Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2010, Article ID 453254, 21 pages
doi:10.1155/2010/453254

Review Article

A Comparative Analysis of Recent Identification Approaches for Discrete-Event Systems

Ana Paula Estrada-Vargas,[1,2] Ernesto López-Mellado,[1] and Jean-Jacques Lesage[2]

[1] *CINVESTAV, Unidad Guadalajara, Avenue Científica 1145, Colonia El Bajío, 45010 Zapopan Jal, Mexico*
[2] *LURPA, ENS de Cachan. 61 Avenue du Président Wilson, 94235 Cachan Cedex, France*

Correspondence should be addressed to Ernesto López-Mellado, elopez@gdl.cinvestav.mx

Received 17 August 2009; Revised 15 December 2009; Accepted 20 May 2010

Academic Editor: Paulo Batista Gonçalves

Analogous to the identification of continuous dynamical systems, identification of discrete-event systems (DESs) consists of determining the mathematical model that describes the behaviour of a given ill-known or eventually unknown system from the observation of the evolution of its inputs and outputs. First, the paper overviews identification approaches of DES found in the literature, and then it provides a comparative analysis of three recent and innovative contributions.

1. Introduction

Analogous to the identification of continuous dynamical systems, identification of discrete-event systems (DESs) consists of determining the mathematical model that describes the behaviour of a given DES from the observation of the evolution of inputs and outputs and possibly from other knowledge about the system behaviour.

The automated building of discrete-event models from external observation of system behaviour interests applications such as reverse engineering for (partially) unknown systems, fault diagnosis, or system verification. The first results that constitute the theoretical basis of DES identification approaches were called grammars inference [1]; the aim was the building of a finite automaton from positive samples of accepted words. Later several techniques that synthesise context-free grammars or Petri nets (PNs) have been proposed.

During the current decade, the interest on the DES identification problem grew yielding methods oriented to industrial systems for discovering or rediscovering the functioning of legacy control/management systems. Based on diverse approaches, these methods obtain mathematical models expressed as finite automata (FA) or PN, from inputs and/or outputs sequences observed in a passive way during the operation of the system

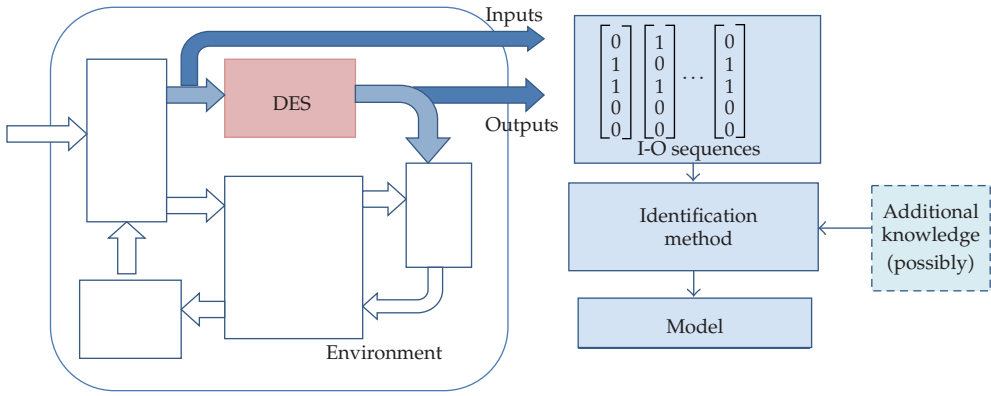

Figure 1: Passive identification of a DES during its operation.

within its environment (see Figure 1). The obtained models are close approximations to the actual system behaviour.

The paper presents a comparative study of identification approaches of DES. It focuses on three different approaches described in recent publications: (i) a progressive identification approach proposed by Meda-Campaña [2] in which several algorithms have been proposed allowing the online identification of concurrent DES, (ii) an offline input-output approach, proposed by Klein [3, 4] in which, through an efficient technique oriented to fault diagnosis, it is obtained a nondeterministic FA representing exactly the observed behaviour, (iii) and an offline approach based on an integer linear programming (ILP) technique initially proposed by Giua and Seatzu [5] and extended by several works like those of Cabasino et al. in [6] and Dotoli et al. in [7] which leads to free-labelled PN models representing observed sequences.

The remainder of this paper is organised as follows. In Section 2 the earlier works on the matter are overviewed. In Section 3, a summarised description of the abovementioned approaches to DES identification is presented. Then, in Section 4 the comparative analysis is developed. Finally concluding remarks and future trends are given.

2. Overview of Identification Techniques

2.1. *Original Approaches from Computer Sciences*

The first identification methods appeared in the field of theoretical computer sciences as a problem of obtaining a language representation from sets of accepted words; such methods are considered as learning techniques.

Gold's method for identification in the limit [1] processes positive samples: an infinite sequence of examples such that the sequences contain all and only all of the strings in the language to learn.

The Probably Approximately Correct (PAC) learning technique in [8] learns from random examples and studies the effect of noise on learning from queries.

The query learning model proposed in [9] considers a learning protocol based on a "minimally adequate teacher"; this teacher can answer two types of queries: membership query and equivalence query.

Several works adopted state machines as representation model, allowing description of the observed behaviour.

In [10] a method to model a language as Moore or Mealy machine is presented. The system under investigation is placed within a test bed and connected to a so-called experimenter, which generates the input signals and records the output signals of the system. The identification can be started considering a very few number of states. If, at some point of the experiment, it is impossible to find a correct machine with the assumed number of states, then the identification is started again considering a machine with one more state.

The method proposed in [11] obtains models representing Mealy machines. The presented method does not require any a priori knowledge of the system, and only a single observed sequence is available. The algorithm lists all reduced machines which may produce the input-output sequence given. The construction principle is the merging of equivalent states.

In [12] a method to identify nondeterministic Moore machines based on a set of input-output sequences is presented. All of the sequences start in the same initial state. The identification principle is the reduction of an initial machine represented as a tree.

In [13] it is presented a method manipulating simultaneously a sample of sequences to produce a convergent series of Mealy machines such that the behaviour of every new machine includes the behaviour of the previous one. The automaton is built step by step. At each step, the already available machine is examined and completed by adding transitions and possibly new states.

Later, in [14] an algorithm to identify a unique Moore machine generating the behaviour observed during m sequences starting at the same initial state is proposed. The learning procedure operates in three steps: induction, contradiction, and discrimination. A state can never be deleted, and only transitions between states can be modified. This method is improved in [15]; it proposes two algorithms to identify multiple systems as well as systems that may not be initialized between two records.

The identification problem for context-free grammars (CFGs) needs, beside given examples, some additional structural information for the inference algorithm [16].

The study in [17] has investigated a subclass of CFGs called simple deterministic grammars and gave a polynomial time algorithm for exactly identifying it using equivalence and membership queries in terms of general CFGs.

In [18] it has been shown that the inference problem for even linear grammars can be solved by reducing it to one for deterministic finite automata (DFA); a polynomial time algorithm for the reduction of the DFA has been presented.

