Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 50-1 (2017) 13434–13440

Conversion of 1-Place-Unbounded Synchronized Petri Nets into Weighted Automata

Changshun WU * **Isabel DEMONGODIN** *
Alessandro GIUA *,**

* *Aix Marseille Université, CNRS, ENSAM, Université de Toulon, LSIS UMR 7296, France*
** *University of Cagliari, DIEE, Italy*
e-mail:{changshun.wu, isabel.demongodin, alessandro.giua}@lsis.org

Abstract: Testing is a fundamental technique for system design and verification to ensure security and reliability. However its application to discrete event systems modeled by unbounded synchronized Petri nets is not straightforward because there exists no finite exact representation for the infinite state space of these models. In this paper, we consider a special class of synchronized Petri nets, called 1-place-unbounded, that contain a single unbounded place. We show how a coverability graph that precisely describes the state space of such a model can be constructed extending to synchronized nets a technique previously presented for place/transition nets. In addition, this algorithm can also be used to verify whether a given synchronized Petri net contains exactly a single unbounded place. Next, we show that any net belonging to this special class can be converted into an equivalent weighted automaton. Based on this conversion, we observe that the testing of 1-place-unbounded synchronized Petri nets can be approached using the methods and results existing in the literature for the testing of weighted automata.

Keywords: Synchronized Petri nets; unbounded nets; coverability graph; weighted automata

1. INTRODUCTION

Testing is an indispensable part of system design and implementation. Since Moore (1956) introduced the framework of testing problems, a significant amount of work has been done in this area. A survey of general principles and methods related to testing of finite state machines and model based checking can be found in (Lee and Yannakakis, 1996) and (Sandberg, 2005), respectively. 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* (without output information) and *homing sequences* (with output information) are two conventional solutions to this problem and a comprehensive survey can be found in (Sandberg, 2005). Most of the literature on this topic uses as model finite state machines.

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 synchronizing sequences for Petri nets with input events, called *synchronized Petri nets* (SynPNs), was recently addressed. Pocci et al. (2014) focused on bounded SynPNs and provided general algorithm for computing synchronizing sequences 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 (Pocci et al., 2016) 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 synchronizing sequences. In particular using the *modified coverability graph* (MCG) in (Pocci et al., 2016) 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 with the long range goal of 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, Doyen et al. (2014) 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 WA to a known state with safety conditions or to a known location with or without safety condition is decidable. Motivated by this, we consider here a particular class of SynPNs, called *1-place-unbounded SynPNs*, in which a single place can be unbounded. We believe it is interesting to study this model because there exists a clear parallelism between such a net and a WA: the unbounded place can be regarded as the storage of system

10.1016/j.ifacol.2017.08.2300

energy in a WA, while the markings of bounded places can be regarded as the locations of a WA.

In this paper we present two original contributions. The first contribution is a new algorithm for constructing a finite graph called *improved modified coverability graph* (IMCG) for 1-place-unbounded SynPNs by extending the method presented in (Wang et al., 2010) for constructing an improved reachability graph for 1-place-unbounded place/transition nets. The IMCG allows one to exactly determine all reachable markings of the given net and thus provides an exact representation of its behavior. We point out that the IMCG can also be used to determine whether a SynPN is 1-place-unbounded. Our second contribution is an algorithm for converting a 1-place-unbounded SynPN into an equivalent WA. By this conversion, the synchronization problem for 1-place-unbounded SynPNs can be proved decidable thanks to the results in (Doyen et al., 2014). Due to space constraints, no formal proof of the correctness of the presented algorithms is given.

The paper is structured as follows. Section 2 presents the basic formalism on synchronized Petri nets and weighted automata. Section 3 focuses on developing the algorithms for constructing a finite graph to represent the exact behavior of a 1-place-unbounded SynPN. Section 4 shows how it is possible to convert a synchronized Petri net into an equivalent weighted automaton. Conclusion and future works are summarized in the last section.

2. SYNCHRONIZED PETRI NETS AND WEIGHTED AUTOMATA

In this section, we present the basic notions concerning SynPNs and WA. Most of them are taken from (Pocci et al., 2016) and (Doyen et al., 2014). For a comprehensive introduction to Petri nets and weighted automata, see (David and Alla, 2010; Droste et al., 2009).

2.1 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 $t \in T_e$.

