Enhanced discrete event model for system identification with the aim of fault detection

Marcos Vincente Moreira, Jean-Jacques Lesage

Enhanced discrete event model for system identification with the aim of fault detection ⋆

Marcos V. Moreira * **Jean-Jacques Lesage** **

* *COPPE, Programa de Engenharia Elétrica, Universidade Federal do Rio de Janeiro, 21949-900, Rio de Janeiro, R.J., Brazil. (e-mail: moreira.mv@poli.ufrj.br)*
** *LURPA, ENS Cachan, Univ. Paris-Sud, Université Paris-Saclay, 94235 Cachan, France (e-mail: jean-jacques.lesage@ens-paris-saclay.fr)*

Abstract: In this paper, we present a new model for discrete-event system identification that is suitable for fault detection, called Deterministic Automaton with Outputs and Conditional Transitions (DAOCT). The model is computed from observed fault-free paths, and represents the fault-free system behavior. In practice, a trade-off between size and accuracy of the identified automaton has to be found. In order to obtain compact models, loops are introduced in the model, which implies that sequences that are not observed can be generated by the model leading to an exceeding language. This exceeding language is associated with possible non-detectable faults, and must be reduced in order to use the model for fault detection. We show, in this paper, that the exceeding language generated by the DAOCT is smaller than the exceeding language generated by other models proposed in the literature, reducing, therefore, the number of possible non-detectable faults. We also show that if the identified DAOCT does not have cyclic paths, then the exceeding language is empty, and the model represents all and only all observed fault-free sequences generated by the system. A practical example is used to illustrate the results of the paper.

Keywords: Discrete-event systems, System identification, Fault detection, Finite automata, Black-box identification.

1. INTRODUCTION

Fault detection and isolation has received considerable attention from the scientific community over the last years. In Sampath et al. (1995), a discrete-event approach for fault diagnosis is introduced, and since then, several works have been proposed for fault detection and isolation, and also for the verification of diagnosability of the system, *i.e.*, the capability of identifying the occurrence of a fault event within a bounded number of occurrences of events (Debouk et al., 2000; Qiu and Kumar, 2006; Moreira et al., 2011; Zaytoon and Lafortune, 2013; Cabral et al., 2015a,b; Cabral and Moreira, 2017). In all these works, it is assumed that the complete system behavior is known, *i.e.*, the system behavior before and after the occurrence of fault events.

Although methods for fault detection based on the complete system behavior can be successfully applied to small systems, they are difficult to be implemented on large and complex systems for the following reasons: *(i)* in general, large automated systems are composed of several components, whose models and interactions between these models, are difficult or even impossible to be obtained; *(ii)* the modeling process requires engineers that know the complete plant behavior, and are also familiar with discrete-event modeling techniques; *(iii)* the post-fault behavior of the system is difficult to be predicted due its size and complexity; and *(iv)* only faults that have been predicted can be detected by the diagnoser computed considering the complete behavior of the system. In order to overcome these problems, fault detection techniques based on an identified fault-free model of the system have been recently proposed (Klein et al., 2005; Roth et al., 2009, 2011). In these works, the two main ideas are: *(i)* to automate the process of obtaining the fault-free model of the system by using identification; and *(ii)* when a fault has been detected through a discrepancy between the system behavior and the model, to use a technique based on residuals for fault localization.

In Klein et al. (2005), a monolithic model for fault detection, that is capable of representing the behavior of a closed-loop system, is proposed. This model is non-deterministic with state outputs, and has been called Non-Deterministic Autonomous Automaton with Output (NDAAO). The NDAAO is obtained from observed sequences of binary signals exchanged between the plant and the controller (sensor signals emitted by the plant and actuator commands generated by the controller), as shown in Figure 1. In Klein et al. (2005), it is shown that the identified NDAAO generates all observed sequences

⋆ This work has been partially supported by the Brazilian Research Council, CNPq, under grants 200536/2017-6 and 309084/2014-8.

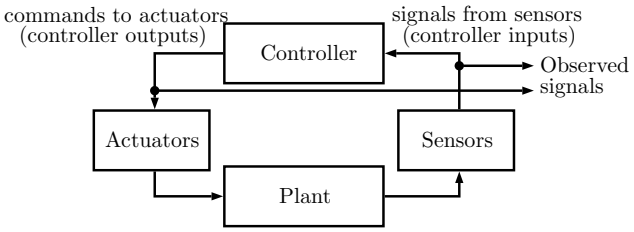

Fig. 1. Closed-loop discrete-event system

of signals used in the identification process. Furthermore, a trade-off between size and accuracy of the identified model can be found thanks to an adequate adjustment of the parametric algorithm used for identification. Indeed, for reducing the size of the model, equivalent states are merged, what introduces loops in the NDAAO, generating sequences that have not been observed. This exceeding language can increase the number of non-detectable faults of the system, and may prevent the fault detection scheme to be implemented. In order to deal with this trade-off, in Klein et al. (2005), a free parameter k, that is used to compute the NDAAO, is introduced, and it is shown that the NDAAO is $k+1$-complete in the sense of Moor et al. (1998), *i.e.*, a sequence of signals of length smaller than or equal to $k+1$ belongs to the identified NDAAO if, and only if, it is observed in the system.

In Roth et al. (2009) and Roth et al. (2011), the fault detection strategy proposed in Klein et al. (2005) is extended to systems with a high degree of concurrency. As in Klein et al. (2005), the NDAAO is used, and the same trade-off between model size and accuracy is observed in these works.

