Observation Equivalence of Petri Net Generators

Yin Tong, Zhiwu Li, Alessandro Giua

Abstract

In the framework of discrete event systems, the observation of events is considered as outputs in most problem settings, such as state estimation and fault diagnosis, while only a few studies exploit the availability of partial state information. In this paper, equivalent relations between Petri nets with different observation structures, i.e., Petri net generators, are discussed. We focus on a model called labeled Petri nets with outputs (LPNO), i.e., labeled Petri nets endowed with an arbitrary observation function of the marking. It has been shown that any LPNO can be converted into an adaptive labeled Petri net (ALPN) whose labeling function depends on the markings, if infinite labels are allowed. We propose an algorithm that converts a bounded LPNO into an equivalent ALPN with minimal alphabet, i.e., an alphabet of minimal cardinality, to avoid introducing a large number of unnecessary labels.

Keywords: Discrete event systems, sensors, Petri nets, observation, equivalence

Published as:
Y. Tong, Z. W. Li, and A. Giua. "Observation Equivalence of Petri Net Generators." WODES14: 12th Int. Work. on Discrete Event Systems (Cachan, France), May 2014. DOI: 10.3182/20140514-3-FR-4046.00060

Yin Tong and Zhiwu Li are with the School of Electro-Mechanical Engineering, Xidian University, Xi'an 710071, China (e-mail: yintong@stu.xidian.edu.cn; systemcontrol@gmail.com)

Alessandro Giua is with Department of Electrical and Electronic Engineering, University of Cagliari, Italy and also with Aix-Marseille University, LSIS, France (e-mail: giua@diee.unica.it)

This work was supported in part by the National Natural Science Foundation of China under Grant No. 61374068, the Recruitment Program of Global Experts, and the NPST project under Grant No. 12-ELE2506-02.

1. Introduction

A discrete event system (DES) is a dynamic system that evolves in accordance with the abrupt occurrence, at possibly unknown irregular intervals, of physical events [11]. DES are described in terms of states and events and, correspondingly, there are two main primitives to associate an observation structure with these models: we may assume that either the states or the event occurrences are observable, i.e., measurable by a sensor.

When automata models are considered, it is commonly assumed that only event occurrences are observed [9, 14]. The same choice is also common for Petri net models and in particular *labeled Petri nets* (LPN), i.e., generators whose outputs are labels associated with transitions, have been used for supervisory control [1], fault diagnosis [2, 3, 6] and state estimation [7] etc.

Several authors [16, 12, 13] consider an extended LPN model also endowed with state observations. They assume that the token content in some places of the net is measurable. However, [13] have shown that such a type of LPN can always be converted into an equivalent standard LPN by a suitable re-definition of the transition labels, and hence the two models have the same modeling power.

In some circumstances, e.g, due to economical or technological issues, state sensors of may only provide aggregate state information instead of a precise measurement. For example, a cheap sensor may just be able to detect if a buffer is empty or not while the exact number of items in the buffer remains unknown. This motivates us to look for more general observation structures in Petri net generators, than those considered in the above mentioned works.

In [15] we introduced *labeled Petri nets with outputs* (LPNO), i.e., labeled Petri nets endowed with state sensors providing an observation that is an arbitrary function of the current net marking. Using an LPNO to model a system, state observations are modeled by the output function and the net structure is identical to the one without considering the state sensors. We tried to unify event and state observations in LPNO without changing the net structure, so that only event occurrences are taken into account, and we may use techniques for LPN to analyze this more general type of generators. Nevertheless, we found that such a conversion is not always possible if transitions have to be assigned a static label, as is the case with LPN. As a result, we define a more general event-based observation structure — *adaptive labeled Petri nets* (ALPN). In this type of Petri net generators only transition firings are observable but labels associated with transitions are adaptive, i.e., depending on the current markings. Furthermore, we have shown that there always exists an ALPN observation equivalent to an LPNO, but this may require an infinite alphabet, i.e., an alphabet with an infinite number of labels.

Herein, we aim at finding a conversion algorithm that determines an equivalent ALPN with minimal alphabet, i.e. an alphabet with minimal cardinality. There are several practical motivations for this: i) reducing the cardinality of the alphabet may correspond to a cost reduction in the implementation of an observation structure; ii) it may allow one to determine an equivalent net with a finite alphabet even when the brute-force procedure [15] generates an infinite number of labels; iii) this procedure may allow one to show that a given LPNO cannot be converted into an LPN: such is the case if the minimal number of labels is greater than the number of transitions. An algorithm realizing the conversion from a bounded LPNO to the equivalent ALPN with minimal alphabet is proposed.

This paper is structured as follows. Section 2 recalls the basic formalism of Petri nets; three Petri net generators discussed in this paper are formally defined and the further discussion about their observation relationships is presented in Section 3; in Section 4 the conversion algorithm of bounded LPNO is proposed; finally, conclusions and future work are presented.

2. Background on Petri Nets

In this section we introduce the formalism used in the paper. For more details on Petri nets, we refer the reader to [10] for details.