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:

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

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 we introduce in Section 3.2 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 $e|\tau$:

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

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

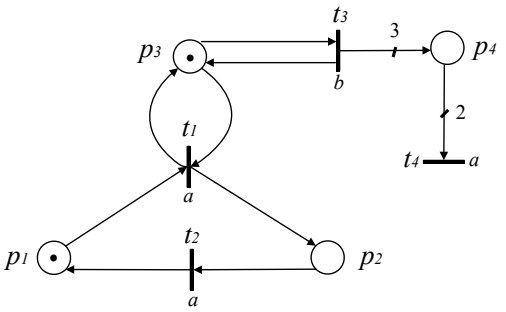

Fig. 1. A synchronized Petri net

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 $P = \{p_1, p_2, p_3, p_4\}$, $T = \{t_1, t_2, t_3, t_4\}$, $E = \{a, b\}$, $f(t_1) = f(t_2) = f(t_4) = a$, and $f(t_3) = b$. Let $M_0 = [1\ 0\ 1\ 0]^T$ be the initial marking. At M_0, the sets of enabled transitions are: $\varepsilon_a(M_0) = \{t_1\}$ and $\varepsilon_b(M_0) = \{t_3\}$. If b occurs at M_0, step $b|\{t_3\}$ fires leading to the new marking $M_1 = [1\ 0\ 1\ 3]^T$, i.e., $M_0[b|\{t_3\}\rangle M_1$. If a occurs at M_1, step $a|\{t_1, t_4\}$ fires leading to the new marking $M_2 = [0\ 1\ 1\ 1]^T$, i.e., $M_1[a|\{t_1, t_4\}\rangle M_2$.

In the following, w is a finite sequence of input evens $w = e_{i1}e_{i2}\cdots e_{ij}$ while σ is the sequence of the corresponding enabled transition sets $\sigma = \tau_1\tau_2\cdots\tau_j$.

Definition 3. (Increasing input sequence) Consider a marked synchronized PN $\langle N, E, f, M_0\rangle$. An input sequence $w \in E^*$ [1] is called *increasing* at marking $M_1 \in R(N, M_0)$ if:

$$\begin{cases} M_1[w|\sigma\rangle M_2[w|\sigma\rangle M_3[w|\sigma\rangle\cdots \\ M_i \lneq M_{i+1} \quad \forall i = 1, 2, \ldots \end{cases}$$

In other words, an increasing input sequence applied repetitively starting from M_1, always produces the same firing step sequence σ leading to a greater marking.

Proposition 1. (Pocci et al., 2016) Consider a synchronized PN $\langle N, E, f, M_0\rangle$ and a marking M. Let M' and M'' be respectively the marking reached after a first and a second application of input sequence w, i.e., $M[w|\sigma\rangle M'[w|\sigma'\rangle M''$. Sequence w is an increasing input sequence at M if the following three conditions hold:
C1) $\sigma' = \sigma$;
C2) $M \lneq M' \lneq M''$;
C3) $(\forall p$ such that $M'(p) > M(p))$ $(\forall t \in p^\bullet)$ $M'(p) \geq Pre(p, t)$.

The above provides a sufficient condition for a sequence to be increasing, because: i) Conditions C1 and C2 ensure that the input sequence w is increasing; ii) Condition C3 guarantees that the greater marking reached after repeating w, will not enlarge the firing step, i.e., the firing step will remain the same as before.

2.2 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.

[1] Here E^* is the set of all finite strings of elements of E.

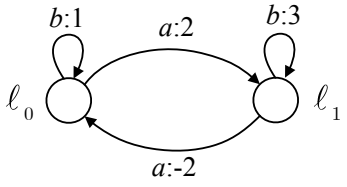

Fig. 2. A weighted automaton.

According to (Droste et al., 2009) and (Doyen et al., 2014), 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 $\delta^*(\theta, aw) = \delta(\delta(\theta, a), w)$.

Example 2. Consider the weighted automaton in Fig.2, where $L = \{\ell_0, \ell_1\}$, $\Sigma = \{a, b\}$, $\mathcal{I} = \{[0, +\infty), [4, +\infty)\}$, $Safe(\ell_0) = [0, +\infty)$ and $Safe(\ell_1) = [4, +\infty)$, ℓ_0 is the initial location and $\rho_0 = 0$ is the initial energy level. Thus, the initial state is $\theta_0 = (\ell_0, 0)$. At θ_0, the occurrence of event a is not feasible because it would lead to a state $(\ell_1, 2)$ that does not respect the safety condition of ℓ_1, as $2 \notin Safe(\ell_1)$. If the initial state is $\theta_0' = (\ell_0, 2)$, the occurrence of event a would lead the system to new state $\theta_1' = (\ell_1, 4)$.

2.3 Synchronization on unbounded SynPNs and WA

Consider a system with unknown current state. We may have partial information concerning the current state: sometimes it is known to belong to a given set of states, sometimes it may belong to the entire state space. A synchronization problem consists in finding an input sequence that drives the system to a known state (and no observation is required to solve this problem).