Other works use as description formalism Petri net models. In [19] an algorithm for synthesising Petri net models is presented. The proposed algorithm has two phases. In the first phase, the language of the target system is identified in the form of DFA. In the second phase, the algorithm guesses from the DFA the structure of a Petri net that accepts the obtained language.

2.2. DES Identification Approaches

In recent years, model identification methods are oriented towards the description of (partially) unknown DES. The observed sequences of DES' outputs and/or inputs are processed for obtaining a model that describes its behaviour.

In [20] an identification method based on the least square estimator has been presented; later, several extensions to this work [2, 21–25] provided solutions to the updating of a model from the continuous recording of output sequences.

Another recent method [3, 4], which has been extended to distributed identification [26, 27], allows to build a non deterministic FA from a set of input-output sequences measured from the DES initial condition of functioning. The method was proposed for obtaining exact models adapted for fault detection in a model-based diagnosis approach [28].

In [5] an approach to build a free-labelled PN from a finite set of transitions strings is presented. The approach is based on the solution of (ILP) an Integer Linear Programming Problem. The obtained PN generates exactly the given strings thanks to the creation of examples and counter examples during the procedure. Several identification techniques have been derived from this seminal work [6, 7, 29–34] for dealing with diverse aspects of DES identification.

The first methods mentioned above deal with the modelling of a given language using different representations, whilst the recent methods addressed the problem of automated modelling of DES from observed behaviour based on model identification techniques. These works are represented in Figure 2 according to a classification, discussed in Section 4.

3. Recent Approaches of DES Identification

3.1. Choice of the Considered Approaches

In this section we overview three different approaches adopted in recent publications addressing the specific problem of DES identification; they have been selected because of the soundness of their results. The first approach deals with unknown partially measurable DES exhibiting cyclic behaviour; overviewed results were reported in [2, 21–25]. The second approach is offline; it is oriented to obtain models devoted to model-based fault diagnosis; literature of this approach can be found in [3, 4, 26–28]. The third approach was initially defined as offline and later extended to be online executed; it deals with DES that does not necessarily have binary outputs; the review is presented from some of the representative works among those in [5–7, 29–34].

3.2. Progressive Identification

Problem

The problem addressed in this work is to build a model for a DES as it evolves from the observation of its output signals [2]. This work can be considered as a basis for verification of systems, hardware or software, or it can be extended to address problems of reverse engineering.

Approach

The identification procedure computes an Interpreted Petri Net (IPN) model describing the behaviour of the unknown DES. Some assumptions are considered on the type of systems

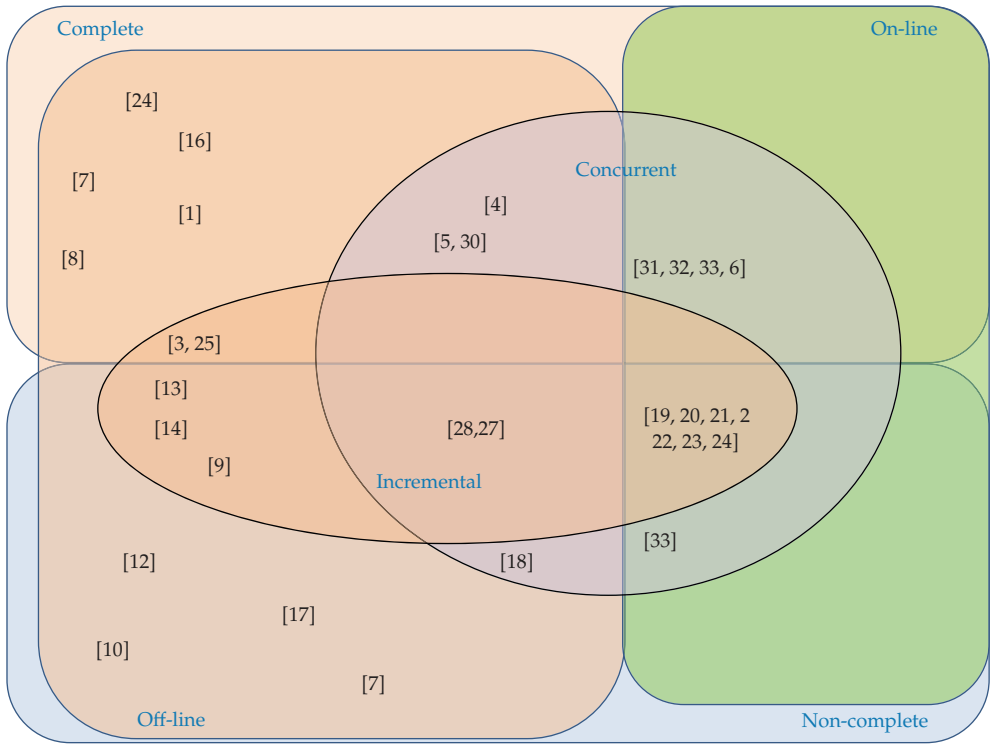

Figure 2: Classification of identification techniques.

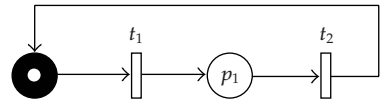

Figure 3: t-component associated with $m_1 = t_1t_2$.

to be identified: they can be described by a live, safe, cyclic, without self-loops, and event-detectable IPN Q whose transitions are not fired simultaneously.

Methodology

A sequence of models is built in such a way that the current model acquires more details than the previous one approaching to the actual model of the system.

The algorithm receives a sequence of output signals obtained from observations during the system operation. These output signals must be binary vectors representing the current state of every one of the sensors measuring the output behaviour of the system.

The procedure returns an IPN whose measurable places represent the sensors of the system and nonmeasurable places represent internal states. Every reachable marking of the Petri net represents the current state of the system at each moment.

The strategy of the identification is based on the reconstruction of the cyclic components of the system model, by processing cyclic sequences of transitions (called m-words) computed from the observed output symbols.

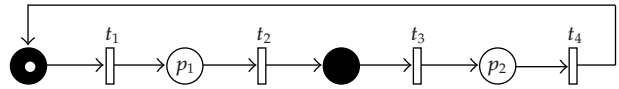

Figure 4: t-semiflow inferred $W_1 = m_1 m_2$.

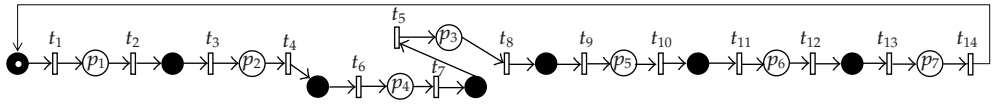

Figure 5: t-semiflow inferred $W_1 = m_1 m_2 m_3 m_4 m_5 m_6 m_7$.

The model identification procedure performs mainly two tasks: the computation of the measurable part of the system and the inference of the nonmeasurable part of the system.

Algorithm

Progressive identification is conducted as follows

(1) Read the vectors of output symbols generated by the system.

