Stepwise Identification of Automated Discrete Manufacturing Systems

Ana-Paula Estrada-Vargas, Jean-Jacques Lesage, Ernesto López-Mellado

Stepwise Identification of Automated Discrete Manufacturing Systems

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

[1]CINVESTAV Unidad Guadalajara. Av. del Bosque 1145. Col. El Bajío 45015 Zapopan, Mexico
[2]LURPA Ecole Normale Supérieure de Cachan. 61, Av. du Président Wilson. 94235 Cachan Cedex, France
{elopez,aestrada}@gdl.cinvestav.mx, Jean-Jacques.lesage@lurpa.ens-cachan

Abstract

This paper deals with the identification of discrete event systems that are automated through a programmable logic controller (PLC). The behavior of the closed loop system (PLC and Plant) is observed during its operation and is represented by a single long sequence of input/output vectors. The proposed method allows building stepwise an interpreted Petri net model, which is updated when new behavior is observed. The identification strategy is composed of several polynomial time algorithms implemented in a software tool that creates and draws the IPN model

1. Introduction

Identification of discrete event systems (DES) from external observation of system behaviour has interesting applications such as reverse engineering for (partially) unknown systems, fault diagnosis, or system verification. Analogously to continuous identification techniques, identification methods for DES yield a mathematical model that represents the observed behaviour and closely approximates the actual DES' behaviour.

In recent years, the scientific community has proposed identification approaches for obtaining approximated models (Petri nets or automata) of DES whose behavior is unknown or ill-known. In the context of automated manufacturing systems, identification methods allow obtaining a first model that can be detailed using established modeling techniques and available knowledge of the system; such a model describes the controller-plant behavior during the closed-loop functioning. Three main approaches for identifying DES have been proposed in literature [1].

The incremental synthesis approach, proposed by Meda et al. in [2], [3] deals with unknown partially measurable DES exhibiting cyclic behavior. Several algorithms have been proposed allowing the on-line identification of concurrent DES from output sequences. Although the techniques are efficient, the obtained models may represent more sequences than those observed.

Another recent method [4] allows building efficiently a non deterministic finite automaton (NFA) from a set of input/output sequences, experimentally measured from DES to be identified. Under several hypotheses, the constructed NFA generates exactly the same input/output (I/O) sequences of given length than observed ones. The method was conceived for fault detection in a model-based approach [5]. Extensions to this work propose an identification method performing optimal partitioning of concurrent subsystems for distributed fault detection purposes [6].

The off-line techniques based on integer linear programming (ILP) approach lead to free-labeled Petri net models representing exactly the observed behavior [7]. However both the ILP problem statement from event sequences and the processing have exponential complexity. This approach is being explored for other IPN classes; representative publications of this approach are [8] and [9].

In this paper we address the problem of identifying a DES controlled by a PLC during its operation. Both controller's inputs and outputs are sampled from the initial state for building a single sequence of I/O vectors, which is processed yielding an interpreted Petri net (IPN) model.

This approach is based on a previously presented efficient method for coping with concurrent partially observable DES [10] which processes a set of cyclic input/output sequences yielding models including silent transitions and non-labelled places. This technique has been extended and adapted for identifying actual PLC-based controlled discrete manufacturing systems, which operate during a long time period performing repetitive tasks. Thus the main contribution of this paper with respect to the previous one is the ability for detecting cyclic behavior from the single I/O sequence and building progressively a safe IPN; this allows updating the model when new input-output vectors are added to the sequence.

The paper is organized as follows. In section 2 the background on Petri nets and languages is outlined. In section 3, the identification problem is stated and the stepwise method is presented. In section 4 a case study is dealt through a software tool implementing the proposed method.

2. Background

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: $P = \{p_1, p_2, ..., p_{|P|}\}$ and $T = \{t_1, t_2, ..., t_{|T|}\}$ 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 (from transitions to places).

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 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 residing inside each place; it is usually expressed as an $|P|$-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 $v_k(i) = 0$, $i \neq j$, $v_k(j) = 1$, this equation is called the PN state equation. The reachability set of a PN is the set of all possible reachable markings from M_0 firing only enabled transitions; this set is denoted by $R(G, M_0)$.

A PN is called *safe* if $\forall M_k \in R(G, M_0)$, $\forall p_i \in P$, $M_k(p_i) \leq 1$.

Now it is defined 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 where:

G is a PN structure, $\Sigma = \{\alpha_1, \alpha_2, ..., \alpha_r\}$ is the input alphabet, and $\Phi = \{\phi_1, \phi_2, ..., \phi_q\}$ is the output alphabet.

$\lambda : T \rightarrow \Sigma \cup \{\varepsilon\}$ is a labeling function of transitions, where ε represents a system internal event externally uncontrollable; it is not allowed that the symbol ε is associated to more than one $t_j \in p_i^\bullet$.

