Off-line Identification of Concurrent Discrete Event Systems Exhibiting Cyclic Behaviour

Ana Paula Estrada-Vargas, E. López-Mellado, Jean-Jacques Lesage

▶ **To cite this version:**

Ana Paula Estrada-Vargas, E. López-Mellado, Jean-Jacques Lesage. Off-line Identification of Concurrent Discrete Event Systems Exhibiting Cyclic Behaviour. IEEE 2009 International Conference on Systems, Man, and Cybernetics, Oct 2009, San Antonio, Texas, United States. pp. 187-192. hal-00425187

HAL Id: hal-00425187
https://hal.science/hal-00425187

Submitted on 20 Oct 2009

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Off-line Identification of Concurrent Discrete Event Systems Exhibiting Cyclic Behaviour

Ana P. Estrada-Vargas,
E. López-Mellado *Member IEEE*
CINVESTAV Unidad Guadalajara
Av. Científica 1145, Col. El Bajío
45015 Zapopan, Mexico
{aestrada, elopez}@gdl.cinvestav.mx

Jean-Jacques Lesage *Member IEEE*
LURPA Ecole Normale Supérieure de Cachan
61, av du Président Wilson
94235 Cachan Cedex, France
Jean-Jacques.lesage@lurpa.ens-cachan.fr

***Abstract*— This paper presents a method for the identification of concurrent Discrete Event Systems (DES) from input-output sequences representing the observed behavior. The proposed off-line technique yields an input-output model expressed as an Interpreted Petri net (IPN), which represents exactly the language than that generated by the observed system, which may include cyclic sequences. First, a sample of input-output vectors words are processed for obtaining sequences of output changes called events; then sequences of κ-length event traces are built and represented by an IPN model composed by non measurable places. Then measurable places are added; they are related to transitions representing pertinent output changes. Finally inputs are associated to transitions and implicit non measurable places are removed.**

***Keywords*—Discrete event systems, Automated model identification, Interpreted Petri nets**

I. INTRODUCTION

Automated building of discrete event models from external observation of system behavior has been addressed as a problem of system identification. The problem has interesting applications such as reverse engineering for (partially) unknown systems, fault diagnosis or system verification.

Previous results on the matter have been proposed as a solution to the problem of grammatical inference. In [5] a finite automaton (FA) is built from positive samples of accepted words. Later several methods for obtaining Mealy [8] [16] and Moore [1] [17] machines have been proposed. Also building of context free grammars has been studied [11], [15], [7].

The identification of Petri net (PN) models has been proposed for coping with more complex systems exhibiting concurrent behavior. In [6] an algorithm for constructing Petri net models is presented. First, the language of the target system is identified in the form of deterministic FA (DFA). Then, the algorithm obtains from the DFA the structure of a PN that accepts the obtained language.

Three different approaches have been adopted in recent publications addressing the problem of identification of DES. The incremental synthesis approach proposed by Meda et al., deals with unknown partially measurable DES exhibiting cyclic behavior; in [12] an identification method based on the least square estimator is presented. Then in [13][14] several algorithms have been proposed allowing the on-line identification of concurrent DES. Although the techniques are efficient, the obtained models may represent more sequences than those observed.

Other technique oriented to fault diagnosis is presented in [10]; it obtains non deterministic FA representing the same observed behavior, which is represented as a set of input/output sequences. The algorithms operate off-line efficiently.

The off-line techniques based on integer linear programming (ILP) approach lead to free-labeled PN models representing exactly the observed behavior [4]. However both the ILP problem statement from samples and their processing have exponential complexity. This approach is being applied to other PN classes in [2] [3].

In this paper a method for the identification of Petri net models from observed input-output sequences representing behavior of concurrent DES is presented. It gathers and extends some strategies form [13] and [10] and focuses on off-line case. Because of the inherent capacity of Interpreted Petri Nets (IPN) models to represent inputs and outputs of DES systems, internal states, non-controllable transitions and parallelism, the proposed method yields an IPN model which represents exactly the same language of length κ+1 than that generated by the system without taking into account information a-priori about the system other than its input and output signals. In the first stage, a sample of output vectors words are processed for obtaining sequences of output changes called events; then sequences of κ-length event traces are considered allowing building an underlying IPN of non measurable places. The IPN orders the events by relating the measurable places according to pertinent output changes. Finally, the inputs are added and the model is simplified by eliminating implicit non-measurable places.

The remainder of this paper is organized as follows. Section 2 overviews basic definitions on PN and IPN. Section 3 presents the proposed identification technique. Finally concluding remarks are given.

II. PETRI NET MODELS

This section presents the basic concepts and notation of PN and IPN used in this paper.

