Asymptotic Identification of Discrete Event Systems*

M. Meda-Campaña, A. Ramírez-Treviño and E. López-Mellado
CINVESTAV-IPN Unidad Guadalajara.
Prol. López Mateos Sur 590, 45090 Guadalajara Jalisco, México
{art@gdl.cinvestav.mx}

Abstract— **This work is concerned with the analysis of the on-line identifiability of the Discrete Event Systems (DES) using Interpreted Petri Nets (IPN). A theoretical framework characterizing identifiable systems when only the input and output system's signals are available is first addressed. Afterwards, based on this framework, an algorithm that progressively builds an IPN model of the system is presented. As a possible application of the problem herein addressed, the on-line identification problem is adapted to address the problem of DES model validation.**

I. Introduction

Loosely speaking, identifiability[11] is a property of dynamic systems that allows, through an entity called identifier, to estimate a mathematical model of the system when the input and output system's signals are available. In Discrete Event Systems (DES) literature there exist several works addressing the very close problem of language identification in the limit[4][1][12], in which a "correct" automata for a given positive sample of a regular language $\mathcal{L}_\alpha$ must be identified. The automata is tuned every time that a new word of the positive sample is presented to the identification algorithm.

In [4] Gold addressed the language identification in the limit to obtain a formal model of human language acquisition, also he shows that for regular languages this problem is NP-Complete. Addressing the same topic, in [1], Angluin proposed polynomial algorithms for some subclasses of regular languages, those called the zero-reversible languages. These ideas are extended in [2], in this work Angluin introduces the query learning model. In this model, every time that the identified automata is tuned, a "teacher" presents a counterexample to the identification algorithm. The counterexample is a word accepted by the guessed automata not belonging to the language $\mathcal{L}_\alpha$ or a word belonging to $\mathcal{L}_\alpha$ that it is not accepted by the guessed automata. Based on this model, Angluin proved that the identification problem of regular languages is polynomial. This problem is also addressed in [6]. In this work Hiraishi addressed the same problem for the I-reversible language class. He used Petri Nets (PN) to represent the language $\mathcal{L}_\alpha$.

*This work was supported by project CONACyT 29278-A

This work presents a method to build an interpreted Petri net (IPN) as a model for a DES. It assumes that the symbols emitted by the DES are available, but the language $\mathcal{L}_\alpha$, however, is unknown. The main contributions of this work are an asymptotic method to build an IPN model of the DES and the application of this method to the model validation problem.

The proposed method is called asymptotic since the IPN model is built as the system evolves. The output symbol sequences are used to determine the net structure and its interpretation (labeling with input and output symbols). This approach avoids the NP complete problem found in the techniques that use positive samples of input languages.

The paper also shows that the asymptotic identification is useful to solve the model validation problem. We use the results of identification for checking the correctness of a design against a given DES specification (an IPN model). First, a reduced input sequence is determined from the specification, then it is applied to the designed DES. The result of the identification is compared with the specification to determine if the designed device fulfills the specification.

The remainder of this paper is organized as follows; section II gives an outline of the used mathematical tool to model DES: interpreted PN. Section III states the asymptotic identification problem, the class of systems that will be studied and the identification algorithm and section IV presents the results for DES validation and gives an illustrative example. Finally the conclusions and future work are outlined.

II. Interpreted Petri Nets

This section introduces the definition of IPN[7] used in this paper. A good PN survey is presented in [8], and for details about marked graphs, state machines and free choice PN, an interested reader can see [9].

Definition 1: An **Interpreted Petri Net** (IPN) is the 6-tuple $Q = (N, \Sigma, \Phi, \lambda, D, \varphi)$ where $N = (G, M_0)$ is a PN[5] [8]; $\Sigma = \{\sigma_1, \sigma_2, ..., \sigma_r\}$ is the input alphabet of the net and σ_i is an input symbol; $\Phi = \{\phi_1, \phi_2, ..., \phi_s\}$ is the output alphabet and ϕ_i is an output symbol; $\lambda : T \rightarrow \Sigma \cup \{\varepsilon\}$ is a labeling function of transitions with the following restriction: $\forall t_j, t_k \in T$, $j \neq k$ if $I(p_i, t_j) = I(p_i, t_k) \neq 0$ and both $\lambda(t_j)$,

$\lambda(t_k) \neq \varepsilon$, then $\lambda(t_j) \neq \lambda(t_k)$, where ε represents an internal system event; $D : T \rightarrow \top$ is a feed-forward function, where $\top = \tau \cup \{\varepsilon\}$ and $\tau \subset T$. When a transition t_j is fired, then its name t_j or the ε symbol are given as an IPN output; $\varphi : \mathbf{R(G,M_0)} \rightarrow \{\Phi \cup \{\boldsymbol{\varepsilon}\}\}^{\mathbf{q}}$ is an output function where $\mathbf{R(G,M_0)}$ is the reachability set. The output vector $\varphi(M_k)$ of the net is a $q-$entry vector.

