Centro de Investigación y de Estudios Avanzados del I.P.N.

Unidad Guadalajara

Identificación de Sistemas de Eventos Discretos Concurrentes a partir de Secuencias de Entrada-Salida

Tesis que presenta:
Ana Paula Estrada Vargas

para obtener el grado de:
Maestro en Ciencias

en la especialidad de:
Ingeniería Eléctrica

Directores de Tesis
Dr. Luis Ernesto López Mellado
Prof. Jean-Jacques Lesage

Guadalajara, Jalisco, Agosto de 2009.

ID. 163425-1001

Centro de Investigación y de Estudios Avanzados
del I.P.N.

Unidad Guadalajara

Identification of Concurrent Discrete Event Systems from Input-Output Sequences

A thesis presented by:
Ana Paula Estrada Vargas

to obtain the degree of:
Master in Science

in the subject of:
Electrical Engineering

Thesis Advisors:
Dr. Luis Ernesto López Mellado
Prof. Jean-Jacques Lesage

Guadalajara, Jalisco, August 2009.

Identificación de Sistemas de Eventos Discretos Concurrentes a partir de Secuencias de Entrada-Salida

Tesis de Maestría en Ciencias
Ingeniería Eléctrica

Por:
Ana Paula Estrada Vargas
Ingeniero en Computación
Universidad de Guadalajara 2003-2007

Becario de CONACYT, expediente No. 50312

Directores de Tesis
Dr. Luis Ernesto López Mellado
Prof. Jean-Jacques Lesage

CINVESTAV del IPN Unidad Guadalajara, Agosto de 2009.

Identification of Concurrent Discrete Event Systems from Input-Output Sequences

Master of Science Thesis
In Electrical Engineering

by:
Ana Paula Estrada Vargas
Engineer in Computer Science
Universidad de Guadalajara 2003-2007

Sponsored by CONACYT, grant No. 50312

Thesis Advisors:
Dr. Luis Ernesto López Mellado
Prof. Jean-Jacques Lesage

CINVESTAV del IPN Unidad Guadalajara, August, 2009.

Identification of Concurrent Discrete Event Systems from Input-Output Sequences

Abstract

This thesis deals with identification of discrete event systems (DES). A method for building interpreted Petri net (IPN) models from observations of DES's inputs and outputs is proposed.

First, a survey on identification methods is presented. It includes an overview of previous works on the matter and a comparative study of three recent approaches for DES identification.

Then, the identification method is presented. It consists of several stages that build systematically an IPN model from input-output sequences representing the external behaviour of partially observable DES. Novel notions are introduced, algorithms and properties of the proposed method are included.

Finally, a software tool based on the presented algorithms is developed; it processes a set of input-output vector sequences yielding the drawing of the computed IPN model. The architecture and the main features of the tool are described and then the test on several case studies is presented.

Identificación de Sistemas de Eventos Discretos Concurrentes a partir de Secuencias Entrada-Salida

Resumen

Esta tesis trata sobre la identificación de Sistemas de Eventos Discretos (SED). Un método para construir modelos en redes de Petri interpretadas (RPI) a partir de observaciones de entradas y salidas del SED es propuesta.

Primeramente, una revisión de los métodos de identificación es presentada. Se incluye una reseña de trabajos previos sobre el tema y un estudio comparativo de tres enfoques recientes sobre identificación de SED.

Después, el método de identificación es presentado. Este consiste de varias etapas que construyen sistemáticamente un modelo en RPI a partir de secuencias de entrada-salida que representan el comportamiento externo observado de de SED parcialmente observables. Nuevas nociones son introducidas y algoritmos y propiedades del método propuesto.

Finalmente, una herramienta computacional basada en los algoritmos propuestos es desarrollada. La herramienta procesa un conjunto de secuencias de vectores de entrada-salida y grafica el modelo en RPI calculado. La arquitectura y las características principales de la herramienta son descritas y su prueba con varios casos de estudio es presentada.

Identification de Systèmes à Evénements Discrets Concurrents à partir des Séquences Entrée-Sortie

Résumé

Cette thèse traite de l'identification des systèmes à événements discrets (SED). Une méthode est proposée pour la construction de réseaux de Petri interprétés (RPI) à partir d'observations des entrées et des sorties du système identifié.

En premier lieu, une synthèse bibliographique sur les méthodes d'identification est présentée. Elle commence par un aperçu général des travaux antérieurs dans le domaine et se prolonge par une étude comparative de trois approches récentes sur l'identification de SED.

Notre méthode d'identification est ensuite présentée. Elle consiste en plusieurs étapes qui permettent de construire systématiquement un modèle RPI à partir des séquences d'entrée-sortie qui décrivent le comportement externe observé d'un SED partiellement observable. De nouvelles notions sont introduites et les algorithmes et propriétés de la méthode proposée sont décrits.

Finalement, un logiciel basé sur les algorithmes présentés est développé. Le logiciel traite un ensemble de séquences de vecteurs entrée-sortie, calcule et dessine le RPI identifié. L'architecture et les caractéristiques principales du logiciel sont décrites; plusieurs cas d'étude sont présentés.

Acknowledgements

This thesis has been developed partially during my stay at LURPA in ENS de Cachan.

I want to thank

To CINVESTAV for the opportunity of developing my Master of Sciences formation

To the ENS de Cachan for the opportunity to work in this collaboration and the financial support in France

To CONACYT for my scholarship during the MSc program

To Prof. Jean-Jacques Lesage for his aid and support during the development of this work

To Dr. Luis Ernesto López Mellado for all the knowledge he gave me, for his assistance throughout the thesis project and for his support during this collaboration

Index

Introduction

Analogously to identification of continuous dynamical systems, identification of discrete event systems (DES) consist of determining the mathematical model that describes the functioning of a DES from the observation of the evolution of inputs and outputs and other knowledge on the system behaviour.

The automated building of discrete event models from external observation of system behaviour has interesting applications such as reverse engineering for (partially) unknown systems, fault diagnosis or system verification. Identification of huge industrial systems requires software tools helping to handle large amount of input data and to draw complex models.

DES identification has been first addressed as a problem of grammatical inference. In [Gold, 1967] a finite automaton (FA) is built from positive samples of accepted words. Later several methods for obtaining Mealy [Kella, 1971] [Veelenturf, 1978] and Moore [Biermann and Feldman, 1972] [Veelenturf, 1981] machines have been proposed. Also building of context free grammars has been studied [Levy, 1978], [Takada, 1998], [Ishizaka, 1990].

Identification methods yielding Petri net (PN) models have been proposed for coping with more complex systems exhibiting concurrent behaviour. In [Hiraishi, 1992] an algorithm for constructing Petri net models is presented. First, the language of the target system is identified in the form of deterministic FA (DFA). Then, the algorithm obtains from the DFA the structure of a PN that accepts the obtained language.

Three different approaches have been adopted in recent publications addressing the problem of DES identification. The incremental synthesis approach proposed by Meda et al., deals with unknown partially measurable DES exhibiting cyclic behaviour; in [Meda, 1998] an identification method based on the least square estimator is presented. Then in [Meda, 2002][Meda, 2005] several algorithms have been proposed allowing the on-line identification of concurrent DES. Although the techniques are efficient, the obtained models may represent more sequences than those observed.

Another technique oriented to fault diagnosis is presented in [Klein, 2005]; it obtains non deterministic FA representing the same observed behaviour, which is represented as a set of input/output sequences. The algorithms operate off-line efficiently.

The off-line techniques based on integer linear programming (ILP) approach lead to free-labeled PN models representing exactly the observed behaviour [Giua, 2005]. However both the ILP problem statement from samples and their processing have exponential complexity. This approach is being applied to other PN classes in [Cabasino, 2006] [Fanti, 2008].

In this thesis, it is addressed the problem of identification of concurrent DES from stored input and output sequences describing the external observed behaviour of the system.

Firstly, different approaches adopted in recent publications are reviewed, and a comparative study is presented.

Afterwards, a method for the identification of Petri net models from observed output sequences including cyclic behaviour of concurrent partially observable DES is proposed. Because of the inherent capacity of IPN models to represent inputs and outputs of DES systems, internal states, non-controllable transitions and parallelism, the proposed method yields an IPN model which represents exactly the same language of length $\kappa+1$ than that generated by the system without taking into account information a-priori about the system other than its input and output signals.

The method gathers and extends some strategies from [Meda, 2002] and [Klein, 2005] and focuses on off-line case. In the first stage, a sample of output vectors words are processed for obtaining sequences of output changes called events; then sequences of κ-length event traces are considered allowing building an underlying IPN of non measurable places. The IPN orders the events by relating the measurable places according to pertinent output changes. Finally, the inputs are added and the model is simplified by eliminating implicit non-measurable places.

Furthermore, it is developed a software tool for the DES identification based on the presented algorithm; the tool has been implemented and tested with an experimental industrial machine.

The remainder of this thesis report is organized as follows:

- Chapter 1 is devoted to the analysis of existing identification techniques, including a comparative study of three recent methods.
- Chapter 2 presents some definitions about DES identification and explains the characteristics of the problem addressed in this work. An algorithm for Off-line DES identification from input-output sequences is presented, including an analysis of its principal properties and characteristics and some examples to illustrate the application of the method.
- Chapter 3 describes the main features of the design, implementation and use of a software tool developed for DES identification, which is based on the algorithm presented in Chapter 2.
- Finally, some concluding remarks and perspectives are given.

Chapter 1

Identification methods of discrete event systems

Abstract. This chapter surveys the identification techniques of discrete event systems found in the literature and analyses three recent approaches addressing the identification problem. A comparative study on such approaches is presented.

1.1. An overview of identification methods

The first results on DES identification appeared as a problem of grammatical inference [Gold, 1967] in which a finite automaton is built from positive samples of accepted words. Also there are approaches that identify context free grammars or Petri nets.

Within computational learning theory, there are three major established formal models for learning from examples or inductive inference:

1. Gold's model of identification in the limit [Gold, 1967], which uses for learning identification from positive data: an infinite sequence of examples such that the sequence contains all and only the strings in the language to learn.

2. The query learning model by [Angluin, 1988], which considers a learning protocol based on what is called "minimally adequate teacher" This teacher can answer two types of queries: membership query and equivalence query.

3. PAC learning model by [Valiant, 1984], which learns from random examples and study the effect of noise on learning from queries.

There are many different methods for identification originating from the field of computer science.

[Booth, 1967] presents a method to model a language as Moore or Mealy machines. The system under investigation is put in a test bed and connected to a so called experimenter who generates the input signals and records the output signals of the system. The identification can be started considering a very few number of states. If, at some point of the experiment, it is impossible to find a correct machine with the assumed number of states, the identification is started again considering a machine with one more state.

[Kella, 1971] presents a method to identify Mealy machines. The presented method does not require any a priori knowledge of the system and only a single observed sequence is available. The algorithm lists all reduced machines which may produce the input output sequence given. The construction principle is the merging of equivalent states.

[Biermann and Feldman, 1972] present a method to identify non deterministic Moore machines based on a set of input output sequences. All the sequences start in the

same initial state. The identification principle is the reduction of an initial machine represented as a tree.

[Veelenturf, 1978] presents a method manipulating simultaneously a sample of sequences to produce a convergent series of Mealy machines such that the behaviour of every new machine includes the behaviour of the previous one. The automaton is built step by step. At each step, the already available machine is examined and completed by adding transitions and maybe new states.

[Veelenturf, 1981] proposes an algorithm to identify a unique Moore machine generating the behaviour observed during m sequences starting in the same initial state. The learning procedure requires three parts: induction, contradiction and discrimination. A state can never be deleted and only transitions between states can be modified.

[Richetin, 1984] improve the method and propose two algorithms to identify multiple systems as well as systems that may not be initialized between two records.

The identification problem for context free grammars needs, beside given examples, some additional structural information for the inference algorithm [Levy and Joshi, 1978]. [Takada, 1998] has shown that the inference problem for even linear grammars can be solved by reducing it to one for DFAs, and presented a polynomial time algorithm for the reduction. [Ishizaka, 1990] has investigated a subclass of CFGs called simple deterministic grammars and gave a polynomial time algorithm for exactly identifying it using equivalence and membership queries in terms of general CFGs.

[Hiraishi, 1992] presented an algorithm for constructing Petri net models. The proposed algorithm has two phases. In the first phase, the language of the target system is identified in the form of DFA. In the second phase, the algorithm guesses from the DFA the structure of a Petri net that accepts the obtained language. In [Meda, 1998] is presented an identification method based on the least square estimator.

[Giua, 2005] presents a method to identify a free labelled Petri net system from the knowledge of a finite set of strings that it generates. The method is based in the solution of an Integer Linear Programming problem. The identified Petri net generates exactly the given language thanks to the creation of examples and counter examples made by the algorithm. Besides this work, there are other approximations like [Cabasino, 2006a] that apply the ILP technique to identify different types of Petri nets.

All methods mentioned above deal with the modelling of a given language using different representations. The works are represented in figure 1.1 in which a classification, discussed later, is given.

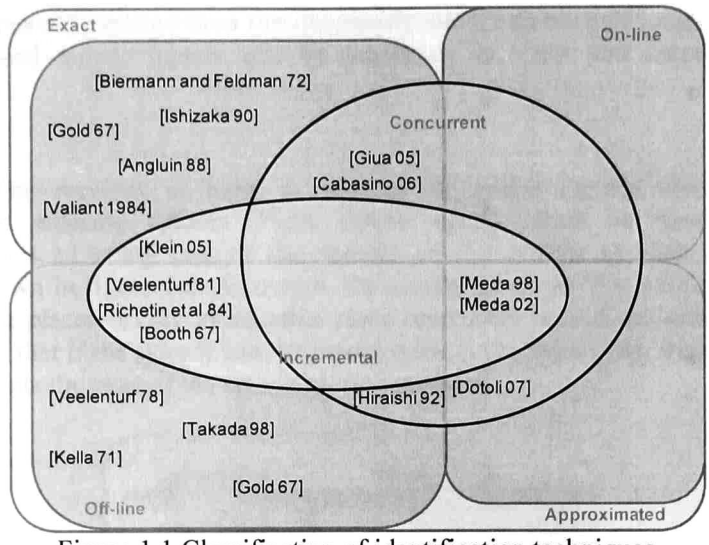

Figure 1.1 Classification of identification techniques

1.2. Recent approaches to identification

In this section we overview three different approaches adopted in recent publications addressing the problem of DES identification. The first approach deals with unknown partially measurable DES exhibiting cyclic behaviour. The second approach is off-line; it is oriented to obtain models that can help to fault diagnosis. The third approach is also off-line; it deals with DES that does not necessarily have binary outputs.

1.2.1. Progressive synthesis

Problem

The problem addressed in this work is to build a model for a DES as it evolves from the observation of its output signals [Meda, 2002]. A sequence of models is built in such a way that the current model acquires more details than the previous one approaching to the actual model of the system. This work is an approximation of the identification problem in DES and it can be considered as a basis for future works on the matter, possibly oriented towards the verification of systems, hardware or software, or it can be extended to address problems of reverse engineering.

Approach

The identification approach proposes to compute an Interpreted Petri Net (IPN) model describing the behaviour of the unknown DES.

Some assumptions are considered on the type of systems to be identified:

- The systems to be identified are DES that can be described by a live, binary and cyclic IPN Q.
- Q is an event-detectable IPN.

- The transitions of Q are not fired simultaneously and Q has not self-loops.
- The input and output signals will be sequences of input and output symbols respectively.

Methodology

The algorithm receives as input a sequence of output signals obtained from observation of the working system. These output signals must be binary vectors representing the state of every one of the sensors of the system at each time. This methodology is shown in figure 1.2. As output, the system gives an IPN with measurable and non-measurable places. Every measurable place represents one of the sensors of the system. That means that if the system and the model work at the same time, the marking of the Petri net represents the state of the system at each moment.

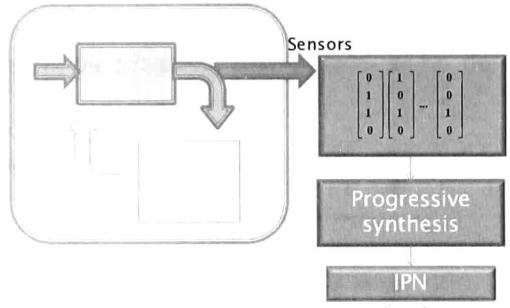

Figure 1.2 Progressive synthesis

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

During the on-line operation of the identification process, the m-words are computed and then the new model is built adding, removing, or updating dependencies between the transitions.

The model synthesis procedure performs mainly two tasks: the computation of the measurable part of the system and the inference of the non measurable part of the system, which is related with the dependencies formed by the non measurable places with respect to the computed transitions. The first task is made directly from the observation of the output system signals, while the second task, rather difficult, derived a more detailed study about the dependencies formed by a non measurable place into a model.

