Inferring Moore Machine for Adaptive Online Hybrid Automaton Identification

Yan Monier, Gregory Faraut, Bruno Denis, Nabil Anwer

Inferring Moore Machine for adaptive online Hybrid Automaton Identification

Yan Monier, Gregory Faraut, Bruno Denis, Nabil Anwer

LURPA, ENS Paris-Saclay, Université Paris-Saclay, 4 Av. des Sciences, 91190 Gif-sur-Yvette.(name.surname@ens-paris-saclay.fr)

Abstract: In the context of CPS modeling, this paper proposes a new method for online adaptive identification of hybrid systems. The method relies on a Moore machine identification process; A mapping between a hybrid automaton and a discrete event Moore machine is proposed, and a new online adaptive Moore machine identification method is proposed. A case study illustrates the proposed method.

Keywords: Cyber-Physical System, Discrete Event System, Hybrid System, Identification.

1. INTRODUCTION

Digital twin (DT) [Qi et al. (2021)] and Cyber-Physical Systems (CPS) [Napoleone et al. (2020)] are well known as key enablers of the industry 4.0 revolution. However, some problems remain when developing and implementing such systems in the industry. It is then essential to develop methods and tools allowing a fast and personalized transition to match the industry 4.0 requirements and leading to models with high fidelity of CPS that fit the industry 4.0 needs [Oztemel and Gursev (2020)]. Such model shall respect a certain number of properties. Among them are the properties of the CPS in the context of digital twin modeling [Qi et al. (2021)]: the model needs to be able to simulate the system on its own and in synchronization with it. Moreover, it needs to follow the system during its life cycle; it may be able to adapt to fit the system's behavior, whatever the evolutions of the system. Finally, it may be connected to the physical and cyber environment of the system. The model needs to be interpretable by humans and take into account external factors from the Cyber and Physical spaces.

To model the dynamics of a CPS, it is necessary to take into account both process and controller, which interact in a closed loop. Early works relied on discrete event models. The first works were based on discrete event models, for example, in the field of diagnosis [Roth et al. (2011)] or supervision [Wonham et al. (2017)].An improvement of those models has been proposed through the development of timed models [de Souza et al. (2020)] incorporating time requirements. Those models are well suited to take into account the discrete dynamics of the controller but cannot take into account the continuous dynamics of the process. The solve this problem, a hybrid model able to represent discrete and continuous dynamics of the system has been produced [Lauer and Bloch (2019)]. Currently, the synthesis of those models is done by experts who design and implement specific models for specific purposes.

While this approach remains conceivable at the system design stage, it would be very time and resource-consuming for the recurrent adaptation of the model to the system behaviors evolutions' over its life cycle. The solution proposed in this work to overcome this problem is the automated generation of hybrid system models from the observation of input-output traces from the CPS. Such process is known as System Identification.

System identification has been performed on several types of models. It first started with continuous dynamical model identification [Liu and Hanssens (1982)], consisting in optimizing transfer function parameters. However, such models could not capture the discrete dynamics of the system representing several functioning modes. More focused on discrete state modeling, identification has also been performed for Discrete Event Systems [Estrada-Vargas et al. (2010)], such as Moore Machine [Giantamidis et al. (2021)] and timed systems [de Souza et al. (2020)]. However, those models could not capture the continuous dynamic of the system. This is the reason why identification methods have been developed to identify hybrid systems [Yang et al. (2021)] capturing both discrete and continuous dynamics of the system. However, the current solutions proposed to identify hybrid systems are inconsistent with the needs for CPS and digital twin modeling. Actual methods don't provide online identification of deterministic hybrid models that could be simulated and used as digital twins of cyber-physical systems. In [Soto et al. (2021)], an online identification method is proposed, but the generated model can only reproduce the observed behavior of the system and can't extrapolate its future evolution. In this paper, we propose to add three assumptions to actual methods of automaton inference :

1) No required knowledge about the initial state of the system. The inference method's input data come from experimental system signal records. The proposed method does not need to associate a unique and known state of the system at the beginning of each signal recording. This feature allows on-the-fly (or online) observations during system operation without setting up controlled observation sessions.

2) The observed repetitive behaviors of the system must be translated into cycles in the underlying directed graph

of the identified hybrid automaton. CPS have many repetitive behaviors. Our method will highlight these repetitive behaviors in the form of a cycle in the directed graph that is underlying in hybrid automaton. This will give more compact and more human-readable automata. A good representation of the control cycle behaviors should also enable the model to extrapolate the simulation of the system beyond the input data used for the inference process.

3) The inference method must be adaptive. In the context of the digital twin, a new challenge appears: updating the model as soon as the real system behavior evolves. When new data is recorded on the system, our adaptive approach takes into account the existing model and updates it with the new data, while a classic method builds a new model from scratch with the history of all recorded data. Then, our inference method should not require the recalculation of the complete automaton but only a local update of the part of the automaton impacted by new data. This should reduce the amount of computation compared to a complete model recalculation.

