Verification of Language-Based Opacity in Petri Nets Using Verifier

Yin Tong[1], Ziyue Ma[1], Zhiwu Li[2], Carla Seatzu[3] and Alessandro Giua[4]

Abstract

A system is said to be *language opaque* if the intruder cannot infer if the generated event sequence belongs to a secret based on its partial observation. In this paper we address the problem of verifying language-based opacity in systems modeled by bounded labeled Petri nets. We generalize the notion of language opacity to *strict language opacity* to deal with the case where the intruder is only interested in a subset of transitions. Furthermore, we show that strict language opacity is identical to language opacity for a special class of secrets. A *verifier* is constructed to analyze strict language opacity under the assumption that the intruder only cares about observable transitions.

To appear as:

Y. Tong, Z.Y. Ma, Z.W. Li, C. Seatzu, A. Giua, "Verification of Language-Based Opacity in Petri Nets Using Verifier," the 2016 American Control Conference (Boston, MA, USA), July 6 - July 8, 2016.

---

[1]Yin Tong and Ziyue Ma are with the School of Electro-Mechanical Engineering, Xidian University, Xi'an 710071, China and also with the Department of Electrical and Electronic Engineering, University of Cagliari, 09123 Cagliari, Italy yintong@stu.xidian.edu.cn; maziyue@gamil.com

[2]Zhiwu Li is with the Institute of Systems Engineering, Macau University of Science and Technology, Taipa, Macau and also with the School of Electro-Mechanical Engineering, Xidian University, Xi'an 710071, China zhwli@xidian.edu.cn

[3]Carla Seatzu is with the Department of Electrical and Electronic Engineering, University of Cagliari, 09123 Cagliari, Italy seatzu@diee.unica.it

[4]Alessandro Giua is with Aix Marseille Université, CNRS, ENSAM, Université de Toulon, LSIS UMR 7296, Marseille 13397, France and also with DIEE, University of Cagliari, Cagliari 09124, Italy alessandro.giua@lsis.org; giua@diee.unica.it

I. Introduction

In recent years opacity has drawn a lot of attention from researchers in the discrete event system (DES) community. This property characterizes the ability of a system to hide a "secret" from an external observer (called an intruder). It is assumed that the intruder knows the structure of the system but has only partial observation of the system's evolution. Opacity in DESs was firstly formalized by Bryans *et al*. [1] using Petri nets and then extended to transition systems [2]. In [3] opacity was defined and investigated in the framework of finite automata.

In DESs, two main types of opacity properties have been defined: *state-based opacity* and *language-based opacity*, where the secret is defined as a set of states and a language, respectively. For state-based opacity properties, many researchers have considered their verification [4], [5], [6], [7], [8] and enforcement [6], [9], [10], [11], [12]. In this work, we focus on the verification of language-based opacity in Petri nets. For a thorough and comprehensive review on opacity in DESs, we refer readers to [5], [13].

Language-based opacity was introduced in [14], [15] in the framework of finite state automata. A system is language-based opaque with respect to a given secret if no execution leads to an estimate that is completely inside the secret. Lin [16] defined two language-based opacity properties: strong opacity and weak opacity, and discussed their analysis. A language is strongly (resp. weakly) opaque if all (resp. some) strings in the language are confused with some strings in another language. Polynomial algorithms to check weak opacity of an automaton have been proposed in [17]. On the contrary, providing a polynomial algorithm to check strong opacity is still an open problem.

Petri nets have been proposed as a fundamental model for DES in a wide variety of applications. Many problems such as supervisory control, fault diagnosis, etc. have been solved using Petri net models, providing efficient and well founded approaches [18], [19], [20]. In this work we first particularize the notion of language-based opacity in Petri nets assuming that the intruder cares about the order of all transitions occurrence, and call it *language opacity*. However, there exist many practical cases where the intruder only cares about a subset of transitions. As an example, in a banking environment, the intruder may not care if a customer checks the account but may only be interested in the withdrawal of money. Motivated by this, we propose a generalization of the language opacity property and introduce the notion of *strict language opacity*. In particular, a system is strictly language opaque if the intruder can never establish if the transitions in which he is interested have fired in some given order (as described by the secret).

Since the verification of language-based opacity in Petri nets has never been discussed before, the only way to solve the problem in bounded Petri nets was to construct the reachability graph and apply the approach in [16]. In this work, a finite structure, called *verifier* is proposed to analyze strict language opacity in bounded Petri nets. Such an approach works under the assumption that the intruder is interested in the set of observable transitions and the secret is the set of all firable transition sequences in a bounded labeled Petri net (excluding the empty string). Given the nets modeling the plant and the secret, the verifier synchronizes the plant and the secret with respect to observable transitions. It keeps track of both the sequences belonging and those not belonging to the secret. In particular, thanks to the notion of minimal explanations and basis markings [21], [22], to characterize

such sequences there is no need to enumerate all of them. Therefore, the construction of the reachability graph is avoided. The efficiency and effectiveness of the proposed approach is shown by comparing it with Lin's method [16] in Section V.

The rest of this paper is organized as follows. Some basic concepts of automata and Petri nets are recalled in Section II. Language and strict language opacity are defined in Section III as well as their properties and relationships. In Section IV, the verifier is introduced to analyze strict language opacity. The advantages of the verifier based approach with respect to the previous one are highlighted in Section V. Finally, conclusions are drawn in Section VI, where our directions of research on this topic are pointed out.

II. Preliminaries and Background

In this section we recall the formalism used in the paper and some results on reachability analysis in Petri nets. For more details, we refer to [22], [23], [24].

A. Automata

A *deterministic finite automaton* (DFA) is a 5-tuple $A = (X, E, \delta, x_0, X_F)$, where X is the finite *set of states*, E is the *alphabet*, $\delta : X \times E \to X$ is the (partial) *transition function*, $x_0 \in X$ is the *initial state*, and $X_F \subseteq X$ is the set of *final states*. The transition function can be extended to $\delta^* : X \times E^* \to X$ in the recursive manner: $\delta^*(x, we) = \delta^*(\delta^*(x, w), e)$ for $w \in E^*$ and $e \in E$. If $w = \varepsilon$, $\delta^*(x, \varepsilon) = x$.

