Automated Modelling of Reactive Discrete Event Systems from External Behavioural Data

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

Automated Modelling of Reactive Discrete Event Systems from External Behavioural Data

Ana Paula Estrada-Vargas, Ernesto López-Mellado
CINVESTAV Unidad Guadalajara
Zapopan, Jal., Mexico
{aestrada, elopez}@gdl.cinvestav.mx

Jean-Jacques Lesage
LURPA Ecole Normale Supérieure de Cachan
Cachan, France
Jean-Jacques.lesage@lurpa.ens-cachan.fr

***Abstract*— This paper deals with automated modelling of reactive discrete event systems (DES). A software tool for building automatically interpreted Petri net models from an observed system's input/output sequence is presented. The tool is based on a black-box identification method that processes the input/output sequence, and synthesises and draws the model corresponding to such a sequence. First, the identification method is outlined; then the developed software is described and applied to an illustrative example from the manufacturing area.**

Keywords— Discrete event systems; Automated modelling; Stepwise identification; Interpreted Petri nets.

I. INTRODUCTION

Modelling is an important stage during the systems' developing life-cycle or during the analysis of an existing system. In the second situation the modelling is necessary when the original model is unknown or ill-known, or the system's operation has changed without updating the original model. Identification methods yield a mathematical model from data representing the behaviour exhibited during the system functioning; in the case of discrete event systems (DES) the obtained models are abstract machines that reproduce the observed sequences of events.

The above described problem has been addressed from several years ago in diverse domains. It has received alternative names such as learning techniques, grammatical inference, system identification, process mining, and process discovery.

The first identification methods appeared in the field of theoretical computer sciences as a problem of obtaining a language representation by finite automata (FA) from sets of accepted words; such methods have been reported as learning techniques [1][2]. Other related works use as description formalism Petri net (PN) models [3].

In recent years, the automation community has proposed identification approaches for obtaining approximated models (PN or FA) of DES whose behaviour is unknown or ill-known. In the context of automated manufacturing systems, identification methods allow obtaining a first model that can be detailed using established modelling techniques and available knowledge of the system; such a model describes the controller-plant behaviour during the closed-loop functioning. Three main approaches for identifying DES have been proposed in literature [4].

The incremental synthesis approach, proposed in [5] [6], deals with unknown partially measurable DES exhibiting cyclic behaviour. Several PN synthesis algorithms have been proposed allowing the on-line identification of concurrent DES from output sequences. Although the techniques are efficient, the obtained models may represent more sequences than those observed.

Other recent method [7] allows building efficiently a non deterministic FA (NFA) from a set of input/output sequences, measured from DES to be identified. The obtained NFA generates exactly the same input/output (I/O) sequences of given length than the observed ones. The method was conceived for fault detection in a model-based approach [8] and extended for obtaining an optimal partitioning of concurrent subsystems for distributed fault detection [9].

The off-line techniques based on integer linear programming (ILP) approach yield free-labelled Petri net models representing exactly the observed behaviour [10]. However both the ILP problem statement from event sequences and the processing have exponential complexity. This approach is being explored for other IPN classes; representative papers of this approach are [11] and [12].

In this paper the problem of identifying reactive DES composed by a controller (a Programmable logic Controller: PLC) and a plant operating in closed loop is addressed. Both controller's inputs and outputs are sampled from the initial state for building a single sequence of I/O vectors, which is processed yielding an interpreted Petri net (IPN) model.

This approach is based on a previously presented efficient method for coping with concurrent partially observable DES [13]. The method is composed by several polynomial time algorithms that process a set of cyclic input/output sequences yielding IPN models including silent transitions and non-labelled places.

This method has been extended and adapted for identifying actual industrial PLC-based controlled discrete manufacturing systems, which operate during a long time period performing repetitive tasks [14]; the proposed technique operates stepwise

allowing updating the model when new input/output vectors are added to the sequence. In the present paper we are focussing on the implementation and application of a software tool that automates the identification method.

The paper is organized as follows. In section II, the stepwise identification method is summarised. In section III a software tool implementing that method is described. Section IV illustrates the application of the developed identification tool to a case study.

II. A Stepwise Identification Method

In this section the identification method is outlined. A detailed description may be found in [14].

