An Optimization-Based Approach to Discover the Unobservable Behavior of a Discrete-Event System Through Interpreted Petri Nets

Francesco Basile, Gregory Faraut, Luigi Ferrara, Jean-Jacques Lesage

An optimization-based approach to discover the unobservable behaviour of a Discrete Event System through Interpreted Petri Nets

Francesco Basile[1], Gregory Faraut[2], Luigi Ferrara[1] and Jean-Jaques Lesage[2]

***Abstract*—This paper deals with the problem of discovering a Petri Net model of a discrete event system, starting from the observation of long event sequences. Precisely, given an Interpreted Petri Net (IPN) system modelling the relations between input and output events of the system (*i.e.* the reactive/observable behaviour), the internal state evolutions of the system (*i.e.* the unobservable behaviour) are firstly discovered and then modelled. The proposed unobservable discovery takes advantage of the novel concept of interpreted sequences, which better characterize the system and model the behaviour by considering both observable markings (outputs) and transition firings (inputs). The unobservable modelling is approached as a net synthesis problem. It relies on an optimization-based procedure that identifies the complementary structure; in particular, places only are added to the original model.**

Note to Practitioners. **Black-box identification procedures process an input/output sequence recorded for a long period of time during the functioning of a closed-loop controlled system, and then return a model of the system. However, even if these models simulate well the recorded sequence, they are not very accurate. Indeed, they simulate also other sequences, that in general are not admitted by the real system. The method proposed here aims to make more accurate these models by discovering the unobservable behaviour of a controlled system, related to evolutions of the internal state (and variables) of the system without changing the capability of simulating the observed behaviour.**

Keywords: Model discovery, Petri Nets, Discrete Event Systems, Identification.

I. Introduction

A. Position of the paper

The problem of identifying a Discrete Event System (DES) is of interest for many applications as, for example, reverse engineering [24] or control [27], [2], [28], [22] of (partially) unknown systems, fault diagnosis [23] or system verification. This task is typically approached by firstly observing the system and then modelling it. In particular, sequences of inputs and outputs are acquired during the operation of the system within its environment. The system is finally identified by building a model that can reproduce the observed sequences, simulating the observed behaviour.

The mathematical models mostly adopted in the literature for the identification of DESs are either Petri Nets (PNs) or finite state automata [17], [11]. When the resulting model is a PN, like in this paper, the net structure (places, transitions and arcs) and its initial marking must be computed. PNs have been successfully used to model and analyze many DESs (e.g. UML modelled systems [7], traffic sysytems [4], manufacturing systems [8] etc.) thanks to their powerful mathematical formalism that enables both qualitative and quantitative analysis.

The language of the identified model, that is the set of sequences it can generate from the initial marking, in general contains a subset of sequences that do not belong to the observed language [23]. Such a subset represents the exceeding language of the identified system. The size of the exceeding language is usually assumed as a measure of the fitness of the obtained model [10]. Indeed, a large exceeding language is certainly undesired when the identified model is used for diagnostic or verification purposes, while it is tolerated for reverse engineering.

The behaviour of a reactive DES, for instance a process and a controller in a closed-loop, can be split into an *Observable behaviour*, related to direct output changes depending on input changes, and an *Unobservable behaviour*, related to evolutions of the internal state (and variables) of the system without changes of observable data (inputs and outputs). Even if the available data only capture the observable behaviour, the identification algorithm should provide a model expressing both input/output causal relationships and internal state evolutions due to input changes [24].

While Petri Nets provide the semantics to express sequentiality, choices and parallelism, Interpreted Petri Nets (IPNs) [16] also add the input/output interpretation to transitions and places, thus being a natural choice to model reactive systems. IPN models are suitable for practical problems where the explicit representation of the physical input and output signals is required, such as model-based fault diagnosis and isolation.

This paper can be positioned in the continuation of [16] as the papers [26], [24]. In the first paper [16],

[1] F. Basile and L. Ferrara are with DIEM, Università di Salerno, Italy.

[2] G. Faraut and J.J. Lesage are with LURPA, ENS Cachan, Univ. Paris-Sud, Univ. Paris-Saclay, France.

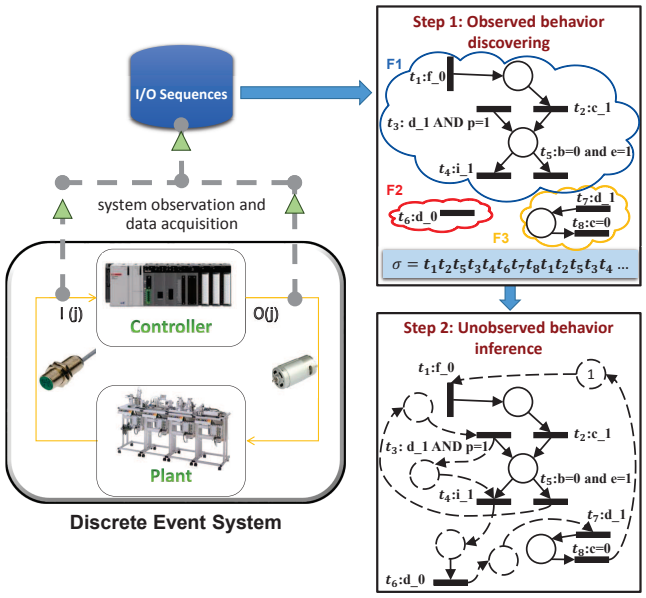

Fig. 1. Identification procedure of reactive systems in two steps.

a black-box identification procedure is provided that discovers the observable behaviour as IPN fragments from an input/output observed sequence. This sequence is also converted into a firing sequence on the alphabet of events associated to transitions (see first step in Fig. 1.). Therefore, the unobservable behaviour is discovered from such a firing sequence, and the IPN fragments are completed by adding connecting (unobservable) places (see second step in Fig. 1.). Such a second step is essential in determining the size of the exceeding language of the model.

Solutions to implement the second step have been presented in [16], [26], [24]. In [16], [26] the added unobservable places reflect the causal and concurrent relationships between transition firings in the firing sequence obtained in the first step. In [24] the projection of the firing sequence obtained in the first step on sub-alphabets is used to discover specific patterns that are characteristic of dependency relationships between the transition firings. Both the approaches return as identified model a 1-bounded net.

B. Paper contribution

This paper features a new approach for the unobservable discovery with the following objectives:

- To model the unobservable behaviour of a reactive DES by separating the dependency between the trajectories generated by the system and its initial state. Indeed, automation systems are designed to execute repetitive actions, and so it is reasonable that, after a certain number of observed events, the exhibited behaviour does not dependent on the initial state.
- To compute the unobservable places that model the unobservable behaviour without searching for specific patterns, as in the previous approaches [16], [26], [24]. As a result, the added places implement general constraints and further reduce the exceeding behaviour of the model.
- To make the accuracy and complexity of the identification procedure tunable with respect to a design parameter. Among the methods proposed for the unobservable discovery, only the one proposed in [24] can be tuned to reduce the exceeding behaviour of the identified model and the approach illustrated in this work will be shown to obtain more accurate models. Moreover, the observable net, as well as the identified unobservable one, are not required to be 1-bounded.
- The identification procedure is not required to be executable with real-time constraints and can be performed off-line.
- To improve the quality of the identified model in terms of accuracy with respect to other approaches. At this aim, the sequences of transition firings of the observable net obtained as result of the first stage of the procedure depicted in Fig. 1 are enriched by the observable markings reached during the firing of these sequences. This makes possible to define the exceeding language and the aforementioned distance with respect to sequences of transitions (associated to system inputs) and markings (associated to system outputs) and not transition only (as done in [16], [26], [24]).

In order to achieve the objectives listed above the following techniques/methodologies are used:

- A synthesis-based problem is set up to reduce the exceeding language. There are approaches to DES identification where it is assumed that either the whole state space of the system, or the whole language generated by it, is known [12], [19], [13], [14]. If this is the case, the tackled problem is more a *net synthesis* problem, rather than a *net identification* one. In this paper a net synthesis approach based on a graph is used. Precisely, the approach presented in this paper, inspired by [18], tries to make the reachability graph of the identified model isomorphic to a graph generating a behaviour having empty exceeding language with respect to words of length r, where r is the design parameter.
- In addition to the size of the exceeding language, that measures the size of the sequences not ob-

served but generated from the initial marking, the concept of distance of order k is introduced to count the number of (sub)words of length k that can be generated from any observable marking of the model while they have not been observed. The aforementioned graph generates a behaviour having also null distance of order k.

- An optimization approach based on an ILP formulation is used for the synthesis of unobervable places. This is in line with a recent trend in PN-related research which indicates that many analysis, control, fault identification and diagnosis problems can be more conveniently formulated and solved as optimization problems, usually in the form of ILP problems (see, *e.g.*, [9], [29], [15]). Indeed, it turns out that, despite their computational complexity, *optimization-based* approaches can be practically more convenient when compared to alternative solutions, since they rely on *off-the-shelf* optimized solvers, as opposed to *ad hoc* algorithms. In particular, various efficient software suites can be employed to tackle ILP problems, such as CPLEX® or FICO™ Xpress [1]

II. Systems, models and languages

In this work, the identification process is assumed to work on a rich data set of observations acquired from the automation system and on an *Interpreted Petri Net* (IPN) system modelling the observable behaviour.

A. Observation data set

Automation systems are typically composed of a plant and a controller in closed-loop, as represented in Fig.1; the controller is commonly a Programmable Logic Controller (PLC). In such a loop, the signals of the plant sensors are the inputs of the PLC; in turn, the outputs of the PLC control the actuators of the plant. For the objectives of this work, such systems are observed from the perspective of the PLC.

A PLC typically works by cyclically reading and storing inputs, executing user program(s), and finally writing the outputs (see Fig. 2). The READ-EXECUTE-WRITE cycle is called the *scan cycle*.

The input signals of the PLC are represented by the set I, where $|I|$ is the cardinality of the set I and $i_j \in I$ is the j-th input; similarly, the output signals of the system are represented by the set O and $o_j \in O$ is the j-th output. Each I/O signal is supposed to be binary.

In order to identify the system, sequences of I/O vectors are acquired during the observation of the system. We denote by V_j the sequence of I/O vectors acquired

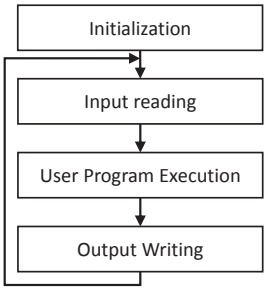

Fig. 2. PLC scan cycle.

at the j-observation; its length, noted $|V_j|$, depends on the acquisition duration

$$V_j = \left[\frac{l_I(1)}{l_O(1)}\right], \left[\frac{l_I(2)}{l_O(2)}\right], ..., \left[\frac{l_I(|V_j|)}{l_O(|V_j|)}\right]$$

where $l_I(k) = [l_{i_1}(k), ..., l_{i_{|I|}}(k)]^T$ and $l_O(k) = [l_{o_1}(k), ..., l_{o_{|o|}}(k)]^T$ are two vectors whose entries are, respectively, the levels of each signal associated to the $|I|$ inputs and $|O|$ outputs observed during the j-th acquisition at the k-th scan cycle.

The automation system exhibits a different behaviour depending on its initial state and on the occurred events over time. It is reasonable that each sequence V_j represents a trajectory generated by the system starting from a specific state, *e.g.* the on-state of the automation system or the state it reaches when a distinctive configuration is observed; this state is referred to as initial state of the system. Observations marked with a cycle (V_j^o) are called *cyclic* since it is assumed, on the basis of an additional knowledge about the observed system, that the state reached by the system at the end of the sequence is the initial state of the system.

The whole set of acquired observations is called *database* and is denoted by D. The acquired sequences are supposed to be sufficiently long and rich to capture the behaviour of the system as best as possible; indeed, the more data on the system is available, the closer to reality the built model is.

The possibility of marking a sequence as cyclic is a contribution of this paper; thanks to this feature, the database is lighter and more expressive, since it is only necessary to acquire each different cyclic trajectory instead of a single long observation which includes all the trajectories in any combination.

B. Background on Interpreted Petri nets

A brief recall on Petri Nets is presented in this section. For a complete review on PNs, the reader can refer to [21].