- **Algorithm: Asymptotic identification**

1. Read the vectors of output symbols generated by the system.
2. Detect an output word when the first and last output symbols are the same.
3. For any two consecutive output symbols compute a transition that represents the output change (if the output was calculated before, take the same transition).
4. Compute an m-word adding each computed transition in the step above.
5. Compute the non measurable places.
   a. to constrain the firing order of the transitions to the order in which they were computed

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

6. Update the computed IPN model with the information provided by the m-word allowing the firing of all computed m-words, inferring t-semiflows of the system.
   a. computing new measurable places and transitions
   b. removing or adding dependencies (possibly merging places) updating the computed real t-semiflows.

Example 1.1 In order to illustrate the method, we take from [Meda, 2003] the following example of a system with 11 output signals. We show the models generated when new m-words are computed from the outputs of the system. For sake of brevity, not all steps are shown.

Step 1. First output symbols:

$o_1 = [00000000000]^T, o_2 = [10000000000]^T, o_3 = [00000000000]^T = o_1, o_4 = ...$

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

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

Step 4. The m-word resulting is $m_1 = t_1 t_2$

Step 5. The t-component associated with the m-word $t_1 t_2$ is shown in figure 1.3

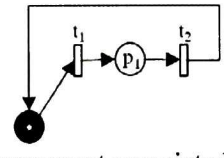

Figure 1.3 t-component associated with $m_1 = t_1 t_2$

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

After the next output word is treated with steps 1-4, it is obtained the m-word $m_2 = t_3 t_4$. Its respective t-component associated is added to infer a new t-semiflow $W_1 = m_1 m_2$ in step 6, as shown in figure 1.4.

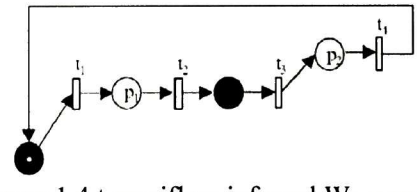

Figure 1.4 t-semiflow inferred $W_1 = m_1 m_2$

After computing of m-words $m_3 = t_6 t_7$, $m_4 = t_5 t_8$, $m_5 = t_9 t_{10}$, $m_6 = t_{11} t_{12}$, $m_7 = t_{13} t_{14}$, it is inferred in step 6 the t-semiflow $W_1 = m_1 m_2 m_3 m_4 m_5 m_6 m_7$ shown in figure 1.5.

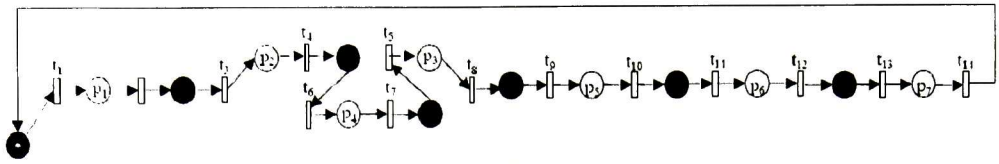

Figure 1.5 t-semiflow inferred $W_1 = m_1m_2m_3m_4m_5m_6m_7$

The arriving m-word $m_1 = t_1t_2$ is the first one of $W_1 = m_1m_2m_3m_4m_5m_6m_7$. Then, it is supposed that W_1 has been completely observed and a new t-semiflow $W_2 = m_1$ is inferred. Observed m-words $m_2 = t_3t_4$, and $m_{3\text{-}4} = t_5t_6t_7t_8$, in step 4 are added to the t-semiflow W_2 and the model is updated in step 6 to allow the firing of all of them, as shown in figure 1.6.

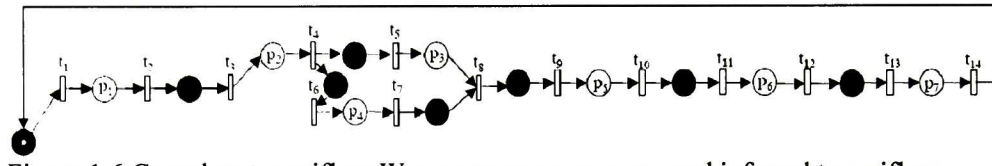

Figure 1.6 Complete t-semiflow $W_1 = m_1m_2m_3m_4m_5m_6m_7$ and inferred t-semiflow $W_2 = m_1m_2m_{3\text{-}4}$

The m-word $m_7 = t_{13}t_{14}$ is observed. It is made a merging of places to allow the firing of the m-words observed in the order the appeared. A new t-semiflow $W_3 = m_5m_6$ is inferred and t-semiflows $W_1 = m_1m_2m_3m_4m_7$ and $W_2 = m_1m_2m_{3\text{-}4}m_7$ are updated (figure 1.7).

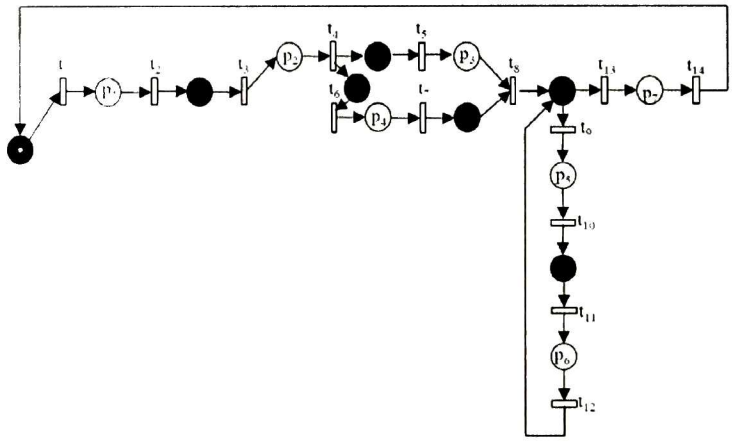

Figure 1.7 t-semiflows $W_1 = m_1m_2m_3m_4m_7$, $W_2 = m_1m_2m_{3\text{-}4}m_7$ and $W_3 = m_5m_6$

The m-word $m_8 = t_{15}t_{16}$, is observed and a new t-semiflow $W_4 = m_8$ is inferred. When m-words $m_9 = t_{17}t_{18}$ and $m_{10} = t_{21}t_{22}$ arrive, they are added, as shown, in figure 1.8

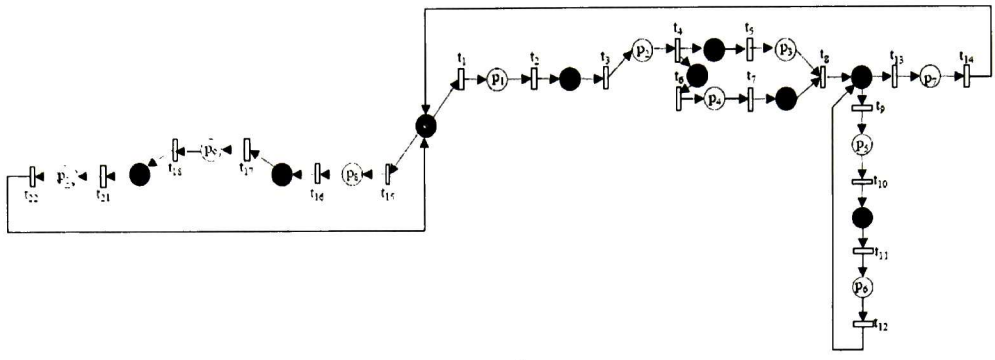

Figure 1.8 Adding of t-semiflow $W_4 = m_8m_9m_{10}$

The m-word $m_7 = t_{13}t_{14}$, is observed again and a merging of places is made. T-semiflows $W_1 = m_1m_2m_3m_4$ and $W_2 = m_1m_2m_{3\text{-}4}$ are updated and $W_5 = m_7$ is created. These changes can be seen in figure 1.9.

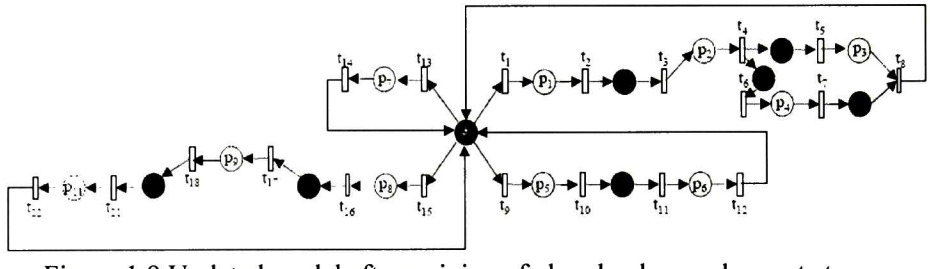

Figure 1.9 Updated model after arriving of already observed $m_7 = t_{13}t_{14}$

When m-word $m_8 = t_{15}t_{16}$ is observed, no changes are made on the model. But when $m_{11} = t_{19}t_{20}$ is observed, a new t-semiflow $W_6 = m_8m_{11}$ is inferred, as observed in figure 1.10.

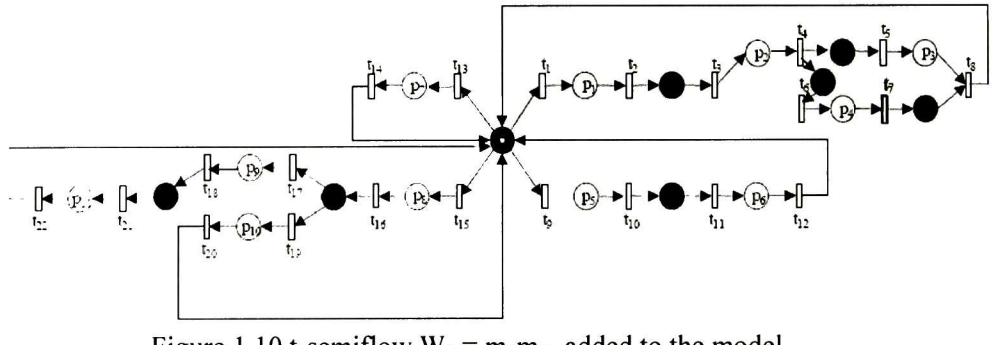

Figure 1.10 t-semiflow $W_6 = m_8m_{11}$ added to the model

The last m-word $m_{10} = t_{21}t_{22}$ is observed and the t-semiflow $W_6 = m_8m_{11}m_{10}$ is updated. The final model can be seen in figure 1.11.

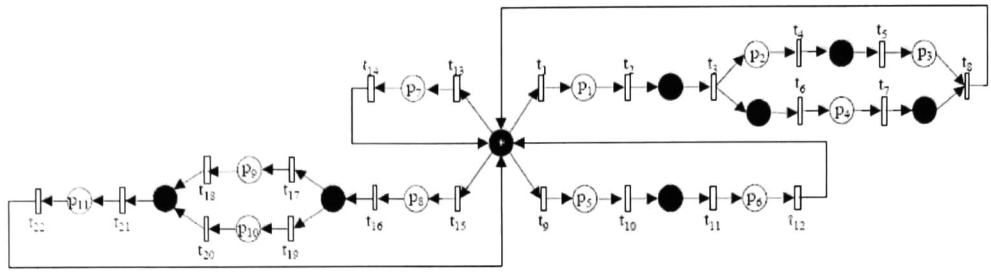

Figure 1.11 Final model for identification

- **Complexity**

The proposed algorithms to update the non measurable places have linear complexity on the number of the transitions computed and the m-words detected, that is, in the size of the identified model.

Then, the general algorithm to update a model that includes all the updating procedures of non measurable places is executed also in polynomial time.

- **Limitations**

In some cases, the obtained model may represent an exceeding behaviour with respect to the observed from the system. Additionally the structures describing the following behaviour cannot be identified using this methodology:

- Simultaneous firing of transitions
- Self loops
- Shared resources in mutual exclusion
- Implicit non-measurable places

1.2.2. Off-line automata construction

- **Problem**

In this work [Klein 2005] is built a finite automaton form a given set of sequences, of the inputs and outputs evolution from the system. The method was proposed for obtaining models adapted for fault detection in a model-based approach.

Approach

The identification approach proposes to compute an NDAAO model describing the behaviour of the unknown DES. The definition of the NDAAO will be presented below.

The system to be identified is a compound system controller + plant running in a closed-loop considered as a generator or an information source.

Methodology

The algorithm receives as input a set of observed production cycles obtained from the system to be identified. Each observed production cycle or observed sequence is an ordered series of input/output (I/O) binary vectors at different times. The observed

sequences do not necessarily have the same length; however, the first and last I/O vectors of different sequences have to be identical. The identification schema is shown in figure 1.12

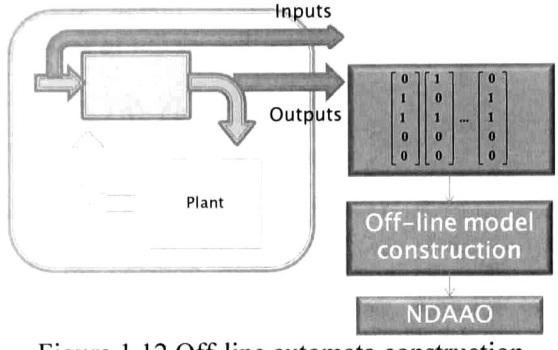

Figure 1.12 Off-line automata construction

As output, the system gives a Non-Deterministic Autonomous Automaton with Output (NDAAO). Below, this term is explained. Each state of the NDAAO gives as output a possible state of the system, representing with a binary vector every one of the observed signals of the system.

The construction principle is to associate each different observed I/O vector with a single state. The transitions between the different states are created after a path between the corresponding I/O vectors has been observed.

- **Algorithm**

First, we present some definitions taken from [Klein, 2005a].

A non-deterministic autonomous automaton with output, denoted NDAAO, is a five-tuple:

NDAAO = $(X, \Omega, r, \lambda, x_0)$

X	finite set of states,
Ω	output alphabet,
$r : X \rightarrow 2^X$	non deterministic transition relation,
$\lambda : X \rightarrow \Omega$	output function,
$x_0 \in X$	initial state

Each observed production cycle of the system – also referred to as an observed sequence – is noted σ_i and formally defined as $\sigma_i = (u_i(1), u_i(2) \ldots u_i(|\sigma_i|))$ where $|\sigma_i|$ represents the length of the considered sequence.

The cyclic production implies that the first and the last observed I/O vectors of the different sequences are identical. This can be formulated as $\forall(i, j), u_i(1) = u_j(1) = u_i(|\sigma i|) = u_j(|\sigma j|)$.

The algorithm proceeds in six steps:

1. For each observed sequence σ_i, construction of sequences of k vectors $u_i(t)$ where k is the a priori fixed parameter.
2. Construction of the NDAAO.
3. Renaming of the output function.
4. Reduction of the last state.

5. Merging of equivalent states.
6. Closure of the automaton.

Example 1.2 Let us consider the example of an elementary plant with a controller having two inputs and one output [Klein, 2005a].

The observed sequences of I/O vectors are:

$$\sigma_1 = \left(\begin{pmatrix}0\\0\\0\end{pmatrix},\begin{pmatrix}0\\1\\0\end{pmatrix},\begin{pmatrix}1\\1\\1\end{pmatrix},\begin{pmatrix}0\\1\\1\end{pmatrix},\begin{pmatrix}0\\0\\1\end{pmatrix},\begin{pmatrix}0\\0\\0\end{pmatrix}\right) \quad \sigma_2 = \left(\begin{pmatrix}0\\0\\0\end{pmatrix},\begin{pmatrix}1\\1\\1\end{pmatrix},\begin{pmatrix}0\\1\\0\end{pmatrix},\begin{pmatrix}1\\1\\1\end{pmatrix},\begin{pmatrix}0\\1\\1\end{pmatrix},\begin{pmatrix}0\\0\\0\end{pmatrix}\right)$$

In order to simplify the notation, each I/O vector is coded as A, B, C, D or E. These letters represent the letters of the observed alphabet. With this coding, the observed sequences are: σ_1 = (A, B, C, D, E, A) and σ_2 = (A, C, B, C, D, A).

