Testing Discrete Event Systems: Synchronizing Sequences using Petri Nets

Marco Pocci$^{(a)}$, Isabel Demongodin$^{(b)}$, Norbert Giambiasi$^{(c)}$, Alessandro Giua$^{(d)}$

$^{(a)-(c)}$Laboratoire des Sciences de l'Information et des Systèmes, Campus de Saint Jérôme, Marseille, France
$^{(d)}$Department of Electrical and Electronic Engineering, University of Cagliari, Cagliari, Italy

{$^{(a)}$marco.pocci, $^{(b)}$isabel.demongodin, $^{(c)}$norbert.giambiasi}@lsis.org, $^{(d)}$giua@diee.unica.it

Abstract

In the field of Discrete Event Systems, an important class of testing problems consists in determining a final state after the execution of a test. This problem was completely solved in the 60's using homing and synchronizing sequences for finite state machines with inputs/outputs. In a synchronizing problem, we want to drive an implementation of a given model, seen as a black-box, to a known state regardless of its initial state and the outputs. In this paper, we propose a first approach to solve the synchronizing problem on systems represented by a class of synchronised Petri nets. We show that, regardless of the number of tokens that the net contains, a synchronizing sequence may be computed in terms of the net structure, thus avoiding the state explosion problem.

Keywords: testing problems, finite state machines, Petri nets, synchronizing sequences

1. Introduction

This paper deals with the problem of determining a final state of an implementation of a Discrete Event Systems (DES), extending the existing approaches for *Finite State Machines* (FSM) to a class of *Petri nets* (PN).

The fundamental problems in testing FSM and the techniques for solving these problems have been pioneered in the seminal paper of Moore (Moore, 1956) and have been reviewed extensively by Lee and Yannakakis (Lee and Yannakakis, 1996). They stated five fundamental problems of testing: i) determining a final state after a test; ii) state identification; iii) state verification; iv) conformance testing; v) machine identification. Among these, we consider the problem of determining a final state after a test. This problem was addressed and essentially completely solved around 1960 using *homing* (HS) and *synchronizing sequences* (SS). An homing sequence is an input sequence that brings a FSM from an initial state —supposed unknown— to a state that can be determined by the observed output event sequence. A synchronizing sequence is an input sequence that brings a FSM from any initial unknown state to a known state regardless of the output sequences.

Several approaches are used to obtain a synchronizing sequence for an FSM. The synchronizing tree method (Kohavi, 1978; Hennie, 1968); since memory required to build up the tree is high, this method is suitable only for small FSM. Simulation-based method, binary decision diagram method (BDD) (Pixley et al., 1994; Rho et al., 1993);the drawback of this method is that the length of the synchronizing sequence may be far away from the lower bound. Homing sequence method (Pomeranz and Reddy, 1994); the method alleviates the need for a machine to have synchronizing sequence but its main disadvantage is that the final state is different for each power-on and we must observe the output responses to determine the final state. Finally Eppstein (Eppstein, 1990) gave an algorithm, based on Natarajan's work (Natarajan, 1986), for *reset sequences*. This algorithm, that in polynomial time either finds such sequences or proves that none exists, has been described by Lee *et al.* (Lee and Yannakakis, 1996) and adapted for synchronizing sequences.

Little has been done in the area of testing of systems specified as Petri Nets. For example Jourdan and Bochmann (G.-V. Jourdan and G.v. Bochmann, 2009) investigated the question of automatically testing Petri Nets, to ensure the conformance and so that an implementation of a specification provided as a Petri Net is correct. Zhu and He gave an interesting classification of testing criteria (Zhu and He, 2002) — without testing algorithm — and presented a theory of testing high-level Petri nets (H. Zhu and X. He, 2000) by adapting some of their general results in testing concurrent software systems. In the Petri net modelling framework, one of the main supervisory control tasks is to guide the system from a given initial marking to a desired one similarly to the synchronisation problem. Yamalidou *et al.* presented a formulation based on linear optimisation (Yamalidou and Kantor, 1991; Yamalidou et al., 1992). Giua and Seatzu (Giua and Seatzu, 2002) defined several observability properties and dealt with the problem of estimating the marking of a place/transition net based on event observation.

The main idea of this paper is to apply the existing techniques developed for FSM to Petri nets. We consider *synchronised nets*, i.e., a net where a label is associated to each transition. The label represents an external input event whose occurrence causes the transition firing, assuming the transition is enabled at the current marking. Note that since we are not considering outputs, we will only be concerned with synchronizing sequences.