Definition 1: An ordinary Petri Net structure G is a bipartite digraph represented by the 4-tuple G=(P,T,I,O) where:

- $P = \{p_1,p_2,...,p_n\}$ and $T = \{t_1,t_2,...,t_m\}$ are finite sets of vertices named places and transitions respectively,
- $I\ (O) : P \times T \rightarrow \{0, 1\}$is a function representing the arcs going from places to transitions (transitions to places).

Pictorially, places are represented by circles, transitions are represented by rectangles, and arcs are depicted as arrows. The symbol ${}^\bullet t_j$ ($t_j^\bullet$) denotes the set of all places p_i such that $I(p_i,t_j)\neq 0$ ($O(p_i,t_j)\neq 0$). Such places are called input (output) places of t_j. Analogously, ${}^\bullet p_i$ ($p_i^\bullet$) denotes the set of all transitions t_j such that $O(p_i,t_j)\neq 0$ ($I(p_i,t_j)\neq 0$). Such transitions are called input (output) transitions of p_i.

The incidence matrix of G is $C = C^+ - C^-$, where $C^- = [c_{ij}^-]$; $c_{ij}^- = I(p_i,t_j)$; and $C^+ = [c_{ij}^+]$; $c_{ij}^+ = O(p_i,t_j)$ are the pre-incidence and post-incidence matrices respectively.

A marking function M: P$\rightarrow \mathbb{Z}^+$ represents the number of tokens (depicted as dots) residing inside each place. The marking of a PN with n places is usually expressed as an n-entry vector. $\mathbb{Z}^+$ is the set of nonnegative integers.

Definition 2: A Petri Net system or Petri Net (PN) is the pair $N = (G,M_0)$, where G is a *PN* structure and M_0 is an initial marking.

In a *PN* system, a transition t_j is *enabled* at marking M_k if $\forall p_i \in P$, $M_k(p_i) \geq I(p_i,t_j)$; an enabled transition t_j can be fired reaching a new marking M_{k+1} which can be computed as $M_{k+1} = M_k + Cv_k$, where $v_k(i)$=0, i≠j, v_k(j)=1, this equation is called the *PN* state equation. The reachability set of a *PN* is the set of all possible reachable marking from M_0 firing only enabled transitions; this set is denoted by $R(G,M_0)$. This work uses IPN, an extension to PN that allows associating input and output signals to PN models.

Definition 3: An IPN *(Q, M_0)* is a net structure $Q = (G,\Sigma,\Phi,\lambda,\varphi)$ with an initial marking M_0.

- G is a *PN* structure
- $\Sigma = \{\alpha_1, \alpha_2, ..., \alpha_r\}$ is the alphabet of input symbols α_i.
- $\Phi = \{\phi_1, \phi_2,..., \phi_q\}$ is the alphabet of output symbols ϕ_i.
- $\lambda : T \rightarrow \Sigma \cup \{\varepsilon\}$ is a labelling function of transitions with the following constraint: $\forall t_j,t_k \in T$, j≠k, if $\forall p_i\ I(p_i,t_j) = I(p_i,t_k) \neq 0$ and both $\lambda(t_j) \neq \varepsilon$, $\lambda(t_k) \neq \varepsilon$, then $\lambda(t_j) \neq \lambda(t_k)$; ε represents a system internal event externally uncontrollable
- $\varphi : R(Q,M_0) \rightarrow (\mathbb{Z}^+)^q$ is an output function, that associates to each marking in $R(Q,M_0)$ a q-entry output vector; q is the number of outputs. φ is represented by a $q \times n$ matrix, in which if the output symbol ϕ_i is present (turned on) every time that $M(p_j)\geq 1$, then $\varphi(i,j)$=1, otherwise $\varphi(i,j)$=0.

If a transition $t_j \in T$ of an *IPN* is enabled at marking M_k, there are two possibilities: a) If $\lambda(t_j) = \alpha_i \neq \varepsilon$ is provided to the system then t_j can be fired. b) If $\lambda(t_j) = \varepsilon$ and t_j is enabled then t_j must be fired. When an enabled transition t_j is fired in a marking M_k, then a new marking M_{k+1} is reached. This behaviour is represented as $M_k \xrightarrow{t_j} M_{k+1}$; M_{k+1} can be computed using the state equation:

$$M_{k+1} = M_k + Cv_k \qquad (1)$$
$$y_k = \varphi M_k$$

where C and v_k are defined as in *PN* and $y_k \in (\mathbb{Z}^+)^q$ is the k-th output vector (or marking projection) of the *IPN*.

According to functions λ and φ, transitions and places of an *IPN* (Q,M_0) can be classified as follows.

Definition 4: If $\lambda(t_i)\neq\varepsilon$ the transition t_i is said to be controllable. Otherwise it is uncontrollable. A place $p_i \in P$ is said to be measurable if the i-th column vector of φ is not null, i.e. $\varphi(\bullet,i)\neq 0$. Otherwise it is non-measurable. $P = P^m \cup P^{\mu}$ where P^m is the set of measurable places and P^{μ} is the set of non-measurable places.