(2) Detect an output word (*m-word*) when the first and last output symbols are the same.

(3) For every two consecutive output symbols, compute a transition representing the output change.

(4) Compute an m-word adding each computed transition in the step above.

(5) Compute non measurable (dark) places

  (a) to constrain the firing order of the transitions to the order in which they were computed,

  (b) to compute the t-component associated with the m-word.

(6) Update the IPN model allowing firing of all computed m-words and inferring t-semiflows by

  (a) computing new measurable places and transitions,

  (b) removing or adding dependencies (possibly merging places) updating the t-semiflows.

Example 3.1. In order to illustrate the method, we take from [24] the following example of a system with 7 output signals. For sake of brevity, only main steps are shown.

Step 1. First output symbols are

$$o_1 = [0000000]^T, \quad o_2 = [1000000]^T, \quad o_3 = [0000000]^T = o_1, \quad o_4 = \cdots. \tag{3.1}$$

Step 2. The first cyclic observed sequence is $o_1 o_2 o_1$.

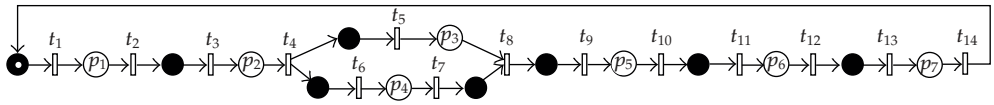

Figure 6: Complete t-semiflow $W_1 = m_1m_2m_3m_4m_5m_6m_7$ and inferred t-semiflow $W_2 = m_1m_2m_{3\text{-}4}$.

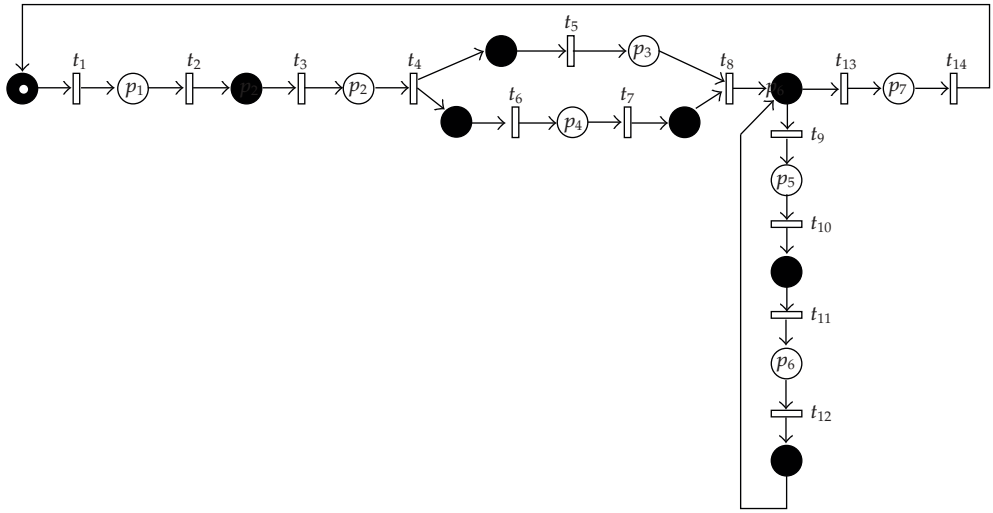

Figure 7: Final model obtained by the progressive identification.

Step 3. t_1 will represent the transition from o_1 to o_2, and t_2 will represent the transition from o_2 to o_1.

Step 4. The m-word resulting is $m_1 = t_1t_2$.

Step 5. The t-component associated with the m-word t_1t_2 is shown in Figure 3.

Step 6. The first t-semiflow inferred is $W_1 = m_1$.

Then the next output word is treated with Steps 1–4; the m-word $m_2 = t_3t_4$ is obtained. Its respective t-component associated is added to infer a new t-semiflow $W_1 = m_1m_2$ in Step 6, as shown in Figure 4.

After computing the m-words $m_3 = t_6t_7$, $m_4 = t_5t_8$, $m_5 = t_9t_{10}$, $m_6 = t_{11}t_{12}$, and $m_7 = t_{13}t_{14}$, it is inferred in Step 6 the t-semiflow $W_1 = m_1m_2m_3m_4m_5m_6m_7$ shown in Figure 5.

The detection of the m-word $m_1 = t_1t_2$ is the first one of $W_1 = m_1m_2m_3m_4m_5m_6m_7$. Then, it is supposed that W_1 has been completely observed and a new t-semiflow $W_2 = m_1$ is inferred. Observed m-words $m_2 = t_3t_4$ and $m_{3\text{-}4} = t_5t_6t_7t_8$ in Step 4 are added to the t-semiflow W_2, and the model is updated in Step 6 to allow the firing of all of them, as shown in Figure 6.

The last m-word $m_7 = t_{13}t_{14}$ is observed. It is made a merging of places to allow the firing of the m-words observed in the order they appeared. A new t-semiflow $W_3 = m_5m_6$ is inferred and t-semiflows $W_1 = m_1m_2m_3m_4m_7$ and $W_2 = m_1m_2m_{3\text{-}4}m_7$ are updated. The final model is shown in Figure 7.

Complexity

The proposed algorithms to update the non measurable places have linear complexity on the number of the transitions computed and the m-words detected. Then, the complete algorithm to update a model that includes all of the updating procedures of non measurable places is executed also in polynomial time.

Limitations

In some cases, the obtained model may represent an exceeding behaviour with respect to the observed one from the system. Furthermore, in this approach only outputs of the DES are observed. Consequently, the state evolution of the systems that does not provoke an output evolution cannot be identified. Additionally, the behaviour represented by structures such as self-loops, shared resources in mutual exclusion, and implicit nonmeasurable places cannot be identified using this methodology

3.3. Parametric Automata Construction

Problem

In this work a method for building finite automaton from a set of inputs and outputs sequences measured during the system evolution is presented [3, 4]. The method was proposed for obtaining models adapted for fault detection in a model-based approach [28].

Approach

The identification approach proposes to compute a nondeterministic finite automaton with outputs (NDAAO) describing the behaviour of the unknown DES. The definition of the NDAAO will be presented below. The system to be identified is a compound system (controller + plant) running in a closed loop considered as an event generator.

Methodology

The algorithm receives a set of observed sequences obtained from the system to be identified. Each observed sequence is an ordered series of input/output (I/O) binary vectors exchanged between controller and plant during operation. As a consequence, observed sequences do not necessarily have the same length; however, the first and last I/O vectors of all sequences are identical (cyclic functioning).

The procedure yields a Nondeterministic Autonomous Automaton with Output (NDAAO). Each state of the NDAAO gives as output a binary vector representing every one of the observed I/O signals of the system.

The first step of the construction of the NDAAO is to fix a parameter k that represents the maximal length of the words (or sequences of I/O vectors) that will be generated by the constructed NDAAO. Basically, the principle of the algorithm is to create states that represent observed words of length k, which are connected through transitions in the order the words have been observed.

