Counterexamples to "Liveness-Enforcing Supervision of Bounded Ordinary Petri Nets Using Partial Order Methods" (*)

Xiaolan Xie
INRIA/MACSI Team
ISGMP – Bat A
Ile du Saulcy, 57045 Metz, France
xie@loria.fr

Alessandro Giua
Dip. di Ingegneria Elettrica ed Elettronica
Universita' di Cagliari
Piazza d'Armi - 09123 Cagliari, Italy
giua@diee.unica.it

Abstract. This note shows by means of simple counterexamples that some key results presented by He and Lemmon on the liveness verification and enforcing of Petri nets using unfolding are incorrect. As a result, the applicability of unfolding for Petri net supervision is still an open issue.

1 - Introduction

Recently, He and Lemmon have presented an original approach for the analysis and control of bounded Petri nets based on *unfolding*. This technique allows one to describe the set of reachable marking of a given net N by means of a finite *occurrence net* (called the unfolding of N) without the necessity of explicitly computing the reachability graph of N [4]. This often leads to significant computational advantages.

In a series of papers [1-3] He and Lemmon have used unfolding for liveness verification and enforcing. We show in this note through a series of counterexamples that some key results of these papers are incorrect. As a result, although we still strongly believe that unfolding is an interesting and potentially fruitful technique for Petri net control, the applicability of unfolding for Petri net supervision is still an open issue.

2 - Counterexamples for [1]

Let us first consider the approach proposed in [1]. The unfolding β_c used in this approach is an extension of the well-known McMillan unfolding [4] and includes cut-off transitions not preceding any other cut-off transitions.

As an example, the unfolding β_c of the net N_1 in Figure 1.(a) is shown in part (b) of the same figure. Note that here we are conventionally using a double arrowed arc to represent two arcs with opposite directions.

The approach proposed in [1] is based on three concepts:
- deadlocked configurations;
- cut graph;
- cut cycles.

(*) Paper to appear as a correspondence on the *IEEE Transactions on Automatic Control, 2004*.

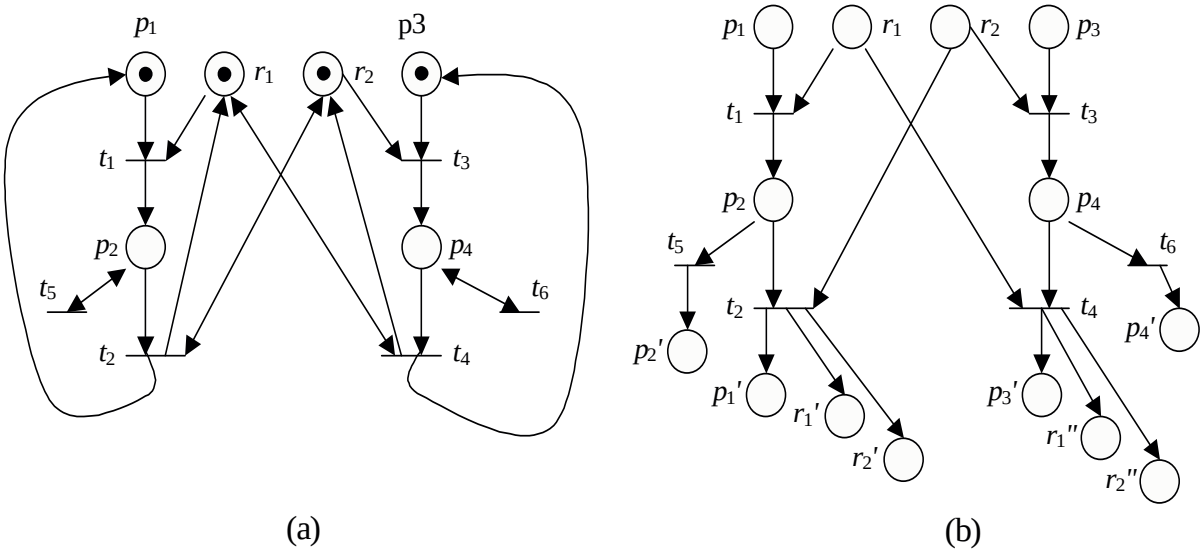

Figure 1: A first counterexample: (a) the net N_1; (b) its unfolding β_c.

A configuration of an unfolding net is said *deadlocked* if it is not a subset of a *deadlock-free configuration*, i.e., of a configuration whose terminating transitions are all cut-off transitions. The unfolding of Figure 1 does not contain any deadlocked configuration.