It is well known that a bounded PN — that is a PN where the number of tokens in each place does not exceed a finite number k for any marking reachable from the initial one — can be represented by a finite *reachability graph*, i.e., a FSM whose behaviour is equivalent to that of the PN (Murata, 1989). Thus the existence of synchronizing sequences for these models can be studied using the classical approach for FSM with just minor changes to take into account that while a FSM is assumed to be *completely specified* — any event can occur from any state — in the case of the PN's reachability graph it is not usually true.

Then we consider a special class of Petri nets called *state machines* (Murata, 1989), characterised by the fact that each transition has a single input and a single output arc. These nets are similar to automata, in the sense that the reachability graph of a state machine with a single token is isomorph — assuming all places can be marked — to the net itself. However, as the number of tokens k in the net increases the reachability graph grows as

k^{m-1} where m is the number of places in the net. We show that for strongly connected state machines even in the case of multiple tokens, the existence of synchronizing sequences can be efficiently determined by just looking at the net structure, thus avoiding the state explosion problem. We also present some extensions of our results to state machines that are not strongly connected.

This paper is organised as follows: Section 2. presents the FSM and PN model together with the analyse of synchronizing sequences for FSM. In Section 3. is shown how the FSM method can be straightforwardly adapted to PN. Afterwards in Section 4. is reviewed the problem and proposed other techniques for the case of State Machine. Finally conclusion are drawn in Section 5.

2. Background

2.1. Mealy Machine

Mealy machines are the class of FSM concerned in this work. A Mealy machine M is a structure

$$M = (I, O, S, \delta, \lambda)$$

where: I, O and S are finite and nonempty sets of input events, output events and states respectively; $\delta : S \times I \to S$ is the state transition function and $\lambda : S \times I \to O$ is the output function.

When the machine is in the current state $s \in S$ and receives an event $i \in I$, it moves to the next state specified by $\delta(s, i)$ producing an output given by $\lambda(s, i)$. Note that functions δ and λ are assumed to be *total functions*, i.e., functions defined on each element (s, i) of their domain. Such a machine is called *completely specified* to denote that for any state and upon any event a transition occurs producing the corresponding output event.

We denote the number of states, input and output events by $n = |S|$, $p = |I|$ and $q = |O|$. We extend the transitions function δ from input events to strings of input events as follows: for an initial state s_1, an input sequence $x = a_1, \ldots a_k$ takes the machine successively to the states $s_{i+1} = \delta(s_i, a_i), i = 1, \ldots, k$ with the final state $\delta(s_1, x) = s_{k+1}$. We also extend transition function δ from being defined for a specified state to a set of state as follows: for a set of states $S' \subseteq S$, an input event $i \in I$ takes the machine to the set of states $S'' = \bigcup_{s \in S'} \delta(s, i)$.

A Mealy machine is said *strongly connected* if there exists a directed path from any vertex to any other vertex.

2.2. Synchronizing Sequences on Mealy Machines

A SS takes a machine to the same final state regardless of the initial state and the outputs. That is, an input sequence x is synchronizing to a state s_r iff $\delta(s_i, x) = \delta(s_j, x) = s_r$ for all pairs of states $s_i, s_j \in S$.

The information about the current state of M after applying an input x is defined by the set $\sigma(x) = \delta(S, x)$, called the *current state uncertainty of* x. In other words x is a SS that takes the machine to the final state s_r iff $\sigma(x) = s_r$.

Given a Mealy machine M with n states, we construct an *auxiliary directed graph* $\mathcal{A}(M)$ with $n(n+1)/2$ nodes, one for every unordered pair (s_i, s_j) of nodes of M, including pairs (s_i, s_i) of identical nodes. There is an edge from (s_i, s_j) to (s_p, s_q) labeled with an input event $a \in I$ iff in M there is a transition from s_i to s_p and a transition from s_j to s_q, and both are labeled by a.

M_0	$[2\,0\,0]^T$
M_1	$[0\,1\,0]^T$
M_2	$[1\,0\,1]^T$

Table 1. Markings of the PN in Fig. 1.(a).

The following algorithm, due to Natarajan (Natarajan, 1986), has been introduced by Eppstein (Eppstein, 1990) for the case of *reset sequences* and re-interpreted and described by Lee *et al.* in (Lee and Yannakakis, 1996). We propose a possible implementation of this work in order to construct a synchronizing sequence x which is not necessarily the shortest possible.

