A hierarchical approach for discrete-event model identification incorporating expert knowledge

Ryan P C de Souza, Marcos V Moreira, Jean-Jacques Lesage

A hierarchical approach for discrete-event model identification incorporating expert knowledge

Ryan P. C. de Souza * **Marcos V. Moreira** * **Jean-Jacques Lesage** **

* *COPPE-Electrical Engineering Program, Federal University of Rio de Janeiro, Cidade Universitária, Ilha do Fundão, Rio de Janeiro, 21.945-970, RJ, Brazil (e-mail: {ryanpitanga,moreira.mv}@poli.ufrj.br).*
** *LURPA, ENS Paris-Saclay, Université Paris-Saclay, 94235 Cachan, France (e-mail: jean-jacques.lesage@ens-paris-saclay.fr).*

Abstract: Recently, a new technique for identification of Discrete-Event Systems (DES) with the aim of fault detection has been proposed in the literature, where a model is obtained from the observation of the fault-free system behavior. In some cases, the system may execute different tasks in a sequential order to perform the complete operation of the system. In these cases, the number of observed paths representing the complete system operation may grow exponentially with the number of tasks. In addition, by using black-box identification methods, it is possible that the sequential order that the tasks must be performed is not represented in the model, reducing the fault detection capability or delaying the fault detection. In this paper, a two-level hierarchical approach for DES identification is proposed. In the higher level of the model hierarchy, the system is described by using some basic knowledge of its functioning provided by an expert, and in the lower level of the hierarchy, the behavior is described by black-box identified models for the system tasks. The modeling framework proposed in this paper reduces the number of observed paths needed for system identification and increases the fault detection capability. 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

For the past decades, an important part of the research on Discrete-Event Systems (DES) has been devoted to the subject of fault diagnosis (Sampath et al., 1995; Debouk et al., 2000; Moreira et al., 2011; Zaytoon and Lafortune, 2013; Cabral and Moreira, 2020). In these works, many techniques for verifying diagnosability and for performing fault diagnosis of DES are presented. These techniques rely on the assumption that the complete analytical model of the system is available. This assumption restricts the application of such techniques to small systems, for which an analytical model of the system behavior may be derived by hand. For large-scale and complex systems, it is generally impracticable to build a suitable model for fault diagnosis, since the fault-free behavior and all possible post-fault behaviors should be modeled.

In order to circumvent the aforementioned problems, fault detection techniques based on a model obtained by black-box identification have been proposed in the literature (Klein et al., 2005; Roth et al., 2009, 2011; Moreira and Lesage, 2019a,b). The idea behind this modeling approach is to obtain an automaton that represents the fault-free behavior of the system. Once the identified model has been obtained, fault diagnosis can be performed by comparing the system evolution with the behavior predicted by the model and, if a discrepancy is observed, a fault is detected. After the detection of a fault, methods of fault isolation based on the concept of residuals can be applied (Roth et al., 2011; Moreira and Lesage, 2019b).

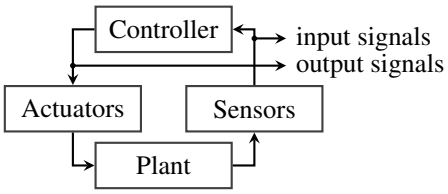


Fig. 1. Closed-loop system showing the signals exchanged between plant and controller.

In Moreira and Lesage (2019a), a model for the identification of closed-loop DES, called Deterministic Automaton with Outputs and Conditional Transitions (DAOCT), is proposed. The construction of this model is based on the acquisition of binary signals exchanged between the programmable logic controller (PLC) and the plant, during fault-free system operation. These signals are formed of the inputs (sensor readings) and the outputs (actuator commands) of the controller, as shown in Figure 1. The inputs for the identification algorithm proposed in Moreira and Lesage (2019a) are the observed paths that the system can execute, where each observed path corresponds to a complete system execution.

Although it has been shown in Moreira and Lesage (2019a,b) that an identified DAOCT model can be successfully obtained for practical systems, in some cases the system may execute different tasks in a sequential order to perform a complete system operation. For instance, the same machine may receive

⋆ This work was partially supported by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, CNPq and FAPERJ.

two types of parts, performing different tasks for each part, and these parts are delivered to the machine always interchangeably. Thus, the complete operation corresponds to processing one type of part, and then processing the other type of part. In these cases, the number of observed paths representing the complete system operation may grow exponentially with the number of tasks. In addition, by using black-box identification methods, it is possible that the sequential order that the tasks must be performed is not represented in the model, reducing the fault detection capability or delaying the fault detection.

We propose, in this paper, a two-level hierarchical approach for identification of a DES with the aim of fault detection. In the lower level of the model structure, DAOCT models are implemented representing the tasks that the system can execute. The DAOCT models are computed using the observed paths associated with each task. In the higher level, the sequential order that the tasks are performed by the system is modeled. This information, provided by an expert, may be given as natural language statements that can be modeled as a Moore automaton representing the possible sequence of tasks. Each vertex of the higher-level automaton is associated with a particular system task, and its transitions describe when there is a change in the task executed by the system. The event labeling the transitions of the higher-level model is associated with the dynamics of the lower-level models, which send information about the occurrence of the event to the higher-level model. The higher-level model, on the other hand, indicates which task must be executed, determining the DAOCT in the lower level that must be run in parallel with the system for fault detection.