$\varphi : R(Q, M_0) \rightarrow (\mathbb{Z}^+)^q$ is an output function, that associates to each marking in $R(Q, M_0)$ a q-entry output vector; $q = |\Phi|$ is the number of outputs. φ is represented by a $q \times |P|$ matrix, such that if the output symbol ϕ_i is present (turned on) every time that $M(p_j) \geq 1$, then $\varphi(i, j) = 1$, otherwise $\varphi(i, j) = 0$.

When an enabled transition t_j is fired in a marking M_k, then a new marking M_{k+1} is reached. This behavior is represented as $M_k \xrightarrow{t_j} M_{k+1}$; the state equation is completed with the marking projection $y_k = \varphi M_k$, where $y_k \in (\mathbb{Z}^+)^q$ is the k-th output vector 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 (t_i can be fired when the associated input symbol is presented). Otherwise it is uncontrollable (t_i is autonomously fired). 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. $P = P^m \cup P^u$ where P^m is the set of measurable places and P^u is the set of non-measurable places.

Definition 5: The *l-length I/O language* of an IPN (Q, M_0) is defined as:

$$\pounds^l(Q, M_0) = \left\{ \left[\frac{\lambda(t_{i+1})}{\varphi(M_{i+1})} \right], \left[\frac{\lambda(t_{i+2})}{\varphi(M_{i+2})} \right], ..., \left[\frac{\lambda(t_{i+l})}{\varphi(M_{i+l})} \right] \right\}$$

where $M_i \xrightarrow{t_{i+1}} M_{i+2} ... \xrightarrow{t_{i+l}} M_{i+l}$; $M_i \in R(Q, M_0)$

3. Stepwise Identification

3.1. Problem statement

Consider a DES composed of a Plant and a Controller (a PLC) operating in a closed-loop as showed in Figure 1. We assume that the data exchanged between Plant and PLC are binary signals. The input signals of the PLC (outputs of the Plant) are generated by the sensors of the Plant. The output signals of the PLC (inputs of the Plant) control the actuators of the Plant. The external behavior of such a DES system can be observed (and characterized) by the evolution of the value of all input/output (I/O) signals exchanged between the controller and the plant.

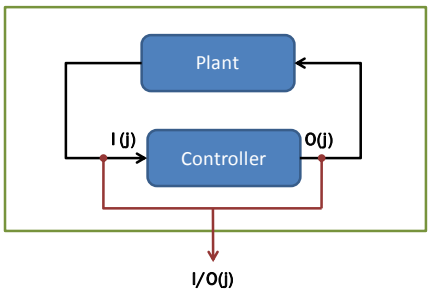

Figure 1. Closed loop controller-plant DES

At each end of cycle of the PLC, the current value of all Inputs and Outputs (called I/O vector) can be easily captured and recorded in a data base. Each new observed I/O vector (when at least one I/O changes its value) belongs to an I/O vector alphabet. We can concatenate such vectors to construct a sequence, which is the input of our identification algorithm.

Definition 6: The I/O vector alphabet of a DES with m inputs and n outputs is $\Sigma_{m,n} = \{1,0\}^{(m+n)}$.

Definition 7: The observed input-output sequence of a DES S with m inputs and n outputs is: $w = \left[\frac{I(1)}{O(1)} \right] \left[\frac{I(2)}{O(2)} \right] ... \left[\frac{I(j)}{O(j)} \right] ...$, where $[I(j)O(j)]^T$ is the j-th observed I/O vector belonging to $\Sigma_{m,n}$.

Definition 8: The observed input-output language of length l of a DES S is defined as $\mathcal{L}^l(S) = \{\varepsilon\} \cup \{w(i+1)\, w(i+2) \ldots w(i+h) | 1 \le i+h \le l\}$.

Now the identification problem can be defined. Given a DES whose only available information is an observed I/O sequence w arbitrarily large 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 [4].

It is important to point out that the aim of obtaining a model through identification is not to represent only the observed language, but to represent the observed behaviour and to infer actual behaviour that has not been observed during data collection. In order to accomplish this inference, the parameter κ, is used as a measure of state equivalence. When κ-equivalent states are found, they are merged, increasing the accepted language of the IPN and consequently the modelled behaviour. The notion of state equivalence with respect to κ is explained below.

Assumption: In this work it is considered that the I/O sequence is measured from the initial state of the global system (Plant and Controller). It is progressively updated with a new I/O vector, when any entry changes its value.

3.2. General strategy

The method allows the progressive construction of a safe IPN representing exactly the sampled input-output language of length κ +1 of the DES.

From the I/O vector sequence, an event sequence is computed and a sequence of event substrings of length κ is built. Every substring is associated to a transition of a PN, which describes the causal relationship between event substrings. A PN node path formed by non-measurable places represents the substring sequence; this path is built taking into account the possible repetitive observed behaviour (internal model). Then simplifications may be applied. Notice that the number of non-observable places is not predefined.