A. Problem statement

The method deals with DES composed of a *Plant* and a *Controller* (a PLC) operating in a closed-loop as showed in Figure 1. It is assumed that the data exchanged between plant and PLC corresponds to binary signals. The input signals of the PLC (outputs of the Plant) are generated by the sensors of the plant. The output signals of the PLC (inputs of the plant) control the actuators of the plant. The external behaviour of such a DES system can be observed (and recorded) by the evolution of the value of all input/output (I/O) signals exchanged between the controller and the plant.

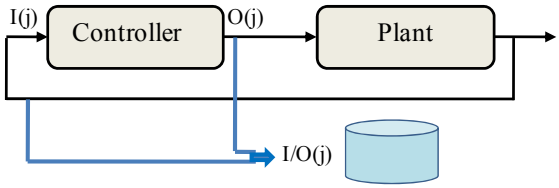


Figure 1. Closed loop controller-plant DES

At each end of cycle of the PLC, the current value of all Inputs and Outputs (called I/O vector) can be easily captured and recorded in a data base. Each new observed I/O vector (when at least one I/O changes its value) belongs to an I/O vector alphabet. The sequence is build with the recorded events; this sequence is the input of the identification algorithm.

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

Definition 2: The observed input/output sequence w of a DES S with m inputs and n outputs is:

$$w = \begin{bmatrix} I(1) \\ - \\ O(1) \end{bmatrix} \begin{bmatrix} I(2) \\ - \\ O(2) \end{bmatrix} \ldots \begin{bmatrix} I(j) \\ - \\ O(j) \end{bmatrix},$$

where $[I(j)|O(j)]^T \in \Sigma_{m,n}$ is the j-th observed I/O vector in w.

Definition 3: The observed input/output language of length l of a DES S is defined as $\mathcal{L}^l(S) = \{\varepsilon\} \cup \{w(i+1)\, w(i+2) \ldots w(i+h) | 1 \le i+h \le l\}$.

Now the identification problem can be defined. Given a DES whose only available information is an observed I/O sequence w arbitrarily large and an accuracy parameter κ, the aim of the identification process is to obtain a safe IPN model (Q, M_0) such that $\mathcal{L}^\kappa(Q, M_0) = \mathcal{L}^\kappa(S)$. The parameter κ is used to adjust the accuracy of the identified model, similarly as proposed in [7].

B. Purpose of identification

The aim of obtaining a model through identification is not to represent only the observed language, but to represent the observed behaviour and to infer actual behaviour that has not been observed during data collection. In order to accomplish this inference, the parameter κ is used as a measure of state equivalence. When κ-equivalent states (states whose past κ events are the same) are found, they are merged, increasing the accepted language of the IPN and consequently the modelled behaviour.

In this method it is considered that the I/O sequence is measured from the initial state of the global system (Plant and Controller). A data base is built with the sequence in which two consecutive I/O vector are different.

C. General strategy

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

From the I/O vector sequence, an event sequence is computed and a sequence of event substrings of length κ is built. Every substring is associated to a transition of a PN, which describes the causal relationship between event substrings.

A PN node path formed by non-observable places represents the substring sequence; this path is built taking into account the possible repetitive observed behaviour (internal model). Then simplifications may be applied. Notice that the number of non-observable places is not predefined.

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

D. Identification algorithm

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

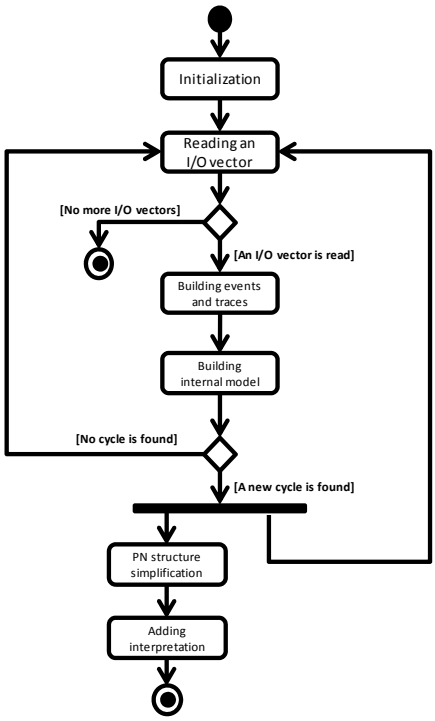


Figure 2. Stages of the identification algorithm

Consider a DES with three output signals, $\Phi = \{A, B, C\}$, and three input signals $\Sigma = \{a, b, c\}$. The entries of the binary I/O vectors have the following correspondence: $[a\ b\ c \mid A\ B\ C]^T$. Consider the following I/O sequence for which the first eight vectors are given below:

