Computation of synchronizing sequences for a class of 1-place-unbounded synchronized Petri nets

Changshun Wu, Isabel Demongodin, Alessandro Giua

Computation of synchronizing sequences for a class of 1-place-unbounded synchronized Petri nets

Changshun WU[1], Isabel DEMONGODIN[1] and Alessandro GIUA[2]

***Abstract*— In this paper, we consider a special class of synchronized Petri nets, called 1-place-unbounded, that contain a single unbounded place. The infinite reachability spaces of such nets can be characterized by two types of finite graphs, called improved modified coverability graph and weighted automata with safety conditions. In the case of that these two finite graphs are deterministic, we develop computation algorithms for synchronizing sequences for this class of nets.**

I. INTRODUCTION

One of the fundamental testing problems for discrete event systems (DESs) is the identification of a final state, i.e., given a system whose current state is unknown, find an input sequence that can drive it to a known state. *Synchronizing sequences* (SSs), without output information, and *homing sequences* (HSs), with output information, are two conventional solutions to this problem and a comprehensive survey can be found in [1]. Most of the literature on this topic uses as model finite state machines[2][3].

Petri net is a graphical and mathematical formalism that has also been widely used to model DESs. However there exist few works on the use of Petri nets for testing. The problem of computing SSs for Petri nets with input events, called *synchronized Petri nets* (SynPNs), was recently addressed. [4] focused on bounded SynPNs and provided general algorithm for computing SSs based on the reachability graph of a net, showing that for particular classes of nets more efficient algorithms based on the net structure exist. In a later work [5] the same authors extended the synchronization problem to classes of unbounded nets, but the algorithm they proposed to construct a modified coverability graph for unbounded nets is not guaranteed to provide an exact description of its behavior, and thus may not always be used to compute SSs. In particular using the *modified coverability graph* (MCG) in [5] there may exist *spurious* markings (i.e., markings in the graph that are not reachable in the net) or *vanishing* markings (i.e., markings that are reachable in the net but are not represented in the graph).

We are also interested in the analysis of unbounded SynPNs for using them as models for synchronization (and more generally testing) problems. However the analysis of systems with an infinite state space is a challenging problem. Recently, [6] have investigated the synchronization problem for *weighted automata* (WA). WA are a special class of infinite state systems, whose state is defined by a finite location and a quantitative weight (or energy) and whose behavior may be subject to quantitative constraints on the energy value. These authors have shown that the existence of sequences synchronizing a deterministic WA to a known state with safety conditions or to a known location with or without safety condition is decidable. Motivated by this, the authors in [7] considered a particular class of SynPNs, called *1-place-unbounded SynPNs*, in which a single place can be unbounded and they found out that this class of SynPNs can be converted into equivalent WAs with safety conditions and the infinite reachability spaces of this class of SynPNs can be exactly represented by a finite graph, called *improved modified coverability graph* (IMCG).

In this paper we develop computation algorithms for two location synchronization problems in the case either the IMCG or the WA is deterministic: synchronization into a single node and synchronization into a subset of nodes of these two graphs. Then we show that this approach can be used to solve two different problems related to the original net: the synchronization into a single node of the graph of its IMCG or WA corresponds to synchronizing the net into a known marking of bounded places and a known set of enabled transitions, while the synchronization into a subset of nodes of the graph of its IMCG or WA with same marking on specified bounded places corresponds to synchronizing the net into a known marking of specified places. The advantage of these computation algorithms are reducing the computation on the global graphs (IMCGs or WAs) to the one on the smaller subgraph: the ergodic strongly connected component (SCC), which can reduce the computational effort and furthermore can also be applied when the converted deterministic IMCG or WA is not strongly connected.

The paper is structured as follows. Section 2 presents the basic formalism on synchronized Petri nets, improved modified coverability graphs and weighted automata. Section 3 formalize the synchronization problem for synchronized Petri nets. Section 4 presents the computation algorithms of SSs on the deterministic IMCG and WA of a given 1-place-unbounded SynPN. Conclusion and future works are summarized in the last section.

II. SYNCHRONIZED PETRI NETS

In this section, we present the basic notions concerning SynPNs and its coverability graph and WA. Most of them are taken from [5] and [6]. For a comprehensive introduction to Petri nets and weighted automata, see [8], [9].

*This work was not supported by any organization

[1]Changshun WU and Isabel DEMONGODIN are with Aix-Marseille Université, CNRS, ENSAM, Université de Toulon, LIS UMR 7020, France {changshun.wu, isabel.demongodin}@lis-lab.fr

[2]Alessandro GIUA is with University of Cagliari, DIEE, Italy giua@diee.unica.it

A. Synchronized Petri nets

In the following, $\mathbb{Z}$ and $\mathbb{N}$ denote the set of integers and nonnegative integers, respectively.