In this paper, we present a new model for discrete-event system identification that is more efficient for fault detection than the method proposed in Klein et al. (2005) and Roth et al. (2009, 2011), called Deterministic Automaton with Outputs and Conditional Transitions (DAOCT). The exceeding language generated by the DAOCT is reduced in comparison with the exceeding language generated by the NDAAO, due to a path estimation function that is added to the model, reducing the number of non-detectable faults. We also show that if the identified DAOCT does not have cyclic paths, then there is no exceeding language. As in Klein et al. (2005), we assume that the binary input and output signals of the controller are measured, generating the observed fault-free paths of the system. Using this information, the DAOCT is computed. The DAOCT also satisfies the property of $k+1$-completeness, if sequences of observed signals are considered, or, equivalently, k-completeness if sequences of events are considered.

This paper is organized as follows. In Section 2, we present some preliminary concepts and the basic ideas of fault detection based on the fault-free behavior of the system. In Section 3, we formulate the problem of system identification with the aim of fault detection, and in Section 4, we introduce the DAOCT model for system identification. In Section 5, we present a practical example to illustrate the results of the paper. Finally, in Section 6, the conclusions are drawn.

2. PRELIMINARIES

2.1 Notation and Definitions

Let $G = (X, \Sigma, f, x_0, X_m)$ denote a deterministic automaton (Cassandras and Lafortune, 2008), where X is the set of states, Σ is the finite set of events, $f : X \times \Sigma^\star \to X$ is the transition function, where $\Sigma^\star$ is the Kleene-closure of Σ, x_0 is the initial state of the system, and X_m is the set of marked states.

The language generated by G is defined as $L(G) = \{s \in \Sigma^\star : f(x_0, s)!\}$, where ! denotes *is defined*. The prefix-closure of a language L is defined as $\overline{L} = \{s \in \Sigma^\star : (\exists t \in \Sigma^\star)(st \in L)\}$. Notice that the language generated by G is prefix-closed by definition.

The function of feasible events $\Gamma : X \to 2^\Sigma$, is defined as $\Gamma(x) = \{\sigma \in \Sigma : f(x, \sigma)!\}$.

The set of all subsequences of a sequence $s \in \Sigma^\star$ is defined as $Sub(s) = \{w \in \Sigma^\star : (\exists t, w, v \in \Sigma^\star)(s = twv)\}$.

A path p of an automaton G is a sequence of states and events that can be executed by the system, *i.e.*, a path $p = (x_1, \sigma_1, x_2, \sigma_2, \ldots, \sigma_{l-1}, x_l)$ is feasible in G if, and only if, $x_i \in X$, for $i = 1, 2, \ldots, l$, $\sigma_i \in \Sigma$, for $i = 1, 2, \ldots, l-1$, and $f(x_i, \sigma_i) = x_{i+1}$, $i = 1, \ldots, l-1$. The length of a path is defined as the number of vertices in the path, and is denoted here as $\|p\|$. Thus, $\|p\| = l$. A path is said to be cyclic if $x_l = x_1$.

Let P be a set of paths, and define function $\psi : P \to \Sigma^\star$, that extracts from a path $p \in P$, the sequence of events associated with p. Thus, if $p = (x_1, \sigma_1, x_2, \sigma_2, \ldots, \sigma_{l-1}, x_l)$, then $\psi(p) = \sigma_1 \sigma_2 \ldots \sigma_{l-1}$.

The length of a sequence of events $s \in \Sigma^\star$ is denoted as $|s|$.

The set of non-negative integers is denoted by $\mathbb{N}$, and the set formed only with 0 and 1 is denoted by $\mathbb{N}_1 = \{0, 1\}$.

The difference between two sets A and B is denoted by $A \setminus B$.

2.2 Fault detection based on the system fault-free behavior

In order to deal with the problem of fault detection of large automated systems, whose complete behavior can be very difficult or even impossible to be obtained, mainly the post-fault behavior, some works in the literature propose the identification of the fault-free behavior of the system. The identified model simulates the observed fault-free behavior of the system, *i.e.*, the language generated by the identified model contains all observed sequences of the system, and is used in the fault detection system. The fault detection system compares the sequences of events or the status of the signals of sensors and actuators, and declares the occurrence of a fault when there is a discrepancy between the observed behavior and the predicted behavior described by the identified model.

In this paper, we propose a fault detection scheme based on the identified fault-free behavior of the system. It is important to remark that since the fault detection scheme is based only on the fault-free behavior, it is capable

of identifying the occurrence of any fault in the system, and not only predicted faults. The drawback of adopting this strategy is that fault isolation is not carried out by the fault detection scheme. This task can be performed offline, after the fault has been detected, by analyzing the history of sequences of events executed by the system and the status of sensors and actuators. Fault isolation is not addressed in this work.

3. DISCRETE-EVENT SYSTEM IDENTIFICATION WITH THE AIM OF FAULT DETECTION

Let us consider the closed-loop system depicted in Figure 1, and assume that the controller has m_i binary input signals, i_h, for $h = 1, \ldots, m_i$, and m_o binary output signals, o_h, for $h = 1, \ldots, m_o$. Let vector

$$u(t_1) = [\, i_1(t_1) \ \ldots \ i_{m_i}(t_1) \ o_1(t_1) \ \ldots \ o_{m_o}(t_1) \,]^T,$$

denote the observation of the controller signals at time instant t_1. Thus, vector $u(t_1)$ represents the status of the system at a given time instant t_1. As the system evolves, the status of the system may change due to changes in sensor readings or actuator commands. Let us consider that there is a change in at least one of the variables of u. Then, at the time instant immediately after this change, t_2, a new vector $u(t_2)$ is observed. Since, in this paper, we consider only untimed system models, we may define the instantaneous changes in the values of the controller signals as the system events, σ, and represent the status of the system $u(t_j)$, by u_j. Thus, the transition from one vector of controller signals u_1 to another vector u_2, is represented by the transition (u_1, σ, u_2). If a sequence of l vectors of controller signals, and the corresponding changes in these signals, is observed, we have an observed path of the system $p = (u_1, \sigma_1, u_2, \sigma_2, \ldots, \sigma_{l-1}, u_l)$.