Algorithm. A nondeterministic autonomous automaton with output, denoted as NDAAO, is a five-tuple NDAAO = $(X, \Omega, r, \lambda, x_0)$, where X is a finite set of states, Ω is an output alphabet, $r : X \to 2^X$ is a nondeterministic transition relation, $\lambda : X \to \Omega$ is an output function, and $x_0 \in X$ is the initial state. The algorithm operates in six steps as follows.

(1) For each observed sequence of I/O vectors σ_i, construct k-length sequences of vectors $u_i(t)$, where k is the a priori fixed parameter.

(2) Construct the NDAAO.

(3) Rename the output function.

(4) Reduct the last state.

(5) Merge the equivalent states.

(6) Close the automaton.

Example 3.2. Let us consider the example of an elementary plant with a controller having two inputs and one output [3]. The observed sequences of I/O vectors are

$$\begin{aligned}
\sigma_1 &= \left(\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \right), \\
\sigma_2 &= \left(\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \right).
\end{aligned} \tag{3.2}$$

For the sake of readability every I/O vector is coded using a symbol, namely, A, B, C, D, and E representing the observed alphabet. Then the observed sequences are: $\sigma_1 = (\mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}, \mathrm{E}, \mathrm{A})$ and $\sigma_2 = (\mathrm{A}, \mathrm{C}, \mathrm{B}, \mathrm{C}, \mathrm{D}, \mathrm{A})$.

Step 1. After choosing a parameter k value ($k = 2$ in the example), construction of vector sequences of length k is given as

$$\begin{aligned}
\sigma_1^2 &= ((\mathrm{A}, \mathrm{A}), (\mathrm{A}, \mathrm{B}), (\mathrm{B}, \mathrm{C}), (\mathrm{C}, \mathrm{D}), (\mathrm{D}, \mathrm{E}), (\mathrm{E}, \mathrm{A}), (\mathrm{A}, \mathrm{A})), \\
\sigma_2^2 &= ((\mathrm{A}, \mathrm{A}), (\mathrm{A}, \mathrm{C}), (\mathrm{C}, \mathrm{B}), (\mathrm{B}, \mathrm{C}), (\mathrm{C}, \mathrm{D}), (\mathrm{D}, \mathrm{A}), (\mathrm{A}, \mathrm{A})).
\end{aligned} \tag{3.3}$$

Step 2. Construction of the NDAAO. The identification principle is to associate each different word to a single state. This step is illustrated in Figure 8.

Step 3. Renaming of the Output Function. Each state of the NDAAO corresponds to a unique and stable value of the input and output signals. This value is described by the last letter of each k-length sequence.

Step 4. Reduction of the Last State. The last k states of each branch ending in x_f are labelled with the same letter. These states can be reduced through a procedure that has to be iterated $k - 1$ times. First, merge the prestates of x_f; second, redefine this new state as the final state x_f and delete the former x_f from the set of states. Steps 3 and 4 are illustrated in Figure 9.

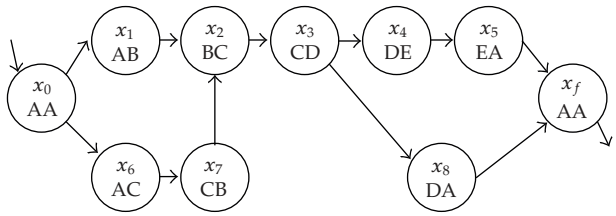

Figure 8: Association of words with states of the NDAAO.

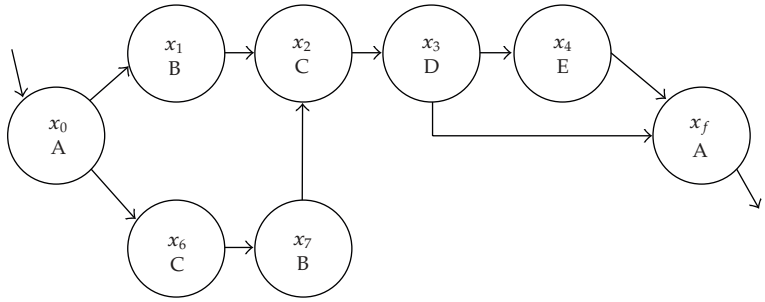

Figure 9: Reduction of the last state.

Step 5. Merging of Equivalent States. Two states are equivalent if and only if

(1) they are associated with the same output,

(2) they have the same set of posterior states.

It has been proved that the merging of equivalent states does not affect the languages accepted by the NDAAO.

Step 6. Closure of the Automaton. Assuming that each observed sequence corresponds to a single production cycle, the states x_0 and x_f of the NDAAO identified are identical. Thus, the NDAAO can be closed resulting in a strongly connected NDAAO. Execution of Steps 5 and 6 can be observed in Figure 10.

Complexity

The time required to build different models is very low and the application of the identification method is efficient. However, the reduction of the NDAAO requires more time than the identification of the model but is not damming.

If new information is available, the time required for the identification of the NDAAO is reduced. However, this gain is not very important since the reduction must be performed again.

Limitations

For a given value of the identification parameter k, the identified NDAAO is $(k+1)$-complete [35], this means that the NDAAO identified for a given value of the parameter k represents exactly the set of observed words of length lower or equal to $k+1$.

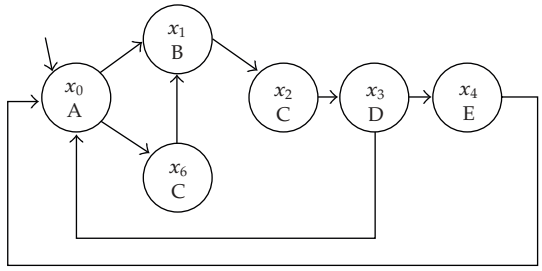

Figure 10: Final model for Example 3.2.

Concurrency cannot be explicitly represented in the obtained automaton, but recent extensions of this work [26] allow performing distributed automation based on an optimal partitioning of I/O that aims at minimizing concurrency between the subsystems. Nevertheless, this technique is dedicated to fault detection and isolation [27].

3.4. Integer Linear Programming Approach

Problem. Several extensions to the original technique presented in [5] have been proposed. We present here only one of the most recent works based on Integer Linear Programming [7]. The problem to be solved is the DES identification by computing a Petri Net model using the observation of events and the available output vectors.

Approach

The identification approach proposes to compute an IPN model such that the observed sequence of events belongs to the language accepted by the IPN. The method considers several hypotheses as follows

(A1) All of the DES events can be detected, distinguished, and not silent.

(A2) The DES can be (partially) observed.

(A3) The DES can be modelled by an IPN system with λ-free labelling function.

(A4) The set of measurable places has a priori a given cardinality q.

(A5) There is an upper bound on the number of places of the IPN.

Methodology

The algorithm receives sequences of events with their corresponding output vectors and the a priori upper bound of the number of nonmeasurable places.