Step 1: Construction of vector sequences. Setting $k = 2$, we obtain for the example:

σ_1^2 = ((A,A),(A,B),(B,C),(C,D),(D,E),(E,A), (A,A))

σ_2^2 = ((A,A),(A,C),(C,B),(B,C),(C,D),(D,A), (A,A))

Step 2: Construction of the NDAAO. The identification principle is to associate each different word with a single state. This step is illustrated by figure 1.13

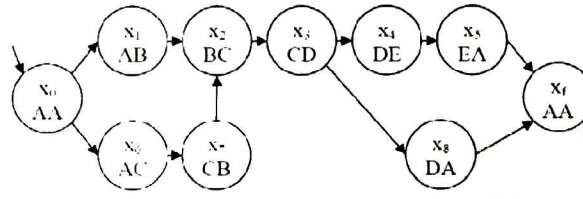

Figure 1.13 Association of words with states of the NDAAO

Step 3: Renaming of the output function. Each state of the NDAAO corresponds to a unique and stable value of the input and output signals. This value is described by the last letter of each sequence of length k, as shown in figure 1.14

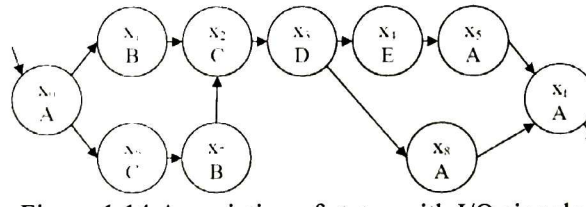

Figure 1.14 Association of states with I/O signals

Step 4: Reduction of the last state. The last k states of each branch ending with x_f are associated with the same letter. These states can be reduced with a procedure that has to be iterated $k - 1$ times. First, merge the pre-states of x_f. Second, redefine this new state as the final state x_f and delete the former x_f from the set of states. This procedure is illustrated in figure 1.15

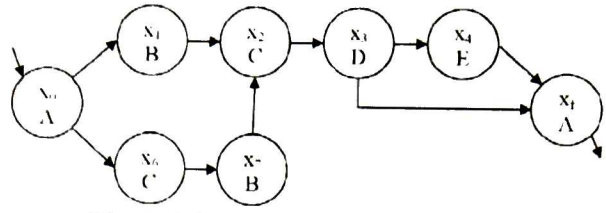

Figure 1.15 Reduction of the last state

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

1. They are associated with the same output
2. They have the same set of post states.

It has been proved that the merging of equivalent states does not affect the languages accepted by the NDAAO. This property can be seen in figure 1.16

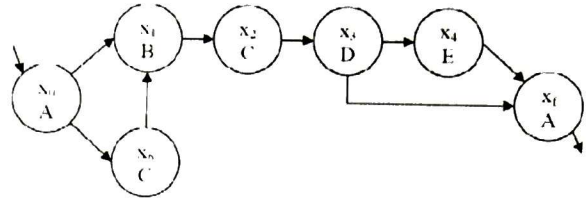

Figure 1.16 Merging of equivalent states

Step 6: Closure of the automaton. With the hypothesis that each observed sequence corresponds to a single production cycle, the states x_0 and x_f of the NDAAO identified are identical. Thus the NDAAO can be closed resulting in a strongly connected NDAAO, as observed in figure 1.17

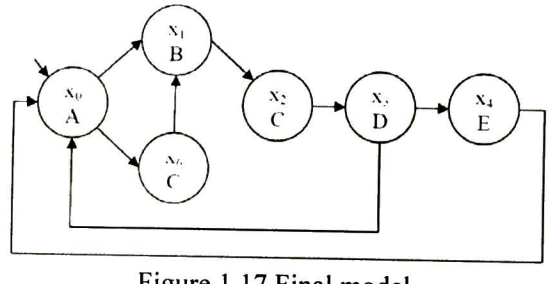

Figure 1.17 Final model

- **Complexity**

The time required to build different models is very low and does not represent any problems for the application of the identification method. However, the reduction of the NDAAO requires more time than the identification of the model.

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

- **Limitations**

The merging of equivalent states could take much time, and the concurrency is not represented by the obtained model.

1.2.3. Integer Linear Programming approach

- **Problem**

The problem to be solved is the DES identification inferring a Petri Net model using the observation of events and the available output vectors. Identification is a classical problem in system theory: given a pair of observed input-output signals it consists in determining a system such that the input-output signals approximate the observed ones.

- **Approach**

The identification approach proposes to compute an IPN model such that the observed sequence of events belongs to the language accepted by the Petri net and the subset of measurable places has a given cardinality. The method observes several hypothesis.

A1. All the DES events can be detected, distinguished and not silent.
A2. The DES can be (partially) observed.
A3. The DES can be modelled by a PN system with λ-free labelling function.
A4. The set of measurable places has cardinality q.
A5. There is an upper bound on the number of places of the Petri net.

- **Methodology**

The algorithm receives as input sequences of events with their corresponding output vectors. Also, it is needed an upper bound of the number of non-measurable places for the net to identify. The schema is shown in figure 1.18.

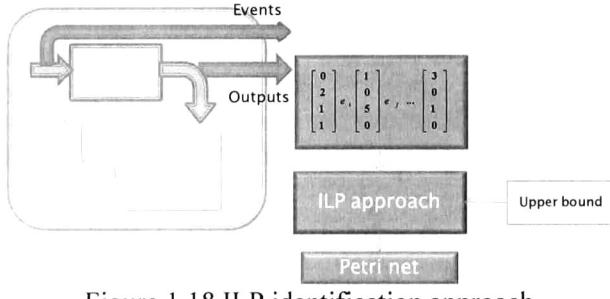

Figure 1.18 ILP identification approach

As output, the system gives a Petri Net with measurable places representing the sensors of the system, non-measurable places and labelled transitions representing the observed events. It is also possible that the algorithm returns a 0 (zero) when there is no possible solution of the problem with the given input.

The strategy of the algorithm is to generate an Integer Linear Programming (ILP) problem adding one linear algebraic constraint for every one of the characteristics that must have a Petri net. To select among the solutions could be given by the problem solver, it is

given a performance index, generally trying to minimize the arcs weights and the initial marking of the Petri net.

It is also possible to force the net to satisfy some structural constraints by adding linear constraints to the ILP.

- **Algorithm**

First, we present some definitions taken from [Dotoli, 2008].

Let us define the PN set $D = \{PN = (P, T, \boldsymbol{Pre}, \boldsymbol{Post}): \boldsymbol{Pre} \in \mathbb{N}^{mxn}, \boldsymbol{Post} \in \mathbb{N}^{mxn}\}$.

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

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

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

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

The first two constraints are derived by the definition of pre and post incidence matrices as well as of markings. The third and fourth linear algebraic constraints are to avoid isolated transitions and places respectively. The fifth and sixth constraints are related with enabling and firing of transitions respectively.

Some constraints can be added if additional structural properties are known on the PN model to identify. For example, if there is no place without successor transitions it can be added:

$$\boldsymbol{Pre} \cdot \vec{1}_{n\times 1} \geq \vec{1}_{m\times 1}$$

If there is no transition without successor places:

$$\boldsymbol{Post}^{\mathrm{T}} \ \vec{1}_{m\times 1} \geq \vec{1}_{n\times 1}$$

If there are no source transitions:

$$\boldsymbol{Pre}^{\mathrm{T}} \ \vec{1}_{m\times 1} \geq \vec{1}_{n\times 1}$$

Since there is not only one PN satisfying the constraint set, it is used a performance index, an indicator of the PN size, as a linear function.

$$\phi(\boldsymbol{Pre}, \boldsymbol{Post}, \mathbf{M}_0) = \vec{\mathbf{a}}^{\mathrm{T}} \boldsymbol{Pre}\, \vec{\mathbf{b}} + \vec{\mathbf{c}}^{\mathrm{T}} \boldsymbol{Post}\, \vec{\mathbf{d}} + \vec{\mathbf{e}}^{\mathrm{T}} \mathbf{M}_0$$

It is presented now the basis of the algorithm that solves the identification problem stated above. The complete algorithm and a best explanation of the solution are given in [Dotoli, 2008].

1. Initialization of the algorithm variables. It is obtained the first output vector and are initialized the set of labels, the set of transitions, and the set of output vectors. Every time it is calculated more information, these sets are actualized.

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

3. Associate a transition to an event.

3.1 The event occurs for the first time. A new transition is created and associated to the event and the observed change of marking.

3.2 If the event occurred previously

3.2.1 A new transition must be associated with the event (if there is no change in the marking associated to any transition).

3.2.2 A fired transition is associated to the event.

4. Solve the ILP problem

$$min\ \phi(\boldsymbol{Pre}_w, \boldsymbol{Post}_w, \boldsymbol{M}_{0w})\ \text{s.t.}\ \xi(w, \mathbf{Y}, \lambda_w, T_w, m')$$

as many times as necessary, starting with m' equal to the number of measurable places and incrementing it, until it is found a solution or until m' is equal to the upper bound of the number of places.

5. Return to the condition of recording the events.

Example 1.3 It is taken from [Dotoli, 2008] an example.

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

Minimize

$$[1 \quad 1 \quad 1 \quad 1 \quad 1](Pre + Post)\begin{bmatrix}1\\1\\1\\1\end{bmatrix} + [1 \quad 1 \quad 1 \quad 1 \quad 1]M_0$$

subject to:

1) $Pre, Post \in \mathbb{N}^{m \times n}$

$$Pre = \begin{bmatrix} pre_{11} & pre_{21} & pre_{31} & pre_{41} \\ pre_{12} & pre_{22} & pre_{32} & pre_{42} \\ pre_{13} & pre_{23} & pre_{33} & pre_{43} \\ pre_{14} & pre_{24} & pre_{34} & pre_{44} \\ pre_{15} & pre_{25} & pre_{35} & pre_{45} \end{bmatrix}, Post = \begin{bmatrix} post_{11} & post_{21} & post_{31} & post_{41} \\ post_{12} & post_{22} & post_{32} & post_{42} \\ post_{13} & post_{23} & post_{33} & post_{43} \\ post_{14} & post_{24} & post_{34} & post_{44} \\ post_{15} & post_{25} & post_{35} & post_{45} \end{bmatrix}$$

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

$$M_0 = \begin{bmatrix} m_{0_1} \\ m_{0_2} \\ m_{0_3} \\ m_{0_4} \\ m_{0_5} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 2 \end{bmatrix}, M_1 = \begin{bmatrix} m_{1_1} \\ m_{1_2} \\ m_{1_3} \\ m_{1_4} \\ m_{1_5} \end{bmatrix} = \begin{bmatrix} 4 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}, M_2 = \begin{bmatrix} m_{2_1} \\ m_{2_2} \\ m_{2_3} \\ m_{2_4} \\ m_{2_5} \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}, M_3 = \begin{bmatrix} m_{3_1} \\ m_{3_2} \\ m_{3_3} \\ m_{3_4} \\ m_{3_5} \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 1 \end{bmatrix}, M_4 = \begin{bmatrix} m_{4_1} \\ m_{4_2} \\ m_{4_3} \\ m_{4_4} \\ m_{4_5} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 2 \end{bmatrix}$$

3) $Post^T \vec{1}_{m \times 1} + Pre^T \vec{1}_{m \times 1} \geq \vec{1}_{n \times 1}$

$$\begin{bmatrix} post_{11} & post_{12} & post_{13} & post_{14} & post_{15} \\ post_{21} & post_{22} & post_{23} & post_{24} & post_{25} \\ post_{31} & post_{32} & post_{33} & post_{34} & post_{35} \\ post_{41} & post_{42} & post_{43} & post_{44} & post_{45} \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} + \begin{bmatrix} pre_{11} & pre_{12} & pre_{13} & pre_{14} & pre_{15} \\ pre_{21} & pre_{22} & pre_{23} & pre_{24} & pre_{25} \\ pre_{31} & pre_{32} & pre_{33} & pre_{34} & pre_{35} \\ pre_{41} & pre_{42} & pre_{43} & pre_{44} & pre_{45} \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \geq \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

4) $Post\vec{1}_{n\times 1} + Pre\vec{1}_{n\times 1} \geq \vec{1}_{m\times 1}$

$$\begin{bmatrix} post_{11} & post_{21} & post_{31} & post_{41} \\ post_{12} & post_{22} & post_{32} & post_{42} \\ post_{13} & post_{23} & post_{33} & post_{43} \\ post_{14} & post_{24} & post_{34} & post_{44} \\ post_{15} & post_{25} & post_{35} & post_{45} \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} + \begin{bmatrix} pre_{11} & pre_{21} & pre_{31} & pre_{41} \\ pre_{12} & pre_{22} & pre_{32} & pre_{42} \\ pre_{13} & pre_{23} & pre_{33} & pre_{43} \\ pre_{14} & pre_{24} & pre_{34} & pre_{44} \\ pre_{15} & pre_{25} & pre_{35} & pre_{45} \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \geq \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

5) $\forall t_{\beta_i}^{\alpha_i} \in \sigma$ with $\lambda(\sigma) = w$, $Pre\vec{t}_{\beta_i}^{\alpha_i} \leq M_{i-1}$

$$\begin{bmatrix} pre_{11} & pre_{21} & pre_{31} & pre_{41} \\ pre_{12} & pre_{22} & pre_{32} & pre_{42} \\ pre_{13} & pre_{23} & pre_{33} & pre_{43} \\ pre_{14} & pre_{24} & pre_{34} & pre_{44} \\ pre_{15} & pre_{25} & pre_{35} & pre_{45} \end{bmatrix} \begin{bmatrix} t_{\beta_1 1}^{\alpha_1} \\ t_{\beta_1 2}^{\alpha_1} \\ t_{\beta_1 3}^{\alpha_1} \\ t_{\beta_1 4}^{\alpha_1} \end{bmatrix} \leq \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 2 \end{bmatrix}, \begin{bmatrix} pre_{11} & pre_{21} & pre_{31} & pre_{41} \\ pre_{12} & pre_{22} & pre_{32} & pre_{42} \\ pre_{13} & pre_{23} & pre_{33} & pre_{43} \\ pre_{14} & pre_{24} & pre_{34} & pre_{44} \\ pre_{15} & pre_{25} & pre_{35} & pre_{45} \end{bmatrix} \begin{bmatrix} t_{\beta_2 1}^{\alpha_2} \\ t_{\beta_2 2}^{\alpha_2} \\ t_{\beta_2 3}^{\alpha_2} \\ t_{\beta_2 4}^{\alpha_2} \end{bmatrix} \leq \begin{bmatrix} 4 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix},$$

$$\begin{bmatrix} pre_{11} & pre_{21} & pre_{31} & pre_{41} \\ pre_{12} & pre_{22} & pre_{32} & pre_{42} \\ pre_{13} & pre_{23} & pre_{33} & pre_{43} \\ pre_{14} & pre_{24} & pre_{34} & pre_{44} \\ pre_{15} & pre_{25} & pre_{35} & pre_{45} \end{bmatrix} \begin{bmatrix} t_{\beta_3 1}^{\alpha_3} \\ t_{\beta_3 2}^{\alpha_3} \\ t_{\beta_3 3}^{\alpha_3} \\ t_{\beta_3 4}^{\alpha_3} \end{bmatrix} \leq \begin{bmatrix} 3 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} pre_{11} & pre_{21} & pre_{31} & pre_{41} \\ pre_{12} & pre_{22} & pre_{32} & pre_{42} \\ pre_{13} & pre_{23} & pre_{33} & pre_{43} \\ pre_{14} & pre_{24} & pre_{34} & pre_{44} \\ pre_{15} & pre_{25} & pre_{35} & pre_{45} \end{bmatrix} \begin{bmatrix} t_{\beta_4 1}^{\alpha_4} \\ t_{\beta_4 2}^{\alpha_4} \\ t_{\beta_4 3}^{\alpha_4} \\ t_{\beta_4 4}^{\alpha_4} \end{bmatrix} \leq \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

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

$$\left(\begin{bmatrix} post_{11} & post_{21} & post_{31} & post_{41} \\ post_{12} & post_{22} & post_{32} & post_{42} \\ post_{13} & post_{23} & post_{33} & post_{43} \\ post_{14} & post_{24} & post_{34} & post_{44} \\ post_{15} & post_{25} & post_{35} & post_{45} \end{bmatrix} - \begin{bmatrix} pre_{11} & pre_{21} & pre_{31} & pre_{41} \\ pre_{12} & pre_{22} & pre_{32} & pre_{42} \\ pre_{13} & pre_{23} & pre_{33} & pre_{43} \\ pre_{14} & pre_{24} & pre_{34} & pre_{44} \\ pre_{15} & pre_{25} & pre_{35} & pre_{45} \end{bmatrix} \right) \begin{bmatrix} t_{\beta_1 1}^{\alpha_1} \\ t_{\beta_1 2}^{\alpha_1} \\ t_{\beta_1 3}^{\alpha_1} \\ t_{\beta_1 4}^{\alpha_1} \end{bmatrix} = \begin{bmatrix} 4 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 2 \end{bmatrix},$$

$$\left(\begin{bmatrix} post_{11} & post_{21} & post_{31} & post_{41} \\ post_{12} & post_{22} & post_{32} & post_{42} \\ post_{13} & post_{23} & post_{33} & post_{43} \\ post_{14} & post_{24} & post_{34} & post_{44} \\ post_{15} & post_{25} & post_{35} & post_{45} \end{bmatrix} - \begin{bmatrix} pre_{11} & pre_{21} & pre_{31} & pre_{41} \\ pre_{12} & pre_{22} & pre_{32} & pre_{42} \\ pre_{13} & pre_{23} & pre_{33} & pre_{43} \\ pre_{14} & pre_{24} & pre_{34} & pre_{44} \\ pre_{15} & pre_{25} & pre_{35} & pre_{45} \end{bmatrix} \right) \begin{bmatrix} t_{\beta_2 1}^{\alpha_2} \\ t_{\beta_2 2}^{\alpha_2} \\ t_{\beta_2 3}^{\alpha_2} \\ t_{\beta_2 4}^{\alpha_2} \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} - \begin{bmatrix} 4 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix},$$

$$\left(\begin{bmatrix} post_{11} & post_{21} & post_{31} & post_{41} \\ post_{12} & post_{22} & post_{32} & post_{42} \\ post_{13} & post_{23} & post_{33} & post_{43} \\ post_{14} & post_{24} & post_{34} & post_{44} \\ post_{15} & post_{25} & post_{35} & post_{45} \end{bmatrix} - \begin{bmatrix} pre_{11} & pre_{21} & pre_{31} & pre_{41} \\ pre_{12} & pre_{22} & pre_{32} & pre_{42} \\ pre_{13} & pre_{23} & pre_{33} & pre_{43} \\ pre_{14} & pre_{24} & pre_{34} & pre_{44} \\ pre_{15} & pre_{25} & pre_{35} & pre_{45} \end{bmatrix} \right) \begin{bmatrix} t_{\beta_3 1}^{\alpha_3} \\ t_{\beta_3 2}^{\alpha_3} \\ t_{\beta_3 3}^{\alpha_3} \\ t_{\beta_3 4}^{\alpha_3} \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 3 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix},$$

$$\left(\begin{bmatrix} post_{11} & post_{21} & post_{31} & post_{41} \\ post_{12} & post_{22} & post_{32} & post_{42} \\ post_{13} & post_{23} & post_{33} & post_{43} \\ post_{14} & post_{24} & post_{34} & post_{44} \\ post_{15} & post_{25} & post_{35} & post_{45} \end{bmatrix} - \begin{bmatrix} pre_{11} & pre_{21} & pre_{31} & pre_{41} \\ pre_{12} & pre_{22} & pre_{32} & pre_{42} \\ pre_{13} & pre_{23} & pre_{33} & pre_{43} \\ pre_{14} & pre_{24} & pre_{34} & pre_{44} \\ pre_{15} & pre_{25} & pre_{35} & pre_{45} \end{bmatrix} \right) \begin{bmatrix} t_{\beta_4 1}^{\alpha_4} \\ t_{\beta_4 2}^{\alpha_4} \\ t_{\beta_4 3}^{\alpha_4} \\ t_{\beta_4 4}^{\alpha_4} \end{bmatrix} \leq \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 2 \end{bmatrix} - \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 1 \end{bmatrix}.$$