The objective of system identification is to find a model that is capable of describing the observed behavior of the system. Let us consider that the observed paths of the system are denoted as $p_i = (u_{i,1}, \sigma_{i,1}, u_{i,2}, \sigma_{i,2}, \ldots, \sigma_{i,l_i-1} u_{i,l_i})$, for $i = 1, \ldots, r$, where r is the number of observed paths, and l_i is the number of vertices of each path p_i. Let us also assume that all paths start at the same vertex, *i.e.*, all I/O vectors $u_{i,1}$, for $i = 1, \ldots, r$, are equal. Thus, associated with each path p_i there is a sequence of events $s_i = \sigma_{i,1}\sigma_{i,2} \ldots \sigma_{i,l_i-1}$ and a sequence of output vectors $\omega_i = u_{i,1}u_{i,2} \ldots u_{i,l_i}$. This leads to the following definition of the language observed by the system:

$$L_{Obs} := \bigcup_{i=1}^{r} \overline{\{s_i\}}. \tag{1}$$

It is important to remark that we assume in this paper that none of the paths p_i has an associated sequence of events s_i that is a prefix of the sequence of events s_j of another path p_j, where $i \neq j$. If this occur, then path p_i does not provide any new information, and can be discarded for system identification.

Since the objective of system identification is to find a model that simulates the observed behavior described by L_{Obs}, then the language generated by the identified model, L_{Iden}, must satisfy $L_{Obs} \subseteq L_{Iden}$. This relation between L_{Obs} and L_{Iden} is depicted in the diagram of Figure 2.

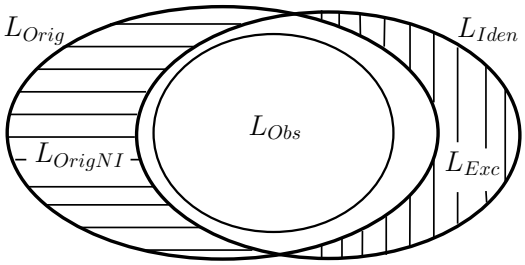

Fig. 2. Relation between the languages L_{Orig}, L_{Iden}, L_{Exc}, and L_{OrigNI}.

In a finite time, only part of the sequences of events that the system can generate can be observed, which means that $L_{Obs} \subset L_{Orig}$, where L_{Orig} denotes the never known language generated by the system. The relation between the observed language and the original language generated by the system is also described in the diagram of Figure 2.

As it can be seen in Figure 2, two other languages can be defined: (*i*) $L_{Exc} = L_{Iden} \setminus L_{Orig}$; and (*ii*) $L_{OrigNI} = L_{Orig} \setminus L_{Ident}$. L_{Exc} represents the sequences of events that can be generated by the identified automaton but do not belong to the original behavior of the system. Since the fault detection strategy is based on the observation of events and comparison with the sequences generated by the model, if a sequence of events that is not in the original fault-free system is observed and is in the language of the identified model, then the fault is not detected. Thus, L_{Exc} represents faulty sequences that cannot be detected by the fault detection system. On the other hand, L_{OrigNI} is associated with the sequences that are in the original fault-free system, but are not identified because the paths associated with these sequences have not been observed. The sequences of events of L_{OrigNI} are associated with false alarms generated by the fault detection system. Clearly, both languages must be reduced in order to obtain an efficient fault detection scheme.

In Klein et al. (2005), it is shown that if the system is observed for a sufficiently long time, then there exists a number $n_0 \in \mathbb{N}$ such that the difference $L_{Orig}^{\leq n_0} \setminus L_{Obs}^{\leq n_0} \approx \emptyset$, where $L_{Orig}^{\leq n_0}$ and $L_{Obs}^{\leq n_0}$ denote the sets formed with all sequences of events of length smaller than or equal to n_0 of L_{Orig} and L_{Obs}, respectively. Thus, since $L_{Obs} \subseteq L_{Iden}$, the subset of L_{OrigNI} formed with all sequences of events of length smaller than or equal to n_0, $L_{OrigNI}^{\leq n_0}$, is also approximately the empty set. Let us assume that $L_{OrigNI}^{\leq n_0} = \emptyset$. Then, all sequences of events of length smaller than or equal to n_0 that does not belong to the identified model are faulty sequences, and the fault detection system will not raise false alarms. This assumption is formalized as follows.

A1. All paths of length $n_0 + 1$ of the original system are observed, and, consequently, $L_{OrigNI}^{\leq n_0} = \emptyset$.

Assumption **A1** being made, the main problem remaining for fault detection and isolation is to reduce the exceeding language. In Klein et al. (2005), the parametric identification algorithm allows to obtain a model satisfying an important property called $k+1$-completeness that guarantees that a sequence of I/O signals u_j, for $j = 1, \ldots, k+1$,

where k is the free parameter used for system identification, belongs to the identified NDAAO if, and only if, it belongs to an observed path p_i, $i = 1, \ldots, r$. By increasing the value of the free parameter k, the exceeding language generated by the NDAAO, L_{Exc}, reduces, but the size of the model grows. Thus, there is a trade-off to be found between complexity and accuracy of the identified model. Some guidelines to choose appropriately k are given in Klein et al. (2005).