Remark 2: In this work (Q, M_0) will be used instead of $Q = (N, \Sigma, \Phi, \lambda, D, \varphi)$ to emphasize the fact that there is an initial marking in an IPN.

Remark 3: This paper focuses on the case where the function φ is linear and can be represented as a $q \times n$ matrix $\varphi = [\varphi_{ij}]$, where the $i-th$ row vector φ_i is the transpose of an elemental vector e_j, n is the number of places, and q is the dimension of the output vector.

A transition $t_j \in T$ of an IPN is enabled at marking M_k if $\forall p_i \in P$, $M_k(p_i) \geqslant I(p_i, t_j)$. If $\lambda(t_j) = a_i \neq \varepsilon$ is present and t_j is enabled, then t_j must fire. If $\lambda(t_j) = \varepsilon$ and t_j is enabled then t_j can 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 fact is represented as: $M_k \xrightarrow{t_j} M_{k+1}$. M_{k+1} can be computed using the dynamical part of the PN state equation: $M_{k+1} = M_k + Cv_k$, with C defined as in a PN and v_k defined as: $v_k(i) = 0$ $i \neq j$, $v_k(j) = 1$.

Note that by definition of λ, IPN are deterministic [3] over labeled transitions, i.e. two transitions with the same associated input symbol cannot have the same input places. However, they can be non-deterministic [3] over unlabeled transitions.

Definition 4: If $\lambda(t_i) \neq \varepsilon$ the transition t_i is said to be **controllable.** Otherwise it is **non-controllable**. 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**. A transition t_j is said to be **measurable** if $D(t_j) = t_j$. Otherwise it is **non-measurable**.

In this work, the measurable places of an IPN are depicted as clear circles, the non-measurable ones as dark circles, the measurable transitions as clear bars, while the non-measurable ones as dark bars.

Let $\sigma = t_i t_j t_k ...$ be a firing transition sequence. The Parikh vector $\overrightarrow{\sigma} : T \longrightarrow \mathbb{Z}^+$ of σ maps every transition t of T (the transitions of the net) to the number of occurrences of t in σ.

The state equation of an IPN is the following:

$$\begin{aligned} M_{k+1} &= M_k + Cv_k \\ y_k &= \varphi \bullet M_k \end{aligned} \tag{1}$$

where v_k is the Parikh vector of a fired transition sequence.

Definition 5: Let (Q, M_0) be an IPN, a firing sequence of (Q, M_0) is a sequence $\sigma = t_i t_j ... t_k$ such that $M_0 \xrightarrow{t_i} M_1 \xrightarrow{t_j} ... M_w \xrightarrow{t_k} M_x ....$

The firing language of (Q, M_0), is the set of all firing sequences of (Q, M_0), denoted as: $\pounds(Q, M_0) = \{\sigma | \sigma = t_i t_j ... t_k$ and $M_0 \xrightarrow{t_i} M_1 \xrightarrow{t_j} ... M_w \xrightarrow{t_k} ...\}$.

The input language of (Q, M_0) is $\pounds_{in}(Q, M_0) = \{\lambda(t_i)\lambda(t_j)... \lambda(t_k) |\ t_i t_j ... t_k \in \pounds(Q, M_0)\}$.

The output language of (Q, M_0) is $\pounds_{out}(Q, M_0) = \{\varphi(M_0)\varphi(M_1)...\varphi(M_w)... |M_0 \xrightarrow{t_i} M_1 \xrightarrow{t_j} ... M_w \xrightarrow{t_k} ...$ and $t_i t_j ... t_k \in \pounds(Q, M_0)\}$.

Definition 6: ${}^\bullet(t_j)$ $({}^\bullet(p_i))$ denotes the set of input places (transitions) to t_j (p_i); $(t_j)^\bullet$ $((p_i)^\bullet)$ is the set of output places (transitions) to t_j (p_i).

The notion of dependency will be useful to establish which transitions are in the same path in an IPN.

