a strongly connected state machine such that every cycle contains p_{0_i}.
4. For each $r \in P_R$ there exists a unique minimal p-semiflow associated to r, $\mathbf{y_r} \in \mathbb{N}^{|P|}$, fulfilling: $\{r\} = \|\mathbf{y_r}\| \cap P_R, P_0 \cap \|\mathbf{y_r}\| = \emptyset, P_S \cap \|\mathbf{y_r}\| \neq \emptyset$, and $\mathbf{y_r}[r] = 1$.[1]
5. $P_S = \bigcup_{r \in P_R} (\|\mathbf{y_r}\| \setminus \{r\})$.

Fig. 13.1 depicts a net system belonging to the S^{4}PR class. Places $R1$ and $R2$ are the *resource places*. A resource place represents a resource type, and the number of tokens in it represents the quantity of free instances of that resource type. If we remove these places, we get two isolated state machines. These state machines represent the different patterns of resource reservation that a process can follow. In the context of FMSs, these two state machines model two different production plans.

[1] The support of a p-semiflow $\mathbf{y}$ (marking $\mathbf{m}$), denoted $\|\mathbf{y}\|$ ($\|\mathbf{m}\|$), is the set of places such that their corresponding components in the vector $\mathbf{y}$ ($\mathbf{m}$) are non-null.

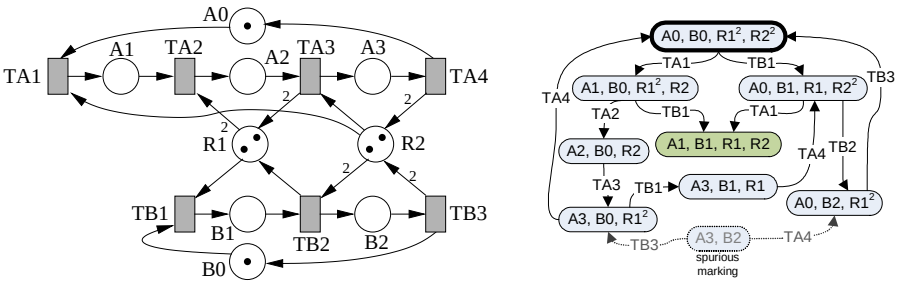

Fig. 13.1 A (non-live) S^4PR with an acceptable initial marking.

Consequently, tokens in a state machine represent parts which are being processed in stages of the same production plan. At the initial state, the unique tokens in each machine are located at the so-called *idle place* (here: $A0$, $B0$). In general, the idle place can be seen as a mechanism which limits the maximum number of concurrent parts being processed in the same production plan. The rest of places model the various stages of the production plan as far as resource reservation is concerned.

Meanwhile, the transitions represent the acquisition or release of resources by the processes along their evolution through the production plan. Every time a transition fires, the total amount of resources available is altered while the part advances to the next stage. The weight of an arc connecting a resource with a transition models the number of instances which are allocated or released when a part advances.

For instance, place $R1$ could model a set of free robotic arms used to process parts in the stage $A2$ of the first production plan (two arms are needed per each part processed there) as well as in the stage $B1$ of the second production plan (only one arm needed per part processed). Consequently, if transition $TB1$ is fired from the initial marking then one robotic arm will be allocated and one part will visit stage $B1$. Still, there will remain one robotic arm to be used freely by other processes.

Finally, it is worth noting that moving one isolated token of a state machine (by firing its transitions) until the token reaches back the idle state, leaves the resource places marking unaltered. Thus, the resource usage is conservative.

The next definition formalizes the fact that there should exist enough free resource instances in the initial state so as that every production plan is realizable:

Definition 2. [28] Let $N = \langle P, T, \mathbf{Pre}, \mathbf{Post} \rangle$ be an S^4PR. An initial marking $\mathbf{m_0}$ is acceptable for N iff $\|\mathbf{m_0}\| = P_0 \cup P_R$ and $\forall p \in P_S, r \in P_R : \mathbf{m_0}[r] \geq \mathbf{y_r}[p]$.

Nowadays, the most general class of the S^nPR family is the S^*PR class [9], in which processes are ordinary state machines with internal cycles. 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 [4].

All the aforementioned Petri net classes are frequently presented in the context of FMS modelling, and make sense as artifacts conceived for properly modelling significant physical aspects of this kind of systems. For all of these, except for S^*PR, a siphon-based liveness characterization is known. Due to its structural nature, it

opens a door to an efficient detection and correction of deadlocks, by implementing controllers (usually by the addition of places) that restrain the behaviour of the net and avoid the bad markings to be reached.

Although there exist obvious resemblances between the RAP in FMSs and that in parallel or concurrent software, previous attempts to bring these well-known RAS techniques into the analysis of multithreaded control software has been, to the best of our knowledge, either too limiting or unsuccessful. Gadara nets [29] constitute the most recent attempt, yet they fall in the over-restrictive side in the way the resources can be used [20]. Presumably, this is a consequence of being conceived with a primary focus on inheriting the powerful structural liveness results which were fruitful in the context of FMSs. Such a bias works against obtaining a model class capable of properly abstracting RASs in many multithreaded systems [20]. In [21], it is basically analyzed why the net classes and results introduced in the context of FMSs can fail when brought to the field of concurrent programming.

In [21], the class of Processes Competing for Conservative Resources (PC^2R) is introduced. This class is aimed to overcome the deficiencies identified in finding models which properly capture the RAS view of multithreaded software systems. Furthermore, it generalizes other subclasses of the S^nPR family while respecting the design philosophy on these. Hence, previous results are still valid in the new framework. However, PC^2R nets can deal with more complex scenarios which were not yet addressed from the domain of S^nPR nets. The generalization is also useful in the context of FMS configuration 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.

Definition 3 presents a subclass of state machines used for modelling the control flow of the processes of a PC^2R net in isolation. Iterations are allowed, as well as decisions within internal cycles, in such a way that the control flow of structured programs can be fully supported. Non-structured processes can be refactored into structured ones as discussed in [21].

Definition 3. [21] An *iterative state machine* $N = \langle P, T, \mathbf{Pre}, \mathbf{Post} \rangle$ is a strongly connected state machine such that: (i) P can be partitioned into three subsets: $\{p_k\}$, P_1 and P_2, (ii) $P_1 \neq \emptyset$, (iii) The subnet generated by $\{p_k\} \cup P_1, {}^\bullet P_1 \cup P_1{}^\bullet$ is a strongly connected state machine in which every cycle contains p_k, and (iv) If $P_2 \neq \emptyset$, the subnet generated by $\{p_k\} \cup P_2, {}^\bullet P_2 \cup P_2{}^\bullet$ is an iterative state machine.

As Fig. 13.2 shows, P_1 contains the set of places of an outermost iteration block, while P_2 is the set of places of the rest of the state machine (the inner structure, which may contain multiple loops within). Consequently, the subnet generated by $\{p_k\} \cup P_1, {}^\bullet P_1 \cup P_1{}^\bullet$ is a strongly connected state machine in which every cycle contains p_k. Meanwhile, inner iteration blocks can be identified in the iterative state machine generated by $\{p_k\} \cup P_2, {}^\bullet P_2 \cup P_2{}^\bullet$. The place p_0 represents the place "p_k" that we choose after removing every iteration block.

The definition of iterative state machine is instrumental for introducing the class of PC^2R nets. PC^2R nets are modular models composed by iterative state machines

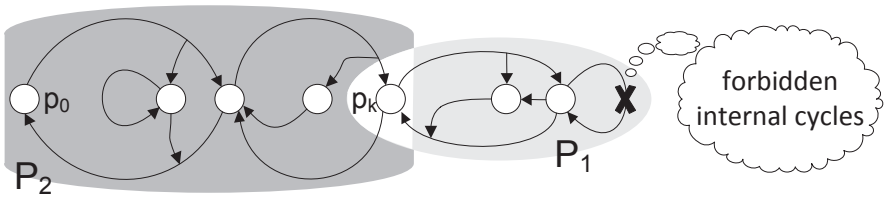

Fig. 13.2 Schematic diagram of an iterative state machine. For simplicity, no transition is drawn.

and shared resources. Two PC^{2}R nets can be composed into a new PC^{2}R model via fusion of the common resources. Note that a PC^{2}R net can simply be one process modelled by an iterative state machine along with the set of resources it uses.