The *generated language* of an automaton $A = (X, E, \delta, x_0, X_F)$ is defined as

$$\mathcal{L}(A) = \{w \in E^* | \exists x \in X : \delta^*(x_0, w) = x\}.$$

The *accepted langue* of A is defined as

$$\mathcal{L}_m(A) = \{w \in E^* | \exists x \in X_F : \delta^*(x_0, w) = x\}.$$

B. Petri Nets

A *Petri net* is a structure $N = (P, T, Pre, Post)$, where P is a set of m *places*, graphically represented by circles; T is a set of n *transitions*, graphically 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 transitions, and vice versa. The incidence matrix of a net is denoted by $C = Post - Pre$. A Petri net is said to be *acyclic* if there are no oriented cycles.

A *marking* is a vector $M : P \to \mathbb{N}$ that assigns to each place a non-negative integer number of tokens, graphically represented by black dots. The marking of place p is denoted by $M(p)$. A marking is also denoted as $M = \sum_{p \in P} M(p) \cdot p$. A *Petri net system* $\langle N, M_0 \rangle$ is a net N with *initial marking* M_0.

A transition t is *enabled* at marking M if $M \geq Pre(\cdot, t)$ and may fire yielding a new marking $M' = M + C(\cdot, t)$. We denote $T(M) = \{t \in T | M[t\rangle\}$ the set of transitions enabled at M. 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

firable transition sequences in $\langle N, M_0\rangle$ is denoted as $L(N, M_0) = \{\sigma \in T^* | M_0[\sigma\rangle\}$. Given a sequence $\sigma \in T^*$, the function $\pi : T^* \to \mathbb{N}^n$ associates with σ the Parikh vector $y = \pi(\sigma) \in \mathbb{N}^n$, i.e., $y(t) = k$ if transition t appears k times in σ.

A marking M is *reachable* in $\langle N, M_0\rangle$ if there exists a sequence σ such that $M_0[\sigma\rangle M$. The set of all markings reachable from M_0 defines the *reachability set* of $\langle N, M_0\rangle$, denoted by $R(N, M_0)$. A Petri net system is *bounded* if there exists a non-negative integer $k \in \mathbb{N}$ such that for any place $p \in P$ and any reachable marking $M \in R(N, M_0)$, $M(p) \leq k$ holds.

A *labeled Petri net* (LPN) is a 4-tuple $G = (N, M_0, E, \ell)$, where $\langle N, M_0\rangle$ is a Petri net system, E is the *alphabet* (a set of labels) and $\ell : T \to E \cup \{\varepsilon\}$ is the *labeling function* that assigns to each transition $t \in T$ either a symbol from E or the empty word ε. Therefore, the set of transitions can be partitioned into two disjoint sets $T = T_o \cup T_u$, where $T_o = \{t \in T | \ell(t) \in E\}$ is the set of observable transitions and $T_u = T \setminus T_o = \{t \in T | \ell(t) = \varepsilon\}$ is the set of unobservable transitions. We denote $T_e = \{t \in T | \ell(t) = e\}$ the set of transitions labeled with $e \in E \cup \{\varepsilon\}$ and $T_e(M) = T_e \cap T(M)$ the set of transitions enabled at a given marking M and labeled with e. If there is no unobservable transition and different transitions are labeled with different labels, the LPN is called *free-labeled*. The labeling function can be extended to sequences $\ell : T^* \to E^*$ as $\ell(\sigma t) = \ell(\sigma)\ell(t)$ with $\sigma \in T^*$ and $t \in T$. The *generated language* of G is $\mathcal{L}(G) = \{w \in E^* | \exists \sigma \in L(N, M_0) : w = \ell(\sigma)\}$. Given a set $T' \subseteq T$, the natural *projection* $P_{T'} : T^* \to T'^*$ on T' is defined as i) $P_{T'}(\varepsilon) = \varepsilon$; ii) for all $\sigma \in T^*$ and $t \in T$, $P_{T'}(\sigma t) = P_{T'}(\sigma)t$ if $t \in T'$, and $P_{T'}(\sigma t) = P_{T'}(\sigma)$, otherwise. The inverse projection operator is defined as $P^{-1}_{T'}(\sigma) = \{\sigma' \in T^* | P_{T'}(\sigma') = P_{T'}(\sigma)\}$. Given a set of sequences $B \subseteq T^*$, we denote $P_{T'}(B) = \bigcup_{\sigma \in B}\{P_{T'}(\sigma)\}$ and $P^{-1}_{T'}(B) = \{\sigma \in T^* \mid \exists \sigma' \in B : P_{T'}(\sigma') = P_{T'}(\sigma)\}$.

Given an LPN $G = (N, M_0, E, \ell)$ and the set of unobservable transitions T_u, the *T_u-induced subnet* $N' = (P, T', Pre', Post')$ of N, is the net resulting by removing all transitions in $T \setminus T_u$ from N, where Pre' and $Post'$ are the restriction of Pre, $Post$ to T_u, respectively. The incidence matrix of the T_u-induced subnet is denoted by $C_u = Post' - Pre'$.

C. Some Results on Reachability in Petri Nets

In this subsection we recall some results on computing reachable markings proposed in [22].

Definition 2.1: Given a marking M and an observable transition $t \in T_o$, we define

$$\Sigma(M, t) = \{\sigma \in T_u^* | M[\sigma\rangle M', M' \geq Pre(\cdot, t)\}$$

as the set of *explanations* of t at M and $Y(M, t) = \{y_u \in \mathbb{N}^{n_u} | \exists \sigma \in \Sigma(M, t) : y_u = \pi(\sigma)\}$ the set of *e-vectors*. $\diamond$

Thus $\Sigma(M, t)$ is the set of unobservable sequences whose firing at M enables t. Among all the explanations, to provide a compact representation of the reachability set we are interested in finding the minimal ones, i.e., the ones whose firing vector is minimal.

Definition 2.2: Given a marking M and an observable transition $t \in T_o$, we define

$$\Sigma_{min}(M, t) = \{\sigma \in \Sigma(M, t) | \nexists \sigma' \in \Sigma(M, t) : \pi(\sigma') \lneqq \pi(\sigma)\}$$

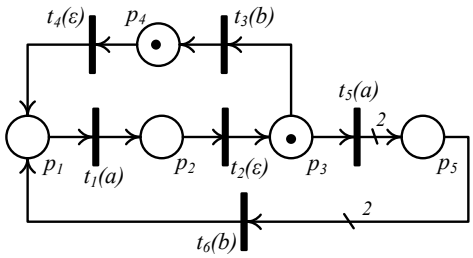

Fig. 1. An LPN system whose T_u-induced net is acyclic.

as the set of *minimal explanations* of t at M and $Y_{min}(M,t) = \{y_u \in \mathbb{N}^{n_u} | \exists \sigma \in \Sigma_{min}(M,t) : y_u = \pi(\sigma)\}$ as the corresponding set of *minimal e-vectors*. ◇

Many approaches can be applied to computing $Y_{min}(M,t)$. In particular, when the T_u-induced subnet is acyclic the approach proposed by Cabasino *et al*. [25] only requires algebraic manipulations. Note that since a given place may have two or more unobservable input transitions, i.e., the T_u-induced subnet is not backward conflict free, the set of minimal explanations is not necessarily a singleton.