A *Petri net* is a structure $N = (P, T, Pre, Post)$, where P is a set of m *places* represented by circles; T is a set of n *transitions* represented by bars; $Pre : P \times T \to \mathbb{N}$ and $Post : P \times T \to \mathbb{N}$ are the *pre-* and *post-incidence functions* that specify the arcs directed from places to transition, vice versa, where $\mathbb{N}$ is a set of non-negative integers. We also demote by $C = Post - Pre$ the incidence matrix of a net.

A *marking* is a vector $M : P \to \mathbb{N}^m$ that assigns to each place of a Petri net a non-negative integer number of tokens, represented by black dots. We denote $M(p)$ the marking of place p. A *Petri net system* $\langle N, M_0 \rangle$ is a net N with an initial marking M_0.

A transition t is *enabled* at M if $M \geq Pre(\cdot, t)$ and may fire yielding a new marking $M' = M_0 + C(\cdot, t)$. We write $M[\sigma\rangle$ to denote that the sequence of transitions $\sigma = t_{j1} \cdots t_{jk}$ is enabled at M, and $M[\sigma\rangle M'$ to denote that the firing of σ yields M'. The set of all sequences that can fire in a net system $\langle N, M_0 \rangle$ is denoted by $\mathcal{L}(N, M_0)$.

A marking M is *reachable* in $\langle N, M_0 \rangle$ if there exists a firable sequence $\sigma \in \mathcal{L}(N, M_0)$ such that $M_0[\sigma\rangle M$. The set of all markings reachable from M_0 defines the *reachability set* of $\langle N, M_0 \rangle$ and is denoted by $R(N, M_0)$. A Petri net

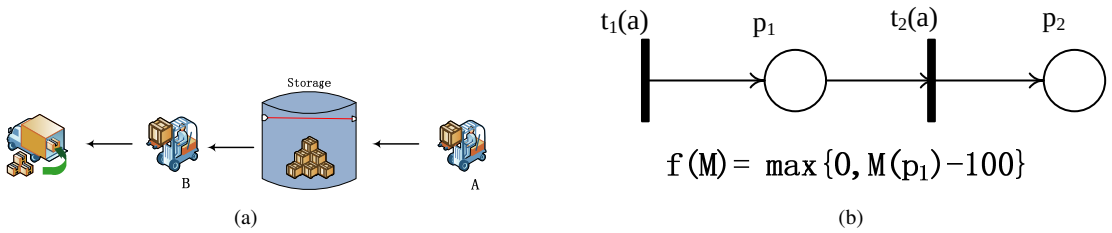


Figure 1. Inventory system and its LPNO model.

system is *bounded* if there exists a $k \in \mathbb{N}$ such that for all places $p \in P$ and for all reachable markings $M \in R(N, M_0)$, $M(p) \leq k$ holds.

3. Classes of Petri net generators

This section recalls the basic definitions of several classes of Petri nets models[1]: labeled Petri nets (LPN), labeled Petri nets with outputs (LPNO) and adaptive labeled Petri nets (ALPN). These models consist of a net structure $\langle N, M_0 \rangle$ plus an observation structure: for this reason we refer to them as *Petri net generators*.

3.1. Labeled Petri Nets

A *labeled Petri net* (LPN) is a generator $G_L = (N, M_0, \Sigma, \ell)$, where N is a Petri net with initial marking M_0, Σ is an *alphabet* (a set of labels), and $\ell : T \to \Sigma \cup \{\varepsilon\}$ is a *labeling function* that assigns to each transition $t \in T$ either a symbol from Σ or the empty string ε.

We denote T_e the set of transitions whose label is $e \in \Sigma \cup \{\varepsilon\}$, i.e., $T_e = \{t \in T \mid \ell(t) = e\}$. A transition $t \in T_e$ is called an e-labeled transition.

To describe the evolution of an LPN, we define its observation function as follows.

Definition 1 Given an LPN $G_L = (N, M_0, \Sigma, \ell)$, its *observation function* $L_L : T^* \to \Sigma^*$ associates a firing sequence $\sigma = t_1 t_2 \cdots t_k$ with the observation $w = L_L(\sigma) = \ell(t_1)\ell(t_2)\cdots\ell(t_k)$. ◇

3.2. Labeled Petri Nets with Outputs

Definition 2 A *labeled Petri net with outputs* (LPNO) is a generator $G_O = (N, M_0, \Sigma, \ell, f)$, where (N, M_0, Σ, ℓ) is an LPN and $f : R(N, M_0) \to \mathbb{R}^k$ is an *output function* associated with $k \in \mathbb{N}$ state sensors. ◇

Definition 3 Given an LPNO $G_O = (N, M_0, \Sigma, \ell, f)$, let $\sigma = t_1 \cdots t_k$ be a firing sequence producing the trajectory $M_0[t_1\rangle M_1 \cdots M_{k-1}[t_k\rangle M_k$. The *observation function* $L_O : T^* \to ((\Sigma \cup \{\varepsilon\}) \times \mathbb{R}^k)^*$ associates sequence σ with the observation

$$s = L_O(\sigma) = (\ell(t_1), \Delta f(M_0, t_1)) \cdots (\ell(t_k), \Delta f(M_{k-1}, t_k)),$$

where $\Delta f(M_{i-1}, t_i) = f(M_i) - f(M_{i-1}) \in \mathbb{R}^k$, $i = 1, 2, \cdots, k$.