A *place/transition* net (PN) is a structure *N=(P, T, Pre, Post)*, where P is a set of m places, T is a set of n transitions, $Pre : P \times T \to \mathbb{N}$ and $Post : P \times T \to \mathbb{N}$ are the $pre-$ and $post-$ incidence matrixes that specify the weights of directed arcs from places to transitions and vice versa. $C = Post - Pre$ is the incidence matrix. A *marking* is a mapping $M : P \to \mathbb{N}$ that assigns to each place of a net a nonnegative integer. A marking is denoted by a vector and the marking of a place is represented by $M(p)$. A *marked* PN $\langle N, M_0\rangle$ is a net N with an initial marking M_0. A transition t is enabled at M iff $M \geq Pre(\cdot, t)$ and its firing yields the marking $M' = M + C(\cdot, t)$. The set of all enabled transitions at M is denoted by $\varepsilon(M)$. We write $M[\sigma\rangle$ to denote that the sequence of transitions $\sigma = t_1 \cdots t_k$ is enabled at M. Moreover $M[\sigma\rangle M'$ denotes the fact that the firing of σ from M leads to M'. A marking M is said *reachable* in $\langle N, M_0\rangle$ iff there exists a firing sequence $M_0[\sigma\rangle M$. The set of all markings reachable from M_0 defines the *reachability set* of $\langle N, M_0\rangle$ and is denoted $R(N, M_0)$. A place is *k-bounded* for a given $k > 0$ if $\forall M \in R(N, M_0)$, $M(p) \leq k$. A marked PN $\langle N, M_0\rangle$ is said to be $k - bounded$ if all places are k-bounded and the marked PN is *unbounded* if $\nexists k \in \mathbb{N}$ such that the net is k-bounded. Notation P_b and P_u denote, respectively, the set of bounded and unbounded places with $P_b \cup P_u = P$ and $P_b \cap P_u = \emptyset$. $M \uparrow_b$ and $M \uparrow_u$ are the *projection* of the marking M onto the set of bounded places P_b and unbounded places P_u, respectively, with $M \uparrow_b + M \uparrow_u = M$. More precisely, $M \uparrow_b (p_i) = M(p_i)$ if $p_i \in P_b$ else $M \uparrow_b (p_i) = 0$ and $M \uparrow_u (p_i) = M(p_i)$ if $p_i \in P_u$ else $M \uparrow_u (p_i) = 0$. We denote ${}^\bullet p$ and $p^\bullet$ the *preset* and *postset* of a place p, respectively: ${}^\bullet p = \{t \in T | Post(p, t) > 0\}$ and $p^\bullet = \{t \in T | Pre(p, t) > 0\}$. The set of input transitions and the set of output transitions for a set of place P' are respectively, defined as: ${}^\bullet P' = \{t \in T \mid (\exists p \in P')\ t \in {}^\bullet p\}$ and $P'^\bullet = \{t \in T \mid (\exists p \in P')\ t \in p^\bullet\}$.

A *synchronized Petri net* (SynPN) is a structure $\langle N, E, f\rangle$ such that: i) N is a PN; ii) E is an alphabet of input events; iii) $f : T \to E$ is a labeling function that associates with each transition t an input event $f(t)$. A *marked synchronized PN* $\langle N, E, f, M_0\rangle$ is a SynPN with an initial marking M_0. We denote the set of transitions associated with the input event e by: $T_e = \{t \in T | f(t) = e\}$ and the set of enabled transitions associated with event e at marking M as: $\varepsilon_e(M) = T_e \cap \varepsilon(M)$. The evolution of a synchronized net is driven by the occurrence of an input event sequence that produces a set of transition firings. At marking M, transition $t \in T$ is fired only if:

1) transition t is enabled, i.e., $t \in \varepsilon(M)$;
2) the event $e = f(t)$ occurs.

Note that the occurrence of an input event $e \in E$ at marking M forces the simultaneous firing of all transitions in $\varepsilon_e(M)$ provided there are no conflicts among them. On the contrary, the occurrence of an event e does not produce the firing of a non enabled transition $t \in T_e$.

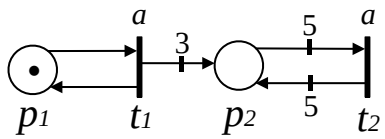

Fig. 1. A 1-place-unbounded synchronized Petri net.

Definition 1: (Deterministic synchronized PN) A marked synchronized PN $\langle N, E, f, M_0\rangle$ is said to be deterministic if the following condition holds:

$$(\forall M \in R(N, M_0))\ \ (\forall e \in E),\quad M \geq \sum_{t \in \varepsilon_e(M)} Pre(\cdot, t)$$

A sufficient structural condition to ensure determinism is the following.

Assumption 1: Given a synchronized PN $\langle N, E, f\rangle$ we assume there exist no place p such that $t, t' \in p^\bullet$ and $f(t) = f(t')$.

In the rest of this paper, for the sake of simplicity, we will focus on deterministic synchronized PNs that satisfy Assumption 1. We point out, however, that for the class of 1-place unbounded SynPNs considered in this paper, the improved modified coverability graph (IMCG) we introduce in [7] can be used to verify if the net is deterministic (necessary and sufficient condition).

Definition 2: (Evolution of a deterministic synchronized PN) In a deterministic synchronized PN, when an input event e occurs at a marking M, all enabled transitions associated with this event, $\varepsilon_e(M) = T_e \cap \varepsilon(M)$, fire simultaneously in a single step $e|\tau$:

$$M[e|\tau\rangle M', \quad with\ \tau = \varepsilon_e(M)$$

$$M' = M + \sum_{t \in \tau} (Post(\cdot, t) - Pre(\cdot, t))$$

Here $M[e|\tau\rangle M'$ denotes that the occurrence of the input event e at M yields marking M' by the firing of τ.

