A Correction and Some Comments on the Article "Polynomially Complex Synthesis of Distributed Supervisors for Large-Scale AMSs Using Petri Nets"

Spyros Reveliotis

Abstract—**The main purpose of this correspondence is to point out the fallacy of Theorem 1 in the paper that is mentioned in the title, a result that attempts to provide a structural characterization for the liveness of the Petri net class considered in the manuscript. The closing part of the correspondence also takes this opportunity to make some further remarks on the results that are claimed in the paper.**

Index Terms—**Resource allocation systems; deadlock avoidance; petri net structural analysis.**

I. Introduction

The paper "Polynomially complex synthesis of distributed supervisors for large-scale AMSs using Petri nets," that appeared in the September 2016 issue of the IEEE Transactions of Control Systems Technology, addresses the problem of deadlock avoidance arising in the context of the contemporary automated manufacturing systems (AMSs), and it does so by introducing a new class of Petri nets (PNs) to model the operations of these environments. This new PN class is named "feedback system of sequential systems with shared resources," a name that is abbreviated to FS^4R in that paper and in the following. Furthermore, this class bears excessive similarity to the PN class of S^3PGR^2 (also called S^4PR) nets, that have been used for a very long time to model resource allocation functions like those considered in the paper; in the authors' own words (see third paragraph of the first column in p. 1613 of that manuscript): "our proposed class of models is originated from and with no inherent difference to some existing ones, e.g., S^3PGR^2 and S^4PR, in [35] and [43]" (the numbers are referring to references that are provided in the considered paper and they correspond to [1] and [2] in this correspondence).

In spite of the above-mentioned acknowledgement, in the paper under consideration, the authors proceed to provide a "liveness analysis" for the new class of FS^4R nets (see Section III-B of that paper), and Theorem 1 of that manuscript, that constitutes the culmination of the undertaken analysis by providing a structural characterization of the liveness of the considered nets, ends up being erroneous. Since the considered PN classes play a central role in a large part of the literature that concerns the liveness analysis and control of complex resource allocation systems (RASs), such as the AMS discussed in the paper under consideration, we feel obliged to point out: 1) the fallacy in the aforementioned result and 2) certain directions for its correction. Also, in our concluding remarks, we take the opportunity to provide some further comments on the contributions that are attempted in the considered manuscript.

Manuscript received September 18, 2016; revised February 11, 2017; accepted February 22, 2017. Manuscript received in final form February 24, 2017. Recommended by Associate Editor M. Dotoli.

S. Reveliotis is with the School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA (e-mail: spyros@isye.gatech.edu).

Digital Object Identifier 10.1109/TCST.2017.2675840

II. New Class of FS^4R PNs and Its Relationship to the Existing Class of S^3PGR^2/S^4PR Nets

To facilitate the subsequent discussion, in this section, we overview the existing class of S^3PGR^2/S^4PR PNs, and subsequently we use this characterization in order to also present the novel class of the FS^4R nets that was introduced in the considered paper. In order to facilitate a clearer exposition and a more straightforward understanding of the points to be made next, and in line with the concise nature of the IEEE TCST correspondences, we carry out the subsequent discussion at a more informal level; formal definitions and analysis of the considered PN models can be found in [1], [3], and [4].

S^3PGR^2/S^4PR PNs essentially model the resource allocation dynamics that are generated by a set of concurrently running process types that compete for the staged allocation of a finite set of resources to their executing processes. Resources are nonperishable and reusable. They are also classified into a number of types, with each resource type consisting of a number of identical discrete units of that type, that define the corresponding "capacity." The system resources are requested in certain "bundles" by the running processes for the execution of their various processing stages, they are allocated to these processes in an exclusive mode, and they are returned to the corresponding pools of free resource types when these processes advance to their next processing stages (having secured, in the meantime, the resources that are necessary for these new stages).

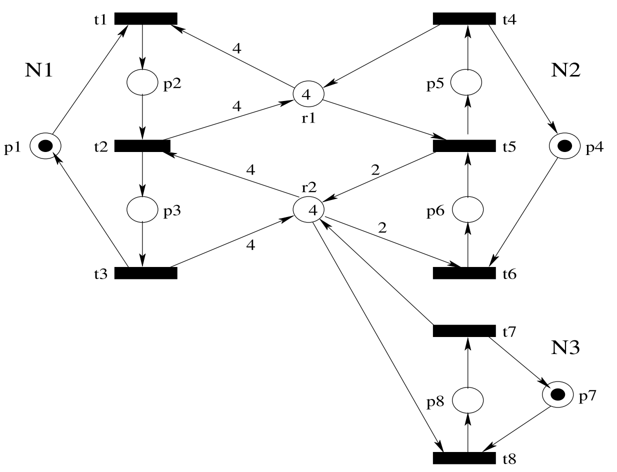

Fig. 1. Example S^3PGR^2/S^4PR net.