Finally, the model is completed by including observable places which are related to pertinent transitions in the PN according to output changes provoked by events; also input symbols are associated. This part of the algorithm can be concurrently performed at any moment, for example when a cycle is identified, whilst the internal model is updated by processing the new I/O vectors.

3.3. I/O sequence processing

3.3.1. Events sequence

The I/O vector sequence w is progressively built by adding new observed I/O vectors. In this way, a string of observed events are computed.

Definition 9: An observed event vector $\tau(j)$ is the variation between two consecutive I/O vectors: $\tau(j) = w(j+1) - w(j)$. The m first entries of $\tau(j)$, denoted as $\beta(\tau(j))$ correspond to the variation between two consecutive input vectors $I(j)$, $I(j+1)$ (input event). A symbolic input event $\lambda'(\tau(j))$ is a string representation of the input event vector $\beta(\tau(j))$; it is computed as:

$$\lambda'(\tau(j)) = \begin{cases} I_i\_1 \textbf{ if } I_i(j+1) - I_i(j) = 1 \\ I_i\_0 \textbf{ if } I_i(j+1) - I_i(j) = -1 \\ \varepsilon \textbf{ if } I_i(j+1) - I_i(j) = 0 \end{cases}$$

Then for a sequence w, a sequence of observed events $\tau(j) = \tau(1)\ \tau(2) \ldots \tau(j) \ldots$ is obtained. During the process, if the difference has not been observed before, a new event e_j is created ($\tau(j) = e_j$).

3.3.2. κ-length event traces

Events e_j represent changes in the observable behaviour of the system. However two identical events may be generated during different internal conditions during the system execution. In order to distinguish within the internal behaviour when these changes are exhibited, κ precedent events are considered. This is embedded in the notion of κ-length event trace.

Definition 10: An event trace $\tau^{\kappa}(j)$ is the substring from τ of length κ whose last event is $\tau^{\kappa}(j)$. $\tau^{\kappa}(j) = \tau(j-\kappa+1)\,\tau(j-\kappa+2) \ldots \tau(j)$.

This notion is useful to determine during the identification process if two states represent the same internal behaviour. Then the notion of equivalent states involves the history of κ events that lead to such states.

Definition 11: Two states of the model representing the identified system are κ-equivalent if the event traces $\tau^{\kappa}(j)$ leading to such states are the same.

3.4. Identification algorithm

The procedure for building the IPN model from the I/O sequence can be summarized on the UML activity diagram of Figure 2. It consists of five main steps that are described below.

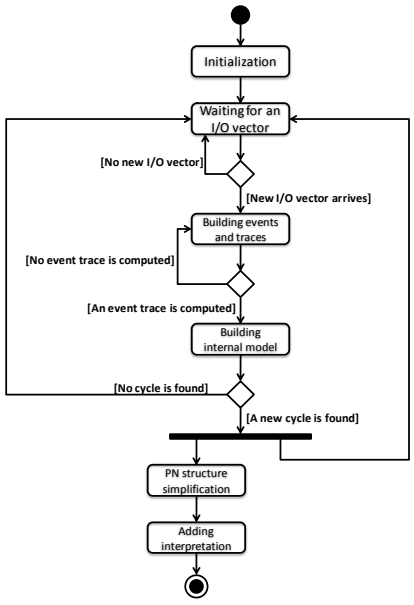

Figure 2. Stages of the identification algorithm

Step 1. *Initialization*

In this step, a PN structure is initiated. This is done by the following statements:

$T \leftarrow \emptyset; ET \leftarrow \emptyset;\ P \leftarrow \{p_{ini}\};$ //Create an initial empty set of transitions T, an initial empty event traces set ET, and an initial set of places P containing a place p_{ini}.

$M_0(p_{ini}) \leftarrow 1;\ \mu(p_{ini}) \leftarrow w(1); current \leftarrow p_{ini};$ *//Put a token on p_{ini} and associate to it the first observed vector w(1). Take such a place as current.*

Step 2. *Building events and traces*

Once the net is initiated, the procedure iterates on new I/O vectors. When an input or an output changes its value, an I/O vector is considered to update the events sequence and the events traces according to *Definition 9* and *Definition 10* respectively.

Let e_j be the last event in the trace $\tau^{\kappa}(j)$; the associated transition will be denoted as $t_r^{e_j}$ (more than one transition may have associated the same e_j). The internal model building can be systematically performed following the next procedure.

