Observer Design for Bounded Output Synchronized Petri Nets

Rabah Ammour, Said Amari, Leonardo Brenner, Isabel Demongodin, Dimitri Lefebvre

Observer Design for Bounded Output Synchronized Petri Nets*

Rabah Ammour[1], Said Amari[2], Leonardo Brenner[1], Isabel Demongodin[1] and, Dimitri Lefebvre[3]

***Abstract*— This paper concerns discrete event systems modeled with a particular class of synchronized Petri nets that include output functions, called Output Synchronized Petri nets. Such a formalism is suitable and tractable to represent a large variety of Cyber-Physical Systems. As a preliminary result on Output Synchronized Petri nets, we propose a method to design observers for such systems based on the input control and output signal that circulate between the controller and the plant.**

I. INTRODUCTION

Cyber-Physical System (CPSs) are intelligent systems that integrate control, communication, and computing. CPSs have diversified applications include smart grid, autonomous automotive systems, medical monitoring, process control systems, distributed robotics, and automatic pilot avionics. This paper is about Cyber-Physical Systems modeled as discrete event systems. In particular we are interested in the modeling and analysis of both the control inputs generated by the controller and the observed outputs delivered by the plant to improve the detectability of the CPS.

Our objectives are first to formalize the information that circulates in the CPS from the perspective of observation, and second to design observers for CPS based on control inputs and observed outputs. Such observers are helpful for a large variety of applications including fault tolerant control and cyber-security. As a consequence, the domains of application range from computer science and networks to Industry 4.0 including robotics and transportation aspects. In order to formalize how the information circulates in CPS, the first contribution of the paper is to introduce synchronized Petri nets combined with output functions (OutSynPN) that are suitable to describe many situations where the control inputs and delivered outputs are considered together. This new formalism enlarges the class of synchronized PNs [1], but also the classes of labeled PNs [2] and of Interpreted PN [3]. In fact, it allows the system to be controlled and observed thanks to the input events associated with controllable transitions and to the output events related to marking changes. Thus, it is possible to represent, in a single formalism, both the plant and the controller of a CPS. The second contribution is to propose a method to design observers based on input and / or output for CPS modeled with OutSynPN. Compared to the existing results, most of them developed with automata, our approach leads to a model that is more compact (due to the tokens semantics) and flexible (due to use of weight and synchronization).

The paper is structured as follows. Section II is about the state of the art on observers in the discrete event system theory. In Section III, basic definitions and terminology on synchronized Petri nets are reviewed. Section IV is dedicated to the definition of the Output Synchronized Petri nets (OutSynPN). Section V deals with the design of input and/or output observers for OutSynPN. Finally, Section VI concludes the paper.

II. STATE OF THE ART

Inference and state estimation problems have been intensively studied in the context of discrete event systems [4], [5]. In particular, standard approaches have been developed to design observers for systems modeled as labeled finite state automata. Observability study and state estimation for finite automata are relevant topics and have been investigated in the literature under different formulations. Based on the matrix representations for finite automata, an observer design procedure is defined in [6]. Authors are particularly interested in studying the observability problem of partially observed non-deterministic automata. The approaches in [7] are focused on the location state recovery problems of non-deterministic finite automata with several observability concepts using the supervisory control principle. The authors of [8] have studied the classical and logic based observer design of finite automata for which several techniques have been developed, such as the determination of location-observers based on the computation of state-observer trees. An observer design of input/output asynchronous sequential machines for the model-matching problem has been developed in [9].

For Petri nets, the problem of marking estimation and observation is well addressed in the literature, and several methods have been proposed for its solution. In [10] the observability problem has been addressed under the hypotheses that the initial marking of the Petri net is completely unknown and all transitions are observable. Authors define several observability properties related to the existence of complete words and to prove some of them, they have provided an observer coverability graph. In the same context, the authors of [11] have solved a slightly different problem considering that the initial marking is assumed to be known and only a subset of transitions is assumed to be observable. We can also mention the existing studies using linear Max-Plus models and timed event graphs, which are a particular

*This work has been partially supported by the CPSecurity project (CNRS-INS2I grant).

[1]Rabah Ammour, Leonardo Brenner, and Isabel Demongodin are with Aix-Marseille University, University of Toulon, CNRS, LIS, Marseille, France rabah.ammour, leonardo.brenner, isabel.demongodin@lis-lab.fr

[2]Said Amari is with LURPA, ENS Paris-Saclay, France. samari@ens-paris-saclay.fr

[3]Dimitri Lefebvre is with Normandy University, GREAH, Le Havre, France. dimitri.lefebvre@univ-lehavre.fr

class of Petri nets. The authors of [12] have proposed a methodology to develop an observer for max-plus linear systems and design an observer-based controller for timed event graphs [13]. The principle of these observers aims at estimating the state for a given plant by using input and output measurements. They are obtained by an analogy with the classical Luenberger observer for continuous linear systems. The authors have considered that the system matrices are assumed to be known, and the observation of the input and of the output is used to compute the estimated state. A similar method with the previous one has been developed in [14] for designing asymptotic observers with the class of interpreted Petri nets.

