Identification of Partially Observable Discrete Event Manufacturing Systems

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

Identification of Partially Observable Discrete Event Manufacturing Systems

Ana Paula Estrada-Vargas[*,♦], Ernesto López-Mellado[*], and Jean-Jacques Lesage[♦]
*CINVESTAV Unidad Guadalajara. Zapopan, México. {aestrada, elopez}@gdl.cinvestav.mx
♦LURPA Ecole Normale Supérieure de Cachan. Cachan, France. Jean-Jacques.lesage@lurpa.ens-cachan.fr

Abstract— **This paper deals with black-box identification of discrete event manufacturing systems that are automated using a programmable logic controller (PLC). The behavior of the system is observed during its operation and is represented by a single long sequence *w* of observed input/output (I/O) signals vectors. The identification method, conceived for addressing large and complex industrial systems, consists of two complementary stages; the first one obtains, from *w*, the observable part of an interpreted Petri net (PN) model composed of observable places and transitions describing the reactive behavior of the system. Afterwards, *w* is transformed into a sequence *S* of transition firings from which a PN model that reproduces *S* is inferred. This paper focuses on the second stage of the method in which a PN is built by adding non-labeled places and arcs that represent the non-observed behavior of the whole system by assuring the reproduction of *w*; this technique is based on discovering the causal and concurrent relationships between transitions in *S*.**

Keywords— Black-box identification; Automated manufacturing systems; Interpreted Petri nets; Internal behaviour model.

I. INTRODUCTION

Identification methods of discrete event systems (DES) allow building systematically a mathematical model (Petri nets, automata) that describes the behavior of an unknown or ill-known system based on the observation of its evolution. Observations consist of data revealing the system activity: sequences of operations, events, messages, signals etc., and the models allow reproducing the observed behavior.

DES identification has been first addressed as a problem of grammatical inference [1] [2] for obtaining finite automata (FA) that represent a given language. Afterwards, Petri net (PN) models have been proposed for coping with more complex systems exhibiting concurrent behavior [3].

Three main approaches more specifically conceived for identifying discrete event manufacturing systems have been proposed in recent literature.

The incremental synthesis approach, proposed in [4] [5], deals with unknown partially measurable DES exhibiting cyclic behavior. Several algorithms for building interpreted PN (IPN) 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 behaviors than those observed.

Other recent method [6] allows building efficiently a non-deterministic FA (NFA) from a set of input/output sequences, measured from the DES to be identified. The obtained NFA generates exactly the same input/output (I/O) sequences of given length than the observed ones. The method was conceived for fault detection in a model-based approach [7] and extended for obtaining an optimal partitioning of concurrent subsystems for distributed fault detection [8].

The off-line approach based on integer linear programming (ILP) yield free-labeled PN models representing exactly the observed behavior expressed as sequences of events [9]. The method is able to handle few short sequences, due to the inherent limitations of ILP regarding its computational complexity. This approach is being explored for other PN classes [10] [11].

A recent stochastic approach allows obtaining timed PN models [12]. Other related works can be found in surveys on identification methods in [13] and [14]. Furthermore, recent publications on process mining techniques, more suitable for event driven organizational systems than for industrial ones, can be found in [15].

In our approach the problem of identifying partially observable discrete manufacturing systems composed by a controller (a Programmable Logic Controller: PLC) and a plant operating in closed loop is addressed. Both controller's inputs and outputs are sampled for building a single sequence of I/O vectors, which is processed yielding an IPN model.

The aim is to discover, from this observation, how operations of the system are interrelated and construct a concise model which can explicitly show the discovered behavior, in particular, concurrency, synchronization, resource sharing, etc.

Identification of systems in operation involves two important aspects to consider: the system operation and the observation process. Technological issues of both aspects must be considered in the proposed algorithms to construct suitable abstractions. In this paper these issues are addressed by analyzing the observed sequence in order to establish a clear relationship between inputs and outputs of the controller. The proposed method allows building a compact and expressive IPN that is ordinary and safe. It consists of two complementary stages; the first one obtains, from the I/O sequence *w*, the observable subnet composed by places and transitions labeled with output and input functions respectively

[16]; during the construction of the model a transition sequence S, which reproduces w, is built.

The paper focuses on the second stage, which allows building efficiently from S, the non-observable part of the model including places (and arcs) ensuring the reproduction of w. The remainder of the paper includes a brief recall of the first stage of the method, and develops the proposed approach for building the non-observable part of the identified PN.

II. 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 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 $M : P \rightarrow Z^+$ represents the number of tokens residing inside each place; it is usually expressed as an $|P|$-entry vector. 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} . This behavior is represented as $M_k \xrightarrow{t_j} M_{k+1}$. The new marking can be computed as $M_{k+1} = M_k + Cu_k$, where $u_k(i) = 0$, i≠j, u_k(j) = 1; this equation is called the PN state equation. The reachability set, denoted by $R(G,M_0)$, of a PN contains all possible reachable markings from M_0 firing only enabled transitions. A PN is said to be 1-bounded or *safe* when $\forall M_k \in R(G,M_0)$, $\forall p_i \in P$, $M_k(p_i) \leq 1$.

Now it is defined IPN [17], 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 $Q = (G, V, \Sigma, \Phi, \lambda, \varphi)$ with an initial marking M_0 where: G is a PN structure, $V = \{v_1, v_2, ..., v_r\}$ is the set of variables, $\Sigma = \{\alpha_1, \alpha_2, ..., \alpha_s\}$ is the set of events, and $\Phi = \{\phi_1, \phi_2, ..., \phi_q\}$ is the output alphabet. $\lambda : T \rightarrow C \times E$ is a labeling function of transitions, where $C = \{C_1, C_2, \ldots\}$ is the set of variable conditions and $E = \{E_1, E_2, \ldots\}$ is the set of events.