In this paper, an equivalent definition of k-completeness, based on sequences of events instead of sequences of observed vectors, is presented. In order to do so, let us first define the set of all observed paths $P := \{p_i : i \in R\}$, where $R = \{1, 2, \ldots, r\}$, and the language formed by all observed subsequences of events of length n, as follows:

$$L^n_{S,Obs} := \{s \in \Sigma^\star : (|s| = n)[\exists i \in R, s \in Sub(\psi(p_i))]\},$$

where $\psi : P \to \Sigma^\star$. Then, a model is said to be k-complete if for all $n \leq k$, $L^n_{S,Obs} = L^n_{S,Ident}$, where $L^n_{S,Ident}$ is the set formed by all subsequences of events of the identified model of length n.

In the next section, we propose an enhanced model for the identification of DES with the aim of fault detection. The model satisfies the k-completeness property, and its exceeding language can be considerably reduced in comparison with the NDAAO for the same value of k. Therefore, with the model proposed in this paper, we can obtain accurate and compact models that describe the fault-free system behavior, and that are suitable for fault detection.

4. DETERMINISTIC AUTOMATON WITH OUTPUTS AND CONDITIONAL TRANSITIONS

We introduce in this paper a modified automaton model that is suitable for fault detection. The modified automaton is deterministic, with a state output function, and the transitions must satisfy a condition to be transposed associated with the observed paths used to construct the model. This automaton is called Deterministic Automaton with Outputs and Conditional Transitions (DAOCT), and is formally defined as follows.

Definition 1. A Deterministic Automaton with Outputs and Conditional Transitions (DAOCT) is the nine-tuple:

$$\text{DAOCT} = (X, \Sigma, \Omega, f, \lambda, R, \theta, x_0, X_f),$$

where X is the set of states, Σ is the set of events, $\Omega \subset \mathbb{N}_1^{m_i+m_o}$ is the set of I/O vectors, $f : X \times \Sigma^\star \to X$ is the deterministic transition function, $\lambda : X \to \Omega$, is the state output function, $R = \{1, 2, \ldots, r\}$ is the set of path indices, $\theta : X \times \Sigma \to 2^R$ is the path estimation function, x_0 is the initial state, and $X_f \subseteq X$ is the set of final states.

The sets of events and I/O vectors associated with each observed path p_i, $i = 1, \ldots, r$, are denoted in this paper, respectively, as Σ_i and Ω_i. Thus, the set of events and the set of I/O vectors of the identified model are, respectively, $\Sigma = \cup_{i=1}^r \Sigma_i$ and $\Omega = \cup_{i=1}^r \Omega_i$.

The DAOCT is obtained from the observed paths p_i, $i = 1, \ldots, r$, by following the steps of Algorithm 1. As in Klein et al. (2005), a free parameter k is used to construct the identified model. Thus, before we present Algorithm 1, let us compute from path p_i, a path p_i^k such that the vertices of p_i^k are sequences of I/O vectors of length at most equal to k as follows:

$$p_i^k = (y_{i,1}, \sigma_{i,1}, y_{i,2}, \sigma_{i,2}, \ldots, \sigma_{i,l_{i-1}}, y_{i,l_i}), \tag{2}$$

where

$$y_{i,j} = \begin{cases} (u_{i,j-k+1}, \ldots, u_{i,j}), \text{ if } k \leq j \leq l_i \\ (u_{i,1}, \ldots, u_{i,j}), \text{ if } j < k \end{cases}. \tag{3}$$

Notice that the sequence of events of p_i^k is equal to the sequence of events of path p_i. Thus, the unique difference between p_i and p_i^k is that each vertex of p_i^k is now associated with a sequence of vectors instead of a unique I/O vector.

Lemma 1. Each vertex $y_{i,j}$ of path p_i^k stores the last $(k-1)$ events executed in path p_i^k, if $j \geq k$, and the last $(j-1)$ events, if $j < k$.

Proof. According to Equation 3, each vertex $y_{i,j}$ of path p_i^k stores the last k I/O vectors generated in path p_i^k, if $j \geq k$, and the last j I/O vectors if $j < k$. Consequently, the last $(k-1)$ signal changes executed in path p_i^k are stored in vertex $y_{i,j}$, if $j \geq k$, and the last $(j-1)$ signal changes are stored in $y_{i,j}$, if $j < k$. Since the events $\sigma_{i,j}$ are associated with the signal changes from vector $u_{i,j}$ to $u_{i,j+1}$, then the proof is concluded. □

In the following example we illustrate the computation of paths p_i^k from observed paths p_i, $i = 1, 2, \ldots, r$.

Example 1. Let us consider a system with three binary controller signals, and let us consider the observation of three paths p_i, $i = 1, \ldots, 3$, given as:

$$p_1 = \left(\begin{bmatrix}1\\0\\0\end{bmatrix}, a, \begin{bmatrix}1\\1\\0\end{bmatrix}, b, \begin{bmatrix}0\\1\\1\end{bmatrix}, c, \begin{bmatrix}0\\0\\0\end{bmatrix}, d, \begin{bmatrix}0\\0\\1\end{bmatrix}, e, \begin{bmatrix}1\\0\\0\end{bmatrix} \right),$$

$$p_2 = \left(\begin{bmatrix}1\\0\\0\end{bmatrix}, g, \begin{bmatrix}0\\0\\0\end{bmatrix}, h, \begin{bmatrix}1\\1\\0\end{bmatrix}, b, \begin{bmatrix}0\\1\\1\end{bmatrix}, c, \begin{bmatrix}0\\0\\0\end{bmatrix}, i, \begin{bmatrix}1\\0\\0\end{bmatrix}, j, \begin{bmatrix}0\\1\\1\end{bmatrix}, l, \begin{bmatrix}1\\0\\0\end{bmatrix} \right),$$

$$p_3 = \left(\begin{bmatrix}1\\0\\0\end{bmatrix}, g, \begin{bmatrix}0\\0\\0\end{bmatrix}, h, \begin{bmatrix}1\\1\\0\end{bmatrix}, b, \begin{bmatrix}0\\1\\1\end{bmatrix}, i, \begin{bmatrix}1\\1\\1\end{bmatrix}, m, \begin{bmatrix}0\\0\\0\end{bmatrix}, d, \begin{bmatrix}0\\0\\1\end{bmatrix}, n, \begin{bmatrix}1\\1\\0\end{bmatrix} \right),$$

where each event is associated with the rising or the falling edge of the controller signals. For instance, $a = \uparrow 2$, denotes the rising edge of the second controller signal, and $b = \downarrow 1 \uparrow 3$, denotes the falling edge of the first controller signal and the rising edge of the third controller signal.