Moreover, a state estimation approach for labeled Petri nets is presented in [15] and upper bounds on the number of system states that are consistent with an observed sequence of labels are given. This study is applicable to the class of Petri nets that may have transitions that share the same label and/or unobservable transitions. In [16] an approach to estimate and recognize the set of consistent markings in labeled Petri nets is introduced, which is focused on the representative marking analysis. The question of state estimation in labeled Petri nets with silent transitions and indistinguishable transitions (i.e.,transitions sharing the same label with other transitions) under partial observation and uncertainty in the initial marking, is resolved in [2]. They prove that all sets of markings consistent with a given sequence of observations can be described in linear algebraic terms and this observation is used to construct a marking observer. In some classes of labeled Petri nets and to handle the unobservable cycles, the authors of [17] and [18] have developed methods based on the notion of reduced consistent markings, which can be used for marking avoidance and probabilistic marking estimation. Compared with labeled Petri net based observers, our work considers both controllable inputs on transitions and delivered labels associated with marking changes.

III. Preliminaries

This section presents some basis on Petri nets and on Synchronized Petri nets (see [3], [1], [19] for more details on these formalisms). In the rest of this paper, the reader will only deal with a class of bounded synchronized PNs, associated with a single server semantics and, that also satisfy some structural restriction to ensure the determinism of the model.

A. Petri nets

A *Petri net* (PN) is a structure $N = \langle P, T, Pre, Post\rangle$, 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 matrices 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 vector $M : P \to \mathbb{N}^m$ that assigns to each place a non-negative integer. We denote by $M(p)$ the marking of place p. A *marked* Petri net $\langle N, M_0\rangle$ is a net N with an initial marking M_0.

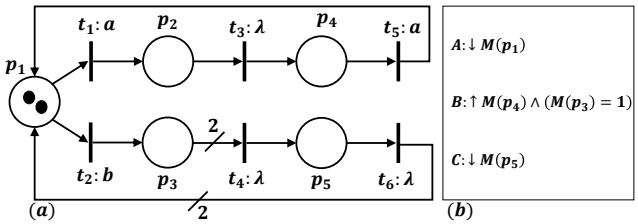

Fig. 1: (a) A synchronized Petri net; (b) Output labels

The *preset* and *postset* of place p are respectively: ${}^\bullet p = \{t \in T \mid Post(p,t) > 0\}$ and $p^\bullet = \{t \in T \mid Pre(p,t) > 0\}$.

A transition t is enabled at M iff $M \geq Pre(\cdot, t)$, denoted as $M[t\rangle$ in this paper. The set of enabled transitions at M is denoted by $\zeta(M) = \{t \in T \mid M \geq Pre(\cdot, t)\}$. The firing of an enabled transition t yields the marking $M' = M + C(\cdot, t)$, written as $M[t\rangle M'$. When a sequence of transitions $\sigma = t_1 t_2 \cdots t_k$ is considered, $M[\sigma\rangle M'$ means that the firing of σ from M leads to M'.

For a given marked Petri net $\langle N, M_0\rangle$, the set of markings reachable from M_0 is called the *reachability set* of $\langle N, M_0\rangle$ and is denoted as $R(N, M_0)$. A marked Petri net $\langle N, M_0\rangle$ is said to be bounded if there exists a positive integer k such that $\forall p \in P,\ \forall M \in R(N, M_0),\ M(p) \leq k$.

B. Synchronized Petri nets

A *synchronized Petri net* (SynPN) is a structure $N_s = \langle N, E, f\rangle$ such that:

- N is a Petri net,
- E is an alphabet of external input events,
- $f : T \to E_\lambda = E \cup \{\lambda\}$ is a labeling function that associates with each transition t either an external input event $f(t) \in E$ or λ, where λ is the "always occurring" event which has priority over all external input events.

In this paper, we consider that the synchronized Petri net is *deterministic*, i.e., transitions which are in structural conflict, do not share the same input event (i.e., there is no place p such that $\exists\ t, t' \in p^\bullet$, $t \neq t'$ and $f(t) = f(t')$).

The set of transitions associated with an event $e \in E_\lambda$ is denoted by $T_e = \{t \in T \mid f(t) = e\}$ while the set of enabled transitions associated with event e at marking M is denoted by $\zeta_e(M) = T_e \cap \zeta(M)$.