***Algorithm** 1.* (Computing SS leading to s_f). Let $M = (I, O, S, \delta, \lambda)$ be the considered Mealy machine.

```
1. Let $s_r$ be the desired final state.
2. Let $x = \varepsilon$, the empty initial input string.
3. Let $\sigma(x) = \{S\}$, the initial current state uncertainty.
4. Pick two states $s_i, s_j \in \sigma(x)$ such that $s_i \neq s_j$.
5. Find a shortest path in $\mathcal{A}(M)$ from node $(s_i, s_j)$ to $(s_r, s_r)$.
   5.1. If no such a path exists, stop the computation, there exists no SS for $s_r$.
   5.2. Else, let $x'$ be the input sequence along this path, do
        5.2.1. $\sigma(x) = \delta(\sigma(x), x')$ and
        5.2.2. $x = xx'$,
6. If $\sigma(x) \neq \{s_r\}$ go to step 4.
7. $x$ is the desired SS. ■
```

To analyse the Algorithm 1 leads us to the following theorem, which provides a reachability condition on the auxiliary graph necessary and sufficient for the existence of a synchronizing sequence for a desired final state.

***Theorem** 2. A FSM M has a synchronizing sequence for a desired final state s_r iff there is a path in its $\mathcal{A}(M)$ from every node (s_i, s_j), $1 \leq i < j \leq n$, to the node (s_r, s_r), with equal first and second components.*

Proof: There is an input sequence that takes the machine from state s_i and s_j, $i \neq j$, to the same state s_r iff there is a path x' in $\mathcal{A}(M)$ from (s_i, s_j) to (s_r, s_r). If such a path does not exist, none of the possible sequences will take the two states to s_r. In that case, when at step 5.2.1. the Algorithm updates the current state uncertainty it always holds that $s_i, s_j \in \sigma(x)$. In no more than n iteration, the Algorithm will be forced to pick the couple (s_i, s_j), then stating that no synchronizing sequence for the state s_r exists. □

We can also propose this theorem to prove the existence of any SSs.

***Theorem** 3. A FSM M has a synchronizing sequence iff there exists a path in its $\mathcal{A}(M)$ from every node (s_i, s_j), $1 \leq i < j \leq n$, to some node with equal first and second components (s_r, s_r), $1 \leq r \leq n$.* ■

The proof is trivial and not reported.

We will use this condition to prove the existence of a synchronizing sequence.

2.3. Synchronised Petri nets

In this section we recall the PN formalism used in the paper. For more details on PN we refer to (David and Alla, 2004).

A *Place/Transition net* (P/T net) is a structure

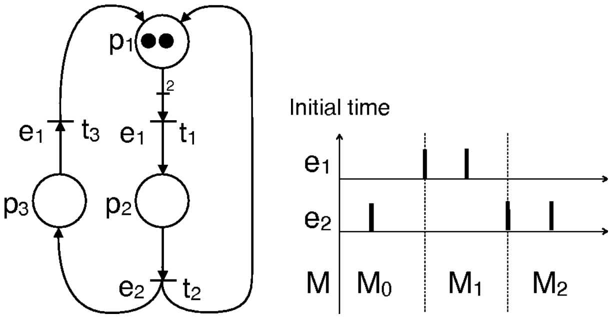

Fig. 1. A synchronised PN (a) and a possible behavior (b).

$$N = (P, T, Pre, Post)\,,$$

where P is the set of m places, T is the set of q transitions, $Pre : P \times T \rightarrow \mathbb{N}$ and $Post : P \times T \rightarrow \mathbb{N}$ are the pre and post incidence functions that specify the arcs.

A *marking* is a vector $M : P \rightarrow \mathbb{N}$ that assigns to each place a nonnegative integer number of tokens; the marking of a place p is denoted with $M(p)$. A *net system* $\langle N, M_0\rangle$ is a net N with initial marking M_0.

A transition t is enabled at M iff $M \geq Pre(\cdot, t)$ and may fire yielding the marking $M' = M - Post(\cdot, t) + Pre(\cdot, t)$. The set of transitions enabled at M is denoted $\mathcal{E}(M)$.

The notation $M[\sigma\rangle$ is used to denote that the sequence of transitions $\sigma = t_1 \ldots t_k$ is enabled at M; moreover we write $M[\sigma\rangle M'$ to denote the fact that the firing of σ from M yields to M'. The set of all sequences that are enabled at the initial marking M_0 is denoted with $L(N, M_0)$.

A marking M is said to be *reachable* in $\langle N, M_0\rangle$ iff there exists a firing sequence $\sigma \in L(N, M_0)$ such that $M_0[\sigma\rangle M$. The set of all markings reachable from M_0 defines the *reachability set* of $\langle N, M_0\rangle$ and is denoted with $R(N, M_0)$.

A Petri net is said *strongly connected* if there exists a directed path from any vertex (place or transition) to any other vertex (place or transition). The strongly connected components of a directed graph G are its maximal strongly connected subgraphs.

Let us recall two structural notions. A strongly connected component is said *transient* if its set of input transitions is included in its set of the output transitions. It is said *ergodic* if its set of output transitions is included in the set of its input transitions. We denote *Tr* and *Er* as the subset of places and transitions determining respectively a transient and an ergodic component.