Proposition 2.3: [22] Let $G = (N, M_0, E, \ell)$ be an LPN whose T_u-induced subnet is acyclic. There exists a firing sequence $\sigma = \sigma_{u1} t_{i1} \cdots \sigma_{uk} t_{ik}$ such that $M_0[\sigma_{u1}\rangle M_1[t_{i1}\rangle M_1' \cdots M_{k-1}[\sigma_{uk}\rangle M_k[t_{ik}\rangle$, where $\sigma_{uj} \in T_u^*$ and $t_{ij} \in T_o$, if and only if there exists $\sigma' = \sigma'_{u1} t_{i1} \cdots \sigma'_{uk} t_{ik}$ such that $M_0[\sigma'_{u1}\rangle \hat{M}_1[t_{i1}\rangle \hat{M}_1' \cdots \hat{M}_{k-1}[\sigma'_{uk}\rangle \hat{M}_k[t_{ik}\rangle$, where $\sigma'_{uj} \in \Sigma_{min}(\hat{M}_j, t_{ij})$.

In simple words, a sequence σ whose projection on T_o is $t_{i1} t_{i2} \cdots t_{ik}$ is firable if and only if there exists σ' such that the minimal explanation of each observable transition is not empty. Namely, to check if $\exists \sigma \in L(N, M_0) : P_{T_o}(\sigma) = t_{i1} t_{i2} \cdots t_{ik}$ there is no need to enumerate all $\sigma_u \in T_u^*$ that enable $t_{i1}, t_{i2}, \cdots, t_{ik}$ but only the minimal explanations. Markings $\hat{M}_j'$ are called *basis markings*, and the set of basis markings of G is denoted as $\mathcal{M}_B(G)$. Clearly, $\mathcal{M}_B(G) \subseteq R(N, M_0)$ and in practical cases the number of basis markings is much smaller than the number of reachable markings [7], [26].

Example 2.4: Consider the LPN system in Fig. 1. At the initial marking, the minimal explanation of t_1 is $\Sigma_{min}(M_0, t_1) = \{t_4\}$. At marking M with $M_0[t_4 t_1\rangle M$, transition t_3 is enabled, therefore $\Sigma_{min}(M, t_3) = \{\vec{0}\}$. Therefore, it holds $M_0[t_4 t_1\rangle M[t_3\rangle$. It can be computed that there are seven firing sequences: $t_4 t_1 t_3$, $t_4 t_1 t_3 t_2$, $t_4 t_1 t_2 t_3$, $t_4 t_1 t_3 t_4$, $t_4 t_1 t_3 t_2 t_4$, $t_4 t_1 t_2 t_3 t_4$, and $t_4 t_1 t_3 t_4 t_2$, whose projection on T_o is $t_1 t_3$. Namely, to prove the existence of strings whose projection is $t_1 t_3$ there is no need to exhaustively enumerate them but we simply need to iteratively compute the minimal explanations. ◇

III. Language-Based Opacity and Problem Formulation

The notion of language opacity has originally been defined in automata [14], [15]. In these works a set of event sequences is defined as the *secret*, and an automaton is opaque wrt the secret if the intruder cannot establish if some secret sequences have occurred based on its observation. This notion can be naturally generalized to Petri

nets as follows.

Definition 3.1: Let $G = (N, M_0, E, \ell)$ be an LPN and $S \subseteq T^*$ be a secret. G is called *language opaque* wrt S if $\forall \sigma \in S \cap L(N, M_0)$, there exists $\sigma' \in L(N, M_0) \setminus S$ such that $\ell(\sigma) = \ell(\sigma')$. $\diamond$

By Definition 3.1, a plant is language opaque wrt a given secret if for any word that can be generated by a sequence in the secret, there exists another nonsecret sequence generating the same word. This means that an intruder who knows the system G observes the word $w = \ell(\sigma)$ but cannot be certain that a transition sequence belonging to S has been generated. Clearly, if $L(N, M_0) \cap S = \emptyset$, the system is language opaque.

Let us now generalize the notion of language opacity to deal with the case where the intruder is only interested in a subset of transitions.

Definition 3.2: Given an LPN $G = (N, M_0, E, \ell)$, a set of transitions $\hat{T} \subseteq T$, and a set of sequences $\hat{S} \subseteq \hat{T}^*$. G is said to be *strictly language opaque* wrt to $(\hat{T}, \hat{S})$ if $\forall \sigma \in L(N, M_0)$ such that $P_{\hat{T}}(\sigma) \in \hat{S}$, there exists $\sigma' \in L(N, M_0)$ such that $\ell(\sigma') = \ell(\sigma)$ and $P_{\hat{T}}(\sigma') \notin \hat{S}$. $\diamond$

In words, a system is strictly language opaque wrt $(\hat{T}, \hat{S})$ if for any observation that can be explained with a sequence whose projection on $\hat{T}$ belongs to the secret, there exists another explanation whose projection on $\hat{T}$ does not belong to the secret. Obviously, if $\hat{T} = T$, strict language opacity is identical to language opacity. Next we further discuss some properties of strict language opacity.

Proposition 3.3: Let $G = (N, M_0, E, \ell)$ be an LPN, $\hat{T}' \subseteq \hat{T} \subseteq T$, and $\hat{S} \subseteq \hat{T}^*$ be a secret. If G is strictly language opaque wrt $(\hat{T}', \hat{S}')$, where $\hat{S}' = P_{\hat{T}'}(\hat{S})$, then G is strictly language opaque wrt $(\hat{T}, \hat{S})$.