The IPN obtained is illustrated in figure 1.19

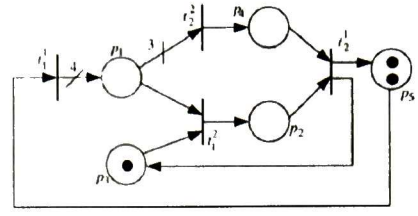

Figure 1.19 Solution for identification problem of Example 1.3

- **Complexity**

In the worst case the number of unknowns is linear with the number of places, of transitions, and with the length h of the firing sequence. Investigating the structure of the problem is a difficult task, however in the examined cases an optimal solution is obtained in a short time implementing and solving the ILP problem on a PC equipped with a standard solver of optimization problems, for example GLPK.

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

- **Limitations**

It is necessary the number upper bound on the number of places.
Integer Linear Programming is a NP problem.
It is not always found a solution.

1.3. Analysis of identification techniques

Below, there is a comparative analysis of the techniques mentioned above. First, are explained the methods features to be considered. Later, it is presented the analysis of every one of the methods. Finally, the information is summarized in a comparative table.

1.3.1. Methods features

Many of these features have been considered by [Klein, 2005]. Some others are added to have a more complete comparative analysis. An identification method can be:

- **On-line.** The construction of the model is performed while the system is working by computing a new model from new measurements of the system outputs. Otherwise, the identification is off-line; the system is not currently running.
- **Incremental.** Progressively integrates to the model the observed information. Otherwise, a new problem must be solved every time new information arrives and the identification is global.
- **Multiple.** The system to identify can be reinitialized; for the identification process there are several sequences starting with the same vector. Otherwise, the identification is simple and the input data of the identification algorithm is a single sequence.
- **Passive.** The applied input sequence cannot be modified. Otherwise, the identification is adaptive; it allows changing the excitation of the real system to

explore parts of the behaviour that may not be included in the normal functioning of the system.

- **Absolute.** The identified machine is considered a black box; no knowledge is available. Otherwise, the identification is relative.
- **Constructive.** It is given only one solution which may not be the unique solution of the identification problem. Otherwise, the identification is enumerative.
- **Measurable.** There are input and output signals for every component of the system.
- **Complete.** The identified model is able to produce the observed behaviour.
- **Solid.** The identified model is able to produce only observed behaviour.
- **Parametric.** It is necessary to know the structure of the system. The identification method consists in the estimation of its parameters. Otherwise, the identification is structural.
- **Concurrent.** The obtained model can represent concurrent processes observed from the system.

1.3.2. Analysis of methods

- **Progressive synthesis**

In this method, the construction of the model is made on-line. Since the system is not supposed to reinitialize, the identification is simple. The applied input sequence cannot be modified and there is no information about the system; that means that the identification is passive and absolute. Since the behaviour of the system is progressively integrated to one unique model, the identification is incremental and constructive. The system can be only partially measurable, but the identification is not solid because in the model could be exceeding behaviour. The system identification approach introduced is a structural approximation in the sense that the representation of the system that will be obtained is an interpreted Petri net. That means that the concurrency can be represented in the model.

This method is good for on-line identification, since it is polynomial and the information observed is progressively integrated to the system. Nevertheless, it represents more output sequences than the observed ones; that could be a problem dealing with fault diagnosis.

- **Off-line model construction**

As input of the algorithm, there are many I/O sequences whose first vector is always the same; that means that the identification is multiple. The applied input sequence cannot be modified, so we have identification passive. The identified machine is considered a black box. The analysis mode is incremental. The method is constructive because it is given a unique solution. The identification is made off-line; it is parametric and the obtained method cannot represent concurrency.

This identification method is not made on-line, but has an important property: the input/output sequences observed of length k+1 are exactly represented.

- **Integer Linear Programming approach**

The identification is off-line, simple, passive and global. Since it is known the upper bound in the number of places of the IPN, this method is not black-box identification, then, it is relative (or gray-box). The identification is also constructive, but there could be no solution for one problem. The method is parametric and the obtained method can represent concurrency.

In this methodology, every time new information is observed, a new ILP problem must be stated. It is known that solution of ILP problems is sometimes very complex.

Interesting characteristics of this method are that the obtained model can be forced to have some structural constraints and that the output signals of the system are not necessarily binary.

- **Remarks**

A more objective comparative of the methods could be made considering for all the methodologies the same work hypothesis, based in some type of application. The main features of the methods are presented in table 1.1.

	Progressive synthesis [Meda 98, 02]	Off-line model construction [Klein, 05]	Integer Linear Programming [Dotoli, 08]
On-line	☑	✖	✖
Incremental	☑	☑	✖
Multiple	✖	☑	✖
Passive	☑	☑	☑
Absolute	☑	☑	✖
Constructive	☑	☑	✖
Measurable	✖	☑	✖
Solid	✖	☑	✖
Complete	☑	☑	☑
Structural	☑	☑	☑
Concurrent	☑	✖	☑

Table 1.1 Comparative of identification approaches

1.4. Remarks

Identification methods of Discrete Event Systems have been reviewed. Three of the main methodologies for this problem have been explained, based on the analysis of their principal characteristics. This analysis has been made regardless any kind of hypothesis in the type of systems to identify. A more objective comparison could be done if it would be specified some kind of application or purpose for the identification. Finally, a comparative table with the three methods above explained has been showed. With a short analysis of that comparative table, we observe that it is possible to create an improvement on the [Meda 98, 02] method, including strategies from [Klein, 05] to create a solid methodology that is able to give a model which includes concurrence. In this way, it is possible to justify the creation of a new approach for DES identification to accomplish with best characteristics of [Meda 98, 02] and [Klein, 05]. This is the main objective for next chapter.

Chapter 2

Off-line Identification of Concurrent Discrete Event Systems

Abstract. This chapter presents a method for Off-line Identification of Concurrent Discrete Event Systems from input-output sequences including cyclic behaviour. First, objectives of the new methodology for identification are explained. Afterwards, several concepts are introduced for state the procedures included in proposed method. Illustrative examples are included along the chapter.

2.1. The notion of DES identification

An identification procedure of a DES builds a mathematical model that represents its behaviour from the measured evolution of the DES' inputs and outputs [Meda, 1998][Klein, 2005][Fanti, 2008] (see figure 2.1). This evolution of inputs and outputs represents the observable behaviour of the system.

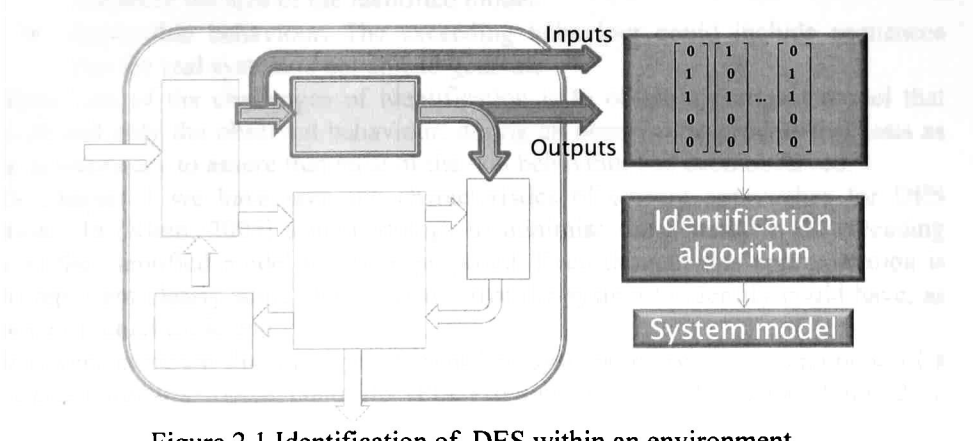

Figure 2.1 Identification of DES within an environment.

- **Challenges in DES Identification**

Since the type of identification is a black-box system, i.e., no information other than the input and output data is provided, some difficulties arise during the identification process. Some problems are related with the different types of behaviour that can be

distinguished in the system to identify and the obtained model. Other difficulties arise during the data recollection from a real system. We discuss this problematic below.

The types of behaviour during the identification process can be classified as:

- Real behaviour. It is the actual behaviour that can be generated by the system to identify.
- Observed behaviour. Normally, only part of the real behaviour is observed in a finite time during the identification process.
- Identified behaviour. It is the behaviour that can be generated by the identified model.

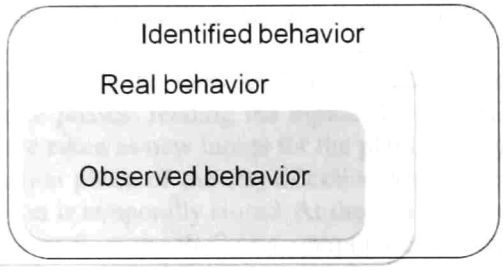

Figure 2.2 Types of behaviour in the identification

Due to the types of behaviour, some problems arise for the identification:

- Partial observation. Since only a part of the real behaviour is observed, the identified model generally will not be able to reproduce the complete behaviour of the real system. The non-observed behaviour can be minimized by observing the system for a very long time.
- Exceeding behaviour. The identified model can produce a behaviour that was not observed. This problem is generally related with the need of minimize the size of the identified model.
- Impossible behaviour. The exceeding behaviour could include sequences that the real system is not able to generate.

Then, one of the challenges of identification is to obtain a compact model that represents all and only the observed behaviour, during an observation process that lasts as much time as necessary to assure that most of the real behaviour has been observed.

In Chapter 1 we have seen the characteristics of current approaches for DES identification. In [Klein, 2005] a methodology to minimise the generation of exceeding behaviour in the identified model has been proposed. Even though, this approximation is not able to represent clearly some characteristics that the system to identify could have, as concurrency or mutual exclusion.

It is very common that the type of identified systems is usually a compound of a plant and a programmable logical controller (PLC) running in a closed loop (see figure 2.3).

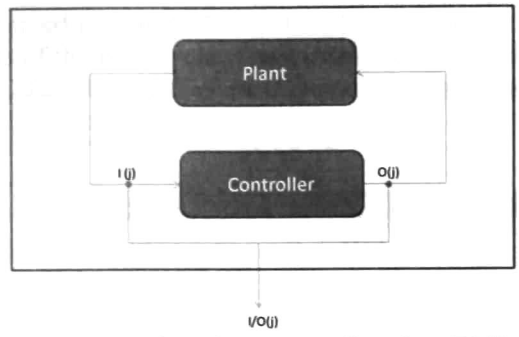

Figure 2.3 Close loop controller-plant DES

The input and output data for the identification procedure is obtained from the PLC, whose cycle involves three phases: reading the inputs (I), control program (P), and setting the outputs (O), which are taken as new inputs for the plant (see figure 2.4).

During the data collection phase of the identification procedure, at the beginning of the cycle, the input information is temporally stored. At the end of a cycle, current values of inputs and outputs (I-O) are sent from the PLC to a computer and stored there for a later treatment of the information.

Figure 2.4 PLC cycle during data collection

Due to the identification process, some situations from data collection must be considered. Below two related situations are described:

- An input evolution does not provoke any state nor output evolution.

  Consider a situation where part of the PLC control program is described by the Petri net in figure 2.5.

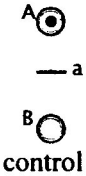

Figure 2.5 Part of a PLC control program described by a PN

No matter which input is given, if it is different from "a", the state and outputs of the system will not have an evolution. Then, can be observed many changes in the input values with no consequence in the outputs. As a consequence, many input changes can be related with the same output change.

- The non simultaneous evolution of many inputs is observed simultaneously.

  During phase I, input values are stored to be sent at the end of the PLC cycle. If many inputs evolve at different times during the phases P and O, their evolutions will be observed as simultaneous when they be sent at the end of O. This

situation can be explained in figure 2.6. Inputs a, b and c change its value from 0 to 1 at different instants of the PLC cycle. These changes are stored in the phase I and sent at the end of the phase O, and they are observed as simultaneous changes.

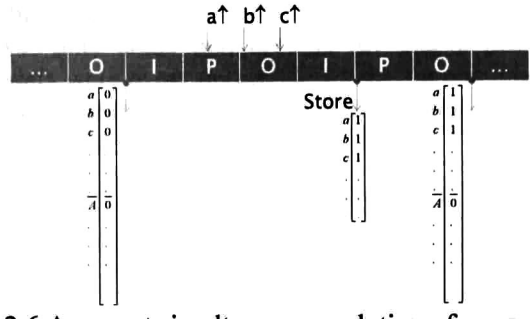

Figure 2.6 Apparent simultaneous evolution of several inputs

- An output change provoked by the evolution of an input is not immediately observed.

Consider a situation where part of the PLC control program is described by the Petri net in figure 2.7.

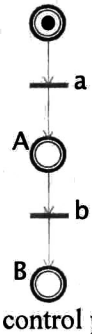

Figure 2.7 Part of a PLC control program described by a PN

If input "b" changes its value from 0 to 1, the correspondent change in "B" is not provoked immediately, since it is necessary first a change in output "A". After input "a" is given and the state of the system evolves, in the next PLC cycle the effect of input "b" is observed. This situation can be illustrated by graphic in figure 2.8, in which values of inputs and outputs evolve through time.

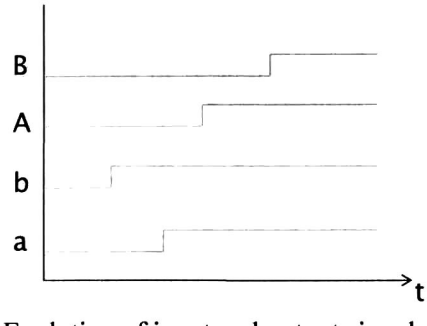

Figure 2.8 Evolution of input and output signals through time

Since it is not possible to determine a complete causality between inputs and outputs of the system, it is necessary to take into account the complete input-output observed vector for an accurate identification, i.e., an exact representation of the observed input and output behaviour of the system.

In [Meda, 2005] it was only taken into account the output information of the system since the identified models were event detectable by the output. Since it is hard to observe this property in the obtained data of a real system, the including of input information for the identification process is important.

2.2. Problem statement

Consider a *DES S* with binary signals revealing part of its internal states in which the inputs and outputs may evolve concurrently. The input data to the identification procedure is a set of input-output words $\Gamma(S)$ that may include cyclic behaviour.

- **Definitions and assumptions**

Definition 1: The set of *observed input-output words* of a DES S with m inputs and n outputs is $\Gamma(S) = \{w_1, w_2, \dots\}$, such that $w_i = (\left[\frac{I_i(1)}{O_i(1)}\right], \left[\frac{I_i(2)}{O_i(2)}\right], \dots \left[\frac{I_i(|w_i|)}{O_i(|w_i|)}\right])$, where $I_i(j)$ and $O_i(j)$ are the j-th observed input and output vectors of size m and n respectively in sequence w_i and $|w_i|$ is the length of the output word w_i.

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

Definition 3: The *input-output language of length l* of a DES S is defined as $\mathcal{L}^l(S) = \{w | w \in \mathcal{L}(S) \text{ and } |w| \leq l\}$.