If $\tau^{\kappa}(j) \notin ET$ *//If the computed trace is new*

then $ET \leftarrow ET \cup \{\tau^{\kappa}(j)\}; T \leftarrow T \cup \{t_r^{e_j}\}; \gamma(t_r^{e_j}) \leftarrow \tau^{\kappa}(j);$

$\forall p_a \in P,\ I(p_a, t_r^{e_j}) \leftarrow 0; O(p_a, t_r^{e_j}) \leftarrow 0;$ *//create a transition $t_r^{e_j}$ to represent the trace $\tau^{\kappa}(j)$;*

$I(current, t_r^{e_j}) \leftarrow 1;$ *//create an arc from current to $t_r^{e_j}$*

$P \leftarrow P \cup \{p_{out}\}; \forall t_b \in T, I(p_{out}, t_b) \leftarrow 0, (p_{out}, t_b) \leftarrow 00;\ \mu(p_{out}) \leftarrow \mu(current) + e_j;$ *//create a new place p_{out} and associate it with correspondent marking ;*

$O(p_{out}, t_r^{e_j}) \leftarrow 1;$ *//create an arc from $t_r^{e_j}$ to p_{out};*

$current \leftarrow p_{out};$ *//take p_{out} as current.*

else*//If the computed trace is not new*

If $t_r^{e_j} \in current^{\bullet}$ and $\gamma(t_r^{e_j}) = \tau^{\kappa}(j)$ *//If one of the output transitions $t_r^{e_j}$ of current place represents the observed trace $\tau^{\kappa}(j)$*

then $current \leftarrow (t_r^{e_j})^{\bullet};$ *take the output place of $t_r^{e_j}$ as current*

else *//If current place has not an output transition representing $\tau^{\kappa}(j)$*

If $\exists t_r^{e_j} \in T | \gamma(t_r^{e_j}) = \tau_j^{\kappa}(j)$ *and* $\mu({}^{\bullet}t_r^{e_j}) = \mu(current)$ *//If there is a transition $t_r^{e_j}$ representing $\tau^{\kappa}(j)$ such that its input place ${}^{\bullet}t_r^{e_j}$ has the same associated marking $\mu({}^{\bullet}t_r^{e_j})$ than current place.*

then take $p_{in} = {}^{\bullet}t_r^{e_j};$ *//take ${}^{\bullet}t_r^{e_j}$ as p_{in}*

$\forall t_b \in T, I(p_{in}, t_b) \leftarrow \vec{\oplus}(I(p_{in}, t_b), I(current, t_b));$

$\forall t_b \in T, O(p_{in}, t_b) \leftarrow \vec{\oplus}(O(p_{in}, t_b), O(current, t_b));$

// $\vec{\oplus}$ 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 consider $\tau_i^{\kappa}(j)$ as new.

Step 4. *PN structure simplification*

After performing step 3, the algorithm waits for an I/O change by returning to step 2. Nevertheless, notice that merging places through step 3 of the algorithm could lead to merging of equivalent transitions. When such a merging is performed, a cycle on the PN is created. This is considered as a representative change in the structure of the model, and thus, simultaneously with launching of step 2, step 4 is executed to make a PN simplification procedure. Such a procedure based on concurrence transformations has already been explained in [11]. It basically consists of the analysis of different paths between two places containing transition permutations leading to concurrent components transformations. If there exist *m!* paths, it is verified if every one of them is a permutation from each other. When this is true, the subnet can be transformed into a concurrent component of G' preserving the same behaviour.

Step 5. *Adding interpretation*

Once the PN has been simplified through step 4, input and output information is included on the model, obtaining an IPN representing the language $\mathcal{L}^{\kappa}(S)$ of the DES that has been observed. Input information is added to the IPN by associating symbols to transitions according to the symbolic event input function of *Definition 9*. The procedures to add output information and simplify implicit non observable places are summarised below and deeply explained on [11].

1. Create n measurable places corresponding to each one of the outputs of the system.
2. Add arcs to and from the measurable places to the transitions of the net, according to its associated event e_j.
3. Put tokens in the corresponding measurable places to represent the first observed output vector.
4. If there is a non-measurable place whose inputs and outputs are exactly the same than any measurable place, remove such a non-measurable place and its input and output arcs.

3.5. Example 1

Consider a DES with three output signals, Φ = {A, B, C}, and three input signals Σ = {a, b, c}. The entries of the binary I/O vectors have the following correspondence: $[a\ \ b\ \ c\ \mid\ A\ \ B\ \ C]^T$. An I/O sequence is progressively observed. The first measured I/O vector corresponds to the initial state of the DES: $w(1) = [0\ \ 0\ \ 0\ \mid\ 0\ \ 0\ \ 0]^T$.

When a second I/O vector $w(2) = [1\ \ 0\ \ 0\ \mid\ 1\ \ 0\ \ 0]^T$ is measured, the event vector $\tau(1) = e_1 = [1\ \ 0\ \ 0\ \mid\ 1\ \ 0\ \ 0]^T$ is computed; the input event vector is $\lambda(1) = [1\ \ 0\ \ 0]^T$ and its corresponding symbolic input event is $\lambda'(1) = a\_1$, i.e. the rising edge of a.

