Automatica 94 (2018) 205–213

Contents lists available at ScienceDirect

Automatica

journal homepage: www.elsevier.com/locate/automatica

On-line verification of current-state opacity by Petri nets and integer linear programming☆

Xuya Cong [a,b], Maria Pia Fanti [c], Agostino Marcello Mangini [c], Zhiwu Li [a,d,*]

[a] School of Electro-Mechanical Engineering, Xidian University No. 2 South Taibai Road, Xi'an 710071, China
[b] Key Laboratory of Electronic Equipment Structure Design (Xidian University), Ministry of Education, Xi'an 710071, China
[c] Department of Electrical and Information Engineering, Polytechnic of Bari, 70125 Bari, Italy
[d] Institute of Systems Engineering, Macau University of Science and Technology, Taipa, Macau, China

ARTICLE INFO

Article history:
Received 31 December 2016
Received in revised form 26 October 2017
Accepted 7 March 2018

Keywords:
Opacity
Petri nets
Integer linear programming
Decentralized architecture

ABSTRACT

Opacity is a security and privacy property that evaluates whether an external observer (intruder) can infer a *secret* of a system by observing its behavior. This paper proposes an on-line approach to address the problem of current-state opacity in discrete event systems (DESs) modeled in a labeled Petri Net (PN) framework and by observing its evolution. An observation of the system is said to be *current-state opaque* if the intruder is unable to determine whether the current state of the system is within a set of secret states, otherwise it is said to be not current-state opaque. The proposed approach to verify the current-state opacity works on-line: the verification algorithm waits for the occurrence of an observable event and uses Integer Linear Programming problem solutions to verify if the behavior of the system is current-state opaque to the intruder under the given observation. Moreover, the proposed method is applied in two different settings: (i) a centralized approach where the intruder has full knowledge of the system model but can partially observe the system behavior; (ii) a decentralized approach where a set of intruders can observe different event sets and collaborate with a coordinator to disclose the same secret. Finally, experimental results are presented to demonstrate the efficiency of the proposed method.

1. Introduction

The problems of security and privacy have received extensive concerns in on-line services of networked and cyber–physical systems over the last few decades. To formulate these problems, various notions of security and privacy have been proposed in the related literature, such as anonymity, non-interference and opacity (Jacob, Lesage, & Faure, 2015). In particular, opacity is a security and privacy property that evaluates whether an external observer (intruder) can infer a *secret* of a system by observing its behavior. Depending on the definition of the secret, there are two main kinds of opacity properties provided in the related literature: language-based opacity and state-based opacity. State-based opacity defines the secret as a set of secret states and it can be further classified as initial-state opacity (Saboori & Hadjicostis, 2013; Tong, Li, Seatzu, & Giua, 2017), initial-and-final-state opacity (Wu & Lafortune, 2013), current-state opacity (Saboori & Hadjicostis, 2014; Tong et al., 2017) and k-step opacity (Saboori & Hadjicostis, 2011).

This paper focuses on the current-state opacity with respect to (wrt) a secret: for any observation, the intruder cannot infer that the current state of the system belongs to the secret. More precisely, opacity requires that for any observation the set of states estimated by the intruder and consistent with the observation is not a subset of the secret.

Current-state opacity property was introduced in Bryans, Koutny, and Ryan (2005), adapted to labeled transition systems in Bryans, Koutny, Mazaŕe, and Ryan (2008) and developed in finite state automata models in Saboori and Hadjicostis (2014). In the framework of Labeled Petri Nets (LPNs) the current-state opacity is solved in Tong et al. (2017) that proves a necessary and sufficient condition for current-state opacity wrt an arbitrary secret, by using the notion of basis markings and explanations. The advantage of this method is avoiding the exhaustive enumeration of the reachable markings. However, it can be only applied to bounded PNs and a large memory may be still required.

Many large real systems are physically distributed then, as in the case of fault diagnosis (Cong, Fanti, Mangini, & Li, 2017; Fanti, Mangini, & Ukovich, 2013) and supervised control (Ye, Li, & Giua,

☆ The material in this paper was not presented at any conference. This paper was recommended for publication in revised form by Associate Editor Christoforos Hadjicostis under the direction of Editor Christos G. Cassandras.