The algorithm returns an IPN with places representing the sensors of the system and labelled transitions representing the observed events. It is also possible that the algorithm returns a 0 (zero) when there is no possible solution of the problem with the given input.

The strategy of the algorithm is to generate an Integer Linear Programming (ILP) problem adding one linear algebraic constraint for every one of the restrictions on the IPN. For selecting among several solutions, it is minimized a performance index. Such an index generally involves arcs weights and number of tokens in the initial marking of the Petri net.

Algorithm. First, we present some definitions taken from [7].

The PN set is given as $D = \{\text{PN} = (P, T, \mathbf{Pre}, \mathbf{Post}) : \mathbf{Pre} \in \mathbb{N}^{mxn}, \mathbf{Post} \in \mathbb{N}^{mxn}\}$.

$\mathcal{L}^E(\text{PN}, \mathbf{M}_0)$ is the λ-free language of the Petri net PN in E^*, given the initial marking $\mathbf{M}_0$.

Let us consider a DES with event set E and language $\mathcal{L}$ verifying assumptions (A1), (A2), and (A3). Let us observe an event sequence $\omega \in \mathcal{L}$ and the corresponding output vectors $\mathbf{y} \in \mathbb{N}^q$. The identification problem consists in determining a place set P and its cardinality m, a transition set T and its cardinality n, as well as a λ-free labelling function λ and a PN system $\{\text{PN}, \mathbf{M}_0\}$ satisfying assumptions (A4) and (A5) such that $\text{PN} \in D$, $\mathbf{M}_0 \in \mathbb{N}^m$ and $w \in \mathcal{L}^E(\text{PN}, \mathbf{M}_0)$.

A net system is a solution of the identification problem if and only if it satisfies the following set of linear algebraic constraints:

$$\xi(w, \mathbb{Y}, \lambda, T, m) = \begin{cases} \text{Pre}, \text{Post} \in \mathbb{N}^{m\times n}, \\ M_i \in \mathbb{N}^m \quad \text{with } i = 0, \ldots, h, \\ \text{Post}^T \vec{1}_{m\times 1} + \text{Pre}^T \vec{1}_{m\times 1} \geq \vec{1}_{n\times 1}, \\ \text{Post}\ \vec{1}_{n\times 1} + \text{Pre}\ \vec{1}_{n\times 1} \geq \vec{1}_{m\times 1}, \\ \forall t_{\beta_i}^{\alpha_i} \in \sigma \quad \text{with } \lambda(\sigma) = \omega, \quad \text{Pre}\ \vec{t}_{\beta_i}^{\alpha_i} \leq M_{i-1}, \\ \forall t_{\beta_i}^{\alpha_i} \in \sigma \quad \text{with } \lambda(\sigma) = \omega, \quad (\text{Post} - \text{Pre})\vec{t}_{\beta_i}^{\alpha_i} = M_i - M_{i-1}. \end{cases} \tag{3.4}$$

The first two constraints are derived from the definitions of markings and Pre and Post incidence matrices. Third and fourth linear algebraic constraints avoid isolated transitions and places, respectively. Constraints five and six are related with enabling and firing of transitions.

Some constraints can be added if additional structural properties are given. For example, if there is no place without successor transitions, then, it can be added $\text{Pre} \cdot \vec{1}_{n\times 1} \geq \vec{1}_{m\times 1}$ and if there are no source transitions, it can be added $\text{Pre}^T \cdot \vec{1}_{m\times 1} \geq \vec{1}_{n\times 1}$.

Since there is not always only one PN satisfying the constraint set, it is used a performance index as follows.

$$\phi(\text{Pre}, \text{Post}, M_0) = \vec{a}^T \text{Pre}\, \vec{b} + \vec{c}^T \text{Post}\, \vec{d} + \vec{e}^T M_0. \tag{3.5}$$

Now, the basis of the algorithm that solves the identification problem stated above is presented. The complete algorithm and a more accurate explanation of the solution are included in [7].

(1) Initiate the algorithm variables.

(2) Wait until a new vector and its corresponding output vector are observed.

(3) Associate a transition to the event as follows.

(3.1) If the event occurs for the first time, a new transition is created

(3.2) If the event occurred previously consider the following.

(3.2.1) If a transition related to the event has the same observed output change, then take such a transition and associate it to the event

(3.2.2) Otherwise, a new transition is created.

(4) Solve the ILP problem

$$\min \phi(\mathrm{Pre}, \mathrm{Post}, M_0), \quad \text{s.t. } \xi\left(w, \mathbb{Y}, \lambda_w, T_w, m'\right). \tag{3.6}$$

Starting with m' equal to the number of measurable places and incrementing its value, until a solution is found or until m' is equal to the upper bound of the number of places.

Example 3.3. The following example is taken from [7]. Let us consider a DES with $y \in \mathbb{N}^5$ and $\overline{m} = q = 5$. Assume that the initial output is $y_0 = [00102]^T$ and the observed sequence is $w = e_{\alpha_1}, e_{\alpha_2}, e_{\alpha_3}, e_{\alpha_4} = e_1, e_2, e_2, e_1$ with the corresponding outputs $y_1 = [40101]^T$, $y_2 = [31001]^T$, $y_3 = [01011]^T$ and $y_4 = [00102]^T$. At each event occurrence, the identification algorithm is applied, adding constraints to obtain a PN without neither transitions nor places without successors. However, no solution is provided until the occurrence of the last event. The ILP solved is

Minimise

$$\begin{bmatrix}1 & 1 & 1 & 1 & 1\end{bmatrix}(\mathrm{Pre} + \mathrm{Post}) \begin{bmatrix}1\\1\\1\\1\end{bmatrix} + \begin{bmatrix}1 & 1 & 1 & 1 & 1\end{bmatrix} M_0, \tag{3.7}$$

subject to

(1) $\mathrm{Pre}, \mathrm{Post} \in \mathbb{N}^{5\times 4}$,

(2) $M_i \in \mathbb{N}^5$ with $i = 0, \ldots, h$,

(3) $\mathrm{Post}^T \vec{1}_{5\times 1} + \mathrm{Pre}^T \vec{1}_{5\times 1} \geq \vec{1}_{4\times 1}$,

(4) $\mathrm{Post}\, \vec{1}_{4\times 1} + \mathrm{Pre}\, \vec{1}_{4\times 1} \geq \vec{1}_{5\times 1}$,

$$\begin{gathered}\mathrm{Pre}\, \vec{t}_{\beta_i}^{\alpha_i} \leq M_{i-1}\mathrm{Pre}\left(t_1^1\right) \leq \begin{bmatrix}0\\0\\1\\0\\2\end{bmatrix}, \qquad \mathrm{Pre}\left(t_1^2\right) \leq \begin{bmatrix}4\\0\\1\\0\\1\end{bmatrix},\\ \mathrm{Pre}\left(t_2^2\right) \leq \begin{bmatrix}3\\1\\0\\0\\1\end{bmatrix}, \qquad \mathrm{Pre}\left(t_2^1\right) \leq \begin{bmatrix}0\\1\\0\\1\\1\end{bmatrix}.\end{gathered} \tag{3.8}$$