Considering a value $\kappa = 1$, we can compute the first event trace $\tau^1(1) = e_1$. Notice that, in this case, trace and event are the same. This event trace is related with a transition of the IPN. The IPN constructed after observing two I/O vectors is on Figure 3.

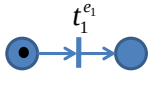

Figure 3. PN representing e_1

When a third I/O vector $w(3) = [1\ \ 0\ \ 0\ \ |\ \ 0\ \ 0\ \ 0]^T$ arrives, $\tau(2) = e_2 = [0\ \ 0\ \ 0\ \ |\ \ -1\ \ 0\ \ 0]^T$, $\lambda(2) = [0\ \ 0\ \ 0]^T$ and $\lambda'(2) = \varepsilon$ are computed and the model is updated, as showed in Figure 4.

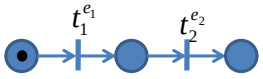

Figure 4. PN representing the sequence e_1e_2

Until 8th I/O vector, the situation is quite similar: a new event is observed and the model is updated.

$$w = \left(\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} \overrightarrow{1}^{e_1} \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} \overrightarrow{0}^{e_2} \\ 0 \\ 0 \\ 0 \\ -1 \\ 0 \end{bmatrix} \begin{bmatrix} \overrightarrow{0}^{e_3} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} \overrightarrow{0}^{e_4} \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} \overrightarrow{0}^{e_5} \\ 0 \\ 0 \\ 0 \\ -1 \\ 0 \end{bmatrix} \begin{bmatrix} \overrightarrow{1}^{e_6} \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} \overrightarrow{0}^{e_7} \\ 0 \\ 0 \\ 0 \\ 0 \\ -1 \end{bmatrix} \begin{bmatrix} \overrightarrow{1}^{e_8} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \right)$$

When 9^{th} vector $w(9) = [1\ \ 0\ \ 0\ \ |\ \ 1\ \ 0\ \ 0]^T$ is measured, the event $\tau(8) = e_1 = [1\ \ 0\ \ 0\ \ |\ \ 1\ \ 0\ \ 0]^T$ is computed and the trace $\tau^1(8)$ is identified through Step 3 as an already observed trace e_1. Since it leads to the same marking than the input place of $t_1^{e_1}$, such a place and the output place of $t_7^{e_7}$ can be merged as observed on Figure 5.

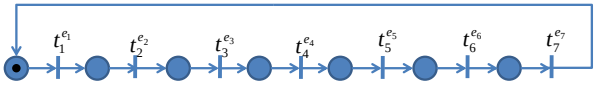

Figure 5. Internal model for the first detected cycle

Since a cycle is found, steps 4 and 5 of the algorithm are executed, leading to an intermediate IPN model showed on Figure 6.

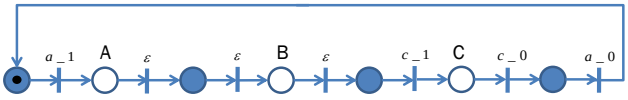

Figure 6. IPN for the first detected cycle

Simultaneously to the creation of the intermediate IPN, more I/O vectors are added to the observed sequence and PN is updated:

$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} \overrightarrow{0}^{e_3} \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} \overrightarrow{0}^{e_2} \\ 0 \\ 0 \\ 0 \\ -1 \\ 0 \end{bmatrix} \begin{bmatrix} \overrightarrow{1}^{e_4} \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} \overrightarrow{0}^{e_7} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} \overrightarrow{1}^{e_1} \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} \overrightarrow{0}^{e_3} \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} \overrightarrow{0}^{e_2} \\ 0 \\ 0 \\ 0 \\ -1 \\ 0 \end{bmatrix} \begin{bmatrix} \overrightarrow{1}^{e_4} \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} \overrightarrow{0}^{e_8} \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Two more cycles are found in this sequence and intermediate IPN models are created. We show only the PN obtained after founding the second cycle (Figure 7) and its equivalent model transformed by analysing concurrency (Figure 8). After applying the steps 4 and 5 the IPN obtained from this PN is showed in Figure 9.

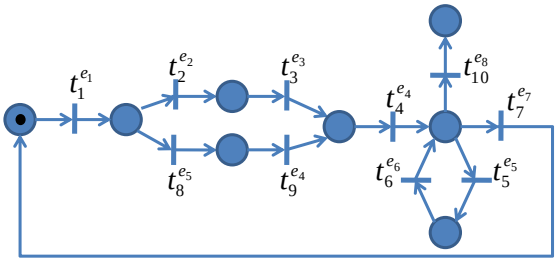

Figure 7. PN corresponding to the whole I/O sequence

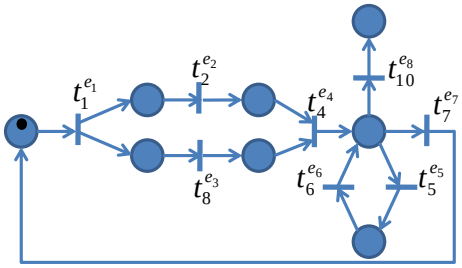

Figure 8. Equivalent internal model representing concurrency

Remark. The simplification by analysis of concurrency in Step 4 is not strictly necessary for representing the event vector sequences; however the equivalent model with concurrent transitions may be simpler. The aim of this simplification is not minimizing the number of nodes in the model, but obtaining fairly reduced models useful for understanding the DES behaviour.