In the case of unbounded SynPNs, the synchronization problem defined in (Pocci et al., 2016) consists in finding an input sequence that drives the net system to a known marking of the bounded places due to the impossibility in the general case of identifying the exact marking of unbounded places. Similarly, the two main synchronization problems for WA with safety condition are synchronization (find an input sequence to drive the system into a known state) and location-synchronization (find an input sequence to drive the system to a known location under safety conditions or not).

3. AN IMPROVED COVERABILITY GRAPH FOR 1-PLACE-UNBOUNDED SynPNs

As we know, it is easy to check whether a given place/transition net is 1-place-unbounded because we just construct the coverability graph by Karp and Miller (1969) and verify that a single place has an ω component. On the contrary, in the case of SynPNs there exist so far no algorithm to check that: the modified coverability graph proposed by Pocci et al. (2016) may contain vanishing markings (i.e., markings that are reachable in the net but are not represented in the graph) and thus may fail to recognize a place as unbounded.

In this section, we introduce a new method to construct a finite graph that will be used to accurately describe the behavior of 1-place-unbounded SynPNs, a subclass of SynPNs which have one and only one unbounded place. Such a graph is called *improved modified coverability graph* (IMCG) because it derives from the construction method of Karp and Miller (1969), Wang et al. (2010) and Pocci et al. (2016). First we recall the notion of ω-numbers from (Wang, 1991), then we extend the method of Wang et al. (2010) to SynPNs.

3.1 ω-number

Definition 4. (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.

An ω-number is a linear set of numbers, thus it contains more information and is more expressive than the simple symbol ω. Let $\mathbb{Z}_\omega$ ($\mathbb{N}_\omega$) denote the union of the set of all integers (nonnegative integers) and the set of ω-numbers. Now we introduce the addition and comparison rules on $\mathbb{Z}_\omega$.

The addition rule is extended to the sum of an integer number and an ω-number as follows: given an integer a and an ω-number $k\omega_n + q$, their sum is $a + (k\omega_n + q) = k\omega_{n+s} + r$, where $q + a = ks + r$, $s \in \mathbb{Z}, 0 \leq r < k$. In simple words, every element of the ω-number is increased by the integer as a result. For example, $3 + (3\omega_0 + 1) = \{(3 + (3 \cdot i + 1) | i \geq 0\} = \{(3 \cdot i + 4) | i \geq 0\} = \{(3 \cdot i + 1) | i \geq 1\} = \{4, 7, 10, \cdots\} = 3\omega_1 + 1$. Note that addition between ω-number is not allowed.

The usual comparison rule between two integers can also be extended to following case: given two ω-numbers $x = k\omega_n + q$, $y = k\omega_m + q$ with the same base and remainder values, $x \leq y$ if $n \leq m$; $x = y$ if $n = m$. Note that for each two comparable ω-numbers x and y, it can be easily concluded that $x \leq y$ iff $x \supseteq y$.

Definition 5. (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.

Naturally, an ω-vector is a set of vectors on $\mathbb{Z}^m$. One can get an instance of ω-vector after assign an instance to its components of ω-number. A special case is that the minimal instance of an ω-vector is such that each ω-number is assigned to its minimal instance. For instance, $\mathrm{V} = [x_1\ x_2\ x_3]^T = [1\ 2\omega_2\ 3] = \{[1\ (2 \cdot i)\ 3]^T | i \geq 2\}$.

Let $\mathrm{V}_1 = [x_1\ x_2\ \cdots\ x_m]^T$, $\mathrm{V}_2 = [y_1\ y_2\ \cdots\ y_m]^T$ be two ω-vectors and they have the kth component as their ω-numbers. If $x_i \leq y_i$ for all $1 \leq i \leq m$, then we say $\mathrm{V}_1 \leq \mathrm{V}_2$. If $\mathrm{V}_1 \leq \mathrm{V}_2$ and $x_i = y_i$ for all $i : i \neq k$, then it has $\mathrm{V}_1 \supseteq \mathrm{V}_2$.

Then addition rule can be also extended for ordinary vector and ω-vector as follows: i) addition between two ordinary vectors is defined in a traditional manner; ii) the addition between an ordinary vector and an ω-vector is performed by adding the numbers of the same coordinate in the two vectors; iii) finally, the addition between two ω-numbers is not allowed. As an example, given an ω-vector $\mathrm{V} = [1\ 2\omega_2\ 3]^T = \{[1\ (2 \cdot i)\ 3]^T | i \geq 2\}$, $\mathrm{V} + [0\ 1\ 1]^T = \{[1\ (2 \cdot i + 1)\ (3 + 1)]^T | i \geq 2\} = \{[1\ (2 \cdot i + 1)\ 4]^T | i \geq 2\} = [1\ (2\omega_2 + 1)\ 4]^T$.