Definition 5: A firing transition sequence of an IPN (Q, M_0) is a sequence $\sigma = t_it_j...t_k$ such that $M_i \xrightarrow{t_i} M_{i+1} \xrightarrow{t_j} ... M_{i+|\sigma|-1} \xrightarrow{t_k} M_{i+|\sigma|}$, where $|\sigma|$ is the length of the sequence σ. The set of all firing sequences is called the language of (Q, M_0) and it is denoted as $\pounds(Q,M_0) = \{\sigma | \sigma = t_it_j \ ...t_k \ and \ M_i \xrightarrow{t_i} M_{i+1} \xrightarrow{t_j} ... M_{i+|\sigma|-1} \xrightarrow{t_k} M_{i+|\sigma|}\}$.

Definition 6: The *input language* of an IPN (Q, M_0) is defined as: $\pounds_{in}(Q, M_0) = \{ \lambda(t_i)\lambda(t_{i+1})... \lambda(t_{i+|\sigma|-1})\lambda(t_{i+|\sigma|}) | M_i \xrightarrow{t_i} M_{i+1} \xrightarrow{t_j} ... M_{i+|\sigma|-1} \xrightarrow{t_k} M_{i+|\sigma|},\ t_it_j...t_k \in \pounds(Q,M_0)\}$. The *output language* of an IPN *(Q, M_0)* is defined as: $\pounds_{out}(Q, M_0)=\{ \varphi(M_i)\varphi(M_{i+1})... \varphi(M_{i+|\sigma|-1})\varphi(M_{i+|\sigma|}) | M_i \xrightarrow{t_i} M_{i+1} \xrightarrow{t_j} ... M_{i+|\sigma|-1} \xrightarrow{t_k} M_{i+|\sigma|},\ t_it_j...t_k \in \pounds(Q,M_0)\}$.

Definition 7: The output language of length l of an IPN (Q, M_0) is defined as:
$\pounds_{out}^l(Q,M_0) = \{w | w \in \pounds_{out}(Q,M_0) and\ |w| \leq l\}$.

III. IDENTIFICATION METHOD

A. Problem statement

An identification procedure of a DES builds a mathematical model that represents its behaviour from the measured evolution of the DES' inputs and outputs [12][10][3]. Below the addressed identification problem is stated.

Consider a *DES S* with binary outputs revealing part of its internal states in which the outputs may evolve concurrently. The input data to the identification procedure is a set of output words Γ(S) describing behaviour and the sequences of input symbols in Σ that yield the output sequences.

Definition 8: The set of observed output words of a DES S is $\Gamma(S) = \{w_1, w_2, ...\}$ such that $w_i = (w_i(1), w_i(2), ... w_i(|w_i|))$, where $w_i(j)$ is the j-th observed vector in sequence w_i and $|w_i|$ is the length of the output word w_i.

Definition 9: The observed output language of a DES S is defined as $\mathcal{L}(S) = \{w_i(j+1)w_i(j+2) ... w_i(j+l) | w_i \in \Gamma(S) \text{ and } j + l \leq |w_i|\}$.

Definition 10: The output language of length *l* of a DES *S* is defined as $\mathcal{L}^l(S) = \{w | w \in \mathcal{L}(S) \text{ and } |w| \leq l\}$.

Given a set of output words $\Gamma(S)$, the input signals Σ of a DES *S*, and an accuracy parameter κ, the aim of the identification process is to obtain a safe *IPN* model (Q, M_0) such that $\mathcal{L}^{\kappa}_{out}(Q, M_0) = \mathcal{L}^{\kappa}(S)$. The parameter κ is used to adjust the accuracy of the identified model, similarly as proposed in [9]. It is assumed that for every output vector change an input signal $\alpha_i \in \Sigma$ is recorded; otherwise ε is taken.

B. General approach

The method for identification consists of several stages that build systematically a safe IPN which represents exactly the sampled output language of length κ +1 of the DES.

From the output vector words, the event sequences are computed and then sequences of event substrings of length κ are built. Then every substring is associated to a transition of a PN, which describes the causal relationship between event substrings through paths including non-measurable places. Finally, the output changes provoked by events are described by marking changes in measurable places and then related to pertinent transitions in the PN. Notice that the number of non-observable places is not predefined.

C. Sample processing

1) Event sequences

As stated before, the data obtained from the system to be identified is a set of sequences of output vectors $w_1, w_2, ...$, such that $w_i = (w_i(1), w_i(2), ...)$, where $w_i(j)$ refers to the j-th observed vector in sequence w_i. Sequences may have different length. From these sequences, strings of observed events are first computed.

Definition 11. An observed event $\tau_i(j)$ is the variation between two consecutive output vectors $w_i(j), w_i(j+1)$; it is computed as $\tau_i(j) = w_i(j+1) - w_i(j)$.