Definition 7: Two transitions t_i, t_j form a **dependency** $[t_i, t_j]$ iff $\exists p_k$ such that $p_k \in t_i^\bullet$ and $p_k \in^\bullet t_j$. If p_k is a non measurable place, then $[t_i, t_j]$ will be called a **non-measurable dependency** and it is denoted as **NDep**, otherwise will be called a **measurable dependency** and it is denoted as **MDep**. The set of all possible dependencies is denoted as Dep.

$Dep = Dep^m \cup Dep^u$, where Dep^m is the set of all measurable dependencies and Dep^u is the set of all non measurable dependencies.

Definition 8: Let (Q, M_0) be an IPN. (Q, M_0) is a **single non-measurable dependency (SDep)** IPN iff each non measurable place belongs to just one non measurable dependency.

III. Asymptotic identification problem

To identify a DES using IPN it should be the case where the DES can be modeled by at least one IPN. If it is not the case, then the DES cannot be identified by any IPN. Moreover, even in the case where the DES can be modeled by an IPN (Q, M_0), more properties must exhibit this net because all non-measurable event and places must be detected from the output symbols of the DES.

One needed property is the observability[10], because it ensures that the events (transitions) can be detected from the output. This property, however, does not ensure that the non-measurable places will be detected. To cope this problem this paper focuses in observable **SDep** IPN. For example, if two non-measurable dependencies $[t_i, t_j]$, $[t_i, t_k]$ are detected, then two non-measurable places are also detected. Next proposition formalizes these ideas.

Proposition 9: Let (Q, M_0) be an observable and **SDep** IPN, then the existence of transitions and places can be detected from the output language.

Proof: Since the IPN is observable, then any fired transition sequence and measurable place can be detected[10]. Non-measurable places are detected when two consecutive transitions t_i, t_j are no connected by a measurable place. Moreover the non-measurable dependency $[t_i, t_j]$ is identified. For each **NDep** $[t_i, t_j]$, a vector $u_{ij} = \begin{bmatrix} \nu_1 & \cdots & \nu_r \end{bmatrix}$ is computed; where r is the number of detected transitions and $\nu_i = 1$, $\nu_j = -1, \nu_k = 0$, $k \neq i, j$. Arranging

$\varphi C(\cdot, t_i)$ and u_{ij} vectors in a matrix fashion, as follows:

$$C = \begin{bmatrix} \varphi C(\cdot,t_1) & \cdots & \varphi C(\cdot,t_r) \\ & u_{ij} & \\ & \vdots & \\ & u_{km} & \end{bmatrix}$$

then all transitions and places can be detected. ■

Now, the asymptotic identification problem for this subclass of IPN can be stated.

Problem 10: Let S_f be a DES that can be modeled by an observable, **SDep** and binary IPN; and $\mathcal{M} = \{\Updownarrow_l, \Updownarrow_\infty, ...\}$ be the non empty set of all IPN. Then the asymptotic identification problem is defined as follows: 1. Select a functional $f : \{S_f\} \times \mathcal{M} \longrightarrow \mathcal{R}^+$ indicating the similitude between S_f and $m_j \in \mathcal{M}$. A lower value of $f(S_f, m_j)$ indicates that S_f and m_j are more similar. 2. Find out a model sequence $m_0, m_1, ..., m_k$, where $m_i \in \mathcal{M}$ such as $f(S_f, m_i) \leq f(S_f, m_{i-1})$

Based on this definition, we proposed a functional f to determine the number of φC columns and non-measurable dependencies that are missed in the model. This functional works fine for the IPN class herein studied because it guaranties that the measurable part of the system is actually represented in the guessed model. Moreover, since every dependence in the system also belongs to the guessed model, then the model just accepts the system words. Formally, the functional f is defined as follows.

Definition 11: Let $\{\varphi Cs_f\}$ be the set of columns of matrix φC of the system S_f and $\{\varphi C_{m_i}\}$ be the set of columns of matrix φC of the identified model m_i. Let $Dep(S_f)$ and $Dep(m_i)$ be the set of dependencies of the system S_f and the identified model m_i.

$$f(S_f, m_i) = |\{\varphi Cs_f\} - \{\varphi C_{m_i}\}| + |Dep(S_f) - Dep(m_i)| \quad (2)$$

Now, every fired transition sequence $\sigma_i = t_1 t_2 \cdots t_{k-1} t_k$ in the system (Q, M_0) yields a dependency sequence $[t_1 t_2][t_2 t_3] \cdots [t_{k-1} t_k]$. Using the dependency sequence, a simple IPN path (a sequence of transitions and places) can be identified such that it accepts the sequence σ_i. Then the simple IPN paths can be merged to obtain a model for (Q, M_0).