The class supports iterative processes, multiple resource acquisitions, non-blocking wait operations and resource lending. Inhibition mechanisms are not natively supported (although some cases can still be modelled with PC^{2}R nets).

Definition 4. [21] Let I_N be a finite set of indices. A PC^{2}R net is a connected generalized self-loop free P/T net $N = \langle P, T, \mathbf{Pre}, \mathbf{Post} \rangle$ where:

1. $P = P_0 \cup P_S \cup P_R$ is a partition such that:
   a. *[idle places]* $P_0 = \bigcup_{i \in I_N} \{p_{0_i}\}$.
   b. *[process places]* $P_S = \bigcup_{i \in I_N} P_{S_i}$, where $\forall i \in I_N : P_{S_i} \neq \emptyset$, and $\forall i, j \in I_N,\ i \neq j : P_{S_i} \cap P_{S_j} = \emptyset$.
   c. *[resource places]* $P_R = \{r_1, r_2, r_3, ..., r_n\}, n > 0$.
2. $T = \bigcup_{i \in I_N} T_i$, where $\forall i \in I_N : T_i \neq \emptyset$, and $\forall i, j \in I_N, i \neq j : T_i \cap T_j = \emptyset$.
3. *[i-th process subnet]* For each $i \in I_N$ the subnet generated by $\{p_{0_i}\} \cup P_{S_i}$, T_i is an iterative state machine.
4. For each $r \in P_R$, there exists a unique minimal p-semiflow associated to r, $\mathbf{y_r} \in \mathbb{N}^{|P|}$, fulfilling: $\{r\} = \|\mathbf{y_r}\| \cap P_R, (P_0 \cup P_S) \cap \|\mathbf{y_r}\| \neq \emptyset$, and $\mathbf{y_r}[r] = 1$.
5. $P_S \subseteq \bigcup_{r \in P_R} (\|\mathbf{y_r}\| \setminus \{r\})$.

Figure 13.3 depicts a PC^{2}R net which models a special version of the classic philosophers problem introduced in [21]. Since this net is modelling software, the state machines represent the control flow for each type of philosopher (thread). Tokens in a state machine represent concurrent processes/threads which share the same control flow. At the initial state, every philosopher is thinking, i.e. the unique token in each machine is located at the idle place. Note that, in order to have a concise model, we considered the simplest case in which there exist only two philosophers.

The resources (here: the bowl of spaghetti and two forks) are shared among both philosophers. From a real-world point of view, the resources in this context are not necessarily physical (e.g., a file) but can also be logical (e.g., a semaphore). The fact that resources in software engineering do not always have a physical counterpart is a peculiar characteristic with consequences. In this context, processes do not only consume resources but also can *create* them. A process will destroy the newly created resources before its termination. For instance, a process can create a shared

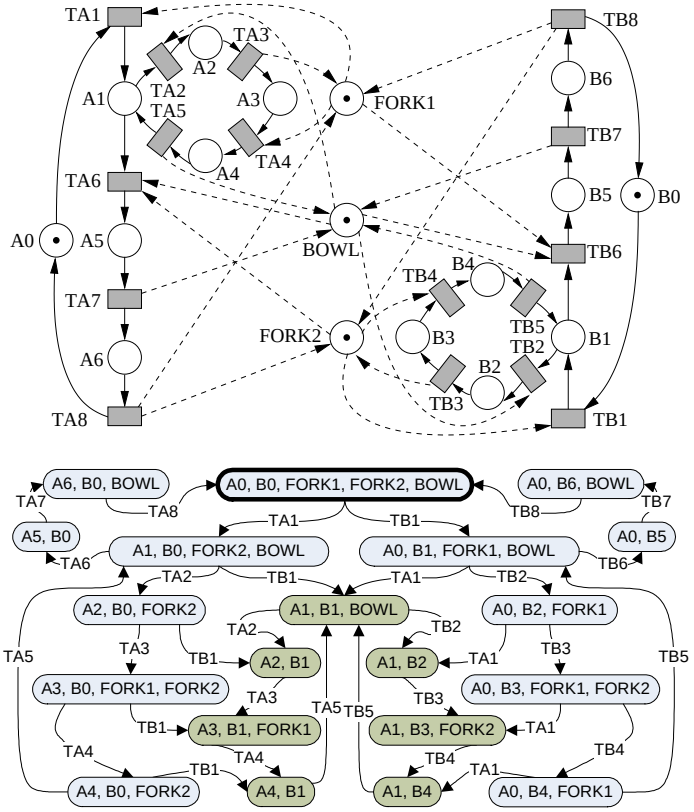

Fig. 13.3 A (non-live) PC^{2}R with an a potentially acceptable initial marking.

memory variable (or a service!) which can be allocated to other processes/threads. Hence the resource allocation scheme is no longer *first-acquire-later-release*, but it can be the other way round too. Still, all the resources are used conservatively by the processes (either by a create-destroy sequence or by a wait-release sequence). As a side effect, and perhaps counterintuitively, there may not be free resources during the system startup (as they still must be created), yet the system is live.

As a result, and unlike S^{4}PR nets, the support of the $\mathbf{y_r}$ p-semiflows (point 4 of Definition 4) may include P_0. For such a resource place r, there exists at least a process which creates (*lends*) instances of r. As a consequence, there might exist additional minimal p-semiflows containing more than one resource place [21].

The next definition is strongly related to the notion of acceptable initial marking introduced for the S^{4}PR class. In software systems all processes/threads are initially inactive and start from the same point (the `begin` statement). Hence, all of the corresponding tokens are in the idle place at the initial marking (the process places being therefore empty). The definition takes this into account and establishes a lower bound for the marking of the resource places.

Definition 5. [21] Let $N = \langle P, T, \mathbf{Pre}, \mathbf{Post} \rangle$ be a PC^{2}R. An initial marking $\mathbf{m_0}$ is potentially acceptable for N iff $\|\mathbf{m_0}\| \setminus P_R = P_0$, and $\forall p \in P_S, r \in P_R : \mathbf{y_r^T} \cdot \mathbf{m_0} \geq \mathbf{y_r}[p]$.

The above definition is syntactically a generalization of the concept of acceptable initial marking for S^{4}PR nets. If the initial marking of some resource place r is lesser than the lower bound established for $\mathbf{m_0}[r]$ by Definition 5, then there exists at least one dead transition at the initial marking.

At this point, it is worth stressing that an S^{4}PR net with an acceptable initial marking cannot have any dead transition at the initial marking (since every minimal t-semiflow is firable in isolation from it). For PC^{2}R nets, however, having a marking which is greater than that lower bound does not guarantee, in the general case, that there do not exist dead transitions, as discussed in Section 13.3. Accordingly, the preceding adverb *potentially* stresses this fact, in contrast to S^{4}PR nets.

Furthermore, according to Definition 5, the initial marking of some resource place r may be empty if some idle place is covered by $\|\mathbf{y_r}\|$, as seen later.

Since the PC^{2}R class generalizes previous classes in the S^{n}PR family, these can be redefined in the new framework as follows.

Definition 6. [21] Previous classes of the S^{n}PR family are defined as follows:

- An S^{5}PR [18] is a PC^{2}R where $\forall r \in P_R : \|\mathbf{y_r}\| \cap P_0 = \emptyset$.
- An S^{4}PR [28] is an S^{5}PR where $\forall i \in I_N$ the subnet generated by $\{p_{0_i}\} \cup P_i, T_i$ is a strongly connected state machine in which every cycle contains p_{0_i} (i.e., a iterative state machine with no internal cycles).
- An S^{3}PR [6] is an S^{4}PR where $\forall p \in P_S : |{}^{\bullet\bullet}p \cap P_R| = 1, ({}^{\bullet\bullet}p \cap P_R = p^{\bullet\bullet} \cap P_R)$.
- An L-S^{3}PR [7] is an S^{3}PR where $\forall p \in P_S : |{}^{\bullet}p| = |p^{\bullet}| = 1$.

Remark 1. L-S^{3}PR $\subseteq$ S^{3}PR $\subseteq$ S^{4}PR $\subseteq$ S^{5}PR $\subseteq$ PC^{2}R.