A *cut graph* shows the reachability of a *base configuration* BC(t^e) associated to an *end transitions* t^e. The unfolding of Figure 1 has four end transitions t_5, t_2, t_4, t_6 whose base configurations are: BC(t_5)= { t_1, t_5 }, BC(t_2)= { t_1, t_2 }, BC(t_4)= { t_3, t_4 }, BC(t_6)= { t_3, t_6 }.

Definition 1: Consider two end transitions t^e_1 and t^e_2 with corresponding base configurations BC(t^e_1) and BC(t^e_2). Configuration BC(t^e_2) is reachable from BC(t^e_1) if either

(i) $M(h_c(Cut([t^e_1]))) = M_0$ or

(ii) there exists a transition $t_1 \in$ BC(t^e_1) such that $M(h_c(Cut([t^e_1]))) = M(h_c(Cut([t_1])))$ and $[t_1] \subset [t^e_2]$

where $M(h_c(Cut(C)))$ is the marking of the configuration C and M_0 is the initial marking.

Note that this definition of reachability is slightly different from the one given in [1] that does not consider the condition (i). However what the authors of [1] want to capture is whether a suffix of $[t^e_2]$ can be reached after an execution of configuration $[t^e_1]$. This is clearly the case if $M(h_c(Cut([t^e_1]))) = M_0$. Furthermore, we observe that the authors of [1] implicitly use our definition in the construction of the cut graph of Figure 5 in [1] (otherwise no base configuration can be reached starting from BC(t_7)).

The *cut graph* is a graph whose nodes are the base configurations and such that there exists an arc connecting BC(t^e_1) to BC(t^e_2) iff BC(t^e_2) is reachable from BC(t^e_1). The cut graph of the unfolding in Figure 1 is given in Figure 2.

A *cut cycle* is a cycle of the cut graph (BC(t^e_1), BC(t^e_2)) (BC(t^e_2), BC(t^e_3)) ... (BC(t^e_k), BC(t^e_1)).

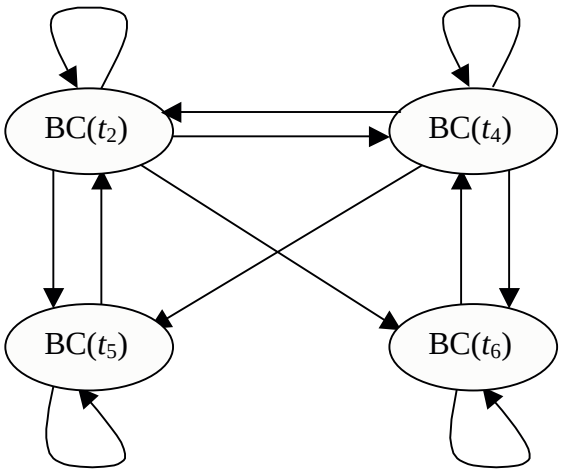

Figure. 2: The cut graph of net N_1 in Figure 1.

Definition 2 ([1] pg, 1049, col 2): A cut cycle is *live* if

(i) either $\cup[t^e_j]$-$[t_j] = T$ where $t_j < t^e_j$ and $h_c(\text{Cut}([t^e_j])) = h_c(\text{Cut}([t_j]))$ (in this case it corresponds to a sequence that fires all transitions)

(ii) or $M(h_c(\text{Cut}([t^e_k]))) = M_0$ (in this case it corresponds to a stationary sequence that returns to the initial marking).

We first show that the following main result of [1] that characterizes liveness is wrong.

Lemma 9 ([1])*: If there does not exist any deadlocked configuration in* β_c*, then the original net is live if and only if every maximal cut cycle is live.*

The net N_1 in Figure 1 is a counterexample. It has no deadlocked configuration and its maximal cut cycle (that includes the whole cut graph) is live because it corresponds to a sequence that fires all transitions. However, the net N_1 is not live because after the firing of the sequence t_1t_3 transitions t_1, t_3, t_2, t_4 are dead.

Remark 1: The counterexample in Figure 1 shows that the cut graph may not completely capture the reachability between base configurations. In other words, if there is an arc from BC1 to BC2 in the cut graph one cannot be sure that all transitions in BC2 are certainly reachable from any marking (or configuration) contained in BC1. This is because the execution of transitions in another base configuration BC3 may "disable" the reachability of BC2 from BC1. In the example, although by definition BC(t_2) is reachable from BC(t_5), the execution of transition t_3 (right after the execution of t_1) in base configuration BC(t_4) makes the base configuration BC(t_2) unreachable from BC(t_5).