In the modeling formalism of the S^3PGR^2/S^4PR nets, each process type, $\mathcal{J}_j$, is modeled by a strongly connected state machine, $\mathcal{N}_j$, such that each directed cycle in that state machine contains a particular place p_{0j} that is known as the corresponding "idle" place. The state machines $\mathcal{N}_j$ are known as the "process" subnets of the S^3PGR^2/S^4PR net, and their "idle" place models the "environment" of the modeled operations; more specifically, tokens in these "idle" places model process instances that either wait to start the execution of the corresponding process type, or have just completed such an execution. In fact, by merging these two process instances in one place, the resulting token recirculation enables the perpetual execution of the considered process types while keeping the marking of the corresponding state machines bounded. The remaining places in each subnet $\mathcal{N}_j$ model the various processing stages of the corresponding process type $\mathcal{J}_j$, and they are called the "operation" places of the net. Furthermore, any directed path of $\mathcal{N}_j$ leading from the idle place p_{0j} back to this place models a particular "process plan" for the execution of the process type $\mathcal{J}_j$. Obviously, in the initial marking of a well-marked S^3PGR^2/S^4PR net, all the "operation" places are empty, while each "idle" place must contain at least one token.

Resource types and the corresponding resource allocation function are modeled in S^3PGR^2/S^4PR nets by an additional set of places that are known as the "resource" places. Each resource type has its own "resource" place, and the tokens in this place model the available free units of this resource type. In particular, the initial marking of each "resource" place defines the corresponding resource capacity. "Resource" places are connected to the various transitions of the process subnets $\mathcal{N}_j$ in a way that models the allocation and deallocation of the various resource units to the executing process instances. A "process" subnet $\mathcal{N}_j$ augmented with the necessary "resource" places is usually characterized as the "augmented process" subnet $\overline{\mathcal{N}}_j$ [4]. The "resource" places of a S^3PGR^2/S^4PR net are well-marked if all the "augmented process" subnets are quasi-live, i.e., each transition in each of these subnets is fireable when starting from the corresponding initial marking. Finally, the entire S^3PGR^2/S^4PR net can be perceived as a set of "augmented process" subnets $\overline{\mathcal{N}}_j$ that are "merged" through their shared resource places.

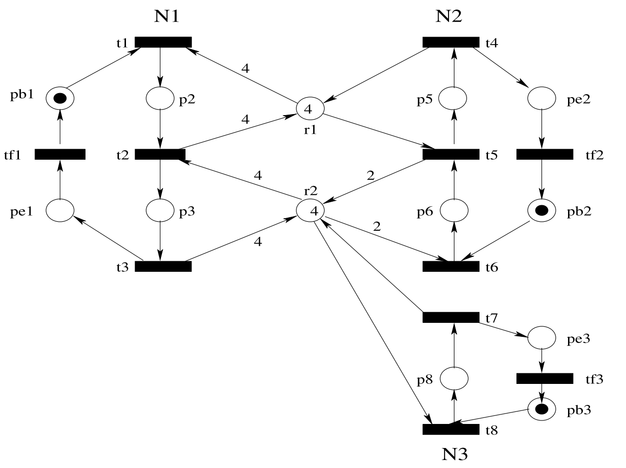

Fig. 2. FS^4R net corresponding to the example S^3PGR^2/S^4PR net of Fig. 1.

An example S^3PGR^2/S^4PR net is modeled in Fig. 1. The "process" subnets of this net are defined by the circuits $\langle p_1, t_1, p_2, t_2, p_3, t_3, p_1\rangle$, $\langle p_4, t_6, p_6, t_5, p_5, t_4, p_4\rangle$, and $\langle p_7, t_8, p_8, t_7\rangle$, with the corresponding "idle" places being p_1, p_4, and p_7. The "resource" places are places r_1 and r_2, each with a capacity of four units.

On the other hand, Fig. 2 provides the FS^4R net that corresponds to the example S^3PGR^2/S^4PR net of Fig. 1. As it can be seen by the juxtaposition of these two nets, their only difference is that, in the FS^4R model, each "idle" place p_{0j} has been split into two places p_{bj} and p_{ej}, holding, respectively, the tokens corresponding to process instances that wait to start or have completed their processing; the recirculation of all these tokens in the corresponding "process" subnets is facilitated by the addition of the transition t_{fj} that is characterized as a "feedback" transition in the considered manuscript, and apparently gives this model its name. In this new representation, the only place marked in each "process" subnet $\mathcal{N}_j$ is (naturally) the place p_{bj}. An additional stipulation in Definition 1 of the considered paper, that is not captured very emphatically by the provided example net, is that the flow relation that defines the connectivity of each transition t_{fj} to its input and output places p_{ej} and p_{bj} must be equal to the initial marking of p_{bj}; this stipulation implies that the firing of t_{fj}, and the corresponding token recirculation that is effected by this firing, can take place only when the entire initial batch of tokens in p_{bj} has executed successfully and it has been transferred in p_{ej}.