A *Place/Transition net* (P/T net) is a 4-tuple $N = (P, T, \mathbf{Pre}, \mathbf{Post})$ where P is a set of places, T

is a set of transitions and $\mathbf{Pre}$ and $\mathbf{Post}$ are the $|P| \times |T|$ sized, natural valued, incidence matrices; $\mathbf{Post}(p,t) = w$ means that there is an arc from $t \in T$ to $p \in P$ with weight w and $\mathbf{Pre}(p,t) = 0$ indicates no arc from p to t. The incidence matrix $\boldsymbol{C}$ of the net is defined as $\boldsymbol{C} = \mathbf{Post} - \mathbf{Pre}$.

A *marking* (the net state) is a vector $\boldsymbol{m} : P \to \mathbb{N}^{|P|}$, that assigns to each place a non-negative integer number of tokens. A P/T *system* or *net system* $\langle N, \boldsymbol{m}_0 \rangle$ is a net N with an initial marking $\boldsymbol{m}_0$.

A transition t is *state-enabled* at $\boldsymbol{m}$ *iff* $\boldsymbol{m} \geq \mathbf{Pre}(\cdot, t)$, noted by $\boldsymbol{m} \xrightarrow{t}$; its firing yields a new marking $\boldsymbol{m}' = \boldsymbol{m} + \boldsymbol{C}(\cdot, t)$, called *state equation*; it is denoted by $\boldsymbol{m} \xrightarrow{t} \boldsymbol{m}'$. The set of reachable marking is denoted by $R(N, \boldsymbol{m}_0)$. If $R(N, \boldsymbol{m}_0)$ is finite, it is possible to compute a graph whose nodes are each one associated to a single net marking; an arc labeled by t exiting from a node associated to $\boldsymbol{m}$ and entering a node associated to $\boldsymbol{m}'$ is associated to each state transition. Such a graph is called *reachability graph* [21].

Extending this reasoning, a sequence $\sigma = t_1 t_2 ... t_l$ is state enabled at $\boldsymbol{m}_1$ *iff* $\boldsymbol{m}_1 \xrightarrow{t_1} \boldsymbol{m}_2 \xrightarrow{t_2} ... \xrightarrow{t_l} \boldsymbol{m}_{l+1}$ and it is denoted by $\boldsymbol{m}_1 \xrightarrow{\sigma}$; the marking $\boldsymbol{m}_{l+1}$ can be easily computed as $\boldsymbol{m}_{l+1} = \boldsymbol{m}_1 + \boldsymbol{C}\boldsymbol{\sigma}$, where $\boldsymbol{\sigma} : \sigma \to \mathbb{N}^{|T|}$ is the firing count vector associated to σ, whose k-th component is the number of occurrences of t_k in σ; furthermore, $\boldsymbol{e}_t$ represents the firing count vector associated to $\sigma = t$.

We denote by $\mathcal{E}(N, \boldsymbol{m}_0, \sigma)$ the set of state enabled transitions at $\boldsymbol{m} = \boldsymbol{m}_0 + \boldsymbol{C}\boldsymbol{\sigma}$ in the net system $\langle N, \boldsymbol{m}_0 \rangle$, *i.e.* $\mathcal{E}(N, \boldsymbol{m}_0, \sigma) = \{t \in T \mid \boldsymbol{m} \xrightarrow{t}, \boldsymbol{m} = \boldsymbol{m}_0 + \boldsymbol{C}\boldsymbol{\sigma}\}$.

To explicitly represent the I/O relationships of a closed-loop controlled system, Interpreted Petri Nets (IPNs) are adopted. An IPN is defined as $N = (\overline{P}, T, \overline{\mathbf{Pre}}, \overline{\mathbf{Post}}, \gamma, \beta)$ and such that:

- $\overline{P}$ is a set of places, T is a set of transitions and $\overline{\mathbf{Pre}}$ and $\overline{\mathbf{Post}}$ are the pre– and $post$–incidence functions that specify the arcs.
- $\gamma : \overline{P} \to O \cup \{\epsilon\}$ is a mapping function which associates to each place an output signal (observable place) or ϵ (unobservable place). The set of places is partitioned into two disjoint sets, the set of observable places P and the set unobservable places P^u, so $\overline{P} = P \cup P^u$ and $P \cap P^u = \emptyset$. As consequence, the following holds:
$$\overline{\mathbf{Pre}} = \begin{bmatrix} \mathbf{Pre} \\ \mathbf{Pre}^U \end{bmatrix}, \overline{\mathbf{Post}} = \begin{bmatrix} \mathbf{Post} \\ \mathbf{Post}^U \end{bmatrix}, \overline{\boldsymbol{C}} = \begin{bmatrix} \boldsymbol{C} \\ \boldsymbol{C}^U \end{bmatrix}.$$
- $\beta : T \to \{0,1\}$ is the logical condition function of transitions and such that $\forall t_i \in T, \beta(t_i) = F_i(I, E_I)$, where $E_I = \{\uparrow i_i (\downarrow i_i) \mid i_i \in I\}$ is the set of the input signals' rising (falling) edges and $F_i(I)$ depicts the conditions on the input signals and edges to fire t_i.

The marking of an IPN is denoted by $\overline{\boldsymbol{m}}$, where $\overline{\boldsymbol{m}} = \begin{bmatrix} \boldsymbol{m} \\ \boldsymbol{m}^U \end{bmatrix}$; the subvector $\boldsymbol{m}$ is called *observable marking* and $\boldsymbol{m}^U$ is called *unobservable marking*.

An *Interpreted PN system* $\langle N, \overline{\boldsymbol{m}}_0 \rangle$ is an IPN N with an initial marking $\overline{\boldsymbol{m}}_0$. In an IPN system, a transition t fires at $\overline{\boldsymbol{m}}$ *iff* it is state enabled and logical condition enabled, *i.e.* $\overline{\boldsymbol{m}} \xrightarrow{t} \wedge \beta(t)$.

C. Background on automata

Automata are used in this work to represent the observed dynamics.

A *Moore Machine* (MM) is a 6-tuple $A = (Q, E, \delta, q_0, \Lambda, \lambda)$ consisting of a finite set of states Q, the finite set of input transition events E, a transition function $\delta : Q \times E \to Q$, an initial state $q_0 \in Q$, a finite set of outputs Λ and an output function $\lambda : Q \to \Lambda$. The output function λ is surjective, since each output $l \in \Lambda$ has to be associated to at least one state $q \in Q$.

The set of admissible events in a state $q \in Q$ can be defined as $\mathcal{A}(q) = \{e \in E \mid \delta(q, e) \text{ is defined}\}$; if an admissible event e occurs in q, a *state transition* occurs and $q' = \delta(q, e)$ is reached, denoted by $q \xrightarrow{e} q'$. As for IPNs, multiple state transitions can be considered at once: a *production* s from the state $q_1 \in Q$ is a sequence $s = e_1 e_2 ... e_l$ such that $q_1 \xrightarrow{e_1} q_2 \xrightarrow{e_2} ... \xrightarrow{e_l} q_{l+1}$.

A MM can be represented by a directed graph, which associates a node to each state and an edge labelled by $e \in E$ to each couple (q, q') such that $q \xrightarrow{e} q'$.

III. Problem statement

The goal of this paper is to identify the unobservable places to be added to an existing IPN model to make it more accurate. In this section, some assumptions on the input model are made and some concepts are introduced to measure the deviance between the given model and the desired one. Then, the problem statement is formally defined as an optimization problem looking for a net minimizing the deviance with respect to the desired behaviour.

A. Assumptions

Assumption 1. In this work, the automation system is assumed to be already modeled by an IPN system $\langle N^{Obs}, \boldsymbol{m}_0 \rangle$, where $N^{Obs} = (P, T, \mathbf{Pre}, \mathbf{Post}, \gamma, \beta)$; this model is also called *observable IPN system*, since it represents the reactive, observable behaviour of the automation system and only contains observable pieces,

as Fig. 1 depicts [24]. It consists in general of different disjoint fragments representing causal relations between inputs and outputs of the system and is not bounded; isolated transitions are also admitted. □

Assumption 2. The database D of acquired input/output observations is assumed to be available. Such a set is assumed to be simulated by $\langle N^{Obs}, \boldsymbol{m}_0 \rangle$ in the sense that, for each observation $V_j \in D$, giving as input the input signals of V_j, the IPN system will produce the same outputs of V_j, in sequence. □

B. Interpreted sequences and model languages

In this section, we characterize the behaviour that a model exhibits in terms of generated sequences. This task is fundamental to evaluate the capability of the model to accurately represent the system.

At this aim, the novel concept of *interpreted sequences* is introduced. These sequences, consisting of transition firings and observable markings, are defined to represent the dynamic behaviour in terms of inputs and outputs that the IPN system is able to model. As it will be shown, a more accurate unobservable discovery can be achieved exploiting the definition of interpreted sequences.

Definition 3. An interpreted sequences is a sequence of observable markings and transition firings, always starting and ending with an observable marking. Let $i\sigma = \boldsymbol{m}_1$ be an *interpreted sequence* of length 1, let $i\sigma = \boldsymbol{m}_1 t_1 \boldsymbol{m}_2 t_2 ... t_{n-2} \boldsymbol{m}_{n-1} t_{n-1} \boldsymbol{m}_n$ be the interpreted sequence of length n, then $|i\sigma|$ denotes the length of $i\sigma$, $first(i\sigma) = \boldsymbol{m}_1$, $last(i\sigma) = \boldsymbol{m}_n$.

It is useful to obtain the firing sequence of an interpreted sequence. At this aim, the following definition comes in help.

Definition 4. The projection operator $\Pi : i\sigma \rightarrow \sigma$ is defined as follows: $\Pi(i\sigma) \equiv \epsilon$, if $i\sigma = \boldsymbol{m}$, while $\Pi(i\sigma) \equiv t_1 t_2 ... t_{n-2} t_{n-1}$, if $i\sigma = \boldsymbol{m}_1 t_1 \boldsymbol{m}_2 t_2 ... t_{n-2} \boldsymbol{m}_{n-1} t_{n-1} \boldsymbol{m}_n$. Note that the length of the firing sequence $\Pi(i\sigma)$ is equal to $|i\sigma| - 1$. □

Given two interpreted sequences $i\sigma_1$ and $i\sigma_2$ such that $last(i\sigma_1) = first(i\sigma_2)$, it makes sense to define their concatenation $i\sigma_1 \, i\sigma_2$ as the sequence consisting of the string $i\sigma_1$ immediately followed by the string $i\sigma_2$ except for the first marking of $i\sigma_2$ that is removed/merged with the last marking of $i\sigma_1$. Note that $\Pi(i\sigma_1 \, i\sigma_2) = \Pi(i\sigma_1)\Pi(i\sigma_2)$.

Observable markings and transition firings of an interpreted sequence can be associated to two kinds of observable symbols that the IPN system is able to generate; in particular, transition firings generate symbols related to physical input events and are contained in a set U, while observable markings generate symbols related to physical output events and are contained in a set Y. We denote by l the labelling function which associates the appropriate symbols to transition firings and observable markings.

For the sake of simplicity, the firing of the i-th transition is associated to the symbol t_i, *i.e.* $l(t_i) = t_i, U = \{\cup_i \, l(t_i)\}$. Similarly, we associate to the i-th observable marking that the IPN system can reach from the initial marking the symbol $\boldsymbol{m}_i$, *i.e.* $l(\boldsymbol{m}_i) = \boldsymbol{m}_i, Y = \{\cup_i \, l(\boldsymbol{m}_i)\}$.

Thus, given the symbols for transition firings and observable markings, the symbolization of an interpreted sequence can be defined. At this aim, let $S = Y \cup U$ be the set of symbols, the symbolization of $i\sigma$ is $w = l(i\sigma) \equiv l(\boldsymbol{m}_1) l(t_1) l(\boldsymbol{m}_2) ... l(t_{n-1}) l(\boldsymbol{m}_n) = \boldsymbol{m}_1 t_1 \boldsymbol{m}_2 ... t_{n-1} \boldsymbol{m}_n$ and is called *word*. Note that the length of the word w is $|w| = |i\sigma|$; furthermore, $last(w) = l(\boldsymbol{m}_n)$.