* Corresponding author at: School of Electro-Mechanical Engineering, Xidian University No. 2 South Taibai Road, Xi'an 710071, China.
E-mail addresses: congxuya@stu.xidian.edu.cn (X. Cong), mariapia.fanti@paliba.it (M.P. Fanti), agostinomarcello.mangini@poliba.it (A.M. Mangini), zhwli@xidian.edu.cn (Z. Li).

https://doi.org/10.1016/j.automatica.2018.04.021
0005-1098/

2015), the opacity verification can exploit the distributed setting of DESs. However, few contributions consider the verification of opacity by taking into account the distributed setting of the system. The study in Badouel, Bednarczyk, Borzyszkowski, Caillaud, and Darondeau (2007) considers several intruders that have different observation masks and secrets. Wu, and Lafortune (2013) extend the opacity notion to a coordinated architecture where multiple intruders work together with a coordinator to discover the same secret. In addition, Paoli and Lin (2012) introduce decentralized opacity definitions for the cases with and without coordination among agents based on languages.

The contribution of this paper is twofold. First, in order to avoid the system state enumeration, this work presents a run-time verification method of current-state opacity by employing LPN models and solving Integer Linear Programming (ILP) problems. Such an approach has been successfully used to solve the on-line fault diagnosis (Basile, Chiacchio, & Tommasi, 2009; Dotoli, Fanti, Mangini, & Ukovich, 2009) and fault diagnosability tests (Basile, Chiacchio, & Tommasi, 2012a). Typically, it is assumed that the structure of the LPN and the initial marking are known by the intruder, which only has partial observation of the events (Tong et al., 2017). In this paper the verification is performed by an algorithm that waits for an observable event and, by ILP problem solutions, determines whether the system behavior remains in the secret or not. If the observation of the system behavior discloses the secret, then the last event is hidden. Hence, the proposed current-state opacity verification falls in the *opacity enforcement at run-time* approach (Jacob et al., 2015): it does not restrict the system behavior and hides some system's output events whenever it is necessary.

In the presented paper, the secret is defined as the conjunction of a set of Generalized Mutual Exclusion Constraints (GMECs) (Giua, DiCesare, & Silva, 1992). Indeed, GMECs describe interesting subsets of the state space of a net and can represent many important state-based specifications (Tong et al., 2017) and control problems (Ma, Li, & Giua, 2015). Moreover, many control specifications and requirements in DESs can be converted into a set of GMECs.

Second, in order to make a contribution in the field of the opacity verification of systems exhibiting a distributed setting, we show how the proposed technique can be extended to a decentralized architecture. We consider a set of local intruders communicating their own output information with a coordinator. Each local intruder has full knowledge of the net structure and its initial marking, but it observes only a subset of the observable events. The coordinator is used to produce the global result as the *global (system) intruder*. For this purpose, we propose a protocol for the communication between the local intruders and the coordinator.

As a conclusion of this section, the two main contributions of this paper are summarized in comparison with the work (Tong et al., 2017).

(1) The presented non-intrusive run-time algorithm can verify: (i) if the observed word (of finite length) is current-state opaque wrt the secret; (ii) if the system is not current-state opaque wrt the secret by avoiding expensive off-line computations. On the contrary, the observer-based approach proposed in Tong et al. (2017) requires the off-line building of the Basis Reachability Graph (BRG) of the LPN wrt the secret.

(2) The proposed methods in centralized and decentralized settings can be applied to the nets with bounded and unbounded state space by avoiding the enumeration of the PN markings as the method in Tong et al. (2017) that can be applied only to bounded LPNs.

However, the limit of both approaches is the computational complexity since it is necessary to solve NP-complete problems in the general case.