As discussed in the introductory section, in the considered manuscript, the authors proceed to a liveness analysis of the class of FS^4R nets. But Theorem 1 in that paper, which provides the main result of that analysis, is erroneous. We establish this fact in Section III by providing a more exact statement of the claimed result, and countering this statement with a refuting example.

III. Counter-Example to Theorem 1 of the Considered Paper

The result of Theorem 1 in Section III-B of the considered manuscript tries to connect the liveness of the FS^4R nets to the presence in their dynamics of a particular structural concept that is characterized as an “undermarked siphon.”[1] More specifically, according to the aforementioned theorem, *a well marked FS^4R net is not live if and only if there is a reachable “undermarked” siphon S in the dynamics of the considered net.*

Next, first we discuss briefly the concept of the “(undermarked) siphon” and the way that it has been connected to the structural analysis of the PN classes that model resource allocation. Subsequently, we demonstrate with a counter-example the error in the claims of Theorem 1, and we provide a brief explanation on the source of this error. We also provide some pointers to additional material that expands on the corresponding theory for S^3PGR^2/S^4PR nets; this material includes a correct siphon-based structural characterization of the liveness and the reversibility of the S^3PGR^2/S^4PR nets, which is also applicable to the class of the FS^4R nets introduced in this paper.

We start by reminding the reader that a siphon S is a subset of places with their “input” transitions (i.e., the transitions that can bring tokens into these places) being a subset of their “output” transitions (i.e., the transitions that need tokens from some place $p \in S$ in order to fire, and therefore, remove tokens from S). Furthermore, according to the definitions that are provided in the considered paper, a siphon S is “undermarked” at a marking M if each “output” transition of this siphon is disabled by a place of S. Since, according to the siphon definition, each “input” transition of S is also one of its “output” transitions, it follows that an “undermarked” siphon will never acquire any new tokens, and this situation will persist, rendering all the “output” transitions of S dead transitions at M.

Besides being able to explain the deadness of certain transitions, in the (generalized) PN theory, “undermarked” (or, as they are more frequently called, “deadly marked”)[2] siphons have also been used to explain the formation of a PN “deadlock,” i.e., a PN marking where no transition is fireable. Also, in the PN-based RAS theory, empty siphons, which is a particular class of “undermarked” siphons, have been used to explain the nonliveness of certain ordinary PN classes that model more restricted resource allocation dynamics [5].

On the other hand, in [1], [3], [4], and [6], it has been shown that the notion of the “deadly marked siphon” is not able to interpret the nonliveness of the S^3PGR^2/S^4PR nets when applied on the dynamics of these nets as represented by the corresponding reachability space. The next example establishes that this is also the case for the “undermarked siphons” defined earlier, and the new class of FS^4R nets.

Example: Fig. 3 shows a reachable marking M of the FS^4R net of Fig. 2 where the tokens in places p_2 and p_6 are in a (partial) deadlock, and therefore, transitions t_2 and t_5 are dead (possibly together with some other transitions). Yet, it

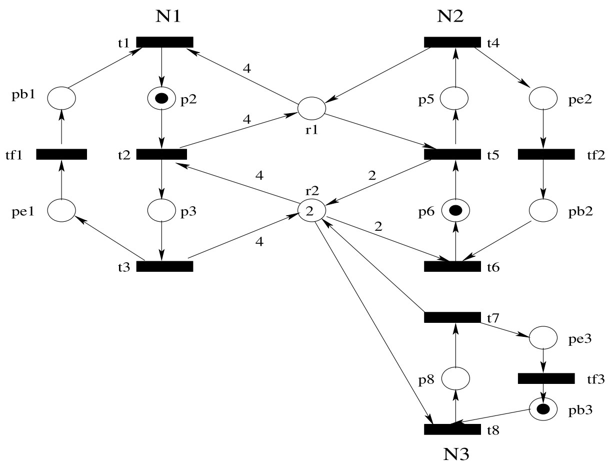

Fig. 3. Marking corresponding to the partial deadlock in the provided counter-example to Theorem 1 of the considered paper.