In an *IPN*, a transition t_j can be fired ***if*** t_j is enabled, ***and if*** condition $C(T_j)$ is true, ***when*** the event in $E(T_j)$ occurs.

$\varphi : R(Q,M_0) \rightarrow (Z^+)^q$ 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 $\varphi(i, j) = 0$.

The state equation is completed with the marking projection $Y_k = \varphi M_k$, where $Y_k \in (Z^+)^q$ is the k-th output vector of the *IPN*.

Definition 4: A place $p_i \in P$ is said to be *observable* if the i-th column vector of φ is not null, i.e. $\varphi(\bullet, i) \neq 0$. Otherwise it is *non-observable*. $P = P^o \cup P^u$ where P^o is the set of observable places and P^u is the set of non-observable places.

III. Input-Output Identification

The problem statement and the main features of the identification method we propose are briefly described. A more detailed presentation can be found in [16].

A. Identification of automated DES

In this work we consider systems composed by a Controller (a PLC) and a Plant, denoted as {PLC + Plant}, working in a closed loop. 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 identification is made from the point of view of the PLC (Fig. 1). A PLC cyclically performs three main steps: a) Input reading, where signals are read from the sensors; b) Program execution, to determine the new outputs values for the actuators; and c) Output writing, where the control signals to the actuators are set. At each end of the Program execution step, the current value of all Inputs and Outputs (called I/O vector) is captured and, if it is different from the previous one, recorded in a data base.

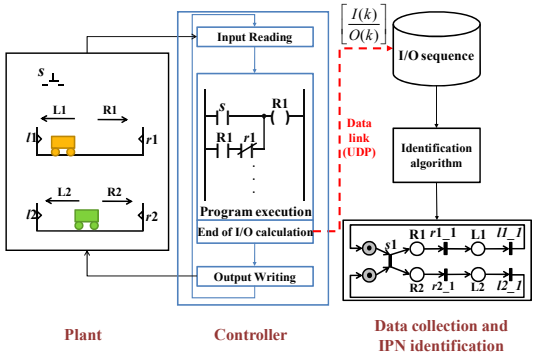


Fig. 1. {PLC + Plant} compound and identification procedure.

Besides the number of inputs and outputs, the only available data for the identification procedure is a single I/O vector sequence, in which two consecutive vectors are different, whose length depends on the observation duration:

$$w = \left[\frac{I(1)}{O(1)}\right], \left[\frac{I(2)}{O(2)}\right], \left[\frac{I(3)}{O(3)}\right], \ldots$$

where $I(j)$ and $O(j)$ are respectively the values of the r inputs and q outputs at the j-th PLC cycle.

In order to analyze signals evolution, we compute *event vectors*, i.e., the difference between two consecutive I/O vectors: $E(k) = w(k + 1) - w(k) \neq 0$. Each event vector can be decomposed into input and output event vectors:

$$E(k) = \left[\frac{IE(k)}{OE(k)}\right]$$

in which there are three possible observable situations:

a) IE(j) ≠ 0 and OE(j) ≠ 0: An input change has provoked directly an output change; consequently, a state evolution. The I/O causality is observed at the same PLC cycle.

b) $IE(j) \neq 0$ and $OE(j) = 0$: Only the input change is observed in the PLC cycle. It could mean that either i) The output change will appear in the next cycle, or ii) The controller is not sensitive to the input change, or iii) It provoked a change of state that is not observable through a change of output.

c) $IE(j) = 0$ and $OE(j) \neq 0$: Only the input change is observed in the PLC cycle. It could be a consequence of the previous situation (case i) or a programmed behavior considering only input levels or timers from the state reached at the event *j-1*.

B. Identification of Partially observable DES

All of the above situations must be taken into account to represent the system dynamics in the model. Besides representing the system's behavior from the I/O vector sequence as an IPN, our purpose in this research is also to provide an identified model as expressive and compact as possible, allowing to represent causal relationships and concurrency of the involved operations. For this purpose we have conceived a two-step method based on the analysis of the relationships between inputs and outputs along the observed behavior represented by the I/O-sequence *w*. The two steps of the method are the following:

Step1. Discovering the input-output reactive behavior. The observable part of the IPN is built, consisting of sub-graphs named "IPN fragments", composed by observable places and transitions labeled with algebraic expressions onto the set of input variables. Figure 2 shows an IPN model including four fragments.

Step2. Inferring the non-observable part of the IPN. All the observable behaviors being captured in IPN fragments previously built, the sequence *w* is transformed into a firing sequence *S* of observable transitions. Non-observable places and arcs, representing an abstraction of the internal and non-observable behavior of the identified system, are added in such a manner that *S* (thus *w*) can be reproduced in the final IPN. In Fig. 2 the non observable part is depicted with dashed lines.