It is important to remark that the construction of the hierarchical model is related to the top-down approach for modeling purposes (Suzuki and Murata, 1983; Zhou et al., 1989). However, in the modeling approach proposed in this paper, the lower-level and the higher-level models are synchronized based on information exchanged between them to keep track of their evolutions.

This paper is organized as follows. In Section 2, the notation and some basic concepts used in this work are introduced. In Section 3, we define the DAOCT model and show how it can be used for fault detection. In Section 4, we describe the hierarchical model identification approach proposed in this work, and, in Section 5, we illustrate the results of the paper with a practical example. Finally, the conclusions are drawn in Section 6.

2. PRELIMINARIES

2.1 Notation and definitions

Let $G = (X, \Sigma, f, x_0)$ be a deterministic automaton, 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, with $\Sigma^\star$ denoting the Kleene-closure of Σ, and x_0 is the initial state (Cassandras and Lafortune, 2008). The language generated by G is defined as $L(G) := \{s \in \Sigma^\star : f(x_0, s)!\}$, where symbol '!' denotes 'is defined'.

The prefix-closure of a language $L \subseteq \Sigma^\star$ is defined as $\bar{L} = \{s \in \Sigma^\star : (\exists t \in \Sigma^\star)[st \in L]\}$.

A path $p = (x_1, \sigma_1, x_2, \sigma_2, ..., x_{l-1}, \sigma_{l-1}, x_l)$, where $x_{i+1} = f(x_i, \sigma_i)$, $i = \{1, ..., l-1\}$, is a sequence of states and events that can be executed by G. 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 $\psi(p) = \sigma_1 \sigma_2 \ldots \sigma_{l-1}$.

Definition 1. (Moore Automaton). A Moore automaton is the six-tuple $H = (X_H, \Sigma_H, f_H, x_{0,H}, O, \Lambda)$, where $(X_H, \Sigma_H, f_H, x_{0,H})$ is a deterministic automaton, O is the set of outputs, and $\Lambda : X_H \to O$ is the output function, which assigns an output of O to each state $x \in X_H$. □

The length of a sequence $s \in \Sigma^\star$ is denoted as $\|s\|$. The set of non-negative integers is denoted by $\mathbb{N}$, and the set $\{0, 1\}$ is denoted by $\mathbb{N}_1$. The cardinality of a set A is denoted by $|A|$.

2.2 Identification of Discrete-Event Systems for 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 I/O vector of the system at a given time instant t_1. As the system evolves, the I/O vector 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 I/O vector 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 $p = (u_1, \sigma_1, u_2, \sigma_2, \ldots, \sigma_{l-1}, u_l)$.

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, and that the paths can have cyclic embedded paths. Thus, associated with each path p_i there is a sequence $s_i = \psi(p_i) = \sigma_{i,1} \sigma_{i,2} \ldots \sigma_{i,l_i-1}$, where $\psi : P \to \Sigma^\star$ with $P = \{p_1, \ldots, p_r\}$. As in Moreira and Lesage (2019b), we assume in this paper that none of the paths p_i has an associated sequence of events $s_i = \psi(p_i)$ that is a prefix of the sequence of events of another path p_j, $s_j = \psi(p_j)$, where $i \neq j$.

The following definition of the language observed by the system can be stated:

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

The objective of system identification is to find a model that simulates the observed fault-free behavior described by L_{Obs}. Thus, the language generated by the identified model, L_{Iden}, must satisfy $L_{Obs} \subseteq L_{Iden}$. After obtaining the identified model that simulates the fault-free behavior, it can be used for fault detection by comparing the observed events generated by the closed-loop system with the behavior of the identified model. If the observed behavior is different from the predicted behavior, the fault is detected. In the next section, the model proposed in Moreira and Lesage (2019a) for the identification of DES, and

the method proposed in Moreira and Lesage (2019b) for using this model for fault detection, are presented.

3. FAULT DETECTION BASED ON A DETERMINISTIC AUTOMATON WITH OUTPUTS AND CONDITIONAL TRANSITIONS

In Moreira and Lesage (2019a), an automaton model suitable for fault detection, called Deterministic Automaton with Outputs and Conditional Transitions (DAOCT), is proposed. The DAOCT is obtained from the observed paths p_i, $i = 1,\ldots,r$, and a free parameter k. In order to do so, it is first computed modified paths p_i^k from paths p_i 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

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

Note 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 single I/O vector. The formal definition of a DAOCT is stated in the sequel.