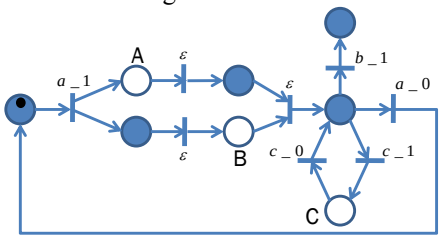

Figure 9. IPN for the complete sequence

4. Method implementation and application

4.1. An identification tool

Based on the algorithms presented in section 3, a software tool has been developed to automate the IPN model synthesis. The architecture of the tool is showed in Figure 10.

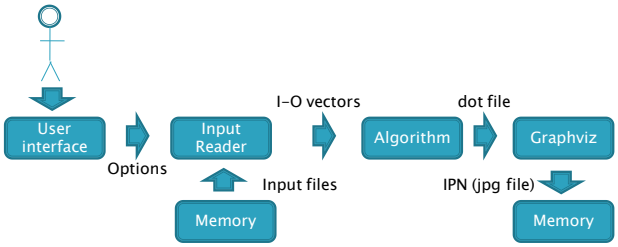

Figure 10. Software architecture

The user interface allows capturing the input/output sequence and shows the obtained model graphically. Several data are provided to the tool: a text file containing the I/O sequence (with one line per I/O vector), the parameter κ, the names of the input and output signals, and the desired name of output file. Additionally it is specified the order in which inputs and outputs appear in the txt file (since due to data collection issues they could be inverted) and the index numbers of the signals to take into account if a mask is going to be applied.

Later, an input reader component processes the input file and transforms the input/output sequence into a vector sequence. These vectors will be delivered to a component called Algorithm in which the identification algorithm is implemented. The output of this component is a dot file that can be given to the Graph Visualization Software (Graphviz) to generate an image file jpg.

The presented identification tool has been tested on several examples of diverse size. Below we illustrate the use of such software tool through a small size case study.

4.2. Case study

For space considerations a small size application example is presented in this paper dealing with an automated manufacturing system; it is taken from [5]. The purpose of such a system is to sort parcels according to their size (Figure 11). It has 9 inputs signal sensor from the system: a_0, a_1, a_2, b_0, b_1, c_0, c_1, k_1, k_2, and 4 outputs (signal to the actuators): A+, A-, B, C.

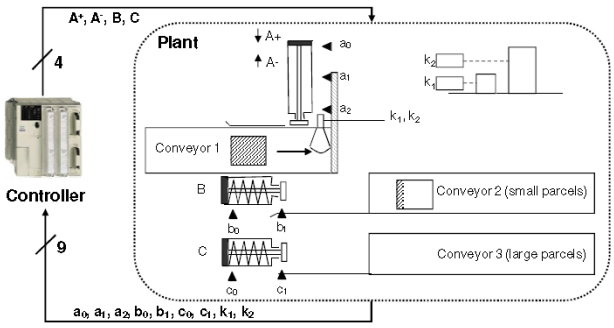

Figure 11. Layout of the system case study

A sequence consisting of 216 I/O vectors has been observed; it is showed on Figure 12. Notice that cycles are not specified on the sequence and they must be identified. Binary values of each I/O vector correspond to signals A^+, A^-, B, C, k_1, k_2, a0, a1, a2, b_0, b_1, c_0, and c_1 respectively. Notice that the input and output signals order in each vector is inverted (first outputs, later inputs).

In a first stage the identified model based on the first 31 I/O observed vectors is showed in Figure 13. It can be noticed that cylinder C has not worked yet, since big parcels have not yet arrived.