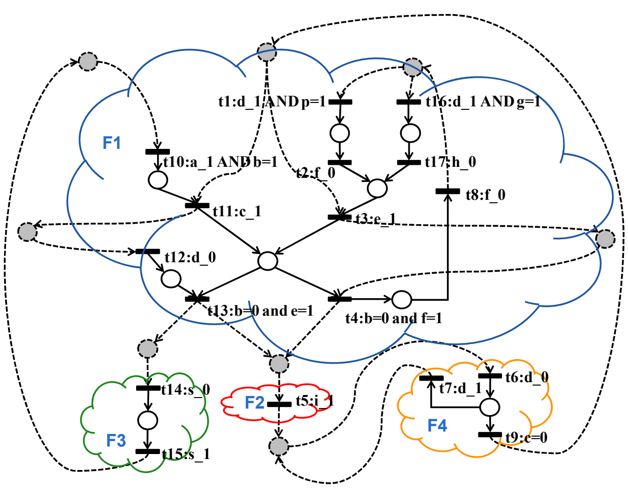


Fig. 2 IPN model including four fragments and non observable nodes

Notice that IPN fragments can be connected, through non-observable places, only sequentially or in parallel. That's why in section III we will present a technique for discovering causal and concurrency relationships between transitions in *S*.

The outcome of the first stage of the method is now briefly recalled with the help of a small example in next section.

C. Example 1

The purpose of this example is to show the outputs of the step 1 of the method when a sequence *w* is processed; it is fully developed in [16]. Consider the manufacturing system shown in figure 3 whose function is to sort parcels according to their size. The PLC has 9 inputs, that are signals generated by the sensors of the plant for detecting positions of cylinders (a_0, a_1, a_2, b_0, b_1, c_0, c_1) and presence of parcels (k_1, k_2), and 4 outputs, that are signals controlling the actuators of the plant (cylinders A+, A-, B, C). The vector entries correspond to the following distribution: $[A+\ A-\ B\ C\ k_1\ k_2\ a_0\ a_1\ a_2\ b_0\ b_1\ c_0\ c_1]^T$.

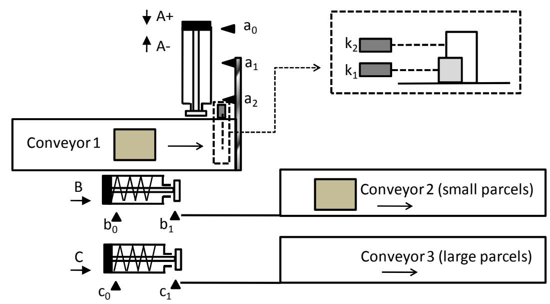
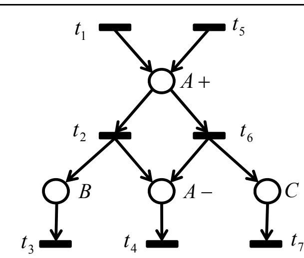


Figure 3. System's layout

A sequence of 216 observed I/O vectors has been processed (only the twenty first vectors are shown below). The IPN fragment, the transition labeling functions, and the transition sequence *S* shown in figure 4 are obtained. Notice that in this example only one fragment is found.

$$w = \left[\begin{array}{c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c}
0&0&1&1&1&0&0&0&0&0&0&0&0&1&1&1&0&0&0&0\\
0&0&0&0&0&1&1&1&0&0&0&0&0&0&0&0&1&1&1&0\\
0&0&0&0&0&1&1&1&1&0&0&0&0&0&0&0&1&1&1&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
\hline
0&1&1&1&1&0&0&0&0&0&0&0&1&1&1&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&1&1&0&0&0&0&0&1&1&1&1&1&1&0&0&0&0&0&1\\
0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&1&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&1&1&1&1&1&0&0&0&0&0&1&1&1&1&1&1&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{array}\right] \cdots$$



$$\lambda(t_1) = (k1 \wedge a0 \wedge b0 \wedge c0) \bullet (\varepsilon)$$
$$\lambda(t_2) = (=1) \bullet (a1\_1)$$
$$\lambda(t_3) = (=1) \bullet (b1\_1)$$
$$\lambda(t_4) = (=1) \bullet (a0\_1)$$
$$\lambda(t_5) = (k2 \wedge a0 \wedge b0 \wedge c0) \bullet (\varepsilon)$$
$$\lambda(t_6) = (=1) \bullet (a2\_1)$$
$$\lambda(t_7) = (=1) \bullet (c1\_1)$$

S = t1 t2 t3 t4 t1 t2 t3 t4 t1 t2 t3 t4 t5 t6 t7 t4 t1 t2 t3 t4 t5 t6 t7 t4 t1 t2 t3 t4 t1 t2 t3 t4 t5 t6 t7 t4 t5 t6 t7 t4 t1 t2 t3 t4 t5 t6 t7 t4 t5 t6 t7 t4 t5 t6 t7 t4 t5 t6 t7 t4 t5 t6 t7 t4 t1 t2 t4 t3 t1 t2 t3 t4 t1 t2 t3 t4 t1 t2 t4 t3 t1 t2 t3 t4 t1

Figure 4. IPN fragment, transition labeling, and sequence of transitions resulting from the first stage of the identification method.

IV. IDENTIFICATION OF THE NON-OBSERVABLE BEHAVIOR

A. Problem (re)statement

The second stage of the method determines the non-observable part of the model consisting of pertinent unlabeled places (and arcs) that rely the fragments and assures that the sequence S (consequently w, which is the actual observed behavior) can be reproduced.