Proof: This is proved by showing that if G is not strictly language opaque wrt $(\hat{T}, \hat{S})$ then G is not strictly language opaque wrt $(\hat{T}', \hat{S}')$. Given a transition sequence $\sigma \in L(N, M_0)$, the set of transition sequences having the same observation with σ is denoted as $\Sigma = \{\sigma' \in L(N, M_0) | \ell(\sigma') = \ell(\sigma)\}$. Since G is not strictly language opaque wrt $(\hat{T}, \hat{S})$, there exists $\sigma \in L(N, M_0)$ such that $P_{\hat{T}}(\sigma) \in \hat{S}$ and $\forall \sigma' \in \Sigma$, $P_{\hat{T}}(\sigma') \in \hat{S}$ holds. While projecting σ and σ' on $\hat{T}'$, it holds $P_{\hat{T}'}(P_{\hat{T}}(\sigma)) \in \hat{S}'$ and $P_{\hat{T}'}(P_{\hat{T}}(\sigma')) \in \hat{S}'$. Since $\hat{T}' \subseteq \hat{T}$, $P_{\hat{T}'}(P_{\hat{T}}(\sigma)) = P_{\hat{T}'}(\sigma)$ and $P_{\hat{T}'}(P_{\hat{T}}(\sigma')) = P_{\hat{T}'}(\sigma')$ hold. Thus, for $\sigma \in L(N, M_0)$ such that $P_{\hat{T}'}(\sigma) \in \hat{S}'$, $P_{\hat{T}'}(\sigma') \in \hat{S}'$ holds for all $\sigma' \in \Sigma$, i.e., G is not strictly language opaque wrt $(\hat{T}', \hat{S}')$. ■

Proposition 3.3 provides a semi-decision procedure (only sufficient) to verify strict language opacity wrt $(\hat{T}, \hat{S})$. More precisely, one can choose a subset $\hat{T}'$ of $\hat{T}$ and verify strict language opacity wrt $(\hat{T}', \hat{S}')$. Note that as aforementioned, if $\hat{T} = T$, strict language opacity is identical to language opacity. Therefore, language opacity wrt a given secret $S \subseteq T^*$ can be sufficiently decided by verifying the strict language opacity property wrt a set $\hat{T} \subseteq T$ and $P_{\hat{T}}(S)$. In the following example, it is shown that the converse of Proposition 3.3 may not hold. Namely, it may occur that even if a system is strictly language opaque wrt $(\hat{T}, \hat{S})$, it may not be strictly language opaque wrt $(\hat{T}', \hat{S}')$.

Example 3.4: Consider again the LPN in Fig. 1. Let $\hat{T} = T$ and $\hat{S} = \{t_4 t_1, t_5\}$. The net system is strictly language opaque wrt $(\hat{T}, \hat{S})$ (equivalently, G is language opaque wrt $\hat{S}$), since there exists $\sigma = t_4 t_1 t_2$ such that $P_{\hat{T}}(\sigma) \notin \hat{S}$ and $\ell(t_4 t_1 t_2) = \ell(t_4 t_1) = \ell(t_5) = a$. Let $\hat{T}' = \{t_1, t_5\} \subset \hat{T}$. The system is not strictly language opaque wrt $(\hat{T}', \hat{S}')$, where $\hat{S}' = P_{\hat{T}'}(\hat{S}) = \{t_1, t_5\}$, since there does not exist a sequence generating a but whose

projection on $\hat{T}'$ is not in $\hat{S}'$. $\diamond$

In the supervisory control, the *normality* [27] property of a language is introduced. Herein this notion is slightly extended as follows.

Definition 3.5: Let G be an LPN and $\hat{T} \subseteq T$ be a subset of transitions. A language $S \subseteq T^*$ is said to be normal wrt $(L(N, M_0), P_{\hat{T}})$ if

$$L(N, M_0) \cap S = L(N, M_0) \cap P_{\hat{T}}^{-1}(P_{\hat{T}}(S)).$$

$\diamond$

A language S is normal wrt a given language $L(N, M_0)$ and the natural projection $P_{\hat{T}}$ if its intersection with $L(N, M_0)$ is the largest sublanguage of $L(N, M_0)$ whose projection is $P_{\hat{T}}(S)$.

Proposition 3.6: Let $G = (N, M_0, E, \ell)$ be an LPN, $\hat{T} \subseteq T$ be a subset of T, $S \subseteq T^*$ be a secret. G is strictly language opaque wrt $(\hat{T}, \hat{S})$, where $\hat{S} = P_{\hat{T}}(S)$, if and only if G is language opaque wrt $P_{\hat{T}}^{-1}(P_{\hat{T}}(S))$.

Proof: ($\Rightarrow$) Follows from Proposition 3.3.

($\Leftarrow$) Given a transition sequence $\sigma \in L(N, M_0)$, the set of transition sequences having the same observation with σ is denoted as $\Sigma = \{\sigma' \in L(N, M_0) | \ell(\sigma') = \ell(\sigma)\}$. Assume that G is not strictly language opaque wrt $(\hat{T}, \hat{S})$, therefore there exists a sequence $\sigma \in L(N, M_0)$ such that $P_{\hat{T}}(\sigma) \in \hat{S}$ and $\forall \sigma' \in \Sigma$, it holds $P_{\hat{T}}(\sigma') \in \hat{S}$. Since $\hat{S} = P_{\hat{T}}(S)$, $\sigma \in L(N, M_0) \cap P_{\hat{T}}^{-1}(P_{\hat{T}}(S))$ and $\sigma' \in L(N, M_0) \cap P_{\hat{T}}^{-1}(P_{\hat{T}}(S))$ hold. Thus, G is not language opaque wrt $P_{\hat{T}}^{-1}(P_{\hat{T}}(S))$. ■

Proposition 3.6 shows that strict language opacity wrt $(\hat{T}, \hat{S})$ is identical to language opacity wrt $P_{\hat{T}}^{-1}(P_{\hat{T}}(S))$.

Corollary 3.7: Let $G = (N, M_0, E, \ell)$ be an LPN and $\hat{T} \subseteq T$. Let $S \subseteq T^*$ be a secret that is normal wrt $(L(N, M_0), P_{\hat{T}})$ and $\hat{S} = P_{\hat{T}}(S)$. G is strict language opaque wrt $(\hat{T}, \hat{S})$ if and only if G is language opaque wrt S.