As for interpreted sequences, concatenation can be defined also for words: given $w_1 = l(i\sigma_1)$ and $w_2 = l(i\sigma_2)$, their concatenation is $w_1 w_2 = l(i\sigma_1 i\sigma_2)$. Thanks to concatenation, a generic word w can always be written as $w = w_1 w_2 w_3$, where $w_1, w_2, w_3 \in S^*$ are words and w_1 is called *prefix* of w, w_2 is a *subword* of w and w_3 is the *suffix* of w.

The set of words of length n generated by an IPN system can be defined as follows: $W^n(N, \overline{\boldsymbol{m}}_0) = \{l(i\sigma) \in S^* \mid \overline{\boldsymbol{m}}_0 \xrightarrow{\sigma}, \sigma = \Pi(i\sigma), |i\sigma| = n\}$. The generated language of length n is defined as $L^n(N, \overline{\boldsymbol{m}}_0) = \bigcup_{k \leq n} W^k(N, \overline{\boldsymbol{m}}_0)$. The generated language of the net is denoted by $L(N, \overline{\boldsymbol{m}}_0)$ and is defined as $L(N, \overline{\boldsymbol{m}}_0) = \bigcup_n L^n(N, \overline{\boldsymbol{m}}_0)$.

The concepts just introduced for IPNs can be analogously defined for Moore machines. We will use automata during the synthesis of the unobservable subnet. At this aim, the set of transition events E is chosen to be isomorphic to T, *i.e.* each event $e_i \in E$ is conveniently noted by $t_i \in T$. The set of outputs of the Moore machine Λ satisfies the relation $\Lambda \subseteq Y$.

Language definitions can finally be stated also for automata. Precisely, the set of words of length n generated by a Moore Machine A from q_0 is defined as: $W^n(A) = \{[\lambda(q_0) t_1 \lambda(q_1) t_2 ... t_{n-1} \lambda(q_{n-1})] \mid t_i \in E, \exists \, q_1, q_2 ... q_{n-1} \in Q : q_i = \delta(q_{i-1}, t_i), \forall i \geq 1\}$. The generated language of length n is defined as $L^n(A) = \bigcup_{k \leq n} W^k(A)$. The generated language of the MM is denoted by $L(A)$ and is defined as $L(A) = \bigcup_n L^n(A)$.

C. The observed language

Given the definitions of languages generated by IPNs and automata, it is fundamental to also define the language observed from the automation system in a similar way.

At this aim, it is firstly necessary to define the interpreted sequences that are directly derived from the observed I/O sequences. Given the database D, the set of *observed interpreted sequences* $I\Sigma = \{i\sigma_1, ..., i\sigma_{|D|}\}$ is obtained. Precisely, simulating each sequence $V_j \in D$ by $\langle N^{Obs}, \boldsymbol{m}_0 \rangle$ and recording each transition firing and the marking it yields, an interpreted sequence $i\sigma_j$ is obtained. Note that if the sequence V_j^o is cyclic, the associated interpreted sequence $i\sigma_j^o$ is cyclic also.

Secondly, the set of the observations is defined:

$$Obs = \{ (I\Sigma^o)^* I\Sigma^{\overline{o}} + I\Sigma^{\overline{o}} \},$$

where $I\Sigma^o$ contains cyclic sequences only while $I\Sigma^{\overline{o}}$ contains non cyclic sequences only, and it holds: $I\Sigma = I\Sigma^o \cup I\Sigma^{\overline{o}}, I\Sigma^o \cap I\Sigma^{\overline{o}} = \varnothing$. Note that the set of the observations Obs, as required, takes into account both observed inputs (through transition firings) and outputs (through observable markings); in subsection III-E an in-depth motivation for this is illustrated. The set of the symbolized observations can then be defined as $S_{Obs} = \{l(i\sigma), \forall i\sigma \in Obs\}$, while the observed language finally is: $L_{Obs} = \{w \in S^* \mid \exists w' \in S^* : w\, w' \in S_{Obs}\}$, which is prefix-closed by definition. L_{Obs} can be composed of words of infinite length. It is useful to introduce a length parameter, as for the language produced by a model. Thus, the observed language of length n is: $L^n_{Obs} = \{w \in L_{Obs} \mid |w| \leq n\}$

Note that, due to Assumption 2, the given observable IPN system satisfies the relation $L(N^{Obs}, \boldsymbol{m}_0) \supseteq L_{Obs}$; the relation of inclusion between these two sets remarks that the model could also produce never observed words, *i.e.* it does not fit with the observations.

D. The model accuracy and the target IPN system

In this section, the concept of *accuracy* of a model is illustrated. At this aim, we refer to a generic model M of the automation system, which either is an IPN model ($M = \langle N, \overline{\boldsymbol{m}} \rangle$) or an automaton model ($M = A$). This is done without loss of generality, since the generated languages are both based on transition firings and observable markings.

The simplest indicator that measures the degree of fitness of a model with the observations takes into account the difference between the words produced by the model and the observed ones. Assuming that a good database has been acquired, the following definition formalizes it.

Definition 5. Consider a model M such that $L(M) \supseteq L_{Obs}$; then, the *exceeding language* of length n of the model is defined as $L^n_{Exc}(M) = L^n(M) \setminus L^n_{Obs}$. □.

The smaller is the exceeding language, the better the model fits with the observations: if $L^n_{Exc}(M) = \varnothing \, \forall n$, it means that the model is able to perfectly reproduce from the initial state the sequential behaviour observed from the system. Such a perfect fit, however, is quite never the target for a good model. In fact, since real systems usually exhibit a rich behaviour, long sequences are needed to capture them, and consequently large values of n should be considered. However, observations are generally not complete, *i.e.* not all the possible trajectories are observed. Thus, building a model that perfectly fits with the acquired trajectories implies that new (future) observations cannot be generated by the identified model.

The empirical experience usually suggests that it makes sense to devise a tolerance length $\tilde{n}$ able to preserve a rich set of observations. In this case a good model M is such that:

- words of length $\tilde{n}$ produced by the model from the initial state are exactly the ones observed from the initial state of the automation system;
- only words $w = \boldsymbol{m}_1 t_1 \boldsymbol{m}_2 t_2 ... t_{\tilde{n}-1} \boldsymbol{m}_{\tilde{n}}$ that were observed from the automation system can be produced by the model; thus, the behaviour of the model M within any horizon that includes $\tilde{n}$ state transitions is coherent with the observations.

The two constraints are both necessary: if the first one only is imposed, unknown words of length $n \leq \tilde{n}$ could be produced after the first $\tilde{n}$ steps from the initial state; if the second one only is imposed, some words of length $n \leq \tilde{n}$ could be produced by the model from the initial state even if they were observed from a state of the system different from the initial one.

In order to formalize these concepts, the operator $\mathcal{V}$ is firstly introduced. It can be applied to the observed language or to the language L generated by a model M and is defined as:

$$\mathcal{V}(L, n) = \{w \in S^* \mid \exists w^i, w^e \in S^* : w^i\, w\, w^e \in L \wedge |w| = n\}.$$

In words, the $\mathcal{V}$ operator yields all the subwords of length n that can be picked from the given language L, starting from any observable marking of the contained words.

Applying this operator to the observed language, *known* words are yielded.

Definition 6. The set of *known* words of length n is defined as: $K^n_{Obs} = \mathcal{V}(L_{Obs}, n)$. □

Thus a word is said *known iff* it was produced from

any state of the automation system, reached from the initial one, during the observation of its behaviour.

In general, a model could produce, at any state reachable from the initial one, words that are not known. In order to evaluate this deviance, the distance between the language L of the model and the observed language is introduced.

Definition 7. Given a model M such that $L(M) \supseteq L_{Obs}$, the distance of order k is defined as: $d^k(M) = |\mathcal{V}(L(M), k) \setminus \mathcal{V}(L_{Obs}, k)|$. □

In words, the distance of order k between the language generated by the model and L_{Obs} yields the number of (sub)words of length k that start from any observable marking and that L contains but L_{Obs} does not.

In the following example, the introduced definitions are illustrated.

Example 8. Assume that the reactive behaviour of a system has been identified through the IPN system $\langle N^{Obs}, \boldsymbol{m}_0 \rangle$ in Fig. 3.

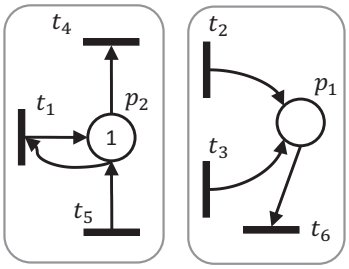

Fig. 3. $\langle N^{Obs}, \boldsymbol{m}_0 \rangle$ depicting the observable behaviour of a system, given under the form of two fragments. The conditions associated to transitions have been omitted.

The dynamic behaviour is represented by $I\Sigma = \{i\sigma_1, i\sigma_2, i\sigma_3^o\}$ where $i\sigma_1 = \boldsymbol{m}_0 t_1 \boldsymbol{m}_0 t_3 \boldsymbol{m}_2 t_1 \boldsymbol{m}_2 t_1 \boldsymbol{m}_2 t_1 \boldsymbol{m}_2 t_4 \boldsymbol{m}_1$, $i\sigma_2 = \boldsymbol{m}_0 t_1 \boldsymbol{m}_0 t_2 \boldsymbol{m}_2 t_1 \boldsymbol{m}_2 t_1 \boldsymbol{m}_2 t_4 \boldsymbol{m}_1 t_5 \boldsymbol{m}_2 t_1 \boldsymbol{m}_2 t_1 \boldsymbol{m}_2 t_4 \boldsymbol{m}_1$, $i\sigma_3^o = \boldsymbol{m}_0 t_3 \boldsymbol{m}_2 t_6 \boldsymbol{m}_0 t_1 \boldsymbol{m}_0$; only the last sequence is cyclic. The observable markings $\boldsymbol{m}_i = [m(p_1)\, m(p_2)]^T$ are: $\boldsymbol{m}_0 = [0\,1]^T, \boldsymbol{m}_1 = [1\,0]^T, \boldsymbol{m}_2 = [1\,1]^T$.

First, consider the unitary language length, *i.e.* $n = 1$. The generated language of length 1 is $L^1(N^{Obs}, \boldsymbol{m}_0) = \{\boldsymbol{m}_0\}$, that is equal to the observed language of length 1, thus $L^1_{Exc}(N^{Obs}, \boldsymbol{m}_0) = \varnothing$. As for the distance measure, it holds $d^1(N^{Obs}, \boldsymbol{m}_0) = \infty$: indeed, source transitions t_2, t_3, t_5 generate infinite observable markings.

If $n = 2$ is chosen, $L^2(N^{Obs}, \boldsymbol{m}_0) = \{\boldsymbol{m}_0, \boldsymbol{m}_0 t_1 \boldsymbol{m}_0, \boldsymbol{m}_0 t_5 \boldsymbol{m}_3, \boldsymbol{m}_0 t_4 \boldsymbol{m}_4, \boldsymbol{m}_0 t_2 \boldsymbol{m}_2, \boldsymbol{m}_0 t_3 \boldsymbol{m}_2, \boldsymbol{m}_0 t_6 \boldsymbol{m}_0\}$, where $\boldsymbol{m}_3 = [0\,2]^T, \boldsymbol{m}_4 = [0\,0]^T$; however $L^2_{Obs} = \{\boldsymbol{m}_0, \boldsymbol{m}_0 t_1 \boldsymbol{m}_0, \boldsymbol{m}_0 t_3 \boldsymbol{m}_2\}$, thus $L^2_{Exc} = \{\boldsymbol{m}_0 t_5 \boldsymbol{m}_3, \boldsymbol{m}_0 t_4 \boldsymbol{m}_4, \boldsymbol{m}_0 t_2 \boldsymbol{m}_2, \boldsymbol{m}_0 t_6 \boldsymbol{m}_0\}$. The distance of length 2 is clearly infinite.

The given IPN system is not a good model of the system, since meaningless observable markings are reached, *i.e.* the ones that are not 1-bounded; in addition, it contains exceeding words for $n = 2$ already. Proper unobservable places should thus be added to better fit the model with the observations. ■

Thanks to the illustrated definitions, the treated problem can be stated as minimization problem on both the exceeding language and the distance between the observed language and the language produced by the model. At this aim, the concept of k-completeness is introduced. For the sake of rigor, the term k-completeness has been firstly used in [20] with a weaker meaning. Here this term is extended to include the exceeding language.