This problem can be stated as follows: given an observable *IPN* model whose structure is (P^{obs}, T, Pre^{obs}, $Post^{obs}$) and a transitions sequence $S = t_1\ t_2\ ...\ t_j\ ... \in T^*$ reproducing the I/O sequence w, a *PN* structure (P^{nobs}, T, Pre^{nobs}, $Post^{nobs}$) that reproduces S and an initial marking M_0 enabling S must be found. The new *PN* structure is $G=(P, T, I, O)$ with $P= P^{obs} \cup P^{nobs}$, $Pre= Pre^{obs} \cup Pre^{nobs}$, $Post= Post^{obs} \cup Post^{nobs}$. The *PN* must be ordinary and safe.

The method proposed herein extracts, from S, precedence and concurrency relations among transitions, which will determine univocally the final structure of the identified model. First, some properties derived from the sequence S are introduced. Afterwards, based on such properties, a technique allowing determining sequential and concurrency relationships among the transitions in S is proposed. Then, the rules for building a net structure rendering the sequential and concurrency relationships are presented.

B. Behavioural properties

Since our construction method is based on the discovering of sequential and concurrency relationships into S, some notions must be defined before introducing the construction procedure of the non-observable behavior.

Definition 5. The relationship between transitions in S that are observed consecutively is expressed in a relation $Seq \subseteq T \times T$ which is defined as $Seq = \{(t_j, t_{j+1}) | 1 \le j < |S|\}$. If $(t_a, t_b) \in Seq$, this is denoted by $t_a < t_b$.

In a *PN* model every pair in *Seq* may in fact be represented differently. If $t_a < t_b$, this behavior could be issued from one of two situations in N described in the following definition.

Definition 6. Every couple of consecutive transitions t_a, t_b in *Seq* can be classified in one of the following situations:
- *Causal relationship.* If the occurrence of t_a enables t_b. In a PN structure, this implies that there must be at least one place from t_a to t_b.
- *Concurrent relationship.* If both t_a and t_b are simultaneously enabled, but t_a occurs first and its firing does not disable t_b. In a PN structure, this implies that it is impossible the existence of a place from t_a to t_b. In this case, t_a and t_b are said to be concurrent, denoted as $t_a || t_b$.

The following notion is the *systematic precedence* of a transition t_j with respect to another transition t_k; it establishes a necessary condition for t_j to occur repeatedly.

Definition 7. A transition t_j is *preceded systematically* by t_k, denoted as $t_k \angle t_j$ iff t_k is always observed between two apparitions of t_j in S. By convention, we say that $t_j \angle t_j$ if t_j was observed at least twice in S. The *Systematic Precedence Set* of t_j is a the function SP: $T \to 2^T$, that indicates which transitions *must* be fired to re-enable the firing of t_j, i.e. $SP(t_j)=\{t_k | t_k \angle t_j\}$. If t_j was observed only once in S, then $SP(t_j) = \varnothing$.

The following notion determines straightforward a particular structure from S.

Definition 8. Two transitions t_a, t_b are called transitions in a two-cycle if S contains the subsequence $t_a t_b t_a$ or the subsequence $t_b t_a t_b$. The two-cycle transitions set TC of S is given by $TC=\{(t_a,t_b) | t_a, t_b$ are in a two-cycle$\}$.

Example 2. In the sequence $S = t_1\ t_2\ t_3\ t_4\ t_1\ t_2\ t_4\ t_3\ t_5\ t_6\ t_7\ t_4\ t_1\ t_2\ t_3\ t_4\ t_5\ t_6\ t_7\ t_4\ t_1\ t_2\ t_3\ t_4\ t_1\ t_2\ t_3\ t_4\ t_5\ t_6\ t_7\ t_4\ t_5\ t_6\ t_7\ t_4\ t_1\ t_2\ t_3\ t_4\ t_5\ t_6\ t_7\ t_4\ t_5\ t_6\ t_7\ t_4\ t_1\ t_2\ t_3\ t_4\ t_5\ t_6\ t_7\ t_4\ t_5\ t_6\ t_7\ t_4\ t_5\ t_6\ t_7\ t_4\ t_5\ t_6\ t_7\ t_4\ t_1\ t_2\ t_4\ t_3\ t_1\ t_2\ t_3\ t_4\ t_1\ t_2\ t_3\ t_4\ t_1\ t_2\ t_4\ t_3\ t_1\ t_2\ t_3\ t_4\ t_1$ from Example 1, it can be easily determined $Seq=\{(t_1,t_2), (t_2,t_3), (t_3,t_4), (t_4,t_1), (t_4,t_5), (t_5,t_6), (t_6,t_7), (t_7,t_4), (t_2,t_4), (t_4,t_3), (t_3,t_1)\}$. Also, the following precedence relationships are found $t_2 \angle t_1$, $t_3 \angle t_1$, $t_4 \angle t_1$, thus $SP(t_1)=\{t_1, t_2, t_3, t_4\}$. Notice that $SP(t_j)$ is the set of transitions that must invariantly occur to fire t_j repeatedly. The rest of the SP sets are $SP(t_2)=\{t_1, t_2, t_3, t_4\}$, $SP(t_3) = \{t_1, t_2, t_3\}$, $SP(t_4) = \{t_4\}$, $SP(t_5) = \{t_4, t_5, t_6, t_7\}$, $SP(t_6) = \{t_4, t_5, t_6, t_7\}$, $SP(t_7) = \{t_4, t_5, t_6, t_7\}$. The set of transitions in a two-cycle is $TC= \varnothing$.

C. Causal and concurrency relationships

Based on the previous definitions we can now determine, for every pair of consecutively observed transitions in S, if the causality or concurrence property is verified. We will therefore deduce some structural properties regarding N.