Proof: According to Definition 3.1, given a secret $S \subseteq T^*$, G is language opaque wrt S is identical to G is language opaque wrt $S \cap L(N, M_0)$. By Proposition 3.6, G is strictly language opaque wrt $(\hat{T}, \hat{S})$ if and only if G is language opaque wrt $L(N, M_0) \cap P_{\hat{T}}^{-1}(P_{\hat{T}}(S))$. Since S is normal, i.e., $L(N, M_0) \cap S = L(N, M_0) \cap P_{\hat{T}}^{-1}(P_{\hat{T}}(S))$, G is strictly language opaque wrt $(\hat{T}, \hat{S})$ if and only if G is language opaque wrt S. ■

In general, language opacity wrt $S \subseteq T^*$ (that is identical to strict language opacity wrt (T, S)) does not imply strict language opacity wrt $(\hat{T}, P_{\hat{T}}(S))$ if $\hat{T} \subseteq T$ (see Proposition 3.3). However, by Corollary 3.7, such an implication holds if S is normal wrt $(L(N, M_0), P_{\hat{T}})$. In other words, given an arbitrary set $\hat{S} \subseteq \hat{T}^*$, verifying strict language opacity wrt $(\hat{S}, \hat{T})$ can be reduced to verifying language opacity wrt $P_{\hat{T}}^{-1}(\hat{S})$. Thus, the complexity of verifying strict language opacity would not be better than that of verifying language opacity. Furthermore, in the remaining sections of the paper we show that under proper assumptions, verification of strict language opacity is of lower complexity.

Corollary 3.7 also implies that if a secret $S \subseteq T^*$ has the normality property, i.e., there exists a subset $\hat{T} \subseteq T$ such that S is normal wrt $(L(N, M_0), \hat{T})$, language opacity wrt S can be verified by strict language opacity wrt $(P_{\hat{T}}(S), P_{\hat{T}})$. However, given a secret $S \subseteq T^*$, finding a set $\hat{T}$ that guarantees its normality is still an open problem. Therefore, the remaining of the paper is focused on verifying strict language opacity.

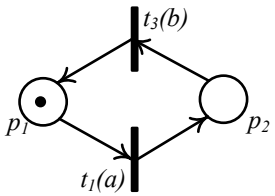

Fig. 2. An LPN that models the secret.

At the end of this section we formalize the problem addressed in the rest of the work. We first introduce the following assumptions:

A1) The T_u-induced subnet of G is acyclic.

A2) The set of transitions of which the intruder cares is the set of observable transitions, i.e., $\hat{T} = T_o$.

A3) The secret is the set of all firable transition sequences in a bounded labeled Petri net system whose set of transitions is T_o, but excluding ε.

These assumptions would bring some properties on which we could build our algorithm for verifying strict language opacity. In practice, the computational load can be greatly reduced as shown in Section IV, since by Assumptions A1 and A2 the order of the unobservable transitions can be abstracted using minimal explanations and basis markings. Note that there is no assumption on the labeling function of the system. Namely, the same label (including ε) can be assigned to different transitions. Assumption A3 indicates that the secret can be described by the generated language of a bounded free-labeled Petri net but excluding the empty word. If ε is in the secret, the system is not strictly language opaque since initially it is at the state in which no observable transition has fired and the intruder can conclude that without ambiguity. It is true that only a special class of secrets satisfy Assumption A3 and thus the application of the proposed approach is not general. However, Assumption 3 leads to a computationally efficient verification procedure. Moreover, in practice we believe that these assumptions, even if quite restrictive, allow to represent some real problems.

We denote $G^S = (N^S, M_0^S, T^S, \ell^S)$, where $T^S \subseteq T_o$, the labeled Petri net describing the secret, i.e., $S = L(N, M_0) \setminus \{\varepsilon\}$. The labeling function of G^S is identical to that of G, i.e., given a transition $t \in T^S$, $\ell^S(t) = \ell(t)$. The problem is stated as follows.

Problem Statement: Given a bounded LPN G and a Petri net G^S describing the secret $S_o \in T_o^*$ satisfying Assumptions A1 to A3, determine whether G is strictly language opaque wrt (T_o, S_o) or not.

In the next section we propose a method based on a structure called *verifier* to solve the above problem.

IV. Verification of Strict Language Opacity

In this section we introduce a structure, called *verifier*, to efficiently verify strict language opacity.

If the plant G and the net G^S describing the secret are bounded, the verifier is a DFA, denoted as $\mathcal{V} = (X, T^S, \delta, x_0)$. A state $x \in X$ of $\mathcal{V}$ is a 3-tuple $(M^S, \mathcal{M}^G, \mathcal{M})$, where M^S is a marking of G^S, $\mathcal{M}^G$ and

$\mathcal{M}$ are subsets of basis markings in G, i.e., $M^S \in R(N^S, M_0^S)$, $\mathcal{M}^G \subseteq \mathcal{M}_B(G)$ and $\mathcal{M} \subseteq \mathcal{M}_B(G)$. The i-th element (for $i = 1, 2, 3$) of x is denoted as $x(i)$. The set of events is the set of transitions of G^S. The initial state of the verifier is $x_0 = (M_0^S, \{M_0\}, \emptyset)$. Algorithm 1 illustrates the construction of the verifier.

Given a state x in the verifier such that $\delta^*(x_0, \sigma) = x$, $x(1)$ is the marking in G^S reachable from M_0^S by firing σ, i.e., $M_0^S[\sigma\rangle x(1)$; $x(2)$ is the set of basis markings in G that are reachable by firing a transition sequence σ' that has σ as its projection on T_o, i.e., $P_{T_o}(\sigma') = \sigma$; finally $x(3)$ is the set of basis markings in G that are reachable by firing a transition sequence σ'' whose projection on T_o is different from σ but generates the same observation, i.e., $P_{T_o}(\sigma'') \neq \sigma$ and $\ell(\sigma'') = \ell(\sigma)$. More precisely, given two states $x_1 = (M_1^S, \mathcal{M}_1^G, \mathcal{M}_1)$, $x_2 = (M_2^S, \mathcal{M}_2^G, \mathcal{M}_2)$ and an event $t \in T^S$ of $\mathcal{V}$, $\delta^*(x_1, t) = x_2$ implies that M_2^S is the marking reachable from M_1^S by firing t in G^S (Step 6), $\mathcal{M}_2^G$ is the set of basis markings reachable from markings in $\mathcal{M}_1^G$ by firing t and the corresponding minimal explanations in G (Steps 7 to 12), and $\mathcal{M}_2$ is the union of two sets: the set of markings reachable from markings in $\mathcal{M}_1^G$ by firing transitions in $t' \in T_e \setminus T_e^S(x(1))$ and the corresponding minimal explanations (Steps 17 to 24) in G, and the set of markings reachable from $\mathcal{M}_1$ by firing transitions in T_e and the corresponding minimal explanations (Steps 25 to 32) in G, where $e = \ell(t)$ and $T_e^S(x(1))$ is the set of transitions labeled with e and enabled at marking $x(1)$ in G^S. If such a transition t is never enabled in G, then a new node would not be created (Steps 13 to 15).