If $\ell(t_i) = \varepsilon$ and $\Delta f(M_{i-1}, t_i) = 0$, the observation $(\varepsilon, 0)$ is the *null observation* since no transition firing is detected.◇

Remark: in the following we will implicitly assume (see Definition 6) that M_0 is known, hence the initial observation $f(M_0)$ provides no additional information. In this case the two sequences $f(M_0), f(M_1), \ldots$ and $\Delta f(M_0, t_1), \Delta f(M_1, t_2), \ldots$ contain the same information.

Since state sensors are modeled by a function, by using such a generator, the model structure of a system is able to be as small as the one without considering state sensors.

Example 1 Let us consider the inventory system shown in Figure 1(a). Forklift A delivers products to the storage and forklift B delivers products from the storage to a transporting truck. There is a sensor on the storage detecting whether the inventory is over a threshold (say 100). The arrival/departure of a forklift is detected by another sensor that does not specify which one is moving. This system can be easily modeled by the LPNO in Figure 1(b), where p_1 and p_2 model the storage and the truck, respectively, and t_1 (resp., t_2) models the operation of forklift A (resp., forklift B). A firing sequence $\sigma = t_1^{100} t_1 t_2$ of the LPNO corresponds to an observation $s = L_O(\sigma) = \underbrace{(a, 0) \cdots (a, 0)}_{100}(a, 1)(a, -1)$. ◇

[1] The last two classes were originally defined in [15].

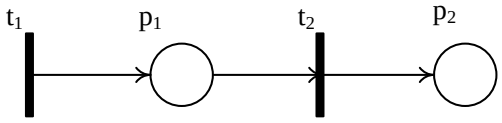


Figure 2. The ALPN in Example 2.

3.3. Adaptive Labeled Petri Nets

Definition 4 An *adaptive labeled Petri net* (ALPN) is a generator $G_A = (N, M_0, \Sigma, \ell_A)$, where $\langle N, M_0 \rangle$ is a Petri net system, Σ is an alphabet and $\ell_A : R(N, M_0) \times T \to \Sigma \cup \{\varepsilon\}$ is an *adaptive labeling function*. $\diamond$

The observation of an ALPN is a sequence of labels in Σ defined as follows.

Definition 5 Given an ALPN $G_A = (N, M_0, \Sigma, \ell_A)$, let $\sigma = t_1 \cdots t_k$ be a firing sequence producing the trajectory $M_0[t_1\rangle M_1 \cdots M_{k-1}[t_k\rangle M_k$. The *observation function* $L_A : T^* \to \Sigma^*$ associates sequence σ with the observation

$$w_A = L_A(\sigma) = \ell_A(M_0, t_1) \cdots \ell_A(M_{k-1}, t_k). \qquad \diamond$$

Example 2 Let us consider the ALPN in Figure 2. Its alphabet is $\Sigma = \{a, a_1, a_2\}$. The adaptive labeling function is $\forall M \in R(N, M_0) : M(p_1) < 100$, $\ell_A(M, t_1) = \ell_A(M, t_2) = a$; otherwise, $\ell_A(M, t_1) = a_1$ and $\ell_A(M, t_2) = a_2$. A firing sequence $\sigma = t_1^{100} t_1 t_2$ of the ALPN corresponds to an observation $w_A = L_A(\sigma) = \underbrace{a \cdots a}_{100} a_1 a_2$. $\diamond$

3.4. Observation equivalence between generators

In the following, we use G_X with $X \in \{L, O, A\}$ to denote a generator without specifying its type and denote the corresponding observation as $x \in \{w, s, w_A\}$, following the notation introduced in the previous subsections.

An observation x allows one to estimate the firing sequence and the current marking of a generator as follows.

Definition 6 Given an observation x in a generator G_X whose underlying net system $\langle N, M_0 \rangle$ is assumed to be known, we define:

- the set of *firing sequences consistent with* x as

$$\mathcal{S}(x) = \{\sigma \in \mathcal{L}(N, M_0) \mid L_X(\sigma) = x\};$$

- the set of *markings consistent with* x as

$$\mathcal{C}(x) = \{M \in \mathbb{N}^m \mid \exists \sigma \in \mathcal{S}(x) : M_0[\sigma\rangle M\}. \qquad \diamond$$

Definition 7 A generator G_X is said to be *observation equivalent* to a generator $G_{X'}$ if i) G_X and G'_X have the same net system $\langle N, M_0 \rangle$, and ii) for all sequence $\sigma \in \mathcal{L}(N, M_0)$ that produce an observation x in G_X and an observation x' in $G_{X'}$, $\mathcal{S}(x) = \mathcal{S}(x')$ holds. $\diamond$

Note that in Definition 7, $\mathcal{S}(x) = \mathcal{S}(x')$ implies $\mathcal{C}(x) = \mathcal{C}(x')$.

Remark: Definition 7 considers generators that have the same behavior (since they have the same underlying system) and defines equivalence with respect to the observation semantics. In this paper, "equivalence" always refers to "observation equivalence".