Example 1: Consider the SynPN in Fig.1 satisfying Assumption 1, where $P = \{p_1, p_2\}$, $T = \{t_1, t_2\}$, $E = \{a\}$, $f(t_1) = f(t_2) = a$. Let $M_0 = [1\ 0]^T$ be the initial marking. At M_0, the sets of enabled transitions are: $\varepsilon_a(M_0) = \{t_1\}$. If a occurs at M_0, step $a|\{t_1\}$ fires leading to the new marking $M_1 = [1\ 3]^T$, i.e., $M_0[a|\{t_1\}\rangle M_1$. If a continue to occur at M_1, the net reaches a new marking $M_2 = [1\ 6]^T$, at which the occurrence of a makes step $a|\{t_1, t_2\}$ fire leading a new marking $M_2 = [1\ 9]^T$.

B. Improved modified coverability graph

The infinite state space of a 1-place-unbounded SynPN can be exactly represented by a finite graph, called improved modified coverability graph (IMCG). Here we recall this technique proposed in [7] by means of a simple example. First we introduce two basic notions used in such graph, called ω-number and ω-vector.

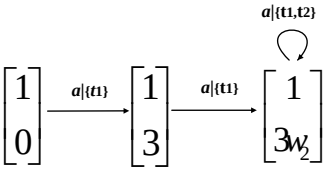

Fig. 2. The IMCG of the net in Fig.1.

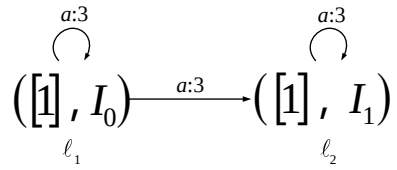

Fig. 3. The equivalent WA of the net in Fig.1.

Definition 3: (ω-*number*) Let $n \in \mathbb{Z}$, $k, q \in \mathbb{N}$, and $0 \leq q < k$. Then $\mathbb{S} = \{(k \cdot i + q) | i \in \mathbb{Z} \wedge i \geq n\}$ is called an ω-*number*, with k as its base, n the lower bound, and q the remainder. An element in $\mathbb{S}$ is called an instance of $\mathbb{S}$, and the minimal instance is $S_{min} = k \cdot n + q$. For the sake of simplicity, we denote $\mathbb{S} = k\omega_n + q$.

For instance, $\mathbb{S} = 3\omega_0 + 1 = \{(3 \cdot i + 1) | i \geq 0\} = \{1, 4, 7, \cdots\}$, where the base is 3, the lower bound is 0 and the remainder is 1.

Definition 4: (ω-*vector*): A m-dimension vector $\mathrm{V} = [x_1 \ x_2 \ \cdots \ x_m]^T$ is called an ω-*vector* if at least one of its components is an ω-number, else it is an ordinary vector.

For example, $[1 \ 0 \ 3\omega_2]^T$ is an ω-vector and called ω-marking in the nets.

Example 2: The improved modified coverability graph of the SynPN in Fig.1 with $M_0 = [1 \ 0]^T$, is shown in Fig.2. In the following, notations "\" and "%" are, respectively, used for quotient and remainder operations.

The initial node of IMCG is q_0 with its marking $M_{q_0} = [1 \ 0]^T$. Consider event a at q_0, it will generate a new node q_1 with $M_{q_1} = [1 \ 3]^T$ after the firing of step $a|t_1$. Although M_{q_1} is strictly greater than the marking $[1 \ 0]^T$, M_{p_2} is not enough bigger that the weight of the arc from p_2. Continuing to apply event a at node q_1, a new node q_2 is generated with marking $M_{q_2} = [1 \ 6]^T$ which is strictly greater than the marking $[1 \ 0]^T$ on the path from q_2 to the initial node. By verifying that a is an increasing sequence which can be repeated infinitely from node q_0 by adding 3 tokens on place p_2, we update the marking of p_2 with an ω-number $k\omega_n + r$ where $k = M_{q_2}(p_2) - M_{q_1}(p_2) = 3$, $n = M_{q_2}(p_2) \backslash k = 2$, and $r = M_{q_2}(p_2) \% k = 1$. Thus the new marking of q_2 is $[0 \ 3\omega_2]$. Finally, we obtain the IMCG depicted in Fig.2 for the net in Fig.1.

C. Weighted automata

A weighted automaton is an automaton endowed with an energy level that is updated by the transition firing. In addition, the energy level must satisfy certain constraints that depend on the automaton discrete location.

According to [9] and [6], a *weighted automaton with safety conditions* is a 7-tuple $\mathcal{A} = \langle L, \Sigma, \Delta, \mathcal{I}, Safe, \ell_0, \rho_0 \rangle$, where L is a finite set of locations; Σ is a finite alphabet; $\Delta \subseteq L \times \Sigma \times \mathbb{Z} \times L$ is a set of edges; $\mathcal{I}$ is a finite set of intervals with integer or infinite endpoints; $Safe : L \to \mathcal{I}$ is the safety condition function which defines the energy scope of each location; $\ell_0 \in L$ and $\rho_0 \in \mathbb{N}$ are the initial location and the initial energy, respectively. A state (ℓ, ρ) is *safe* if and only if the energy ρ belongs to the safety condition of $Safe(\ell)$, i.e., $\rho \in Safe(\ell)$.