A *synchronised PN* (David and Alla, 2004) is a structure $\langle N, M_0, E, f\rangle$ such that: i) $\langle N, M_0\rangle$ is a net system; ii) E is an alphabet of external events; iii) $f : T \rightarrow E$ is a labeling function that associates to each transition t an input event $f(t)$. This type of labelling function is called λ*-free* in the literature (Gaubert and Giua, 1999).

The set T_e of transitions associated to the input event e is defined as follows: $T_e = \{t \mid\ t \in T, f(t) = e\}$.

The previous syntactic definition is also common to the so-called *labeled PN* (Gaubert and Giua, 1999). However, while in the case of labelled PN the evolution is autonomous and the events are usually interpreted as outputs, in the case of synchronised nets the events are inputs that drive the net evolution as explained in the following. Furthermore as showed later, in a synchronised nets two or more transitions can simultaneously fire, while this is not possible for labelled nets.

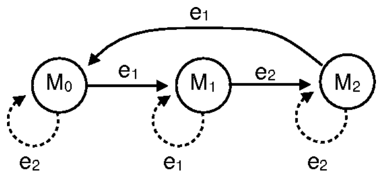

Fig. 2. The completely specified reachability graph of the PN in Fig. 1(a)

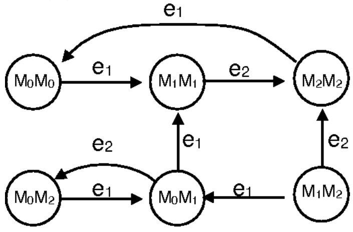

Fig. 3. The auxiliary graph of the PN in Fig. 1.(a).

Let M be the current marking and τ be the current time. A transition $t \in T$ fires at τ only if:

(1) it is enabled, i.e., $t \in \mathcal{E}(M)$;
(2) the event $e = f(t)$ occurs at time τ.

On the contrary, the occurrence of an event associated to a transition $t \notin \mathcal{E}(M)$ does not produce any firing.

In Fig. 1 is shown an example of synchronised PN. The label next to each transition denotes the transition name and the corresponding input event at once.

It can be possible to have two enabled transitions receptive to the same event. When this event occurs, both transitions are fireable and either both fire simultaneously, if there is no conflict between them, or have a non-determinism otherwise.

***Example** 4.* Consider the PN of Fig. 1(a) and let $M = [1\,0\,1]^T$ be the current marking. It is trivial to see that t_1 and t_3 are enabled and the occurrence of the event E_1 will make them fire leading to the marking $M' = [1\,1\,0]^T$. ■

As a consequence, the reachability set of a synchronised PN could be either a subset or equal to the reachable marking set of the underlying net, depending on the labelling function.

***Definition** 1. A synchronised PN system $\langle N, M_0\rangle$ is said to be deterministic if the following relation holds:*

$$\forall M \in R(N, M_0), \forall e \in E :\ M \geq \sum_{t \in T_e \bigcap \mathcal{E}(M)} Pre(\cdot, t). \quad ■$$

In others words, a synchronised PN is said to be deterministic if for all reachable markings there is no conflict between two or more enabled transitions that share the same label.

A net system $\langle N, M_0\rangle$ is said to be bounded if there exists a positive constant k such that for all $M \in R(N, M_0)$, $M(p) \leq k\ \forall p \in P$. A bounded net has a finite reachability set. In such a case, the behavior of the net can be represented by the *reachability graph*, a directed graph whose vertices correspond to reachable marking and whose edges correspond to the transitions that cause a change of marking. In the case of synchronised PN it is common to show the event next to the arc.

In Fig. 2, paying no attention to the dashed edges, it is shown the reachability graph of to the PN in Fig. 1(a).

In the rest of the paper, we only deal with the class of bounded PN.

3. Synchronizing Sequences for Bounded PN

The use of Petri nets offers significant advantages because of their intrinsically distributed nature where the notions of state (i.e., marking) and action (i.e., transition) are local. In this section, we will be concerned with synchronised and deterministic PN. Given a PN system $\langle N, M_0\rangle$, a straightforward approach to determine a SS consists in adapting the existing FSM approach to the reachability graph. This could be summarised it in the following steps:

- computation of the reachability graph G;
- computation of the auxiliary graph $\mathcal{A}(G)$;
- verification of the reachability condition.

It's easy to verify that this straightforward approach presents one shortcoming that makes it not always applicable: FSM approach requires the graph to be completely specified, while in a PN this condition is not usually true. In fact, the reachability graph of the PN is partially specified because from many marking not all the transitions are enabled. In order to use the mentioned approach it is consequently necessary to make its reachability graph G completely specified, obtaining $\tilde{G}$.