$$w = \begin{bmatrix}0\\0\\0\\\hline 0\\0\\0\end{bmatrix} \xrightarrow{e_1} \begin{bmatrix}1\\0\\0\\\hline 1\\0\\0\end{bmatrix} \xrightarrow{e_2} \begin{bmatrix}1\\0\\0\\\hline 0\\0\\0\end{bmatrix} \xrightarrow{e_3} \begin{bmatrix}1\\0\\0\\\hline 1\\0\\0\end{bmatrix} \xrightarrow{e_4} \begin{bmatrix}1\\0\\0\\\hline 0\\0\\0\end{bmatrix} \xrightarrow{e_5} \begin{bmatrix}1\\0\\0\\\hline 0\\1\\0\end{bmatrix} \xrightarrow{e_6} \begin{bmatrix}1\\0\\0\\\hline 0\\0\\0\end{bmatrix} \xrightarrow{e_7} \begin{bmatrix}0\\0\\0\\\hline 0\\0\\0\end{bmatrix} \xrightarrow{e_1} \begin{bmatrix}1\\0\\0\\\hline 1\\0\\0\end{bmatrix} \cdots$$

The algorithm starts by initializing a Petri net structure with a non observable place initially marked associated to the first observed vector $w(1) = [0\ 0\ 0 \mid 0\ 0\ 0]^T$. (*Step 1*).

The second I/O vector $w(2) = [1\ 0\ 0 \mid 1\ 0\ 0]^T$ of the sequence is considered and, according to the *Step 2* of the algorithm, an event vector $\tau(1) = w(2) - w(1) = e_1 = [1\ 0\ 0 \mid 1\ 0\ 0]^T$ is computed, as well as an input event vector $\lambda(1) = [1\ 0\ 0]^T$ and its corresponding symbolic input event $\lambda'(1) = a\_1$, i.e. the rising edge of a. Also, considering $\kappa = 1$, a first event trace $\tau^1(1) = e_1$ is computed. Notice that, in this case, trace and event are the same.

Step 3 of the identification algorithm relates computed event traces to transitions of an IPN. In this case $\tau^1(1)$ is related to $t_1^{e_1}$ (Figure 3).

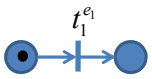


Figure 3. PN representing e_1

Considering the third I/O vector $w(3) = [1\ 0\ 0 \mid 0\ 0\ 0]^T$, the event $\tau(2) = w(3) - w(2) = e_2 = [0\ 0\ 0 \mid -1\ 0\ 0]^T$, the input event $\lambda(2) = [0\ 0\ 0]^T$ and the symbolic input event $\lambda'(2) = \varepsilon$ are computed; then the model is updated, as showed in Figure 4.

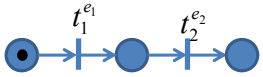


Figure 4. PN representing the sequence e_1e_2

Until 8th I/O vector, the situation is quite similar: a new event is computed and the model is updated. When 9th vector $w(9) = [1\ 0\ 0 \mid 1\ 0\ 0]^T$ is considered, the event $\tau(8) = w(9) - w(8) = e_1 = [1\ 0\ 0 \mid 1\ 0\ 0]^T$ is computed and the trace $\tau^1(8)$ is identified through *Step 3* as an already computed trace e_1. Since it leads to the same marking than the input place of $t_1^{e_1}$, a cycle is found and the model is updated as observed in Figure 5.

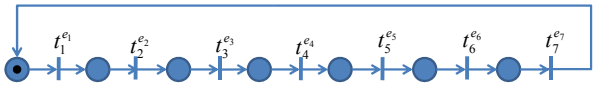


Figure 5. Internal model for the first detected cycle

Since a cycle has been found, *Step 4* and *Step 5* of the algorithm are executed, leading to an intermediate IPN model showed in Figure 6.