Now the identification problem can be defined. Given a set of input-output words $\Gamma(S)$ and an accuracy parameter κ, the aim of the identification process is to obtain a safe IPN model (Q, M_0) such that $\mathcal{L}_{out}^{\kappa}(Q, M_0) = \mathcal{L}^{\kappa}(S)$. The parameter κ is used to adjust the accuracy of the identified model, similarly as proposed in [Klein, 2005b].

It is assumed that the input-output words are measured from the initial state; this implies that the DES has no permanent faults and that it can be reinitialized arbitrarily.

2.3. Identification method

2.3.1. General strategy

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

From the input-output vector words, the event sequences are computed and then sequences of event substrings of length κ are built. Then every substring is associated to a transition of a PN, which describes the relation precedence among the event substrings; this PN is formed by non-measurable places. Finally, the output changes provoked by events are described by marking changes in measurable places and then related to pertinent transitions in the PN; input changes are associated to such transitions.

Now in the following sub-sections we present in detail the above outlined strategy.

2.3.2. Sample processing

- **Event sequences**

As stated before, the data obtained from the system to be identified is a set of sequences of input-output vectors $w_1, w_2, ...$, such that $w_i = (w_i(1), w_i(2), ...)$, where $w_i(j)$ refers to the j-th observed vector in sequence w_i. Sequences have not necessarily the same length.

Example 2.1. Consider an illustrative example of a DES with four output signals, Φ = {A, B, C, D}, and three input signals Σ = {a, b, c}. Three input-output sequences have been observed. Notice that, in this example, the first and last output vector are always the same (cycles):

$$w_1 = \left(\begin{bmatrix}0\\0\\0\\1\\0\\0\end{bmatrix}\begin{bmatrix}1\\0\\0\\0\\0\\0\end{bmatrix}\begin{bmatrix}1\\0\\0\\0\\1\\0\end{bmatrix}\begin{bmatrix}0\\0\\0\\1\\0\\0\end{bmatrix}\right), w_2 = \left(\begin{bmatrix}0\\0\\0\\1\\0\\0\end{bmatrix}\begin{bmatrix}1\\0\\0\\0\\0\\0\end{bmatrix}\begin{bmatrix}1\\1\\0\\0\\1\\0\end{bmatrix}\begin{bmatrix}1\\0\\0\\1\\0\\1\end{bmatrix}\begin{bmatrix}1\\0\\0\\0\\0\\1\end{bmatrix}\begin{bmatrix}0\\0\\0\\1\\0\\0\end{bmatrix}\right),$$

$$w_3 = \left(\begin{bmatrix}0\\0\\0\\1\\0\\0\end{bmatrix}\begin{bmatrix}1\\0\\0\\0\\0\\0\end{bmatrix}\begin{bmatrix}1\\1\\0\\0\\1\\0\end{bmatrix}\begin{bmatrix}1\\0\\0\\0\\0\\0\end{bmatrix}\begin{bmatrix}1\\0\\0\\0\\0\\1\end{bmatrix}\begin{bmatrix}0\\0\\0\\1\\0\\0\end{bmatrix}\right)$$

From these sequences, strings of observed events are first computed.

Definition 4: An observed *event vector* $\tau_i(j)$ is the variation between two consecutive input-output vectors $w_i(j)$, $w_i(j+1)$; it is computed as $\tau_i(j) = w_i(j+1) - w_i(j)$. An *input event vector* $\lambda(\tau_i(j))$ is the variation between two consecutive input vectors $I_i(j)$, $I_i(j+1)$; it is computed as $\lambda(\tau_i(j)) = I_i(j+1) - I_i(j)$. A symbolic input event $\lambda'(\tau_i(j))$ is a string representation of the *input event vector* $\lambda(\tau_i(j))$; it is computed as:

$$\lambda'(\tau_i(j)) = \prod_{i=1}^{m} \begin{cases} I_i\_1 & if\ I_i(j+1) - I_i(j) = 1 \\ I_i\_0 & if\ I_i(j+1) - I_i(j) = -1 \\ \varepsilon & if\ I_i(j+1) - I_i(j) = 0 \end{cases}$$

According to definition 4, for every sequence w_i, a sequence of observed event vectors $\tau_i = \tau_i(1)\tau_i(2) \ldots \tau_i(|\tau_i|)$ is obtained. During the process, if the difference has not been observed before, a new event vector e_j is created and stored ($\tau_i(j) = e_j$).

Now, for the Example 1, the sequences τ_i of the detected event vectors e_j associated to input-output changes are obtained:

$$w_1 = \left(\begin{bmatrix} 0 \\ 0 \\ \frac{0}{1} \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} \overrightarrow{e_1} \\ 1 \\ 0 \\ \frac{0}{-1} \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ \frac{0}{0} \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} \overrightarrow{e_2} \\ 0 \\ 0 \\ \frac{0}{0} \\ 1 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ \frac{0}{0} \\ 1 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} \overrightarrow{e_3} \\ -1 \\ 0 \\ \frac{0}{1} \\ -1 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \frac{0}{1} \\ 0 \\ 0 \\ 0 \end{bmatrix} \right), \tau_1 = e_1 e_2 e_3$$

$$\lambda'(e_1) = a\_1, \lambda'(e_2) = \varepsilon, \lambda'(e_3) = a\_0$$

$$w_2 = \left(\begin{bmatrix} 0 \\ 0 \\ \frac{0}{1} \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} \overrightarrow{e_1} \\ 1 \\ 0 \\ \frac{0}{-1} \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ \frac{0}{0} \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} \overrightarrow{e_4} \\ 0 \\ 1 \\ \frac{1}{0} \\ 0 \\ 1 \\ 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ \frac{1}{0} \\ 0 \\ 1 \\ 0 \end{bmatrix} \begin{bmatrix} \overrightarrow{e_5} \\ 0 \\ -1 \\ \frac{0}{0} \\ 0 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ \frac{1}{0} \\ 0 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} \overrightarrow{e_6} \\ 0 \\ 0 \\ \frac{-1}{0} \\ 0 \\ -1 \\ 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ \frac{0}{0} \\ 0 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} \overrightarrow{e_7} \\ -1 \\ 0 \\ \frac{0}{1} \\ 0 \\ 0 \\ -1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \frac{0}{1} \\ 0 \\ 0 \\ 0 \end{bmatrix} \right), \tau_2 = e_1 e_4 e_5 e_6 e_7$$

$$\lambda'(e_4) = b\_1\ c\_1, \lambda'(e_5) = b\_0, \lambda'(e_6) = c\_0, \lambda'(e_7) = a\_0$$

$$w_3 = \left(\begin{bmatrix} 0 \\ 0 \\ \frac{0}{1} \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} \overrightarrow{e_1} \\ 1 \\ 0 \\ \frac{0}{-1} \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ \frac{0}{0} \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} \overrightarrow{e_4} \\ 0 \\ 1 \\ \frac{1}{0} \\ 0 \\ 1 \\ 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ \frac{1}{0} \\ 0 \\ 1 \\ 0 \end{bmatrix} \begin{bmatrix} \overrightarrow{e_6} \\ 0 \\ 0 \\ \frac{-1}{0} \\ 0 \\ -1 \\ 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ \frac{0}{0} \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} \overrightarrow{e_5} \\ 0 \\ -1 \\ \frac{0}{0} \\ 0 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ \frac{0}{0} \\ 0 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} \overrightarrow{e_7} \\ -1 \\ 0 \\ \frac{0}{1} \\ 0 \\ 0 \\ -1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \frac{0}{1} \\ 0 \\ 0 \\ 0 \end{bmatrix} \right), \tau_3 = e_1 e_4 e_6 e_5 e_7$$

- **Sequences of κ -length event traces**

As seen before, one of the challenges for the identification problem is the reduction in the size of the model allowing representing the behaviour of the system to identify. The easiest way to perform this is the association of every one of different observed input-output vectors to a single state in the model.

Example 2.2. For example, to represent the next single sequence of vectors:

$$w_1 = \left(\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \right)$$

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

we could simply associate every different vector to a non-observable place of the IPN, as shown in figure 2.9.

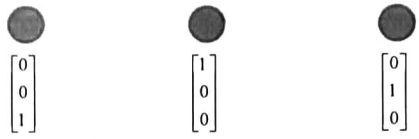

Figure 2.9 Association of input-output vectors to a non-observable place

After that, we could create transitions between the non-observable places to represent the observed event vectors (figure 2.10):

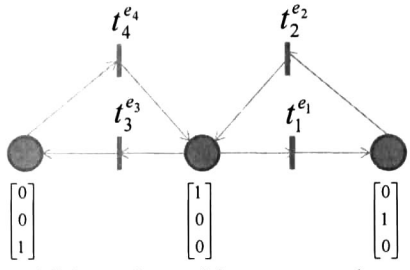

Figure 2.10 Addition of transition to non-observable places

If we put a token into the place associated to the first observed vector, in Figure 2.7 we have a "dark places net" modelling the behaviour of the system. The problem is that in this net, we can fire sequences of events that were not actually observed. For example, we could observe $e_4e_3e_4e_3$, and lead to a sequence of input-output vectors that was not really observed:

$$\begin{bmatrix}1\\0\\0\end{bmatrix},\begin{bmatrix}0\\0\\1\end{bmatrix},\begin{bmatrix}1\\0\\0\end{bmatrix},\begin{bmatrix}0\\0\\1\end{bmatrix},\begin{bmatrix}1\\0\\0\end{bmatrix}$$

Then, in order to avoid exceeding behaviour, we could try to represent every one of the observed output cyclic sequences into one single path of the model starting and ending at the initial state, without making any fusion between intermediate states. A single path for the sequence of Example 2 can be seen in figure 2.11.

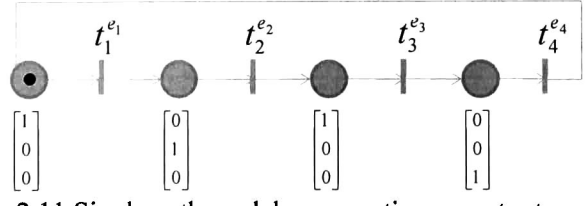

Figure 2.11 Single path model representing an output sequence

This solution could work with small size systems, but the identification of a real industrial system generally implies the computing of several vector sequences of big size. In that case, the size of the obtained model would be very large requiring computers with larger memory size.

In order to avoid this problem, merging states may reduce the size of the model, nevertheless we must preserve the language of the observed sequences. This can be accomplished by the introduction of an accuracy parameter κ for identification, which will

determine the equivalence of internal states with respect to a trace of a given length of previous observed events.

Definition 5: Two states of the identified system are κ-equivalent if their input-output vectors are the same and if the κ last observed event vectors that lead to these states are the same.

Example 2.3. Consider an illustrative system in which two robots A and B handle a work piece when the input signals a or b are given. If the input is b, they must finish their work at the same time. Consider next sequences of vectors with binary input-outputs obtained from such a system:

$$\begin{bmatrix} a \\ b \\ A \\ B \end{bmatrix} \quad \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} a \\ b \\ A \\ B \end{bmatrix} \quad \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

The first sequence represents that input a has been given to the system. Then, robot A has started to work, followed by robot B. Then, robot B has finished its work before than robot A. The second sequence represents that input b has been given to the system. Again robot A has started to work before robot B, but this time both robots finished at the same time.

If we execute an identification process with these sequences considering equal vectors as equivalents, we would obtain a model as depicted in figure 2.12.

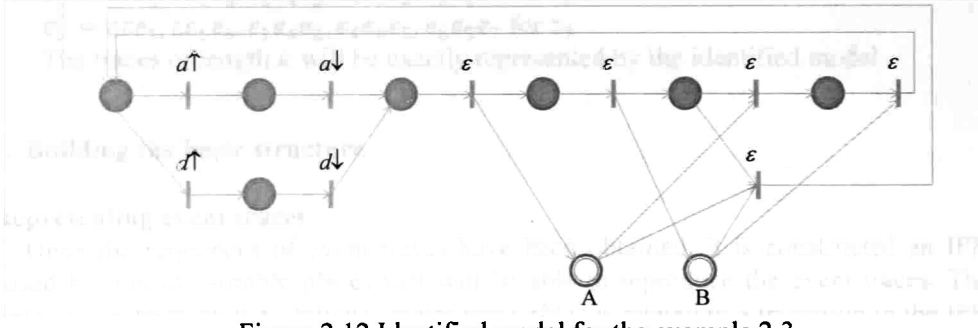

Figure 2.12 Identified model for the example 2.3

Since we are considering vectors $\begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$ of the two sequences as equivalent, in the obtained model, no matter if the observed input signal a or b is given, the activity of robot A ends always before than activity of B . But if we take into account a string of events of length 3, we can differentiate sequence $\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$ from $\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$ and then we

can state that the two vectors $\begin{bmatrix}0\\0\\1\\1\end{bmatrix}$ are not equivalent. This difference will lead us to an accurate identification process, since we will obtain a model in which incorrect behaviour is not allowed.

As it can be noticed, the chosen value of κ depends then on how much it is necessary to differentiate between system states apparently equivalents. Since we are dealing with a black-box identification, it is hard a correct election of κ. In general, this choice has been made by experimentation.

Now we can define the process for introducing the identification parameter κ. From every sequence of events $\tau_i = \tau_i(1)\tau_i(2) \dots \tau_i(|\tau_i|)$ we compute sequences of event traces $\tau_i^\kappa = \tau_i^\kappa(1), \tau_i^\kappa(2), \dots \tau_i^\kappa(|\tau_i|)$ such that every $\tau_i^\kappa(j)$ is the substring of length κ of τ_i that finishes with $\tau_i(j)$. For the first $\kappa - 1$ elements of the trace sequence the event ε is used.

Following with the Example 1, the sequences of traces using $\kappa = 1$ are:
$\tau_1^1 = e_1, e_2, e_3$ for τ_1
$\tau_2^1 = e_1, e_4, e_5, e_6, e_7$ for τ_2
$\tau_3^1 = e_1, e_4, e_6, e_5, e_7$ for τ_3
Using $\kappa = 2$
$\tau_1^2 = \varepsilon e_1, e_1 e_2, e_2 e_3$ for τ_1
$\tau_2^2 = \varepsilon e_1, e_1 e_4, e_4 e_5, e_5 e_6, e_6 e_7$ for τ_2
$\tau_3^2 = \varepsilon e_1, e_1 e_4, e_4 e_6, e_6 e_5, e_5 e_7$ for τ_3
Using $\kappa = 3$
$\tau_1^3 = \varepsilon\varepsilon e_1, \varepsilon e_1 e_2, e_1 e_2 e_3$ for τ_1
$\tau_2^3 = \varepsilon\varepsilon e_1, \varepsilon e_1 e_4, e_1 e_4 e_5, e_4 e_5 e_6, e_5 e_6 e_7$ for τ_2
$\tau_3^3 = \varepsilon\varepsilon e_1, \varepsilon e_1 e_4, e_1 e_4 e_6, e_4 e_6 e_5, e_6 e_5 e_7$ for τ_3
The traces of length κ will be exactly represented by the identified model.

2.3.3. Building the basic structure

Representing event traces

Once the sequences of event traces have been obtained, it is constructed an IPN composed by non-measurable places that will be able to reproduce the event traces. The principle of construction is as follows: every trace $\tau_i^\kappa(j)$ is related to a transition in the IPN through a function γ:T$\rightarrow\{\tau_i^\kappa(j)\}$; the firing of a transition implies that κ consecutive events related to such a transition have been observed. In order to preserve firing order between transitions, dependencies are created between them and associated with an observed marking through function $\mu: P^u \rightarrow M$, which relates every non-measurable place with an observed vector.

Following with this procedure, every transition will have only one input place and one output place ($\forall t_r \in T$, ${}^\bullet t_r = t_r^\bullet = 1$); when an event trace $\tau_i^\kappa(j)$ is found again in a τ_i^κ the associated input dependency must be used if it leads to the same observed marking. Let e_j be the last event in the trace $\tau_i^\kappa(j)$; the associated transition will be denoted as $t_r^{e_j}$ (more than one transition may have associated the same e_j).

This strategy can be systematically performed following the next procedure.

Algorithm 1. Building the basic IPN structure

Input: The set Γ^{κ} of τ_i^{κ}

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

Step 1.

$T \leftarrow \varnothing$; *//Create an initial empty set of transitions*

$ET \leftarrow \varnothing$; *//Create an initial empty event trace set*

$P \leftarrow \{p_{ini}\}$; *//Create an initial set of places with only a place* p_{ini}

$M_0(p_{ini}) \leftarrow 1$; *//Put a token in* p_{ini}

$\mu(p_{ini}) \leftarrow \tau_i(1)$;*//Associate* p_{ini} *with the first observed vector.*

Step 2.