Then for every sequence w_i, a sequence of observed events $\tau_i = \tau_i(1)\tau_i(2) \ldots \tau_i(|\tau_i|)$ is obtained. During the process, if the difference has not been observed before, a new event e_k is created ($\tau_i(k) = e_k$).

The following example illustrates that the method copes also with DES exhibiting cyclic behaviour. It has five output signals, Φ = {A, B, C, D, E}, and five input signals Σ = {a, b, c, d, e}. Two output sequences have been observed:

$$\begin{bmatrix} A \\ B \\ C \\ D \\ E \end{bmatrix} \quad w_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}\begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, w_2 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

The sequences τ_i of the detected events e_j associated to output changes are obtained:

$$w_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}\overset{e_1}{\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}}\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}\overset{e_2}{\begin{bmatrix} -1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}}\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}\overset{e_3}{\begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}}\overset{e_4}{\begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}}\begin{bmatrix} 0 \\ 0 \\ 1 \\ -1 \\ 0 \end{bmatrix}\begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}\overset{e_5}{\begin{bmatrix} 0 \\ -1 \\ 0 \\ 0 \\ 1 \end{bmatrix}}\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}\overset{e_6}{\begin{bmatrix} 0 \\ 0 \\ -1 \\ 0 \\ -1 \end{bmatrix}}\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\tau_1 = e_1 e_2 e_3 e_4 e_5 e_6$$

$$w_2 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}\overset{e_1}{\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}}\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}\overset{e_2}{\begin{bmatrix} -1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}}\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}\overset{e_3}{\begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}}\begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}\overset{e_5}{\begin{bmatrix} 0 \\ -1 \\ 0 \\ 0 \\ 1 \end{bmatrix}}\begin{bmatrix} 0 \\ 0 \\ 1 \\ -1 \\ 0 \end{bmatrix}\overset{e_4}{\begin{bmatrix} 0 \\ 0 \\ 1 \\ -1 \\ 0 \end{bmatrix}}\begin{bmatrix} 0 \\ 1 \\ 0 \\ -1 \\ 0 \end{bmatrix}\overset{e_6}{\begin{bmatrix} 0 \\ 0 \\ -1 \\ 0 \\ -1 \end{bmatrix}}\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\tau_2 = e_1 e_2 e_3 e_5 e_4 e_6$$

Additionally consider that the input signals measured at the output changes are: a, ε, b, c, d, e, for e1, e2, e3, e4, e5, e6, respectively.

2) Sequences of κ-length event traces

In order to introduce the accuracy parameter κ for identification, from every sequence of events $\tau_i = \tau_i(1)\tau_i(2) \ldots \tau_i(|\tau_i|)$ we compute sequences of event traces $\tau_i^{\kappa} = \tau_i^{\kappa}(1), \tau_i^{\kappa}(2), \ldots \tau_i^{\kappa}(|\tau_i|)$ such that every $\tau_i^{\kappa}(j)$ is a substring of length κ of τ_i that finishes with $\tau_i(j)$. For the first $\kappa - 1$ elements of the trace sequence the event ε is used.

Following with the previous example, the sequences of traces using $\kappa = 2$ are:

$\tau_1^2 = \varepsilon e_1, e_1 e_2, e_2 e_3, e_3 e_4, e_4 e_5, e_5 e_6$ for τ_1

$\tau_2^2 = \varepsilon e_1, e_1 e_2, e_2 e_3, e_3 e_5, e_5 e_4, e_4 e_6$ for τ_2

D. Building the basic structure

1) Representing event traces

Once the sequences of event traces have been obtained, every trace $\tau_i^{\kappa}(j)$ is related to a transition in the IPN through a function γ:T→$\{\tau_i^{\kappa}(j)\}$; the firing of a transition implies that κ consecutive events related to such a transition have been observed. In order to preserve firing order between transitions, dependencies are created between them and associated with an observed marking through the function $\mu: P^u \rightarrow \{\varphi M_i | M_i \in R(N, M_0)\}$, which relates every non-measurable place with an observed marking, such that every transition has only one input place and one output place ($\forall t_r \in T$, $^{\bullet}t_r = t_r^{\bullet} = 1$); when an event trace $\tau_i^{\kappa}(j)$ is found again in a τ_i^{κ} the associated dependency must be used if it leads to the same observed marking. Let e_j be the last event in the trace $\tau_i^{\kappa}(j)$; the associated transition will be denoted as $t_r^{e_j}$ (more than one transition may have associated the same e_j). This strategy can be systematically performed following the next procedure.