The preceding remark is straightforward from Definition 6. It is worth remarking that Definition 5 collapses with the definition of acceptable initial markings respectively provided for those subclasses [6, 7, 28, 21]. For all of these, the same properties than for S^{4}PR (i.e., no dead transitions at $\mathbf{m_0}$, non-empty resources) apply.

Finally, there exists another class for S-RASs, called System of Processes Quarrelling over Resources (SPQR) [18], which does not strictly contain or is contained by the PC^{2}R class. Yet, there exist transformation rules to travel between PC^{2}Rs and Structurally Bounded (SB) SPQRs. Note that, by construction, PC^{2}R nets are conservative, and hence SB, but this is not true for general SPQRs. The SPQR class seems interesting from an analytical point of view thanks to its syntactic simplicity, as discussed in [18].

In the following, we will make clear that the approach followed to date does not work with the new net classes for modelling multithreaded control applications (PC^{2}R, SPQR). 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 aforementioned superclasses.

Figure 13.4 summarizes the inclusion relations between the reviewed Petri net classes for S-RASs. The lefter on the x-axis, the more complex the process structure can be (i.e., linear state machines are on the right while general state machines are on the left). The upper on the y-axis, the higher degree of freedom in the way the processes use the resources (resource lending is on top). The figure also illustrates the fact that every model of the S^nPR family can be transformed into a PC^2R net.

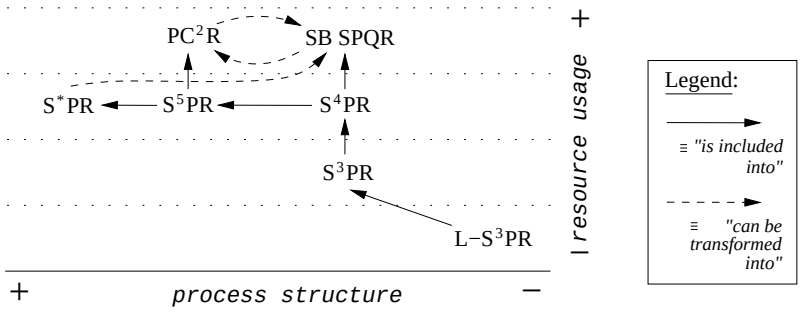

Fig. 13.4 Inclusion relations between Petri net classes for RASs

13.3 Liveness analysis and related properties

Traditionally, empty or insufficiently marked siphons have been a fruitful structural element for characterizing non-live RASs. The more general the net class, however, the more complex the siphon-based characterization is. The following results can be easily obtained from previously published works. The originality here is to point out the strict conditions that the siphons must fulfil.

Theorem 1. *[6, 21] Let $\langle N, \mathbf{m_0}\rangle$ be a marked S^3PR with an acceptable initial marking. $\langle N, \mathbf{m_0}\rangle$ is non-live iff $\exists \mathbf{m} \in \mathrm{RS}(N, \mathbf{m_0})$ and a minimal siphon D: $\mathbf{m}[D] = \mathbf{0}$.*

For instance, the marked S^3PR in Fig. 13.5 is non-live with $K_0 = K_1 = 1$, $K_3 = 2$. From this acceptable initial marking, the marking $(A4 + B4 + R2 + 2 \cdot R3)$ can be reached by firing σ, where $\sigma = TB1\ TA1\ TB2\ TA2\ TB3\ TA3\ TB4\ TA4$. This firing sequence empties the minimal siphon $\{A1, B1, A5, B5, R1, R4\}$.

However, this characterization is sufficient, but not necessary, in general, for S^4PR nets. Hence, the concept of *empty siphon* had to be generalized. Given a marking $\mathbf{m}$ in an S^4PR net, a transition t is said to be $\mathbf{m}$-process-enabled ($\mathbf{m}$-process-disabled) iff it has (not) a marked input process place, and $\mathbf{m}$-resource-enabled ($\mathbf{m}$-resource-disabled) iff its input resource places have (not) enough tokens to fire it, i.e., $\mathbf{m}[P_R, t] \geq \mathbf{Pre}[P_R, t]$ ($\mathbf{m}[P_R, t] \ngeq \mathbf{Pre}[P_R, t]$).

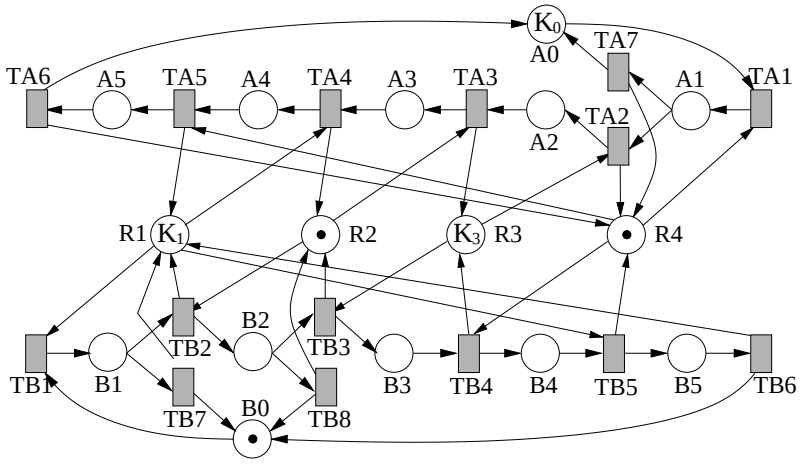

Fig. 13.5 An S^{3}PR which is non-live iff $(K_0 \geq K_1, K_3 \geq 2) \vee (K_0 \cdot K_1 \cdot K_3 = 0)$. Note that $\mathbf{m_0}$ is an acceptable initial marking iff $(K_0 \cdot K_1 \cdot K_3 \neq 0)$

Theorem 2. *[28] Let $\langle N, \mathbf{m_0} \rangle$ be a marked S^{4}PR with an acceptable initial marking. $\langle N, \mathbf{m_0} \rangle$ is non-live iff $\exists \mathbf{m} \in \mathrm{RS}(N, \mathbf{m_0})$ and a siphon D such that:* i) *There exists at least one* **m***-process-enabled transition;* ii) *Every* **m***-process-enabled transition is* **m***-resource-disabled by some resource place in D;* iii) *All process places in D are empty at* **m***.*

Such a siphon D is said to be insufficiently marked at $\mathbf{m}$ (also: *bad* siphon). In Theorems 1 and 2, the siphon captures the concept of circular wait, revealing it from the underlying net structure. In contrast to the S^{3}PR class, it is worth noting the following fact about *minimal* siphons in S^{4}PR nets, which emerges because of their minimal p-semiflows not being strictly binary.

Property 1 *[21] There exists a S^{4}PR net with an acceptable initial marking which is non-live but every siphon characterizing the non-liveness is non-minimal, i.e., minimal siphons are insufficient to characterize non-liveness.*

For instance, the S^{4}PR net in Fig. 13.1 is non-live, but there is no minimal siphon containing both resource places $R1$ and $R2$. Note that the siphon $D = \{R1, R2, A3, B2\}$ becomes insufficiently marked at $\mathbf{m}$, where $\mathbf{m} = A1 + B1 + R1 + R2$, but it contains the minimal siphon $D' = \{R2, A3, B2\}$. D' is not insufficiently marked for any reachable marking. It is also worth noting that no siphon is ever emptied.

Thus non-minimal siphons must be considered in order to deal with deadlocks in systems more complex than S^{3}PR.

On the other hand, insufficiently marked siphons (even considering those non-minimal) are not enough for characterizing liveness for more complex systems such as S^{5}PR models. This means that siphon-based control techniques for RASs do not work in general for concurrent software, even in the 'good' case in which every `wait`-like operation precedes its complementary `signal`-like operation.

Property 2 *[21] There exists an S^{5}PR with an acceptable initial marking $\langle N, \mathbf{m_0}\rangle$ which is non-live but insufficiently marked siphons do not characterize non-liveness (dead markings).*

The S^{5}PR net in Fig. 13.3 evidences the claim stated above. The figure depicts a non-live system with three possibly bad siphons. These siphons are $D_1 = \{A2, A3, A4, A5, A6, B2, B4, B5, B6, FORK2, BOWL\}$, $D_2 = \{A2, A4, A5, A6, B2, B3, B4, B5, B6, FORK1, BOWL\}$ and $D_3 = \{A2, A4, A5, A6, B2, B4, B5, B6, FORK1, FORK2, BOWL\}$. Besides, every transition in the set $\Omega = \{TA2, TA3, TA4, TA5, TB2, TB3, TB4, TB5\}$ is an output transition of D_1, D_2 and D_3. After firing transitions *TA1* and *TA2* starting from $\mathbf{m_0}$, the state $A1 + B1 + BOWL$ is reached. This marking belongs to a livelock with other six markings. The reader can check that there exists a firable transition in Ω for every marking in the livelock, and in any case there is no insufficiently marked siphon.