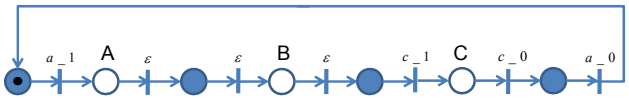


Figure 6. IPN for the first detected cycle

Simultaneously to the creation of the intermediate IPN, more I/O vectors can be processed from the observed sequence; consider the next subsequence of 10 vectors starting from $w(9)$:

$$\begin{bmatrix}1\\0\\0\\\hline 1\\0\\0\end{bmatrix} \xrightarrow{e_3} \begin{bmatrix}1\\0\\0\\\hline 0\\1\\0\end{bmatrix} \xrightarrow{e_2} \begin{bmatrix}1\\0\\0\\\hline 0\\0\\0\end{bmatrix} \xrightarrow{e_4} \begin{bmatrix}1\\0\\0\\\hline 0\\0\\0\end{bmatrix} \xrightarrow{e_7} \begin{bmatrix}0\\0\\0\\\hline 0\\0\\0\end{bmatrix} \xrightarrow{e_1} \begin{bmatrix}1\\0\\0\\\hline 1\\0\\0\end{bmatrix} \xrightarrow{e_3} \begin{bmatrix}1\\0\\0\\\hline 0\\1\\0\end{bmatrix} \xrightarrow{e_2} \begin{bmatrix}1\\0\\0\\\hline 0\\0\\0\end{bmatrix} \xrightarrow{e_4} \begin{bmatrix}1\\0\\0\\\hline 0\\0\\0\end{bmatrix} \xrightarrow{e_8} \begin{bmatrix}1\\1\\0\\\hline 0\\0\\0\end{bmatrix}$$

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

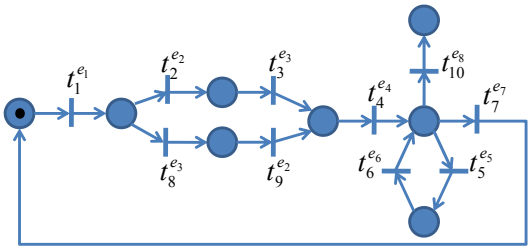


Figure 7. PN corresponding to the whole I/O sequence

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

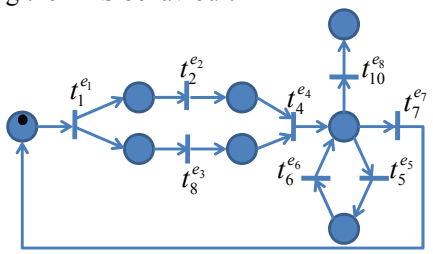


Figure 8. Equivalent internal model representing concurrency

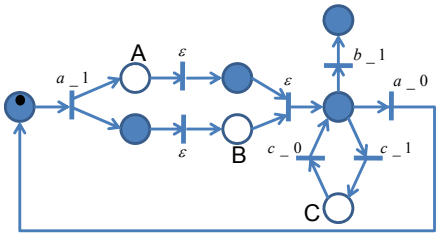


Figure 9. IPN for the complete sequence

III. AN IDENTIFICATION TOOL FOR AUTOMATED MODELLING

Based on the algorithms presented in section III, a software tool has been developed to automate the IPN model synthesis. It has been tested on several examples of diverse complexity which generate long sequences. The software architecture is shown in Figure 10.

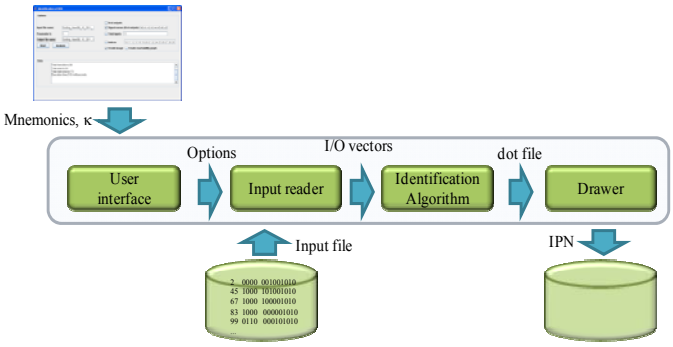


Figure 10. Software architecture

The algorithm has been implemented with IDE Netbeans 6.5, java jdk 1.6.0 [15]. In order to manipulate matrices, the library Java Matrix Package (Jama v1.0.2) [16] has been used. For creating the graphic model images, the hierarchical layout algorithm dot of Graphviz [17] has been applied. Actually, command dot can be invoked from the application without need to open the Graphviz interface.