Algorithm 1. Building the basic IPN structure

Input: The set $\Gamma^{\kappa} = \{\tau_i^{\kappa}\}$

Output: An IPN model G composed by non-measurable places

Step1. $T \leftarrow \emptyset; ET \leftarrow \emptyset;\ P \leftarrow \{p_{ini}\}; M_0(p_{ini}) \leftarrow 1;$ $\mu(p_{ini}) \leftarrow \tau_i(1)$. *//Create an initial empty set of transitions, an initial empty event trace set, and an initial set of places with only a marked place p_{ini} associated with the first observed vector.*

Step 2. $\forall \tau_i^{\kappa} \in \Gamma^{\kappa}$:

2.1 $current \leftarrow p_{ini}$ //Take p_{ini} as *current*.

2.2 $\forall \tau_i^{\kappa}(j)$ belonging to $\tau_i^{\kappa}, 1 \leq j \leq |\tau_i^{\kappa}|$. *//For every event trace*:

2.2.1 If $\tau_i^{\kappa}(j) \notin ET$ *//If it is new*

then $ET \leftarrow ET \cup \{\tau_i^{\kappa}(j)\}; T \leftarrow T \cup \{t_r^{e_j}\}; \gamma(t_r^{e_j}) \leftarrow \tau_i^{\kappa}(j);$ $\forall p_a \in P,\ I(p_a, t_r^{e_j}) \leftarrow 0; O(p_a, t_r^{e_j}) \leftarrow 0$ *//create a transition* $t_r^{e_j}$ *to represent the trace;* $I(current, t_r^{e_j}) \leftarrow 1$ *//create an arc from current to* $t_r^{e_j}$

If $j = |\tau_i^{\kappa}|$ *//If it is the last trace of the sequence,*

then $O(p_{ini}, t_r^{e_j}) \leftarrow 1$ *//add an arc from its transition to* p_{ini}.

else $P \leftarrow P \cup \{p_{out}\}; \forall t_b \in T, I(p_{out}, t_b) \leftarrow 0,$ $(p_{out}, t_b) \leftarrow 0; \mu(p_{out}) \leftarrow \mu(current) + e_j$ *//create a new place* p_{out}; $O(p_{out}, t_r^{e_j}) \leftarrow 1$ *//create an arc from its transition to* p_{out}; $current \leftarrow p_{out}$ *//take such place as current.*

2.2.2 If $\tau_i^{\kappa}(j) \in ET$*//If it is not new*

then find $t_r^{e_j} \in T | \gamma(t_r^{e_j}) = \tau_j^{\kappa}(j)$ *//find the transitions which represent the sequence.* For every one of those transitions, let be $p_{in} = {}^{\bullet}t_r^{e_j}$ *//take the input place of the transition as* p_{in};

If there is any p_{in} such that $\mu(p_{in}) = \mu(current)$

then $\forall t_b \in T,\ I(p_{in}, t_b) \leftarrow \overrightarrow{\oplus}(I(p_{in}, t_b), I(current, t_b));$ $\forall t_b \in T,\ O(p_{in}, t_b) \leftarrow \overrightarrow{\oplus}(O(p_{in}, t_b), O(current, t_b));$ where $\overrightarrow{\oplus}$ is a vector bitwise or operation; $P \leftarrow P \backslash \{current\}$ *//merge current place with such an input place;* $current \leftarrow (t_r^{e_j})^{\bullet}$*//take the output place of the transition as current.*

else go to step 2.2.1, and take $\tau_i^{\kappa}(j)$ as new.

If $j = |\tau_i^{\kappa}|$ *//If it is the last trace of the sequence,*

then $\forall t_b \in T,\ I(p_{ini}, t_b) \leftarrow \overrightarrow{\oplus}(I(p_{ini}, t_b), I(current, t_b));$ $\forall t_b \in T,\ O(p_{ini}, t_b) \leftarrow \overrightarrow{\oplus}(O(p_{ini}, t_b), O(current, t_b));$ $P \leftarrow P \backslash \{current\}$ *//merge current with the initial place.*

2.3 $\forall t_r^{e_j} \in T, \lambda'(t_r^{e_j}) \leftarrow e_j$ *//associate every transition* t_r *to the last symbol* e_j *of the event trace it represents.*

Since search operation has linear complexity and the algorithm implies the addition of a transition for every computed trace that has not been yet observed, then Algorithm1 is executed in polynomial time on the number and maximum length of observed output sequences.

Property 1. The IPN G built through algorithm 1 represents all and only all the sequences (sub-sequences) in Γ^{κ}

Proof. By construction, every sequence τ_i^{κ} is represented in G by a circuit starting from p_{ini} including the sequence of $t_r^{e_j}$; this circuit represents also the sequence τ_i of events. Furthermore, the reuse of computed transitions having associated the same event traces, during the processing of subsequent τ_i^{κ}, is done only when common paths of length κ are built, which does not introduce other sequences. ■

Using the previous algorithm the obtained IPN corresponding to the two sequences of event traces of the example is showed in figure 1.