We obtain the graph $\tilde{G}$ by adding a self loop labelled $e = f(t)$, for every marking $M \in G$ and for every transition not enabled $t \notin \mathcal{E}(M)$.

We can summarise in the following algorithm the modified approach for PN:

***Algorithm** 5.* (Computing a SS leading to M_r). Let $\langle N, M_0\rangle$ be a deterministic bounded synchronised PN system and let $M_r \in R(N, M_0)$ be a desired final marking.

1. Let G be the reachability graph of $\langle N, M_0\rangle$

2. Let $\tilde{G}$ be the modified reachability graph obtained by completing G.

3. Construct the corresponding auxiliary graph $\mathcal{A}(\tilde{G})$.

4. A SS for the marking M_r, if such a sequence exists, is given by the direct application of Algorithm 1 to $\mathcal{A}(\tilde{G})$. ■

***Example** 6.* Consider the PN in Fig. 1.(a). The initial marking $M_0 = [2\ 0\ 0]^T$ enables only the transition t_1. Then for that marking we add a self loop labelled e_2 and so on. We obtain the completely specified reachability graph — taking into account also dashed edges — in Fig. 2, whose auxiliary graph is shown in Fig. 3. ■

In Fig. 2 and Fig. 3 are shown the reachability graph and the corresponding auxiliary graph of the PN in Fig. 1.(a). Note that dashed edges are added in order to make it completely specified.

We can now state the following theorem.

***Theorem** 7. A deterministic bounded synchronised PN $\langle N, M_0\rangle$ has a synchronizing sequence leading to a marking $M_r \in R(N, M_0)$ iff the reachability condition on its auxiliary graph $\mathcal{A}(G)$ is verified, i.e., there is a path from every node (M_i, M_j), with $M_i, M_j \in R(N, M_0)$, to node (M_r, M_r).*

Proof: **(only if)** If no path exists in the auxiliary graph from (M_i, M_j) to node (M_r, M_r), then by Theorem 2 no input sequences w exists such that $\exists \sigma_i, \sigma_j \in f^{-1}(w)|\ M_i[\sigma_i > M_r,\ M_j[\sigma_j > M_r$.

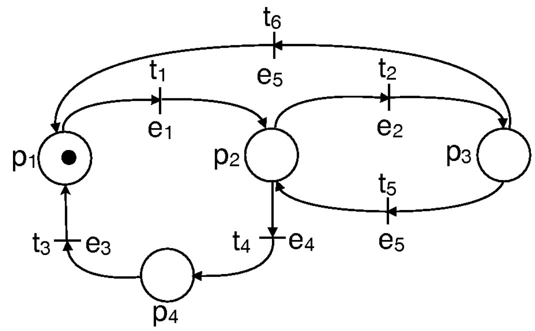

Fig. 4. A synchronised strongly connected PN.

(if) If in the auxiliary graph for each couple (M_i, M_j) there exists a path with input sequence w to some node (M_r, M_r), then by Theorem 2 in the completely specified reachability graph $\tilde{G}$ there exist a path labeled w from any mode M to M_r and this path is unique being the graph deterministic. Thus if the net is in any marking M, the input sequence w drives it to M_r. □

4. Synchronizing Sequences for State Machines

We now discuss the synchronizing sequences problem with reference to a class of PN, called *state machines* (SMs). The approach we propose does not require an exhaustive enumeration of the states in which the system may be. We will initially consider strongly connected PN; later we will relax this constraint to address the general case.

***Definition** 2. A* state machine *is a PN where each transition has exactly one input and one output place.* ■

In Fig.4 is shown an example of a synchronised SM. Note also that such a net is strongly connected.

When the initial marking assigns to this net a single token, it will move from place to place and the corresponding reachability graph will have as many marking as there are places. However, if the number of tokens in the initial marking is greater that one, the cardinality of the reachability graph may significantly increase. It can be shown in fact that for a strongly connected SM with m places and k tokens the cardinality of the reachability set is given by:

$$|R(N, M_0)| = \binom{m+k-1}{m-1} \leq \frac{1}{(m-1)!}k^{m-1}$$

In Table 2 — consider only the first three columns — it is shown the number of states of the reachability graph $|G|$ and of the modified auxiliary graph $|\mathcal{A}(\tilde{G})|$ of the net in Fig. 4 with $m = 4$ places, for different values of the total number of tokens k.

The forth and fifth columns show the computational times necessary to construct the reachability graph $|G|$ and the modified auxiliary graph $|\mathcal{A}(\tilde{G})|$ of the same net. The used functions, together with other MATLAB ones for observability and determination of synchronizing sequences on PN, can be downloaded at the web address in reference (Pocci, 2010).