An input file is a set of rows representing the cyclic sequence obtained from measure of the system to identify.

The user interface can be observed in Figure 11. In order to start an identification process, the user must insert the name of the file containing the observed I/O sequence, the desired name for the dot file (and the jpg file), the accuracy identification parameter κ, the index mask (if only some of the inputs or outputs of the I/O vectors will be considered), and the mnemonics of inputs and outputs.

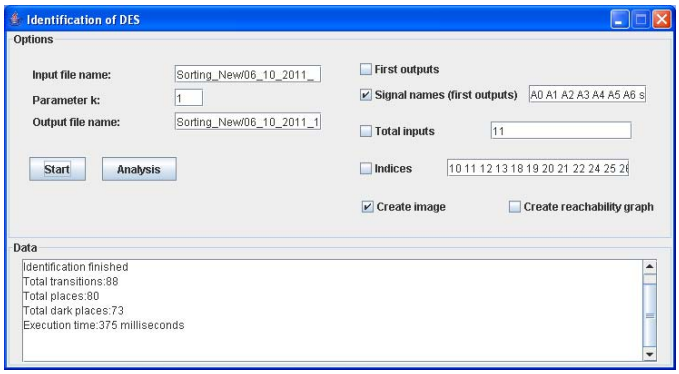


Figure 11. Graphic user interface

Once the identification algorithm has been executed, the user interface displays into the text area called *Data* some information about the identification process, such as the number of transitions and places of the IPN obtained and the execution time for identification.

Now, we present an experimental system on which the tool has been tested.

IV. CASE STUDY

The Interactive Training System for PLC® (ITS PLC) Professional Edition is a tool for PLC programming which offers virtual processes for education and training in PLC programming [18]. Each virtual process allows behavioural and visual simulation of an industrial process including virtual sensors and actuators; so its state can be sensed, and the components can be controlled by a real PLC. The sensors and actuators data is exchanged between the PLC and the system by a data acquisition board (DAQ) with 32 I/O isolated channels and USB interface.

For our experimental work, we have chosen the so called Sorting system, which transports parcels from a feeder to a couple of elevators; parcels are sorted according to their height (Figure 12). The controller handles 11 inputs (s0, s1, s2, s3, s4, s5, s6, s7, s8, s9, s10) and 7 outputs (A0, A1, A2, A3, A4, A5, A6).

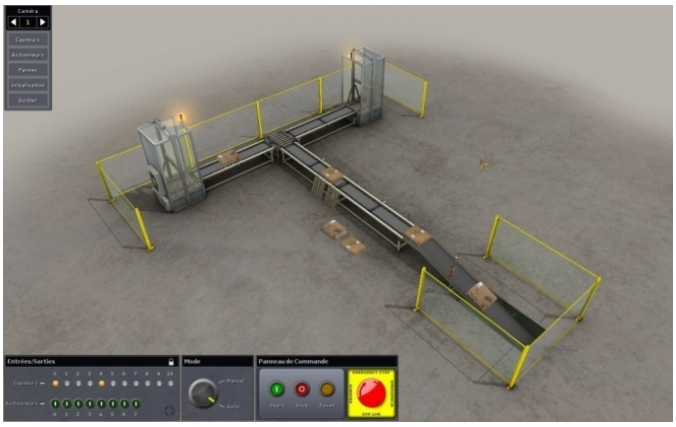


Figure 12. The sorting system from ITS PLC

For the collection of I/0 vectors, a routine written on Python has been launched in the computer using the ITS PLC (see Figure 13). It uses the Modbus communication protocol to read values of the inputs and outputs at each PLC cycle. The PLC is a Modicon TSX Premium.