The following definition defines the m_i-word, it is a fired transition sequence in the system (Q, M_0), and will be useful to identify the dependency sequences.

Definition 12: Let (Q, M_0) be an observable, **SDep** and binary IPN and $m_i = (N_i', \hat{M}_0^i)$ be a proposed model for (Q, M_0). If $w_o = \varphi(M_0) \cdots \varphi(M_j)$ is an output word generated by (Q, M_0); and $w = t_1 \cdots t_k$ is the firing sequence detected when w_o was observed. w is an m_{i+1}-word iff $\varphi(M_0) = \varphi(M_j)$.

The next theorem proofs that the problem 10 has a solution.

Theorem 13: Let (Q, M_0) be an observable, **SDep** and binary IPN and $m_i = (N_i', \hat{M}_0^i)$ be a proposed model for (Q, M_0). If w is an m_{i+1}-word then a model $m_{i+1} = (N_{i+1}', \hat{M}_0^{i+1})$ can be built such that $f(S_f, m_i) \geq f(S_f, m_{i+1})$

Proof: The first model $m_0 = (N_0', \hat{M}_0^0)$ is built as follows. The number of places of N_0' is equal to the number of measurable places in (Q, M_0). This value can be computed from $\varphi(M_0)$. The number of transitions is zero and $\hat{M}_0^0(i) = \begin{cases} 1 \text{ if } \varphi(M_0)(i) \neq \varepsilon \\ 0 \text{ other case} \end{cases}$. Then, the initial error is: $f((Q, M_0), m_0) = |T| + |Dep|$.

Now the sets $\mathbf{Dep}^u$, $\mathbf{Dep}^m$, **Dep** and **Dep**$*$ (non-measurable dependences between m_i models) are declared to be empty. Proceeding by induction on the number x of detected m_x-words.

• Let $x = 1$. In this case an m_1-word= $t_1 t_2 ... t_r$ is determined when the output word $w_o = \varphi(M_0)\varphi(M_1) \cdots \varphi(M_k)$ of (Q, M_0) is observed. Such m_1-word= $t_1 t_2 ... t_r$ and columns $\varphi C(\bullet, t_1), ..., \varphi C(\bullet, t_r)$ can be computed using proposition 9. The following cases arise:

1.- t_1 is the first fired transition in m_1-word. A dependence $D = [t_r, t_1]$ is detected. If $D = [t_i, t_k]$ is an **NDep**, then it must be added[1] to $\mathbf{Dep}^u$ otherwise to $\mathbf{Dep}^m$.

2.- The firing of t_i occurs before the firing of t_j in the m_1-word. A dependence $D = [t_i, t_j]$ is detected. If $D = [t_i, t_k]$ is an **NDep**, then it must be added to $\mathbf{Dep}^u$ otherwise to $\mathbf{Dep}^m$.

For each **NDep** $[t_i, t_j]$, a vector $u_{ij} = \begin{bmatrix} \nu_1 & \cdots & \nu_r \end{bmatrix}$ is computed as in the proof of proposition 9 and an incidence matrix C_1' is computed. A new model m_1 is determined, where the incidence matrix of m_1 is C_1' and the initial marking $\hat{M}_0^1(i)$ is the inherited projection from $\hat{M}_0^0(i)$ with the new marked non measurable places. Since some transitions are detected by the m_1-word, then some columns $\varphi C(\bullet, t_i)$ are computed. Hence $f((Q, M_0), m_0) \geqslant f((Q, M_0), m_1)$.

• By the induction hypothesis, if $x = n$, then the m_n-word= $t_a t_b ... t_q$ is detected and $f((Q, M_0), m_{n-1}) \geqslant f((Q, M_0), m_n)$.

• Let $x = n + 1$. In this case m_{n+1}-word= $t_i t_j ... t_l$ is determined when the output word $w_o = \varphi(M_s)\varphi(M_t) \cdots \varphi(M_v)$ of (Q, M_0) is observed. Such m_{n+1}-word= $t_i t_j ... t_l$ and columns $\varphi C(\cdot, t_i), ..., \varphi C(\cdot, t_l)$ can be computed using proposition 9.

The dependencies of the m_{n+1}-word will be updated as follows:

- t_i is the first transition in m_{n+1}-word. A dependence $D = [t_l, t_i]$ is detected. If $D = [t_l, t_i]$ is an **NDep**, then it must be added to $\mathbf{Dep}^u$ otherwise to $\mathbf{Dep}^m$.