In other words, for every reachable marking in the livelock, there exist output transitions of the siphons which are firable. As a result, the siphon-based non-liveness characterization for earlier net classes (such as S^{4}PR [28]) is not sufficient in the new framework.

This is an a priori unexpected result since these nets fulfil some strong structural properties which also S^{4}PR nets hold. PC^{2}R nets are well-formed, i.e. SB and Structurally Live (SL), and therefore also conservative and consistent [21]. Structural boundedness is straightforward, since PC^{2}R nets are conservative by construction (every place is covered by at least one minimal p-semiflow), and a well-known general result of Petri nets is that conservativeness implies structural boundedness [26]. Structural liveness is derived from the fact that every resource place is a structurally implicit place, and therefore its initial marking can be increased enough so as to make it implicit. When every resource place is implicit, the net system behaves like a set of isolated and marked strongly-connected state machines, and therefore the system is live. Obviously, all the subclasses (S^{3}PR, S^{4}PR, etc.) are also well-formed. However, SPQR nets are not necessarily SL [18].

By carefully observing the net in Fig. 13.3, it might seem that the difficulty in finding a liveness characterization for PC^{2}R nets lies in the appearance of certain types of livelocks. In general, livelocks with dead transitions are not a new phenomenon in the context of Petri net models for RASs. Fig. 13.6 shows that, even for L-S^{3}PR nets, deadlock-freeness does not imply liveness.

Property 3 *[7] There exists a marked L-S^{3}PR with an acceptable initial marking such that it is deadlock-free but not live.*

This L-S^{3}PR net system has no deadlock but two reachable livelocks: (i) $\{(A0 + B2 + C0 + D1 + R1), (A1 + B2 + C0 + D1)\}$, and (ii) $\{(A0 + B1 + C0 + D2 + R3), (A0 + B1 + C1 + D2)\}$. Nevertheless, these livelocks are captured by insufficiently marked siphons. Unfortunately, this no longer holds for some kind of livelocks in S^{5}PR or more complex systems. Indeed, PC^{2}R nets feature some complex properties which complicate the finding of a liveness characterization.

Another relevant property for studying liveness is its monotonicity. In this sense, the negative results emerge further back than probably expected.

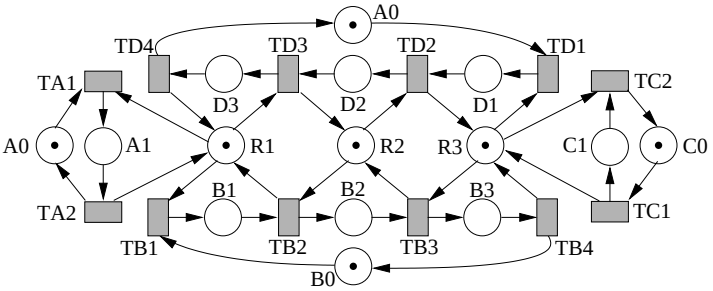

Fig. 13.6 A non-live L-S^{3}PR which is deadlock-free.

Property 4 *[21] There exists an S^{3}PR such that liveness is not monotonic, neither with respect to the marking of the idle/process places, nor that of the resource places, i.e., liveness is not always preserved when those are increased.*

The net depicted in Fig. 13.5 illustrates this fact:

- With respect to P_R: The system is live with $K_0 = K_1 = K_3 = 1$ and non-live with $K_0 = K_1 = 1$, $K_3 = 2$ (however, it becomes live again if the marking of $R1$, $R2$ and $R4$ is increased enough so as to make every resource place an implicit place).
- With respect to P_0: The system is live with $K_0 = 1$, $K_1 = K_3 = 2$ and non-live with $K_0 = K_1 = K_3 = 2$.

Note that liveness is monotonic for every net belonging to the L-S^{3}PR class [7] with respect to the resource places. But, from S^{3}PR nets upwards, there is a discontinuity zone between the point where the resource places are empty enough so that every transition is dead (also held for lower markings), and the point where every resource place is implicit (liveness is preserved if their marking is increased). Markings within these bounds fluctuate between liveness and non-liveness. The location of those points also depends on the marking of the idle/process places: the more tokens in them, the farther the saturation point.

Nevertheless, an interesting property of S^{4}PR nets is that liveness equals reversibility. This, along with the fact that the idle place does not belong to any p-semiflow $\mathbf{y_r}$, is a powerful feature. If every token in a process net can be moved to the idle place, then the net is not dead (yet).

Theorem 3. *[21] Let $\langle N, \mathbf{m_0} \rangle$ be an S^{4}PR with an acceptable initial marking. $\langle N, \mathbf{m_0} \rangle$ is live iff $\mathbf{m_0}$ is a home state (i.e., the system is reversible).*

However, Theorem 3 is false in general for S^{5}PR nets. In fact, the directedness property [1] does not even hold: a live S^{5}PR may have no home state.

Property 5 *[21] There exists an S^{5}PR with an acceptable initial marking $\langle N, \mathbf{m_0} \rangle$ such that the system is live but there is no home state.*

The net system in Fig. 13.7 has no home state in spite of being live. It is worth noting that this net is ordinary. Yet S^{5}PR nets still retain an interesting property: its minimal t-semiflows are eventually realizable from an acceptable initial marking.

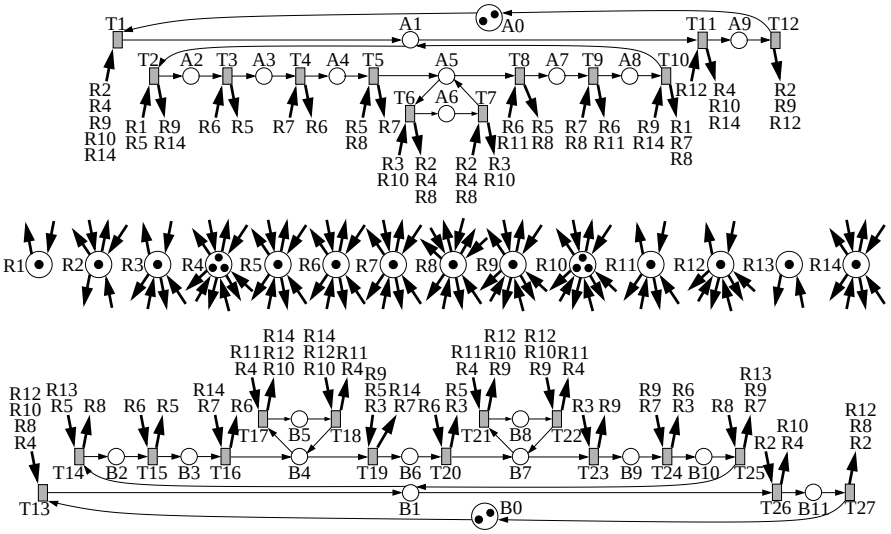

Fig. 13.7 A live S^{5}PR which has no home state. The arcs from/to P_R are omitted for clarity. Instead, the set of input and output resource places are listed next to each transition.

Theorem 4. *[21] Let $\langle N, \mathbf{m_0}\rangle$ be an S^{5}PR with an acceptable initial marking. For every (minimal) t-semiflow* $\mathbf{x}$*, there exists a reachable marking* $\mathbf{m} \in \mathrm{RS}(N, \mathbf{m_0})$ *such that* $\mathbf{x}$ *is realizable from* $\mathbf{m}$*, i.e.* $\exists \sigma$ *such that* $\mathbf{m} \xrightarrow{\sigma}$*,* $\boldsymbol{\sigma} = \mathbf{x}$*, where* $\boldsymbol{\sigma}$ *is the firing count vector of* σ*.*

However, for PC^{2}R nets there may not exist minimal t-semiflows being eventually realizable; even for live systems.