If a measured I/O vector is equal to the previous one, it is ignored. Otherwise, the I/O vectors as well as the time value are written into a file. Figure 14 shows the experimental environment.

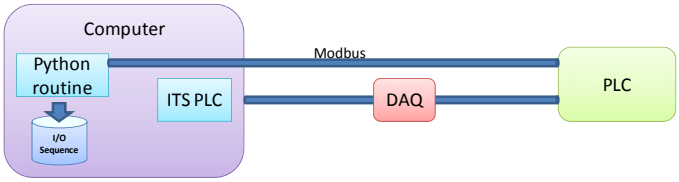


Figure 13. Scheme of the data collection procedure

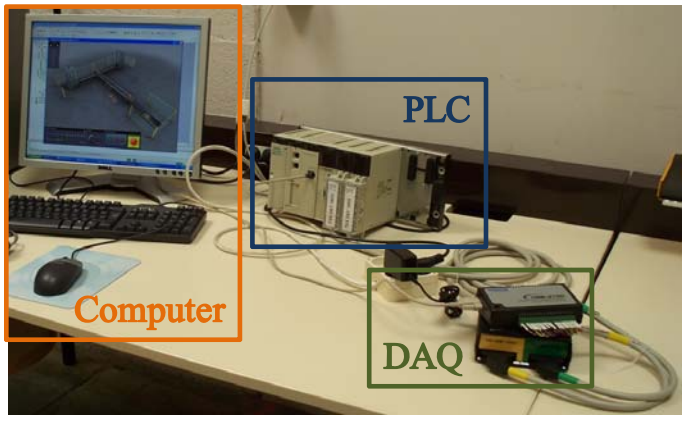
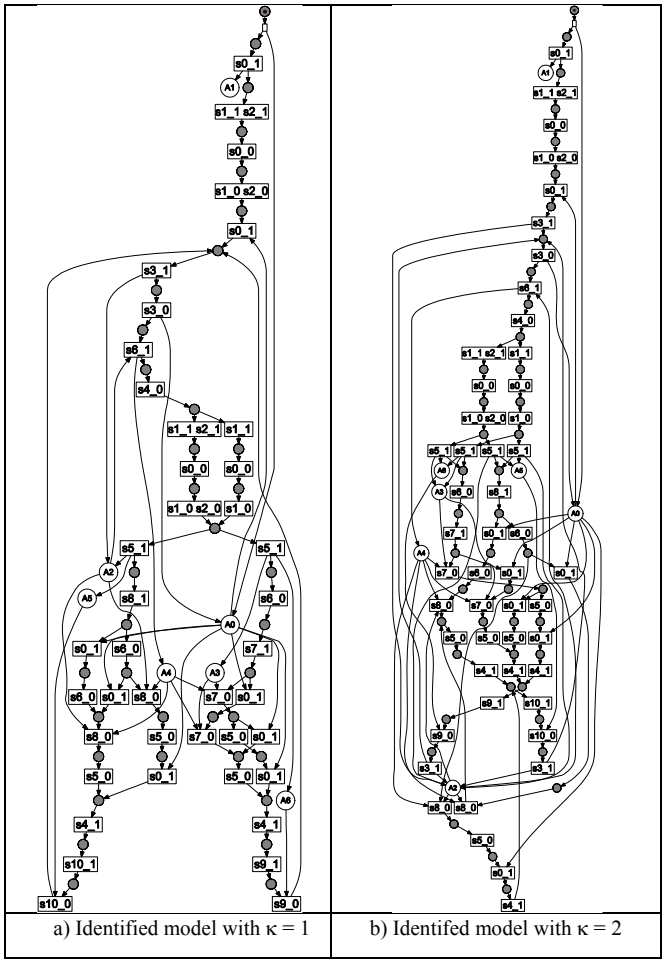


Figure 14 Experimental environment

After processing 30 parcels, the data collection has been stopped, obtaining a sequence of 472 I/O vectors in the input file. The identification procedure has been applied with different values of κ. Identified models for κ = 1 and κ = 2 are showed in Figure 15. The execution time for the identification was 156ms and 157ms respectively.