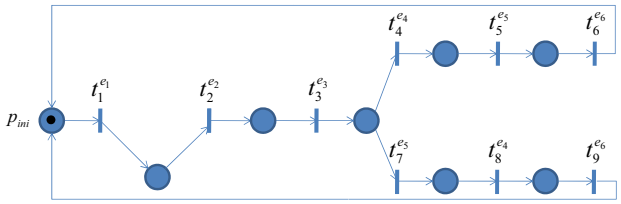

Figure 1. Basic model describing the sequences of event traces

2) Simplifying the basic structure

Additional node merging operations can be performed on the basic structure in order to obtain an equivalent trace model. Now we can take into account the event e_j associated to transitions. Consider the following transformation rules.

Algorithm 2. Simplifying basic structure.

Input: G

Output: G': an equivalent IPN

Apply the following rule on the initial place

Rule 1: $\forall t_a^{e_j}, t_b^{e_j} \in p_{ini}{}^{\bullet} | a \neq b$ *//If* p_{ini} *has several output transitions with the same associated event*, then merge $t_a^{e_j}$ and $t_b^{e_j}$ and their output places accordingly. This test is performed iteratively on the new obtained place, until there are no many output transitions of such a place sharing the same event.

Rule 2: $\forall t_a^{e_j}, t_b^{e_j} \in {}^{\bullet}p_{ini} | a \neq b$ *//If* p_{ini} *has more than one input transition with the same associated event*, then merge $t_a^{e_j}$ and $t_b^{e_j}$ and their respective input places accordingly.

Since merging has linear order and rules cannot be applied more times than the number of places in the net, Algorithm 2 is executed in polynomial time on the number of repeated transitions in the same place.

Property 2. G' preserves the sequences in G.

Proof. Let $Lf_{p_i} = \{\lambda'(t_1)\lambda'(t_2) \ldots \lambda'(t_r) |\ t_1 \in p_i{}^{\bullet}, t_{i+1} \in (t_i{}^{\bullet})^{\bullet}\}$ be the set of observable sequences from place p_i. Consider $t_a^{e_j}, t_b^{e_j} \in p_{ini}{}^{\bullet} | a \neq b$. Before applying rule 1, $Lf_{p_{ini}} = e_j L_{p_a} \cup e_j L_{p_b} \cup (\cup \lambda'(t_i) L_{p_i})$, with $p_a = (t_a^{e_j})^{\bullet}$, $p_b = (t_b^{e_j})^{\bullet}$; $t_i \in p_{ini}{}^{\bullet} | t_i \neq t_a^{e_j}, t_i \neq t_b^{e_j}$, $p_i = (t_i)^{\bullet}$. After applying rule 1, $Lf_{p_{ini}} = e_j(L_{p_a} \cup L_{p_b}) \cup (\cup \lambda'(t_i) L_{p_i})$. Thus $Lf_{p_i} = Lf_{p_{ini}}$, and then language of G is not changed by the application of rule 1. A similar reasoning can be done for rule 2. ■

In the example the application of the rules lead to the model of figure 2. Since the initial place has two input transitions associated to e_6, they can merge and their input places.

3) Concurrent transitions

Other transformations may be performed when there are transitions that appear in the sequences in different order

describing their interleaved firing; this behaviour is exhibited by concurrent transitions. The analysis can be performed on a model component comprised between two transitions relied by several paths containing the concurrent transitions. If there are *m!* paths, we can explore if there exists *m* different transitions in the paths and every sequence is a permutation from each other. When it is verified, the subnet can be transformed into a concurrent component of G' preserving the same behaviour.

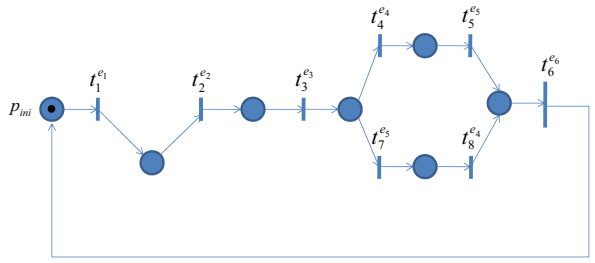

Figure 2. Model after merging

In figure 2 notice that between the transition associated to e_3 and the new transition associated to e_6 there are paths with all possible permutations of e_4 and e_5; then, we can transform this into a concurrent component and we obtain the net showed in figure 3. Notice that this model preserves the same event sequences of the previous one.

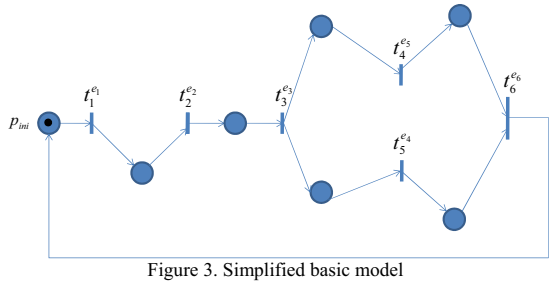

Figure 3. Simplified basic model

The simplification by analysis of concurrency is not strictly necessary for representing the event sequences; however the equivalent model with concurrent transitions may be simpler. Although the analysis could be inefficient when the number of paths in the subnet is large, usually this number is reduced.