Example 3 According to Definition 7, the LPNO in Example 1 and the ALPN in Example 2 are equivalent.$\diamond$

We now compare the classes of generators previously defined. An LPN is a restricted type of LPNO whose output function is null. Analogously, we can see an LPN as a restricted type of ALPN such that the label of a transition is the same at every marking. Hence, for an LPN there also exists an equivalent LPNO and an equivalent ALPN while, as shown in [15], the converse is not true. In Figure 5 this is denoted by drawing an arrow from class LPN to the each of the other two classes.

In [15], we have shown that there exists a brute-force procedure to convert an LPNO into an equivalent ALPN with infinite alphabet. We have also shown that for some LPNO an equivalent ALPN with a finite alphabet does not always exist. In Figure 5, this is denoted by drawing a dashed arrow labeled "$+\infty$" from class LPNO to class ALPN.

Here we complete the comparison between these classes showing, by means of Example 4, that there exist ALPN that cannot be converted into an equivalent LPNO.

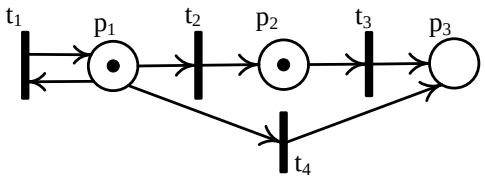


Figure 3. ALPN that cannot be converted into an LPNO.

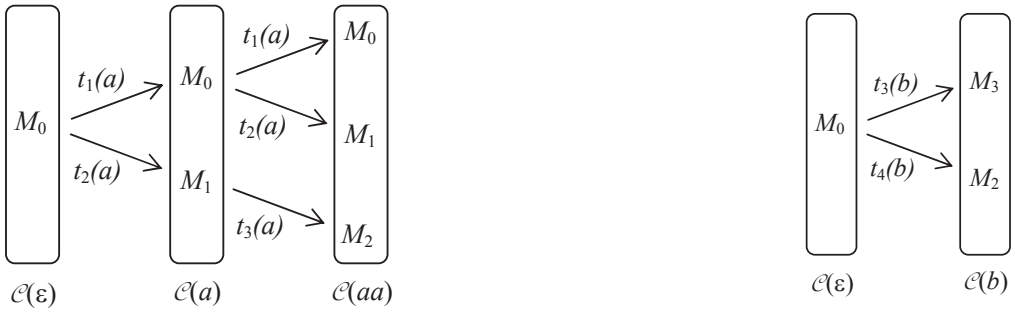


Figure 4. Computation of sets $\mathcal{C}(aa)$ and $\mathcal{C}(b)$ for the ALPN in Example 4.

Example 4 Let us consider the ALPN in Figure 3 with initial marking $M_0 = [1\ 1\ 0]^T$ and the adaptive labeling function given by the following table.

$\ell_A(M,t)$	t_1	t_2	t_3	t_4
M_0	a	a	b	b
$M \neq M_0$	b	b	a	a

For observed words $w_A = aa$ and $w_A = b$, the set $\mathcal{C}(aa)$ and $\mathcal{C}(b)$ can be iteratively computed as shown in Figure 4, where $M_1 = [0\ 2\ 0]^T$, $M_2 = [0\ 1\ 1]^T$ and $M_3 = [1\ 0\ 1]^T$.

We claim that there does not exist an LPNO equivalent to this ALPN. We prove this by contradiction. If we assume such a generator exists, then its observation function must satisfy $f(M_0) = f(M_1) = f(M_2) = f(M_3)$ since otherwise we would be able to distinguish between the first three markings after the firing of two a's or between the last two after the firing of b. In additional, all transitions must necessarily have the same label, say a. However such an LPNO after a would produce a set of consistent markings $\mathcal{C}((a,0)) = \{M_0, M_1, M_2, M_3\} \neq \mathcal{C}(a) = \{M_0, M_1\}$, which contradicts the assumption. ◇

In conclusion, the relationships between LPN, LPNO and ALPN in terms of observation equivalence are summarized in Figure 5.

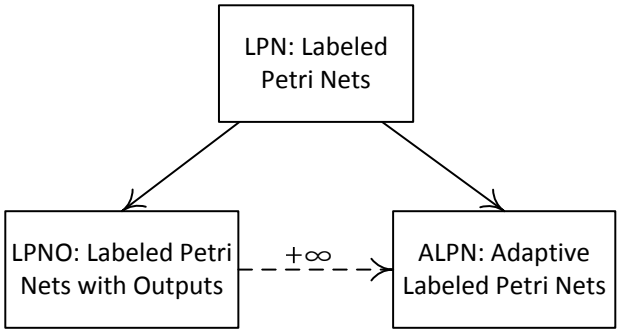


Figure 5. Classes of Petri nets generators and their relationships in terms of observation equivalence.