We denote by $\ell \xrightarrow{a:z} \ell'$, the occurrence of event a at location ℓ that moves the automaton to location ℓ' and updates the value of the energy from ρ to ρ' such that $\rho' = \rho + z$. Let Θ be the state space of a weighted automaton and a state $\theta = (\ell, \rho) \in \Theta$. For event $a \in \Sigma$, let $\delta(\theta, a) = \{\theta' : \exists (\ell, a, z, l') \in \Delta\}$. For a sequence of events, $w \in \Sigma^*$, we recursively define $\delta^*(\theta, aw) = \delta(\delta(\theta, a), w)$.

Definition 5: A weighted automaton $\mathcal{A}$ is deterministic if for all edges (ℓ, a, z_1, ℓ_1), $(\ell, b, z_2, \ell_2) \in \Delta$, if $a = b$, then $z_1 = z_2$ and $\ell_1 = \ell_2$, i.e., for all locations $\ell \in L$ and all input event $a \in E$, there must exist a single edge exiting location ℓ and labeled a.

Example 3: Consider the weighted automaton in Fig.3, where $L = \{\ell_1, \ell_2\}$, $\Sigma = \{a, b\}$, $\mathcal{I} = \{I_0 = [0, 4], I_1 = [5, +\infty)\}$, $Safe(\ell_0) = I_0$ and $Safe(\ell_1) = I_1$. This WA is non-deterministic because there are two arcs labeled with a outgoing from ℓ_1. If the initial location is ℓ_1 and the initial energy is $\rho_0 = 0$, then the initial state is $\theta_0 = (\ell_1, 0)$. At θ_0, the occurrence of event a yields a new state $\theta_1 = (\ell_1, 3)$ but not $\theta_1'(\ell_2, 3)$ because this does not respect the safety condition of ℓ_2, as $3 \notin Safe(\ell_2) = [5, +\infty)$.

For a 1-place-unbounded SynPN $\mathcal{PN}$, the authors in [7] have proven that it can be converted into an equivalent WA $\mathcal{A}$ with safety conditions. Let $R = \{r_1, \cdots, r_k\}$ with $r_i < r_j$ for $1 \leq i < j \leq k$ be the set of weights on the arcs from the unique unbounded place p_u to its post transitions of $\mathcal{PN}$, then the set of safety conditions of the equivalent $\mathcal{A}$ is $Safe$=$\{ \ I_0 = [0, w_1 - 1], \ I_1 = [w_1, w_2 - 1], \ \cdots, \ I_{k-1} = [w_{k-1}, w_k - 1], \ I_k = [w_k, +\infty) \ \}$. For instance, the WA in Fig.3 is the equivalent WA of the 1-place-unbounded SynPN in Fig.1. Due to the fact that there is only one arc outgoing from unbounded place p_2 and its weight is 5, the set of safety conditions is $\{I_0 = [1, 4], I_1 = [5, +\infty)\}$.

III. Synchronization problem for Synchronized Petri nets

In this section, we formalize the general synchronization problems for synchronized Petri nets.

Problem 1: (*Synchronization for SynPNs*) Given a marked SynPN $\langle N, E, f, M_0 \rangle$ with its reachability set $R(N, M_0)$, a target subset of reachable markings $\mathcal{M}_{tar}$ and an initial marking uncertainty $\mathcal{M}_{uncer}$, find a sequence of input events $w \in E^*$ such that from any marking in $\mathcal{M}_{uncer}$ the application of this sequence yields a marking in $\mathcal{M}_{tar}$. Such a sequence is called a *synchronizing sequence* (from $\mathcal{M}_{uncer}$ to $\mathcal{M}_{tar}$).

In plain words, the marking of the given net will belongs to $\mathcal{M}_{tar}$ after applying such sequences on the net whatever the starting marking in $\mathcal{M}_{uncer}$.

One applicable case of this setting is that the non-autonomous systems modeled by synchronized nets lost observation for a while and the current state may be any one in a subset of states reached from initial state, then one aims to find some inputs to drive the systems into a certain state or a set of states with some properties.

As we previously stated in last section, the IMCGs and WAs well characterize the infinite state spaces and dynamic behaviours of 1-place-unbounded SynPNs. Moreover they are finite. Thus we investigate the synchronization problem for 1-place-unbounded SynPNs based on its IMCG and WA.

In this paper, we assume that the initial state uncertainty is the entire reachability set, i.e., $\mathcal{M}_{uncer} = R(N, M_0)$ and consider two scenarios for the target subsets of reachable markings.

1) the subset of reachable markings represented by a node of the IMCG or the WA. Consider a 1-place-unbounded SynPN. If we can find a SS into a single node of its IMCG, then we can know the exact marking of bounded places and the set to which the marking of the single unbounded place belongs is a ω-number. If we can find a SS into a single node of its WA, then we can know the exact marking of bounded places and the set to which the marking of the single unbounded place belongs is an interval of integers.
2) the subset of reachable markings represented by a subset of nodes with same properties of the IMCG or the WA. Consider a 1-place-unbounded SynPN and a subset of bounded places P_s. If we can find a SS into a set of nodes of its IMCG or WA with the same marking of places in P_s, then we can know the marking of places in P_s.