E. Adding outputs and inputs

1) Representing outputs changes

Once the event sequences are represented in the basic model, it must be completed by adding the output changes represented by the events and their respective inputs. Recall that events are vectors computed from the difference of consecutive output vectors; thus e_j relates measurable places representing the outputs yielding the incidence matrix corresponding to measurable places. The inputs are straightforward included in the model. This procedure is detailed below.

Algorithm 3. Representing outputs changes

Input: G', $\{e_i(j)\}$

Output; Q : the IPN including measurable places and inputs

Step 1. $P \leftarrow P \cup \{p_1, p_2, \ldots, p_q\}$. *Create q measurable places for every one of the components in the output vectors.*

Step 2. $\forall t_r^{e_j} \in T$: if $e_j(i) = -1$ then $I(p_i, t_r^{e_j}) = 1$ and $O(p_i, t_r^{e_j}) = 0$, if $e_j(i) = 1$ then $I(p_i, t_r^{e_j}) = 0$ and $O(p_i, t_r^{e_j}) = 1$, if $e_j(i) = 0$ then $I(p_i, t_r^{e_j}) = 0$ and $O(p_i, t_r^{e_j}) = 0$ *(add arcs to and from the measurable places according to its associated vector event* e_j*).*

Step 3. If component *i* of vector $w_j(1)$ is 1 then $M_0(p_i) = 1$, otherwise $M_0(p_i) = 0$ *(put tokens in the measurable places to represent the first output vector).*

Step 4. Associate to each $t_r^{e_j}$ the input symbol registered at the detection of e_j.

The net with measurable places and inputs for the example is showed in figure 4.

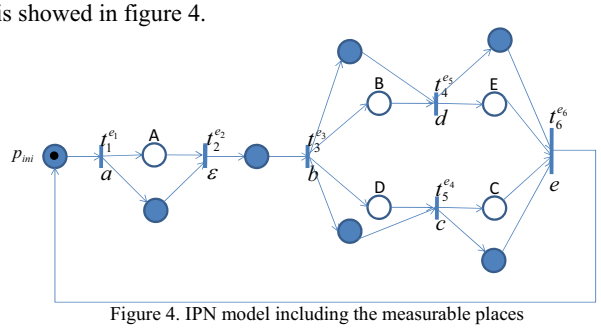

Figure 4. IPN model including the measurable places

2) Model simplifying

Implicit non-measurable can be removed: if there is a non-measurable place whose input and output transitions are exactly the same than any measurable place, then delete such a place and its input and output arcs.

The final model for the illustrative example is showed in figure 5; the incidence matrix, output function and initial marking of this IPN are given below. The associated inputs for transitions is given by the lambda function: $\lambda(t_1)$=a, $\lambda(t_2)$= ε, $\lambda(t_3)$=b, $\lambda(t_4)$=d, $\lambda(t_5)$=c, $\lambda(t_6)$=e.

$$C = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 1 \\ 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix}, \varphi = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}, M_0 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

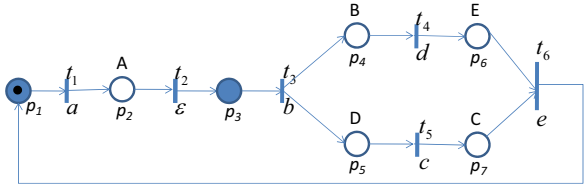

Figure 5. Simplified IPN model

Since every one of the transitions in the net actually represents a sequence of events of length κ, the output language of length $\kappa + 1$ of the net is equal to the observed output language. Even, for the illustrative example, the output language of the IPN is equal to the observed output language, i.e. only the observed cyclic output sequences are represented by the evolution of the net.

F. Procedure

Now, we can summarise the stages of the method for *IPN* model identification.

Algorithm 4. Building an IPN model

Inputs: A DES and an accuracy parameter κ
Output: (Q,M_0): an IPN model

1. Obtain the input symbols and the cyclic sequences of observed output vectors.
2. Compute the event sequences from the observed vectors.
3. For every sequence of events, create traces of length κ.
4. Create the non-observable behaviour of the IPN (Algorithm 1) and simplify it (algorithm 2).
5. Complete the Petri net adding the observed behaviour and delete implicit places (algorithm 3).

Proposition 1. For a DES *S* that holds the hypothesis given in section III.A and an identification parameter κ, Algorithm 4 yields an IPN model (Q,M_0) which represents exactly $\mathcal{L}^{\kappa+1}(S)$.