Definition 2. A Deterministic Automaton with Outputs and Conditional Transitions (DAOCT) is the eight-tuple:

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

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, and x_0 is the initial state. □

The labeling function $\tilde{\lambda} : X \to \Omega^k$, where Ω^k is formed of all sequences of symbols of Ω of length smaller than or equal to k, is used in Moreira and Lesage (2019a) to associate to each state $x \in X$, a vertex of one of the paths p_i^k. By increasing the value of k, the language of the identified model, L_{Iden}, is reduced, and the size of the model is increased. Thus, there is a trade-off between size and accuracy of the identified model depending on the choice of the free parameter k. The output $\lambda(x)$ is defined for each state $x \in X$ as the last I/O vector of $\tilde{\lambda}(x)$.

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 contain 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 $\theta_s : X \times \Sigma^\star \to 2^R$, defined recursively as:

$$\theta_s(x,\varepsilon) = R,$$

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

The language generated by the DAOCT is given by

$$L(\text{DAOCT}) := \{s \in \Sigma^\star : f(x_0,s)! \wedge \theta_s(x_0,s) \neq \emptyset\}. \tag{4}$$

Note that a sequence of events $s \in \Sigma^\star$ 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 $\theta_s(x_0,s) \neq \emptyset$.

In Moreira and Lesage (2019b), a procedure for fault detection using the DAOCT model is proposed. The procedure is based on four conditions that the observed path must satisfy to be viable in the model. The first condition is associated with the feasibility of the observed event, and the second condition is associated with the existence of a path in the path estimate. The other two conditions can be easily checked by counting the number of observed events. In the third condition, the minimum number n_i of event observations to distinguish the observed path p_i from the other paths p_j, $j \in R$ and $i \neq j$, is used. It is important to remark that, since it is assumed that each trace $s_i = \psi(p_i)$ cannot be a prefix of another trace $s_j = \psi(p_j)$, where $i \neq j$ and $i,j \in R$, then there always exists a number $0 < n_i < l_i$ associated with each path p_i. Thus, if path p_i is wrongly estimated as the only possible path before n_i event occurrences, then the fault is detected. In the fourth condition, if the final vertex y_{i,l_i} of the estimated path p_i is not reached after $l_i - 1$ event occurrences, then the path estimate is wrong and the fault has occurred. The four conditions are formally presented in the following definition (Moreira and Lesage, 2019b).

Definition 3. Let $s \in \Sigma^\star$ be a model run such that $x = f(x_0,s)$. Then, an event $\sigma \in \Sigma$ is said to be viable in state $x \in X$ of the DAOCT model, if it satisfies the following four conditions:

- **C1.** $f(x,\sigma)!$;
- **C2.** $\theta_s(x_0,s\sigma) \neq \emptyset$;
- **C3.** If $\|\theta_s(x_0,s)\| > 1$ and $\theta_s(x_0,s\sigma) = \{i\}$, then $\|s\sigma\| \geq n_i$;
- **C4.** If $\|s\sigma\| = l_i - 1$, for $i \in \theta_s(x_0,s\sigma)$, then $\tilde{\lambda}(x') = y_{i,l_i}$, where $x' = f(x,\sigma)$, or there exists $j \in \theta_s(x_0,s\sigma)$ such that $\|s\sigma\| < l_j - 1$. □

Conditions **C1** and **C2** guarantee that $s\sigma \in L(\text{DAOCT})$. If Condition **C3** is not true, then path p_i is identified before the minimum number n_i of events that must be observed in order to estimate it. Thus, a fault has occurred. Finally, if Condition **C4** is not true, then the length of the observed trace $s\sigma$ is equal to the maximum length among all sequences of the estimated paths in $\theta_s(x_0,s\sigma)$, without reaching the final vertex of any of these paths, which implies that a fault has occurred.

Another important characteristic that the DAOCT model must satisfy to be used in the fault detection scheme is its reinitializability defined as follows (Moreira and Lesage, 2019b).

Definition 4. Let $s = \psi(p_i^k)$, for $i \in \{1,2,\ldots.r\}$. Then, the DAOCT model is said to be reinitializable if there does not exist $s' \in \overline{\{s\}}$ of length $\|s'\| = l_j - 1$, where $j \in \theta_s(x_0,s')$ and $l_j < l_i$, such that $x' = f(x_0,s')$, and $\tilde{\lambda}(x') = y_{j,l_j}$. □

If the DAOCT model is reinitializable, then if $s = \psi(p_i^k)$ is observed, path p_i^k is uniquely determined, which implies that the DAOCT model can be used for fault detection.

The basic idea of the fault detection scheme is to compare the viable events of the identified fault-free model with the observed events. If the observed event does not satisfy conditions **C1**-**C4** to be viable, then the fault is detected.

4. HIERARCHICAL MODEL IDENTIFICATION

In the black-box identification approach, the model is computed from the observed paths of the system. Thus, in order to obtain an accurate model, it is necessary to observe all paths that the

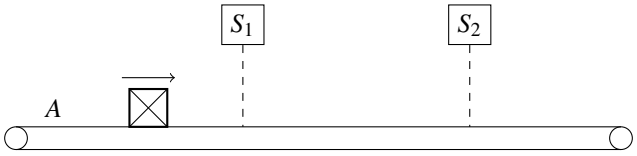


Fig. 2. Scheme for the system described in Example 1.