Definition 9. A PN *circuit* is a path starting and ending in the same node. A circuit is said to be *simple* if it does not use the same transition more than once, and *elementary* if it does not use the same place more than once.

Relations between transitions in S can be determined. Proofs are omitted for the sake of brevity; they can be found in [18].

C.1 Causal relationship

Proposition 1. If $t_a \angle t_b$ ($t_a \in SP(t_b)$) then, there exist in N a simple elementary (SE) circuit to which both t_a and t_b belong.

Proposition 2. If $t_a < t_b$ and either $t_a \angle t_b$ or $t_b \angle t_a$ then there must exist in N a place from t_a to t_b.

Proposition 3. If $(t_a,t_b) \in TC$, then there must exist in N a place from t_a to t_b and a place from t_b to t_a.

Notice that when two transitions are observed consecutively and one is systematically preceded by the other, a causal relationship is found. Also, when two transitions are involved in a two-cycle relation, they are in a causal relationship each other. Observe that all of these relationships are structural, and thus do not depend of the initial marking of the net.

Definition 10. The causal relationship set *CausalR* keeps track of all the causal relationships between transitions in S.
$CausalR = \{(t_a,t_b) \mid (t_a < t_b)$ and $(t_a \angle t_b$ or $t_b \angle t_a$ or $(t_a,t_b) \in TC)\}$.

Example 3. From the *SP* and *Seq* sets of Example 2, we compute $(t_1,t_2) \in CausalR$ because $t_1 < t_2$ and $t_1 \angle t_2$ (*Proposition 2*). So, $CausalR=\{(t_1,t_2), (t_2,t_3), (t_4,t_1), (t_4,t_5), (t_5,t_6), (t_6,t_7), (t_7,t_4), (t_2,t_4), (t_3,t_1)\}$.

If a couple of transitions (t_a,t_b) in *Seq* belongs also to *CausalR*, there must be a place from t_a to t_b in order to

preserve the observed firing order. For the rest of the transition pairs in *Seq*, we must decide if a place should exist to relate them. Below, we will discuss some cases where the existence of a place can be discarded.

C.2 Concurrency relationship

If two transitions t_a and t_b are concurrent, there must not exist a place neither from t_a to t_b nor from t_b to t_a; otherwise, the firing of one would constrain the firing of the other one.

Definition 11. The set of all pairs of concurrent transitions is called $ConcR=\{(t_a,t_b)|\ t_a||t_b\}$.

If the sequence w is complete, (consequently, S) i.e., if it shows all of the possible behavior of the observed system, we can find concurrency between transitions that are not in a causal relation, as showed in the next proposition.

Proposition 4. Let t_a, t_b be two transitions which have been observed consecutively in a complete sequence S in both orders, i.e. $(t_a, t_b)\in Seq$, $(t_b, t_a)\in Seq$. Then $(t_a, t_b)\notin CausalR$ and $(t_b, t_a)\notin CausalR$ if and only if $t_a||t_b$.

In the sequence of Example 2, t_3 and t_4 fulfill the conditions to be concurrent. They are the only concurrent transitions $ConcR=\{(t_3,t_4)\}$.

It is well known that, in practice, the sequence w is not complete since it is not possible to assure that the whole behavior of a system has been observed in a finite time. The condition of Proposition 5 is therefore very restrictive, since it requires the observation of all possible behaviors; it could lead to the construction of incorrect models in case of incomplete sequences. Then, less constraining rules to find concurrence must be considered. Next, we present several properties which allow us to identify pairs of transitions which must be concurrent in the identified net N.

First, the notion of *Sequential Independence*, which is a characteristic of concurrent transitions, is introduced. Later, the propositions to find concurrency will be presented.

Definition 11. Two transitions t_a and t_b are *Sequentially Independent* if $(t_a,t_b) \notin CausalR$ and $(t_b,t_a) \notin CausalR$.

By inspection of the set *CausalR* of Example 3, it can be found that t_1 and t_5 are sequentially independent, as well as t_2 and t_6, and t_3 and t_4.

Proposition 5. Let t_a and t_b be two transitions in S which have been observed consecutively in both orders ($t_a<t_b$ and $t_b<t_a$). $t_a||t_b$ if:

a) t_a and t_b are *Sequentially Independent*

b) $|SP(t_a)| >1$ and $|SP(t_b)| >1$

In the example, we know that $t_3||t_4$, however, condition b) of Proposition 5 is not fulfilled because $SP(t_4) = \{t_4\}$. The $SP(t_j)$ is a singleton, meaning that t_j belongs to more than one elementary circuits and then Proposition 5 does not allow to find concurrent transitions to t_j. But if t_j is included in the *SP* of other transitions, we can find some concurrency relations, as shown in the next proposition.

Proposition 6. Let t_a and t_b be two transitions in S that have been observed consecutively in both orders ($t_a < t_b$ and $t_b < t_a$). $t_a||t_b$ if t_a and t_b

a) are *Sequentially Independent,* and

b) there exists a transition t_k such that $t_a\angle t_k$ ($t_a\in SP(t_k)$) and $t_b\angle t_k$ ($t_b\in SP(t_k)$)

In the current example t3 and t4 fulfill the conditions of this proposition, i.e. $t_3<t_4$, $t_4<t_3$, both transitions are sequentially independent and belong to $SP(t_2)$.

If concurrent transitions do not belong to synchronized threads, conditions of the next propositions help us to find a subset of concurrent transitions in which their firings do not depend from the occurrence of another transition t_k.