- t_i, t_k are two consecutive transitions in m_{n+1}-word. Then a dependency $D = [t_i, t_k]$ is identified. If there exists a dependency sequence $Dseq = [t_k, t_y][t_y, t_n]...[t_z, t_i]$ (taking dependencies from Dep) where no one dependency appears more than once, then $\mathbf{Dep}^u = \mathbf{Dep}^u - [t_y, t_n]$, where $[t_y, t_n] \in \mathbf{Dep}*$ ap-

[1] In this case, added means the union of the dependency set with the new found out dependencies

pears in $Dseq$, i.e. some dependencies between models must be eliminated[2] from $\mathbf{Dep}^u$ because they contradict the new observed behavior. Otherwise two cases arise:
a) There no exists $D_x = [t_x, t_k] \in \mathbf{Dep}^u$. In this case there exists no a contradiction in the net (i.e. there exists no a non-measurable dependency that invalidates the sequence $...t_i t_k...$ because t_k has not input places). If $D = [t_i, t_k]$ is an **NDep**, then it must be added to $\mathbf{Dep}^u$ otherwise to $\mathbf{Dep}^m$.
b) There exists $D_z = [t_z, t_k] \in \mathbf{Dep}^u$ (i.e. t_k belongs to an **NDep** that could be eliminated). If there exists a dependency sequence $Dseq = [t_i, t_y][t_y, t_n]...[t_z, t_k]$ (taking dependencies from Dep) where no one dependency appears more than once, then $[t_z, t_k]$ must be eliminated from Dep^u and a new **NDep** dependences $[t_i, t_q]$ must be added to $\mathbf{Dep}^u$. In the next step new **NDep** dependencies $[t_z, t_q]$ are added, where the t_q are taken from the dependences $[t_k, t_q]$.In both cases, if $D = [t_i, t_k]$ is an **NDep**, then it must be added to $\mathbf{Dep}^u$ otherwise to $\mathbf{Dep}^m$.

The dependencies between m_i-words will be updated as follows:

- Since m_n-word$= t_a t_b...t_q$ precedes m_{n+1}-word$= t_i t_j...t_l$, then a dependency $D = [t_q, t_i]$ is identified. Two cases arises:
a) If there exists a dependency sequence $Dseq = [t_i, t_y]...[t_z, t_q]$ (taking dependencies from **Dep**) where no one dependency appears more than once, then $\mathbf{Dep}^u = \mathbf{Dep}^u - [t_y, t_n]$, where $[t_y, t_n] \in \mathbf{Dep}*$ appears in $Dseq$.
b) If there no exists a dependency sequence $Dseq = [t_i, t_y]...[t_z, t_q]$ (taking dependencies from **Dep**) where no one dependency appears more than once, then the identified dependency $D = [t_q, t_i]$ must be added to $\mathbf{Dep}^u$ and to $\mathbf{Dep}*$ sets.

For each **NDep** $[t_i, t_j]$ in $\mathbf{Dep}^u$, a vector $u_{ij} = \begin{bmatrix} \nu_1 & \cdots & \nu_r \end{bmatrix}$ is computed as in the proof of proposition 9 and an incidence matrix C'_{n+1} is computed. A new model m_{n+1} could be determined, where the incidence matrix of m_{n+1} is C'_{n+1} and the initial marking $\hat{M}_0^{n+1}(i)$ is the inherited projection from $\hat{M}_0^0(i)$ with the new marked non measurable places.

If new transitions are found out, then the error $f((Q, M_0), m_{n+1})$ decreases. If the dependencies are updated to accept the new detected word, then eliminated dependencies do not belong to the system and also the error $f((Q, M_0), m_{n+1})$ decreases. Otherwise $f((Q, M_0), m_{n+1})$ will not increase, hence $f((Q, M_0), m_{n+1}) \geqslant f((Q, M_0), m_{n+1})$ and the theorem holds. ■

[2] In this case eliminated means removing from the dependency set the indicated dependency and the removing of the non-measurable place from the IPN.