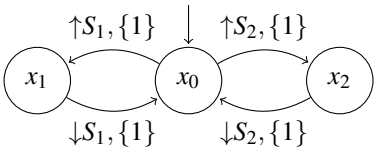


Fig. 3. DAOCT model obtained for the system in Example 1.

system can execute. However, the unique way of guaranteeing that all possible behaviors have been observed is to observe the system for an infinite time. Thus, in practice, we observe the system for a sufficiently long time until we do not observe any new path. Although this strategy works for several practical systems, in some cases, different tasks might need to be carried out sequentially by the system in order to perform the complete system operation, increasing the number of observed paths needed for identification. For instance, let us suppose that two types of parts are delivered to a system, and that each type of part describes a different task t_q, $q = 1,2$, executed by the system. Let us assume that the system always starts executing task t_1. Suppose also that we know that the parts are delivered to the system interchangeably, such that if a part is delivered in a different order, a fault has occurred. In this case, a path of the complete system operation must describe the correct order that the parts are delivered to the system, *i.e.*, the path of the complete operation must be formed of a possible observed path of task 1 concatenated with a possible path associated with task 2. If we consider that each task t_q has r_q possible paths, then the number of possible paths of the complete system operation would be the product $r_1 \times r_2$. This shows that the number of paths needed to obtain the DAOCT model grows exponentially with the number of tasks that the system executes.

Another problem related to the direct construction of a black-box model for the complete system is the reduction of fault detection capability, or the increase in the delay for diagnosis. This problem is illustrated in the following example.

Example 1. Consider the system presented in Figure 2 composed of a conveyor belt and two sensors S_1 and S_2. The conveyor belt is always turned on, and a parcel is always placed at point A of Figure 2. Consider also that at most one parcel can be placed on the conveyor belt, and that a new parcel can be placed on the conveyor only after the removal of the previous one. Assuming that the parcel is moved to the right, then the correct order that the sensors must be activated is S_1 and then S_2. In this case, the fault-free observed path of the system is:

$$p = \left(\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \sigma_1, \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, \sigma_2, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \sigma_3, \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}, \sigma_4, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right),$$

where the first entry of the I/O vector is associated with S_1, the second entry with S_2, and the third entry with the command to turn on the conveyor. In this example σ_1 and σ_2 are the rising and falling edges of S_1, respectively, and σ_3 and σ_4 are the rising and falling edges of S_2, respectively.

The DAOCT obtained for $k = 1$ is presented in Figure 3, where $\lambda(x_0) = [\,0\ 0\ 1\,]^T$, $\lambda(x_1) = [\,1\ 0\ 1\,]^T$, and $\lambda(x_2) = [\,0\ 1\ 1\,]^T$. Note that, since there is only one possible path, the path estimator will always indicate estimate $\{1\}$, as represented in the transitions of the DAOCT. In this case, it can be seen that if sensor S_1 stops working, then none of the conditions **C1-C4** of Definition 3 will be violated, and the fault detector will not be capable of identifying the fault occurrence. An alternative to circumvent this problem would be to use a larger value for k in order to model the correct sequence of observations of the sensor signals. However, this leads to an increase in the size of the DAOCT model. □

In this paper, we present an hierarchical method to model the DES, incorporating some basic knowledge about the system behavior provided by an expert.

4.1 Hierarchical structure

In this paper, we assume that the complete system operation consists of the execution of tasks in a sequential order, and that there is only one sequence of tasks. Let T denote the set of tasks t_q, $q = 1,\ldots,\eta$, where η denotes the number of tasks. Then, the sequence of tasks that the system executes is given by $w = w_1 w_2 \ldots w_\nu$, where $w_z \in T$, for $z = 1,\ldots,\nu$. It is important to remark that the same task can be performed several times in w, and that w can be executed cyclically, *i.e.*, after the execution of task w_ν, the system returns to task w_1 and executes w again. We assume in this paper that the sequence of tasks w is provided by an expert.

A path $p_i \in P$ of the complete system operation is formed of a sequence of subpaths, where each subpath corresponds to a task t_q executed by the system. Let $\pi^q_\xi = (u^q_{\xi,1}, \sigma^q_{\xi,1}, u^q_{\xi,2}, \sigma^q_{\xi,2}, \ldots, u^q_{\xi,l^q_\xi})$, where $u^q_{\xi,j} \in \Omega$, for $j = 1,\ldots,l^q_\xi$, and $\sigma^q_{\xi,j} \in \Sigma$, for $j = 1,\ldots,l^q_\xi - 1$, be a subpath associated with task t_q, and let P_q be the set of subpaths π^q_ξ for $\xi = 1,\ldots,r_q$, where r_q is the number of observed paths of P_q. Let μ_q denote the number of times that t_q occurs in w. Then, the number of possible paths $p_i \in P$ is given by $\Pi^\eta_{q=1} r_q^{\mu_q}$, which shows that the number of paths of P grows exponentially with the number of tasks in w.