A second main result of [1] that is wrong concerns the existence of liveness enforcing supervisors.

Theorem 2 ([1]): *There exists a liveness-enforcing supervisor for a completely controllable bounded system iff there exists at least one live cut cycle in the cut graph.*

Our second counterexample and its unfolding is given in Figure 3. The unfolding has three cut-off transitions t_2, t_4, t_5 and no deadlocked configuration. Its cut graph is given in Figure 4.

The cut cycle (BC(t_2), BC(t_4))(BC(t_4), BC(t_2)) is live because it corresponds to a stationary sequence that returns to the initial marking. However, there exists no supervisor that makes this net

live: in fact, to fire t_5 it is necessary to reach the marking $[0\ 1\ 0\ 1\ 0\ 0]^T$ from which all other transitions are dead.

Remark 2: The counterexample in Figure 3 shows that a live cut cycle that goes back to the initial marking may still correspond to a behavior that is not "live" (i.e. the execution of base configuration [t_5]). As a result, the marking-based supervisor defined in the sufficiency proof of Theorem 2 of [1] may still contain a behavior that is not "live" since it does not prevent a cut cycle that goes back to the initial marking.

Other key results of [1] can be summarized informally as follows.

- Theorem 1 is based on Lemma 9 and claims that a bounded Petri net is live iff there is no deadlock configuration and every maximal cut cycle in the cut graph is live.
- The proof of Theorem 2 is based on the existence of a maximally permissive marking-based supervisor that disables any transition leading out of live cut cycles.
- Corollary 1 claims that such a controller is maximally permissive.
- Theorem 3 extends Theorem 2 to Petri nets with uncontrollable transitions.

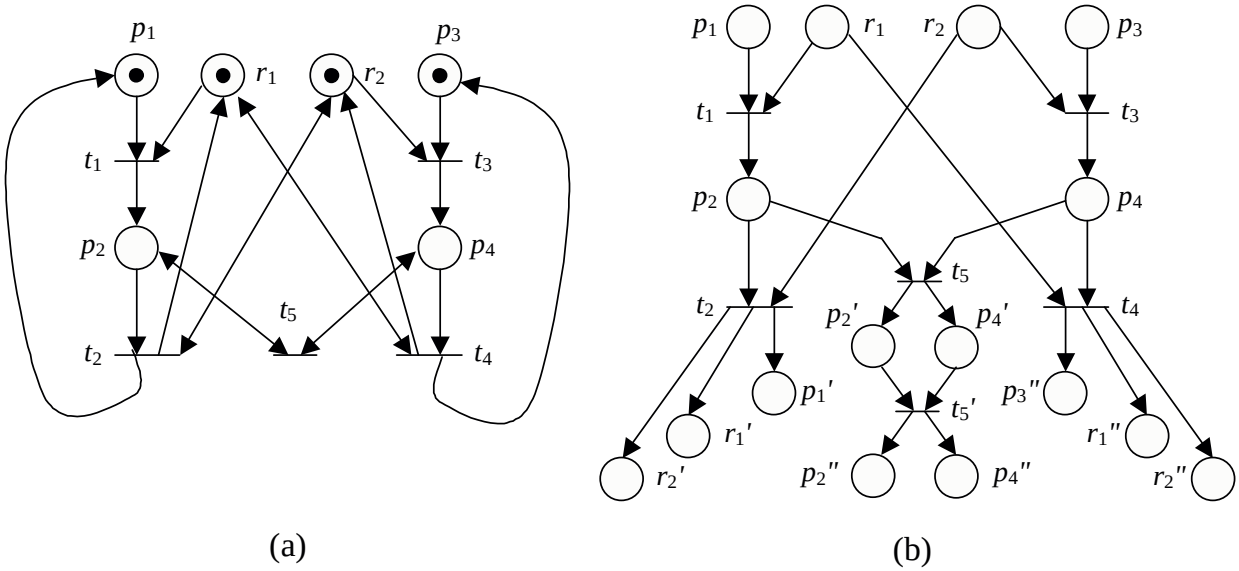

Figure 3: A second counterexample: (a) the net N_2; (b) its unfolding.