According to Equations (2) and (3), and choosing the free parameter $k = 2$, we obtain the following modified path p_1^2:

$$p_1^2 = \left(\begin{bmatrix}1\\0\\0\end{bmatrix}, a, \begin{bmatrix}1&1\\0&1\\0&0\end{bmatrix}, b, \begin{bmatrix}1&0\\1&1\\0&1\end{bmatrix}, c, \begin{bmatrix}0&0\\1&0\\1&0\end{bmatrix}, d, \begin{bmatrix}0&0\\0&0\\0&1\end{bmatrix}, e, \begin{bmatrix}0&1\\0&0\\1&0\end{bmatrix} \right).$$

The other paths p_2^2 and p_3^2 are omitted here due to the lack of space. □

In order to present the algorithm for the computation of the identified DAOCT, it is also necessary to define the labeling function $\tilde{\lambda} : X \to \Omega^k$, where Ω^k is formed by all sequences of symbols of Ω of length smaller than or equal to k. Function $\tilde{\lambda}$ associates to each state $x \in X$, a sequence of symbols $\omega^k \in \Omega^k$. Let $\tilde{\lambda}_l(x)$ denote the last output vector of $\tilde{\lambda}(x)$.

Algorithm 1. Identification algorithm

Input: Modified observed paths p_i^k, for $i = 1, \ldots, r$

Output: DAOCT $= (X, \Sigma, \Omega, f, \lambda, R, \theta, x_0, X_f)$

Create an initial state x_0, and define $\lambda(x_0) = \tilde{\lambda}(x_0) = y_{1,1}$
$X = \{x_0\}$, $X_f = \emptyset$
for $i = 1$ **to** r
  for $j = 1$ **to** $l_i - 1$
    Find the state $x \in X$ such that $\tilde{\lambda}(x) = y_{i,j}$
    if $\tilde{\lambda}(x) \neq y_{i,j+1}$ for all $x \in X$ **then**
      Create state x' and define $\tilde{\lambda}(x') = y_{i,j+1}$
      $X = X \cup \{x'\}$
      $\lambda(x') = \tilde{\lambda}_l(x')$
    else
      Find $x' \in X$ such that $\tilde{\lambda}(x') = y_{i,j+1}$
    end if
    $f(x, \sigma_{i,j}) = x'$
    Add i to $\theta(x, \sigma_{i,j})$
    if $j = l_i - 1$
      $X_f = X_f \cup \{x'\}$
    end if
  end for
end for

Each transition $x' = f(x, \sigma)$ of automaton DAOCT has a corresponding set $\theta(x, \sigma)$ of indices that is associated with the paths p_i that contains transition (x, σ, x'). Function θ is used in the DAOCT evolution rule to provide a path estimator, such that if the paths associated with a transition are not coherent with the paths of the observed sequence of events, then the transition is not enabled. This fact is clearly presented in the definition of the language generated by the DAOCT. In order to present the language generated by the DAOCT, it is first necessary to extend the domain of function θ to consider the execution of sequences of events, obtaining the extended path estimation function $\theta_s : X \times \Sigma^\star \to 2^R$. θ_s can be defined recursively as:

$$\theta_s(x, \epsilon) = R,$$
$$\theta_s(x, s\sigma) = \begin{cases} \theta_s(x, s) \cap \theta(x', \sigma), & \text{where } x' = f(x, s), \\ & \text{if } f(x, s\sigma)! \\ \text{undefined, otherwise.} \end{cases} \quad (4)$$

The language generated by the DAOCT is given by

$$L(\text{DAOCT}) := \{s \in \Sigma^\star : f(x_0, s)! \wedge \theta_s(x_0, s) \neq \emptyset\}. \quad (5)$$

Notice that a sequence of events s is only feasible in the DAOCT, if $f(x_0, s)$ is defined, and there is at least one path in the path estimate after the occurrence of s, represented by condition $\theta_s(x_0, s) \neq \emptyset$.

It is also possible to define the language formed by all subsequences of events of length n generated by the DAOCT as follows:

$$L_S^n(\text{DAOCT}) := \{s \in \Sigma^\star : (|s| = n)[\exists x_i \in X, f(x_i, s)! \wedge \theta_s(x_i, s) \neq \emptyset]\}. \quad (6)$$

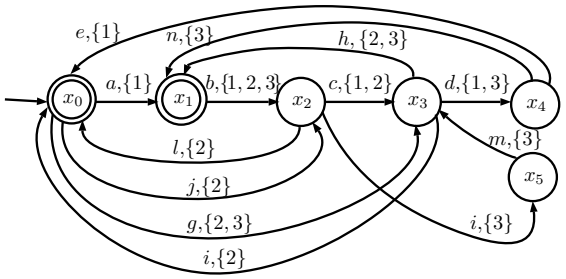

Fig. 3. DAOCT computed considering $k = 1$.