We now define the concept of a path leading from a set of places $\hat{P}$ to a place p_r.

***Definition** 3. Given a SM $N = (P, T, Pre, Post)$, let $\gamma = p_{i_1} t_{i_1} p_{i_2} t_{i_2} \cdots p_{i_k} t_{i_k} p_{i_1}$ with $\hat{P} = \{p_{i_1}, p_{i_2}, \ldots, p_{i_k}\} \subseteq P$ be a directed cycle that touches all places in $\hat{P}$. Assume*

k	$\lvert G\rvert$	$\lvert\mathcal{A}(\tilde{G})\rvert$	t_G [s]	$t_{\mathcal{A}(\tilde{G})}$ [s]
1	4	10	0.001	0.002
2	10	55	0.002	0.059
3	20	210	0.006	0.285
4	35	630	0.021	1.087
5	56	1596	0.047	3.730
6	84	3570	0.108	11.41
7	120	7260	0.215	32.24
8	165	13695	0.418	79.96
9	220	24310	0.724	186.7
10	286	41041	1.233	411.1

Table 2. Cardinality and computational time of G and $\mathcal{A}(\tilde{G})$ of the PN in Fig. 4.

$p_{i_k} = p_r$ and let $\sigma = t_{i_1} \cdots t_{i_{k-1}}$ be the firing sequence obtained by opening this cycle in p_r and removing its output transition t_{i_k}. We define σ a synchronizing path *leading from $\hat{P}$ to p_r if the following two conditions are verified:*

C1) there do not exist two transitions $t, t' \in T$ such that $t \in \sigma$, $t' \notin \sigma$, and $f(t) = f(t')$;

C2) for all indices $j, h \in \{1, \dots, k-1\}$, if $p_{i_j} = p_{i_h}$ then $f(t_{i_j}) \neq f(t_{i_h})$. ■

Condition C1 implies that there do not exist any pair of transitions sharing the same event such that the first one belongs to the synchronizing path but not the second one.

Condition C2 states that if the synchronizing path is touching more than once the same place, then the output transitions will not have associated the same event.

***Proposition** 8. Let us consider a strongly connected synchronised SM $N = (P, T, Pre, Post)$ containing a single token. Let σ be a synchronizing path leading from P to a place p_r. It holds that $w = f(\sigma)$ is a SS for the PN that brings the token to place p_r.*

Proof: Let $\sigma = t_{i_1} \cdots t_{i_{k-1}}$ be the corresponding synchronizing path and $\gamma = p_{i_1} t_{i_1} p_{i_2} t_{i_2} \cdots p_{i_k} t_{i_k} p_{i_1}$ the corresponding cycle, that does not need to be elementary. It is assumed that $p_{i_k} = p_r$. Let w be the corresponding input event sequence such that $w = f(\sigma) = e_{i_1} \cdots e_{i_{k-1}}$. At the beginning, the event e_{i_1} drives the token to either p_{i_2} — if p_{i_1} was marked — or from p_{i_j} to $p_{i_{j+1}}$ — if p_{i_j} was marked and $f(t_{i_j}) = e_{i_1}$. At any rate, the token could only be in a place p_{i_k} such that $k > 1$. Afterwards, if p_{i_2} is marked, e_{i_2} drives the token to p_{i_3}. That is because, if p_{i_2} is not repeated in γ — it belongs to more than one elementary cycle —, C1) assures only transition belonging to the cycle to be receptive to e_{i_j}; thus the token can only be driven to p_{i_3}. Otherwise, C1) assures as previously the token to remain in γ whereas C2) assures the token not to come back in the previously marked places. If the tokens is in some of the other places p_{i_j} — $j > 2$ — , for the same reasoning it can only be driven to $p_{i_{j+1}}$. Thus, after the application of the event e_{i_j}, the token could only be in a place p_{i_k} such that $k > j$. □

Note that Condition C1) is sufficient to assure the sequence to be a synchronizing one if γ is an elementary cycle.

***Example** 9.* Let consider the PN in Fig. 4. We want to find a SS that leads the system to the marking $[0\,0\,1\,0]^T$. Let $\gamma = p_2 t_4 p_4 t_3 p_1 t_1 p_2 t_2 p_3 t_5 p_2$ be the direct cycle that touches all the places and $\sigma = t_4 t_3 t_1 t_2$ the synchronizing path for p_3, having ${}^\bullet t_5 = p_3$. It holds that $w = f(\sigma) = e_4 e_3 e_1 e_2$ is the searched SS. ■

We now give a method to obtain a SS in the case of multiple tokens.