Proof. Since the deletion of implicit places does not alter $\pounds_{out}^{\kappa}(Q, M_0)$, we make the proof with the model obtained before this procedure. The firing of a transition t in the system is not affected by the addition of arcs to, and from t, since these arcs were computed from differences of vectors in $\Gamma(S)$. Then, also in this model, every event sequence σ of length less or equal than κ belongs to the language of the net iff it was observed.

The sequences of transitions of length less or equal than κ that can be fired lead to markings in the measurable places that also belong to $\Gamma(S)$ (since the marking change provoked in the measurable places was obtained from the difference of observed vectors). Then, we have that sequences of observed output vectors of length less or equal than $\kappa + 1$ correspond to sequences of marking vectors in the net and $\pounds_{out}^{\kappa+1}(Q) = \mathcal{L}^{\kappa+1}(S)$.■

IV. Conclusion

In this paper we provided a method to create an IPN model for a given DES described by a set of input-output sequences that may contain cycles representing infinite length behaviour. The proposed off-line procedure operates considering an accuracy parameter κ such that in the output language of the obtained IPN only and all observed output sequences of length $\kappa + 1$ are represented. This approach avoids including in the model more sequences than that observed.

The proposed algorithms have polynomial complexity on the input data size; they have been tested with numerous examples of diverse complexity including an industrial experimental system; in general κ must be greater than one and the obtained model allows representing complex behaviour such as concurrency and non determinism.

Current research deals with a depth study on the choice of the accuracy parameter κ and the definition of an efficient procedure for the analysis of concurrency in the basic non-measurable model.

Acknowledgement

A.P Estrada-Vargas was sponsored by CONACYT Grant No. 50312, and by ENS de Cachan International Scholarship.

References

[1] A.W. Biermann and J.A. Feldman, "On the Sysnthesis of Finite-State Machines from Samples of Their Behavior", IEEE Trans. on Computers, Vol. 21, pp. 592-597, 1972

[2] M. P. Cabasino, A. Giua, C. Seatzu, "Identification of deterministic Petri nets", Proc. of the 8th International Workshop on Discrete Event Systems, Ann Arbor, Michigan, USA, July 10-12, 2006.

[3] M. P. Fanti and C. Seatzu, "Fault diagnosis and identification of discrete event systems using Petri nets", Proc. of the 9th International Workshop on Discrete Event Systems, Göteborg, Sweden, pp. 432-435, May 28-30, 2008

[4] A. Giua and C. Seatzu, "Identification of free-labeled Petri nets via integer programming", Proc. of the 44th IEEE Conference on Decision and Control, and the European Control Conference 2005, Seville, Spain, December 12-15, 2005

[5] E.M. Gold, "Language Identification in the Limit", Information and Control, Vol. 10, pp. 447-474, 1967

[6] K. Hiraishi, "Construction of Safe Petri Nets by Presenting Firing Sequences", LNCS, 616, pp. 244-262, 1992

[7] H. Ishizaka, "Polynomial Time Learnability of Simple Deterministic Languages", Machine Learning, Vol. 5, pp. 151-164, 1990

[8] J. Kella, "Sequential Machine Identification", IEEE Trans. on Computers, Vol. 20, pp. 332-338, 1971

[9] S. Klein, L. Litz, J.-J. Lesage, "Fault detection of Discrete Event Systems using an identification approach", 16th IFAC World Congress, CDROM paper n°02643, 6 pages, Praha(CZ), July 4-8, 2005

[10] S. Klein, "Identification of Discrete Event Systems for Fault Detection Purposes", Ph. D. Thesis, Ecole Normale Supérieure de Cachan, Oct 2005

[11] L.S. Levy and A.K. Joshi, "Skeletal structural descriptions", Information and Control, Vol. 39, pp. 192-211, 1978

[12] M.E. Meda, "DES identification using Interpreted Petri nets", International Symposium on Robotics and Automation, pp 353-357, December 1998

[13] M. E. Meda-Campaña, "On-line Identification of Discrete Event Systems: Fundamentals and Algorithms for the Synthesis of Petri Net Models", Ph. D. Thesis, CINVESTAV, Unidad Guadalajara, Nov 2002

[14] M.E. Meda-Campaña, E. Lopez-Mellado, "Identification of Concurrent Discrete Event Systems using Petri Nets", 2005 IMACS Conference, Paris, France, Jul. 2005

[15] Y. Takada, "Grammatical Inference for Even Linear Languages Based on Control Sets", Information Proc. Letters, Vol. 28, pp. 193-199, 1998

[16] L.P.J. Veelenturf, "Inference of sequential machines from Sample Computations", IEEE Trans. on Computers, Vol. 27, pp. 167-170, 1978

[17] L.P.J. Veelenturf, "An Automata theoretical approach to developing learning neural networks", Cybernetics and Systems, 12, pp. 179-202, 1981