(5) for all $t_{\beta_i}^{\alpha_i} \in \sigma$ with $\lambda(\sigma) = w$

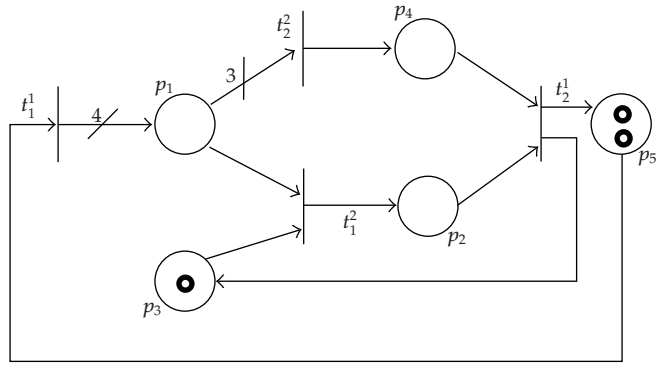

Figure 11: Solution for identification problem of Example 3.3.

(6) for all $t_{\beta_i}^{\alpha_i} \in \sigma$ with $\lambda(\sigma) = w$, $(\text{Post} - \text{Pre})\vec{t}_{\beta_i}^{\alpha_i} = M_i - M_{i-1}$

$$
\begin{aligned}
(\text{Post} - \text{Pre})\left(t_1^1\right) &= \begin{bmatrix}4\\0\\1\\0\\1\end{bmatrix} - \begin{bmatrix}0\\0\\1\\0\\2\end{bmatrix}, \qquad (\text{Post} - \text{Pre})\left(t_1^2\right) = \begin{bmatrix}3\\1\\0\\0\\1\end{bmatrix} - \begin{bmatrix}4\\0\\1\\0\\1\end{bmatrix},\\
(\text{Post} - \text{Pre})\left(t_2^2\right) &= \begin{bmatrix}0\\1\\0\\1\\1\end{bmatrix} - \begin{bmatrix}3\\1\\0\\0\\1\end{bmatrix}, \qquad (\text{Post} - \text{Pre})\left(t_2^1\right) \leq \begin{bmatrix}0\\0\\1\\0\\2\end{bmatrix} - \begin{bmatrix}0\\1\\0\\1\\1\end{bmatrix}.
\end{aligned}
\tag{3.9}
$$

The IPN obtained is illustrated in Figure 11.

Complexity

In small-size examples, an optimal solution is obtained in a short time implementing and solving the ILP problem on a computer equipped with a standard solver of optimization problems.

In order to apply the identification algorithm online, the dynamics of the DES has to be slow with respect to the time required to solve the ILP problem at each occurrence.

Limitations

It is necessary to fix a-priori the upper bound on the number of places. The statement of ILP problem from a set of observed sequences is exponential. ILP is nondeterministic polynomial, and a solution is not always found.

4. Comparative Analysis of DES Identification Approaches

In this section a comparative analysis of the approaches mentioned above is provided. First, a set of criteria are introduced; then the analysis of every one of the methods regarding the given features is presented. Finally, the analysis is summarized in a comparative table.

4.1. Methods Characteristics

Several features have been considered in [4]; some others are added to have a more complete scope during comparative analysis. Considered characteristics are structured into 4 categories: those characterizing the DES to be identified, those describing the identification process, those qualifying the identified model, and those considering general algorithm features.

DES Characteristics

(i) *Type of inputs/outputs*. In the general case, inputs and outputs of DES to be identified are *discrete* (they can take a finite number of values). If all inputs and outputs can only take two values (on/off), the DES is called *logic.*

(ii) *Iterative behaviour*. A DES is called *cyclic* if it iteratively reaches the initial state during its operation. If it iterates over the same behaviour revisiting a state that is not the initial one, then it is called *repetitive.*

Identification Process Characteristics

(i) *Operation Mode*. If the input sequences cannot be modified, identification is *passive*. Otherwise, the identification is *active*; it is allowed to force input sequences to the actual system to explore behaviours that may not be included in the observed functioning of the system. Since all identification methods considered here are passive, this criterion is not taken into account for comparison aspects.

(ii) *A Priori Information*. If there is no available knowledge about the DES other than its inputs and outputs evolution, then the identification is *absolute* (commonly called black-box). Otherwise, the identification is *relative*.

(iii) *Model Updating*. When the model construction is *incremental*, the method progressively updates the model from observed information; otherwise, the identification procedure is *global*: it must be executed on the whole of the observed sequences every time new sequences are collected.

Identified Model Characteristics

(i) *Concurrency*. This feature considers whether the obtained model can represent *explicitly* concurrent behaviour observed from the system.

(ii) *Accuracy*. This term is related with completeness of the identified model. If this model represents exactly the observed behaviour, then it is *complete*.

Algorithm Characteristics

(i) *Considered Data*. The identification algorithm constructs an identified model starting from experimental data that can be *inputs* and/or *outputs* of the observed system.

(ii) *Strategy*. If the identification algorithm returns all possible models representing the observed behaviour, the algorithm is called *enumerative*. If only one of the possible models is given, it is *constructive*.

(iii) *Execution*. If the construction of the model can be performed during the system operation by computing a new model from new measurements of the system inputs and/or outputs, the execution is made *online*. Otherwise, the execution is *offline*; the algorithm is not able to run at the same time than the system.

(iv) *Complexity*. This term refers to the computational complexity of the identification algorithm. *Polynomial* time procedures are better than *exponential* ones for coping with large systems exhibiting a large amount of input-output sequences.

4.2. Analysis of Methods

According to the abovementioned features, the identification techniques are analysed.

Progressive Identification

This approach only considers *logical* systems which are *not assumed to be cyclic*. Systems to identify are not assumed to be reinitialized: a single output sequence measured from an arbitrary instant is processed as input to the identification algorithms. However, for long sequence observation if the system exhibits iterative behaviour, this can be captured into the model.

The identification process is *passive*; inputs given to the system cannot be forced. It is considered that there is no information a-priori about the system; only the output sequences are taken into account; this means that the identification is *absolute*.

Since the observed behaviour of the system is progressively integrated to a model, the identification is *incremental*: every time a change on the outputs is observed, the model is updated computing observable part of the system and inferring internal states.

The system identification approach introduced obtains as model an interpreted Petri net. As a consequence of using Petri nets, the concurrency can be *explicitly* represented in the model.

All of the observed sequences are represented in the model, but this approach is *not complete* because the model could represent exceeding behaviour, that is, nonobserved sequences.

Only one solution is given; the solution strategy is *constructive*.

The algorithm provides procedures to be *online* executed, since it is supposed to construct a model while the system is working.

The algorithms are executed in *polynomial* time.

Parametric Automata Construction

This method works on *cyclic logical* systems. The identification procedure receives a set of recorded sequences whose first and last vectors are always the same.

The identification approach is *passive*: inputs given to the system cannot be manipulated. The DES to identify is considered a black-box: *absolute* identification is made.