Since a path representing the complete system operation is formed of subpaths, then, if task t_{q+1} succeeds task t_q, for every $\xi \in \{1,...,r_q\}$ and $\xi' \in \{1,...,r_{q+1}\}$ we have that $u^q_{\xi,l^q_\xi} = u^{q+1}_{\xi',1}$.

In this paper, we propose a two-level hierarchical structure, where, in the higher level, the sequence of tasks w that the system must execute to complete its operation is described by a Moore automaton, and, in the lower level, each task t_q is modeled as a DAOCT identified by using the subpaths of P_q, for $q = 1,\ldots,\eta$. The higher-level model indicates which task should be executed according to the fault-free system behavior, determining which DAOCT model of the lower-level part must be run in order to carry out fault detection. After the execution of a complete subpath in the DAOCT model in the lower level, the information about the conclusion of the task is communicated to the higher-level model. Then, a transition in the higher-level model is transposed, indicating the next task of w that will be executed. The flow of information between the higher-level and the lower-level models is presented in Figure 4.

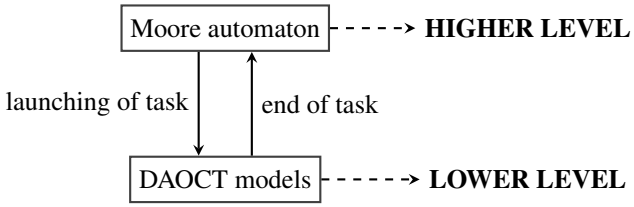


Fig. 4. Scheme showing both levels of the complete model and the flow of information.

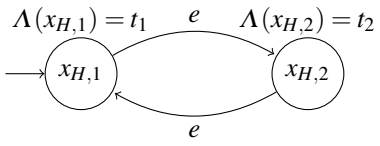


Fig. 5. Moore Automaton describing the sequence of tasks for Example 1.

4.2 Higher-level model

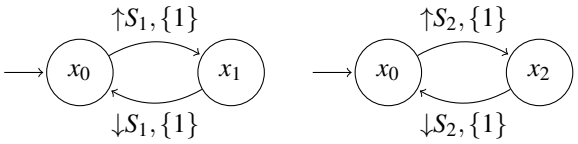
In the higher-level model, the sequence of tasks w provided by the expert is modeled as the Moore Automaton $H = (X_H, \Sigma_H, f_H, x_{0,H}, O, \Lambda)$, where each task w_z of w is associated with a state $x_{H,z} \in X_H$, for $z = 1, \ldots, \nu$. The set of events Σ_H is a singleton $\Sigma_H = \{e\}$, where e denotes the event *end of task*, whose occurrence is associated with the completion of a subpath in a DAOCT model in the lower level. The transition function is defined as $f_H(x_{H,z}, e) = x_{H,z+1}$, for $z = 1, \ldots, \nu - 1$, and $f_H(x_{H,\nu}, e) = x_{H,1}$, if w is cyclical, or $f_H(x_{H,\nu}, e)$ is undefined, otherwise. The initial state of H is defined as $x_{0,H} = x_{H,1}$, and the output of each state $x_{H,z}$ is the task associated with w_z. Thus, $O = T$, and $\Lambda(x_{H,z}) = w_z$, for $z = 1, \ldots, \nu$.

We present in the sequel an example to illustrate the computation of H.

Example 2. Consider the system presented in Example 1, whose scheme is depicted in Figure 2. In this case, an expert could define two tasks for the system, where task t_1 is associated with the detection of the parcel by sensor S_1, and task t_2 the detection of the parcel by sensor S_2. Since it is assumed that the conveyor can only have one parcel at a time, and a parcel is always placed at point A on the conveyor, then the cyclical sequence of tasks $w = t_1 t_2$ would be provided by the expert. Thus, the higher-level automaton H, describing the sequence of tasks w, is obtained as presented in Figure 5. □

4.3 Lower-level model

In the lower level, a DAOCT model, called in this paper $\text{DAOCT}_q = (X_q, \Sigma_q, \Omega, f_q, \lambda_q, R_q, \theta_q, x_{0,q})$, is identified for each task t_q, $q = 1, \ldots, \eta$, of the system, where $\Sigma_q \subseteq \Sigma$ is the set of events observed in t_q and $R_q = \{1, 2, \ldots, r_q\}$. In order to do so, the subpaths $\pi_\xi^q \in P_q$, for $\xi = 1, \ldots, r_q$, are used as the inputs of the algorithm presented in Moreira and Lesage (2019a) for the computation of the identified DAOCT. Thus, it is assumed that the sets P_q, $q = 1, \ldots, \eta$, are given, *i.e.*, all subpaths are correctly obtained and classified for identification. The lower-level model is, therefore, formed of η identified models DAOCT_q. Note that the vertices of the subpaths π_ξ^q, for $q = 1, \ldots, \eta$, are formed of all entries of the I/O controller vector, even if in task t_q not all sensors or actuators are activated. Note also that each DAOCT_q can be obtained using a different free parameter k to model task t_q, and that it must be reinitializable according to Definition 4.