IV. Computation of Synchronizing sequences for nets based on IMCGs and WAs

Consider a 1-place-unbounded SynPN. If its IMCG is deterministic, this graph can be seen as a deterministic finite state machine (FSM) where the input label associated to each transition is the input event labeling the corresponding net transition. In a similar way, a deterministic WA equivalent to a 1-place-unbounded SynPN can also be seen as a deterministic FSM.

In this section we will consider the problem of determining SSs for these deterministic FSMs corresponding to a given SynPN and later we will show how these sequences can be used to solve synchronization problems for the original SynPN. We first recall the definition of FSMs and the notions of SSs for FSMs.

Definition 6: (Finite state machine) A deterministic *FSM* G is a 5-tuple denoted by $G = (X, I, O, \delta, \lambda)$ where X, I and O are finite sets of states, input events and output events respectively; $\delta : X \times I \to X$ is the transition function, $\lambda : X \times I \to O$ is the output function.

Note that we assume functions δ and λ are completely defined, i.e., for any state, the occurrence of any event causes a transition producing a output. However, the output of FSMs is not considered in this paper. When the machine is in a current state $x \in X$ and receives an input event a from I, it will move to the next state specified by transition function $\delta(x, a)$. For a sequence of events, $w \in \Sigma^*$, we recursively define $\delta(\theta, aw) = \delta(\delta(\theta, a), w)$. Function δ can also be extended to a set of states, $X' \subseteq X$ as following: $\delta(X', i) = \{\delta(x, i) : x \in X'\}$.

Next we recall the notion of SSs in deterministic FSMs from [1].

Definition 7: (SSs for FSMs) Given a deterministic FSM $G = (X, I, O, \delta, \lambda)$ over the alphabet I and a subset $X' \subseteq X$, a sequence of input events $w \in I^*$ (respectively, $z \in I^*$) is a SS for G into a single state (respectively, into a subset of states X') if $\delta(X, w)$ is a singleton (respectively, $\delta(X, z) \subseteq X'$).

Namely SSs are sequences which can drive a given FSM into a final state or a desired subset of states regardless of the initial state. In [1], it was shown that the existence problems of SSs into a single state and into a specified subset of states are, respectively, NLOGSPACE and PSPACE-complete.

The basic idea for synchronizing a given FSM into a single state when the initial uncertainty is the whole state space consists in searching for merging sequences for any two pair states.

Definition 8: (*Merging sequences*) A merging sequence for two states $x, x' \in X$ is a sequence $w \in I^*$ such that $\delta(x, w) = \delta(x', w)$.

Definition 9: (*Auxiliary graph*) Given a FSM G with n states, let $Aux(G)$ be its auxiliary graph. $Aux(G)$ contains $n(n+1)/2$ nodes, one for every unordered pair (x', x'') of states of G, including pairs (x, x) of identical states. There exists an edge from node (x', x'') to $(\hat{x}', \hat{x}'')$ labeled with input event $a \in I$ iff $\delta(x', e) = \hat{x}'$ and $\delta(x'', e) = \hat{x}''$.

The merging sequence of two different states x and y corresponds to the path from the node (x, y) to an identical node in the auxiliary graph.

Algorithm 1: [1] **Computation of synchronizing sequences into a single state for FSMs.**
Input: *a FSM* $G = (X, I, O, \delta, \lambda)$.
Output: *a synchronizing sequence* w.

1. $w := \varepsilon$,
2. ***while*** $|\delta(X, w)| > 1$,
   2.1. *find two different states* x *and* $y \in \delta(X, w)$,
   2.2. *let* w' *be a merging sequence for* x *and* y,
      2.2.1. ***if*** *none exists, return* ***failure***.
      2.2.2. ***else*** $w := ww'$.
3. *return* w.

For synchronizing a given FSM into a subset of states the following algorithms can be used.

Algorithm 2: [1] **Subset construction algorithm.**
Input: A FSM $G = (X, I, O, \delta, \lambda)$ and a subset of states $X' \subseteq X$.
Output: A synchronizing sequence into subset X'.

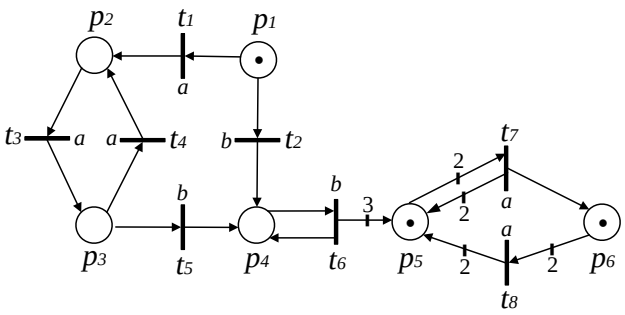

Fig. 4. A synchronized Petri net.

1. Set the initial state $v_0 = X$ as the root and label it new;
2. While there exist a node v tagged "new" do

2.1. For all $a \in I$:

2.1.1. $v_{new} = \delta(v, a)$;

2.1.2. if $v_{new} \subseteq X'$, return w the sequence of input events along the path from initial node v_0 to v_{new}, else if v_{new} already exists in the tree,

2.1.2.1. tag it "duplicate", else

2.1.2.2. tag it "new".