Property 6 *[21] There exists a PC^{2}R with a potentially acceptable initial marking $\langle N, \mathbf{m_0}\rangle$ such that the system is live and there exists a minimal t-semiflow* $\mathbf{x}$ *such that* $\forall \mathbf{m} \in \mathrm{RS}(N, \mathbf{m_0})$*,* $\nexists \sigma$ *such that* $\mathbf{m} \xrightarrow{\sigma}$ *and* $\boldsymbol{\sigma} = \mathbf{x}$*, i.e.* $\mathbf{x}$ *is not realizable from* $\mathbf{m}$*.*

The reader can check that the PC^{2}R net system in Fig. 13.8 has no home state in spite of being live. Depending on which transition is fired first (either $T1$ or $T8$) a different livelock is reached. Besides, for every reachable marking, there is no minimal t-semiflow such that it is realizable, i.e. firable in isolation. Instead, both state machines need each other to progress from the very beginning.

Counterintuitively, the impossibility of realizing every t-semiflow in a live PC^{2}R net cannot be directly linked to the system reversibility. The net system in Fig. 13.8 has no home state. However, the net system in Fig. 13.9 is reversible, live, but no minimal t-semiflow is realizable. In fact, these two properties (reversibility and t-semiflow realizability) are usually strongly linked to the property of liveness for many Petri net classes. Particularly, reversibility is powerful since its fulfilment implies that the net is live iff there are no dead transitions at the initial marking. Both properties together imply that the net is live, as the next theorem states.

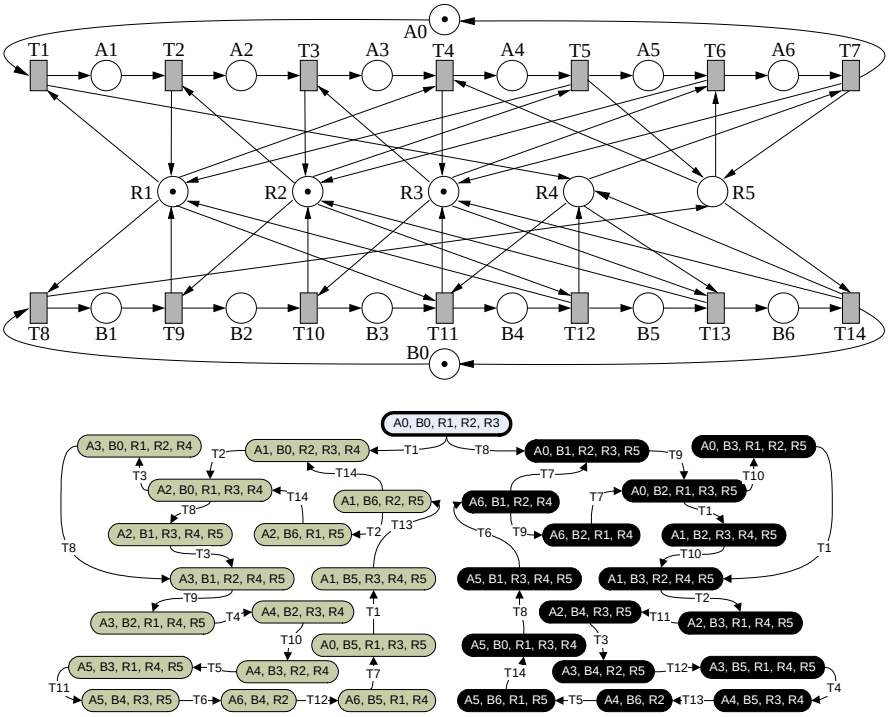

Fig. 13.8 A live PC^{2}R for which no minimal t-semiflow is ever realizable.

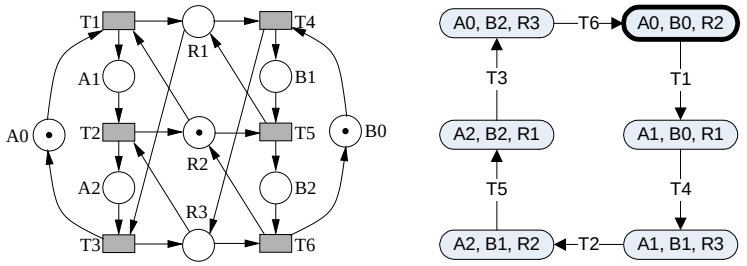

Fig. 13.9 A live and reversible PC^{2}R for which no minimal t-semiflow is ever realizable.

Theorem 5. *[17] Let $\langle N, \mathbf{m_0} \rangle$ be an PC^{2}R with a potentially acceptable initial marking. If the net system is reversible and every (minimal) t-semiflow* $\mathbf{x}$ *is eventually realizable (i.e., there exist a reachable marking $\mathbf{m} \in \mathrm{RS}(N, \mathbf{m_0})$ and a firing sequence σ such that $m \xrightarrow{\sigma}$, $\boldsymbol{\sigma} = \mathbf{x}$) then the net is live.*

Table 13.1 illustrates in a concise way the relation between those three properties (liveness, reversibility, and eventual firability of all t-semiflows) in the context of general PC^{2}R nets with potentially acceptable initial markings. The table highlights the fact that those properties are not totally independent because of PC^{2}R nets being

consistent, as proved by Theorem 5. It also reveals that the simpler the subclass, the less combinations of the three properties are possible (up to the point that liveness equals reversibility for S^4PR and simpler subclasses). Figures 13.10 and 13.11, which have not been introduced before, are used to complete the table.

$\boxed{\mathrm{L\,R\,T}}$ From L-S^3PR upwards Fig. 13.5 with $K_0 = K_1 = K_3 = 1$	$\boxed{\mathrm{L\,R\,\overline{T}}}$ PC^2R only Fig. 13.9	$\boxed{\mathrm{L\,\overline{R}\,T}}$ From S^5PR upwards Figs. 13.7
$\boxed{\mathrm{L\,\overline{R}\,\overline{T}}}$ PC^2R only Fig. 13.8	$\boxed{\mathrm{\overline{L}\,R\,T}}$ IMPOSSIBLE Theorem 5	$\boxed{\mathrm{\overline{L}\,R\,\overline{T}}}$ PC^2R only Fig. 13.10
	$\boxed{\mathrm{\overline{L}\,\overline{R}\,T}}$ From L-S^3PR upwards Fig. 13.5 with $K_0 = K_1 = 1$ and $K_3 = 2$	$\boxed{\mathrm{\overline{L}\,\overline{R}\,\overline{T}}}$ PC^2R only Fig. 13.11

Table 13.1 Summary of the relationship between liveness (L), reversibility (R), and eventual realizability of every t-semiflow (T) for PC^2R nets with a potentially acceptable initial marking. For each cell, the first line indicates which (sub)class such combination of properties is possible from. The second line references a proof of such behaviour.

13.4 Structure-based synthesis methods: An iterative control policy

In Section 13.3, we have seen that there exists a structural characterization of the deadlock problem for Petri net classes usually explored in the context of FMSs, i.e., S^4PR nets and subclasses. Unfortunately, only necessary or sufficient siphon-based conditions have been found for more complex superclasses such as PC^2R or SPQR, in the context of static analysis of multithreaded control software.

In the present section, we review an algorithm for computing the system control for S^4PR nets [28] which is based in that structural characterization and is successfully deployed in the context of FMSs. With the help of the net state equation, a set of Integer Linear Programming Problems (ILPPs) is constructed which prevents the costly exploration of the state space. The foundation for such approach relies on bad siphons fully capturing non-live behaviours. Since bad siphons do no longer chacterize non-liveness for models beyond the S^4PR frontier (Property 2) this also delimits the applicability of such kind of structural techniques in more complex scenarios.

Prior to the introduction of the algorithm, some basic notation must be settled.