$\forall \tau_i^{\kappa} \in \Gamma^{\kappa}$:

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

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

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

then

$ET \leftarrow ET \cup \{\tau_i^{\kappa}(j)\}$; *//Add the trace to the event trace set*

$T \leftarrow T \cup \{t_r^{e_j}\}$; *//create a transition* $t_r^{e_j}$ *to represent the trace*

$\gamma(t_r^{e_j}) \leftarrow \tau_i^{\kappa}(j)$; *//associate the trace to the created transition*

$\forall p_a \in P,\ I(p_a, t_r^{e_j}) \leftarrow 0; O(p_a, t_r^{e_j}) \leftarrow 0;$ *//make 0 input and output functions of the transition*

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

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

then

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

else

$P \leftarrow P \cup \{p_{out}\}$; *//create a new place* p_{out}

$\forall t_b \in T, I(p_{out}, t_b) \leftarrow 0, (p_{out}, t_b) \leftarrow 0$; *//make 0 input and output functions of the place*

$\mu(p_{out}) \leftarrow \mu(current) + e_j$; // associate p_{out} to the computed observed marking

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

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

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

then

$\forall t_r^{e_j} \in T | \gamma(t_r^{e_j}) = \tau_j^{\kappa}(j)$,$p_{in} = {}^{\bullet}t_r^{e_j}$ *//for every transition which represents the sequence, take its input place* p_{in}

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

then

$\forall t_b \in T, I(p_{in}, t_b) \leftarrow \vec{\oplus}(I(p_{in}, t_b),\ I(current, t_b))$; where $\vec{\oplus}$ is a vector bitwise or operation

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

$P \leftarrow P \backslash \{current\}$;

//last three steps are to merge current place with such an input place

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

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

If $j = |\tau_i^{\kappa}|$ and $\mu(current) = \mu(p_{ini})$*//If it is the last trace of the sequence and the sequence is cyclic*

then

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

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

$P \leftarrow P \backslash \{current\}$;

//merge current with the initial place.

2.3 $\forall t_r^{e_j} \in T, \lambda\big(t_r^{e_j}\big) \leftarrow e_j$ *//associate every transition t_r to the last event vector e_j of the event trace it represents.*

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

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

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

Using the previous algorithm with a value $\kappa = 2$, the obtained IPN corresponding to the first sequence of event traces of the Example 1 is showed in figure 2.13. Notice how every one of the computed traces is related to a transition.

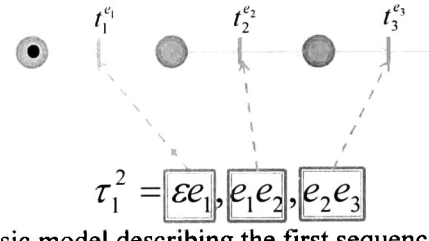

Figure 2.13 Basic model describing the first sequence of event traces

Starting from the same place, we add the second sequence of event traces to the model. This second basic structure can be seen in Figure 2.14.

Finally, the third sequence of event traces is added to the model. This final basic structure can be seen in Figure 2.15. Notice that there are different transitions related to the same event, since their respective event traces are not the same.

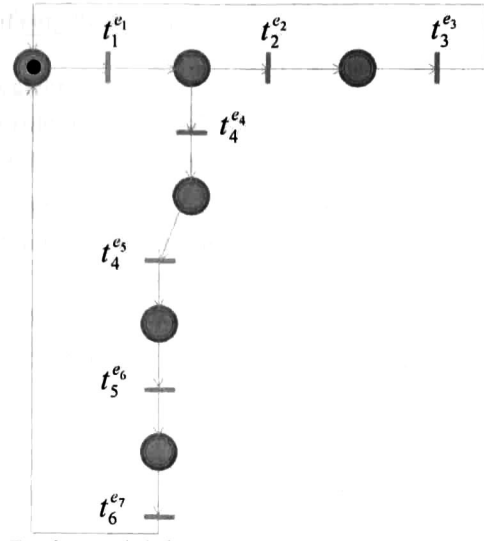

Figure 2.14 Basic model describing two sequences of event traces

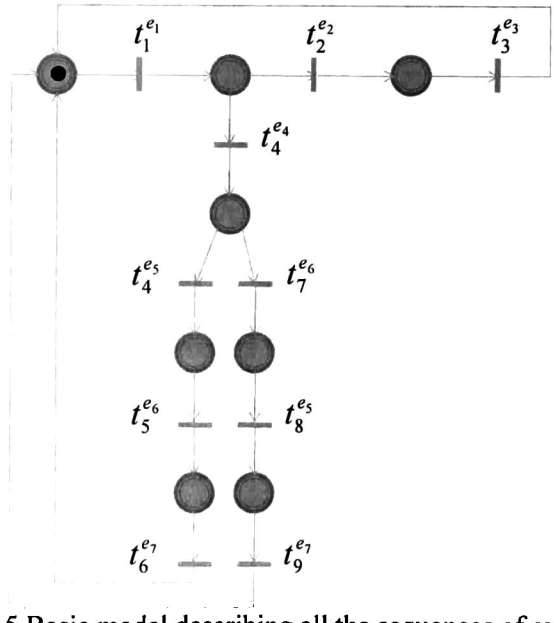

Figure 2.15 Basic model describing all the sequences of event traces

2.3.4. Simplifying the basic structure

Transitions fusion

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

Algorithm 2. Simplifying basic structure.

Input: G

Output: G': an equivalent IPN

Apply the following rule on the initial place

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

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

Property 2. G' preserves the sequences in G.

Proof. Let $Lf_{p_i} = \{\lambda(t_1)\lambda(t_2) \dots \lambda(t_r) |\ t_1 \in p_i^{\bullet}, t_{i+1} \in (t_i^{\bullet})^{\bullet}\}$ be the set of observable sequences from place p_i. Consider $t_a^{e_j}, t_b^{e_j} \in p_{ini}^{\bullet} | a \neq b$. Before applying rule 1, $Lf_{p_{ini}} = e_j L_{p_a} \cup e_j L_{p_b} \cup (\cup \lambda(t_i) L_{p_i})$, with $p_a = (t_a^{e_j})^{\bullet}$ $p_b = (t_b^{e_j})^{\bullet}$; $t_i \in p_{ini}^{\bullet} |\ t_i \neq t_a^{e_j}, t_i \neq t_b^{e_j}$ $p_i = (t_i)^{\bullet}$ After applying rule 1, $Lf_{p_{ini}} = e_j(L_{p_a} \cup L_{p_b}) \cup (\cup \lambda(t_i) L_{p_i})$. These two languages are the same, and then language of G is not changed by the application of rule 1.

Let $Lt_{p_i} = \{\lambda(t_r)\lambda(t_{r-1}) \dots \lambda(t_1) | t_1 \in {}^{\bullet}p_i, t_{i+1} \in (t_i^{\bullet})^{\bullet}\}$ be the set of observable sequences till place p_i. Consider $t_a^{e_j}, t_b^{e_j} \in {}^{\bullet}p_{ini} | a \neq b$. Before applying rule 2, $Lt_{p_{ini}} = L_{p_a} e_j \cup L_{p_b} e_j \cup (\cup L_{p_i} \lambda(t_i))$, with $p_a = {}^{\bullet}(t_a^{e_j})$, $p_b = {}^{\bullet}(t_b^{e_j})$; $t_i \in {}^{\bullet}p_{ini} |\ t_i \neq t_a^{e_j}, t_i \neq t_b^{e_j}$ $p_i = {}^{\bullet}(t_i)$. After applying rule 2, $Lt_{p_{ini}} = (L_{p_a} \cup L_{p_b}) e_j \cup (\cup L_{p_i} \lambda(t_i))$. These two languages are the same, and then language of G is not changed by the application of rule 2. ■

Since merging procedures are executed in linear time on the number of transitions and rules cannot be applied more times than the number of places in the net, Algorithm 2 is executed in polynomial time on the number of repeated transitions in the same place (starting from the initial place).

In the example the applications of the rules lead to the model depicted in figure 2.13. Since the initial place has two input transitions associated to e_7, they can merge and their input places.

- **Concurrent transitions**

Other transformations may be performed when there are transitions that appear in the sequences in different order describing their interleaved firing; this behaviour is exhibited by concurrent transitions.

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

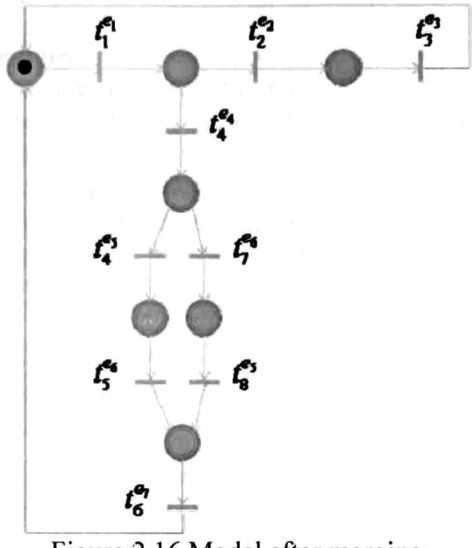

Figure 2.16 Model after merging

In figure 2.16 notice that between the transition associated to e_4 and the new transition associated to e_7 there are paths with all possible permutations of e_5 and e_6; then, we can transform this into a concurrent component and we obtain the net showed in figure 2.17. Notice that this model preserves the same event sequences of the previous one.

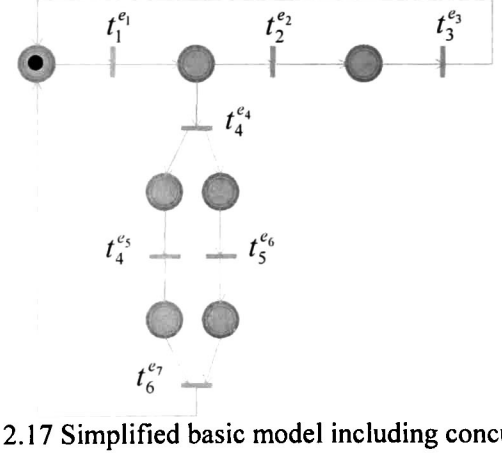

Figure 2.17 Simplified basic model including concurrence

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

2.3.5. Adding outputs and inputs

- **Representing output changes**

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

Algorithm 3. Representing outputs changes

Input: G', {e(j)}

Output; Q : the IPN including measurable places and inputs

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

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

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

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

The net with measurable places for the example 1 is showed in figure 2.18.

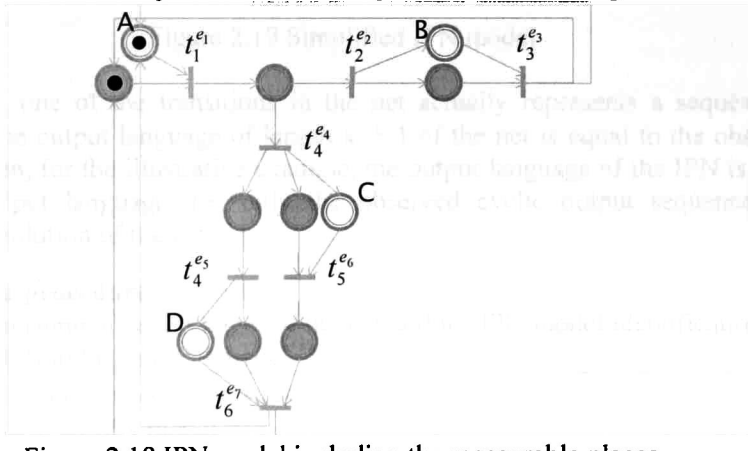

Figure 2.18 IPN model including the measurable places

2.3.6. Model simplifying

Since implicit non-measurable places in the net are not necessary to constrain the firing order of the transitions in the net we can remove them: if there is a non-measurable place whose input and output transitions are exactly the same than any measurable place, then delete such a place and its input and output arcs.

The final model for the illustrative example is showed in figure 2.19; the incidence matrix, output function and initial marking of this IPN are given below. The associated inputs for transitions are given by the symbolic event input function: $\lambda'(e_1) = a_1, \lambda'(e_2) = \varepsilon, \lambda'(e_3) = a_0, \lambda'(e_4) = b_1c_1, \lambda'(e_5) = b_0, \lambda'(e_6) = c_0, \lambda'(e_7) = a\_0$

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

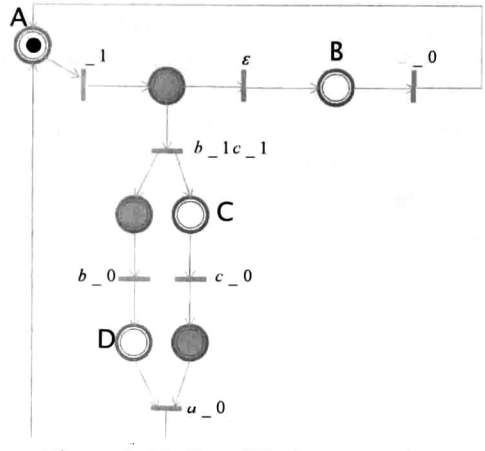

Figure 2.19 Simplified IPN model

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

2.3.7. Identification procedure

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

Algorithm 4. Building an IPN model

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

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

5. Complete the Petri net adding the observed behaviour and delete implicit places (Algorithm 3).

2.4. Properties of the identified model

Proposition 1. For a DES S that holds with the hypothesis given in section 2.2 and an identification parameter κ, Algorithm 4 yields an IPN model (Q,M_0) which represents exactly $\mathcal{L}^{\kappa+1}(S)$.

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

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

2.5. Other simplification strategies

As explained in section 2.1, there could be some phenomena during the input-output data measuring appearing as input changes without any consequent output change; also it may appear spontaneous output changes without any related input variation (internal events). When these two events are detected sequentially it is possible to consider that the output change is provoked by the input change. Then a more compact representation may be obtained; this is illustrated in the following example.

Example 2.4 Consider a situation involving one input x and two outputs A, B, and the following sub-sequence:

$$\begin{bmatrix} x \\ \hline A \\ \hline B \end{bmatrix} \quad \begin{bmatrix} 0 \\ \hline 1 \\ \hline 0 \end{bmatrix}, \begin{bmatrix} 1 \\ \hline 1 \\ \hline 0 \end{bmatrix}, \begin{bmatrix} 1 \\ \hline 1 \\ \hline 0 \end{bmatrix}$$

This subsequence shows that there is a change in input x, but without any observed output change. Then, a sudden change from A to B is observed. This can be represented as:

$$A \xrightarrow{x\_1} A \xrightarrow{\varepsilon} B$$

If we consider that it is possible that the observed behaviour was provoked by the data extraction and we assume that in the identified system there are no input changes that do not provoke any output change, the previous behaviour can be represented as follows:

$$A \xrightarrow{x\_1} B$$

If the above assumption is held, the case when several input changes that do not provoke output changes are observed can be handled similarly. Thus the next transformation may be applied for an example involving three input changes:

$$A \xrightarrow{x\_1} A \xrightarrow{y\_1} A \xrightarrow{z\_0} A \xrightarrow{\varepsilon} B \cong A \xrightarrow{x\_1,y\_1,z\_0} B$$

These kinds of transformations allow obtaining more compact models useful for process reengineering when there is some knowledge on the actual identified system. However, the obtained model is no more an exact representation of the observed behaviour.

2.6. Examples

We present now several examples that have been tested with the proposed algorithm.

Example 2.5 It is taken from [Meda, 2002] an example with two sequences. Since the approximation of this work does not take into account the input information, we apply the method considering that the system to identify has not input signals; then, all the events will be related with the symbolic event input ε.

$$w_1 = \begin{bmatrix}1\\0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\\1\end{bmatrix}, \begin{bmatrix}0\\0\\0\\1\end{bmatrix}, \begin{bmatrix}1\\0\\0\\0\end{bmatrix}; \quad w_2 = \begin{bmatrix}1\\0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\0\\1\end{bmatrix}, \begin{bmatrix}1\\0\\0\\0\end{bmatrix}$$

A representative image of the system that produces these two sequences is shown in figure 2.20. The identified model using the progressive synthesis algorithm is showed in figure 2.21.

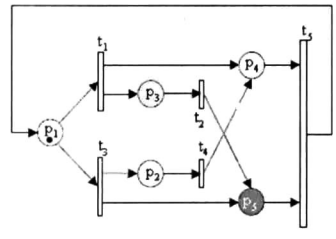

Figure 2.20 IPN model generating the output sequences

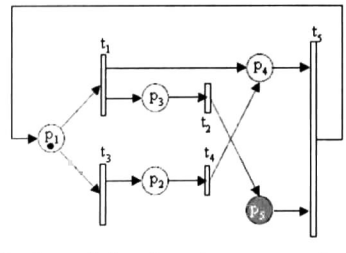

Figure 2.21 Identified model using the progresive synthesis method

Notice that the obtained model is not live. We proceed now with our identification algorithm. First, we obtain the sequences of events:

$$w_1 = \begin{bmatrix}1\\0\\0\\0\end{bmatrix}\begin{bmatrix}\overset{e_1}{\rightarrow}\\-1\\0\\1\\1\end{bmatrix}\begin{bmatrix}0\\0\\1\\1\end{bmatrix}\begin{bmatrix}\overset{e_2}{\rightarrow}\\0\\0\\-1\\0\end{bmatrix}\begin{bmatrix}0\\0\\0\\1\end{bmatrix}\begin{bmatrix}\overset{e_3}{\rightarrow}\\1\\0\\0\\-1\end{bmatrix}\begin{bmatrix}1\\0\\0\\0\end{bmatrix}, \tau_1 = e_1e_2e_3$$

$$w_1 = \begin{bmatrix}1\\0\\0\\0\end{bmatrix}\begin{bmatrix}\overset{e_4}{\rightarrow}\\-1\\1\\0\\0\end{bmatrix}\begin{bmatrix}0\\1\\0\\0\end{bmatrix}\begin{bmatrix}\overset{e_5}{\rightarrow}\\0\\-1\\0\\1\end{bmatrix}\begin{bmatrix}0\\0\\0\\1\end{bmatrix}\begin{bmatrix}\overset{e_3}{\rightarrow}\\1\\0\\0\\-1\end{bmatrix}\begin{bmatrix}1\\0\\0\\0\end{bmatrix}, \tau_1 = e_4e_5e_3$$

Using $\kappa = 1$, we compute the event traces:

$\tau_1^1 = e_1, e_2, e_3$ for τ_1

$\tau_2^1 = e_4, e_5, e_3$ for τ_2

We obtain the basic structure. As it can be seen in figure 2.22, no simplification can be made on this model. Then we add the observable places and obtain model showed in figure 2.23.