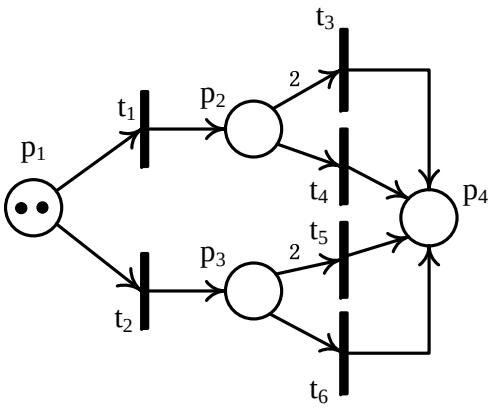


Figure 6. LPNO with a nonlinear output function.

4. Conversion from bounded LPNO to ALPN

In this section we propose an approach to convert a bounded LPNO into an equivalent ALPN. The conversion only requires finding the adaptive labeling function of the ALPN since, according to Definition 7, the two generators must have the same net system.

As mentioned in the previous section, to convert an LPNO into an equivalent ALPN a brute-force procedure can be used: this requires assigning to each transition $t \in T$ at every marking $M \in R(N, M_0)$, where it is enabled, a new label $[\ell(t), \Delta f(M, t)]$. This procedure may produce an infinite number of labels when the LPNO is not bounded and thus may not terminate. When the net is bounded the procedure determines an equivalent ALPN with finite number of labels. Even though for a bounded LPNO an equivalent ALPN with finite alphabet can be obtained, this generator may have a large number of unnecessary labels. We look for a better conversion algorithm that determines an equivalent ALPN with minimal alphabet. Thus, the considered problem is the following.

Problem statement: convert a given bounded LPNO into an equivalent ALPN whose alphabet has minimal cardinality.

The conversion procedure we propose is based on three steps.

Step 1 Since observation equivalence requires the set of consistent markings of the two generators to be identical for all observations, we first determine which pairs of markings are *confusable*.

Step 2 Using this information, we determine which pairs $[M, t] \in R(N, M_0) \times T$ should have the same label in the ALPN constructing the *agreement graph* $\mathcal{G}(e)$ for all $e \in \Sigma \cup \{\varepsilon\}$. We also determine which pairs $[M, t]$ should have a different label in the ALPN constructing the *conflict graph* $\hat{\mathcal{G}}$.

Step 3 Finally, we reduce the problem of finding the label assignment that determines the alphabet of minimal cardinality to the *vertex coloring problem* of graph $\hat{\mathcal{G}}$.

All these steps will be presented and discussed in the following subsections. At the end of the section an algorithm that summarizes all these steps will be given.

4.1. Computation of the confusion relation

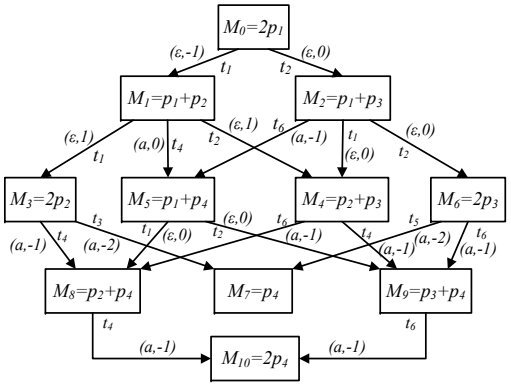
Given an observation in an LPNO, there may be more than one marking consistent with this observation. To describe such a relation between markings, we define the *confusable* relation.

Definition 8 Given an LPNO G_O, a marking M is said *confusable* with M', denoted by $M \sim M'$, if there exists an observation s s.t. $M, M' \in \mathcal{C}(s)$. ◇

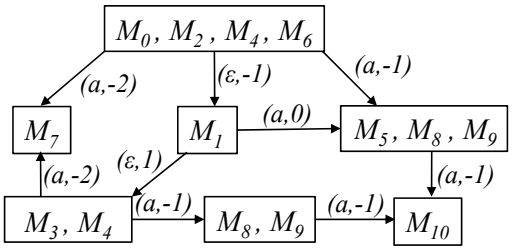
One can readily verify that $M \sim M'$ is a symmetric, refl-exive but not transitive relation.

An intuitive way to compute the confusion relation among all markings is to construct an *observer* [4]. First, since the net is bounded, its reachability graph (RG) can be constructed. This is a graph where each node is a marking M and each arc corresponds to a transition t. We tag each arc t exiting node M with the label $(\ell(t), \Delta f(M, t))$ thus constructing a nondeterministic finite automaton (NFA). Then, the corresponding observer, i.e., the equivalent deterministic finite automaton (DFA), can be constructed. In the DFA, each state is a set $\mathcal{C}(s)$ and all markings in $\mathcal{C}(s)$ are confusable with each other.

Example 5 Consider the LPNO $G_O = (N, M_0, \Sigma, \ell, f)$ in Figure 6, where $T_\varepsilon = \{t_1, t_2\}$, $T_a = \{t_3, t_4, t_5, t_6\}$ and $f(M) = max\{M(p_1), M(p_2)\} + M(p_3)$. Its RG and the observer are shown in Figure 7. ◇



(a)



(b)

Figure 7. The RG and the observer of G_O.