(a) DAOCT_1. (b) DAOCT_2.

Fig. 6. DAOCT model for each task in Example 1.

In the sequel we present an example to illustrate the lower-level DAOCT models.

Example 3. In Figure 6 we present the identified models DAOCT_1 and DAOCT_2 obtained for $k = 1$ from the subpaths:

$$\pi_1^1 = \left(\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \sigma_1, \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, \sigma_2, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right),$$

$$\pi_1^2 = \left(\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \sigma_3, \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}, \sigma_4, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right),$$

where π_1^1 is the subpath associated with task t_1, and π_1^2 is the subpath associated with task t_2. □

4.4 Fault detection using the hierarchical model

The fault detection is carried out using the higher- and lower-level models. In the initial state $x_{0,H}$ of the higher-level model, H outputs the first task to be performed by the system w_1, and the DAOCT_q associated with w_1 is used in the lower-level model to detect faults. Then, for each observed event $\sigma \in \Sigma$, the four conditions presented in Definition 3 are used to verify if σ is viable in the corresponding DAOCT model. If any of the four conditions is violated, then a fault is detected. On the other hand, if all conditions are verified, and the final vertex of the estimated path π_ξ^q is reached after $l_\xi^q - 1$ events, then the task has been completed, and event e representing the *end of task*, is generated. The higher-level model H observes the occurrence of e and makes a transition to the next state $x_{H,2}$, that outputs task w_2, defining the new task model to be run in the lower-level. The process is repeated until a fault is detected. This procedure is described in Algorithm 1.

Algorithm 1. Fault detection algorithm

Input: Automaton H, and DAOCT_q, for $q = 1, \ldots, \eta$.

Output: Fault detection

1: $c \leftarrow 0$
2: Define the current state of H as $x_{curr} \leftarrow x_{H,c+1}$
3: Run the fault detection algorithm proposed in Moreira and Lesage (2019b) having as input the DAOCT_q model associated with task $\Lambda(x_{curr})$
4: **if** the fault is detected **then** stop the algorithm
5: **if** a path of task $\Lambda(x_{curr})$ is completed **then**
  5.1: Communicate the occurrence of event e to H
  5.2: **if** $f_H(x_{curr}, e)!$ **then**
    5.2.1: $c \leftarrow (c+1) mod |X_H|$
    5.2.2: Go to Step 2
  5.3: No fault has been detected

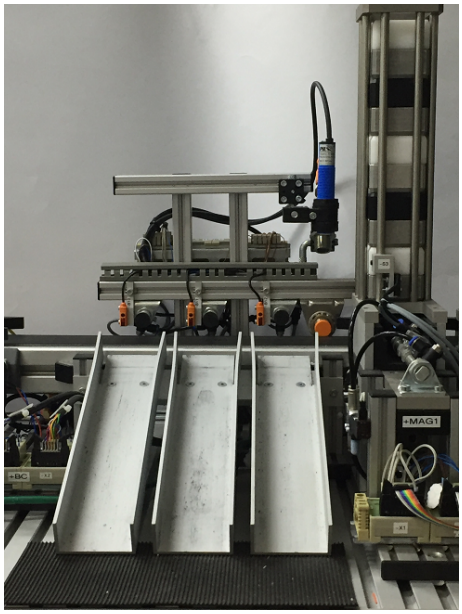


Fig. 7. Sorting unit system of the practical example.

Let L_{HM} denote the language formed of all sequences of the hierarchical model corresponding to a complete system operation. Define language L_{t_q} formed of all sequences of DAOCT$_q$ corresponding to the completeness of task t_q. Then, $L_{HM} = L_{w_1} L_{w_2} \dots L_{w_\nu}$. The following theorem shows that the hierachical model simulates the observed system language.

Theorem 1. $L_{Obs} \subseteq \overline{L_{HM}}$.

Proof. Let us denote by L_{Obs,t_q} the observed language formed of the sequences of the subpaths associated with task t_q, *i.e.*, $L_{Obs,t_q} := \bigcup_{\xi=1}^{r_q} \{\psi_q(\pi_\xi^q)\}$, where function $\psi_q : P_q \to \Sigma_q^\star$ returns the sequence of observed events of a subpath $\pi_\xi^q \in P_q$. Let $L'_{Obs} \subset L_{Obs}$ be the set formed of all sequences $s \in L_{Obs}$ corresponding to a complete system operation. Then, any observed event sequence $s' \in L'_{Obs}$ can be written as $s' = s_1 s_2 \dots s_\nu$, with $s_z \in L_{Obs,w_z}$, $z = 1, \dots, \nu$. According to Moreira and Lesage (2019a), $L_{Obs,t_q} \subset L(\text{DAOCT}_q)$, $q = 1, \dots, \eta$, which implies that $s_z \in L_{w_z}$, $z = 1, \dots, \nu$. It follows that $s' = s_1 s_2 \dots s_\nu \in L_{w_1} L_{w_2} \dots L_{w_\nu} = L_{HM}$, and thus $L'_{Obs} \subseteq L_{HM}$. Since, according to Equation (1), $L_{Obs} = \overline{L'_{Obs}}$, then $L_{Obs} \subseteq \overline{L_{HM}}$. ■