2.2. Untag node v.

Now we recall some basic notions from graph theory. A directed graph is called *strongly connected* if there exists a directed path from each of its nodes to any other. A *strongly connected component* (SCC) of a directed graph is a maximal strongly connected subgraph. According to [4], such a component is called *ergodic* (denoted by C_{er}) if the set of its output arcs is a subset of the set of its input arcs, otherwise is called *transient* (denoted by C_{tr}). The strongly connected components of a directed graph can be computed in linear time. One of the main computation algorithms is *Tarjan's algorithm* [10].

Definition 10: (*Condensed graph*) For a given directed graph, if each strongly connected component is contracted to a single vertex, the resulting graph is a directed acyclic graph, called condensed graph.

Example 4: Consider the WA in Fig. 5 with 4 SCCs. The set of transient components is $\{C_1, C_2, C_3\}$, while the set of ergodic SCCs is $\{C_4\}$.

Now we introduce a notion about the distance between a transient SCC and the ergodic SCCs of a given directed graph.

Definition 11: (*Distance to ergodic components*) Given a transient SCC C^i_{tr} of a deterministic FSM, we define the distance of the component C^i_{tr} to the ergodic components, denoted by $D(C^i_{tr})$, as the maximum length among all paths that start from C^i_{tr} and end in an ergodic component in the condensed graph.

Example 5: Consider the WA in Fig. 5. It is evident $D(C_3) = 1$ while $D(C_1) = 3$ because the maximum path from C_1 to C_4 is $C_1C_2C_3C_4$.

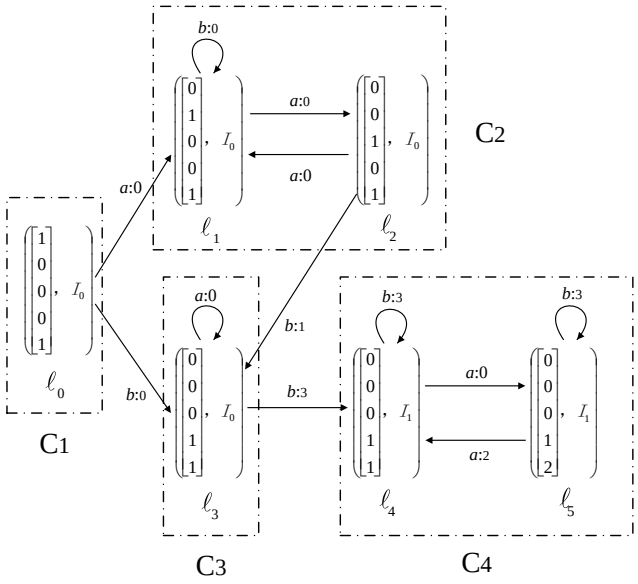

Fig. 5. The equivalent WA of the net in Fig.4.

A. Leading-out sequences for transient SCCs

In this subsection, we first introduce a notion of leading-out sequences (LoSs) for the transient SCCs of FSMs and then show how a well order composition of such sequences can be a SS into the set of states in the ergodic SCCs. The formal definition is as following.

Definition 12: (*Leading-out sequences*) Given a transient SCC C^i_{tr} of a deterministic FSM, denote v and V, respectively, a node and the set of nodes of C^i_{tr}. The sequences of input events w and z are, respectively, the leading-out sequence for v and C^i_{tr} if $\delta(v, w) \notin V$ and $\delta(V, w) \cap V = \phi$, respectively.

In plain word, the LoSs drive a node or a set of nodes of a given transient SCC to the outside of this transient component. Next example clarifies this notion.

Example 6: Consider the transient SCC C_2 consisting of ℓ_1 and ℓ_2 in Fig. 5. Sequences b and ab are, respectively, the LoS for state ℓ_2 and C_2, because $\delta(\ell_2, b) = \ell_3 \notin \{\ell_1, \ell_2\}$ and $\delta(\{\ell_1, \ell_2\}, bab) \cap \{\ell_1, \ell_2\} = \{\ell_3, \ell_4\} \cap \{\ell_1, \ell_2\} = \phi$.

Now we discuss the existence of LoSs.

Theorem 1: For all transient SCC C^i_{tr} with V the set of its states of a given deterministic FSM and all state $v \in V$ of C^i_{tr}, there exist LoSs for v and C^i_{tr}.

Proof: We prove this theorem in two parts as follows. First, we prove that there exist LoSs for all states v of all transient SCCs C^i_{tr} of a given deterministic FSM. By the definition of transient SCC, for all C^i_{tr}, there must exist at least one state, called outgoing state and denoted by v_{out}, which has at least one outgoing arc, denoted by a, pointing to the outside states of this SCC, i.e., $\delta(v_{out}, a) \notin V$. Because v and v_{out} are in the same SCC, there must exists a path w' from v to v_{out}, i.e., $\delta(v, w') = v_{out}$. Due to $\delta(v, w'a)$=$\delta(\delta(v, w'), a)$=$\delta(v_{out}, a)$ and $\delta(v_{out}, a) \notin V$, we have $\delta(v, w'a) \notin V$, i.e. sequence $w'a$ is the LoS for state v. Thus first part has been proved.