In the following, for a given insufficiently marked siphon D, $D_R = D \cap P_R$ and $\mathbf{y_{D_R}} = \sum_{r \in D_R} \mathbf{y_r}$. Notice that $\mathbf{y_{D_R}}$ is the total amount of resource units belonging to D (in fact, to D_R) used by each active process in their process places. Also:

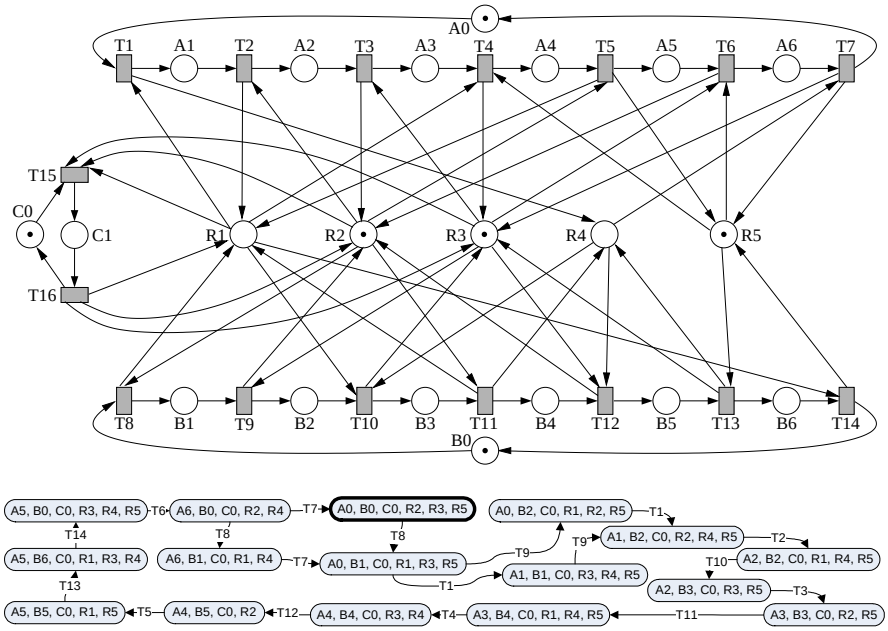

Fig. 13.10 A non-live but reversible PC^{2}R with no realizable minimal t-semiflow. It is worth noting that transitions $T15$ and $T16$ are dead at $\mathbf{m_0}$, despite being a *potentially* acceptable initial marking.

Definition 7. Let $\langle N, \mathbf{m_0} \rangle$ be a marked S^{4}PR. Let D be a siphon of N. Then, $\mathscr{T}h_D = \|\mathbf{y_{D_R}}\| \setminus D$ is the *set of thieves* of D, i.e. the set of process places of the net that use resources of the siphon and do not belong to that siphon.

In each iteration, the algorithm searches for a bad siphon. If found, a control place is suggested to prevent that siphon from ever becoming insufficiently marked. Such control place will be a virtual resource, in such a way that the resulting Petri net remains into the S^{4}PR class. Thanks to this, a new iteration of the algorithm can be executed. The algorithm terminates as soon as there do not exist more siphons to be controlled, i.e., the system is live.

The next system of restrictions relates the liveness characterization introduced in Theorem 2 with the ILPPs which are used in the forthcoming algorithm. Essentially, the structural characterization is reformulated into a set of linear restrictions given a reachable marking and a related bad siphon. It is worth noting that, in order to compact the system of linear restrictions, three sets of variables have been parameterized: v_p, e_t and e_{rt} (the latter being doubly parameterized, both by resource places r and transitions t). Obviously, the ILPP would have appeared considerably larger if it had not been compacted in this way for the sake of concision.

Proposition 1. *[28] Let $\langle N, \mathbf{m_0} \rangle$ be a marked S^{4}PR. The net is non-live if and only if there exist a siphon D and a marking $\mathbf{m} \in \mathrm{RS}(N, \mathbf{m_0})$ such that the following set of inequalities has, at least, one solution ($D = \{p \in P_S \cup P_R \mid v_p = 0\}$):*

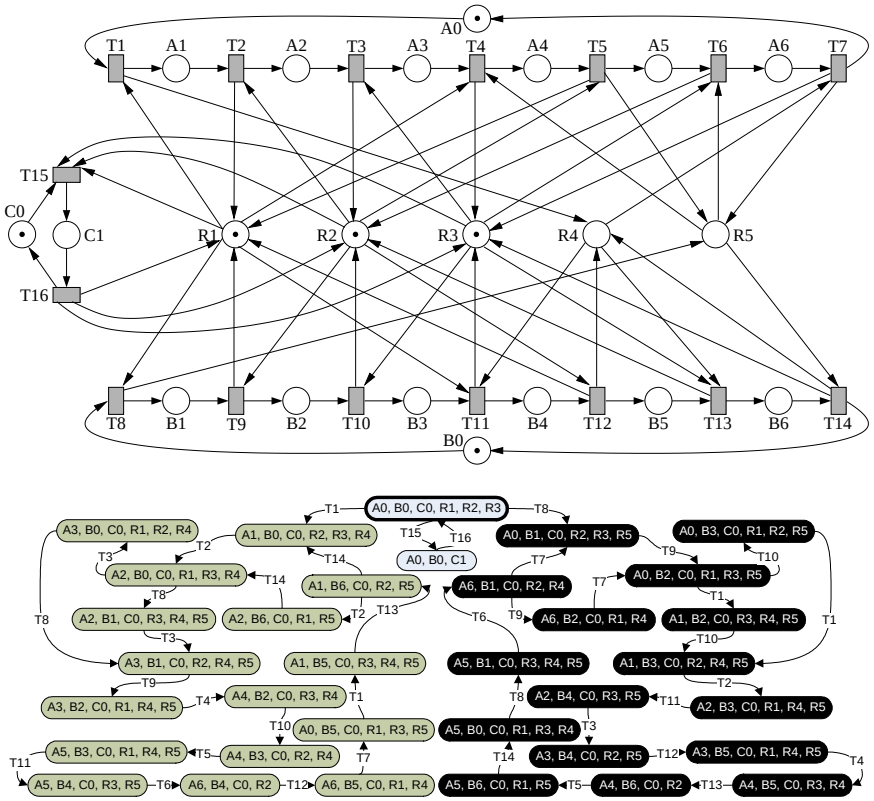

Fig. 13.11 A non-live, non-reversible PC^{2}R with no realizable minimal t-semiflow.

$$
\left\{
\begin{array}{lll}
\mathbf{m}[P_S] \geq \mathbf{0} \wedge \mathbf{m}[P_S] \neq \mathbf{0} & & \\
\forall t \in T \setminus P_0^\bullet : \mathit{being}\{p\} = {}^\bullet t \cap P_S, & & \\
\qquad \mathbf{m}[p] \geq e_t & & \\
\qquad e_t \geq \frac{\mathbf{m}[p]}{sb[p]} & & \\
\forall r \in P_R : \quad \forall t \in r^\bullet \setminus P_0^\bullet : & \frac{\mathbf{m}[r]}{\mathbf{Pre}[r,t]} + v_r \geq e_{rt} & \\
& e_{rt} \geq \frac{\mathbf{m}[r] - \mathbf{Pre}[r,t] + 1}{\mathbf{m_0}[r] - \mathbf{Pre}[r,t] + 1} & \\
& e_{rt} \geq v_r & \\
\quad \forall t \in T \setminus P_0^\bullet : & \sum_{r \in {}^\bullet t \cap P_R} e_{rt} < |{}^\bullet t \cap P_R| + 1 - e_t & \\
\quad \forall p \in P \setminus P_0 : & v_p \in \{0,1\} & \\
\quad \forall t \in T \setminus P_0^\bullet : & e_t \in \{0,1\} & \\
\quad \forall t \in r^\bullet \setminus P_0^\bullet : & e_{rt} \in \{0,1\}. &
\end{array}
\right\} \tag{13.1}
$$

where $sb[p]$ denotes the structural bound[2] *of p [5].*

[2] sb[p] is the max. of the following ILPP: $sb[p] = \max \mathbf{m}[p]$ s.t. $\mathbf{m} = \mathbf{m_0} + \mathbf{C} \cdot \boldsymbol{\sigma}, \mathbf{m} \geq \mathbf{0}, \boldsymbol{\sigma} \in \mathbb{N}^{|T|}$

Thanks to the addition of the net state equation as another linear restriction, the following theorem constructs an ILPP which can compute a marking and a bad siphon holding System (13.1). Nevertheless, that marking can be a spurious solution of the state equation. Since this kind of nets can have killing spurious solutions (i.e, spurious solutions which are non-live when the original net system is live) then the theorem establishes a necessary but not sufficient condition. This is usually not a problem when the objective is to obtain a live system: the only consequence can be that some harmless, unnecessary control places are added. These control places would forbid some markings which are not really reachable.