5. PRACTICAL EXAMPLE

The identification method proposed in this paper is illustrated using part of a system that assembles two half cubes to form a cube. The part of the system used in this example is the sorting unit system presented in Figure 7. Three different types of half cubes are sorted in the system: white plastic half cubes (WP), black plastic half cubes (BP), and metallic half cubes (M). Each type of part is pushed to one of the three slides shown on the bottom of Figure 7, such that half cubes of type WP are pushed to the right slide, half cubes of type M are pushed to the slide in the middle, and half cubes of type BP are pushed to the left slide.

On the right of Figure 7, there is a stack magazine where the half cubes are stored in the following order: M, BP, M and WP. This order corresponds to assembling two cubes, where the first one is formed with a metallic half and a black plastic half, and the second one is formed with a metallic half and a white plastic half. If this order is changed, then a fault has occurred. In the sorting process, the parts at the bottom of the stack magazine are placed onto the conveyor belt by a pneumatic pusher. Then, the conveyor belt is turned on, and the part is moved in the direction of two sensors in order to determine its type. An inductive sensor detects metallic parts (type M), and an optical sensor detects metallic (type M) and white plastic parts (type WP). If a black plastic part (type BP) is on the conveyor, then none of the sensors is capable of detecting it. The optical sensor is located close to the inductive sensor, such that metallic parts are detected by both sensors almost at the same time.

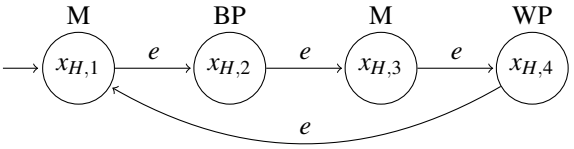


Fig. 8. Higher-level automaton H for the practical example.

It is also important to remark that there is a photoelectric sensor next to each sorting pusher on the conveyor. When a part is detected by the photoelectric sensor next to the pusher that should remove it from the conveyor, the conveyor is stopped and the pusher is extended. Then, the pusher is retracted and a new part can be placed on the conveyor by the pusher of the stack magazine.

5.1 Higher-level part of the model

We assume that the processing of a different type of part can be seen as a task executed by the system. Therefore, there are 3 tasks given by: $t_1 = \text{M}$, $t_2 = \text{BP}$, and $t_3 = \text{WP}$. The sequence of tasks given by the expert is $w = t_1 t_2 t_1 t_3$ to capture the order in which parts of different types must arrive in the system. It is also informed by the expert that this sequence is cyclical. The higher-level automaton, modeling the order of execution of tasks, is shown in Figure 8.

5.2 Lower-level part of the model

The sorting unit system has 13 sensors and 6 actuator signals. Thus, the controller has 19 input and output signals. The initial state of all observed paths of the three tasks is defined as the I/O vector corresponding to the case where the conveyor belt is turned off, and all pushers are retracted.

In the identification procedure, nine subpaths associated with task t_1 (M) were observed, one subpath was observed for task t_2 (BP), and one subpath was observed for task t_3 (WP), which sums up to 11 observed subpaths. Thus, $r_1 = 9$, $r_2 = 1$ and $r_3 = 1$. The reason for which there are multiple subpaths associated with task t_1 is that, since the inductive and optical sensors are very close to each other, the order of sensor readings (rising and falling edges) may change for different sorting cycles of metallic parts, increasing the number of subpaths associated with task t_1.

The DAOCT models for each task were computed using the algorithm proposed in Moreira and Lesage (2019a), where the inputs for building automaton DAOCT$_q$ are the subpaths in P_q, for $q = 1, 2, 3$, and the free parameter $k = 1$. Concerning the number of states and transitions, we have that DAOCT$_1$ has

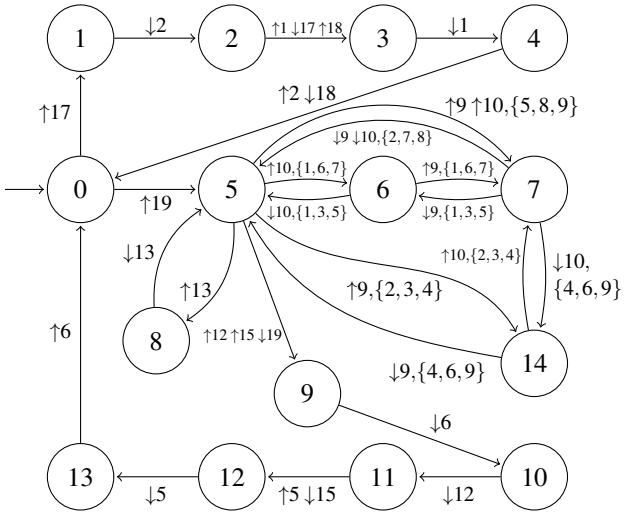


Fig. 9. DAOCT_1.