The main idea behind Algorithm 1 is to compute a sort of parallel composition between G^S and G, where synchronization is performed wrt T_o. As a result, the generated language of $\mathcal{V}$ is equal to $P_{T_o}(L(N, M_0)) \cap L(N^S, M_0^S)$. Moreover, this enables one to understand if to a transition sequence in the secret it corresponds a transition sequence in the system such that its projection on T_o is not in the secret, and it generates the same observation. The properties of the verifier are formally presented as follows.

Proposition 4.1: Let $G = (N, M_0, E, \ell)$ be an LPN satisfying Assumption A1, and $G^S = (N^S, M_0^S, T^S, \ell^S)$ be the LPN describing the secret and satisfying Assumptions A2 and A3. Let $\mathcal{V} = (X, T^S, \delta, x_0)$ be the verifier constructed by using Algorithm 1. Given a state $x \in X$ and $\sigma \in \mathcal{L}(\mathcal{V})$ such that $\delta^*(x_0, \sigma) = x$, the following implication holds:

$$x(2) \neq \emptyset \Leftrightarrow \exists \sigma' \in L(N, M_0) : P_{T_o}(\sigma') = \sigma.$$

Proof: Let $\delta^*(x_0, \sigma) = x$ and $\sigma = t_{i1}t_{i2}\cdots t_{ik} \in (T^S)^*$. According to Algorithm 1, $x(2)$ is the set of markings reachable from M_0 by firing the sequence $\hat{\sigma} = \sigma_{u1}t_{i1}\sigma_{u2}t_{i2}\cdots\sigma_{uk}t_{ik}$ in G, where σ_{uj} is a minimal explanation of t_{ij}. Therefore, $P_{T_o}(\hat{\sigma}) = \sigma$. By Proposition 2.3, a sequence $\sigma' = \sigma'_{u1}t_{i1}\sigma'_{u2}t_{i2}\cdots\sigma'_{uk}t_{ik}$ such that $M_0[\sigma'\rangle$ and $P_{T_o}(\sigma') = P_{T_o}(\hat{\sigma})$ exists (where $\sigma'_{ui} \in T_u^*$) if and only if $x(2) \neq \emptyset$. ■

Proposition 4.2: Let $G = (N, M_0, E, \ell)$ be an LPN satisfying Assumption A1, and $G^S = (N^S, M_0^S, T^S, \ell^S)$ be the LPN describing the secret S_o and satisfying Assumptions A2 and A3. Let $\mathcal{V} = (X, T^S, \delta, x_0)$ be the verifier constructed by using Algorithm 1. Given a state $x \in X \setminus \{x_0\}$ and $\sigma \in \mathcal{L}(\mathcal{V})$ such that $\delta^*(x_0, \sigma) = x$, the following implication holds:

$$x(3) \neq \emptyset \Leftrightarrow \exists \sigma' \in L(N, M_0) : P_{T_o}(\sigma') \notin S_o \wedge \ell(\sigma') = \ell(\sigma).$$

Proof: This is proven by induction.

Algorithm 1 Construction of the verifier

Input: A bounded labeled Petri net $G = (N, M_0, E, \ell)$ and a bounded LPN $G^S = (N^S, M_0^S, E, \ell)$.

Output: Verifier $\mathcal{V} = (X, T^S, \delta, x_0)$

$x_0 := (M_0^S, \{M_0\}, \emptyset)$;
$X := \{x_0\}$;
for all $x \in X$ with no tag, **do**
  for all $t \in T^S(x(1))$, **do**
    $\mathcal{M}^G := \emptyset$, $\mathcal{M} := \emptyset$;
    $M^S := x(1) + C^S(\cdot, t)$;
    for all $M \in x(2)$ **do**
      for all $y_u \in Y_{min}(M, t)$ **do**
        $M' := M + C_u \cdot y_u + C(\cdot, t)$;
        $\mathcal{M}^G := \mathcal{M}^G \cup \{M'\}$;
      end for
    end for
    if $\mathcal{M}^G = \emptyset$, **then**
      Break;
    end if
    $e := \ell(t)$;
    for all $M \in x(2)$, **do**
      for all $t' \in T_e \setminus T_e^S(x(1))$, **do**
        for all $y_u \in Y_{min}(M, t')$, **do**
          $M' := M + C_u \cdot y_u + C(\cdot, t')$
          $\mathcal{M} := \mathcal{M} \cup \{M'\}$;
        end for
      end for
    end for
    for all $M \in x(3)$, **do**
      for all $t' \in T_e$, **do**
        for all $y_u \in Y_{min}(M, t')$, **do**
          $M' := M + C_u \cdot y_u + C(\cdot, t')$;
          $\mathcal{M} := \mathcal{M} \cup \{M'\}$;
        end for
      end for
    end for
    if $(M^S, \mathcal{M}^G, \mathcal{M}) \notin X$, **then**
      $X := X \cup \{(M^S, \mathcal{M}^G, \mathcal{M})\}$;
      $\delta(x, t) := (x, t, (M^S, \mathcal{M}^G, \mathcal{M}))$;
    end if
  end for
  tag x "checked";

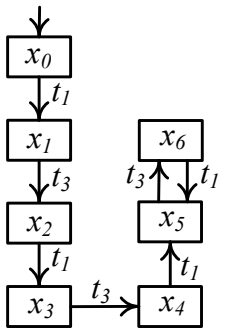

Fig. 3. The verifier constructed in Example 4.4.

(Basis step) For $\sigma \in \mathcal{L}(\mathcal{V})$ with length 1. Let $\delta^*(x_0, t) = x$ and $e = \ell(t)$. Since $x_0(3) = \emptyset$, if and only if $x(3) \neq \emptyset$, there exists a sequence $\sigma_u t'$ such that $M_0[\sigma_u t'\rangle M \in x(3)$, where $t' \in T_e \setminus T_e^S(x(2))$, and $\sigma_u \in \Sigma_{min}(M_0, t')$. Namely, $P_{T_o}(\sigma_u t') \notin S_o$ and $\ell(t') = \ell(t)$.

(Inductive step) Assume that for $\sigma_k \in \mathcal{L}(\mathcal{V})$ with length k, the result is valid. We prove that it is also true for $\sigma_{k+1} \in \mathcal{L}(\mathcal{V})$ with length $k+1$. Let $\sigma_{k+1} = \sigma_k t$ and $e = \ell(t)$.