3.2 Improved Modified Coverability Graph

Before presenting the algorithm for constructing the IMCG of 1-place-unbounded synchronized PNs, several relevant definitions must be introduced.

In an IMCG, there exist two types of markings. The first type is an non-negative integer vector denoted by M^I; the second type is an ω-vector denoted by M^ω, an infinite set of integer markings. Note that in the case of 1-place-unbounded SynPNs, an ω-vector only contains a single ω-number.

Note that an ω-vector is a set of integer markings, because one or more of its components is an ω-number which is a linear set containing an infinite number of integer numbers, i.e., the numbers of tokens in the corresponding unbounded place. Given an ω-vector M^ω and a transition t, it may be possible that some integer markings in M^ω can not enable t while some others can. Therefore, different enabling conditions for a transition at an ω-vector should be distinguished.

Consider a SynPN and one of its ω-vector M^ω. A transition t is:

- **i)** not enabled if there exists no marking in M^ω that enables t;
- **ii)** totally enabled if all markings in M^ω enable t;
- **iii)** partially enabled if there exists at least one marking in M^ω that enables t and another one that does not enable t.

When t is partially enabled at M^ω, M^ω can be partitioned into two sets: $M^\omega_{dis,t}$ a finite set of integer markings those do not enable t and $M^\omega_{en,t}$ an ω-vector, i.e, an infinite set of markings, which enable t. Due to the fact that there is only one unbounded place in the net and to Assumption 1, there exists at most one transition partially enabled at an ω-vector.

Definition 6. (Generation of next markings after the occurrence of an event) Consider a marked 1-place-unbounded synchronized PN $\langle N, E, f, M_0 \rangle$, the occurrence of input event e at current marking M yields a set of markings Q in the following two cases:

1) if $\forall t \in \varepsilon_e(M)$ is totally enabled, $Q=\{M+\sum_{t\in\varepsilon_e(M)} C(\cdot,t)\}$,
2) if $M = M^\omega$ and $\exists t^* \in \varepsilon_e(M)$ is partially enabled, $Q=\{M^I + \sum_{t\in\varepsilon_e(M^I)} C(\cdot,t) \mid M^I \in M^\omega_{dis.,t^*}\} \cup \{M^\omega_{en.,t*} + \sum_{t\in\varepsilon_e(M^\omega_{en.,t^*})} C(\cdot,t)\}$.

Let q be a node of the tree. We use M_q and $M_q(p)$ to denote its marking and the marking of place p, respectively. Before introducing the construction algorithm, the following procedure, untitled *duplication*, allows a node to be labeled as *"new"* or *"duplicate"*.

Procedure 1: **Duplication check**.
Input: a node q.
Output: q is duplicate or q is new.
If there exists already a node $\tilde{q}$ in the tree with $M_q{=}M_{\tilde{q}}$ or $M_q \subsetneq M_{\tilde{q}}$, **then** tag it "duplicate", **Else** tag it "new".

In following procedure, notations "/" and "%" are, respectively, used for quotient and remainder operators and "\" is used to denote set difference.

Algorithm 1. **IMC Tree construction for a 1-place-unbounded deterministic SynPN.**
Input: *a deterministic marked SynPN* $\langle N, E, f, M_0\rangle$.
Output: *a IMC Tree* $\mathcal{T}$ *(if the SynPN is 1-place unbounded) or "This net has more than one unbounded place".*

1. *Let* $P_u = \emptyset$, $Q = \emptyset$ *and label the root node* q_0 *with the initial marking* M_0 *and tag it "new".*
2. ***While*** *a node tagged "new" exists,* ***do***
   - *2.1. Select a node* q *tagged "new".*
   - *2.2.* ***For*** *all* $e \in E$ *such that* $\varepsilon_e(M_q) \neq \emptyset$*:*
     - *2.2.1. Generate the set of next markings* Q*, according to def.6.*
     - *2.2.2.* ***For*** *each node* q' *in* Q*,* ***do***
       - *2.2.2.1.* ***If*** M_q *is an* ω*-marking and* $M_{q'}$ *is an integer marking,* ***then*** *add "*$e|(\varepsilon_e(M_q)\backslash t)$*" from* q *to* q'*, where* t *is partially enabled at* M_q*,* ***else*** *add an arc labeled* $e|\varepsilon_e(M_q)$ *from* q *to* q'*.*
       - *2.2.2.2. Apply* Duplication check for q'*.*
       - *2.2.2.3.* ***If*** q' *is "new",* ***then***
         (i) ***If*** *there exists a node* $\hat{q}$ *on the path from* q_0 *to* q' *such that* $\hat{q}[w|\sigma\rangle q'$ *and* w *is an increasing input sequence satisfying* proposition *1,* ***then*** *let* $P_{u,new}$ *be the set of the places such that* $M_{\hat{q}}(p) < M_{q'}(p)$*.*
         (ii) $P_u = P_u \cup P_{u,new}$*.*
         (iii) ***If*** $|P_u| \leq 1$*,* ***then for*** $p \in P_u$*,* $M_{q'}(p) = k\omega_n + r$*, where:* $k = M_{q'}(p) - M_{\hat{q}}(p)$*,* $n = M_{q'}(p)/k$*,* $r = M_{q'}(p)\%k$*.* ***Else:*** *return* "This net has more than one unbounded place"*, go to step 3.*
   - *2.3. Untag* q *and let* $Q = \emptyset$*.*
3. *End.*