Proposition 7. Let be two transitions t_a and t_b which have been observed consecutively in both orders ($t_a < t_b$ and $t_b < t_a$). $t_a||t_b$ if t_a and t_b are:

a) *Sequentially Independent,* and

b) $\exists\, t_k$ such that $t_k\in SP(t_b)$, $t_k\notin SP(t_a)$, and $(t_a,t_k)\in Seq$

Definition 12. The *Inverse Systematic Precedence* set of a transition $SP^{-1}:T \rightarrow 2^T$ contains the transitions which are dependent of a common transition to re-enable their firing:

$$SP^{-1}(t_j) = \{t_k \mid t_k \neq t_j \text{ and } t_j \in SP(t_k)\}$$

Proposition 8. Let be t_a and t_b two transitions which have been observed consecutively in both orders ($t_a < t_b$ and $t_b < t_a$). $t_a||t_b$ if t_a and t_b are:

a) *Sequentially Independent*, and

b) $SP^{-1}(t_a) \neq \varnothing$, $\forall t_j\in SP^{-1}(t_a)$, $t_j \,||\, t_b$.

D. Building the non-observable PN

The computed causal and concurrency relations from sequence S are used to infer internal evolutions of the system by computing non-observable places of the net.

Definition 13. The set $Seq' = (Seq \setminus CausalR) \setminus ConcR$ contains the set of transition pairs (t_a,t_b) which have been observed consecutively, but are not in a causal relationship or in a concurrency relationship. If $Seq' \neq \varnothing$, there are two possibilities for the remaining transition pairs (t_a,t_b) in *Seq*':

a) They are input and output transitions respectively of a place with several input and output transitions.

b) They are concurrent, but w (thus, S) is not complete enough to find such a relationship.

Since our goal is to approximate as much as possible the language generated by the net N to the observed sequence S, we assume that if we have observed two transitions consecutively ($t_a<t_b$) but none of the previous propositions have determined that they are concurrent, thus the firing of t_a has enabled t_b. This is made in order to preserve in N the firing order observed in S. Then, a place will be added from t_a to t_b; this is denoted by $[t_a, t_b]$.

In general a place p can be denoted as $[t_{a1}\ t_{a2\ldots}\ t_{al}, t_{b1}\ t_{b2\ldots}\ t_{bh}]$, where t_{ai} are the input transitions of p and t_{bi} are the output transitions of p, and $l=|{}^\bullet p|$, $h=|p^\bullet|$.

The same place could be used to relate several consecutive transitions. If a transition t_k has been observed followed by two transitions t_{ai}, t_{aj} in S ($t_k<t_{ai}$ and $t_k<t_{aj}$), there are two cases to represent such observations into the PN model:

a) the case of selection, where t_{ai}, t_{aj} are neither concurrent nor consecutive; thus they are represented with the same place $[t_k, t_{ai}t_{aj}]$, or

b) the case of concurrency, where t_{ai}, t_{aj} are concurrent or consecutive and thus they are represented with different places $[t_k, t_{ai}]$ $[t_k, t_{aj}]$.

The same reasoning can be applied to couples of places sharing the same output transition $[t_{ai}, t_k]$ *and* $[t_{aj}, t_k]$.

In the current example, from *CausalR*, the causal dependencies $[t_4, t_1]$ and $[t_4, t_5]$ can be represented by a single place $[t_4, t_1t_5]$ in a selection structure, and $[t_2, t_3]$ and $[t_2, t_4]$ are represented by two places with a single input transition t_2.

For the parcels sorting system of Fig. 1, the PN structure as well as the computed initial marking (which is computed by allocating tokens enabling the sequence *S*) is shown in Fig. 5.a. Observe that in Fig. 4 observable places $[t_2, t_3]$ and $[t_6, t_7]$ already exist. By adding non-observable places to such a model and deleting implicit places $[t_2, t_3]$ and $[t_6, t_7]$, we obtain the IPN model shown in Fig. 5.b, which reproduces *w*.

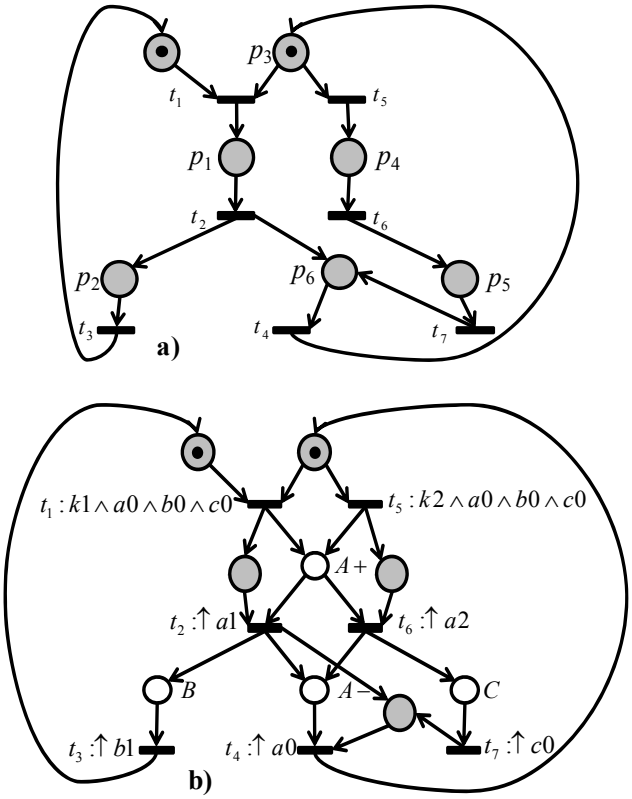


Fig. 5. a) Non observable model; b) Final identified IPN model

E. Token flow verification

The sequence *w* may not have shown enough combinations allowing determining concurrency. If *w* were complete, all the concurrent and sequential behavior could be found and represented, according to Proposition 5. However, since we know that *w* could not be complete, in order to approximate the language of *N* to *S* as much as we can, we have considered that if two transitions have not been declared as concurrent, they must be in a sequential relationship. However, if the transitions are actually concurrent, the sequential consideration could lead us to create arcs or places in the model constraining too much the behavior of the system and do not allow the firing of *S*. Now, we present some notions to verify if added places do not interfere in the correct reproduction of *S*.