The rest of this paper is organized as follows. Section 2 briefly introduces some basics of the PN formalism. Section 3 defines the intruder and Section 4 proposes the on-line algorithm to verify current-state opacity and discusses experimental results to show the efficiency of the approach. Section 5 extends the algorithm proposed in Section 4 to a decentralized architecture and an example illustrates this distributed approach. Finally, Section 6 draws the conclusion.

2. Preliminaries

2.1. Petri nets

This section reviews some basics of PNs (Peterson, 1981) used in the paper.

A PN is a 4-tuple $PN = (P, T, Pre, Post)$, 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 matrices, respectively, which specify the arcs connecting places and transitions. More precisely, for each $p \in P$ and $t \in T$ element $Pre(p, t)$ ($Post(p, t)$) is equal to a natural number indicating the arc multiplicity if an arc going from p to t (from t to p) exists, and it is equal to 0 otherwise. Note that $\mathbb{N}$ is the set of non-negative integers. Matrix $C = Post - Pre$ is the $m \times n$ incidence matrix of the PN.

The state of a PN is given by its current marking that is a mapping $M : P \to \mathbb{N}^m$, assigning to each place an integer number of tokens. A PN system $\langle PN, M_0\rangle$ is a net PN with an initial marking M_0.

A transition $t_j \in T$ is enabled at M if $M \geq Pre(\cdot, t_j)$ holds and $M[t_j\rangle$ is used to denote that $t_j \in T$ is enabled at marking M. When fired, t_j produces a new marking M', denoted by $M[t_j\rangle M'$ that is computed by the PN state equation $M' = M + C \cdot \overrightarrow{t_j}$, where $\overrightarrow{t_j}$ is an n-dimensional firing vector corresponding to the jth canonical basis vector.

Let $\sigma = t_1 t_2 \dots t_k$ be a sequence of transitions (firing sequence) and let k be its length, given by the number of transitions that σ contains. The fact that a transition $t \in T$ appears in the sequence σ is denoted by $t \in \sigma$. Moreover, the notation $M[\sigma\rangle$ denotes that σ is enabled at M and $M[\sigma\rangle M'$ denotes that the firing of σ yields M'. The set of all sequences that can fire in a net system $\langle PN, M_0\rangle$ is denoted by $L(PN, M_0) = \{\sigma \in T^* | M_0[\sigma\rangle\}$. In addition, $\vec{\sigma} : T \to \mathbb{N}^n$ is the firing vector associated with a sequence σ.

A marking M is said to be reachable from $\langle PN, M_0\rangle$ if there exists a firing sequence σ such that $M_0[\sigma\rangle M$. The set of all markings reachable from M_0 defines the reachability set of $\langle PN, M_0\rangle$, which is denoted as $R(PN, M_0)$.

A PN having no directed cycles is said to be *acyclic*. Now, if the PN system $\langle PN, M_0\rangle$ is acyclic, then it is proved that a marking M is reachable from M_0 if and only if there exists a non-negative integer vector $\vec{y}$ satisfying the state equation $M_0 + C \cdot \vec{y} \geq \vec{0}$ (Corona, Giua, & Seatzu, 2004).

2.2. Labeled Petri nets

An LPN is a 4-tuple $G = (PN, M_0, E, \lambda)$ where $\langle PN, M_0\rangle$ is a PN system, E is an alphabet (a set of labels) and $\lambda : T \to E \cup \{\varepsilon\}$ is a labeling function that assigns to each transition $t \in T$ either a symbol $e \in E$ or the empty word ε.

We assume that the intruder has complete knowledge of the net system but partial observation of its behavior. Namely, the set of transitions can be partitioned into $T = T_o \cup T_u$ with $T_o \cap T_u = \emptyset$, where T_o (resp. T_u) is the set of $|T_o| = n_o$ (resp. $|T_u| = n_u$) observable (resp. unobservable) transitions whose occurrence can (resp. cannot) be detected by the intruder. Hence, the labeling function λ is defined as follows: if $t \in T_o$ then $\lambda(t) = e \in E$, and if $t \in T_u$ then $\lambda(t) = \varepsilon$. Here, we assume that the same label $e \in E$ can be associated to more than one transition. In the following, we