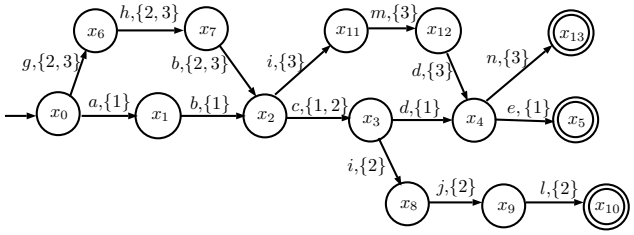

Fig. 4. DAOCT computed considering $k = 2$.

Example 2. Let us now compute the DAOCT obtained according to Algorithm 1 considering the observed paths p_i, $i = 1, 2, 3$, presented in Example 1. In Figures 3 and 4, we present the DAOCT obtained by choosing $k = 1$ and $k = 2$, respectively. The final states are represented by double circles. Notice that each transition is labeled with an event from Σ, and a set associated with the paths p_i^k where each transition is defined, *i.e.*, each transition (x, σ, x'), where $x' = f(x, \sigma)$, of the DAOCT is labeled with σ and $\theta(x, \sigma)$. In addition, notice that, for $k = 2$, the corresponding DAOCT is acyclic. □

The following theorem shows that the observed fault-free language L_{Obs} is a subset of the language generated by the DAOCT, L(DAOCT), *i.e.*, the identified DAOCT simulates the observed fault-free language of the system.

Theorem 1. $L_{Obs} \subseteq L(\text{DAOCT})$.

Proof. Let $s = \sigma_{i,1}\sigma_{i,2}\ldots\sigma_{i,l_i-1}$ be a sequence of events of an observed path $p_i^k = (y_{i,1}, \sigma_{i,1}, y_{i,2}, \sigma_{i,2}, \ldots, \sigma_{i,l_{i-1}}, y_{i,l_i})$. According to Algorithm 1, there is a path in the DAOCT $(x_1, \sigma_{i,1}, x_2, \sigma_{i,2}, \ldots, \sigma_{i,l_i-1}, x_{l_i})$, associated with p_i^k, where x_i is not necessarily distinct from x_j, for $i \neq j$, $i, j = 1, 2, \ldots, l_i$, and $i \in \theta(x_j, \sigma_{i,j})$, for $j = 1, \ldots, l_i - 1$. Thus, any prefix of s belongs to the language generated by DAOCT, which implies that $L_{Obs} \subseteq L(\text{DAOCT})$. □

In the sequel, we prove that the identified model is k-complete.

Theorem 2. For a given value of k, the identified DAOCT is k-complete, *i.e.*, $L_S^n(\text{DAOCT}) = L_{S,Obs}^n$, for all $n \leq k$.

Proof. Since, according to Theorem 1, $L(\text{DAOCT}) \supseteq L_{Obs}$, then $L_S^n(\text{DAOCT}) \supseteq L_{S,Obs}^n$, for all $n \leq k$. Let us now prove that $L_S^n(\text{DAOCT}) \subseteq L_{S,Obs}^n$. Let $p = (x_q, \sigma_q, x_{q+1}, \sigma_{q+1}, \ldots, \sigma_{q+n}, x_{q+n+1})$ be a feasible path of the DAOCT of length $n + 1$, *i.e.*, $\theta_s(x_q, \sigma_q \ldots \sigma_{q+n}) \neq \emptyset$. According to Algorithm 1, any transition of p is associated with a transition in at least one path p_i^k, $i = 1, \ldots, r$. Let us consider the last transition of p, $(x_{q+n}, \sigma_{q+n}, x_{q+n+1})$, and assume that $(\tilde{\lambda}(x_{q+n}), \sigma_{q+n}, \tilde{\lambda}(x_{q+n+1}))$ is the transi-

tion in path p_x^k, where $x \in \{1, 2, \ldots, r\}$, associated with transition $(x_{q+n}, \sigma_{q+n}, x_{q+n+1})$. Let $\tilde{\lambda}(x_{q+n}) = y_{x,q+n}$. According to Lemma 1, all suffixes of length $1, \ldots, k-1$ of the sequences that reach $y_{x,q+n}$ must also belong to p_x^k. Consequently, $\sigma_q \sigma_{q+1} \ldots \sigma_{q+n} \in L^n_{S,Obs}$, for all $n \leq k$, which implies that $L^n_S(\text{DAOCT}) \subseteq L^n_{S,Obs}$, for $n \leq k$. □

It is important to remark that $L_{Exc} = L(\text{DAOCT}) \setminus L_{Orig}$ can be different from the empty set, which means that the fault detection system may be not capable of identifying all system faults. However, this exceeding language is smaller than the exceeding language $L(\text{NDAAO}) \setminus L_{Orig}$, *i.e.*, the exceeding language that is obtained using the model proposed in Klein et al. (2005). Thus, the enhanced model proposed in this paper reduces the number of non-detectable faults in comparison with the NDAAO. In the next theorem we show that if the DAOCT does not have cyclic paths, then $L_{Exc} = \emptyset$.

Theorem 3. If the identified DAOCT does not have cyclic paths for a given value of k, then $L_{Exc} = \emptyset$.