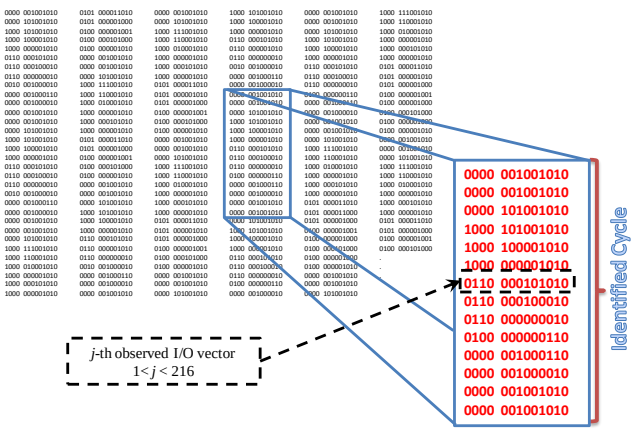

Figure 12. I/O sequences of the case study

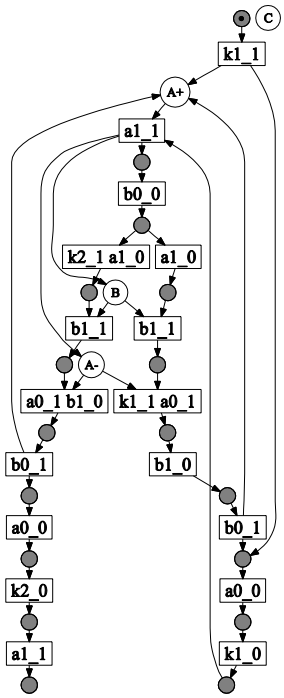

Figure 13. Obtained model after 31 PLC cycles

Using the complete sequence of I/O vectors the obtained model is showed in Figure 14. The identification procedure finds successfully cyclic behaviour in the single sequence of I/O vectors.

Notice that in this IPN model there are paths formed by non observable places. This is due to the observation of input changes that do not affect the outputs. In order to obtain a more compact model a simplification strategy has been presented in [10] and is recalled below. It consists in merging several places, representing internal behaviour whose detected events do not have effect on the outputs, into a single one where an output event must occur. Consider the following I/O vector sequence involving one input *x* and two outputs *A*, *B*:

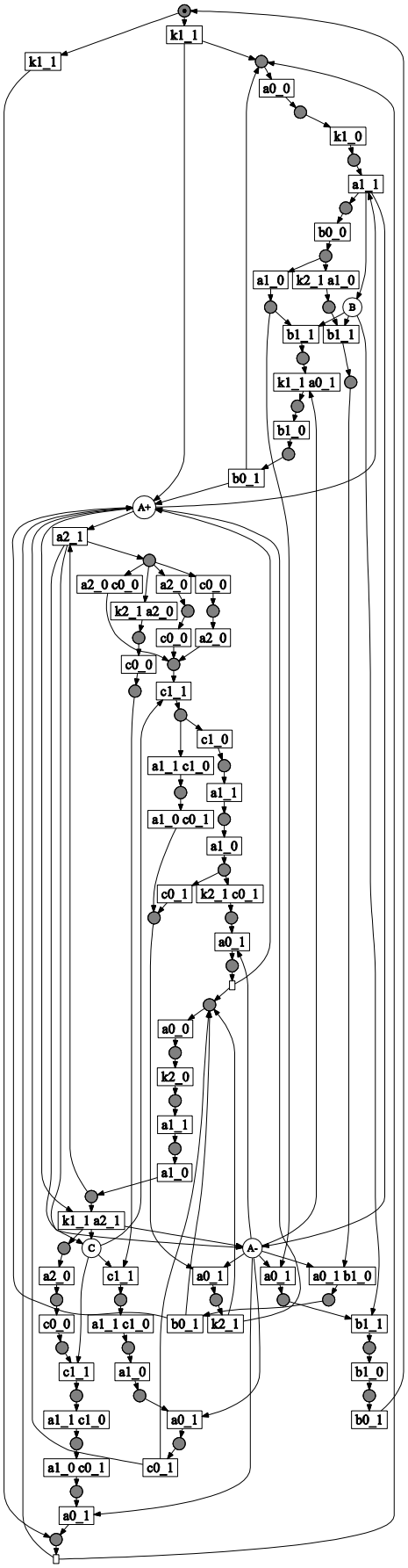

Figure 14. IPN for the case study

$$\begin{bmatrix} x \\ A \\ B \end{bmatrix} \qquad \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

This sequence can be represented as: $A \xrightarrow{x\_1} A \xrightarrow{\varepsilon} B$, which can be compacted as: $A \xrightarrow{x\_1} B$. This can be generalized to:

$$A \xrightarrow{e_i} A \xrightarrow{e_j} \ldots A \xrightarrow{e_k} B \cong A \xrightarrow{e_i e_j \ldots e_k} B$$

The application of this simplification procedure, also included in the software tool yields the IPN model showed in Figure 15.