We adopt for this formalism, a *single server semantics* [1], i.e., at most one instance of a firing per transition in every state. This means that, when input event $e \in E_\lambda$ occurs from marking M, each transition t in T_e which is enabled, fires only once, i.e., transition t appears only once in $\zeta_e(M)$.

A marking M is said to be an *unstable marking* if there exists a transition $t \in \zeta_\lambda(M)$. Otherwise the marking is a *stable marking* and will be denoted as M^s in the next.

Example 1: Consider the SynPN represented in Figure 1(a) and defined by $N_s = \langle N, E, f\rangle$ with: $P = \{p_1, p_2, p_3, p_4, p_5\}$, $T = \{t_1, t_2, t_3, t_4, t_5, t_6\}$, $E = \{a, b\}$. The labeling function is given by $f(t_1) = f(t_5) = a$,

$f(t_2) = b$ and $f(t_3) = f(t_4) = f(t_6) = \lambda$. The initial marking depicted on the figure is $M_0 = (2\ 0\ 0\ 0\ 0)^T$ which can be denoted as $M_0 = 2p_1$. Note that $\zeta_a(M_0) = \{t_1\}$ since $T_a = \{t_1, t_5\}$ and $\zeta(M_0) = \{t_1\}$. Finally, M_0 is a stable marking since $\zeta_\lambda(M_0) = \emptyset$.

The firing of transitions in a SynPN is driven by the occurrence of external input events and by the always occurring event λ. Thus, at marking M, transition $t \in T$ fires if one of following conditions is satisfied:
(i) transition t is enabled and $f(t) = \lambda$, i.e., $t \in \zeta_\lambda(M)$.
(ii) transition t is enabled, $\zeta_\lambda(M) = \emptyset$, and the input event $e = f(t) \in E$ occurs, i.e., $t \in \zeta_e(M)$.

Observe that for a transition both conditions cannot be satisfied simultaneously. For two concurrent transitions t and t' such that $f(t) = \lambda$ and $f(t') = e$, transition t fires first and then condition (i) has priority over condition (ii) in a certain sense.

Given a marking M, the group of k transitions belonging to the set $\zeta_e(M) = \{t_{j_1}, \cdots, t_{j_k}\}$ with $e \in E_\lambda$ fire simultaneously in one step when event e occurs. This leads to the concept of *Elementary Firing Sequence (EFS)*, denoted as $\tau_e(M) = [t_{j_1}, \cdots, t_{j_k}]$, where the brackets are used to denote that the firing order of the k transitions can be arbitrary chosen since they fire simultaneously. The notation $M[\tau_e(M)\rangle M'$ means that the firing of the EFS $\tau_e(M)$ yields marking M'. Due to the single firing policy, one can remark that each transition $t \in T$ may appear at most one time in a given EFS.

Finally, in a deterministic bounded SynPN with a single server semantics, the net evolution is given, from a stable marking M_1^s, as follows:

1) capture the new event, $e \in E$;
2) fire simultaneously all enabled transitions $\zeta_e(M_1^s)$ associated with this event in a single step, i.e., $M_1^s[\tau_e(M_1^s)\rangle M'$ with $M' = M_1^s + \sum_{t \in \tau_e(M_1^s)} C(\cdot, t)$;
3) fire zero, one or more EFS of the always occurring event, $\tau_\lambda(M')$, $\tau_\lambda(M''), \cdots$ corresponding to the successive enabled transitions sets $\zeta_\lambda(M'), \zeta_\lambda(M''), \cdots$ leading to the next stable marking M_2^s i.e. $M'[\tau_\lambda(M')\rangle M''[\tau_\lambda(M'')\rangle \cdots \rangle M_2^s$.

The set of reachable markings from an initial marking M_0 in a synchronized Petri net N_s is denoted as $R(N_s, M_0)$. The synchronization generates constraints in the evolution of $\langle N_s, M_0\rangle$ compared with $\langle N, M_0\rangle$. It follows that [3], the set of stable reachable markings of $\langle N_s, M_0\rangle$ (and even the set of all the reachable markings) is included in the set of reachable markings of its corresponding $\langle N, M_0\rangle$, i.e., $R(N_s, M_0) \subseteq R(N, M_0)$. In the case of bounded SynPN, the set of reachable markings is finite and could be represented by a reachability graph, i.e., a directed graph whose vertices correspond to reachable markings and whose edges correspond to the event causing the marking change associated with the fired EFS, i.e., $e|\tau_e(M)$.