Since one siphon must be selected, the ILPP selects that with a minimal number of places, hoping that controlling the smallest siphons first may prevent controlling the bigger ones. Other works present analogous techniques with a different objective function for this ILPP [12].

Theorem 6. *[28] Let $\langle N, \mathbf{m_0}\rangle$ be a marked S^4PR. If the net is nonlive, then there exist a siphon D and a marking $\mathbf{m} \in \mathrm{PRS}(N, \mathbf{m_0})$ such that the following set of inequalities has, at least, one solution with $D = \{p \in P_S \cup P_R \mid v_p = 0\}$:*

$$
\begin{array}{ll}
\max & \sum_{p\in P\setminus P_0} v_p \\
\textit{s.t.} & \mathbf{m} = \mathbf{m_0} + \mathbf{C}\cdot\boldsymbol{\sigma} \\
& \mathbf{m} \geq \mathbf{0}, \boldsymbol{\sigma} \in \mathbb{N}^{|T|} \\
& \forall p \in P\setminus P_0, \forall t \in {}^\bullet p : v_p \geq \sum_{q\in p^\bullet t} v_q - |{}^\bullet t| + 1 \\
& \sum_{p\in P\setminus P_0} v_p < |P\setminus P_0| \\
& \text{System (13.1)}
\end{array}
$$

The previous theorem can compute a marking $\mathbf{m}$ and a related bad siphon D. However, siphon D can be related with a high number of deadlocks, and not only with that represented with $\mathbf{m}$. For that reason, the aim is to compute a control place able to cut every *unwanted* marking which the siphon D is related to. Consequently, two different strategies are raised from the observation of the set of unwanted markings: (i) adding a place that introduces a lower bound of the number of available resources in the siphon for every reachable marking (*D-resource-place*), or (ii) adding a place that introduces an upper bound of the number of active processes which are retaining tokens from the siphon (*D-control-place*).

In order to define the initial marking of such places, two constants must be computed which are the result of two ILPPs. These ILPPs evaluate every unwanted marking that a bad siphon is related to:

Definition 8. [28] Let $\langle N, \mathbf{m_0}\rangle$ be a marked $\mathrm{S}^4\mathrm{PR}$. Let D be an insufficiently marked siphon, m_D^{max} and m_D^{min} are defined as follows, with $v_p = 0$ iff $p \in D$:

$$
\begin{array}{ll}
m_D^{max} = \max & \sum_{r\in D_R} \mathbf{m}[r] \\
\text{s.t.} & \mathbf{m} = \mathbf{m_0} + \mathbf{C}\cdot\boldsymbol{\sigma} \\
& \mathbf{m} \geq \mathbf{0}, \boldsymbol{\sigma} \in \mathbb{N}^{|T|} \\
& \mathbf{m}[P_S \setminus \mathscr{T}h_D] = \mathbf{0} \\
& \text{System (13.1)}
\end{array}
\qquad
\begin{array}{ll}
m_D^{min} = \min & \sum_{p\in \mathscr{T}h_D} \mathbf{m}[p] \\
\text{s.t.} & \mathbf{m} = \mathbf{m_0} + \mathbf{C}\cdot\boldsymbol{\sigma} \\
& \mathbf{m} \geq \mathbf{0}, \boldsymbol{\sigma} \in \mathbb{N}^{|T|} \\
& \mathbf{m}[P_S \setminus \mathscr{T}h_D] = \mathbf{0} \\
& \text{System (13.1)}
\end{array}
$$

The next definition establishes the connectivity and the initial marking of the control place proposed for a given bad siphon D, both whether that place is a D-process-place or a D-resource place.

Definition 9. [28] Let $\langle N, \mathbf{m_0} \rangle$ be a non-live marked S^{4}PR. Let D be an insufficiently marked siphon, and m_D^{max} and m_D^{min} as in Definition 8. Then, the associated D-resource-place, p_D, is defined by means of the addition of the following incidence matrix row and initial marking: $\mathbf{C^{p_D}}[p_D, T] = -\sum_{p \in \mathscr{I}h_D} \mathbf{y_{D_R}}[p] \cdot \mathbf{C}[p, T]$, and $\mathbf{m_0^{p_D}}[p_D] = \mathbf{m_0}[D] - (m_D^{max} + 1)$. The associated D-process-place, p_D, is defined by means of the addition of the following incidence matrix row and initial marking: $\mathbf{C^{p_D}}[p_D, T] = -\sum_{p \in \mathscr{I}h_D} \mathbf{C}[p, T]$, and $\mathbf{m_0^{p_D}}[p_D] = m_D^{min} - 1$.

Finally, we can enunciate the algorithm that computes the control places for a given S^{4}PR net. In those cases in which a D-resource-place with an admissible initial marking cannot be computed, the algorithm proposes the corresponding D-process-place, which always has an admissible initial marking [28].

Algorithm 1 [28] Synthesis of live S^{4}PR net systems

1. Compute an insufficiently marked siphon using the ILPP of Theorem 6.
2. Compute m_D^{max} (Definition 8).
   a. If the associated D-resource-place (Definition 9) has an acceptable initial marking according to Definition 2, then let p_D be that place, and go to step 3.
   b. Else, compute m_D^{min} (Definition 8). Let p_D be the associated D-process-place (Definition 9).
3. Add the control place p_D.
4. Go to step 1, taking as input the partially controlled system, until no insufficiently marked siphons exist.

Theorem 7. *[28] Let $\langle N, \mathbf{m_0} \rangle$ be a marked S^{4}PR. Algorithm 1 applied to $\langle N, \mathbf{m_0} \rangle$ terminates. The resulting controlled system, $\langle N^C, \mathbf{m_0^C} \rangle$, is a live marked S^{4}PR such that* $\mathrm{RS}(N^C, \mathbf{m_0^C}) \subseteq \mathrm{RS}(N, \mathbf{m_0})$.

Let us apply Algorithm 1 to the net depicted in figure 13.1. There exists one deadlock ($\mathbf{m} = A1 + B1 + R1 + R2$) and two insufficiently marked siphons at $\mathbf{m}$, $D_1 = \{R1, R2, A3, B2\}$ and $D_2 = D_1 \cup \{A2\}$. None of these is minimal. When applied step 1 of Algorithm 1, the ILPP of Theorem 6 returns D_1. In step 2, we compute $m_D^{max} = 2$. Since the associated D-resource-place has not an acceptable initial marking, then we compute $m_D^{min} = 2$. In step 3, we add the associated D-process-place p_D to the net. And finally, we go back to step 1. But now the net is live and the ILPP of Theorem 6 has no solution, so the algorithm terminates. The resulting controlled system is depicted in figure 13.12.

Let us now apply the algorithm to the S^{3}PR net depicted in figure 13.5 with $K_0 = K_1 = 1$, $K_3 = 2$. In this case, there exists one deadlock ($\mathbf{m} = A4 + B4 + R2 + 2 \cdot R3$) which is reachable by firing the sequence $\sigma = TB1\ TA1\ TB2\ TA2\ TB3\ TA3\ TB4\ TA4$.

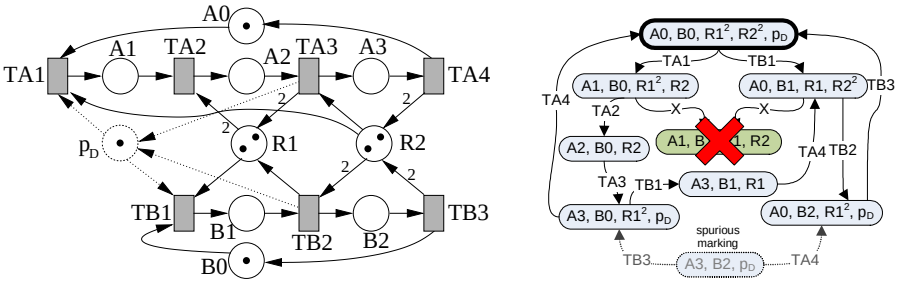

Fig. 13.12 The controlled system after applying Algorithm 1 on the net depicted in figure 13.1.