When new cyclic sequences are provided, they are processed and the model can be updated. It is not necessary to process the whole set of sequences; thus the method is *incremental.*

Due to the limitations of the NDAAO, the concurrency *cannot be explicitly* expressed within the model structure.

Obtained model represents all and only all the observed sequences of a given length; thus the algorithm is *complete*.

Algorithm receives a set of I/O vector sequences and an identification parameter. It returns a unique solution. Thus, it is *constructive.*

I/O sequences are recorded *offline*. The algorithm works in *polynomial* time.

Integer Linear Programming Approach

Considered systems on this approach are *not necessarily cyclic*. Its outputs can be *discrete*(i.e., they can have a finite number of values).

The identification method is *passive*. Since it is required to fix an upper bound for the number of places of the PN, this method is not considered as black-box identification; then, it is *relative* (also-called gray-box identification).

Every time new information is observed, a new ILP is stated and solved; that is, the identification procedure is *global*.

Structure of the obtained model has the form of a PN, which can include *concurrency*.

All observed behaviour is represented on the language of the obtained IPN, but, since counterexamples are not considered in the statement of the ILP problem [5], nonobserved behaviour could be included in the obtained model: the methodology is *not complete*.

The identification algorithm constructs a model able to reproduce a single *event-output* sequence obtained from the system to identify.

The identification algorithm is *enumerative* if we do not consider a performance index. Otherwise, it is constructive, but there could be no solution for a given problem.

In order to apply the identification algorithm *online*, the dynamics of the DES has to be slow with respect to the time required to solve the ILP problem at each occurrence. If this condition is not fulfilled, then the algorithm must run *offline*.

The application is limited to small-length sequences because the ILP statement grows *exponentially*; besides, it is known that solution of ILP is computationally expensive.

4.3. Discussion

Main features of the analysed methods are summarized in Table 1. It can be observed that the progressive identification approach is well adapted for online identification, since it works incrementally in polynomial time. Nevertheless, since it is not a solid methodology, there could be more output sequences than the observed ones; that could be a problem dealing with some type of applications, such as fault diagnosis.

The parametric identification method is not conceived for online execution, but the current model can be incrementally updated when new behaviour is recorded. Although the synthesised model does not represent explicitly the concurrency, the observed input/output sequences of length $k + 1$ are exactly represented.

Table 1: Main characteristics of identification approaches.

Comp. criteria	Identif. approach		
	Progressive approach	Parametric automata approach	Integer programming approach
DES to be identified characteristics			
Type of inputs/outputs	Logical	Logical	Discrete
Iterative behaviour	Repetitive	Cyclic	None
Identification process characteristics			
A-priori information	Absolute	Absolute	Relative
Model updating	Incremental	Incremental	Global
Identified model characteristics			
Concurrency	Explicit	Implicit	Explicit
Accuracy	Noncomplete	Complete	Noncomplete
Algorithm characteristics			
Considered data	Outputs	Inputs and outputs	Events and outputs
Strategy	Constructive	Constructive	Enumerative
Execution	Online	Offline	Offline/online
Complexity	Polynomial	Polynomial	Exponential

Despite the inefficiency inherent to integer linear programming, which limits the identification method to process small-size sequences, the ILP procedure yields concurrent models allowing nonbinary markings, when a solution is found.

A more detailed comparison of the methods cannot be made because they do not consider the same hypothesis, and the provided sequences representing the observed behaviour have different formats. Furthermore, the synthesised model is not expressed using the same formalism.

However, from this study we can point out several key features that an identification method should have for dealing with large and complex DESs.

First and foremost, the method must take into account both inputs and outputs of the system, and yield a model expressing explicitly concurrent behaviours; thus PN seems to be a more appropriate modelling formalism.

An efficient (polynomial time) technique based on a progressive strategy makes irrelevant the way the input/output sequences are collected; the sequences may be processed as they are obtained allowing updating the model, if necessary, during the system operation.

Regarding the accuracy of the obtained model, one may think that exceeding behaviour must be avoided. Nevertheless, it is not possible to assure that all of the behaviours have been exhibited by the system during the sequence observations; thus, it could be interesting to infer nonobserved behaviour from the collected sequences. The challenge is discerning among possible exceeding representations by determining plausible model structures; partial knowledge on system components and operations could be useful for this task (independent operations, mutual exclusions, etc.).

5. Summary and Perspectives

An overview of identification techniques for Discrete-Event Systems as well as of the theoretical results on which they are based has been given. Three of the most innovative contributions to this open problem have been outlined. Based on the analysis of their main characteristics, this analysis has been done regardless of any hypothesis on the technology of the systems to be identified (manufacturing, communication, management systems, etc.). These three approaches are very recent and must not be considered as completely finalised. Nevertheless, the study of published results allows perceiving their enormous potential. Currently the interest for DES identification is considerably increasing, and probably, as it is the case since a long time in the field of continuous systems, identification techniques will offer powerful alternatives to classical modelling techniques "by knowledge" for complex DES.

The proposed comparative study allows exhibiting the advantages and drawbacks of the reviewed methods. In some words, we can summarize that, even if it allows identifying systems with logical and discrete (i.e., taking a finite number of values) inputs and outputs, the ILP approach cannot be applied today to identify real complex DES, because of the computational complexity of the algorithm. The main advantage of the parametric method is to generate a complete identified model, but the obtained model cannot represent explicitly the concurrency. The progressive identification method is well adapted to deal with concurrent systems yielding an updated model as the systems evolve; however, the model represents more behaviour than that observed.

New approaches that combine the advantages of these pioneer ones have now to be explored. In [36] we describe the first results obtained in a recent project that aims providing an efficient method for building incrementally, as the DES evolves, a complete Petri net model capturing concurrency.

Acknowledgment

A. P. Estrada-Vargas has been supported by CONACYT, Mexico, Grant no. 50312.

References

[1] E. M. Gold, "Language identification in the limit," *Information and Control*, vol. 10, no. 5, pp. 447–474, 1967.

[2] M. E. Meda-Campaña, *On-line identification of discrete event systems: fundamentals and algorithms for the synthesis of Petri net model*, Ph.D. thesis, Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, Unidad Guadalajara, Mexico, November 2002.

[3] S. Klein, L. Litz, and J.-J. Lesage, "Fault detection of discrete event systems using an identification approach," in *Proceedings of the 16th IFAC World Congress*, p. 6, Praha, Czech Republic, July 2005, CDROM, paper no 02643.

[4] S. Klein, *Identification of discrete event systems for fault detection purposes*, Ph.D. thesis, Ecole Normale Supérieure de Cachan, Paris, France, October 2005.

[5] A. Giua and C. Seatzu, "Identification of free-labeled Petri nets via integer programming," in *Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference (CDC-ECC '05)*, pp. 7639–7644, Seville, Spain, December 2005.