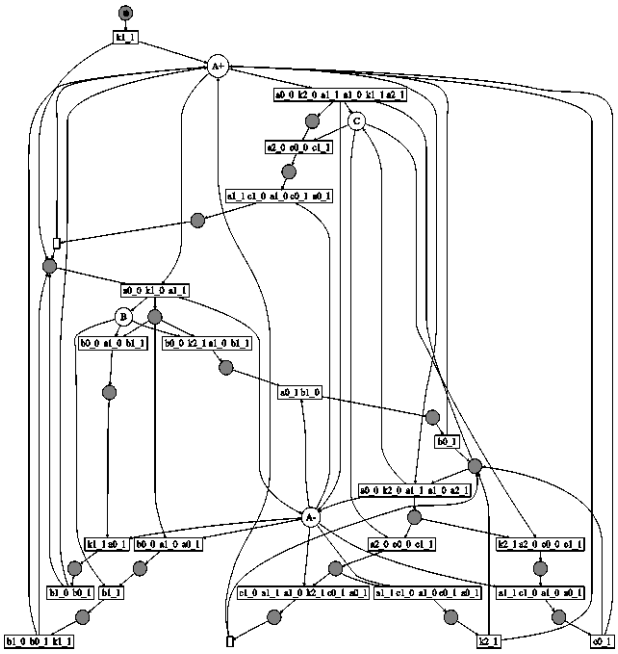

Figure 15. Model after applying simplification rule

5. Discussion

The method herein proposed allows dealing with complex automated DES because it takes into account technological characteristics of actual controlled systems, and because it is based on efficient algorithms. This feature is not still addressed in current literature on the matter in which several features considered in the current stated problem have not been dealt.

Although in [2] and [3] cycle finding from single sequence is dealt, it is based on the observation of the same initially observed outputs vector, which is not very often the case for real systems; besides system's inputs are not taken into account. In [4], [6], [10] are considered both inputs and outputs, but cyclic sequences are supposed to be known. Techniques in [7], [8], [9] process as input data a language generated by the system, which is given as events sequences, regardless how the events are obtained from I/O data.

6. Conclusions

Stepwise identification of automated manufacturing systems has been addressed. This black box approach allows obtaining IPN models from a single input/output sequence that exhibits the closed loop behaviour of PLC-based controlled plants. The proposed technique builds progressively the IPN when new I/O vectors are measured during the operation of the automated DES. The identified model is a close approximation of the compound controller-plant behaviour, which can be detailed for controller redesign or model-based diagnosis purposes.

Current research focuses on the reduction of the obtained model by the analysis of the ulterior influence of inputs that apparently do not provoke changes in the outputs. Also, the inference of non observed behavior regarding concurrent sub-processes is an issue to deal with.

Acknowledgement

The first author is sponsored by CONACYT Mexico, grant number 50312.

References

[1] A.P. Estrada-Vargas, E. López-Mellado, J.-J. Lesage. "A Comparative Analysis of Recent Identification Approaches for Discrete-Event Systems", Mathematical Problems in Engineering. Volume 2010, Hindawi. doi:10.1155/2010/453254

[2] M. Meda-Campaña, E. López-Mellado, "A passive method for on-line identification of discrete event systems", Proc. of the IEEE Int. Conf. on Decision and Control, Orlando, Florida, USA. pp. 4990-4995, Dec 2001

[3] M. Meda-Campaña, E. López-Mellado, "Identification of Concurrent Discrete Event Systems Using Petri Nets", Proc. of the IMACS 2005 World Congress, Paris, France, pp.1-7, Jul 2005

[4] 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 (Czech Republic), July 2005

[5] M. Roth, J.-J. Lesage, L. Litz, "An FDI Method for Manufacturing Systems Based on an Identified Model", Proc. of IFAC Symposium on Information Control Problems in Manufacturing (INCOM 2009), Moscow, Russia, pp. 1389-1394, June 2009

[6] M. Roth, J.-J. Lesage, L. Litz, "Black-box identification of discrete event systems with optimal partitioning of concurrent subsystems", Proc. of the American Control Conf. (ACC 2010), Baltimore, Maryland, USA, pp. 2601-2606, June 2010

[7] M.P. Cabasino, A. Giua and C. Seatzu, "Identification of Petri Nets from Knowledge of Their Language", Discrete Event Dynamic Systems, Vol. 17, No. 4, pp. 447-474, 2007

[8] M. P. Cabasino, A. Giua, C. Seatzu, "Identification of unbounded Petri nets from their coverability graph", Proc. of the IEEE Int. Conf. on Decision & Control, San Diego, CA, USA, pp. 434 – 440, Dec 2006.

[9] M. Dotoli, M. P. Fanti, A. M. Mangini, "Real time identification of discrete event systems using Petri nets", Automatica, Vol. 44, No. 5, pp. 1209-1219, May 2008.

[10] A.P. Estrada-Vargas, E. Lopez-Mellado, J-J. Lesage. "An Identification Method for PLC-based Automated Discrete Event Systems". Proc. of the IEEE Int. Conf. on Decision and Control, pp.6740-6746. Atlanta, USA. Dec. 2010.

[11] A.P. Estrada-Vargas, E. López-Mellado, J.-J. Lesage. "Off-line Identification of Concurrent Discrete Event Systems Exhibiting Cyclic Behaviour". Proc. of IEEE Int. Conf. on Systems Man and Cybernetics, San Antonio Tx, USA, pp. 181-186, Oct 2009.