In the algorithm, only the node labeled "new" should be analyzed. Step *1* initialize the root node of the tree. Step 2 first selects a node q to analyze, then generates its child nodes and determines which ones should be analyzed tagging them with labels. If to a child node is assigned label "duplicate", it means that it is not needed to be analyzed. The child nodes labeled "new" are checked to identify if a place p becomes unbounded (see step *2.2.2.3.)*.

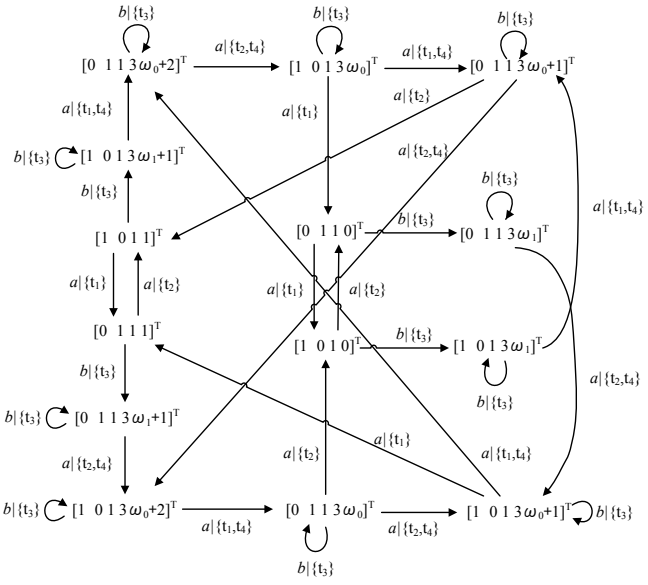

Fig. 3. The IMCG of the synchronized PN of Fig.1.

If there is a child node violating the criterion that the number of unbounded places is at most one, the algorithm terminates, else it updates the making of these child nodes as ω-vectors and then finish analyzing node q. Step *2* is repeated until no node in the tree needs to be analyzed.

The IMC graph (IMCG) is obtained from the IMC tree by merging duplicate nodes with the corresponding untagged node. We let $IR(N, M_0)$ denote the set of all markings associated to nodes of the IMCG, either single integer markings labeling a node or integer markings represented by an ω-vector labeling a node.

The following results are presented without proof due to lack of space.

Theorem 1. The improved modified coverability graph constructed by algorithm 1 is finite.

Theorem 2. Given a synchronized Petri net $\langle N, E, f, M_0\rangle$, let $R(N, M_0)$ be the set of its reachable markings and $IR(N, M_0)$ be the set of all the markings contained in its improved modified coverability graph constructed by Algorithm 1. We have $R(N, M_0) = IR(N, M_0)$.

In other words, an improved modified coverability graph constructed by Algorithm 1 contains exactly all its reachable markings.

Example 3. The improved modified coverability graph of the SynPN in Fig.1 with $M_0 = [1\ 0\ 1\ 1]^T$, is shown in Fig.3. Following the construction procedure is illustrated by two cases:

1) Create a node q_0 as the root node with marking $M_{q_0} = [1\ 0\ 1\ 1]^T$ and tag it "new". Consider event b at node q_0. Since $\varepsilon_b(M_{q_0})$ is not empty set, we generate the set Q of next nodes and compute their markings after the occurrence of b at q_0 by definition 6. Given that M_{q_0} is an integer marking, $Q = \{M_{q_0} + \sum_{t\in\varepsilon_b(M_{q_0})} C(\cdot,t)\}$= $\{[1\ 0\ 1\ 1]^T + [0\ 0\ 0\ 3]^T\}$=$\{[1\ 0\ 1\ 4]^T\}$. For the only node q' in

Q, add the arc $b|\{t_3\}$ from q_0 to q' and then apply Procedure 1: q' is labeled "new". Now we do the ω-number check in step 2.2.2.3., we find that b is an increasing sequence, $k = M_{q'}(p_4)\text{-}M_{q_0}(p_4) = 3$, $n = M_{q'}/3 = 1$, and $r = M_{q'}\%3 = 1$. Now we update $M_{q'} = [1\ 0\ 1\ 3\omega_1 + 1]^T$.