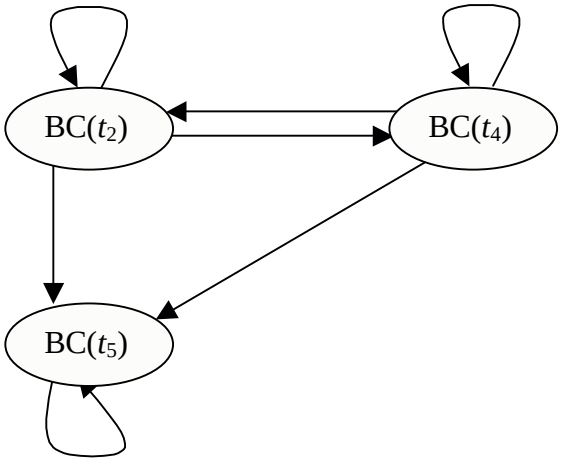

Figure 4: The cut graph of the net N_2 in Figure 3.

On the base of the two counterexamples, we summarize what we feel are the main problems of the approach presented in [1].

1) *Liveness analysis*. The first counterexample shows that Lemma 9 is not correct. From this follows that the sufficient part of Theorem 1 is not correct (possibly, the necessity part of Theorem 1 is correct but a rigorous proof is still needed).

2) *Liveness enforcing supervision*. The second counterexample shows that the sufficiency part of Theorem 2 is not correct (possibly, the necessity part is correct). It might be possible to correct the problem changing the definition of live cut cycle to require that it always contains all transitions but a formal proof is still needed.

3) *Maximally permissiveness*. The sufficiency part of Theorem 2 (assuming it can be corrected with the changes suggested in 2) is more or less trivial: basically it says that if there exists a set of repetitive sequences containing all transitions then the net can be made live. This is obvious if one fires these sequences one at a time. What is important is a way to find a maximally permissive supervisor. However, the construction of the marking-based supervisor P in the proof of the theorem is based on Theorem 1 that is not correct. As an example, the marking-based supervisor P does not disable any transition of net N_1 in Figure 1 (because the whole cut graph is a live cut cycle) and hence is not a liveness-enforcing supervisor. As a result, the proof of Theorem 2 is incorrect and Corollary 1 does not apply any more. Also Theorem 3 on uncontrollable transitions is not correct: in fact, it uses the same argument given in the proof of Theorem 2.

The sound contribution of [1] is thus reduced to just a way of characterizing by means of unfolding nets where "there exist a set of repetitive sequences containing all transitions" that (trivially) can be made live by a suitable supervisor not necessarily maximally permissive. Note, however, that there exist structural ways of determining repetitive sequences (T-invariants): the only use of the unfolding could be that of checking the existence of a marking enabling such a sequence without constructing the reachability graph (but in this case McMillan unfolding is enough).

3 - Counterexample for [2-3]

Let us consider now the paper [2] that uses the same unfolding but different concepts for liveness verification. We prove that the following key result of [2] is incorrect. Hence, the liveness enforcing supervision of [3] that is based on this result is also incorrect.

Theorem 1 ([2]). A bounded Petri net (N, M_0) is live iff its unfolding net β_c has the following properties:

1. β_c contains all transitions of N,
2. There is no dead transitions in β_c,
3. every cycle of β_c is live, and
4. there does not exist a set of cycles of β_c that is in cyclic lock.

Recall that the concept of cycles is related to cut-off transitions. Let t' be a cut-off transition and let t be a transition preceding t' in bc such that $M(h_c(Cut([t])))$ = $M(h_c(Cut([t'])))$. We define a cycle $C_t^{t'}$ as the set of transitions $C_t^{t'} = [t'] - [t]$. A cycle $C_t^{t'}$ is said reversible if $M(h_c(Cut([t'])))$ = M_0. A cycle $C_t^{t'}$ is said live if there exists a set of cycles $C_{t1}^{t1'}$, $C_{t2}^{t2'}$, ..., $C_{tk}^{tk'}$ such that (i) $[t] \subseteq [t'_1]$, (ii) $[t_j]$

$\subseteq$ [t'$_{j+1}$], for j = 1, ..., k-1, and (iii) either $C_{tk}^{tk'}$ is reversible or $h_c\left(\bigcup_{j=1}^{k} C_{t_j}^{t_j'} \cup C_t^{t'}\right) = T$. A set of cycles $C_{t1}^{t1'}$, $C_{t2}^{t2'}$, ..., $C_{tk}^{tk'}$ is said to be in cyclic lock if there exists two transitions t_j^c and $t_j^{c'}$ in [t'$_j$] for each cycle such that (i) t_j^c precedes $t_j^{c'}$, (ii) transitions t_j^c for all j are concurrent, (iii) transitions $t_j^{c'}$ and t_{j+1}^c for j = 1, ..., k-1 are in conflict, (iv) transitions $t_k^{c'}$ and t_1^c are in conflict.