[6] M. P. Cabasino, A. Giua, and C. Seatzu, "Identification of deterministic Petri nets," in *Proceedings of the 8th International Workshop on Discrete Event Systems (WODES '06)*, pp. 325–331, Ann Arbor, Mich, USA, July 2006.

[7] M. Dotoli, M. P. Fanti, and A. M. Mangini, "Real time identification of discrete event systems using Petri nets," *Automatica*, vol. 44, no. 5, pp. 1209–1219, 2008.

[8] L. G. Valiant, "A theory of the learnable," *Communications of the ACM*, vol. 27, no. 11, pp. 1134–1142, 1984.
[9] D. Angluin, "Queries and concept learning," *Machine Learning*, vol. 2, no. 4, pp. 319–342, 1988.
[10] T. L. Booth, *Sequential Machines and Automata Theory*, John Wiley & Sons, New York, NY, USA, 1967.
[11] J. Kella, "Sequential machine identification," *IEEE Transactions on Computers*, vol. 20, no. 3, pp. 332–338, 1971.
[12] A. W. Biermann and J. A. Feldman, "On the synthesis of finite-state machines from samples of their behavior," *IEEE Transactions on Computers*, vol. 21, no. 6, pp. 592–597, 1972.
[13] L. P. J. Veelenturf, "Inference of sequential machines from sample computations," *IEEE Transactions on Computers*, vol. 27, no. 2, pp. 167–170, 1978.
[14] L. P. J. Veelenturf, "An Automata-theoretical approach to developing learning neural networks," *Cybernetics and Systems*, vol. 12, no. 1-2, pp. 179–202, 1981.
[15] M. Richetin, M. Naranjo, and P. Luneau, "Identification of automata by sequential learning," *Pattern Recognition Letters*, vol. 2, no. 6, pp. 379–385, 1984.
[16] L. S. Levy and A. K. Joshi, "Skeletal structural descriptions," *Information and Control*, vol. 39, no. 2, pp. 192–211, 1978.
[17] H. Ishizaka, "Polynomial time learnability of simple deterministic languages," *Machine Learning*, vol. 5, no. 2, pp. 151–164, 1990.
[18] Y. Takada, "Grammatical inference for even linear languages based on control sets," *Information Processing Letters*, vol. 28, no. 4, pp. 193–199, 1988.
[19] K. Hiraishi, "Construction of a class of safe Petri nets by presenting firing sequences," in *Proceedings of the 13th International Conference on Application and Theory of Petri Nets*, vol. 616 of *Lectures Notes in Computer Sciences*, pp. 244–262, Sheffield, UK, June 1992.
[20] M. E. Meda-Campaña, "DES identification using interpreted Petri nets," in *Proceedings of the International Symposium on Robotics and Automation*, pp. 353–357, Saltillo, Mexico, December 1998.
[21] M. E. Meda-Campaña, A. Ramírez-Treviño, and E. López-Mellado, "Asymptotic identification of discrete event systems," in *Proceedings of the 39th IEEE Confernce on Decision and Control*, pp. 2266–2271, Sydney, Australia, December 2000.
[22] M. E. Meda-Campaña and E. López-Mellado, "A passive method for on-line identification of discrete event systems," in *Proceedings of the 40th IEEE Conference on Decision and Control (CDC '01)*, pp. 4990–4995, Orlando, Fla, USA, December 2001.
[23] M. E. Meda-Campaña and E. López-Mellado, "Incremental synthesis of Petri nets models for identification of discrete event systems," in *Proceedings of the 41st IEEE Conference on Decision and Control*, pp. 805–810, Las Vegas, Nev, USA, December 2002.
[24] M. E. Meda-Campaña and E. López-Mellado, "Required event sequences for identification of discrete event s ystems," in *Proceedings of the 42nd IEEE Conference on Decision and Control (CDC '03)*, vol. 4, pp. 3778–3783, Maui, Hawaii, USA, December 2003.
[25] M. E. Meda-Campaña and E. López-Mellado, "Identification of concurrent discrete event systems using Petri nets," in *Proceedings of the 17th IMACS World Congress on Computational and Applied Mathematics*, pp. 11–15, Paris, France, July 2005.
[26] M. Roth, J.-J. Lesage, and L. Litz, "Distributed identification of concurrent discrete event systems for fault detection purposes," in *Proceedings of the European Control Conference (ECC '09)*, Budapest, Hungary, August 2009.
[27] M. Roth, J.-J. Lesage, and L. Litz, "Black-box identification of discrete event systems with optimal partitioning of concurrent subsystems," in *Proceedings of the American Control Conference (ACC '10)*, Baltimore, Md, USA, June 2010.
[28] M. Roth, J.-J. Lesage, and L. Litz, "An FDI method for manufacturing systems based on an identified model," in *Proceedings of the 13h IFAC Symposium on Information Control Problems in Manufacturing (INCO '09)*, pp. 1389–1394, Moscow, Russia, June 2009.
[29] M. P. Cabasino, A. Giua, and C. Seatzu, "Computational complexity analysis of a Petri net identification procedure," in *Proceedings of the International Symposium on Nonlinear Theory and Its Applications (NOLTA '08)*, Bologna, Italy, September 2006.
[30] M. P. Cabasino, A. Giua, and C. Seatzu, "Identification of unbounded Petri nets from their coverability graph," in *Proceedings of the 45th IEEE Conference on Decision and Control (CDC '06)*, pp. 434–440, San Diego, Calif, USA, December 2006.
[31] M. Dotoli, M. P. Fanti, and A. M. Mangini, "An optimization approach for identification of Petri Nets," in *Proceedings of the 8th International Workshop on Discrete Event Systems (WODES '06)*, pp. 332–337, Ann Arbor, Mich, USA, July 2006.

[32] M. Dotoli, M. P. Fanti, and A. M. Mangini, "On-line identification of discrete event systems: a case study," in *Proceedings of the IEEE International Conference on Automation Science and Engineering (CASE '07)*, pp. 405–410, Shangai, China, October 2006.

[33] M. Dotoli, M. P. Fanti, and A. M. Mangini, "On line identification of discrete event systems via Petri nets: an application to monitor specification," in *Proceedings of the 3rd Annual IEEE International Conference on Automation Science and Engineering (CASE '07)*, pp. 893–898, Scottsdale, Ariz, USA, September 2007.

[34] M. P. Fanti and C. Seatzu, "Fault diagnosis and identification of discrete event systems using Petri nets," in *Proceedings of the 9th International Workshop on Discrete Event Systems (WODES '08)*, pp. 432–435, Göteborg, Sweden, May 2008.

[35] T. Moor, J. Raisch, and S. O'Young, "Supervisory control of hybrid systems via l-complete approximations," in *Proceedings of the 4th IEEE Workshop on Discrete Event Systems (WODES '98)*, pp. 426–431, Cagliari, Italy, August 1998.

[36] A.P. Estrada-Vargas, E. López-Mellado, and J.-J. Lesage, "Off-line identification of concurrent discrete event systems exhibiting cyclic behaviour," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, pp. 181–186, San Antonio Tex, USA, October 2009.