This sequence empties the minimal siphon $D = \{A1, B1, A5, B5, R1, R4\}$, which is also the siphon returned by the computation in step 1 of the algorithm. In step 2, we obtain $m_D^{max} = 0$. Then the associated D-resource-place p_D has an acceptable initial marking ($\mathbf{m_0^{p_D}} = 1$), with $\mathbf{y_r} \in \{0,1\}^{|P|}$ and $\|\mathbf{y_r}\| = \{A4, B4\}$. Therefore, p_D can be aggregated to the net (step 3), and we go back to step 1. Since the resulting net is live and the ILPP of Theorem 6 has no solution, the algorithm terminates.

Further readings

The material of this chapter is essentially related to structural techniques for dealing with deadlocks in S-RAS with online routing decisions. The reader can find some reference works on the family of Petri net models for tackling this kind of systems in the following references: [6, 7, 27, 28, 22, 9, 21]

From a modelling point of view, the problem of integrating assembly/dissassembly operations has been approached in works such as [30, 13, 9]. However, structural liveness enforcing approaches can be computationally demanding in this scenario, as evidenced for augmented marked graphs in [3]. An insight on the computational complexity of such approaches on the S-RAS context is driven in [19].

Most works on modelling RAS by way of Petri nets are focused on FMSs. The article [4] can serve as a succinct introduction to the discipline. The books [16, 24] also focus on the RAP from this perspective. Besides, the latter book also features some approximation to Automated Guided Vehicle (AGV) Transportation Systems from the point of view of RAS modelling through Petri nets. Recently, AGVs have been comprehensively tackled in [25]. Other application domains in which similar methodological approaches have been deployed include multiprocessor interconnection networks [25] and multithreaded software [29, 20, 21].

The most significant proliferation of works can be found in the context of synthesis. Most of this papers are related to siphon computation (two recent works on this issue can be found in [2, 15]) as well as to applying Mixed Integer Programming to liveness enforcing [28, 12]. Another family of works focuses on synthesis based on reachability state analysis and on the theory of regions [11, 23].

References

1. E. Best and K. Voss. Free choice systems have home states. *Acta Informatica 21*, pages 89–100, 1984.
2. E-E. Cano, C-A. Rovetto, and J-M. Colom. On the computation of the minimal siphons of S^4PR nets from a generating family of siphons. In *15th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA'2010)*, Bilbao, Spain, September 2010. IEEE.
3. F. Chu and X. Xie. Deadlock analysis of Petri nets using siphons and mathematical programming. *IEEE Transactions on Robotics and Automation*, 13(6):793–804, December 1997.
4. J-M. Colom. The resource allocation problem in flexible manufacturing systems. In van der Aalst, W-M-P. and Best, E., editor, *Proc. of the 24th Int. Conf. on Applications and Theory of Petri Nets*, volume 2679 of *LNCS*, pages 23–35, Eindhoven, Netherlands, June 2003. Springer–Verlag.
5. J-M. Colom and M. Silva. Improving the linearly based characterization of P/T nets. *Advances in Petri Nets 1990*, 483:113–145, 1991.
6. J. Ezpeleta, J-M. Colom, and J. Martínez. A Petri net based deadlock prevention policy for flexible manufacturing systems. *IEEE Transactions on Robotics and Automation*, 11(2):173–184, April 1995.
7. J. Ezpeleta, F. García-Valles, and J-M. Colom. A class of well structured Petri nets for flexible manufacturing systems. In J. Desel and M. Silva, editors, *Proc. of the 19th Int. Conf. on Application and Theory of Petri Nets*, volume 1420 of *LNCS*, pages 65–83, Lisbon, Portugal, June 1998. Springer–Verlag.
8. 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*, 34(1), January 2004.
9. J. Ezpeleta, F. Tricas, F. García-Vallés, and J-M. Colom. A banker's solution for deadlock avoidance in FMS with flexible routing and multiresource states. *IEEE Transactions on Robotics and Automation*, 18(4):621–625, August 2002.
10. 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*, 13(3):347–363, 1997.
11. A. Ghaffari, N. Rezg, and X. Xie. Design of a live and maximally permissive Petri net controller using the theory of regions. *IEEE Transactions on Robotics*, 19(1):137–141, 2003.
12. H. Hu, M.C. Zhou, and Z.W. Li. Supervisor optimization for deadlock resolution in automated manufacturing systems with Petri nets. *IEEE Transactions on Automation Science and Engineering*, 8(4):794–804, October 2011.
13. M-D. Jeng, X-L. Xie, and M-Y. Peng. Process nets with resources for manufacturing modeling and their analysis. *IEEE Transactions on Robotics*, 18(6):875–889, 2002.
14. K. Lautenbach and P.S. Thiagarajan. Analysis of a resource allocation problem using Petri nets. In Syre, J.C., editor, *Proc. of the 1st European Conf. on Parallel and Distributed Processing*, pages 260–266, Toulouse, 1979. Cepadues Editions.
15. Z-W. Li and M-C. Zhou. Control of elementary and dependent siphons in Petri nets and their application. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 38(1):133–148, January 2008.
16. Z-W Li and M-C Zhou. *Deadlock Resolution in Automated Manufacturing Systems: A Novel Petri Net Approach*. Springer Publishing Company, Incorporated, 1st edition, 2009.
17. J-P. López-Grao. *Contributions to the deadlock problem in multithreaded software applications observed as Resource Allocation Systems*. PhD thesis, University of Zaragoza, Zaragoza, 2012.
18. J-P. López-Grao and J-M. Colom. Lender processes competing for shared resources: Beyond the S^4PR paradigm. In *Proc. of the 2006 Int. Conf. on Systems, Man and Cybernetics*, pages 3052–3059. IEEE, October 2006.

19. J-P. López-Grao and J-M. Colom. Resource Allocation Systems: Some complexity results on the S^4PR class. In E. Najm, J-F. Pradat-Peyre, and V-V. Donzeau-Gouge, editors, *Formal Techniques for Networked and Distributed Systems - FORTE 2006*, volume 4229 of *LNCS*, pages 324–339. Springer, 2006.
20. J-P. López-Grao and J-M. Colom. On the deadlock analysis of multithreaded control software. In *Proceedings of the 16th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA'2011)*, Toulouse, France, September 2011. IEEE.
21. J-P. López-Grao and J-M. Colom. A Petri net perspective on the Resource Allocation Problem in software engineering. *Transactions On Petri Nets and Other models of Concurrency V (ToPNoC V)*, 6900:181–200, 2012.
22. 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*, 46(10):1572–1583, 2001.
23. L. Piroddi, R. Cordone, and I. Fumagalli. Combined siphon and marking generation for deadlock prevention in Petri nets. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 39(3):650–661, May 2009.
24. S.A. Reveliotis. *Real-Time Management of Resource Allocation Systems: A Discrete Event Systems Approach*. International Series in Operations Research & Management Science. Springer, December 2004.
25. C-A. Rovetto. *Métodos basados en Redes de Petri para el diseño de algoritmos de encaminamiento adaptativos mínimos libres de bloqueos*. PhD thesis, University of Zaragoza, Zaragoza, September 2011.
26. M. Silva and J-M. Colom. On the computation of structural synchronic invariants in P/T nets. In G. Rozenberg, editor, *Advances in Petri Nets 1988*, volume 340, pages 386–417. Springer–Verlag, Berlin, 1988.
27. F. Tricas. *Deadlock analysis, prevention and avoidance in sequential resource allocation systems*. PhD thesis, University of Zaragoza, Zaragoza, May 2003.
28. F. Tricas, F. García-Valles, J-M. Colom, and J. Ezpeleta. A Petri net structure-based deadlock prevention solution for sequential resource allocation systems. In *Proc. of the 2005 Int. Conf. on Robotics and Automation (ICRA)*, pages 272–278, Barcelona, Spain, April 2005. IEEE.
29. Y. Wang, H. Liao, S.A. Reveliotis, T. Kelly, S. Mahlke, and S. Lafortune. Gadara nets: Modeling and analyzing lock allocation for deadlock avoidance in multithreaded software. In *Proc. of the 49th IEEE Conf. on Decision and Control*, pages 4971–4976, Atlanta, Georgia, USA, December 2009. IEEE.
30. X. Xie and M-D. Jeng. ERCN-merged nets and their analysis using siphons. *IEEE Transactions on Robotics and Automation*, 29(4):692–703, 1999.