Observe how incrementing the value of κ provokes that several transition paths are created. However, the net computed with κ = 2 does not represent a significative change from the net computed with κ = 1 and thus, we decide that it is not necessary to compute new nets with largest values of κ, and the value κ = 1 is enough.



a) Identified model with κ = 1 b) Identifed model with κ = 2

Figure 15 Identified models for the Sorting system

By expertise knowledge, we have validated the obtained models by checking that they are able to reproduce the observed behaviour of the compound system formed by the plant and controller. However, it is hard for someone not familiarized with the system to know in detail how the system works by looking at the models.

Notice that in the IPN model there are numerous paths formed by non observable places. This is due to the observation of input changes that do not affect the outputs, but maybe they affect the controller state.

In order to obtain a more compact model, a simplification strategy has been presented in [13] and is recalled below. It consists in merging several places, representing internal behaviour whose detected events do not have effect on the

outputs, into a single one where an output event must occur. Consider the following I/O vector sequence involving one input *x* and two outputs *A, B*:

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

This sequence can be represented as follows: $A \xrightarrow{x\_1} A \xrightarrow{\varepsilon} B$, which can be transformed into a more compact one: $A \xrightarrow{x\_1} B$. This can be generalized to the following situation in which several events do not produce any change in the outputs:

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

The application of this simplification procedure yields the IPN model showed in Figure 16. The execution time to produce this model has been 125ms. Notice how the model is easier to read and gives a better notion of how the system is actually working.

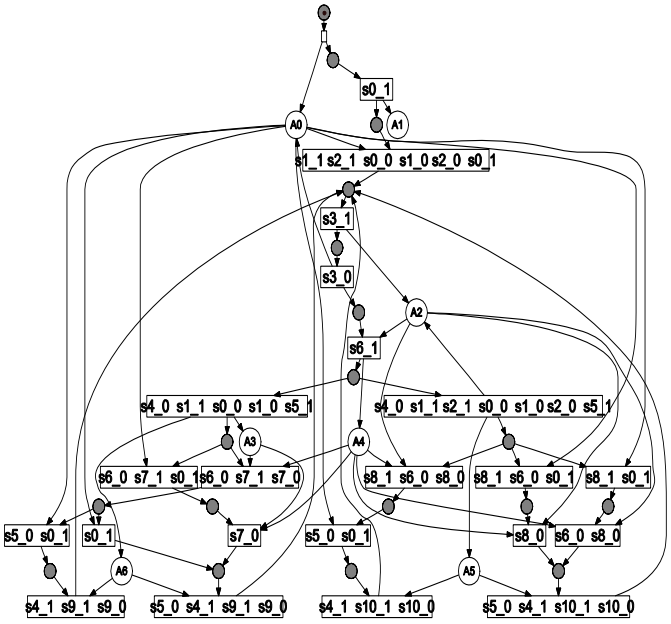


Figure 16. Reduced model for the sorting system

V. CONCLUDING REMARKS

Automated modelling of reactive DES can be achieved by efficient identification algorithms that cope with large and complex processes. A software tool based on identification algorithms is a valuable resource for computer-aided reverse engineering of controlled manufacturing systems, which allows obtaining a comprehensive Petri net model of the closed-loop controlled system.

Identified models approximate closely the actual behaviour of the compound system controller-plant. The processing of sequences, corresponding to observations during long production time, composed by thousands of I/O vectors, is performed in few seconds thanks to efficient algorithms issued from the identification method.

ACKNOWLEDGMENT

The first author is sponsored by CONACYT Mexico, Grant number 50312 and Région Ile de France.

REFERENCES

[1] E.M. Gold, "Language Identification in the Limit", *Information and Control*, Vol. 10, No.5 pp. 447-474, 1967

[2] D. Angluin, "Queries and Concept Learning", *Machine Learning*, Vol. 2, No.4 pp. 319-342, 1988

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

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

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

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

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

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

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

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

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

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

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

[14] A. P. Estrada-Vargas, J-J. Lesage, E. Lopez-Mellado "Stepwise Identification of Automated Discrete Manufacturing Systems" Proc. of IEEE International Conference on Emergency Technologies and Factory Automation, pp. 1-8, Toulouse, France. September 2011

[15] http://netbeans.org/

[16] http://math.nist.gov/javanumerics/jama/

[17] http://www.graphviz.org/

[18] http://www.realgames.pt/