Second, we prove that there exist LoSs for all transient SCCs C_{tr}^i of a given deterministic FSM. Due to the determinism of FSMs, for a set of states V, the number of reached states from V is less than or equal to the number of states in V, i.e., $|\delta(V,w)| \leq |V|$. Since $\delta(V,w)$=$\delta(V/v,w) \cup \delta(v,w)$, it holds $|\delta(V,w)| \leq |\delta(V/v,w)|$+$|\delta(v,w)|$. By the proof of first part, there must exist LoSs for all states v in V. Let w be the LoS for v, then $|\delta(v,w)|$=$|\phi|$=0. Consequently $|\delta(V,w)| \leq |\delta(V/v,w)|$+$|\delta(v,w)|$=$|\delta(V/v,w)| + 0 \leq |\delta(V/v,w)| \leq |V/v| < |V|$, i.e., $|\delta(V,w)| < |V|$ if w is the LoS for any state in V. Therefore each time we select one state from $\delta(V,w)$ and apply its LoS, the number of reached states from V will be decreased and reach to zero in at most $|V|$ steps. The sequence concatenated of the LoS of each step is the LoS for the transient SCC C_{tr}^i. Thus the second part has also been proved. ■

The proof of Theorem 1 also shows how to compute the LoSs for transient SCCs and its states therein. Now the formal algorithm for computing the LoSs is given.

Algorithm 3: **Computation of leading-out sequences for a transient SCC.**

Input: a transient SCC C_{tr}^i of a deterministic FSM and its node set V.

Output: a leading-out sequence for C_{tr}^i, denoted as $Los(C_{tr}^i)$.

1. $w := \varepsilon$;
2. While V is not empty:
   2.1. Select a node v from V;
   2.2. Find a LoS w' for v to leave out from this SCC;
   2.3. update $V = \delta(V,w') \cap V$;
   2.4. $w = ww'$;
3. Return $Los(C_{tr}^i) = w$.

Now we give an algorithm to show how to compute a SS into the set of states of the ergodic SCCs for a given FSM.

Algorithm 4: **Computation of leading-out sequences for a deterministic FSM.**

Input: a deterministic FSM G with η transient SCCs.

Output: a SS into the set of states of ergodic SCCs for G.

1. For $j = 1 : \eta$:
   1.1. $u_j = LoS(C_{tr}^j)$;
   1.2. $d_j = D(C_{tr}^j)$;
2. Let $\xi = maximum(d_j)$, $1 \leq j \leq \eta$;
3. For $i = 1 : \xi$:
   3.1. $w_i = \varepsilon$;
   3.2. for $j = 1 : \eta$:
      3.2.1. if $d_j = i$ then $w_i = w_i u_j$;
4. Return $w = w_\xi w_{\xi-1} \cdots w_1$.

Step 1 computes the LoS for each transient SCC (note that these computations are independent and can be parallelized). The number ξ obtained in Step 2 is the longest distance between transient SCCs and ergodic SCCs. Step 3 compose a sequence w_i by concatenating the LoSs of transient SCCs whose distance to ergodic components is i. After applying w_i on the given FSM, the states of transient components with same distance i will be driven into a component more closer to ergodic component. The last step is an iteration aiming at driving all states from far to near and at last into the ergodic components.

Example 7: The LoSs of the transient SCCs C_1, C_2, and C_3 in the WA in Fig.5 are, respectively, $w_1 = a$, $w_2 = bab$, and $w_3 = b$. Since there exist only one ergodic SCC C_4 and the distance from the transient SCCs to ergodic SCCs are $D(C_1) = 3$, $D(C_2) = 2$, and $D(C_3) = 1$. Thus the SS into the set of states of C_4 is $w = w_1 w_2 w_3$=$ababb$.

B. Computation of SSs on IMCG and WA

Before we formally introduce the computation algorithms, we give two conditions that characterize two interesting classes of FSMs.

Condition 1: The graph of the deterministic FSM has a unique ergodic SCC and there exists a SS into a single node in this ergodic SCC.

Condition 2: Given a subset of nodes V of a deterministic FSM, there exists a SS into V on the ergodic SCCs of this FSM.

Condition 1 can be tested by first checking whether there is only one node with no outgoing arc in the condensed graph of a given FSM. If so, using the method of auxiliary graph to verify if there exists a sequence $w \in I^*$ such that $\delta(v,w) = \delta(v',w)$ for any two different nodes v and v' in the unique ergodic SCC. Meanwhile, Condition 2 can be tested by verifying whether V is reached in the subset construction of the ergodic SCCs.

Theorem 2: Given a deterministic FSM: a) there exists a SS into a single node if and only if Condition 1 holds; b) there exists a SS into a subset of nodes V if and only if Condition 2 holds.

Proof: The proof has two parts: sufficiency and necessity. The necessity part can be obviously proved by the fact that for a given deterministic FSM and a subset $X_s \subseteq X$, if $\exists w \in I^*$ and $z \in I^*$ such that $|\delta(X,w)| = 1$ and $\delta(X,z) \subseteq X_s$, thus for any subset $X' \subseteq X$ it holds $|\delta(X',w)| = 1$ and $\delta(X',z) \subseteq X_s$.

The sufficiency part can be proved as follows. Since for a deterministic FSM, from all nodes of its graph there exists a leading-out sequence w_1 into its ergodic SCCs, if there exists a SS w_2 on the ergodic SCCs, then sequence $w_1 w_2$ is a SS for the given FSM. ■

Now we present the computation algorithms for the SSs into a single node and into a subset of nodes for a deterministic IMCG and WA.

Algorithm 5: **SSs into a single node.**

Input: a deterministic IMCG or WA of a given 1-place-unbounded SynPN.