Proposition 9. If the IPN model has been correctly build, every computed non-observable place *p* in *N* must fulfill the place input/output flow equation:

$$\sum_{t_i \in {}^\bullet p} Occ(t_i) = \sum_{t_i \in p^\bullet} Occ(t_i) \pm 1$$

where $Occ(t_i)$ is the number of occurrences of t_i in *S*.

Proposition 10. If there exists a place *p* such that $|{}^\bullet p|=1$, then $\forall t_j \in p^\bullet$, $t_k \in SP(t_j)$, where t_k is the input transition of *p*. Also, if there exists a place *p* such that $|p^\bullet|=1$, then $\forall t_j \in {}^\bullet p$, $t_k \in SP(t_j)$, where t_k is the output transition of *p*.

Correction rule. If the input/output flow equation or the conditions in Proposition 9 are not satisfied by some place, the arcs relating transitions which are not in *CausalR* are removed. If there are not pairs in *CausalR* representing such a place, it is deleted.

For the model of fig. 5.b the flow equation is verified for all the transitions.

F. Summary of the method

All the algorithms described in this section to construct the non-observable part of the IPN are summarized in the following polynomial-time procedure.

Algorithm 1. **Non-observable behavior construction**

Input: The sequence *S*
Output: Non-observable model representing *S*

1. Compute *Seq*, *SP* and *TC* from *S*
2. From *Seq*, *SP* and *TC* compute *CausalR*
3. From *Seq* and *CausalR*, compute *ConcR*
4. Merge transitions as specified in section *IV*.D
5. Validate and correct places as specified in section *IV.E*

Characteristics of Algorithm 1.In step 4 the pairs $(t_a, t_b) \in$ *CausalR* are represented by places $[t_a, t_b]$ and then merged to build the PN structure; this assures that many pairs in *Seq* are taken into account. Furthermore, the pairs $(t_a, t_b) \in$ *ConcR*, imply that both (t_a, t_b) and (t_b, t_a) in *Seq* are represented in the behavior of the PN. Therefore, if Seq'=((*Seq* \ *CausalR*) \ *ConcR*) \ $ConcR^{-1} = \varnothing$ then *S* can be reproduced by the model built.

If $Seq' \neq \varnothing$, some (t_b, t_a) in *Seq* have not been characterized as concurrent or causal. If they are actually sequential, all the verification rules are satisfied; otherwise, they are concurrent and they are corrected in step 5. Once they are corrected, the remaining places relate sequential transitions and thus the sequence *S* is reproducible.

All the procedures and tests derived from the defined sets and propositions involve simple operations on arrays of size |*S*| or |T|×|T|, which are performed efficiently.

G. Implementation issues

The algorithm has been implemented and tested on many examples exhibiting diverse situations using the following scheme: a PN is built with the help of a PN editor (PIPE), and then S is created by firing (enabled) transitions randomly. After the processing of S a PN model is identified, and then coded in XML, to be displayed again with PIPE. As an example, an interesting feature of the method is showed. The identification method applied to the following sequence:

$S = t_4\ t_1\ t_2\ t_3\ t_1\ t_5\ t_6\ t_7\ t_2\ t_3\ t_4\ t_5\ t_6\ t_1\ t_2\ t_3\ t_7\ t_1\ t_4\ t_2\ t_5\ t_4\ t_3\ t_5\ t_1\ t_4\ t_5\ t_2$
$t_6\ t_3\ t_1\ t_7\ t_6\ t_7\ t_6\ t_2\ t_3\ t_1\ t_2\ t_3\ t_7\ t_1\ t_4\ t_5\ t_2\ t_4\ t_5\ t_3\ t_4\ t_5\ t_4\ t_5\ t_1\ t_6\ t_7\ t_4\ t_2$
$t_3\ t_1\ t_5\ t_6\ t_7\ t_6\ t_2\ t_3\ t_7\ t_4\ t_1\ t_5\ t_2\ t_4\ t_5\ t_6\ t_7\ t_3\ t_1\ t_6\ t_2\ t_7\ t_6\ t_7\ t_6\ t_7\ t_6\ t_3\ t_7$
$t_4\ t_1\ t_2\ t_3\ t_5\ t_1\ t_4\ t_5\ t_6\ t_7\ t_2\ t_6\ t_3\ t_1\ t_7\ t_4\ t_2\ t_3\ t_5\ t_6\ t_1\ t_2\ t_3\ t_1\ t_7\ t_4\ t_5\ t_6\ t_7$
$t_4\ t_2\ t_3\ t_1\ t_5\ t_4\ t_2\ t_5\ t_6\ t_3\ t_7\ t_6\ t_1\ t_7\ t_4\ t_2\ t_3\ t_5\ t_4\ t_5\ t_1\ t_6\ t_2\ t_7\ t_4\ t_5\ t_6\ t_3\ t_7$
$t_1\ t_6\ t_7\ t_2\ t_3\ t_1\ t_2\ t_6\ t_7\ t_4\ t_5\ t_4\ t_5\ t_3\ t_4\ t_5\ t_4\ t_1\ t_5\ t_4\ t_5\ t_4\ t_5\ t_2\ t_6\ t_3\ t_1\ t_2\ t_3$
$t_1\ t_7\ t_2\ t_3\ t_1\ t_4\ t_5\ t_6\ t_2\ t_3\ t_1\ t_2\ t_3\ t_1\ t_2\ t_3\ t_1\ t_7\ t_2\ t_6\ t_7\ t_3\ t_1\ t_2\ t_4\ t_5\ t_4$