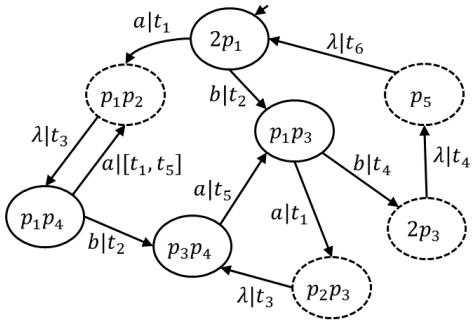

Fig. 2: Reachability graph of the SynPN in Figure 1

Example 2: Let us consider the bounded deterministic SynPN of Figure 1(a) with the initial marking $M_0 = (2\ 0\ 0\ 0\ 0)^T$. With a single server policy, its reachability graph, given in Figure 2, shows four unstable markings: $(0\ 1\ 1\ 0\ 0)^T$, $(0\ 0\ 2\ 0\ 0)^T$, $(0\ 0\ 0\ 0\ 1)^T$, $(1\ 1\ 0\ 0\ 0)^T$ (see dashed circles) and four stable markings: M_0, $(1\ 0\ 0\ 1\ 0)^T$, $(0\ 0\ 1\ 1\ 0)^T$, $(1\ 0\ 1\ 0\ 0)^T$.

IV. Output Synchronized Petri nets and Labeled Finite Automata with Inputs

This section presents a particular class of synchronized Petri nets that generates output events, called Output Synchronized Petri nets. This new formalism allows the system to be controlled through input event associated with controllable transitions and to be observed thanks to output events related to marking changes and possibly with marking states. Next, the reachability graph of this model is represented by a particular class of automata, called a labeled finite state automaton with inputs.

A. Output Synchronized Petri nets

Let us recall that only deterministic bounded Output Synchronized Petri nets with a single server semantics are considered in this paper. Let us first introduce some necessary notions. A Boolean algebra is defined on $\{0, 1\}$ with conjuction, disjunction and negation operators, respectively denoted as $\wedge, \vee, \neg$. A Boolean function F is defined as $F : \{0,1\}^r \to \{0,1\}$, where $\{0,1\}^r$ is the set of r Boolean variables. We denote by $\mathcal{F}_r$ the set of Boolean functions that can be defined on $\{0,1\}^r$.

Definition 1: An *output synchronized Petri net* (OutSynPN) is a structure $N_{os} = \langle N, E, f, \Sigma, \Gamma, Q, g\rangle$ such that:

- $\langle N, E, f\rangle$ is a SynPN;
- $\Sigma \subseteq \{\uparrow M(p_i), \downarrow M(p_i) \mid p_i \in P\}$ is a non empty set of events associated with a marking change of places, where $\downarrow$ and $\uparrow$ represent any decreasing and any increasing of a place marking, respectively;
- $\Gamma \subseteq \{M(p_i) \sim h, \mid p_i \in P, h \in \mathbb{N}, \sim \in \{=, \neq, \geq, \leq, >, <\}\}$ is a set of conditions on the place marking;
- Q is an alphabet of output events;
- $g : Q \to \{0, 1\}$ is an output function such that $\forall q_i \in Q$, $g(q_i) = \Theta(F_\Sigma(q_i)) \wedge \Upsilon(F_\Gamma(q_i))$, where $F_\Sigma(q_i) \in \mathcal{F}_{|\Sigma|}$, $F_\Gamma(q_i) \in \mathcal{F}_{|\Gamma|}$ and :

– $\Theta : \mathcal{F}_{|\Sigma|} \rightarrow \{0,1\}$ is a Boolean function depicting the conditions on the marking change events to generate output q_i and $\Theta(.) = 0$ when no event on the marking change is involved for output q_i;
– $\Upsilon : \mathcal{F}_{|\Gamma|} \rightarrow \{0,1\}$ is a Boolean function depicting the conditions on the marking value of places to generate output q_i and $\Upsilon(.) = 1$ when no condition on the marking values is involved for output q_i. ▲

An OutSynPN is a SynPN that delivers output events. Each output event, $q \in Q$, is associated with at least one marking change of a given place, i.e., $F_\Sigma(q_i)$ exists, along with possible conditions on the marking values of one or several places. Hence, a single firing of an EFS may generate zero, one or more output events. Let us denote $Q(M,e) = \{q_i, \cdots, q_k\} \in 2^Q$ with $q_i \neq q_k$, the subset of output events generated by the firing of EFS $\tau_e(M)$. Observe that the firing of $\tau_e(M)$ generates at most one occurrence of each output event.