Definition 9. Consider a model M such that $L(M) \supseteq L_{Obs}$; it is said to be k-complete *iff* it holds $L^k_{Exc}(M) = \varnothing$ and $d^k(M) = 0$. □

Clearly, the main target is the building of a $\tilde{n}$-complete model. At this aim, the design parameter $\tilde{n} \leq r \leq l^{i\sigma}_{max} \equiv \max_{i\sigma_k \in I\Sigma} |i\sigma_k|$ is introduced; note that $\tilde{r}$-completeness entails $\tilde{n}$-completeness. If $r = l^{i\sigma}_{max}$ is chosen, the desired model is said *maximally accurate*, since it perfectly fits with the observations.

The problem statement can be finally expressed.

Problem statement. Given an IPN system $\langle N^{Obs}, \boldsymbol{m}_0 \rangle$ modelling the automation system, the goal is to construct a new IPN system $\langle N', \overline{\boldsymbol{m}'_0} \rangle$ so that it is r-complete, by adding a set of unobservable places to $\langle N^{Obs}, \boldsymbol{m}_0 \rangle$. □

In conclusion, note that $\tilde{n}$ cannot be a free parameter. Indeed, the value $l^{i\sigma}_{max}$ is clearly an upper bound for $\tilde{n}$. A lower bound also exists; indeed $\tilde{n} \geq 2$ must hold since at least one transition firing must be considered. However, in case of cyclic sequences in the acquired database, the relation $\tilde{n} \geq l^{i\sigma^o}_{max}$ must hold, where $l^{i\sigma^o}_{max} = \max_{i\sigma \in I\Sigma^o} |i\sigma|$; indeed, the identity of cyclic sequences is preserved and made discernable from any other observed behaviour.

E. The importance of markings in language definitions

Standard definitions of Petri net languages are usually based on transition firings only, while in this work also observable markings are used. The main motivation is that both inputs (transitions) and outputs (observable places) are representative of the automation system and must be taken into account when evaluating the generated language of the model; indeed, if only sequences of transitions are considered for the unobservable discovery, then a less accurate model is obtained.

To prove this difference, an example is now given. At this aim, languages generated by sequences of transitions

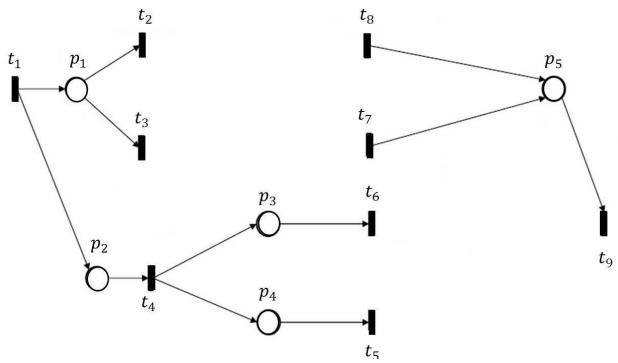

Fig. 4. The observable IPN system $\langle N^{Obs}, \boldsymbol{m}_0 \rangle$.

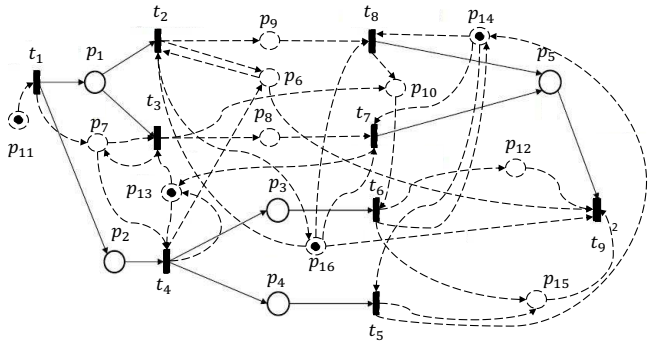

Fig. 5. The IPN system $\langle N, \overline{\boldsymbol{m}}_0 \rangle$ satisfying $\tilde{d}^2(N, \overline{\boldsymbol{m}}_0) = 0$ and $\tilde{L}^2_{Exc}(N, \overline{\boldsymbol{m}}_0) = \varnothing$.

only should be considered; however, we do not formally recall these languages since they can be easily derived by those generated by interpreted sequences using the projection operator $\Pi(i\sigma)$. Hence, the definition of exceeding language and distance can be immediately extended to such languages. We simply refer to them by applying the tilde symbol to the already defined languages and measures, *e.g.* $\tilde{L}^n_{Obs}$ is the observed language of length n containing sequences of transitions only. The example can now be illustrated.

Consider the observable IPN system $\langle N^{Obs}, \boldsymbol{m}_0 \rangle$ in Fig. 4. The set of observed interpreted sequences is: $I\Sigma = \{i\sigma_1, i\sigma_2, i\sigma_3\}$, where $i\sigma_1 = \boldsymbol{m}_0 t_1 \boldsymbol{m}_{11} t_4 \boldsymbol{m}_{10} t_5 \boldsymbol{m}_9 t_2 \boldsymbol{m}_3 t_8 \boldsymbol{m}_4 t_6 \boldsymbol{m}_1 t_9 \boldsymbol{m}_0$, $i\sigma_2 = \boldsymbol{m}_0 t_1 \boldsymbol{m}_{11} t_4 \boldsymbol{m}_{10} t_2 \boldsymbol{m}_5 t_5 \boldsymbol{m}_3 t_8 \boldsymbol{m}_4 t_6 \boldsymbol{m}_1 t_9 \boldsymbol{m}_0$, $i\sigma_3 = \boldsymbol{m}_0 t_1 \boldsymbol{m}_{11} t_3 \boldsymbol{m}_7 t_7 \boldsymbol{m}_8 t_4 \boldsymbol{m}_6 t_6 \boldsymbol{m}_2 t_5 \boldsymbol{m}_1 t_9 \boldsymbol{m}_0$; they are not cyclic.

Suppose that an IPN system $\langle N, \overline{\boldsymbol{m}}_0 \rangle$, differing from $\langle N^{Obs}, \boldsymbol{m}_0 \rangle$ by a set of unobservable places, must be found such that it ensures $\tilde{L}^2_{Exc}(N, \overline{\boldsymbol{m}}_0) = \varnothing$ and $\tilde{d}^2(N, \overline{\boldsymbol{m}}_0) = 0$.

The IPN system in Fig. 5. satisfies these constraints; however, it is able to generate words of length 3 (containing 2 transition firings) that were never observed. For example, the word $w = \boldsymbol{m}_{10} t_2 \boldsymbol{m}_5 t_8 \boldsymbol{m}_6$ does not belong to the known words, but can be produced by the model when $\sigma = t_1 t_4 t_2 t_8$ is fired. Indeed, once t_2 is fired, the firing of t_8 or t_5 is enabled. In other words, the model is not 3-complete.

When also observable markings are considered in addition to transition firings, then t_8 is only expected when the firing of t_2 produces $\boldsymbol{m}_3$, and t_5 is only expected when the firing of t_2 produces $\boldsymbol{m}_5$. Thus, applying to this example the method proposed in this work, a 3-complete IPN system $\langle N', \overline{\boldsymbol{m}}'_0 \rangle$ is found that better models the system and still satisfies $\tilde{L}^2_{Exc}(N', \overline{\boldsymbol{m}}'_0) = \varnothing$ and $\tilde{d}^2(N', \overline{\boldsymbol{m}}'_0) = 0$.

IV. A SOLUTION BASED ON ILP PROBLEMS SOLVING

In this section we present the methodology adopted to enforce r-completeness. Firstly a Moore machine E_r is constructed which models the dynamics that the new IPN system should exhibit. Then new unobservable places implementing the dynamics of E_r are computed by solving ILP problems and added to the given IPN system. These places act exclusively on the state enabling of a given transition t of the IPN model without affecting the logical condition function $\beta(t)$.

A. Modelling of the desired behaviour

In this subsection, a Moore machine E_r, depending on the design parameter r, is built such that it is r-complete. At this aim, some considerations are useful.

For the sake of simplicity, firstly suppose that a single interpreted sequence $i\sigma_1 = \boldsymbol{m}_0 t_1 \boldsymbol{m}_1 \dots t_{|i\sigma_1|-1} \boldsymbol{m}_{|i\sigma_1|-1}$ has been observed from the system ($I\Sigma = \{i\sigma_1\}$). The most trivial Moore Machine A representing the system is made of $|i\sigma_1|$ states, where q_0 is the initial state and a state q_i is associated to each marking $\boldsymbol{m}_i$ in the interpreted sequence, *i.e.* $\lambda(q_i) = l(\boldsymbol{m}_i), 0 \leq i \leq |i\sigma_1| - 1$. Such states are linked as a *chain*: $q_i \overset{t_{i+1}}{\rightarrow} q_{i+1}, 0 \leq i \leq |i\sigma_1| - 1$. Note that each state has a single output and input arc; in addition, the automaton is *acyclic*, *i.e.* no directed path starts from and returns to the same state q_i. This model guarantees by construction $L(A) = L_{Obs}$, and thus is maximally accurate.

Now consider the case of two interpreted sequences observed from the system ($I\Sigma = \{i\sigma_1, i\sigma_2\}$). They necessarily share a prefix $i\sigma$ of length 1 due to the common initial marking, *i.e.* $i\sigma_1 = i\sigma i\sigma'_1$ and $i\sigma_2 = i\sigma i\sigma'_2$; however, in the general case $|i\sigma| \geq 1$ holds. Assume that the sequence $i\sigma_1$ is firstly represented in the trivial automaton A. It can be noted that the states $q_i, 0 \leq i \leq |i\sigma| - 1$ already model the prefix of $i\sigma_2$; thus, it is sufficient to continue the chain from $q_{|i\sigma|-1}$ to

model $i\sigma'_2$. As a result, the state $q_{|i\sigma|-1}$ has two output arcs, one for $i\sigma'_1$ and one for $i\sigma'_2$. This trivial automaton is still acyclic and maximally accurate and can model two interpreted sequences.

Acyclic automata are not suitable for representing a cyclic observed interpreted sequence $i\sigma^o$. At this aim, we introduce *cycles* in the model. Let $I\Sigma = \{i\sigma^o\}$, the trivial automaton A is built as before, except for $q_{|i\sigma^o|-2}$ being linked to q_0 instead of $q_{|i\sigma^o|-1}$, *i.e.* $q_{|i\sigma^o|-2} \overset{t_{|i\sigma^o|-1}}{\rightarrow} q_0$. Such an automaton is cyclic and maximally accurate.

Note that using a trivial automaton the number of states grows with the length of the observed interpreted sequences; furthermore, these sequences are expected to be long because rich trajectories of the system must be acquired. Hence, it makes sense to compact the model by using cycles also for non cyclic interpreted sequences, instead of using chains only. As a result the model generates, in addition to the observed words, exceeding words also, since infinite paths exist due to cycles. As discussed, such exceeding behaviour is desirable if it is "controlled".

In conclusion, it is clear that a proper management of cycles allows the building of compact models that guarantee a certain degree of accuracy. The algorithm proposed in this subsection builds a cyclic, compact, non-maximally accurate model E_r that ensures r-completeness.

The basic idea for building E_r is to split it into two sides: the left side, that includes the initial state, is constructed as a trivial automaton; the right side, instead, contains cycles to compact the model. The typical structure of E_r is exemplified in Fig. 6, where $r = 5$ is chosen without loss of generality; obviously, it is supposed that r satisfies the discussed bounds, *i.e.* $l^{i\sigma^o}_{max} \leq \tilde{n} \leq r \leq l^{i\sigma}_{max}$. The two sides share the states along the boundary line ($l = r$).

In detail, in the left side:

- words of length $l \leq r$ are generated from the initial state, such that the constraint $L^r_{Exc}(E_r) = \varnothing$ is ensured;
- if a cycle exists, it only returns to the initial state and never passes through any state on the boundary line. Such cycles represent cyclic interpreted sequences.

Instead, in the right side:

- words of length $l \geq r$ are generated from the initial state;
- cycles are allowed *iff* 1) from any state only words can be generated such that $d^r(E_r) = 0$, 2) they do not return to any state at the left of the boundary line.