2) Select the node q with its marking M_q=$[1\ 0\ 1\ 3\omega_0]^T$. For event a, it has $\varepsilon_a(M_q) = \{t_1, t_4\}$ such that t_1 is totally enabled and t_4 is partially enabled. Then ω-vector M_q is split into two sets: $M_{q,dis,t_4} = \{[1\ 0\ 1\ 0]^T\}$ and $M_{q,en,t_4} = [1\ 0\ 1\ 3\omega_1]^T$. Therefore the occurrence of a at q generates two possible behaviors: step $a|\{t_1\}$ leads to $[0\ 1\ 1\ 0]^T$ from $[1\ 0\ 1\ 0]^T$ and step $a|\{t_1, t_4\}$ leads to $[0\ 1\ 1\ 3\omega_0 + 1]^T$ from $[1\ 0\ 1\ 3\omega_1]^T$.

4. CONVERSION OF SynPNs TO WA

In this section, we propose an algorithm to convert a SynPN into an equivalent WA. Based on this conversion, the synchronization problem for deterministic 1-place-unbounded SynPNs can be addressed using the existing results related to synchronization problems for WA. First several notions and functions relevant to this conversion algorithm are presented.

4.1 Intervals

For an interval, two functions, denoted min and max, are defined respectively to extract its lower and upper bounds. For example, consider interval $I = [2, 5]$, $min(I) = 2$ and $max(I) = 5$. Meanwhile for a group of distinct intervals with indexes, function $In : \mathbb{N} \times \mathcal{I} \to \mathbb{N}$ is defined for a non-negative integer to return the index of its belonging interval. As a simple example, if we consider $\mathcal{I} = \{I_0 = [0, 3],\ I_1 = [4, 6],\ I_2 = [7, 12],\ I_3 = [13, +\infty)\}$, then it holds that $In(5, \mathcal{I}) = 1$ and $In(18, \mathcal{I}) = 3$.

Given an ordered set $W = \{w_1, \cdots, w_k\}$ of positive integers, with $w_i < w_j$ for $1 \leq i < j \leq k$, *function* Ξ defines a set of intervals $\Xi(W) = \{I_0, \cdots, I_k\}$, where $I_0 = [0, w_1 - 1]$, $I_1 = [w_1, w_2 - 1], \cdots, I_{k-1} = [w_{k-1}, w_k - 1]$ and $I_k = [w_k, +\infty)$. For instance, if $W = \{2\}$, function $\Xi(W)$ defines two intervals: $I_0 = [0, 1]$ and $I_1 = [2,\ +\infty)$.

4.2 Conversion of SynPNs to WA

In this subsection, we introduce an algorithm for converting a deterministic SynPN with a single unbounded place into a weighted automaton. We denote by p_u and $M(p_u)$, respectively, the single unbounded place and its marking which will be modeled as the energy of the WA because both of them can be changed after the occurrence of an input event, in particular they are both nonnegative integer numbers in our paper.

The weights on the arcs outgoing from place p_u determine a partition of the nonnegative set into different intervals $\Xi(W)$ with ascending order $W = \{Pre(p_u, t) | t \in p_u^\bullet\}$. These intervals will be used as the safety conditions of converted WA. Note that the cardinality of the weight set W is less than or equal to the cardinality of set $p_u^\bullet$ because there may exist two or more output arcs of place p_u, which have the same weight.

A location ℓ in a WA converted from a given SynPN is labeled as (M_ℓ, x), where $M_\ell = M \uparrow_b$ is the projection on the bounded places of a marking of the Petri net and x is the index of the energy interval to which the current value of the energy (i.e., $M(p_u)$) belongs, i.e., $M(p_u) \in I_x$ (an interval in $\Xi(W)$). Note that a location in the converted WA is determined not only by the marking of bounded places but also by an energy interval. That means given two nodes (M_ℓ, x) and (M_ℓ, y), if $x \neq y$, they are two different locations in the converted WA.

Moreover a location of the WA corresponds to a set of integer markings which have the same set of enabled transitions and the same marking of bounded places. We use $\varepsilon(\ell)$ to denote the set of enabled transitions at location ℓ and $\varepsilon_e(\ell)$ to note the transitions in $\varepsilon(\ell)$ but associated with event e. Therefore the occurrence of e changes the markings in ℓ by the same vector $\sum_{t \in \varepsilon_e(\ell)} C(\cdot, t)$. It is obvious that the new markings of bounded places remain the same. However the set of enabled transitions may be different, thus we may need to create several locations and to add the same arc "$e|\sum_{t \in \varepsilon_e(\ell)} C(p_u, t)$" to them.