***Proposition** 10. Let us consider a synchronised strongly connected SM $\langle N, M_0\rangle$ with k tokens. A SS for the PN, such that brings all the tokens to the place p_r, can be determined as follows. Let w be the SS leading to the place p_r determined for the case with only one token as in Proposition 8. It holds that w^k is a SS.*

Proof: Let w be a SS leading to the place p_r. By applying this sequence for the same reasoning of Proposition 8 at least one of the tokens will be driven to p_r. Any further application of w does not move the token from p_r as C1) assures every output transition for this place to be not receptive to any of the event $e_i \in w$. Thus w^k takes the k tokens at least one at time to the place p_r. □

***Example** 11.* Let consider the PN in Fig. 4. Let the PN have 2 tokens. We want to find a SS that leads the system to the marking $[0\,0\,2\,0]^T$. Let $w = e_4 e_3 e_1 e_2$ be the SS founded in Example 9 with one token. It holds that $w^2 = e_4 e_3 e_1 e_2 e_4 e_3 e_1 e_2$ is a SS for the desired final marking. ■

In the following, we extend our analyse by relaxing the assumption that a SM has to be strongly connected and we show how the computation of SS is possible by considering the same net with only one token using Proposition 10.

***Proposition** 12. Let us consider a synchronised SM $\langle N, M_0\rangle$ with two or more ergodic components. This net has a SS iff there is no marked transient component from where two paths leading to two different ergodic components can be determined.*

Proof: By definition of transient component, is known that if for a marking M none of the output transitions are fireable, then none of these transitions will ever be fireable form any marking obtained from M. Thus it is proved that the existence of such two paths implies that none of the possible sequences can synchronise tokens from one ergodic components to another one. Consecutively there does not exist any sequence able to synchronise the net to a final marking regardless of the initial one. □

It is possible to find a SS for a synchronised PN not strongly connected, depending on the initial marking.

***Proposition** 13. Let us consider a totally synchronised SM $\langle N, M_0\rangle$ with two or more ergodic components. It holds that if the only one initially marked component is an ergodic one then it exists a SS for all the markings M such that $M(p) = 0$ $\forall p$ is not belonging to such a component.*

Proof: It is easy to verify that for $M_0(p) = 0$ $\forall p$ not belonging to one ergodic component the set $R(N, M_0)$ determines a unique strongly connected component for whose markings a SS can be trivially determined by applying Proposition 8 to the subnet induced by that component. □

We now give a method to obtain a SS in a synchronised PN not strongly connected for the case of one token.

***Proposition** 14. Let us consider a totally synchronised SM $\langle N, M_0\rangle$ with a unique ergodic component Er and a set of transient components $Tr_1, Tr_2, \dots, Tr_S$. A SS for the PN, such that brings the unique token to the place $p_i \in Er$, can be determined concatenating w_1 and w_2.*

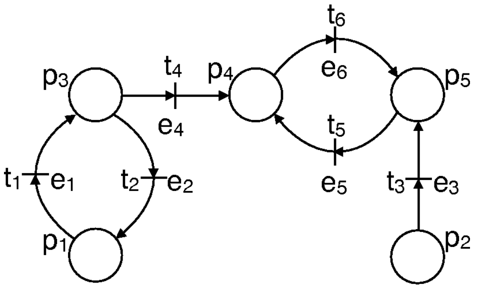

Fig. 5. An example of synchronised PN not strongly connected.

These two sequences are constructed as follows. Let $\sigma_1 = \sigma_1^1 t_1 \ldots \sigma_1^S t_S$*, where the couple* (σ_1^i, t_i) *is determined as follows: let* t_i *be the transition from some* $p' \in Tr_i$ *to some* $p'' \in Er$ *such that there do not exist any* $t' \in p'^{\bullet}$ *leading to* Tr_i *such that* $f(t') = f(t_i)$*; let* σ_1^i *be the synchronizing path from the set of all places belonging to* Tr *leading to that place* p'*. Let* σ_2 *be the synchronizing path from the set of all places belonging to* Er *leading to the place* p_i*. Let* $w_1 = f(\sigma_1)$ *and* $w_2 = f(\sigma_2)$*.*

Proof: By definition of synchronizing path the input sequences corresponding to each couple (σ_1^i, t_i) make the token entering the unique ergodic component. The application of the input sequence w_2 matches with the problem already solved in proposition 8 wrt to the subnet induced by the ergodic component. □