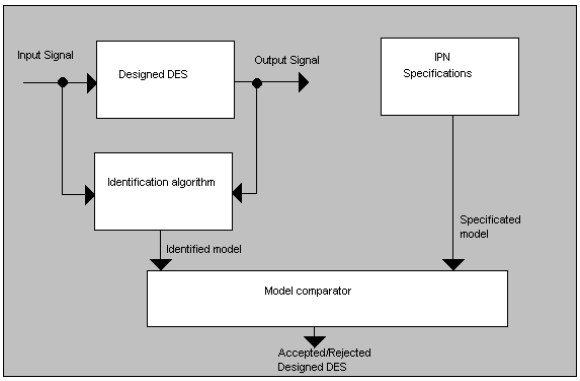


Fig. 1. Validation scheme

Algorithm: Asymptotic identification algorithm
Input: $\mathbf{Dep}^u = \emptyset$, $\mathbf{Dep}^m = \emptyset$, $\mathbf{Dep} = \emptyset$, $\mathbf{Dep}* = \emptyset$ and $\{\varphi C_{S_f}\} = \emptyset$
Output: An IPN model for the system behavior observed $\{\varphi C_{S_f}\}$ and **Dep**

Compute m_0 as was mentioned in the proof of theorem 13, $q = 0$
loop
1. Read the output symbol $\varphi(M_i)$ of the DES until m_q-word is detected.
2. Uses m_q-word to compute C'_1 and the current dependency sets Dep^u and Dep^m.
3. Updated the dependencies as in the proof of theorem 13. $q = q + 1$
4. Build model m_q.
5 End_loop

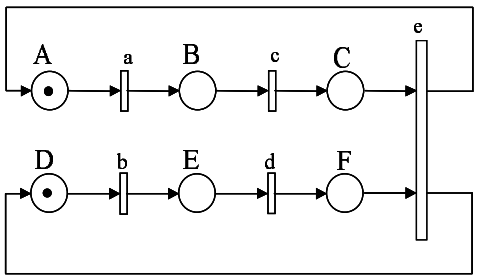
IV. Identification for validation

This section addresses the problem where a designed device must be tested to determine if it fits a set of preestablished specifications. This problem arises in different engineering areas, such as testability of IC or software verification. The scheme presented in figure 1 is introduced to solve the validation problem using the asymptotic identification algorithm.

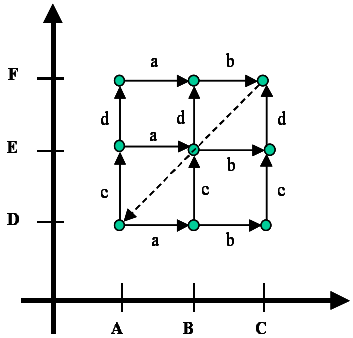
Problem 14: Let S be the specification in IPN terms of a DES. The specification is an observable, **SDep** and binary IPN. Let I be an implemented device (physical device) which has been designed to meet the specification S. Then the validation problem consists in determine if I fulfills the specification S.

Since the specification is known, then the identification process can be improved using a special set of input words. This set of words is chosen to help to efficiently compute the dependencies of the system. The approach herein used to find out the reduced set of firing transition sequences implies the study of the T and P components.

Proposition 15: Let (Q, M_0) be a live, structurally bounded IPN and T_Q an elemental T-component of



a) An IPN with two p-semiflows



b) The phase diagram of previous IPN

Fig. 2. An IPN with two P-Semiflow and its phase diagram.

(Q, M_0). Let $\sigma_i = ...t_i...t_j...$ and $\sigma_j = ...t_j...t_i... \in \pounds(T_Q, M_0)$. Then t_i and t_j belong to different p-components of T_Q.

Proof: If both transitions t_i and t_j belong to same P-component, then there only exists a directed path from t_i to t_j or from t_j to t_i (without passing twice by the same transition). Then just sequence σ_i or σ_j can be fired; so t_i and t_j must belong to different P-components. ■

Definition 16: Let (Q, M_0) be a live, structurally bounded IPN. Two transitions t_i, t_j of (Q, M_0) are said to be in contradiction iff there exist words σ_i,$\sigma_j \in \pounds(Q, M_0)$ such that $\sigma_i = ...t_i...t_j...$, $\sigma_j = ...t_j...t_i...$.

A. Reduced set of firing transition sequences for identification

Definition 17: Let (Q, M_0) be an observable, **SDep** binary and structurally bounded IPN. The Phase Diagram of (Q, M_0) is the reachability graph drawn in an N-dimensional space, where each axis is a discrete variable representing a P-semiflow; the values of the variable are the markings of the associated P-semiflow.

Example 18: Let (Q, M_0) be the IPN depicted in figure 2.a). The phase diagram of (Q, M_0) is shown in figure 2.b).

Definition 19: Let (Q, M_0) be an observable, **SDep,** binary and structurally bounded IPN and **PD** its phase diagram. The set of external sequences (those cyclic sequences forming the unbounded face) of the **PD** is denoted **ESeq**.

Example 20: In the **PD** of the figure 2.b), **ESeq**= $\{abcde, cdabe\}$.