Table 1. All possible observations after a transition firing

(e, Δ)	$\{[M,t] \mid \ell(t) = e, \Delta f(M,t) = \Delta\}$
$(\varepsilon, 0)$	$[M_0, t_2], [M_2, t_1], [M_2, t_2], [M_5, t_1], [M_5, t_2]$
$(\varepsilon, -1)$	$[M_0, t_1]$
$(\varepsilon, 1)$	$[M_1, t_1], [M_1, t_2]$
$(a, -2)$	$[M_3, t_3], [M_6, t_5]$
$(a, -1)$	$[M_2, t_6], [M_3, t_4], [M_4, t_4], [M_4, t_6], [M_6, t_6],$ $[M_8, t_4], [M_9, t_6]$
$(a, 0)$	$[M_1, t_4]$

It is known that the worst-case complexity of computing a DFA equivalent to a NFA is exponential in the number of states of the NFA, therefore, the complexity to determine the confusion relation is exponential in the number of markings.

4.2. Construction of the agreement and conflict graphs

If two transitions t and t' are enabled at two confusable markings M and M', respectively, and produce the same observation, then the two labels $\ell_A(M,t)$ and $\ell_A(M,t')$ must coincide in the ALPN. This can be captured, for all labels $e \in \Sigma \cup \{\varepsilon\}$, by an agreement graph $\mathcal{G}(e)$.

Definition 9 Given an LPNO $G_O = (N, M_0, \Sigma, \ell, f)$ and a label $e \in \Sigma \cup \{\varepsilon\}$, the *agreement graph* of e is an undirected graph $\mathcal{G}(e) = (V(e), E(e))$ where $V(e) = \{[M,t] \in R(N,M_0) \times T_e | M[t\rangle\}$ and $E(e) = \{([M,t],[M',t']) \in V(e) \times V(e) | M \sim M', \Delta f(M,t) = \Delta f(M',t')\}$ ($[M,t] \neq [M',t']$). ◇

The set of nodes $V(e)$ of such a graph includes all pairs $[M,t]$ where t is an e-labeled transition enabled at a marking M. Two nodes $[M,t]$ and $[M',t']$ are connected by an edge in $\mathcal{G}(e)$ if M and M' are confusable and $\Delta f(M,t) = \Delta f(M',t')$. In an agreement graph, there is no self loop.

Example 6 Consider again the LPNO in Example 5. Table 1 is built based on the RG. According to Definition 9, the agreement graphs $\mathcal{G}(\varepsilon)$ (Figure 8(a)) and $\mathcal{G}(a)$ (Figure 8(b)) are constructed. ◇

If $e \in \Sigma$, then the set of nodes of an agreement graph $\mathcal{G}(e)$ can be partitioned into

$$V(e) = \hat{v}_1(e) \dot{\cup} \hat{v}_2(e) \dot{\cup} \cdots \dot{\cup} \hat{v}_l(e)$$

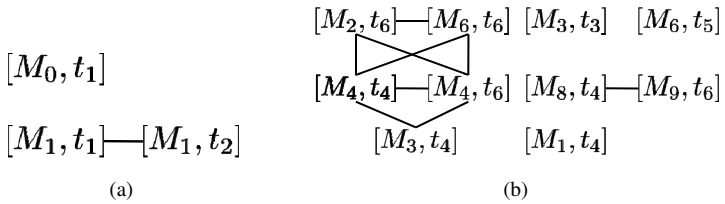


Figure 8. The agreement graphs of G_O.

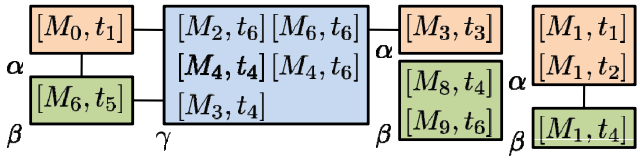


Figure 9. The (colored) conflict graphs of G_O.

where the $\hat{v}_i(e)$-induced subgraph with $i \in \{1, 2, \cdots, l\}$ is a component of $\mathcal{G}(e)$.

If $e = \varepsilon$, then

$$V(\varepsilon) = \hat{v}_0(\varepsilon) \dot{\cup} \hat{v}_1(\varepsilon) \dot{\cup} \hat{v}_2(\varepsilon) \dot{\cup} \cdots \dot{\cup} \hat{v}_l(\varepsilon),$$

where $\hat{v}_0(\varepsilon) = \{[M, t] \in V(\varepsilon) | \Delta f(M, t) = 0\}$, $\hat{v}_i(\varepsilon) \subseteq V(\varepsilon) \setminus \hat{v}_0(\varepsilon)$ and $\hat{v}_i(\varepsilon)$-induced subgraph is a component of $\mathcal{G}(\varepsilon)$ when $i \in \{1, 2, \cdots, l\}$. This is due to the requirement that an ε-labeled transition t at a marking M where its firings produce the null observation should be labeled by ε no matter whether this marking is confusable with other markings or not, i.e., in the ALPN it should hold $\ell_A(M, t) = \varepsilon$. For a label $e \in \Sigma \cup \{\varepsilon\}$, we also define