Example 3: Consider the OutSynPN, $N_{os} = \langle N, E, f, \Sigma, \Gamma, Q, g\rangle$ where $\langle N, E, f\rangle$ is the SynPN of Example 1. The set of marking events and the set of marking conditions are respectively, $\Sigma = \{\downarrow M(p_1), \uparrow M(p_4), \downarrow M(p_5)\}$ and $\Gamma = \{(M(p_3) = 1)\}$, while the alphabet of output events is $Q = \{A, B, C\}$. The Boolean functions are given by: $F_\Sigma(A) = (\downarrow M(p_1))$, $F_\Sigma(B) = (\uparrow M(p_4))$, $F_\Sigma(C) = (\downarrow M(p_5))$ and $F_\Gamma(B) = (M(p_3) = 1)$. Let us remark that $\Upsilon(F_\Gamma(A)) = \Upsilon(F_\Gamma(C)) = 1$. Finally, with a slight abuse of notations, the output functions are given by: $g(A) = (\downarrow M(p_1))$, $g(B) = ((\uparrow M(p_4)) \wedge (M(p_3) = 1))$ and $g(C) = (\downarrow M(p_5))$, as depicted in Figure 1(b).

From $M_0 = 2p_1$, the occurrence of input event a drives the model to marking $M_1 = p_1 p_2$ by firing one time transition t_1 (due to the single server semantics assumption). As the marking of p_1 decreases, $g(A) = 1$ and consequently, output event A is generated, i.e., $Q(M_0, a) = A$.

B. Labeled Finite Automata with Inputs

The reachability set of an OutSynPN corresponds to the reachability set of its corresponding SynPN, i.e., $R(N_{os}, M_0) = R(N_s, M_0)$. The reachability graph of an OutSynPN is represented by a particular class of labeled finite automaton called in this paper a *labeled finite state automaton with inputs (LFAI)*, defined below.

Definition 2: A *labeled finite automaton with inputs* (LFAI) is a 6-tuple $G = (X, E_\lambda, \delta, x_0, Q, Obs)$, where

- X is a finite *set of states*,
- E is a finite set of *symbols (i.e., input events)* and $E_\lambda = E \cup \{\lambda\}$,
- $\delta : X \times E_\lambda \rightarrow X$ is a (possibly partially defined) *transition function*,
- $x_0 \in X$ is an *initial state*,
- Q is a finite set of *labels (i.e., output events* and $Q_\varepsilon = Q \cup \{\varepsilon\}$,
- $Obs : X \times E_\lambda \rightarrow 2^Q \cup \{\varepsilon\}$ is a *labeling function* . ▲

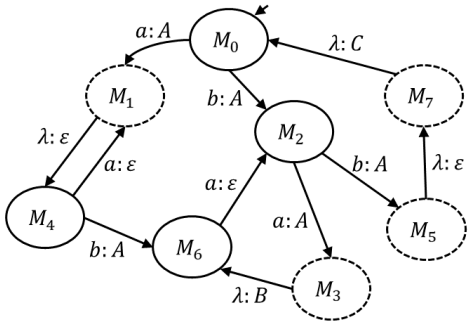

Fig. 3: LFAI of the OutSynPN in example 3

TABLE I: States of the LFAI

X	M	X	M
M_0	$(2\ 0\ 0\ 0\ 0)^T$	M_4	$(1\ 0\ 0\ 1\ 0)^T$
M_1	$(1\ 1\ 0\ 0\ 0)^T$	M_5	$(0\ 0\ 2\ 0\ 0)^T$
M_2	$(1\ 0\ 1\ 0\ 0)^T$	M_6	$(0\ 0\ 1\ 1\ 0)^T$
M_3	$(0\ 1\ 1\ 0\ 0)^T$	M_7	$(0\ 0\ 0\ 0\ 1)^T$

The reachability graph of a given OutSynPN is a particular LFAI with $X = R(N_{os}, M_0)$ and $x_0 = M_0$, each state being a reachable marking M of the OutSynPN. The transition function satisfies $\delta(M, e) = M'$ if $M[\tau_e(M)\rangle M'$. The labeling function depends on the symbols and also on the current state. Obs is defined such that $Obs(M, e) = Q(M, e)$ if $Q(M, e) \neq \emptyset$ and $Obs(M, e) = \varepsilon$ otherwise. Note that ε is used when no label is generated. Given a state, $M \in X$, and a symbol, $e \in E_\lambda$, such that $M' = \delta(M, e)$, we refer to the transition from M to M' as an e-transition.

Example 4: Consider the OutSynPN of Example 3. Its LFAI, given in Figure 3, is defined by a set of 8 states $X = \{M_0, \cdots M_7\}$ that correspond to the 8 reachable markings of the OutSynPN (see Table I), including the initial state M_0. The transition and labeling functions, defined according to Definition 2, are directly derived from the OutSynPN. For example, $\delta(M_0, a) = M_1$ and $Obs(M_0, a) = A$, the same holds for the other transitions. The notation "$a : A$" means that the system switches from state M_0 to state M_1 when it receives symbol a and that this change delivers label A.