Proposition 21: Let (Q, M_0) be an observable, **SDep**, binary and structurally bounded IPN, and T_Q be an elemental T-component of (Q, M_0). **ESeq** is a reduced set of firing transition sequences allowing to identify correctly T_Q.

Proof: Let σ_i, σ_j $\in$**ESeq** two transition sequences such that:

$$\sigma_i = \overbrace{t_{1p_1} t_{2p_1} \cdots t_{m_1 p_1}}^{axis_1} \cdots \overbrace{t_{1p_n} t_{2p_n} \cdots t_{m_n p_n}}^{axis_n}$$

$$\sigma_j = \overbrace{t_{1p_n} t_{2p_n} \cdots t_{m_n p_n}}^{axis_n} \cdots \overbrace{t_{1p_1} t_{2p_1} \cdots t_{m_1 p_1}}^{axis_1}$$

Note that for any pair of contradictory transitions t_k, t_l of T_Q, there exist a pair of sequences σ_i, σ_j $\in$**ESeq**, where t_k, t_l are in different order. Therefore, by firing σ_i, the dependency **NDep** $[t_k, t_l]$ will be generated; and by firing σ_j, the dependency **NDep** $[t_l, t_k]$ will be generated. Now, using the algorithm derived from the proof of theorem 13, these dependencies will be eliminated. Other dependencies are identified by the same algorithm and a model of T_Q can be built.

Remark 22: Since the specification S is live and structurally bounded IPN, then it is conservative and repetitive and each place p_k and each transition t_j of S belongs to a T-Component and to a P-Component [5]. Moreover, extending proposition 21, a complete model for the specification can be built.

■

B. An illustrative example

Example 23: Let the system depicted on figure 3 There exist two cars C_1 and C_2 and four sensors. R_1 is devoted to detect the movement of C_1 to the right direction; L_1 is devoted to detect the movement of C_1 to the left direction and the same for the R_2 and L_2 sensors for C_2.

Specification: The cars start in the leftmost position, then when input signal m is given, both cars start the motion to their right position. When C_1 (C_2) press switch c (d) the car C_1 (C_2) moves to left. Once car C_1 (C_2) press switch a (b), then it stops and remains in this state until both cars are in their leftmost positions and signal m is given again starting a new cycle again.

The IPN specification is also depicted on figure 3. This IPN is mono T-component system, and generates six output words, by the study above we conclude that we need only two input sequences $\sigma_1 = acbdm$ and $\sigma_2 = bdacm$, generating the output words $\delta_1 =$ $\begin{bmatrix} 1 \\ 1 \\ \epsilon \\ \epsilon \end{bmatrix} \begin{bmatrix} \epsilon \\ 1 \\ 1 \\ \epsilon \end{bmatrix} \begin{bmatrix} \epsilon \\ 1 \\ \epsilon \\ \epsilon \end{bmatrix} \begin{bmatrix} \epsilon \\ \epsilon \\ \epsilon \\ 1 \end{bmatrix} \begin{bmatrix} \epsilon \\ \epsilon \\ \epsilon \\ \epsilon \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ \epsilon \\ \epsilon \end{bmatrix}$ and

$\delta_2 = \begin{bmatrix} 1 \\ 1 \\ \epsilon \\ \epsilon \end{bmatrix} \begin{bmatrix} 1 \\ \epsilon \\ \epsilon \\ 1 \end{bmatrix} \begin{bmatrix} 1 \\ \epsilon \\ \epsilon \\ \epsilon \end{bmatrix} \begin{bmatrix} \epsilon \\ \epsilon \\ 1 \\ \epsilon \end{bmatrix} \begin{bmatrix} \epsilon \\ \epsilon \\ \epsilon \\ \epsilon \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ \epsilon \\ \epsilon \end{bmatrix}$ to