Consider $\delta^*(x_0, \sigma_k) = x_1$ and $\delta^*(x_1, t) = x_2$. According to Algorithm 1 and Proposition 2.3, $x_2(3) \neq \emptyset$ if and only if one of the following conditions holds:

1) there exists $M \in x_1(2)$ such that $M[\sigma_u t'\rangle$, where $t' \in T_e \setminus T_e^S(x_1(1))$ and $\sigma_u \in \Sigma_{min}(M, t')$;
2) there exists $M \in x_1(3)$ such that $M[\sigma_u t'\rangle$, where $t' \in T_e$ and $\sigma_u \in \Sigma_{min}(M, t')$.

Assume condition 1) holds. Since $x_1(2) \neq \emptyset$, by Proposition 4.1, there exists $\sigma \in L(N, M_0)$ such that $P_{T_o}(\sigma) = \sigma_k$ and $M_0[\sigma\rangle M$. Since $t' \in T_e \setminus T_e^S(x_1(1))$, $P_{T_o}(\sigma\sigma_u t') \notin S_o$ but $\ell(\sigma\sigma_u t') = \ell(\sigma_{k+1})$.

Assume condition 2) holds. Since $x_1(3) \neq \emptyset$, there exists $\sigma \in L(N, M_0)$ such that $P_{T_o}(\sigma) \notin S_o$ and $M_0[\sigma\rangle M$. Therefore, $P_{T_o}(\sigma\sigma_u t') \notin S_o$ but $\ell(\sigma\sigma_u t') = \ell(\sigma_{k+1})$. Thus, this concludes the proof. ■

Theorem 4.3: Let $G = (N, M_0, E, \ell)$ be an LPN satisfying Assumption A1, and $G^S = (N^S, M_0^S, T^S, \ell^S)$ be the LPN describing the secret S_o and satisfying Assumptions A2 and A3. Let $\mathcal{V} = (X, T^S, \delta, x_0)$ be the verifier constructed by using Algorithm 1. G is strictly language opaque wrt (T_o, S_o), if and only if $\forall x \in X \setminus \{x_0\}$, $x(3) \neq \emptyset$ holds.

Proof: Follows from Propositions 4.1 and 4.2. ■

Example 4.4: Consider the LPN in Fig. 1 modeling the plant, and the LPN in Fig. 2 describing the secret $S_o = \{t_1, (t_1t_3)^n, (t_1t_3)^n t_1\}$ with $n = 1, 2, 3, \ldots$. By Algorithm 1, the verifier is constructed in Fig. 3. Table I presents the state space of the verifier. By Theorem 4.3, since in no state of the verifier the third entry is empty, the LPN is strictly language opaque wrt (T_o, S_o). ◇

V. Computational Complexity Analysis

In this section we compare the computational complexity (with respect to the number of states) of the proposed approach with a previous approach in the literature.

TABLE I

STATES OF THE VERIFIER

X	$(M^S, \mathcal{M}^G, \mathcal{M})$
x_0	$(p_1, \{p_3 + p_4\}, \emptyset)$
x_1	$(p_2, \{p_3 + p_2\}, \{2p_5 + p_4\})$
x_2	$(p_1, \{p_4 + p_2\}, \{p_1 + p_4\})$
x_3	$(p_2, \{2p_2\}, \{p_4 + 2p_5, p_2 + p_4\})$
x_4	$(p_1, \{p_2 + p_4\}, \{p_4 + p_1, 2p_4\})$
x_5	$(p_2, \{2p_2\}, \{p_4 + p_2\})$
x_6	$(p_1, \{p_2 + p_4\}, \{2p_4\})$

Let n_s, n_b and n_r be the number of reachable markings of G^S, basis markings of G and reachable markings of G, respectively. The number of states of the verifier $\mathcal{V}$ constructed by Algorithm 1 is bounded by

$$\mathcal{X}_V = n_s \times 2^{n_b} \times 2^{n_b}. \tag{1}$$

The notion of basis markings enables one to avoid enumerating all transition sequences whose projection on T_o does not belong or belongs to the secret. Since $n_b \leq n_r$ and in many cases n_b is much smaller than n_r as shown in [7], [26], using basis markings provides significant advantages.

Let us compare the proposed approach with other methods in the literature. As pointed out in Section I, there is no method for language-based opacity analysis in Petri nets. However, some approaches in the automata framework can be used. More precisely, based on Proposition 3.6, given an arbitrary secret, the problem of verifying strict language opacity wrt $(\hat{T}, \hat{S})$ can be reduced to verifying language opacity wrt $P_{\hat{T}}^{-1}(\hat{S})$. Therefore, the method of verifying language opacity in [16] can also be used to verify strict language opacity in Petri nets. Note that an LPN G is said to be language opaque wrt a secret S is equivalent to saying $L(N, M_0)$ is language opaque wrt S in the automata formalism. In such a case, the reachability graph of the net should be constructed and used as the automaton model. Given two automata A and A_S whose number of states are n_1 and n_2, respectively, let B be the automaton constructed to verify if $\mathcal{L}_m(A)$ is language opaque wrt $\mathcal{L}_m(A_S)$. The number of states of B is

$$\mathcal{X}_B = 2^{n_1 \times n_2} \times 2^{n_1}. \tag{2}$$

Let A and A_S be the reachability graphs of G and G^S, respectively, $\mathcal{L}_m(A) = L(N, M_0)$, and $\mathcal{L}_m(A_S) = L(N^S, M_0^S) \setminus \{\varepsilon\} = S_o$. To construct the automaton whose accepted language is $P_{T_o}^{-1}(S_o)$, one just needs to add self loops of transitions in $T \setminus T_o$ on each state. We still denote A_S the obtained automaton to avoid introducing further notation. Namely, verifying if G is strictly language opaque wrt (T_o, S_o) is reduced to verifying if $\mathcal{L}_m(A)$ is language opaque wrt $\mathcal{L}_m(A_S)$. By eq. (2), the number of states of automaton B is

$$\mathcal{X}_B = 2^{n_r \times n_s} \times 2^{n_r}. \tag{3}$$

Compared with eq. (1), the proposed approach is shown to be more efficient. We also point out that in practice for large-sized bounded Petri nets reachability sets may not be computable even if they are finite [7], [26] due to the

state explosion problem. However, to construct the verifier there is no need to compute all reachable markings but a subset of basis markings.