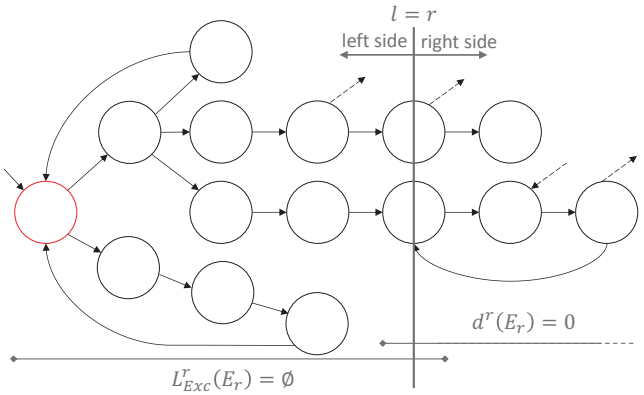

Fig. 6. An example of E_r where $r = 5$. The left side of E_r guarantees $L^r_{Exc}(E_r) = \varnothing$; the right side ensures $d^r(E_r) = 0$. Labels to states and arcs have been omitted for the sake of readability.

Finally note that, in both sides, *terminal states* can exist, *i.e.* states without output arcs.

To present the algorithm that constructs this automaton, some definitions are necessary. For the sake of brevity, we will denote by σ_k each sequence $\sigma_k = \Pi(i\sigma_k), i\sigma_k \in I\Sigma$; in addition, the length $|\sigma_k|$ is denoted by l_k. Finally, given a sequence σ, the first transition of the sequence is given by $\sigma(1)$ and the k-th, $k \leq |\sigma|$, is given by $\sigma(k)$; furthermore, $\sigma(m, n)$ denotes the subsequence of σ having all its elements from the m-th to the n-th; if $n < m$ the empty sequence is returned.

Definition 10. Given the set of observed interpreted sequences $I\Sigma$, we firstly denote by $\boldsymbol{m}_{k,s}$ ($t_{k,s}$) the s-th marking (transition firing) observed during the k-th acquisition, where $0 \leq s \leq l_k$ and $\boldsymbol{m}_{k,0} = \boldsymbol{m}_0, \forall k$. Then, we denote by $w^n(i\sigma_k, s)$ a *direct* word, defined as: $w^n(i\sigma_k, s) = \boldsymbol{m}_{k,s} t_{k,s+1} ... t_{k,s+n-1} \boldsymbol{m}_{k,s+n-1}$, where $s + n - 1 \leq l_k$. Each $w^n(i\sigma_k, s)$ is a *known* word by definition. □

The algorithm that builds E_r is based on a sliding window mechanism. This window has a dynamic size and slides on each observed word $l(i\sigma)$ ($i\sigma \in I\Sigma$); at any moment, it only contains a subword of $l(i\sigma)$. The automaton is constructed by adding at each step a state associated to the subword contained in the window; we denote by $\omega(q)$ the word w associated to a state q. In particular:

1) firstly, $l(i\sigma_1)$ is selected. The window is positioned at the beginning of the observed word ($s = 0$); the length of the window is $l = 1$ and contains the word $w_0 = w^1(i\sigma_1, 0) = \boldsymbol{m}_0$: the initial state q_0 is created and associated to w_0, *i.e.* $\omega(q_0) = w_0$, $\lambda(q_0) = last(w_0) = \boldsymbol{m}_0$. The window's length l is then incremented by a unit and the word $w_1 = w^2(i\sigma_1, 0)$ is encountered. A state q_1

is thus constructed, such that $\omega(q_1) = w_1$ and $\lambda(q_1) = last(w_1)$; the state q_0 is linked to q_1 by an arc labelled by $\sigma_1(1)$. This procedure repeats until $l = r - 2$;

2) $l = r - 1$ is set (maximum length of the window) and the word $w = w^{r-1}(i\sigma_1, 0)$ is encountered. The state q_{r-3} is linked to q_{r-2} through an arc labelled by $\sigma_1(r-2)$;
3) the window, now, slides by one step ($s = 1$) keeping the same length; the contained word is $w' = w^{r-1}(i\sigma_1, 1)$. This word is said to be a *successor* of the previously encountered word w. A new state q_{r-1} is created and associated to w'. The previous state is linked to q_{r-1} by an arc labelled by $\sigma_1(r-1)$. The window then slides again ($s = 2$) and a new state is created and properly linked. This process, whose details are given in the following, continues until: a) the window reaches the end of $i\sigma_1$, if $i\sigma_1$ is not cyclic; b) the window reaches the second-last marking of $i\sigma_1$, otherwise. As usual, a state q_f is created and associated to the word contained in the window. If $i\sigma_1$ is cyclic, q_f is directly linked to q_0 through an arc labelled by $\sigma_1(|\sigma_1|)$; in this way, the state resetting property of cyclic sequences is implemented in E_r;
4) when the processing for $l(i\sigma_1)$ is finished, the following observed word $l(i\sigma_2)$ (if it exists) is selected and the procedure is re-executed from the step 1); new states and arcs are thus added to E_r.

If, at any step, it happens that a word w is encountered again in the window, the state q_w, associated to w and created the first time that w was encountered in that step, is *re-used* and new input/output arcs added. State re-using only produces new output arcs when applied at step 1) and 2), while it can also produce cycles when applied at step 3); thus, in line with the discussed structure of E_r (Fig. 6.), the algorithm ensures that cycles can only return to states of the right side.

As a result of state re-using, a single state exists for each encountered word of length $l < r-1$, while at most two states associated to the same word of length $r - 1$ can exist, one created at step 2) and positioned at the left of the boundary line and one created at step 3) and belonging to the right side; in this case, the state created at step 2) is called *early* (E) state, while the one created at step 3) is called *late* (L) state. The remaining states are said to be *regular* (R) states.

In the following example, the illustrated algorithm is applied.

Example 11. Consider again the observable IPN system of Example 8 and suppose that $\tilde{n} = 5$ is given.

Applying the illustrated algorithm for $r = 5$, the MM in Fig. 7. is obtained: each state q contains the associated word $w = \omega(q)$ and the output λ in the form $\lambda(q) = l(\boldsymbol{m}_i) = \boldsymbol{m}_i$. All the states are regular (R). ■

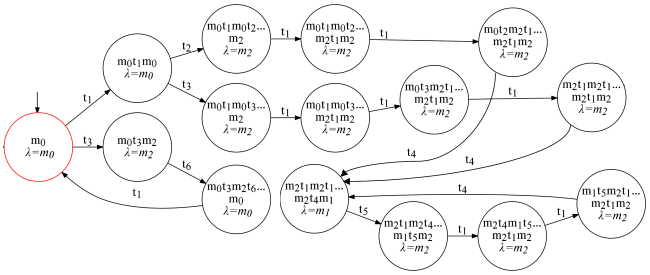

Fig. 7. E_r for the system and $r = 5$.

In the last part of this subsection, r-completeness of E_r is demonstrated and the algorithm is formalized.

Lemma 12. The automaton E_r ensures $d^{r-1}(E_r) = 0$.

Proof. The thesis holds because:

- taken any state $q \in Q$, the associated word $w = \omega(q)$ can be produced by E_r, since a path exists that starts $|w|-1$ states backward and ends in q and such that: 1) the j-th state along the path, $1 \leq j \leq |w|$, has as output the j-th marking that composes w; 2) the j-th transition between the j-th state and the $j+1$-th state, $1 \leq j \leq |w| - 1$, is the j-th transition that composes w.
- words due to cyclic sequences can be produced, thanks to the backward arcs directed to the initial state;
- E_r does not produce any other word other than the cited ones; indeed, if a word $w \in \mathcal{V}(L(E_r), r-1)$ existed such that $w \notin \mathcal{V}(L_{Obs}, r-1)$, then a path leading to a state associated to w or an incorrect arc to any state would exist; however, this is impossible by construction. □

The main property of E_r can now be demonstrated.

Proposition 13. The automaton E_r is r-complete.

Proof. The thesis holds *iff*:

- $d^r(E_r) = 0$; this is true thanks to Lemma 12 and considering that each state $q \in Q$ only admits transitions that generate observed successors after $\omega(q)$, thus only known words of length r are produced.
- $L^r_{Exc}(E_r) = \varnothing$; this is true since: 1) in the left side E_r is constructed as the trivial automaton; 2) no cycles return to the states created at step 1) and 2), which belong to the left side. □

In Algorithm 1 the function $createState(w, type)$, used also in Algorithm 2, creates a new state q such that $\omega(q) = w$ and whose type is contained in the variable $type$, which either is R,E or L.

Algorithm 1: construction of E_r

input : $I\Sigma, r$
output: $E_r = (Q, E, \delta, q_0, \Lambda, \lambda)$
Initialize:
$q_0 = createState(w^1(i\sigma_1, 0), \mathrm{R})$
$Q = \{q_0\}$, $E = \varnothing$
for $k = 1$ *to* $|I\Sigma|$ **do**
  $l = 2$, $q_{prec} = q_0$
  for $j = 2$ *to* $|i\sigma_k|$ **do**
    if $l > r - 1$ **then** $l = r - 1$;
    if $j < |i\sigma_k| \vee i\sigma_k$ *is not cyclic* **then**
      $q_{curr} = getState(Q, i\sigma_k, l, j - l, r)$
      $Q = Q \cup q_{curr}$
    else
      $q_{curr} = q_0$, since cyclic sequence lead the model to the initial state
    $e = \sigma_k(j - 1)$
    $\delta(q_{prec}, e) = q_{curr}$, *i.e.* the succession of two words is translated into an arc that links the two associated states
    $E = E \cup e$, $q_{prec} = q_{curr}$, $l = l + 1$
$\lambda(q) = last(\omega(q)) \, \forall q \in Q$, $\Lambda = \{\lambda(q), \forall q \in Q\}$.

The function $getState$ is also used and its pseudocode is presented in Algorithm 2. This function takes as input Q, the selected $i\sigma$, the length of the window l, the current sliding step s and the value of the design parameter r and returns the state associated to the encountered word. This state is firstly searched in Q using the function $find$, which yields, if exists, the already existing state in Q; if the state is not found in Q, then a new one is created.

B. Strategy for computing the unobservable places

The given IPN system $\langle N^{Obs}, \boldsymbol{m}_0 \rangle$ is in general not r-complete since some sequences could exist that can be produced by N^{Obs} from $\boldsymbol{m}_0$ but not by E_r from q_0. In the general case, each one of these sequences can be split in two parts: the first sub-sequence (that can be empty) is still feasible in E_r; the second one begins with the first infeasible transition firing, *i.e.* the first transition that is not expected in E_r. If the first transition firing of each infeasible (sub)path is blocked through an unobservable place, then only the first half will be firable and the exceeding language will be reduced. The addition of new

Algorithm 2: the $getState$ function

Function *q=getState(*$Q, i\sigma, l, s, r$*)***is**
  $w = w^l(i\sigma, s)$, $type = \mathrm{R}$, $q = \varnothing$
  if $l = r - 1$ **then**
    if $s = 0 \wedge \exists \, w^l(i\sigma_i, s') = w, i\sigma_i \in I\Sigma, s' > 0$ **then** $type = E$;
    if $s > 0 \wedge \exists \, w^l(i\sigma_i, s') = w, i\sigma_i \in I\Sigma, s' = 0$ **then** $type = L$;
  $q = find(Q, w, type)$
  if $q = \varnothing$ **then**
    $q = createState(w, type)$

unobservable places to N^{Obs}, leads to a new IPN system $\langle N', \overline{\boldsymbol{m}}'_0 \rangle$. Constraints that involve the disabling of a single transition firing by a single unobservable place are called *local* (I). Obviously, the added unobservable places must ensure the firability of all the observed sequences in $\Sigma \equiv \{\sigma_1, ..., \sigma_{|I\Sigma|}, i\sigma_k \in I\Sigma\}$ (II) and reset their status at the end of each cyclic sequence (III), otherwise the condition $L^n(N', \overline{\boldsymbol{m}}'_0) \supseteq L^n_{Obs}$, for any finite n, could not be satisfied anymore. The constraints II and III are called *global* constraints, since they are shared among all the unobservable places.

The three constraints just illustrated, however, are in general not sufficient to reach the objective: indeed, additional *global* constraints are needed to implement in the new IPN system the same concept of state re-using of E_r (IV).