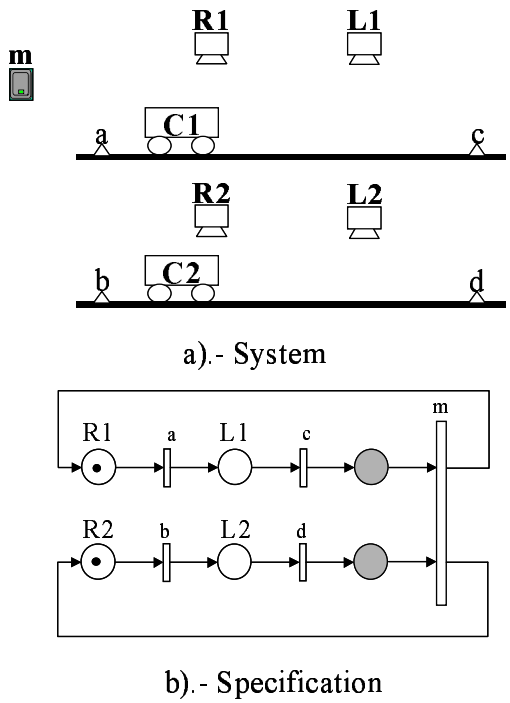


Fig. 3. The system and its specification

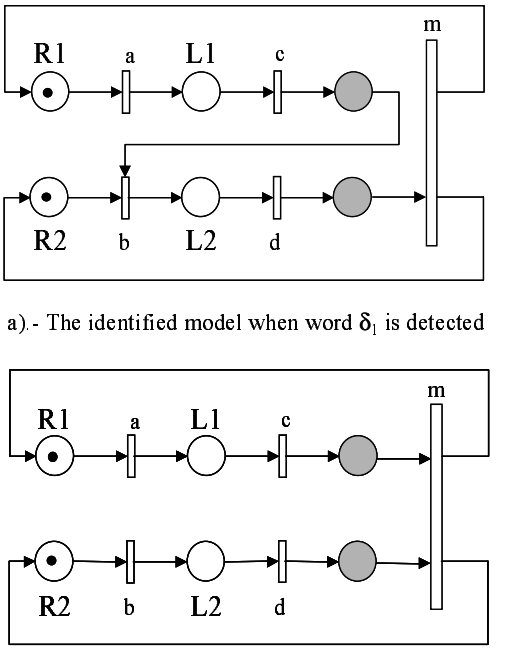


Fig. 4. The IPN models constructed when the words δ_1, δ_2 were detected from the system 3.

identify the designed DES.

Using the asymptotic identification algorithm (obtained from theorem 13) for the observed output word δ_1 when σ_1 is given as a input to the designed device, we construct the model depicted in figure 4.a).

Once the firing transition sequence σ_2 is given and δ_2 is observed, the algorithm yields to the model depicted in figure 4.b) which is the equal to the specification.

Notice that if the identified model is not equal to the specification, then the designed DES is wrong.

V. Conclusions and future work

This paper addressed the problems of on-line identification and design validation of discrete event systems. The proposed scheme for identification provides an algorithm allowing the progressive construction of IPN models; this approach avoids the NP complete problem found in techniques using positive samples of input languages. The application of this method to system validation yields an efficient correctness-checking technique for DES that provides a reduced set of test vector sequences.

Current research deals with extensions of the presented results to the cases of no binary IPN models and more complex non measurable dependencies.

References

[1] Dana Angluin. *Inference of Reversible Languages.* Journal of the Association for Computing Machinery, Vol. 29, No. 3, pp. 741-765, 1982.

[2] Dana Angluin. *Learning regular sets from queries and counter-examples.* Inform. Comput. 75, pp. 87-106, 1987

[3] John E. Hopcroft and Jefrey D. Ullman. *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley Publishing Company 1979, USA.

[4] E. Mark Gold. *Complexity of Automaton Identification from Given Data.* Information and Control, Vol. 37, pp. 302-320, 1978.

[5] M. Silva. *Petri Nets: in automatics and informatics.* (in Spanish) Ed. AC 1985. Madrid, Spain.

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

[7] M.E. Meda & A. Ramírez. *Identification in Discrete Event Systems.* IEEE-Systems Man and Cybernetics Conference, San Diego Ca., USA, pp. 747-745, 1998.

[8] Tadao Murata. *Synthesis of Decision-free Concurrent Systems for Prescribed Resources and Performance.* IEEE Trans. on Software Engineering, Vol. SE-6, No. 6, 525-530, 1980.

[9] Javier Esparza and Jörg Desel. *Free Choice Petri Nets.* Cambridge University Press 1995.

[10] A. Ramirez-Treviño, I. Rivera-Rangel and E. Lopez-Mellado. *Observer Design for Discrete Event Systems Modeled by Interpreted Petri Nets.* Proceedings of the IEEE ICRA, pp. 2871-287, San Francisco, USA. Apr. 2000.

[11] Chen, Chi-Tsong. *Linear System Theory and Design.* Harcourt Brace Jovanovich Inc. Sanders College Publishing. 1970.

[12] Sakakibara, Y. *Recent Advances of Grammatical Inference.* Theoretical Computer Science, 185, pp. 15-45, 1997.