Now we give the complete algorithm for converting a 1-place-unbounded SynPN into an equivalent WA. The notion of equivalence will be discussed at the end of this section.

Algorithm 2. **Conversion of a SynPN into a WA.**
Input: *a deterministic SynPN $\langle N, E, f, M_0\rangle$ with a single unbounded place p_u.*
Output: *a WA $\mathcal{A} = \langle L, \Sigma, \Delta, \mathcal{I}, \text{Safe}, \ell_0, \rho_0\rangle$.*

1. *Let $\Sigma = E$, $\rho_0 = M_0(p_u)$, $L = \emptyset$, and $\mathcal{I} = \emptyset$.*
2. *Create $\mathcal{I} = \Xi(W)$ with $W = \{Pre(p_u, t) | t \in p_u^\bullet\}$.*
3. *Create an initial node ℓ_0 labeled $(M_0 \uparrow_b, v)$ and tag it "new" where $v = In(M_0(p_u), \mathcal{I})$. Let $L = \{\ell_0\}$ and define $Safe(\ell_0) = I_v$.*
4. ***While** a location tagged "new" exists **do**:*
   - *4.1. Select a location $\ell = (M_\ell, x)$ tagged "new".*
   - *4.2. **For** all $e \in E$:*
     - *4.2.1. Let $M'(p) = M_\ell(p) + \sum_{t \in \varepsilon_e(\ell)} C(p, t)$ for all $p \in P_b$ be the new marking of bounded places by firing all enabled $t \in T_e$ and $\Delta M(p_u) = \sum_{t \in \varepsilon_e(\ell)} C(p_u, t)$ be the energy difference.*
     - *4.2.2. Determine $i = In(min(I_x) + \Delta M(p_u), \mathcal{I})$ and $j = In(max(I_x) + \Delta M(p_u), \mathcal{I})$.*
     - *4.2.3. **For** y from i to j:*
       - *4.2.3.1. Add a new location ℓ' and label it (M', y).*
       - *4.2.3.2. Add an arc labeled $e : \Delta M(p_u)$ from ℓ to ℓ'.*
       - *4.2.3.3. **If** there exists already a location in L with label (M', y), **then** tag ℓ' "duplicate", **else** tag it "new" and let $L = L \cup \{\ell'\}$ and define $Safe(\ell') = I_y$.*
   - *4.3. Untag node ℓ.*
5. *End.*

Theorem 3. The weighted automaton constructed by algorithm 2 is finite.

Proof. Due to the finiteness of the markings of the bounded places and of the energy intervals, their cartesian product is finite. □

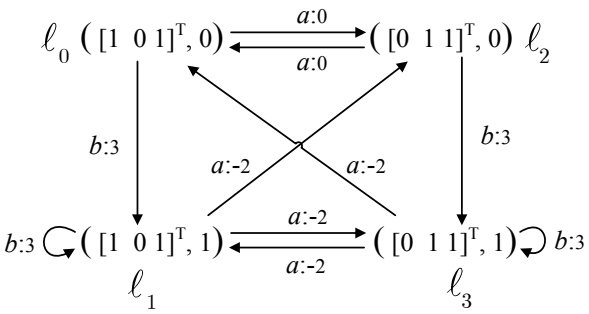

Fig. 4. Weighted automaton from SynPN in Fig. 1.

Example 4. From previous section, we know that the SynPN of Fig.1 is one place unbounded and p_4 is the single unbounded place. After applying the previous algorithm to this SynPN with initial marking $M_0 = [1\ 0\ 1\ 0]^T$, we obtain its corresponding weighted automaton given in Fig.4.

In the following example, we select some nodes to analyze and show how this algorithm works.

- $\Sigma = E = \{a, b\}$, $\rho_0 = M_0(p_4) = 0$, $L = \emptyset$, and $\mathcal{I} = \emptyset$.
- Create energy intervals $\mathcal{I} = \Xi(W) = \{I_0 = [0, 1], I_1 = [2, +\infty)\}$ with $W = \{2\}$.
- Create the initial node $\ell_0 = ([1\ 0\ 1]^T, 0)$ and tag it "new", then define $Safe(\ell_0) = I_0$ and update $L = \{\ell_0\}$.
- Consider node ℓ_0 labeled "new", the occurrence of b at ℓ_0 increases the energy by 3 and reaches a new marking $M' = [1\ 0\ 1]^T$. Thus the new possible bounds of the energy are $minimum = 0 + 3 = 3$ and $maximum = 1 + 3 = 4$. Since both bounds are in I_1, we only create one node $\ell_1 = ([1\ 0\ 1]^T, 1)$ tagging it "new" and update $L = \{\ell_0, \ell_1\}$. Similarly we can obtain node $\ell_2 = ([0\ 1\ 1]^T, 0)$ reached from ℓ_0 by the occurrence of a.
- Now we consider node ℓ_1. The occurrence of a generates the marking $[0\ 1\ 1]^T$ and brings possible bounds $minimum = 2 - 2 = 0 \in I_0$ and $maximum = +\infty - 2 = +\infty \in I_1$. Therefore we link ℓ_1 with two nodes: $\ell_2 = ([0\ 1\ 1]^T, 0)$ and $\ell_3 = ([0\ 1\ 1]^T, 1)$ with arcs labeled by $a : -2$.
- Finally we obtain the WA with four locations as that depicted in Fig.4.