The general idea given, in the following a formalization is illustrated. At this aim, the following notation is introduced: it is denoted by $\delta_r(q, t)$ the transition function of E_r, and similarly by Q_r the state space of E_r. Furthermore, it is denoted by $\mathcal{A}_r(q)$ the function that yields the set of admissible transition events in the state $q \in Q_r$.

C. Global algebraic constraints

In this section the set of constraints that each new unobservable place must ensure is devised; to resume:

1) each observed sequence in Σ must be enabled at the initial marking;
2) the state must be reset at the end of each cyclic sequence;
3) if two productions s_1 and s_2 of E_r, starting from q_0 and not passing through it, lead to the same state, then the associated sequences of transition firings σ_1 and σ_2, fired from the initial marking, must lead to the same marking.

In order to formalize these constraints, the following definitions come in help.

Definition 14. The function $Z^r(q) : Q_r \rightarrow \{\sigma_i\}$ is defined as:
$Z^r(q) = \{\sigma_k(1, s+|w|-1) \mid w^{|w|}(i\sigma_k, s) = w, w = \omega(q)$ $i\sigma_k \in I\Sigma, s = 0$ if $|w| < r-1$, else $s \geq 0$ if q is R or E, else $s > 0\}$. □

In words, $Z^r(q)$ returns all the sub-sequences of transition firings obtained from the observed firings and corresponding to the productions in E_r that start from q_0 and end in q. Considered the Example 11 and the state q such that $\omega(q) = \boldsymbol{m}_2 t_1 \boldsymbol{m}_2 t_1 \boldsymbol{m}_2 t_4 \boldsymbol{m}_1$, then $Z^r(q) = \{\sigma', \sigma'', \sigma'''\}$, where $\sigma' = \sigma_1, \sigma'' = t_1 t_2 t_1 t_1 t_4, \sigma''' = \sigma_2$; note that σ'' is a sub-sequence of σ_2.

Definition 15. Consider the automaton E_r and an unobservable place p identified by a row vector $\boldsymbol{c}^U$ of the incidence matrix. The place is said *r-consistent iff* $\forall q \in Q_r$ it holds $\boldsymbol{c}^U \boldsymbol{\sigma}_i = \boldsymbol{c}^U \boldsymbol{\sigma}_j$ where $\sigma_i, \sigma_j \in Z^r(q), i \neq j$.□

The previously enumerated constraints can now be mathematically expressed.

Proposition 16. Given a r, consider the following set of *global constraints*, denoted by $\mathcal{C}(\Sigma, r)$

$$\begin{cases} m_0^U + \boldsymbol{c}^U \tilde{\boldsymbol{\sigma}}_1 \geq \mathbf{pre}^U \tilde{\boldsymbol{\sigma}}_2,\ \tilde{\sigma}_1 = \sigma_k(1, l-1), \\ \quad \tilde{\sigma}_2 = \sigma_k(l), \sigma_k \in \Sigma, \forall 1 \leq l \leq l_k; & (16.1) \\ \boldsymbol{c}^U \boldsymbol{\sigma} = 0, \forall \sigma^o \in \Sigma; & (16.2) \\ \boldsymbol{c}^U \overline{\boldsymbol{\sigma}} = \boldsymbol{c}^U \boldsymbol{\sigma}_i,\ \forall \sigma_i \in Z^r(q), \sigma_i \neq \overline{\sigma}, \forall q \in Q_r, \\ \quad \text{and } \overline{\sigma} \text{is arbitrary chosen from } Z^r(q); & (16.3) \end{cases}$$

where $\mathbf{pre}^U, \mathbf{post}^U, m_0^U$ are the unknowns, and precisely the pre and post incidence row vectors and the initial marking of an unobservable place to be computed, and $\mathbf{c}^U = \mathbf{post}^U - \mathbf{pre}^U$. The set of constraints ensures that the unobservable place $\langle(\mathbf{pre}^U, \mathbf{post}^U), m_0^U\rangle$ satisfies the global constraint 1) through (16.1), 2) through (16.2) and 3) through (16.3).

Proof. The proof comes straightforwardly, noticing that:

- Inequalities (16.1) ensure that legal firings are not affected, through the condition of transition abilitation in a state.
- Equalities (16.2) guarantee state resetting on cyclic sequences, by setting the initial marking to the reached one through the state equation.
- Equalities (16.3) ensure that the marking reached from m_0^U by firing any sequence in $Z^r(q)$ always leads to the same marking $m_q^U, \forall q \in Q_r$ (r-consistency). At this aim, an arbitrary sequence $\overline{\sigma}$ is chosen from $Z^r(q)$ which leads to a certain marking m^U; then, it is imposed that the same marking m^U is reached when all the other sequences in $Z^r(q)$ are fired. □

In the following proposition, an important property on the set of constraints just illustrated is given.

Proposition 17. The set $\mathcal{C}(\Sigma, r+1)$ is equally or less restrictive than $\mathcal{C}(\Sigma, r)$.

Proof. In order to prove the thesis, only the equalities (16.3) must be considered, since (16.1) and (16.2) do not depend on r.

First, note that the boundary line ($l = r+1$) in E_{r+1} is positioned just at the right of the boundary line ($l = r$) in E_r; thus, the states on the boundary line ($l = r$) of E_r are at the left of the boundary line ($l = r+1$) of E_{r+1}. As a consequence, possible cycles returning to those states are not present in E_{r+1}, being not permitted, *i.e.* some of the equalities (16.3) could not be contained in $\mathcal{C}(\Sigma, r+1)$.

Secondly, each state q_r of E_r at the right of the boundary line ($l = r$) becomes a state q_{r+1}^i of E_{r+1} belonging to the right side and such that $\omega(q_{r+1}^i) = w_p^i \omega(q_r)$, where $|w_p^i| = 2$. Since $\omega(q_{r+1}^i)$ is encountered no more frequently with respect to $\omega(q_r)$ in the sliding window, no more times q_{r+1}^i is re-used, thus no more equalities (16.3) are contained in $\mathcal{C}(\Sigma, r+1)$. □

D. Local algebraic constraints

In the following proposition, *local* constraints related to firing disablings are treated.

Proposition 18. Consider the IPN system $\langle N^{Obs}, \boldsymbol{m}_0\rangle$, a state $q \in Q_r$ and a sequence $\overline{\sigma} \in Z^r(q)$. The following set of constraints

$$\begin{cases} m_0^U + (\mathbf{post}^U - \mathbf{pre}^U)\overline{\boldsymbol{\sigma}} < \mathbf{pre}^U \boldsymbol{e}_{t_u}; & (18.1) \\ \mathbf{pre}^U \boldsymbol{e}_{t_u} > 0; & (18.2) \end{cases}$$

where $\mathbf{pre}^U, \mathbf{post}^U, m_0^U$ are the unknowns, disables the firing of t_u after $\overline{\sigma}$.

Proof. The proof comes straightforwardly since by inequality (18.1) the undesired firing is disabled while by (18.2) the presence of an output arc directed to t_u is imposed. □

E. The core algorithm

In this section, the core algorithm that produces the desired IPN system $\langle N', \overline{\boldsymbol{m}}_0'\rangle$ is presented. The following propositions illustrate some properties of the proposed algorithm.

Proposition 19. Algorithm 3 produces a r-complete IPN system only if the places of the input IPN system are r-consistent.

Proof. For each state $q \in Q_r$, undesired firings are searched among the state enabled transitions in $\mathcal{E}(N', \overline{\boldsymbol{m}}_0', \overline{\sigma}), \overline{\sigma} \in Z^r(q)$; they are then disabled through

Algorithm 3: r-completeness enforcement

input : $E_r, I\Sigma_{Obs}, r, \langle N^{Obs}, \boldsymbol{m}_0\rangle$.
output: A new IPN system $\langle N', \overline{\boldsymbol{m}}'_0\rangle$ that ensures r-completeness if all ILPs have a solution.

Initialize: $\langle N', \overline{\boldsymbol{m}}'_0\rangle = \langle N^{Obs}, \boldsymbol{m}_0\rangle$
$f_{min} = (\mathbf{pre}^U + \mathbf{post}^U)\cdot \mathbf{1} + m_0^U$
foreach $q \in Q_r$ **do**
  Select an arbitrary $\overline{\sigma} \in Z^r(q)$
  $T_u^{treated} = \varnothing$
  while $\{\{\mathcal{E}(N', \overline{\boldsymbol{m}}'_0, \overline{\sigma}) \setminus \mathcal{A}_r(q)\} \setminus T_u^{treated}\} \neq \varnothing$ **do**
    Select an arbitrary $t_u \in \{\mathcal{E}(N', \overline{\boldsymbol{m}}'_0, \overline{\sigma}) \setminus \mathcal{A}_r(q)\} \setminus T_u^{treated}$
    $S =$

$$\begin{cases} m_0^U + \boldsymbol{c}^U\tilde{\boldsymbol{\sigma}}_1 \geq \mathbf{pre}^U\tilde{\boldsymbol{\sigma}}_2,\ \tilde{\sigma}_1 = \sigma_k(1, l-1), \\ \quad \tilde{\sigma}_2 = \sigma_k(l), \sigma_k \in \Sigma, \forall 1 \leq l \leq l_k; \\ \boldsymbol{c}^U\boldsymbol{\sigma} = 0, \forall \sigma^o \in \Sigma; \\ \boldsymbol{c}^U\overline{\boldsymbol{\sigma}} = \boldsymbol{c}^U\boldsymbol{\sigma}_i,\ \forall \sigma_i \in Z^r(q), \forall q \in Q_r; \\ m_0^U + (\mathbf{post}^U - \mathbf{pre}^U)\overline{\boldsymbol{\sigma}} < \mathbf{pre}^U\,\boldsymbol{e}_{tu}; \\ \mathbf{pre}^U\,\boldsymbol{e}_{tu} > 0; \end{cases}$$

    if $\nexists\, p^U \in P'$ *s.t. S is satisfied* **then**
      a new unobservable place p^U is constructed:
      $[\mathbf{pre}^U, \mathbf{post}^U, m_0^U, solved] = solveILP(S, f_{min})$
      if *solved* **then**

$$P' = P' \cup p^U,\ \overline{\mathbf{Pre}}' = \begin{bmatrix}\overline{\mathbf{Pre}}' \\ \mathbf{pre}^U\end{bmatrix},\quad \overline{\mathbf{Post}}' = \begin{bmatrix}\overline{\mathbf{Post}}' \\ \mathbf{post}^U\end{bmatrix},\ \overline{\boldsymbol{m}}'_0 = \begin{bmatrix}\overline{\boldsymbol{m}}'_0 \\ m_0^U\end{bmatrix}$$

    $T_u^{treated} = T_u^{treated} \cup t_u$

ILPs at the marking $\overline{\boldsymbol{m}}'$, where $\overline{\boldsymbol{m}}'_0 \xrightarrow{\overline{\sigma}} \overline{\boldsymbol{m}}'$. Successively, the next state $q' \in Q_r$ is selected and the procedure repeated.

Since the disabling of undesired firings is only enforced at the marking $\overline{\boldsymbol{m}}'$, it is necessary that the marking reached when any other sequence $\sigma_i \neq \overline{\sigma} \in Z^r(q)$ is fired from $\overline{\boldsymbol{m}}'_0$ is equal to $\overline{\boldsymbol{m}}'$; in other words, all the places of $\langle N', \overline{\boldsymbol{m}}'_0\rangle$ must be r-consistent. Noted that the unobservable places added by the algorithm are r-consistent, only the places of the IPN system given as input must be r-consistent. □

It is immediate to recognize that the observable IPN system $\langle N^{Obs}, \boldsymbol{m}_0\rangle$ has k-consistent places whatever k is chosen; indeed, for any sequence $\sigma_i \in Z^k(q), \forall q \in Q_r$, the IPN always reaches the same (observable) marking, whose symbol is $last(\omega(q))$.

Proposition 20. Given the IPN system $\langle N^{Obs}, \boldsymbol{m}_0\rangle$, if all the ILP problems of Algorithm 3 are solved then the produced IPN system $\langle N', \overline{\boldsymbol{m}}'_0\rangle$ is r-complete. In the general case, it holds $L^n_{Obs} \subseteq L^n(N', \overline{\boldsymbol{m}}'_0) \subseteq L^n(N^{Obs}, \boldsymbol{m}_0)$ for any finite n.