Proof. Notice, according to Algorithm 1, that each transition of the DAOCT is associated with at least one observed path p_i, $i = 1, \ldots, r$. Moreover, since all events of path p_i, $i = 1, \ldots, r$ are observable, then, associated with each path p_i, there is a number $n_i < l_i$ such that p_i can be distinguished from all other paths after the observation of n_i events. Consequently, since the DAOCT does not have cyclic paths, then, after the occurrence of the observed sequence of events $s_i = \psi(p_i)$, we have that $\theta(x_0, s_i) = \{i\}$. In addition, since the DAOCT is acyclic, the intersection of the path estimates of two transitions leaving the same state of the DAOCT must be empty, which implies that all paths p_i are uniquely determined before reaching its corresponding final state. Thus, if a sequence $s \in \Sigma^\star \setminus L_{Obs}$ is observed, then two possibilities may happen: (i) $f(x_0, s)$ is not defined; or (ii) $f(x_0, s)$ is defined, but $\theta_s(x_0, s) = \emptyset$. Thus, $s \notin L(\text{DAOCT})$, which implies that $L(\text{DAOCT}) \subseteq L_{Obs}$, and, therefore, $L_{Exc} = \emptyset$. □

Let us introduce the language generated by the DAOCT formed with all traces of length smaller than or equal to a given value n as follows:

$$L^{\leq n}(\text{DAOCT}) := \left(\bigcup_{i=0}^{n} L^i_S(\text{DAOCT}) \right) \cap L(\text{DAOCT}).$$

According to Theorem 3, if k is chosen such that the DAOCT does not have cyclic paths, then, $L_{Exc} = L(\text{DAOCT}) \setminus L_{Orig} = \emptyset$, and there is no non-detectable faults. In addition, if Assumption **A1** also holds, the observed language $L^{\leq n_0}_{Obs}$ is equal to the original system language $L^{\leq n_0}_{Orig}$, and there is no false alarms for all observed traces of length smaller than or equal to n_0. Thus, under both assumptions, $L^{\leq n_0}(\text{DAOCT}) = L^{\leq n_0}_{Orig}$.

Let us now define the subset of the exceeding language L_{Exc} formed with all sequences of length smaller than or equal to n as $L^{\leq n}_{Exc} = L^{\leq n}(\text{DAOCT}) \setminus L^{\leq n}_{Orig}$. If we consider that $n \leq n_0$, then, according to Assumption **A1**, $L^{\leq n}_{Exc}$ can be rewritten as $L^{\leq n}_{Exc} = L^{\leq n}(\text{DAOCT}) \setminus L^{\leq n}_{Obs}$. In the following example we compare the exceeding language $L^{\leq n}_{Exc}$, for different values of $n \leq n_0$, generated by the DAOCT model proposed in this paper with the NDAAO model presented in Klein et al. (2005).

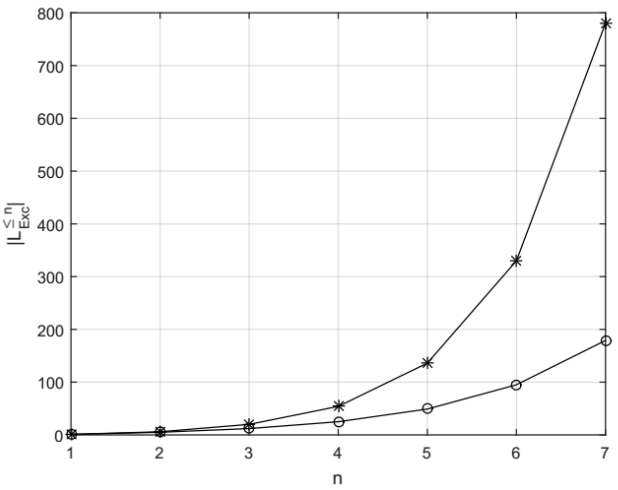

Fig. 5. Comparison between the cardinality of the exceeding language generated by the DAOCT (o) and NDAAO (∗) models, computed considering $k = 1$, for different values of n.

Example 3. In Figure 5, we compare the cardinality of the exceeding language $L^{\leq n}_{Exc}$, generated by the DAOCT model (o) of Example 2, with the exceeding language generated by the NDAAO model (∗), for $k = 1$. Notice that the exceeding language is greatly reduced using the DAOCT model in comparison with the NDAAO model. Moreover, in this case, both models have the same number of states (6 states). This shows that the DAOCT model is more appropriate for fault detection then the NDAAO model proposed in Klein et al. (2005).

It is important to remark that, since, as shown in Example 2, the DAOCT model does not have cyclic paths for $k = 2$, then, in accordance with Theorem 3, the exceeding language generated by the DAOCT model for $k = 2$ is empty for all values of n.

5. PRACTICAL EXAMPLE

A sorting unit is depicted in Figure 6 (Estrada-Vargas et al., 2015). The objective of this system is to sort parcels, that are randomly delivered to Conveyor 1, according to their size. Sensors k_1 and k_2 inform the presence of a parcel and its corresponding size. If the parcel is a small one then $k_1 = 1$ and $k_2 = 0$, and if the parcel is a big one, $k_1 = k_2 = 1$. Cylinder A send big parcels to Conveyor 3, and small parcels to Conveyor 2. When a parcel is positioned in front of Cylinder B or C, it is sent to Conveyor 2 or 3, respectively. Notice that concurrent behavior is possible in this system, since a new parcel may arrive at the system while another one is being processed.

The controller of the system has 4 outputs and 9 inputs. Thus, the vector of controller signals u_j has, in this case, 13 entries. In order to identify the fault-free behavior of the system, six paths p_i, $i = 1, \ldots, 6$, have been observed, and we have computed the DAOCT and the NDAAO models for $k = 2$, $k = 3$, and $k = 7$. In Figure 7, we compare the cardinality of the exceeding language $L^{\leq n}_{Exc}$