yields a PN model composed of two independent subnets shown in Fig. 6.

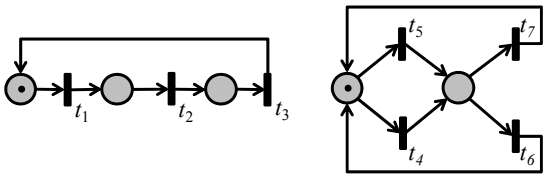


Fig 6. Two different components running concurrently

The complete method including both stages has been implemented as a software tool that process a sequence w and yields an IPN, reproducing exactly w. It has been tested on sequences obtained from real manufacturing systems. For a detailed description please consult chapter 5 in [18].

ACKNOWLEDGEMENT

The first author has been supported by CONACYT, Mexico, Grant No. 50312 and Région Ile de France.

CONCLUSIONS

A novel approach for identifying partially observable DES systems has been proposed. This method is composed of two steps: the first one is devoted to build the observable reactive behavior [16]; the second step, devoted to infer the non-observable behavior, has been described in this paper.

Although in literature there are techniques addressing the reformulated problem dealing with sequences of transitions, in this paper neither the cycles nor the language generated by the system are known *a priori*, and this method can handle very long sequences efficiently.

The complete method copes with complex automated DES because it takes into account technological characteristics of both actual controlled systems and the observation process. Implemented as a software tool, the method has been tested on experimental case studies which are very close to actual industrial discrete event processes. The performed tests reveal the efficiency of the methods when sequences formed by thousands of input-output vectors are processed in few seconds [18].

Since the approach is black-box, the obtained models represent the observed behavior; consequently, when the observation is made for a long time, the IPN approximates closely the actual behavior; afterwards the model can be completed using available knowledge on the process.

Current research addresses the problem of discovering indirect causal relationships of transitions that have not been observed consecutively by determining the t-invariants from S.

REFERENCES

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

[2] D. Angluin, "Queries and Concept Learning", *Machine Learning*, 2(4), pp. 319-342, 1988

[3] K. Hiraishi, "Construction of Safe Petri Nets by Presenting Firing Sequences", *Lectures Notes in Computer Sciences*, Vol. 616, pp. 244-262, 1992

[4] 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, FL, USA. pp. 4990-4995, Dec. 2001

[5] 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, July 2005

[6] S. Klein, L. Litz, J.-J. Lesage, "Fault detection of Discrete Event Systems using an identification approach", Proc. of 16th IFAC World Congress, Paper n°02643, 6 pages, Praha (Czech Republic), July 2005

[7] 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

[8] 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

[9] A. Giua and C. Seatzu, "Identification of free-labeled Petri nets via integer programming", Proc. of the 44th IEEE Conf. on Decision and Control, Seville, Spain, Dec. 2005

[10] M.P. Cabasino, A. Giua and C. Seatzu, "Identification of Petri Nets from Knowledge of Their Language", *Discrete Event Dynamic Systems*, 17(4), pp. 447-474, 2007

[11] M. Dotoli, M. P. Fanti, A. M. Mangini, "Real time identification of discrete event systems using Petri nets", *Automatica*, 44(5), pp. 1209-1219, May 2008

[12] Ould El Mehdi S., Bekrar R., Messai N., Leclercq E., Lefebvre D., Riera B., Design and identification of Stochastic and Deterministic-Stochastic Petri Nets, Trans. IEEE-SMCA, Part A, 42(4), pp.931-946, July 2012

[13] 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*, Vol. 2010, Hindawi. doi:10.1155/2010/453254

[14] M. P. Cabasino, P. Darondeau, M. P. Fanti, C. Seatzu, "Model identification and synthesis of discrete-event systems", *Contemporary Issues in System Science and Engineering*, IEEE/Wiley Press Book Series, M. Zhou, H.-X. Lim M. Weijnen (Eds), 2013.

[15] W.Van der Aalst, "Process Mining: Discovery, Conformance and Enhancement of Business Processes". Springer-Verlag, Berlin 2011.

[16] A. P. Estrada-Vargas, J.-J. Lesage, E. López-Mellado, "Identification of Industrial Automation Systems: Building Compact and Expressive Petri Net Models from Observable Behavior", Proc. of American Control Conference, pp. 6095 - 6101, Montréal (Canada), June 2012.

[17] R. David and H. Alla, "Petri Nets for Modeling of Dynamic Systems–A Survey", Automatica, 30(2), pp. 175-202, 1994

[18] A. P. Estrada-Vargas, "Black-box Identification of Automated Discrete Event Systems", Ph.D. Thesis, CINVESTAV Unidad Guadalajara, Guadalajara, Mexico. *http://www.gdl.cinvestav.mx/dehs-2013/tae.pdf*