Proof. Suppose that a complete solution has been found, *i.e.* the unobservable places implement for the IPN system $\langle N', \overline{\boldsymbol{m}}_0\rangle$ the same dynamics of E_r. Then, the new model is r-complete.

On the other hand, if some ILP fails, the resulting IPN system is in general not r-complete; in addition, it could be unbounded, which can happen when the given IPN system $\langle N^{Obs}, \boldsymbol{m}_0\rangle$ contains isolated and source transitions. However, the added places (if any) still guarantee the removing of certain unexpected words and thus a reduction of the exceeding language. □

Unsolvability of some ILPs is due to the nature of a Petri net place: it is mostly like a counter since it only maintains memory of its current status. Being it insensible to the order in which increments and decrements occur, it is not able to implement any constraint that is order-sensible.

The main property of the method is expressed by the following proposition.

Proposition 21. Let $\langle N'_{r+1}, \overline{\boldsymbol{m}}'_0\rangle$ and $\langle N'_r, \overline{\boldsymbol{m}}_0\rangle$ be the solutions given by Algorithm 3 when $r+1$ and r are respectively chosen as parameters. Then, it holds $L^n(N'_{r+1}, \overline{\boldsymbol{m}}'_0) \subseteq L^n(N'_r, \overline{\boldsymbol{m}}_0)$, for any finite n, independently of the number of solved ILPs.

Proof. To prove the thesis, only equalities (16.3) (the ones due to state re-using) and inequalities (18.1), (18.2) (the ones due to firing disablings) must be considered, since the remaining are common in the two problems.

As illustrated for the proof of Proposition 17, each state q^i_{r+1} of E_{r+1} associated to a word of length r has no more output arcs of the corresponding state q_r of E_r such that $\omega(q^i_{r+1}) = w^i_p\omega(q_r)$. As a result, no more words can be produced by the automaton E_{r+1} with respect to the automaton E_r, *i.e.* $L^n(E_r) \supseteq L^n(E_{r+1}), \forall n$ holds.

On the basis of these considerations, it is clear that the $r+1$ problem has to guarantee at least the same firing disablings of the r problem or even more.

Since global constraints are no more restrictive in the $r+1$ problem (due to Proposition 17), no less transition disablings are implemented through ILPs with respect to the r problem; in addition, also new local constraints are enforced, due to $L^n(E_r) \supseteq L^n(E_{r+1}), \forall n$. The

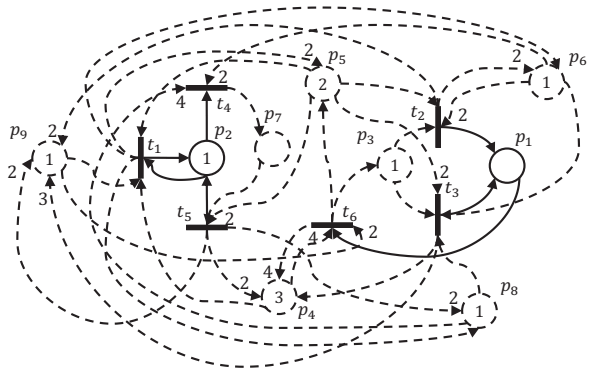

Fig. 8. The IPN system $\langle N', \overline{\boldsymbol{m}}_0 \rangle$ computed by Algorithm 3 for the system of example 11 and $r = 5$. Arc weight is shown close to the arc arrow.

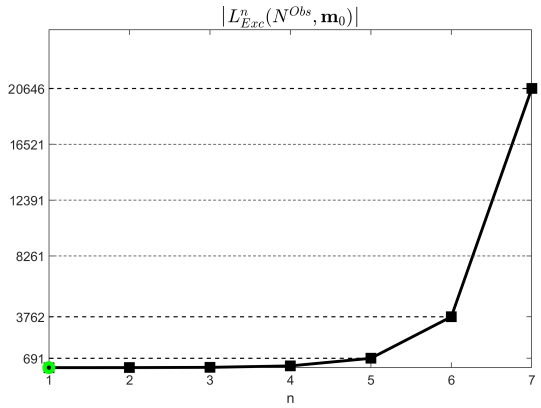

Fig. 9. Exceeding language produced by the IPN $\langle N^{Obs}, \boldsymbol{m}_0 \rangle$. A green point means empty exceeding language.

resulting net, thus, exhibits a non-greater exceeding behaviour. □

Note that Algorithm 3 produces unobservable places with minimum arcs weight and minimal initial tokens, as the function $f_{min} = (\mathbf{pre}^U + \mathbf{post}^U) \cdot \mathbf{1} + m_0^U$ imposes, where $\mathbf{1}$ is a column vector having all elements equal to 1. Notice that Proposition 20 still holds if a trade-off between arc weights and the number of initial tokens is imposed, *i.e.* if the three terms $(\mathbf{pre}^U, \mathbf{post}^U, m_0^U)$ are weighted. In addition, the algorithm still works if the IPN system $\langle N, \overline{\boldsymbol{m}}_0 \rangle$ given as input is the result of Algorithm 3 applied for $k < r$.

The following example shows the method applied.

Example 22. Consider the system presented in example 11. The method is applied for $r = 5$. Recall that the target behaviour is represented by the automaton E_r in Fig. 7.

Seven unobservable places are produced, as shown in Fig. 8. Each unobservable place is constructed to implement a specific linear system. In particular, p_7 is constructed to disable t_5 just after $w = \boldsymbol{m}_0$. However, p_7 already disables t_5 after $\boldsymbol{m}_2 t_1 \boldsymbol{m}_2 t_4 \boldsymbol{m}_1 t_5 \boldsymbol{m}_2$, as checked at line 8 in Algorithm 3. In the very same way, p_3 is originally computed for disabling t_2 after $w = \boldsymbol{m}_0 t_1 \boldsymbol{m}_0 t_2 \boldsymbol{m}_2$, but also disables t_3 after $\boldsymbol{m}_0 t_1 \boldsymbol{m}_0 t_3 \boldsymbol{m}_2$.

The solution given by the algorithm is r-complete (all the ILPs are solved). The distance (exceeding language) is 0 (empty) for lengths $n \leq 5$. In addition, the distance linearly grows for $n \geq 6$; for example, the word $w^e = \boldsymbol{m}_0 t_1 \boldsymbol{m}_0 t_2 \boldsymbol{m}_2 t_1 \boldsymbol{m}_2 t_1 \boldsymbol{m}_2 t_4 \boldsymbol{m}_1 t_5 \boldsymbol{m}_2$ was never observed but can be produced. The method, thus, greatly reduces the original exceeding language of the observable IPN system, which is depicted in Fig. 9. ■

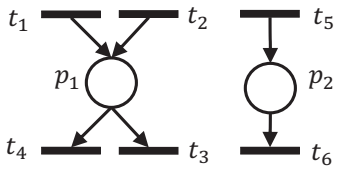

Fig. 10. An observable IPN system $\langle N^{Obs}, \boldsymbol{m}_0 \rangle$.

Example 23. This example compares the method proposed in this work with the approach illustrated in [24]. Such pattern-based approach is chosen for the comparison because, on the one hand, it generalizes the patterns considered in [16] and, on the other hand, it is not limited to t-invariants as in [26]. Thus, at the best of our knowledge, it is the best preforming method presented in the literature.

Consider the IPN shown in Fig. 10 and suppose $I\Sigma = \{i\sigma\}$, where $i\sigma = \boldsymbol{m}_0 t_1 \boldsymbol{m}_1 t_3 \boldsymbol{m}_0 t_2 \boldsymbol{m}_1 t_4 \boldsymbol{m}_0 t_5 \boldsymbol{m}_2 t_6 \boldsymbol{m}_0 t_5 \boldsymbol{m}_2 t_1 \boldsymbol{m}_3 t_3 \boldsymbol{m}_2 t_2 \boldsymbol{m}_3 t_4 \boldsymbol{m}_2 t_6 \boldsymbol{m}_0$ and $\boldsymbol{m}_0 = [0\,0]^T, \boldsymbol{m}_1 = [1\,0]^T, \boldsymbol{m}_2 = [0\,1]^T, \boldsymbol{m}_3 = [1\,1]^T$. Let $\langle N', \boldsymbol{m}'_0 \rangle$ be the IPN computed using the approach proposed in [24]; this method can be tuned by the free parameter m in such a way that higher is m more accurate is the computed model. For the comparison, the maximum value $m = |T| = 6$ has been chosen. Similarly, let $\langle N'', \boldsymbol{m}''_0 \rangle$ be the IPN computed with the method proposed in this paper for $r = 3$.

Fig. 11 and 12 compare the two IPNs respectively on the exceeding language and on the distance measure. As depicted, the IPN computed by the proposed method is far more accurate.

In addition, if $r = 4$ is chosen, the obtained model exhibits empty exceeding language and distance for each length n, *i.e.* it is maximally accurate.

The method proposed in [24] fails to deliver a more accurate model because:

- it is based on sequences of transition firings instead

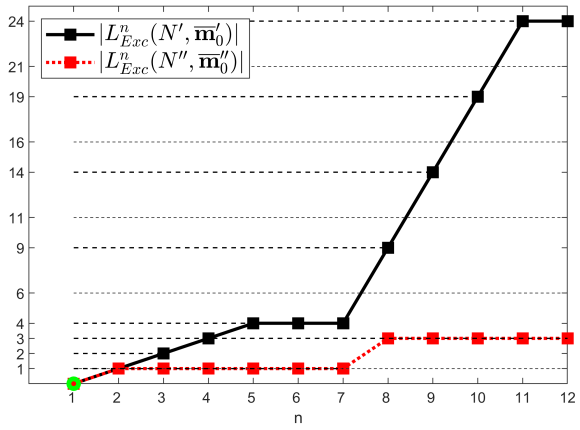

Fig. 11. Comparison of the exceeding language generated by $\langle N', \boldsymbol{m}'_0 \rangle$ and $\langle N'', \boldsymbol{m}''_0 \rangle$. A green point means empty exceeding language.

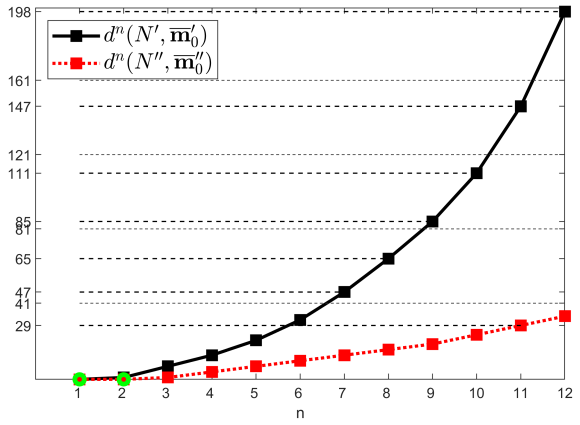

Fig. 12. Comparison of the distance between the observed language and the generated one by $\langle N', \boldsymbol{m}'_0 \rangle$ and $\langle N'', \boldsymbol{m}''_0 \rangle$. A green point means distance equal to 0.

of interpreted sequences, which account for observable markings also;
- it searches in the firing sequence for patterns based on rules and, thus, is limited to a number of cases.

In this specific example, no pattern is found relating the firing of t_5 with other transition firings, thus more exceeding words are generated. ■

F. Evaluation of the computational complexity

The computational complexity required by the approach presented in this paper depends on two main tasks: 1) building of E_r; 2) execution of the optimization technique based on ILPs.

As for the first task, let $l^{i\sigma}_{max}$ the longest observed interpreted sequence, then:

- The spatial complexity is $O(|D|\, l^{i\sigma}_{max})$; indeed, in the worst case the automaton E_r contains $|D|\, l^{i\sigma}_{max}$ states, independently on r. However, in real-case scenarios the required spatial complexity is far from the worst case, since state re-using is expected to occur frequently, especially for small values of r;
- The temporal complexity is $O(|D|\, l^{i\sigma}_{max} log(|D|\, l^{i\sigma}_{max}))$. To illustrate this, first note that data structures minimizing both *insertion* and *search* time complexity are $O(log(n))$, where n is the number of elements in the structure; an example of these structures are Red-Black trees, which take a space complexity of $O(n)$. In the computation of E_r, every time a word w is encountered in the window, a state q is *searched* such that $\omega(q) = w$; if it does not exist, then a new state is created and *inserted*. Considered that: 1) at each step, a search operation (and possibly an insertion) is performed; 2) the maximum number of steps is $|D|\, l^{i\sigma}_{max}$, the thesis holds.