Moreover, as pointed out in the discussion about Proposition 3.3, language opacity can be semi-decided by strict language opacity. Therefore, compared with the automaton-based approach, using the proposed approach to verify language opacity is more efficient. Note that given an arbitrary secret $S \subseteq T^*$, its projection on T_o may not be a prefix-closed bounded free-labeled Petri net language, i.e., Assumption A3 may not be satisfied. Therefore, the proposed approach cannot be used. However, for bounded LPNs the previous method provides a necessary and sufficient condition of language opacity wrt a given secret which is a regular language.

VI. Conclusions and Future Work

In this paper, a new language-based opacity property is introduced. More precisely, strict language opacity is defined. A novel approach to verifying strict language opacity of bounded Petri nets is developed under the assumption that the set of transitions in which the intruder is interested is identical to the set of observable transitions and the secret is a prefix-closed bounded free-labeled Petri net language. For Petri nets whose unobservable subnet is acyclic, the strict language opacity property can be decided by just constructing an appropriate automaton, called verifier. Since a complete enumeration of possible firing sequences is avoided by using minimal explanations, the proposed approach is of lower complexity than the previous approaches.

Our future research is to extend such results to strict language opacity where the intruder cares of an arbitrary set of transitions and characterize a class of language whose projection on a given set is prefix-closed.

Acknowledgement

This work was supported by the National Natural Science Foundation of China under Grant Nos. 61374068, 61472295, the Recruitment Program of Global Experts, the Science and Technology Development Fund, MSAR, under Grant No. 066/2013/A2.

References

[1] J.W. Bryans, M. Koutny, and P.Y. Ryan. Modelling opacity using Petri nets. *Electronic Notes in Theoretical Computer Science*, 121:101–115, 2005.

[2] J.W. Bryans, M. Koutny, L. Mazaré, and P.Y. Ryan. Opacity generalised to transition systems. *International Journal of Information Security*, 7(6):421–435, 2008.

[3] A. Saboori and C.N. Hadjicostis. Notions of security and opacity in discrete event systems. In *46th IEEE Conference on Decision and Control*, pages 5056–5061. IEEE, 2007.

[4] A. Saboori and C.N. Hadjicostis. Verification of initial-state opacity in security applications of discrete event systems. *Information Sciences*, 246:115–132, 2013.

[5] Y.C. Wu and S. Lafortune. Comparative analysis of related notions of opacity in centralized and coordinated architectures. *Discrete Event Dynamic Systems*, 23(3):307–339, 2013.

[6] F. Cassez, J. Dubreil, and H. Marchand. Synthesis of opaque systems with static and dynamic masks. *Formal Methods in System Design*, 40(1):88–115, 2012.

[7] Y. Tong, Z. Li, C. Seatzu, and A. Giua. Verification of current-state opacity using Petri nets. In *2015 American Control Conference*, pages 1935–1940, July 2015.

[8] Y. Tong, Z.W. Li, C. Seatzu, and A. Giua. Verification of initial-state opacity in Petri nets. In *2015 54th IEEE Conference on Decision and Control*, pages 344–349, Dec 2015.

[9] Y.C. Wu and S. Lafortune. Synthesis of insertion functions for enforcement of opacity security properties. *Automatica*, 50(5):1336–1348, 2014.

[10] J. Dubreil, P. Darondeau, and H. Marchand. Supervisory control for opacity. *IEEE Transactions on Automatic Control*, 55(5):1089–1100, 2010.

[11] A. Saboori and C. N. Hadjicostis. Opacity-enforcing supervisory strategies via state estimator constructions. *Automatic Control, IEEE Transactions on*, 57(5):1155–1165, 2012.

[12] X. Yin and S. Lafortune. A new approach for synthesizing opacity-enforcing supervisors for partially-observed discrete-event systems. In *American Control Conference (ACC), 2015*, pages 377–383, July 2015.

[13] R. Jacob, J. Lesage, and J. Faure. Opacity of discrete event systems: models, validation and quantification. In *5th International Workshop on Dependable Control of Discrete Systems*, 2015.

[14] E. Badouel, M. Bednarczyk, A. Borzyszkowski, B. Caillaud, and P. Darondeau. Concurrent secrets. *Discrete Event Dynamic Systems*, 17(4):425–446, 2007.

[15] J. Dubreil, P. Darondeau, and H. Marchand. Opacity enforcing control synthesis. In *9th International Workshop on Discrete Event Systems*, pages 28–35. IEEE, 2008.

[16] F. Lin. Opacity of discrete event systems and its applications. *Automatica*, 47(3):496–503, 2011.

[17] B. Zhang, S.L. Shu, and F. Lin. Polynomial algorithms to check opacity in discrete event systems. In *24th Chinese Control and Decision Conference*, pages 763–769. IEEE, 2012.

[18] Z. Ma, Z.W. Li, and A. Giua. Design of optimal petri net controllers for disjunctive generalized mutual exclusion constraints. *IEEE Transactions on Automatic Control*, 60(7):1774–1785, July 2015.

[19] Y.F. Chen, Z.W. Li, K. Barkaoui, and A. Giua. On the enforcement of a class of nonlinear constraints on petri nets. *Automatica*, 55:116–124, 2015.

[20] M. P. Cabasino, A. Giua, and C. Seatzu. Diagnosability of discrete-event systems using labeled petri nets. *IEEE Transactions on Automation Science and Engineering*, 11(1):144–153, Jan 2014.

[21] A. Giua, C. Seatzu, and D. Corona. Marking estimation of Petri nets with silent transitions. *IEEE Transactions on Automatic Control*, 52(9):1695–1699, Sept 2007.

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

[23] T. Murata. Petri nets: Properties, analysis and applications. *Procedings of the IEEE*, 77(4):541–580, April 1989.

[24] C.G. Cassandras and S. Lafortune. *Introduction to discrete event systems*. Springer, 2008.

[25] M.P. Cabasino, A. Giua, M. Pocci, and C. Seatzu. Discrete event diagnosis using labeled Petri nets. An application to manufacturing systems. *Control Engineering Practice*, 19(9):989–1001, 2011.

[26] M.P. Cabasino, A. Giua, L. Marcias, and C. Seatzu. A comparison among tools for the diagnosability of discrete event systems. In *2012 IEEE International Conference on Automation Science and Engineering*, pages 218–223. IEEE, 2012.

[27] F. Lin and W.M. Wonham. On observability of discrete-event systems. *Information sciences*, 44(3):173–198, 1988.