15 states and 24 transitions, DAOCT_2 has 13 states and 16 transitions, and DAOCT_3 has 12 states and 14 transitions. Due to lack of space, only DAOCT_1 is shown in Figure 9, where the rising and the falling edge of the j-th element of vector u is denoted by $\uparrow j$ and $\downarrow j$, respectively. For simplicity, when $\theta_q(x,\sigma) = R_q$, the transition is not labeled with set R_q.

5.3 Fault detection using the complete hierarchical model

The advantages of using a fault detector based on the hierarchical model proposed in this paper can be illustrated as follows. Assume that, during system operation, the current state estimate in the higher-level model is $x_{H,2}$, and that the system is currently executing task $t_2 = \text{BP}$ with no fault occurrences. Thus, a black part is being processed in the sorting unit system. As soon as this part is pushed to the left slide, event e is communicated to the higher-level model to inform the end of the subpath that was being executed. Then, the current state estimate becomes $x_{H,3}$, meaning that a subpath associated with a metallic part is expected to be executed. However, suppose that a fault occurs in the inductive sensor and it stops working. Then, if a metallic part is pushed by the stack magazine, the observed subpath will be equivalent to a subpath of a white plastic part. In this case, the subpath executed by the system will be associated with task t_3 instead of task t_1, which is the task that is supposed to be performed by the system. Since the lower-level model which is being run to keep track with the system evolution during this subpath is DAOCT_1, the fault will be detected by applying Algorithm 1. Note that all subpaths associated with task t_1, which are all simulated by DAOCT_1, must contain the event corresponding to the activation of the inductive sensor.

Note that if a unique DAOCT were used to model the behavior of the system instead of the hierarchical model proposed in this paper, paths corresponding to the complete system operation would have to be considered. These paths would be composed of the concatenation of all subpaths associated with each task, following the sequence of tasks provided by the expert. The number of times tasks t_1, t_2 and t_3 appear in sequence w are, respectively, $\mu_1 = 2$, $\mu_2 = 1$, and $\mu_3 = 1$. Thus, the total number of paths associated with the complete system operation is given by $r_1^{\mu_1} \times r_2^{\mu_2} \times r_3^{\mu_3} = 81$. We can see, in this example, the reduction in the number of paths needed for identification. While using a unique DAOCT it is necessary to observe 81 paths, using the hierarchical model proposed in this paper it is necessary to observe only 11 subpaths. It is also important to remark that, in this example, the fault in the inductive sensor would be detected using a unique DAOCT model with $k = 1$, but with a much larger delay for its detection.

6. CONCLUSIONS

In this paper, we propose a two-level hierarchical model for the identification of Discrete-Event Systems, by using expert knowledge about the sequence of tasks that must be executed by the system in order to complete its operation. The new model reduces the number of observed paths needed for identification, and also increases the fault detection capability of the system. We are currently investigating the modeling of concurrent tasks in the higher-level model, and also the case that there are several distinct sequential tasks that the system can perform to complete an operation.

REFERENCES

Cabral, F.G. and Moreira, M.V. (2020). Synchronous diagnosis of discrete event systems. *IEEE Transactions on Automation Science and Engineering*, 17(2), 921–932.

Cassandras, C.G. and Lafortune, S. (2008). *Introduction to Discrete Event Systems*. Springer Publishing Company, Inc., New York, NY, USA, 2nd edition.

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

Klein, S., Litz, L., and Lesage, J.J. (2005). Fault detection of Discrete Event Systems using an identification approach. In *16th IFAC World Congress*. Praha, Czech Republic.

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*, 56(7), 1679–1684.

Moreira, M.V. and Lesage, J.J. (2019a). Discrete event system identification with the aim of fault detection. *Discrete Event Dynamic Systems*, 29(2), 1–19.

Moreira, M.V. and Lesage, J.J. (2019b). Fault diagnosis based on identified discrete-event models. *Control Engineering Practice*, 91, 104101.

Roth, M., Lesage, J.J., and Litz, L. (2009). An FDI method for manufacturing systems based on an identified model. *IFAC Proceedings Volumes*, 42(4), 1406 – 1411. 13th IFAC Symposium on Information Control Problems in Manufacturing.

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 Transactions on Automatic Control*, 40(9), 1555 – 1575.

Suzuki, I. and Murata, T. (1983). A method for stepwise refinement and abstraction of Petri nets. *Journal of Computer and System Sciences*, 27(1), 51 – 76.

Zaytoon, J. and Lafortune, S. (2013). Overview of fault diagnosis methods for discrete event systems. *Annual Reviews in Control*, 37(2), 308 – 320.

Zhou, M., DiCesare, F., and Desrochers, A.A. (1989). A top-down approach to systematic synthesis of Petri net models for manufacturing systems. In *Proceedings, 1989 International Conference on Robotics and Automation*, volume 1, 534 – 539.