A labeled finite automaton with inputs differs from a Mealy machine [20] as in this formalism, with each transition are associated a single input event and a single output event, while in a LFAI a non negative number of output labels could be associated with a transition. It is also different from an I/O automaton [21] where the notion of output refers to output actions and not to observations. In addition, compared to I/O automata, inputs events and output observations are both possible in the same transition of a LFAI.

V. Input / Output - Based Observer

In this section, a standard approach, that transforms a non deterministic automaton into a deterministic one [4], [5], is used to compute logical observers for a given OutSynPN. Each state of the resulting observer is a subset of LFAI states, and consequently of the OutSynPN markings. In particular, depending on the information (inputs, outputs or inputs/outputs) used to infer the system states consistent with the observations, three logical observers of a given OutSynPN could be obtained. In the sequel, an input / output - based observer, denoted as OBS_{IO}, that uses both the input symbols and the output labels as observations is detailed.

Let us consider next an OutSynPN system $\langle N_{os}, M_0\rangle$ with $N_{os} = \langle N, E, f, \Sigma, \Gamma, Q, g\rangle$ and its reachability graph represented with the LFAI $G = (X, E_\lambda, \delta, x_0, Q, Obs)$. The input / output - based observer uses both the input symbols and the output labels as observations.

Definition 3: The *input / output - based observer* of OutSynPN system $\langle N_{os}, M_0\rangle$ is defined by the 4-tuple $OBS_{IO} = (S_{IO}, Q_{IO}, \delta_{IO}, s_0)$ with:

- $S_{IO} \subseteq 2^X$, the set of observer states with $X = R(N_{os}, M_0)$;
- $Q_{IO} \subseteq (E \times 2^Q) \cup E \cup 2^Q$, the set of extended observable labels (that includes the input symbols);
- δ_{IO}, the transition function defined by $\delta_{IO}(s, q_{IO}) = s'$ if there exists two markings $M \in s$, $M' \in s'$ and $q_{IO} \in Q_{IO}$ that satisfy either,
  - $q_{IO} = (e, Q(M, e))$ with $e \in E$, $\delta(M, e) = M'$ and $Obs(M, e) \neq \varepsilon$;
  - $q_{IO} = e$ with $e \in E$, $\delta(M, e) = M'$ and $Obs(M, e) = \varepsilon$;
  - $q_{IO} = Q(M, \lambda)$ with $\delta(M, \lambda) = M'$ and $Obs(M, \lambda) \neq \varepsilon$;
- s_0, the observer initial state. ▲

Algorithm 1 details the computation of the observer OBS_{IO}. It uses the two following subsets of states:

- for each $M \in X$, $e \in E_\lambda$, $Q' \subseteq Q$, let $S(M, e, Q')$ be the set of states that are reachable from M by firing exactly one e-transition (i.e., transition of the LFAI associated with symbol e) with $Obs(M, e) = Q'$. Observe that e may be λ.
- for each state $M \in X$, let $S(M, \lambda, \varepsilon)$ be the set of states reachable from M by firing zero or more silent λ-transitions (i.e., $Obs(M, \lambda) = \varepsilon$). Note that $M \in S(M, \lambda, \varepsilon)$.

Note that the sets $S(M, \lambda, \varepsilon)$ and $S(M, e, Q')$ are computed according to the transition function δ and the labeling function Obs of the LFAI. The complexity of OBS_{IO} is $O(2^{|R(N_{os}, M_0)|})$.

Algorithm 1: Logical observer OBS of an LFAI

Require: : X, M_0, E, Obs, Q

Ensure: : S, δ_{IO}, s_0

```
s ← S(M_0, λ, ε), s_0 ← s, S ← {s}, UNEXs ← {s},
δ_IO ← ∅,
while UNEXs ≠ ∅ do
   let s be the first element of UNEXs
   remove s from UNEXs
   for each symbol e ∈ E_λ do
      for each label Q' ⊆ Q_ε do
         if (e, Q') ≠ (λ, ε) then
            y ← ∅,s' ← ∅,
            for each M ∈ s do
               y ← y ∪ S(M, e, Q')
            end for
            for each M ∈ y do
               s' ← s' ∪ S(M, λ, ε)
            end for
            if s' ∉ S then
               S ← S∪{s'}, UNEXs ← UNEXs∪{s'}
               δ_IO(s, (e, Q')) ← s'
            end if
         end for
      end for
   end for
end while
```

Proposition 1: Given an OutSynPN $\langle N_{os}, M_0\rangle$, its input / output - based observer $OBS_{IO} = (S_{IO}, Q_{IO}, \delta_{IO}, s_0)$ is a deterministic finite automaton.