can be checked that there is no “undermarked” siphon S in the depicted marking, or in any marking reachable from it, that can explain the deadness of the aforementioned transitions. *Hence, Theorem 1 of Section III-B in the considered paper is wrong.* □

A simple intuitive explanation of the negative result that is established with the above-mentioned example is as follows. Even though the two tokens in resource place r_2 cannot enable the advancement of the token in p_2, they can support the repetitive execution of the process type $\mathcal{J}_3$; indeed, the token in place p_{b3} can recirculate *ad infinitum* in the corresponding “process” subnet $\mathcal{N}_3$, But then, the resource place r_2 will lose and regain tokens *ad infinitum*, as well, and therefore, it cannot be part of the undermarked siphon that would interpret the partial (resource allocation) deadlock of Fig. 3.

In more general terms, the reason that deadly marked siphons are unable to interpret the nonliveness of the S^3PGR^2/S^4PR nets, is the nonhomegeneity, in terms of the required number of units, of the resource allocation requirements that are supported by these nets. This nonhomegeneity leads to situations where some resource types are involved in a deadlock, but still they have a sufficient number of free units to support the perpetual execution of some other process types. Hence, the places modeling these resources lose and regain tokens *ad infinitum*, and therefore, they cannot be part of an “undermarked” siphon (as it would be required by the typical siphon-based interpretations of nonliveness).

Given the affinity between the PN class of S^3PGR^2 / S^4PR nets and the FS^4R nets that was defined in Section II, it is not surprising that the above-mentioned remarks carry over to the FS^4R nets. We also mention, for completeness, that the works of [1], [4], and [6] provide a methodology for reconnecting the lack of liveness of the S^3PGR^2/S^4PR nets to the notion of “deadly marked siphon.” But this reconnection is attained in an alternative representation of the underlying net dynamics, that has come to be known as the “modified reachability space” in the corresponding theory; interested readers are referred to those works for the further details. Also, the work of [7]–[9] exemplifies the employment of the “modified reachability space” and the corresponding structural

[1]This type of results are characterized as “structural analysis” of the corresponding PN models.

[2]There is a slight difference in the definition of these two concepts, in that “deadly marked siphons” request the disabling by the siphon places of the input transitions only, but this difference is not essential for the role that these two types of siphons play in the structural analysis of the liveness of the considered PN classes.

characterization of liveness that is based on this concept, in the context of some additional PN classes that have been studied by the current RAS theory.

IV. Some Concluding Remarks

The above-mentioned discussion has rendered clear the erroneous nature of the result of Theorem 1 in the considered paper. This situation is more unfortunate when taking into account the "no inherent difference [of the FS^4R PN class] to some existing ones, e.g., S^3PGR^2 and S^4PR in [35] and [43]" (in the words of the authors of the considered paper). In view of this last statement, and the further substantiation of this statement through the discussion of Section II, the author of this correspondence cannot help wondering what is the actual need covered by the introduction of this new PN class, in the first place.

References

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

[2] F. Tricas, J. M. Colom, and J. Ezpeleta, "A solution to the problem of deadlock in concurrent systems using Petri nets and integer linear programming," in *Proc. 11th Eur. Simulation Symp.*, 1999, pp. 542–546.

[3] F. Tricas, F. Garcia-Valles, J. M. Colom, and J. Ezpeleta, "A Petri net structure-based deadlock prevention solution for sequential resource allocation systems," in *Proc. IEEE ICRA*, Sep. 2005, pp. 271–277.

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

[5] J. Ezpeleta, J. M. Colom, and J. Martínez, "A Petri net based deadlock prevention policy for flexible manufacturing systems," *IEEE Trans. Robot. Autom.*, vol. 11, no. 2, pp. 173–184, Apr. 1995.

[6] S. A. Reveliotis, "On the siphon-based characterization of liveness in sequential resource allocation systems," in *Applications and Theory of Petri Nets*, LNCS 2679, Berlin, Germany: Springer, 2003, pp. 241–255.

[7] H. Liao *et al.*, "Concurrency bugs in multithreaded software: Modeling and analysis using Petri nets," *Discrete Event Syst. Theory Appl.*, vol. 23, no. 2, pp. 157–195, 2013.

[8] H. Liao, S. Lafortune, S. Reveliotis, Y. Wang, and S. Mahlke, "Optimal liveness-enforcing control of a class of Petri nets arising in multithreaded software," *IEEE Trans. Autom. Control*, vol. 58, pp. 1123–1138, Sep. 2013.

[9] H. Liao *et al.*, "Eliminating concurrency bugs in multithreaded software: A new approach based on discrete-event control," *IEEE Trans. Control Syst. Technol.*, vol. 21, no. 6, pp. 2067–2082, Nov. 2013.