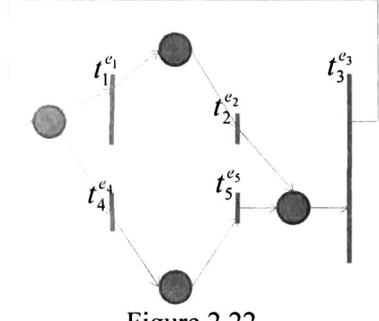

Figure 2.22

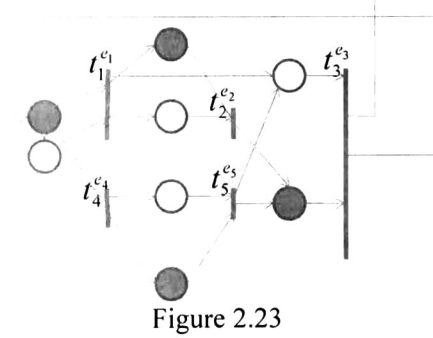

Figure 2.23

Finally we simplify deleting implicit places and we obtain the model depicted in figure 2.24. Notice the obtained model is not the same than the presented in figure 2.20, but it has the same represented behaviour.

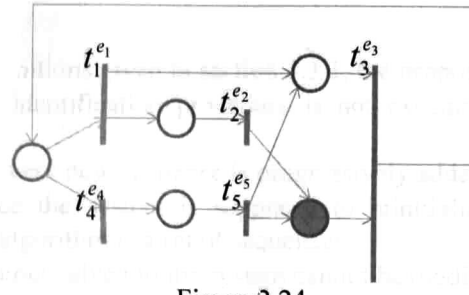

Figure 2.24

Example 2.6. Consider the cyclic sequence:

$$\begin{bmatrix}1\\0\\0\end{bmatrix},\begin{bmatrix}0\\1\\0\end{bmatrix},\begin{bmatrix}1\\0\\0\end{bmatrix},\begin{bmatrix}0\\0\\1\end{bmatrix},\begin{bmatrix}1\\0\\0\end{bmatrix}$$

In which every component of the vectors is an output signal. This sequence can be modelled exactly by the net in Figure 2.25.

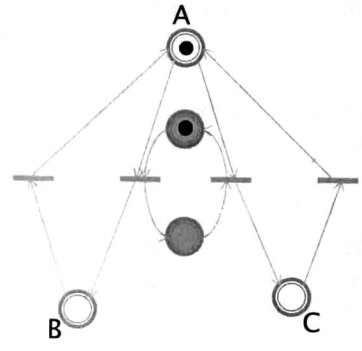

Figure 2.25

After applying the identification algorithm, we obtain the net in figure 2.26. Notice that it has the same behaviour than the net in figure 2.25.

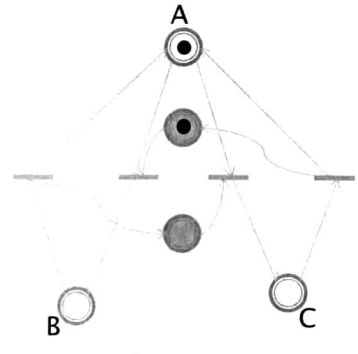

Figure 2.26

2.7. Features of the method

According to the definitions given in section 1.3.1, the proposed algorithm is:

- Off-line. The identification procedure is not executed during the system operation.
- Incremental. Every new sequence is progressively added to the model.
- Multiple. Since the system is supposed to reinitialize, the input for the identification algorithm is a set of sequences.
- Passive. The inputs given to the system cannot be modified.
- Absolute. The system to identify is considered as a black box.
- Constructive. It is given only one solution.
- Measurable. The identified system can be partially measurable.
- Solid. Only observed sequences of length κ are represented.
- Complete. All observed sequences are accepted.
- Structural. We know the representation of the system that we want to obtain.
- Concurrent. Concurrency is captured in the model.

With these characteristics, we can complete the table given in Section 1.3.2 of this work adding a column for our approach:

	Progressive synthesis [Meda 98, 02]	Off-line model construction [Klein, 05]	Integer Linear Programming [Dotoli, 08]	Off-line I-O Identification [Estrada, 09]
On-line	☑	✖	✖	✖
Incremental	☑	☑	✖	☑
Multiple	✖	☑	✖	☑
Passive	☑	☑	☑	☑
Absolute	☑	☑	✖	☑
Constructive	☑	☑	✖	☑
Measurable	✖	☑	✖	☑
Solid	✖	☑	✖	☑
Complete	☑	☑	☑	☑
Structural	☑	☑	☑	☑
Concurrent	☑	✖	☑	☑

Table 2.1 Comparative including the new methodology

2.8. Remarks

In this chapter we provided an off-line methodology to create an IPN model for a given DES described by a set of cyclic output sequences. The proposed off-line procedure operates considering an accuracy parameter , introducing the notion of -equivalent states for a given model. The parameter is considered such that in the output language of the obtained IPN only and all observed output sequences of length + 1 are represented. This approach avoids including in the model non observed behaviour, as found in previous techniques. The obtained model captures non observable behaviour represented by non-observable places and uncontrollable (silent) transitions.

The proposed algorithms have linear complexity on the input data size; in general κ must be greater than one and the obtained model allows representing complex behaviour such as concurrence, non determinism and sequencing.

Chapter 3

A software tool for DES identification

Abstract. This chapter presents a software tool for identification of Discrete Event Systems implementing the proposed method in the previous chapter. First, the specification, architecture, and implementation of the tool are presented. Afterwards, the software is demonstrated on several case studies, including a small size industrial experimental system.

3.1. Specification

Since even with a methodical algorithm it is very hard to do the identification of a big DES by hand, it is necessary the implementation of the algorithm given in last section. We present here characteristics that must be accomplished by the software tool.

- Requirements

Since the recollection data process gives as output sequences of vectors into text files, the tool must be able to read such series of I/O vectors to generate the IPN resulting from applying the algorithm proposed in this work, taking into account a parameter κ introduced by the user. Also it is necessary that the user can introduce the names of the input and output signals of the system.

At the end of the algorithm, a graphical representation of the identified IPN must be given, with every transition of the net associated with the input change it represents and every observable place with the name of the output of the system it represents.

Since it is important size information of the IPN, it will be displayed at the end of the identification process, with data such as number of transitions and number of places (making difference between observable and non-observable places).

To be able to make a comparison with previous works on the matter, in which result is given as an automaton, the tool must be able to generate the reachability graph of the identified IPN, showing the number of places and transitions of such a graph.

For a later analysis on the number of new traces discovered at the treatment of every vector sequence, the tool must be able to create an output file showing the number of transitions and places that compose the built net after adding of every sequence vector on the net.

3.2. Architecture

The architecture of the tool is showed in the diagram of figure 3.1. First, the user will introduce through an user interface the generic name of the text files containing the input-output obtained sequences and some identification options such as the parameter κ, the names of the input and output signals, the output file name desired, the order in which inputs and outputs appear in the txt files and the index numbers of the signals to take into account if a mask is going to be applied.

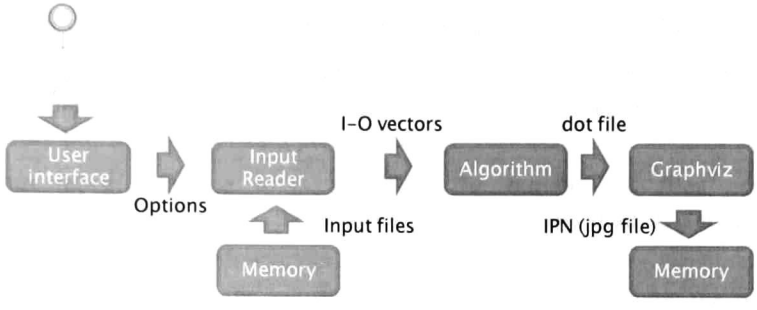

Figure 3.1 Software architecture

Later, an input reader component processes these input files and transform every input-output sequence into a vector. These vectors will be sent to a component called Algorithm in which the algorithm of Chapter 2 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.

3.3. Implementation

The algorithm has been implemented with IDE Netbeans 6.5, java jdk 1.6.0. In order to manipulate matrices, the library Java Matrix Package (Jama v1.0.2) has been used. For creating the graphic model images (IPN and reachability graph), the hierarchical layout algorithm dot of Graphviz has been applied. Actually, command dot can be invoked from the application without need to open the Graphviz interface. We present next general characteristics of the implemented tool.

An input file is a set of rows representing one cyclic sequence obtained from measure of the system to identify. Each row is composed of three parts. Sometimes it can represent the obtained input (second part) and output (third part) vectors obtained at time shown in first word. In other times, the second part can represent the output and third part the input. The time information could be used in future proposes.

An example of an input file representing one I/O vector sequence is:

```
10   000000100   1010
12   000010100   1010
14   100010100   1010
15   100010000   1010
```

```
18   100000000   1010
20   011000010   1010
23   011000000   1010
25   011000000   0010
27   001000100   0010
30   000000100   0110
31   000000100   0010
35   000000100   1010
```

We can see that this file represents an observed sequence of 12 vectors with 9 inputs and 4 outputs. Notice that first and last vectors must always be the same.

All input files have the same root name and finish with the number of represented cycle. For example, if we have 12 input files whose root name is cycle, file names are cycle1.txt, cycle2.txt, ..., cycle11.txt,cycle12.txt.

In the input interface it is specified the number of files considered for identification, the root files name, the desired accuracy parameter κ and the output file name.

If we want to show in the net the name of inputs and outputs, we select the field Signal names and add into the text field the names of the outputs and inputs (in that order) separated with a blank. If on the input files the second column has outputs, the field First outputs must be selected.

If we want to take into account only some of the inputs or outputs of the I/O vectors, we can use a mask by selecting the field Indices and inserting the indices of fields we want to take into account.

If we want to obtain an image file of the identified IPN, we select the field Create image. The output is a file named as specified in the output file name and with extension svg.

If we want to obtain the reachability graph of the IPN, we select the field Create reachability graph. The output is a file with extension svg named as specified in the output file name, adding -graph at the end and with extension svg.

In the field data it is displayed the information of the identification made. That is, the number of transitions and places of the IPN obtained and the execution time of identification. Also in case of creation of the reachability graph, the number of places and transitions of such a graph is shown.

The button *Start* launches a complete identification process. The button Analysis makes the identification process many times, starting with only one cycle and progressively incrementing the number of cycles to create a file showing the number of transitions and places created at each step. Figure 3.2 shows the identification tool user interface.

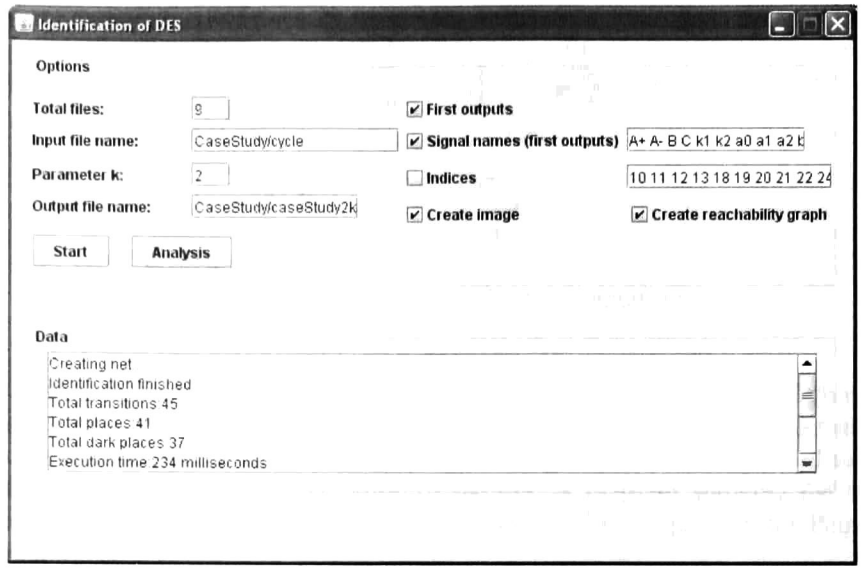

Figure 3.2 User Interface

After the button Start is pressed, it begins a cycle where every input file is read. When a line of such a file is read, time information is ignored and input and output are transformed into Matrix. With those matrices, the identification algorithm described in section 2 is applied and the IPN is constructed as an object of the class PetriNet.

A *PetriNet* class contains two lists: one to store Place objects and the other to store Transition objects. Place and Transition objects can be identified with a label. Every transition has a list containing the labels of its input places and a list containing the labels of its output places. Similarly, every place has a list containing the labels of its input transitions and a list containing the labels of its output transitions. Also, every place contains a boolean variable specifying if it is a dark place or not.

For example, following with syntax:

(label, inputTransitions, outputTransitions, dark) for places and

(label, inputPlaces, outputPlaces, dark) for transitions

we could specify Petri net in Figure 3.3 as follows:

$(t_1, (p_4), (p_1, p_2, p_3)), (t_2, (p_2, p_3), (p_4))$

$(p_1, (t_1), (), \text{false}), (p_2, (t_1), (t2), \text{false}), (p3, (t1), (t2), \text{true}), (p4, (t2), (t1), \text{true})$.

The redundancy introduced by the double storing of the arcs of the net (the same arc is stored in the list of one place and in the list of one transition) is to facilitate the search of inputs of outputs of transitions and places.

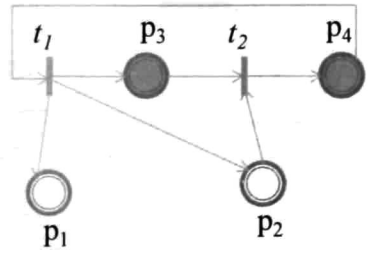

Figure 3.3 IPN given by a list representation

After the identification algorithm has finished, the Petri net is translated into a file and with an invocation to graphviz it is transformed into an image. Also, an object of the class IncidenceMatrix is created to store the Pre and Post matrices of the obtained model and be able to generate the reachability graph of the net. A diagram showing the steps followed for an identification process can be seen in figure 3.4. The diagram of the PetriNet class is shown in figure 3.5.

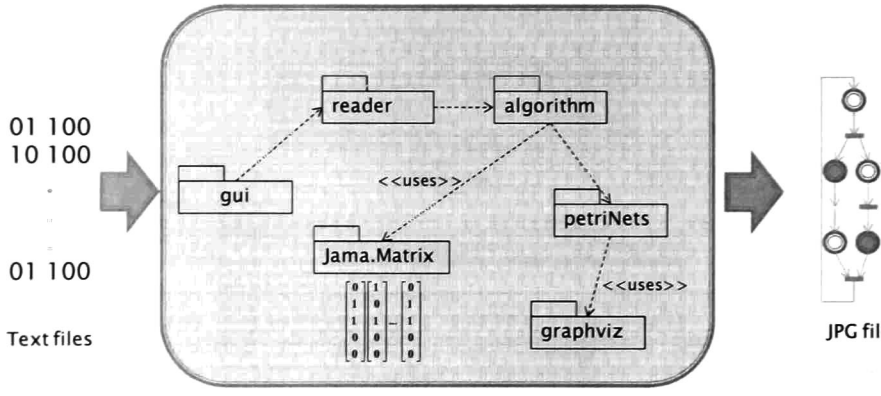

Figure 3.4 Tool diagram

Figure 3.5 PetriNet Class diagram

3.4. Examples

Example 3.1 The presented identification tool has been used to identify many theoretical examples. To check the capture of concurrency with the method, it has been identified a small hypothetical manufacturing system obtained from [Roth, 2009]. It is shown in figure 3.6 The purpose of such a system is to sort parcels according to their size. It has 9 inputs and 4 outputs.