$$\Pi_e = \{\hat{v}_1(e), \hat{v}_2(e), \cdots, \hat{v}_l\}.$$

Example 7 Consider Table 1 and the agreement graphs $\mathcal{G}(\varepsilon)$ and $\mathcal{G}(a)$ in Example 6.
$\hat{v}_0(\varepsilon) = \{[M_0, t_2], [M_2, t_1], [M_2, t_2], [M_5, t_1], [M_5, t_2]\}$,
$\hat{v}_1(\varepsilon) = \{[M_0, t_1]\}$, $\hat{v}_2(\varepsilon) = \{[M_1, t_1], [M_1, t_2]\}$;
$\hat{v}_1(a) = \{[M_3, t_3]\}$, $\hat{v}_2(a) = \{[M_6, t_5]\}$,
$\hat{v}_3(a) = \{[M_2, t_6], [M_3, t_4], [M_4, t_4], [M_4, t_6], [M_6, t_6]\}$,
$\hat{v}_4(a) = \{[M_8, t_4], [M_9, t_6]\}$, $\hat{v}_5(a) = \{[M_1, t_4]\}$.
$\Pi_\varepsilon = \{\hat{v}_1(\varepsilon), \hat{v}_2(\varepsilon)\}$ and $\Pi_a = \{\hat{v}_1(a), \cdots, \hat{v}_5(a)\}$. ◇

By means of the agreement graph, we have determined the classes of pairs $[M, t]$ that produce the same observation. We now determine, by means of the conflict graph, which classes must be assigned a different label in the ALPN.

Definition 10 Given an LPNO $G_O = (N, M_0, \Sigma, \ell, f)$, the *conflict graph* is an undirected graph $\hat{\mathcal{G}} = (\hat{V}, \hat{E})$ that $\hat{V} = \bigcup_{e \in \Sigma \cup \{\varepsilon\}} \Pi_e$ and $\hat{E} = \{(\hat{v}_i(e), \hat{v}_j(e')) \in \hat{V} \times \hat{V} | \exists [M, t] \in \hat{v}_i(e), \exists [M', t'] \in \hat{v}_j(e') : M \sim M' (e, \Delta f(M, t)) \neq (e', \Delta f(M', t'))\}$. ◇

The nodes of graph $\hat{\mathcal{G}}$ are classes of nodes $\bigcup_{e \in \Sigma \cup \{\varepsilon\}} \Pi_e$ and all nodes in a class produce the same observation. Consider now any two classes $\hat{v}_i(e)$ and $\hat{v}_j(e')$. If there exist $[M, t] \in \hat{v}_i(e)$ and $[M', t'] \in \hat{v}_j(e')$ such that M and M' are confusable but t and t' will produce different observation $(e, \Delta f(M, t))$, $(e', \Delta f(M', t'))$, then in the ALPN, different labels must be assigned to them, i.e., $\ell_A(M, t) \neq \ell_A(M', t')$. Thus we add an arc between $\hat{v}_i(e)$ and $\hat{v}_j(e')$ to denote that the two classes must be assigned a different label.

Example 8 Based on the result in Example 7 and Definition 10, the conflict graph $\hat{\mathcal{G}}$ of G_O is shown in Figure 9. ◇

4.3. Solving the vertex coloring problem

The conflict graph $\hat{\mathcal{G}}$ of an LPNO exactly describes the relabeling rule following which an equivalent ALPN can be obtained. In order to meet the relabeling rule and the requirement of an minimal alphabet, the problem of assigning labels to every node of $\hat{\mathcal{G}}$ such that two adjacent nodes have a different label can be reduced to the vertex coloring problem defined as follows.

Definition 11 [5] A *vertex coloring* of a graph $\mathcal{G} = (V, E)$, where V is the set of nodes and E is the set of edges, is a map $\ell_{col} : V \to \Sigma_{col}$ such that $\ell_{col}(v) \neq \ell_{col}(v')$ if v and v' are adjacent, where $v, v' \in V$ and Σ_{col} is a set of available colors. The *vertex coloring problem* is to find a vertex coloring with the minimal number of colors. ◇

It is well known that the coloring problem is in general NP-complete, while there exist effective polynomial algorithms for some special classes of graphs, see [8]. Let the vertex coloring ℓ_{col} and the set of colors Σ_{col} be the solution of graph $\hat{\mathcal{G}}$, then the corresponding adaptive labeling function is $\forall[M,t] \in \hat{v}_i(e)$, $\ell_A(M,t) = \overline{e}$ if $\ell_{col}(\hat{v}_i(e)) = \overline{e}$, where $\hat{v}_i(e) \in \hat{V}$, and the alphabet of the ALPN is $\Sigma = \Sigma_{col}$. Therefore, an equivalent ALPN with minimal alphabet is obtained. Note that at markings where a transition t is not enabled, the adaptive labeling function is not defined, since t could never produce any observation at those markings.

Example 9 The minimal number of colors to color the conflict graph of GO in Example 5 is three and one possible way to color it is shown in Figure 9. ◇

4.4. Conversion algorithm

The procedure to convert a bounded LPNO into an equivalent ALPN with minimal alphabet can be summarized as in Algorithm 1. To avoid confusion between alphabets in the LPNO and the ALPN, Σ_A is used to denote the alphabet in the ALPN obtained.