Proof: To prove that OBS_{IO} is a deterministic finite automaton, we prove that OBS_{IO} has no non-determinism.

First, observe that no transition is labeled with (λ, ε) in OBS_{IO}. If such a transition exists from state M to state M' in the LFAI, then $M' \in S(M, \lambda, \varepsilon)$ and both states M and M' belong to the same observer state s.

Second, multiple transitions outgoing from one state s in OBS_{IO} cannot produce the same extended label $q_{IO} = (e, Q')$. If there are two states $M_1, M_2 \in s$ and two transitions in the LFAI, the first one such that $\delta(M_1, e) = M_1'$ and the second one such that $\delta(M_2, e) = M_2'$ with $Q' = Q(M_1, e) = Q(M_2, e)$, then $M_1' \in S(M_1, e, Q')$, $M_2' \in S(M_2, e, Q')$ and there exists $s' \in S_{IO}$, with $S(M_1, e, Q') \subseteq s'$ and $S(M_2, e, Q') \subseteq s'$ and, both states M_1' and M_2' belong to the same observer state s'. □

Let us consider a sequence of k successive observations $\sigma = q_{IO,1}...q_{IO,k}$ with $q_{IO,h} \in Q_{IO}$, $h = 1, ..., k$. The current state is estimated according to the following steps:

1) Let s_0=$S(M_0, \lambda, \varepsilon)$ be the set of states reachable from M_0 by executing zero or more silent λ-transitions. Before the occurrence of the first observation q_1, one can state that the current state of the system belongs to the set s_0.

2) Consider now the first observation $q_{IO,1}$ of σ with $q_{IO,1}$ being in the form of $q_{IO,1} = (e_1, Q_1')$. For each $M \in s_0$, one can compute first the subset of states $S(M, e_1, Q_1')$ that are reachable from M_0 firing only one e_1-transition that delivers the set of output labels Q_1', and second the

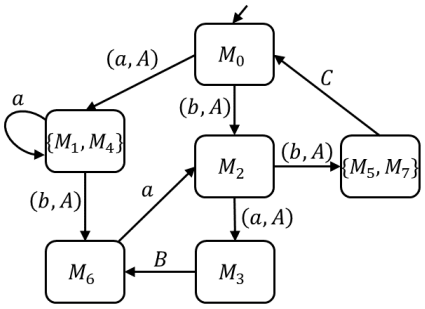

Fig. 4: Input/Output observer of the OutSynPN of Example 3

subset of states $S(M', \lambda, \varepsilon)$ reachable from any of the states $M' \in S(M, e_1, Q'_1)$ by executing zero or more silent λ-transitions. Let us define $s'_1 = \cup_{M \in s_0} S(M, e_1, Q'_1)$ and $s_1 = \cup_{M' \in s'_1} S(M', \lambda, \varepsilon)$. So, s_1 is the subset of states that are consistent with the first observation (e_1, Q'_1).

3) The same is repeated for each observation $q_{IO,h} = (e_h, Q'_h)$, $h = 1, ..., k$. Consider the last observation $q_{IO,k} = (e_k, Q'_k)$ of σ. For each $M \in s_{k-1}$ one can compute first the subset of states $S(M, e_k, Q'_k)$ that are reachable from M executing only one e_k-transition that delivers the output label Q'_k, and second the subset of states $S(x', \lambda, \varepsilon)$ reachable from any of the states $M' \in S(M, e_k, Q'_k)$ by executing zero or more silent λ-transitions. Let us define $s'_k = \cup_{M \in s_{k-1}} S(M, e_k, Q'_k)$ and $s_k = \cup_{M' \in s'_k} S(M', \lambda, \varepsilon)$. So, s_k is the subset of states that are consistent with the whole sequence of observations σ.

Example 5: The input / output - based observer of the OutSynPN of Example 3 is detailed in Figure 4. This observer has 6 states and a set of extended observable labels defined by $Q_{IO} = \{(a, A), (b, A), B, a, C\}$. The notation "$(a, A)$" means that the observation is a pair of information formed by symbol a and label A, and that this observation is used to estimate the current state of the system. It is able to track the system states that are consistent with a given sequence of input and output observations. For instance, from M_0 and the observation of the following sequence of extended labels (a, A) (b, A) a, one can conclude that the system state is currently M_2.

For the proposed OBS_{IO}, only the pair (λ, ε) is regarded as a silent event, and other pairs of symbols and labels are regarded as observable events. Observe that an input - based observer, i.e., that uses only the symbols as observations, and an output - based observer, i.e., that uses only the labels as observations, can be computed in a similar way.