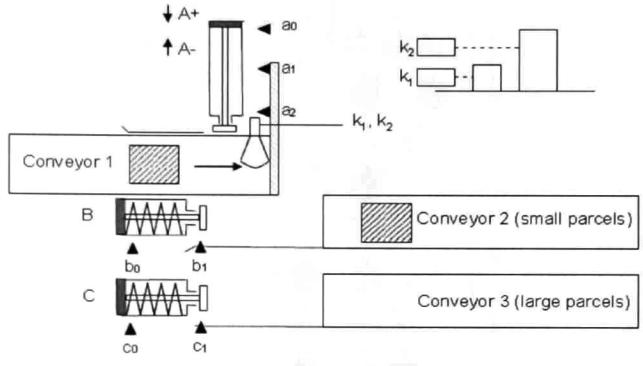

Figure 3.6 Case study

The system works as follows: a parcel arrives through conveyor 1. If it is a small parcel, k1 sets its value to 1 and then it is transported to conveyor 2 using cylinder A (setting A+=1 and A-=0). There, it is pushed by cylinder B. If the parcel that arrives in conveyor 1 is a big one, k1 and k2 set its values to 1 and parcel is transported to conveyor 3. Once in that conveyor, it is pushed with cylinder C.

After the identification process, the net in figure 3.7 is obtained. Currently transformations by concurrency analysis are not implemented the software tool; this transformation requires a deepest study to improve its performance. Notice that there are many input changes that do not lead to any change in the output. This is represented by transitions that do not have any arc to or from a measurable place.

We can apply to this model a transformation as explained in section 2.5 of this work. The obtained model after transformations is given in figure 3.8.

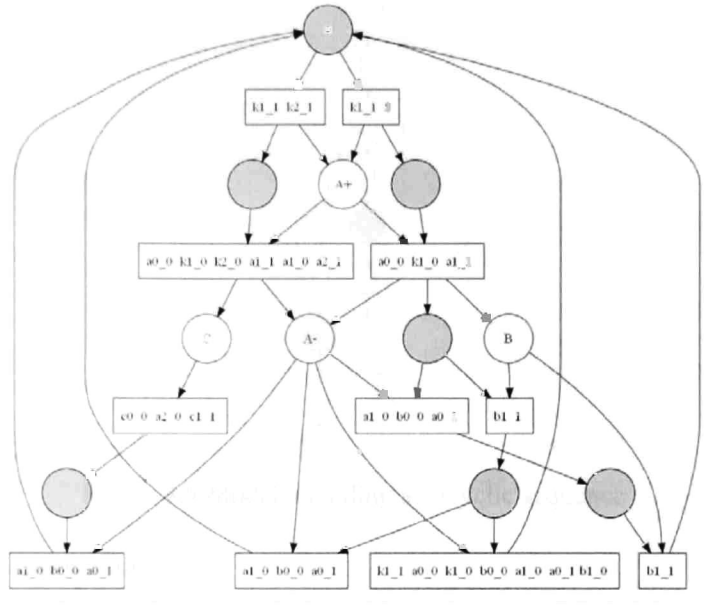

Figure 3.8 Alternative reduced model for the case study

Notice that this model is a close structural representation of the presented system. Even though.

Example 3.2 Consider next two sequences of vectors with outputs A, B, C, D, E and no inputs. Notice that the first sequence is not cyclic.

$$\begin{bmatrix}0\\0\\0\\0\\0\end{bmatrix},\begin{bmatrix}1\\0\\0\\0\\0\end{bmatrix},\begin{bmatrix}0\\1\\0\\0\\0\end{bmatrix},\begin{bmatrix}0\\0\\1\\0\\0\end{bmatrix},\begin{bmatrix}0\\0\\0\\0\\0\end{bmatrix},\begin{bmatrix}0\\0\\0\\1\\0\end{bmatrix}$$

$$\begin{bmatrix}0\\0\\0\\0\\0\end{bmatrix},\begin{bmatrix}1\\0\\0\\0\\0\end{bmatrix},\begin{bmatrix}0\\1\\0\\0\\0\end{bmatrix},\begin{bmatrix}0\\0\\0\\0\\0\end{bmatrix},\begin{bmatrix}0\\0\\0\\0\\1\end{bmatrix},\begin{bmatrix}0\\0\\0\\0\\0\end{bmatrix}$$

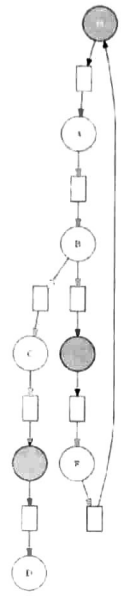

Figure 3.9 Model including an acyclic sequence

3.5. Industrial application

The algorithm has been tested also with a laboratory industrial system. The input and output information that we had of identification was already in text files, since an identification process was applied before with the algorithm of [Klein, 2005] by Matthias Roth at the university of Kaiserslautern in Germany.

3.5.1. Characteristics of the system

The system is called "Fischertechnik" and it is a flexible manufacturing line. It consists of three machines and three conveyors connecting them. The plant has 30 binary I/Os. During one production cycle, it threats three work pieces.

3.5.2. Identified model

An identification process for 100 input-output sequences has been applied to the Fischertechnik with different values of κ. Results are shown in table. Since models are huge, the figures of identified models are not shown.

κ	Transitions	Places	Dark places
1	308	265	250
2	367	329	314
3	420	388	373

4	468	439	424
5	512	490	475
6	549	535	520

Table 3.1 Identification data for the Fischertechnik

3.6. Remarks

A software tool for DES identification has been described. It has been tested with several case studies of diverse complexity, going from cases including few input-output short sequences to the case of an industrial experimental system providing a large amount of input data. An alternative representation of the obtained model has been presented, merging transitions of the IPN that do not provoke any observable output change. The obtained model is more compact and gives a closer idea on the structure of the real system.

Conclusions

In this thesis the problem of identification of Discrete Event Systems (DES) has been addressed. A method for building Interpreted Petri Net models from input-output sequences has been conceived and implemented.

The proposed method allows dealing with DES as black-boxes; the external observed behaviour is recorded as sequences of input-output binary signals from an initial situation. The sequences include the behaviour observed in actual systems namely, cyclic functioning, input changes without an associated change in the outputs, and spontaneous output changes. The identified model captures this behaviour including places without associated output (non-measurable places) and transitions without associated inputs (uncontrollable or silent transitions) representing internal events.

The identification technique holds some strategies from [Meda, 2002] and [Klein, 2005] overcoming the abilities of both methods; the resulting off-line procedure operates considering an accuracy parameter κ such that in the output language of the obtained IPN only and all observed output sequences of length $\kappa + 1$ are represented. This approach avoids including in the model more sequences than that observed.

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

Although the presented method goes ahead the previous identification techniques regarding most of the analysed features, several issues can be addressed for improving our work described herein:

- A deeper study on concurrency analysis for improving the performance of the simplification process.
- Automated determination of cyclic sequences form non-interrupted observation of the input-output signals. This will allow the continuous updating of the identified model when new behaviour will be observed. The relaxing of the hypothesis that constraints the observations from a known initial state can be studied.
- The relaxing of the hypothesis that constraints the inputs and outputs to binary signals.
- The inclusion of time as a parameter of the identified model.

Appendix A

Interpreted Petri Nets

This section presents the basic concepts and notation of *PN* and IPN used in this thesis report.

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

$P = \{p_1,p_2,...,p_n\}$ and $T = \{t_1,t_2,...,t_m\}$ are finite sets of vertices named places and transitions respectively,

I (O) $P \times T \rightarrow Z^+$ is a function representing the weighted arcs going from places to transitions (transitions to places); Z^+ is the set of nonnegative integers.

Pictorially, places are represented by circles, transitions are represented by rectangles, and arcs are depicted as arrows.

The symbol ${}^\bullet t_j$ ($t_j{}^\bullet$) denotes the set of all places p_i such that $I(p_i,t_j)\neq 0$ ($O(p_i,t_j)\neq 0$). Such places are called input (output) places of t_j. Analogously, ${}^\bullet p_i$ ($p_i{}^\bullet$) denotes the set of all transitions t_j such that $O(p_i,t_j)\neq 0$ ($I(p_i,t_j)\neq 0$). Such transitions are called input (output) transitions of p_i.

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

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

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

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

This work uses Interpreted Petri Nets (IPN), an extension to *PN* that allow associating input and output signals to *PN* models.

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

G is a *PN* structure

$\Sigma = \{\alpha_1, \alpha_2, ..., \alpha_r\}$ is the alphabet of input symbols α_i.

$\Phi = \{\phi_1, \phi_2,..., \phi_q\}$ is the alphabet of output symbols ϕ_i.

$\lambda : T \rightarrow \Sigma\cup\{\varepsilon\}$ is a labelling function of transitions with the following constraint: $\forall t_j,t_k \in T$, j≠k, if $\forall p_i$ $I(p_i,t_j) = I(p_i,t_k) \neq 0$ and both $\lambda(t_j) \neq \varepsilon$, $\lambda(t_k) \neq \varepsilon$, then $\lambda(t_j) \neq \lambda(t_k)$; ε represents a system internal event externally uncontrollable

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

If a transition $t_j \in T$ of an IPN is enabled at marking M_k, there are two possibilities: a) If $\lambda(t_j) = \alpha_i \neq \varepsilon$ is provided to the system then t_j can be fired. b) If $\lambda(t_j) = \varepsilon$ and t_j is

enabled then t_j must be fired. When an enabled transition t_j is fired in a marking M_k, then a new marking M_{k+1} is reached. This behaviour is represented as $M_k \xrightarrow{t_j} M_{k+1}$; M_{k+1} can be computed using the state equation:

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

where C and v_k are defined as in PN and $y_k \in (Z^+)^q$ 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 A.4: If $\lambda(t_i)\neq\varepsilon$ the transition t_i is said to be controllable. Otherwise it is uncontrollable. A place $p_i \in P$ is said to be measurable if the i-th column vector of φ is not null, i.e. $\varphi(\bullet,i)\neq 0$. Otherwise it is non-measurable. In this way, places set can be partitioned in two subsets $P = P^m \cup P^u$ where P^m is the set of measurable places and P^u is the set of non-measurable places.

Definition A.5: A firing transition sequence of an IPN (Q, M_0) is a sequence $\sigma = t_i t_j \ldots t_k$ such that $M_i \xrightarrow{t_i} M_{i+1} \xrightarrow{t_j} \ldots M_{i+|\sigma|-1} \xrightarrow{t_k} M_{i+|\sigma|}$, where $|\sigma|$ is the length of the sequence σ. The Parikh vector $\vec{\sigma}$ of σ is the given by $\vec{\sigma}(r) = 1$ if t_r is in σ and $\vec{\sigma}(r) = 0$ otherwise. The set of all firing sequences is called the language of (Q, M_0) and it is denoted as $\pounds(Q, M_0) = \{\sigma | \sigma = t_i t_j \ldots t_k \; and \; M_i \xrightarrow{t_i} M_{i+1} \xrightarrow{t_j} \ldots M_{i+|\sigma|-1} \xrightarrow{t_k} M_{i+|\sigma|}\}$.

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

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

Definition A.8: A t-semiflow of a PN N is a nonnegative vector X fulfilling $C \cdot X = 0$. Consider a finite sequence σ of N which is enabled at marking M. The Parikh vector $\vec{\sigma}$ is a t-semiflow iff $M \xrightarrow{\sigma} M$.

Definition A.9: Let Q be an IPN. Q is event-detectable by the output if any transition firing can be uniquely determined by the knowledge of the output signals that it produces.

References

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

[Bekrar, 2006] R. Bekrar, N. Messai, N. Essounbouli, A. Hamzaoui, B. Riera, "Identification of Discrete Event Systems using Ordinary Petri Nets", *IAR Annual Meeting*, Nancy, Nov 2006.

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

[Booth, 1967] T.L. Booth, "Sequential Machines and Automata Theory", *John Wiley and Sons*, Inc. New York, London, Sidney, 1967.

[Cabasino, 2006a] M. P. Cabasino, Alessandro Giua, Carla Seatzu, "Identification of deterministic Petri nets", *Proceedings of the 8th International Workshop on Discrete Event Systems*, Ann Arbor, Michigan, USA, July 10-12, 2006.

[Cabasino, 2006b] M. P. Cabasino, Alessandro Giua, Carla Seatzu, "Computational complexity analysis of a Petri net identification procedure", *International Symp. on Nonlinear Theory and its Applications*, Bologna, Italy, Sep 2006.

[Cabasino, 2006c] M. P. Cabasino, A. Giua, Carla Seatzy, "Identification of unbounded Petri nets from their coverability graph", *Proceedings of the 45th IEEE Conference on Decision & Control*, San Diego, CA, USA, Dec 2006.

[Dotoli, 2006a] Ma. Dotoli, M. Pia Fanti, A. Marcello Mangini, "An Optimization Approach for Identification of Petri Nets", *Proceedings of the 8th International Workshop on Discrete Event Systems*, Ann Arbor, Michigan, USA, Jul 10-12, 2006.

[Dotoli, 2006b] M. Dotoli, M. P. Fanti, A. M. Mangini, "On-Line Identification of Discrete Event Systems: a Case Study", *Proceeding of the 2006 IEEE International Conference on Automation Science and Engineering*, Shangai, China, Oct 2006.

[Dotoli, 2007a] M. Dotoli, M. P Fanti, Agostino M. Mangini, "On Line Identification of Discrete Event Systems via Petri Nets: an Application to Monitor Specification", *Proceedings of the 3rd Annual IEEE Conference on Automation Science and Engineering*, Scottsdale, AZ, USA, Sep 2007.

[Dotoli, 2008] M. Dotoli, M. P Fanti, Agostino Marcello Mangini, "Real time identification of discrete event systems using Petri nets", *Automatica*, Volume 44, Issue 5, pp 1209-1219, May 2008.

[Estrada, 2009] 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, Oct 2009.

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

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

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

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

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

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

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

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

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

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

[Meda, 2000] M. Meda, A. Ramírez, E. López "Asymptotic Identification for DES", *IEEE Conference on Decision and Control*, Sydney, Australia, pp. 2266-2271, Dec 2000.

[Meda, 2000] M. Meda, A. Ramírez, E. López, "Behavioral properties for the control of discrete event systems modeled by interpreted Petri nets", *IEEE International Conference on Systems, Man, and Cybernetics*, Nashville, E.U., pp. 2150-2155, Oct 2000.

[Meda, 2001] M. Meda, E. López, "A passive method for on-line identification of discrete event systems", *IEEE International Conference on Decision and Control*, Orlando, Florida, E. U. pp. 4990-4995, Dec 2001.

[Meda, 2002] M. E. Meda-Campaña, "On-line Identification of Discrete Event Systems: Fundamentals and Algorithms for the Synthesis of Petri Net Models", Ph. D. Thesis, Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, Unidad Guadalajara, Nov 2002.

[Meda, 2002] M. Meda, E. López, "Incremental Synthesis of Petri Net Models for Identification of Discrete Event Systems", *IEEE Conference on Decision and Control*, Las Vegas, pp.805-810, EU Dec 2002.

[Meda, 2003] M. Meda, E. López, "Required Transition Sequences for DES identification", *IEEE Conference on Decision & Control (CDC 2003)*, Maui, Hawaii E. U. pp. 3778-3782, December 9-12, 2003.

[Meda, 2005] M. Meda, E. López, "Identification of concurrent Discrete Event Systems Using Petri Nets", *IMACS 2005 World Congress*, Paris, France, pp.1-7, Jul 2005.

[Richetin, 1984] M. Richetin, M. Naranjo and P Luneau, "Identification of Automata by Sequential Learning Pattern" *Recognition Letters 2*, Vol. 2, No. 6, pp. 379-385, Nort-Holland, 1984.

[Roth, 2009] Matthias Roth, Jean-Jacques Lesage, Lothar Litz, "A residual inspired approach for fault localization in DES" *Proceedings of the 2nd IFAC Workshop on Dependable Control of Discrete Event Systems,*pp. 347-352, Bary, Italy, Jun 2009.

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

[Valiant, 1984] L.G. Valiant, "A theory of the Learnable", *Comm ACM,* Vol. 27, pp. 1134-1142, 1984.

[Veelenturf, 1978] L.P.J. Veelenturf, *"Inference of Sequential Machines from Sample Computations"*, *IEEE Transactions on Computers*, Vol. 27, pp. 167-170, 1978.

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

CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL I.P.N.
UNIDAD GUADALAJARA

El Jurado designado por la Unidad Guadalajara del Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional aprobó la tesis

Identification of Concurrent Discrete Event Systems from Input-Output Sequences - Identificación de Sistemas de Eventos Discretos Concurrentes a partir de Secuencias de Entrada-Salida

del (la) C.

Ana Paula ESTRADA VARGAS

el día 21 de Agosto de 2009.

Dr. Luis Ernesto López Mellado
Investigador CINVESTAV 3B
CINVESTAV Unidad Guadalajara

Dr. José Luis Leyva Montiel
Investigador CINVESTAV 3B
CINVESTAV Unidad Guadalajara

Dr. Félix Francisco Ramos Corchado
Investigador CINVESTAV 3A
CINVESTAV Unidad Guadalajara

Dr. Antonio Ramírez Treviño
Investigador CINVESTAV 3A
CINVESTAV Unidad Guadalajara