The net N_3 in Figure 5 is a counterexample disproving Theorem 1 in [2]. The cycles C_Φ^{t2} and C_Φ^{t4} forms a cyclic lock with the set $\{t_j^c\} = \{t_1, t_3\}$ and the set $\{t_j^{c'}\} = \{t_2, t_4\}$. As a result, condition (4) of Theorem 1 of [2] does not hold and hence, from this theorem, the net N_3 is not live. This contradicts the liveness of the net N_3 that can be easily checked by its marking graph and concludes the incorrectness of Theorem 1 of [2]. Note that what we actually prove is that the necessity part of this theorem is wrong. On the contrary the sufficiency part might be correct: however, we feel that the proof given in [2] is not convincing because it uses several unproved arguments.

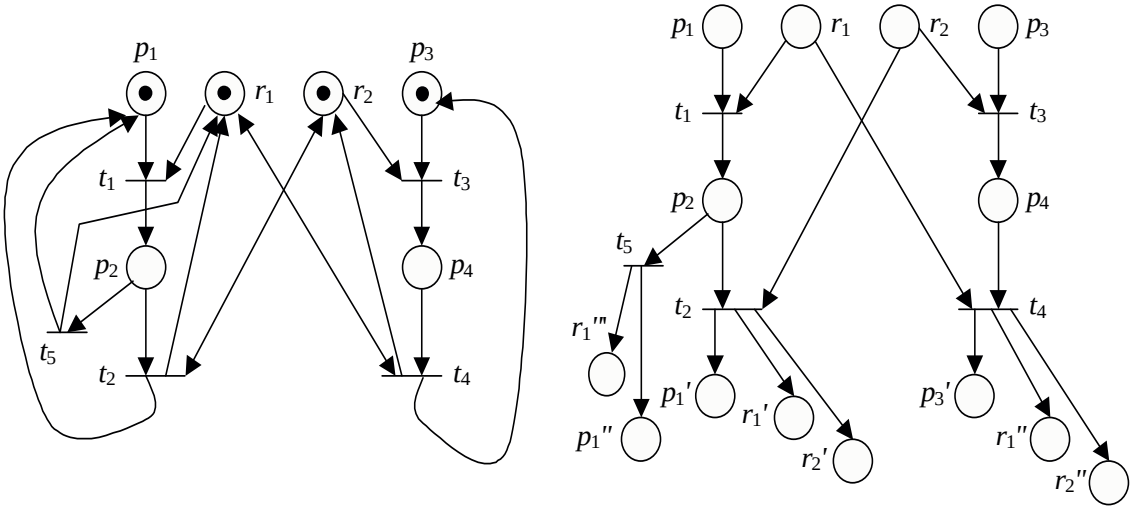

Figure 5: A third counterexample: (a) the net N_3; (b) its unfolding.

Acknowledgements
Remarks 1 and 2 are based on the comments of an anonymous referee: we are grateful for these clear intuitive explanations.

References

[1] K.X. He and M.D. Lemmon, "Liveness-enforcing supervision of bounded ordinary Petri nets using partial order methods", *IEEE Transactions on Automatic Control*, Vol. 47, No. 7, pp. 1042-1055, July 2002.

[2] K.X He and M.D. Lemmon, "Liveness verification of discrete-event systems modeled by n-safe ordinary Petri Nets", *Proc. 21st International Conference on Application and Theory of Petri Nets* (Aarhus Denmark), Lecture Notes in Computer Science 1825, pp. 227-243, Springer, 2000.

[3] K.X He and M.D. Lemmon, "Liveness enforcing supervision of discrete-event systems modeled by n-safe Petri nets", *Proc. of the IFAC International Conference on Control System Design* (Bratislava Slovakia), June 2000.

[4] K.L. McMillan, "A technique of state space search based on unfolding," *Formal Methods in System Design*, 6(1):45-65, 1995.