The discussion of the equivalence of a SynPN and its corresponding WA will end this section.

Given a 1-place-unbounded synchronized Petri net, $\mathcal{PN} = \langle N, E, f, M_0\rangle$ with p_u its unbounded place, let $\mathcal{A}$ be the converted weighted automaton of $\mathcal{PN}$, constructed by Algorithm 2. Let $R(N, M_0)$ be the set of the reachable markings of $\mathcal{PN}$ and Θ be the state space of $\mathcal{A}$. We define a one to one *function* $\gamma : R(N, M_0) \to \Theta$ as: $\gamma(M) = \theta = (\ell, \rho)$ where $\rho = M(p_u)$ and $\ell = (M \uparrow_b, x)$ with $x = In(M(p_u), \mathcal{I})$. The inverse of this function is $\gamma^{-1} : \Theta \to R(N, M_0)$, $\gamma^{-1}(\theta) = \gamma^{-1}(\ell, \rho) = M$ with $M \uparrow_b = M_\ell$ for all $p \in P_b$ and $M(p_u) = \rho$.

For instance, given the initial making M_0, we have $\gamma(M_0) = \theta_0 = (\ell_0, \rho_0)$ and $\gamma^{-1}(\theta_0) = M_0$.

We can show that γ defines an equivalence between the SynPN and the converted WA. In fact, consider $M[e|\tau\rangle M'$ and $\theta' = \delta(\theta, e)$ with $\gamma(M) = \theta$: we have $\theta' = \gamma(M')$. This can be checked by step *4* in Algorithm 2. Inductively, it can be shown that if $M_0[w|\sigma\rangle M$, then we have a state $\theta = \delta(\theta_0, w)$ in the converted WA and $\gamma(M) = \theta$. In this sense Algorithm 2 converts a 1-place-unbounded SynPN into an equivalent WA.

5. CONCLUSION

In this paper we study the class of 1-place-unbounded synchronized Petri nets and present two original contributions. The first contribution is a tool for the analysis of this class of nets, called improved modified coverability graph. It can exactly represent the state space of the net and can also be used to test if the net belongs to the class. The second contribution consists in showing how a synchronized Petri net can be converted into an equivalent weighted automaton with safety conditions. Consequently one can address the synchronization problems for this class of synchronized Petri nets by referring to the methods and results already existing for weighted automata. In future, we will try to extend these results to derive efficient computation algorithm for synchronizing and homing sequences based on improved modified coverability graph and equivalently converted weighted automata.

REFERENCES

David, R. and Alla, H. (2010). *Discrete, continuous, and hybrid Petri nets*. Springer Science & Business Media.

Doyen, L., Juhl, L., Larsen, K.G., Markey, N., and Shirmohammadi, M. (2014). Synchronizing words for weighted and timed automata. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 29.

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

Karp, R.M. and Miller, R.E. (1969). Parallel program schemata. *Journal of Computer and System Sciences*, 3(2), 147 – 195.

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

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

Pocci, M., Demongodin, I., Giambiasi, N., and Giua, A. (2014). Testing experiments on synchronized Petri nets. *IEEE Transactions on Automation Science and Engineering*, 11(1), 125–138.

Pocci, M., Demongodin, I., Giambiasi, N., and Giua, A. (2016). Synchronizing sequences on a class of unbounded systems using synchronized Petri nets. *Discrete Event Dynamic Systems*, 26(1), 85–108.

Sandberg, S. (2005). Homing and synchronizing sequences. In *Model-based testing of reactive systems*, 5–33. Springer.

Wang, F.Y. (1991). A modified reachability tree for Petri nets. In *Systems, Man, and Cybernetics, Decision Aiding for Complex Systems, Conference Proceedings.*, 329–334. IEEE.

Wang, Y., Jiang, B., and Jiao, L. (2010). Property checking for 1-place-unbounded Petri nets. In *Theoretical Aspects of Software Engineering (TASE), 2010 4th IEEE International Symposium on*, 117–125.