***Example** 15.* Let us consider again the net in Example 5. Let the initial marking be $M_0 = [0\,1\,0\,0\,0]^T$. We want to find a SS that leads the system to the marking $M = [0\,0\,0\,1\,0]^T$. Let it be $(\sigma_1^1, t_1) = (t_1, t_4)$ and $(\sigma_1^2, t_2) = (\varepsilon, t_3)$, it holds that $\sigma_1 = t_1 t_4 t_3$. Let $\sigma_2 = t_5$ be the synchronizing path for the ergodic component and the place p_4, having ${}^{\bullet}t_6 = p_4$. It holds that $w = f(\sigma_1 \sigma_2) = e_1 e_4 e_3 e_5$ is the searched SS. ■

5. Conclusion

In this paper, we have shown how Mealy machines techniques can be applied easily to the class of bounded synchronised PN, by using some arrangement. We also proposed a method that allows to determine a SS for a bounded synchronised PN by only looking at its static structure. Even in the case of multiple tokens, the existence of SS can be efficiently determined avoiding the state explosion problem.

Our approach uses synchronised PN that can have two o more transitions sharing the same event. This case introduces a nondeterminism, having a PN that in response to the same input events sequence can produce different firing sequences, reaching different markings.

A future work will be to extend our approach to synchronised PN considering the presence of *always occurring event*, the neutral event of E^*. In this case we lead with the problem of unstable markings, those markings whose outgoing transitions are receptive to the always occurring event that immediately fires.

References

David, R. and Alla, H. (2004). *Discrete, Continuous and Hybrid Petri Nets*. Springer-Verlag.

Eppstein, D. (1990). Reset sequences for monotonic automata. *SIAM J. Computing*, 19, 500–510.

G.-V. Jourdan and G.v. Bochmann (2009). On testing 1-safe Petri nets. In *Theoretical Aspects of Software Engineering, 2009. TASE 2009. Third IEEE International Symposium on*, 275 –281. doi:10.1109/TASE.2009.20.

Gaubert, S. and Giua, A. (1999). Petri net languages and infinite subsets of $\mathcal{N}^m$. *J. of Computer and System Sciences*, 59(3), 373–391.

Giua, A. and Seatzu, C. (2002). Observability of place/transition nets. *Automatic Control, IEEE Transactions on*, 47(9), 1424 – 1437. doi: 10.1109/TAC.2002.802769.

H. Zhu and X. He (2000). A theory of testing high level Petri nets. In *In Proceedings of the International Conference on Software: Theory and Practice, 16th IFIP World Computer Congress*, 443–450.

Hennie, F. (1968). *Finite-State Models for Logical Machines*. New York: John Wiley, 2 edition.

Kohavi, Z. (1978). *Switching and Finite Automata Theory*. The McGraw-Hill College, 2 edition.

Lee, D. and Yannakakis, M. (1996). Principles and methods of testing finite state machine – a survey. *Proceedings of the IEEE*, 84(8), 1090–1123.

Moore, E.F. (1956). Gedanken-experiments on sequential machines. *Automata Studies, Annals of Mathematical Studies*, (34), 129 – 153.

Murata, T. (1989). Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4), 541 –580. doi:10.1109/5.24143.

Natarajan, B.K. (1986). An algorithmic approach to the automated design of parts orienters. In *SFCS '86: Proc. of the 27th Annual Symposium on Foundations of Computer Science*, 132–142. IEEE Computer Society, Washington, DC, USA. doi: http://dx.doi.org/10.1109/SFCS.1986.5.

Pixley, C., Jeong, S.W., and Hachtel, G. (1994). Exact calculation of synchronizing sequences based on binary decision diagrams. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 13(8), 1024 –1034. doi:10.1109/43.298038.

Pocci, M. (2010). Finite state machines and Petri nets synchronisation toolbox. `http://www.lsis.org/poccim/SYNCH_TOOL.zip`.

Pomeranz, I. and Reddy, S. (1994). Application of homing sequences to synchronous sequential circuit testing. *Computers, IEEE Transactions on*, 43(5), 569 –580. doi: 10.1109/12.280804.

Rho, J.K., Somenzi, F., and Pixley, C. (1993). Minimum length synchronizing sequences of finite state machine. In *Design Automation, 1993. 30th Conference on*, 463 – 468.

Yamalidou, E., Adamides, E., and Bovin, D. (1992). Optimal failure recovery in batch processing using Petri net models. *Proceedings of the 1992 American Control Conference*, 3, 1906–1910.

Yamalidou, E. and Kantor, J. (1991). Modeling and optimal control of discrete-event chemical processes using Petri nets. *Computers & Chemical Engineering*, 15(7), 503 – 519. doi:DOI: 10.1016/0098-1354(91)85029-T.

Zhu, H. and He, X. (2002). A methodology of testing high-level Petri nets. *Information and Software Technology*, 44(8), 473 – 489. doi:DOI: 10.1016/S0950-5849(02)00048-4.