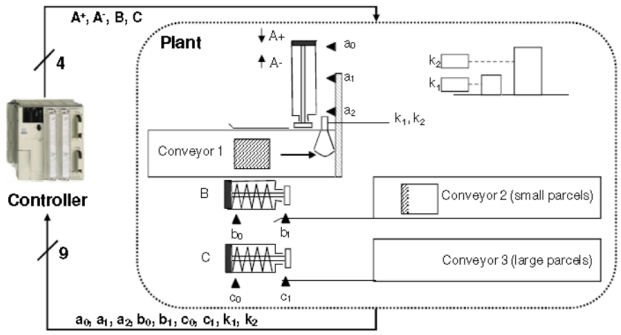

Fig. 6. Sorting unit system

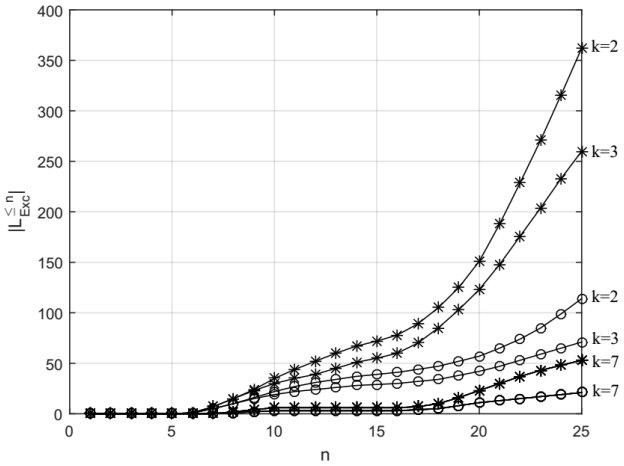

Fig. 7. Exceeding language generated by the DAOCT (o) and NDAAO (∗) models, computed considering $k = 2$, $k = 3$ and $k = 7$, for the sorting unit.

for different values n. As it can be seen from Figure 7, the exceeding language of the DAOCT model is greatly reduced in comparison with the exceeding language for the NDAAO model for all values of k. The number of states of the DAOCT is 60, 70, and 97, for $k = 2$, $k = 3$, and $k = 7$, respectively, while the number of states of the NDAAO is 49, 61, and 92, for $k = 2$, $k = 3$, and $k = 7$, respectively. Although the NDAAO leads to more compact models than the DAOCT for the same value of k, the exceeding language is much larger. Notice that the number of states of the NDAAO for $k = 3$ is close to the number of states of the DAOCT for $k = 2$. However, the exceeding language generated by the NDAAO obtained considering $k = 3$ is greater than twice the exceeding language of the DAOCT obtained considering $k = 2$, for $n = 25$. These results show that using the DAOCT model, more accurate and compact models can be obtained than using the NDAAO model. It is also important to remark that, in this example, the identified DAOCT does not have cyclic paths for $k \geq 19$. Thus, only for $k \geq 19$, the exceeding language $L_{Exc}^{\leq n}$ associated with the DAOCT model is empty for all values of n.

6. CONCLUSIONS

We present in this paper a new model for the identification of the fault-free system behavior. Since the exceeding language of the model is reduced in comparison with other models proposed in the literature, it is more suitable for fault detection. We are currently investigating a fault detection strategy based on the identified model.

REFERENCES

Cabral, F.G. and Moreira, M.V. (2017). Synchronous codiagnosability of modular discrete-event systems. In *IFAC World Congress*, 7025–7030. Toulouse, France.

Cabral, F.G., Moreira, M.V., Diene, O., and Basilio, J.C. (2015a). A Petri net diagnoser for discrete event systems modeled by finite state automata. *IEEE Transactions on Automatic Control*, 59–71.

Cabral, F.G., Moreira, M.V., and Diene, O. (2015b). Online fault diagnosis of modular discrete-event systems. In *Decision and Control (CDC), 2015 IEEE 54th Annual Conference on*, 4450–4455. IEEE.

Cassandras, C. and Lafortune, S. (2008). *Introduction to Discrete Event System*. Springer-Verlag New York, Inc., Secaucus, NJ.

Debouk, R., Lafortune, S., and Teneketzis, D. (2000). Coordinated decentralized protocols for failure diagnosis of discrete event systems. *Discrete Event Dynamic Systems: Theory and Applications*, 10(1), 33–86.

Estrada-Vargas, A.P., López-Mellado, E., and Lesage, J.J. (2015). A black-box identification method for automated discrete event systems. *IEEE Transactions on Automation Science and Engineering*, 14(3), 1321–1336.

Klein, S., Litz, L., and Lesage, J.J. (2005). Fault detection of discrete event systems using an identification approach. In *16th IFAC World Congress*, 92–97. Prague, Czech Republic.

Moor, T., Raisch, J., and Young, S. (1998). Supervisory control of hybrid systems via l-complete approximations. In *Proceedings of the IEEE Workshop on Discrete-Event Systems*, 426–431. Cagliari, Italy.

Moreira, M.V., Jesus, T.C., and Basilio, J.C. (2011). Polynomial time verification of decentralized diagnosability of discrete event systems. *IEEE Transactions on Automatic Control*, 1679–1684.

Qiu, W. and Kumar, R. (2006). Decentralized failure diagnosis of discrete event systems. *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans*, 36(2), 384–395.

Roth, M., Lesage, J.J., and Litz, L. (2009). An FDI method for manufacturing systems based on an identified model. In *13th IFAC Symposium on Information Control Problems in Manufacturing (INCOM2009)*, 1406–1411. Moscow, Russia.

Roth, M., Lesage, J.J., and Litz, L. (2011). The concept of residuals for fault localization in discrete event systems. *Control Engineering Practice*, 19(9), 978–988.

Sampath, M., Sengupta, R., Lafortune, S., Sinnamohideen, K., and Teneketzis, D. (1995). Diagnosability of discrete-event systems. *IEEE Trans. on Automatic Control*, 40(9), 1555–1575.

Zaytoon, J. and Lafortune, S. (2013). Overview of fault diagnosis methods for discrete event systems. *Annual Reviews in Control*, 308–320.