VI. Conclusions

This paper has proposed a method to design observers for discrete event systems that are synchronized thanks to a set of control events and that deliver sets of observations at each state transition. The approach is based on a new class of synchronized Petri nets with outputs that are suitable to model a large variety of Cyber-Physical Systems. Another advantage of the proposed approach is to formalize the observers in a quite standard way using labeled finite automata with inputs.

Our future works will be to extend the design of observers in a time setting for input / output discrete event systems. Another perspective will consider applications of the approach to important problems such as the detection of cyber-attacks in the framework of Cyber-Physical Systems.

References

[1] M. Moalla, J. Pulou, and J. Sifakis, "Synchronized Petri nets : A model for the description of non-autonomous sytems," in *Mathematical Foundations of Computer Science 1978*, J. Winkowski, Ed. Springer Berlin Heidelberg, 1978, pp. 374–384.

[2] M. P. Cabasino, C. N. Hadjicostis, and C. Seatzu, "Marking observer in labeled Petri nets with application to supervisory control," *IEEE Trans. on Automatic Control*, vol. 62, no. 4, pp. 1813–1824, 2017.

[3] R. David and H. Alla, *Discrete, continuous, and hybrid Petri nets*. Springer, 2010, vol. 1.

[4] C. G. Cassandras and S. Lafortune, *Introduction to discrete event systems*. Springer Science & Business Media, 2009.

[5] C. N. Hadjicostis, *Estimation and Inference in Discrete Event Systems*. Springer, 2020.

[6] X. Xiangru and H. Yiguang, "Observability analysis and observer design for finite automata via matrix approach," *IET Control Theory & Applications*, vol. 7, no. 12, pp. 1609–1615, 2013.

[7] C. M. Ozveren and A. S. Willsky, "Observability of discrete event dynamic systems," *IEEE Trans. on Automatic Control*, vol. 35, no. 7, pp. 797–806, 1990.

[8] A. Balluchi, L. Benvenuti, M. D. Di Benedetto, and A. L. Sangiovanni-Vincentelli, "Design of observers for hybrid systems," in *International Workshop on Hybrid Systems: Computation and Control*. Springer, 2002, pp. 76–89.

[9] X. Geng and J. Hammer, "Input/output control of asynchronous sequential machines," *IEEE Trans. on Automatic Control*, vol. 50, no. 12, pp. 1956–1970, 2005.

[10] A. Giua and C. Seatzu, "Observability of place/transition nets," *IEEE Trans. on Automatic Control*, vol. 47, no. 9, pp. 1424–1437, 2002.

[11] M. P. Cabasino, A. Giua, and C. Seatzu, "Fault detection for discrete event systems using Petri nets with unobservable transitions," *Automatica*, vol. 46, no. 9, pp. 1531 – 1539, 2010.

[12] L. Hardouin, C. A. Maia, B. Cottenceau, and M. Lhommeau, "Observer design for $(\max, +)$ linear systems," *IEEE Trans. on Automatic Control*, vol. 55, no. 2, pp. 538–543, 2010.

[13] L. Hardouin, Y. Shang, C. A. Maia, and B. Cottenceau, "Observer-based controllers for max-plus linear systems," *IEEE Trans. on Automatic Control*, vol. 62, no. 5, pp. 2153–2165, 2017.

[14] A. Ramírez-Treviño, I. Rivera-Rangel, and E. López-Mellado, "Observability of discrete event systems modeled by interpreted Petri nets," *IEEE Trans. on Robotics and Automation*, vol. 19, no. 4, pp. 557–565, 2003.

[15] Y. Ru and C. N. Hadjicostis, "Bounds on the number of markings consistent with label observations in Petri nets," *IEEE Trans. on Automation Science and Engineering*, vol. 6, no. 2, pp. 334–344, 2009.

[16] Z. Ma, Y. Tong, Z. Li, and A. Giua, "Marking estimation in labelled Petri nets by the representative marking graph," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 11 175 – 11 181, 2017, 20th IFAC World Congress.

[17] Y. Ru, M. P. Cabasino, A. Giua, and C. N. Hadjicostis, "Supervisor synthesis for discrete event systems under partial observation and arbitrary forbidden state specifications," *Discrete Event Dynamic Systems*, vol. 24, no. 3, p. 275–307, Sept. 2014.

[18] M. P. Cabasino, C. N. Hadjicostis, and C. Seatzu, "Probabilistic marking estimation in labeled Petri nets," *IEEE Trans. on Automatic Control*, vol. 60, no. 2, pp. 528–533, 2015.

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

[20] G. H. Mealy, "A method for synthesizing sequential circuits," *Bell System Technical Journal*, vol. 34, no. 5, pp. 1045–1079, 1955.

[21] N. A. Lynch and M. R. Tuttle, "An introduction to input/output automata," *CWI Quarterly*, vol. 2, pp. 219–246, 1989.