Output: a SS into a single node.

1. Test Condition 1 by Algorithm 1. If it holds, let w_1 be the output of Algorithm 1 and go to step 2, else there is no SS into a single node on the FSM.
2. Find a sequence w_2 to drive the graph into the unique ergodic component by Algorithm 4.
3. Concatenate the previous two sequences $w = w_2 w_1$.

Example 8: Consider the WA in Fig. 5 of the net in Fig. 4. We can conclude that there exists no SS for the net in Fig. 4 because it has only one ergodic components on which there exists no SS. However, the IMCG in Fig. 2, there exists a SS $w = aaa$ leading the graph to node $[1\ 3\omega_2]^T$.

Complexity: Both step 1 and 2 can be solved in linear time, while step 3 can be completed in NLOGSPACE in the input size of the only ergodic SCC. For step 4, it only needs a constant time for concatenating two sequences. Thus the upper bound of the complexity of Algorithm 1 is NLOGSPACE in the size of the input IMCG or WA and it can be reached in the case of that the input IMCG or WA is strongly connected.

Now we end this section with an algorithm to compute a SS which drive a deterministic IMCG or WA into a set of nodes V_{P_s} corresponding the same marking of a set of bounded places P_s.

Algorithm 6: **SSs into a subset of nodes with the same marking of a subset of bounded places P_s.**
Input: a deterministic IMCG or WA of a given 1-place-unbounded SynPN and V_{P_s} with the same marking of a subset of bounded places P_s.
Output: a SS into a subset of nodes V_{P_s} with the same marking of a subset of bounded places P_s.

1. Test Condition 2 by Algorithm 2. If it holds, let w_1 be the output of Algorithm 2 and go to step 2, else there is no SS into a subset of nodes V_{P_s} with the same marking of a subset of bounded places P_s.
2. Find a sequence w_2 to drive the condensed graph into the ergodic component by Algorithm 4.
3. Concatenate the previous two sequences $w = w_2 w_1$.

Example 9: Consider the WA in Fig. 5 of the net in Fig. 4. If specify $P_s = \{p_1, p_2, p_3, p_4\}$ and $V_{P_s} = \{\ell_4, \ell_5\}$, $w = ababb$ is the SS into V_{P_s}. Because w can drive all the nodes in this WA into the unique ergodic SCC C_4 meanwhile all the states in C_4 are with the same marking of places in P_s.

Complexity: Similar with Algorithm 5, the step 1 and 2 in this algorithm need linear time and step 4 needs a constant time for concatenating two sequences but the complexity of step 3 is PSPACE-complete. Thus the upper bound of the complexity of Algorithm 6 is PSPACE-complete in the size of the input IMCG or WA and it can be reached in the case of that the input IMCG or WA is strongly connected.

V. Conclusion

In this paper, we consider one class of 1-place-unbounded synchronized Petri nets. In the case of that either its improved modified coverability graph or weighted automaton is deterministic, the synchronization problem for the net can be addressed by studying the deterministic finite state machine underlying the IMCG or the WA. We showed that the problems of synchronizing a deterministic finite state machines into one single state and into a specified subset of states can be reduced to synchronizing its ergodic strongly connected components. Thus the computation of these kinds of synchronizing sequences can be completed in two steps: first find out the sequences driving all states in transient components into ergodic components and then compute the synchronizing sequences on the ergodic components. In future work, we plan to explore the case in which the converted weighted automaton and the improved modified coverability graph of given 1-place-unbounded synchronized Petri nets are non-deterministic.

References

[1] S. Sandberg, "Chapter 1. homing and synchronizing sequences," *Model-based testing of reactive systems*, pp. 5–33, 2005.

[2] E. F. Moore, "Gedanken-experiments on sequential machines," *Automata studies*, vol. 34, pp. 129–153, 1956.

[3] D. Lee and M. Yannakakis, "Principles and methods of testing finite state machines-a survey," *Proceedings of the IEEE*, vol. 84, no. 8, pp. 1090–1123, 1996.

[4] M. Pocci, I. Demongodin, N. Giambiasi, and A. Giua, "Testing experiments on synchronized Petri nets," *IEEE Transactions on Automation Science and Engineering*, vol. 11, no. 1, pp. 125–138, 2014.

[5] ——, "Synchronizing sequences on a class of unbounded systems using synchronized Petri nets," *Discrete Event Dynamic Systems*, vol. 26, no. 1, pp. 85–108, 2016.

[6] L. Doyen, L. Juhl, K. G. Larsen, N. Markey, and M. Shirmohammadi, "Synchronizing words for weighted and timed automata," in *LIPIcs-Leibniz International Proceedings in Informatics*, vol. 29, 2014.

[7] C. Wu, I. Demongodin, and A. Giua, "Conversion of 1-place-unbounded synchronized petri nets into weighted automata," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 13 434–13 440, 2017.

[8] R. David and H. Alla, *Discrete, continuous, and hybrid Petri nets*. Springer Science & Business Media, 2010.

[9] M. Droste, W. Kuich, and H. Vogler, *Handbook of weighted automata*. Springer Science & Business Media, 2009.

[10] R. Tarjan, "Depth-first search and linear graph algorithms," *SIAM journal on computing*, vol. 1, no. 2, pp. 146–160, 1972.