As for the second task, as many ILPs are created for each state as the number of unexpected transition firings the net enables. This number is not known a-priori, since it depends on the observations, the accuracy of the given IPN and on r; in general, given the observations and the IPN, when r increases, the number of ILPs can increase (see Proposition 21). However, in the worst case, $|T|-1$ ILPs need to be solved for each non terminal state of E_r and $|T|$ for each terminal state. In detail, for each ILP:

- The number of unknowns is $1+2|T|$, due to the initial marking m^U_0 and the row vectors $\mathbf{pre}^U, \mathbf{post}^U$. Each unknown is an integer varying between 0 and $maxVal$; $maxVal$ is an upper bound that the user can set to bound the search space;
- The number of total constraints is $\sum_k |i\sigma_k| - 1 + |I\Sigma^o| + k l^{i\sigma}_{max}$, respectively due to (16.1), (16.2) and (16.3), where k is a not known a-priori proportional constant depending on r and the observations. In addition, the ILP also contains $1+1$ inequalities, respectively due to (18.1) and (18.2).

The complexity of each ILP is known to be NP-complete and exponentially grows with $|T|$. In principle, this could be a limiting factor for the identification of complex systems; however several measures can be taken to dominate the space and time resources required by the method, such as:

- Using *off-the-shelf* optimized software.
- Identifying the individual subsystems of the system, and then merging the computed models. Methods in the literature have been proposed for automated partitioning of a system for identification purposes [25].
- Decreasing the value of the design parameter r.

V. Concluding remarks

An approach for the identification of the unobservable behaviour of a Petri Net model from long event sequences has been presented. The proposed approach exploits the mathematical representation of PNs and improves previous approaches by evaluating the accuracy of the identified model with respect to sequences of transitions (inputs) and markings (outputs) and not only transitions. At this aim, ILPs are employed as the core element of an optimization-based procedure; this formulation is particularly convenient since off-the-shelf optimization tools can be employed to solve it. The resulting model is a general net, while previous solutions only return 1-bounded nets. It is important to notice that the net produced is not minimal in the number of unobservable places. In fact, the result strongly depends on the order in which linear systems are solved. Furthermore, the method allocates an unobservable place for solving a single transition disabling but, in the general case, a place could implement multiple transition disablings; this possibility has not been taken in consideration and will be investigated in future research.

The proposed approach works on logical models, while the explicit consideration of time is becoming crucial for the specification and verification of systems such as transportation systems [4], real-time systems, as well as the study of problems such as state estimation, and fault diagnosis [3]. First results to the identification of timed net models have been proposed in [6], [5] to identify a Time Petri Net from sequences of timed transition firings; future research will be devoted also on the identification of timed IPN models.

References

[1] Xpress-Optimizer - Reference manual, release 31.01. FICO^{TM} Xpress Optimization Suite, April 2017.

[2] L. Bai, N. Wu, Z. Li, and M. Zhou. Optimal one-wafer cyclic scheduling and buffer space configuration for single-arm multicluster tools with linear topology. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 46(10):1456–1467, Oct 2016.

[3] F. Basile, M. P. Cabasino, and C. Seatzu. Diagnosability analysis of labeled time Petri net systems. *IEEE Trans. Autom. Control*, 62(3):1384–1396, March 2017.

[4] F. Basile, C. Carbone, P. Chiacchio, R.K. Boel, and C.C. Avram. A hybrid model for urban traffic control. *2004 IEEE International Conference on Systems, Man and Cybernetics (SMC'04)*, 2:1795–1800, 2004.

[5] F. Basile, P. Chiacchio, and J. Coppola. A Novel Model Repair Approach of Timed Discrete-Event Systems With Anomalies. *IEEE Transactions on Automation Science and Engineering*, 13(4):1541–1556, October 2016.

[6] F. Basile, P. Chiacchio, and J. Coppola. Identification of time petri net models. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(9):2586–2600, Sep. 2017.

[7] F. Basile, P. Chiacchio, and D. Del Grosso. A two-stage modelling architecture for distributed control of real-time industrial systems: Application of UML and Petri net. *Computer Standards & Interfaces*, 31(3):528–538, March 2009.

[8] F. Basile, P. Chiacchio, A. Giua, and C. Seatzu. Deadlock recovery of Petri net models controlled using observers. *8th International Conference on Emerging Technologies and Factory Automation, (ETFA'01), Antibes, France*, 2:441–449, Oct 2001.

[9] F. Basile, R. Cordone, and L. Piroddi. A branch and bound approach for the design of decentralized supervisors in Petri net models. *Automatica*, 52:322–333, 2015.

[10] Joos C. A. M. Buijs, Boudewijn F. van Dongen, and Wil M. P. van der Aalst. On the role of fitness, precision, generalization and simplicity in process discovery. In Robert Meersman, Herve Panetto, Tharam Dillon, Stefanie Rinderle-Ma, Peter Dadam, Xiaofang Zhou, Siani Pearson, Alois Ferscha, Sonia Bergamaschi, and Isabel F. Cruz, editors, *On the Move to Meaningful Internet Systems: OTM 2012*, pages 305–322, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[11] M.P. Cabasino, P. Darondeau, M.P. Fanti, and C. Seatzu. Model identification and synthesis of discrete-event systems. In Mengchu Zhou, Han-Xiong Li, and Margot Weijnen, editors, *Contemporary Issues in Systems Science and Engineering*, IEEE/Wiley Press Book Series, pages 343–366. John Wiley & Sons, Inc., 2015.

[12] M.P. Cabasino, A. Giua, and C. Seatzu. Identification of Petri nets from knowledge of their language. *Discrete Event Dynamic Systems*, 17:447–474, December 2007.

[13] J. Cortadella, M. Kishinevsky, L. Lavagno, and A. Yakovlev. Deriving Petri nets from finite transition systems. *IEEE Trans. on Computers*, 47(8):859–852, August 1998.

[14] P. Darondeau. Region Based Synthesis of P/T-Nets and Its Potential Applications. In *21st International Conference on Application and Theory of Petri Nets 2000 (ICATPN 2000) Aarhus, Denmark*, volume 1825 of *Lecture Notes in Computer Science*, pages 16–23. Springer, June 2000.

[15] M Dotoli, M.P. Fanti, and A.M. Mangini. Real time identification of discrete event systems using Petri nets. *Automatica*, 44(5):1209 – 1219, 2008.

[16] A. P. Estrada-Vargas, E. Lopez-Mellado, and J. Lesage. A black-box identification method for automated discrete-event systems. *IEEE Transactions on Automation Science and Engineering*, 14(3):1321–1336, July 2017.

[17] A.P. Estrada-Vargas, E. Lopez-Mellado, and J. Lesage. A Comparative Analysis of Recent Identification Approaches for Discrete-Event Systems. *Mathematical Problems in Engineering*, 2010.

[18] A. Ghaffari, N. Rezg, and Xiaolan Xie. Design of a live and maximally permissive petri net controller using the theory of regions. *IEEE Transactions on Robotics and Automation*, 19(1):137–141, Feb 2003.

[19] K. Hiraishi. Construction of a class of safe Petri nets by presenting firing sequences. In Jensen, K., editor, *Lecture Notes in Computer Science; 13th International Conference on Application and Theory of Petri Nets 1992, Sheffield, UK*, volume 616, pages 244–262. Springer-Verlag, June 1992.

[20] Stephane Klein. *Identification of Discrete Event Systems for fault detection purposes.* PhD thesis, ENS Cachan, 2005.

[21] T. Murata. Petri nets: Properties, analysis and applications. *Proc. of the IEEE*, 77(4):541–580, April 1989.

[22] Y. Qiao, N. Wu, F. Yang, M. Zhou, and Q. Zhu. Wafer sojourn time fluctuation analysis of time-constrained dual-arm cluster tools with wafer revisiting and activity time variation. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 48(4):622–636, April 2018.

[23] Matthias Roth, Jean-Jacques Lesage, and Lothar Litz. The concept of residuals for fault localization in discrete event systems. *Control Engineering Practice*, 19(9):978 – 988, 2011.
[24] J. Saives, G. Faraut, and J. Lesage. Identification of discrete event systems unobservable behaviour by petri nets using language projections. *2015 European Control Conference (ECC'15)*, pages 464–471, July 2015.
[25] Jeremie Saives, Gregory Faraut, and Jean-Jacques Lesage. Automated partitioning of concurrent discrete-event systems for distributed behavioral identification. *IEEE Transactions on Automation Science and Engineering*, PP:1–10, 07 2017.
[26] T. Tapia-Flores, E. Lopez-Mellado, A. P. Estrada-Vargas, and J. Lesage. Discovering petri net models of discrete-event processes by computing T-invariants. *IEEE Transactions on Automation Science and Engineering*, 15(3):992–1003, July 2018.
[27] N. Wu and M. Zhou. Schedulability analysis and optimal scheduling of dual-arm cluster tools with residency time constraint and activity time variation. *IEEE Transactions on Automation Science and Engineering*, 9(1):203–209, Jan 2012.
[28] Naiqi Wu, MengChu Zhu, LiPing Bai, and Zhiwu Li. Short-term scheduling of crude oil operations in refinery with high-fusion-point oil and two transportation pipelines. *Enterprise Information Systems*, 10:1–30, 09 2014.
[29] G. Zhu, Z. Li, and N. Wu. Model-based fault identification of discrete event systems using partially observed petri nets. *Automatica*, 96:201 – 212, 2018.

Francesco Basile (SM'11) received the Laurea degree cum laude in electronic engineering and the Ph.D. degree in electronic and computer engineering from the University of Naples, Naples, in 1995 and 1999, respectively. In 1999, he was a Visiting Researcher with the Departamento de Ingenieria Informatica y Systemas, University of Zaragoza, Zaragoza, Spain, for six months. He is currently an Associate Professor of Automatic Control with the Dipartimento di Ingegneria dell'informazione ed elettrica e matematica applicata, Università di Salerno, Fisciano, Italy. He has published over 100 papers on international journals and conferences. His current research interests include modeling and control of discrete event systems, automated manufacturing, and robotics. Prof. Basile has been Associate Editor of the International Journal of Robotics and Automation, IEEE Transactions on Control Systems Technology and IEEE Transactions on Automation Science and Engineering. He has been member of IEEE Control System Society Conference Editorial Board. He is Associate Editor of IEEE Control Systems Letters. He has beeen General Chair of 14th International Workshop on Discrete Event Systems (WODES 2018).

Gregory Faraut received the B.S. degree in Electrical Engineering in 2004 and a M.S. degree in Computer Science in 2006 from the University of Nice-Sophia Antipolis, and the Ph.D. degree in Automatic Control from INSA Lyon, Ampere Lab. in 2010. Since 2011, he is Associate Professor of Automatic Control at LURPA, ENS Paris-Saclay, France. His research interests concern the field of formal methods and models of Discrete Event Systens (DES) with applications to manufacturing systems and energy production systems. He is a member of IEEE and was the founding co-chair of the Technical Committee on Automation in Health Care Management of the IEEE Robotics & Automation Society and is AE of this journal. The most recent research interests are the detection and evaluation of behaviour's deviations by machine learning approaches, and improvement of identification technic by Digital Twin for manufacturing and transport systems.

Luigi Ferrara received the the M.Sc. (B.Sc) degree in Computer Engineering cum laude from the University of Salerno, Fisciano, Italy, in 2018 (2015). Currently, he is a PhD student at the University of Salerno. His research interests include identification of industrial automation systems and formal modelling by means of Petri Nets.

Jean-Jacques Lesage received the Ph.D. degree from the Ecole Centrale de Paris, France and the Habilitation Ãă Diriger des Recherches from the University of Nancy, France, in 1989 and 1994 respectively. Currently, he is Professor of Automatic Control with the Ecole Normale SupÃl'rieure Paris-Saclay, France. His research interests include formal methods and models for identification, analysis and diagnosis of Discrete Event Systems, as well as applications to manufacturing systems, network automated systems, energy production, and more recently ambient assisted living.