Algorithm 1 Conversion of an LPNO into an equivalent ALPN with minimal alphabet

Input: a bounded LPNO $G_O = (N, M_0, \Sigma, \ell, f)$
Output: an equivalent ALPN $G_A = (N, M_0, \Sigma_A, \ell_A)$

```
1: Compute confusion relations.
2: for all e ∈ Σ ∪ {ε}, do
3:     construct G(e) according to Definition 9;
4:     compute Π_e;
5:     if e = ε ∧ v̂_0(e) ≠ ∅, then
6:         ℓ_col(v̂_0(e)) = ε;
7:     end if
8: end for
9: Construct Ĝ according to Definition 10.
10: Solve the vertex coloring problem of Ĝ.
11: Σ_A := Σ_col, ℓ_A := ℓ_col.
```

Example 10 Based on Example 9 and Algorithm 1, the equivalent ALPN with minimal alphabet is $G_A = (N, M_0, \Sigma_A, \ell_A)$, where $\Sigma_A = \{\alpha, \beta, \gamma\}$,
$\ell_A(M_0,t_2) = \ell_A(M_2,t_1) = \ell_A(M_2,t_2) = \ell_A(M_5,t_1) = \ell_A(M_5,t_2) = \varepsilon$, $\ell_A(M_0,t_1) = \ell_A(M_3,t_3) = \ell_A(M_1,t_1) = \ell_A(M_1,t_2) = \alpha$, $\ell_A(M_1,t_4) = \ell_A(M_6,t_5) = \ell_A(M_8,t_4) = \ell_A(M_9,t_6) = \beta$, $\ell_A(M_2,t_6) = \ell_A(M_3,t_4) = \ell_A(M_4,t_4) = \ell_A(M_4,t_6) = \ell_A(M_6,t_6) = \gamma$. ◇

The algorithm outlined above may assign to two pairs $[M,t]$ and $[M',t']$ the same label in the ALPN even if in the original LPNO transitions t and t' have a different label, i.e., $\ell(t) \neq \ell(t')$.

5. Conclusions and Future Work

This paper discusses the observation equivalence relation between different Petri net generators and provides an algorithm converting a bounded LPNO into an equivalent ALPN with minimal alphabet. The complexity of such an algorithm mainly depends on the computation of confusion relations and solving the coloring problem on a particular graph that we call a conflict graph. In the future we plan to solve state estimation and fault diagnosis problems in LPNO generators.

References

[1] K. Barkaoui, A. Chaoui, and B. Zouari. Supervisory control of discrete event systems based on structure theory of petri nets. In *IEEE International Conference on Systems, Man, and Cybernetics*, volume 4, pages 3750–3755, Orlando, FL, 1997.

[2] F. Basile, P. Chiacchio, and G. D. Tommasi. An efficient approach for online diagnosis of discrete event systems. *IEEE Transactions on Automatic Control*, 54(4):748–759, April 2009.

[3] M. Cabasino, A. Giua, and C. Seatzu. Fault detection for discrete event systems using Petri nets with unobservable transitions. *Automatica*, 46(9):1531–1539, September 2010.

[4] C. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Springer, 2nd edition, 2008.

[5] R. Diestel. *Graph theory*. Springer, New York, 3rd edition, 2006.

[6] A. Giua. State estimation and fault detection using Petri nets. In *Proc. 32nd International Conference on Applications and Theory of Petri nets*, Newcastle, UK, June 2011.
[7] A. Giua, C. Seatzu, and D. Corona. Marking estimation of petri nets with silent transitions. *IEEE Transactions on Automatic Control*, 52(9):1695–1699, 2007.
[8] T. R. Jensen and B. Toft. *Graph coloring problems*, volume 39. John Wiley & Sons, 2011.
[9] F. Lin and W. M. Wonham. On observability of discrete-event systems. *Information sciences*, 44(3):173–198, 1988.
[10] T. Murata. Petri nets: properties, analysis and applications. *In Proc. IEEE*, 77(4):541–580, April 1989.
[11] P. J. Ramadge and W. M. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77(1):81–98, 1989.
[12] A. Ramírez-Treviño, I. Rivera-Rangel, and E. López-Mellado. Observability of discrete event systems modeled by interpreted petri nets. *IEEE Transactions on Robotics and Automation*, 19(4):557–565, 2003.
[13] Y. Ru and C. Hadjicostis. Fault diagnosis in discrete event systems modeled by partially observed Petri nets. *Discrete Event Dynamic Systems*, 19(4):551–575, December 2009.
[14] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Diagnosability of discrete-event systems. *IEEE Transactions on Automatic Control*, 40(9):1555–1575, 1995.
[15] Y. Tong, Z. Li, and A. Giua. General observation structures for Petri nets. In *18th IEEE International Conference on Emerging Technologies and Factory Automation*, Cagliari, Italy, September 2013.
[16] T. Ushio, I. Onishi, and K. Okuda. Fault detection based on Petri net models with faulty behaviors. In *IEEE International Conference on Systems, Man, and Cybernetics*, pages 113–118, San Diego, USA, October 1998.