This paper will focus on adaptive hybrid automaton inference for online identification of hybrid systems [Yang et al. (2021)]. In section 2, the hybrid automaton identification problem is presented. In section 3, we then give the global framework of the Identification method we are developing. In section 4, we present the automaton inference algorithms developed. In Section 5, the process is illustrated through a simple academic example. Finally, we conclude our paper in Section 6 with a summary of research findings and future perspectives.

2. HYBRID AUTOMATON IDENTIFICATION AND PROBLEM STATEMENT

In the CPS context where controllers interact with sensor actuator signals, deterministic hybrid automaton with output and input will be considered for identification. When identifying the model, especially under the form of a Finite-State-Machine, most existing works assume that the set of traces used for identification starts from the same state. However, for the identification of CPS, it is impossible to guarantee that the observed traces start from the same place without prior knowledge of the system. We then suppose that the system's initial state for each trace observed is unknown. Identifying our system as a hybrid Automaton raises the problem of considering several possible initial states.

2.1 Preliminaries

Definition 1. (Hybrid Automaton). A hybrid automaton is a formal model of cyber-physical systems. Some external physical quantity (discrete, binary, or continuous) influences the system and makes it switch mode if a certain condition is satisfied. We define a Deterministically guarded hybrid automaton with Input and Output with Several initial states (DHAIOS) as a tuple of the following components. A hybrid automaton H is a 9-uplet: $(L, X_i, X_o, Flow, S, Guard, Jump, L_{init}, X_{init})$ with:

- L is a finite set $\{l_1, l_2, ...l_n\}$ of locations that represent modes of the hybrid system.
- X_i is a finite set of real-valued input variables while X_o is the output variables one. $X = X_i \cup X_o$ is set of all variables. We write $\dot{X}$ the set of $\dot{x}$, the first derivative variable of $x \in X$ during continuous evolution inside a mode, and we write X' the set of x', the update of $x \in X$ at the conclusion of a discrete change.
- $Flow$ is a function that associates a flow to a location. A flow is a predicate whose free variables are from $X \cup \dot{X}$; it states the possible continuous evolution when the hybrid system is in the mode represented by location l. The set of flows is noted F.
- S is a finite set of switches $s = ({}^\bullet s, s^\bullet) \in L \times L$, where ${}^\bullet s$ is the source location of s, while $s^\bullet$ is the target location of s. Switches represent discrete transitions.
- $Guard$ is a function that assign a condition $g \in G$ to every switch $s \in S$ where g is a predicate over the set X of variables and G the set of all guards conditions. An additional constraint guarantees that switches can not occur simultaneously. $\forall (s_1, s_2) \in S^2$ if ${}^\bullet s_1 = {}^\bullet s_2$ then predicate $g_1 \wedge g_2$ is false, where g_1, resp. g_2, is the guard condition of switch s_1, resp. s_2.
- $Jump$ is a function that maps every switch to a jump condition. A jump condition is a predicate over $X \cup X'$. It states the update of the variables' values when the hybrid system makes the discrete change. τ symbolizes the jump $x' = x$.
- $L_{init} \subset L$ is an non empty set of potential initial locations.
- X_{init} is a function that maps $l \in L_{init}$ to a predicate X^l_{init} on the initial valuation of X. $\forall l \in L_{init}$ and $\forall x \in X$ then $X^l_{init}(x) = x^l_{init} \in \mathbb{R}$.

Definition 2. (Guard/Jump/Flow Trace).

We define a Guard/Jump/Flow Tuple Gtu as a 3-tuple (g, j, fl) with : $g \in Guard$, $j \in Jump$ and $fl \in Flow$.

We define a Guard/Jump/Flow Traces Gtr of length $|Gtr| = z$, $z \in \mathbb{N}$, as a sequence of Guard/Jump/Flow tuples: $Gtr = Gtu_1...Gtu_z$ We note ϵ the empty guard, the guard used in the first tuple of the Guard/Jump/flow Traces if no guard is satisfied at the beginning of the Trace.

2.2 Related works

Hybrid system identification consists in building mathematical models of dynamical systems switching between different operating modes from experimental observations [Lauer and Bloch (2019)]. In the literature, several solutions are proposed to perform hybrid automaton identification over several research communities. [Yang et al. (2021)] proposes a framework for hybrid identification that is adapted by several other works depending on the purpose of the identification method [Lamrani et al. (2018)]. Others prefer to use artificial intelligence to solve the problem [Brusaferri et al. (2020)]. Computer scientists propose an online synthesis of a hybrid automaton [Soto et al. (2021)] which could be helpful for diagnostics but less for simulation.

However, the methods proposed to identify hybrid automata remain relatively similar, with each their advantages and flaws. We could, for most of the paper studied in

the literature, devise the hybrid automaton Identification Problem into five sub-problems:

- SP1: Change-point Identification (divides sampled signals into several groups)
- SP2: Mode Classification (regroups similar group of sampled data)
- SP3: Flow Identification (associate a flow to each previously identified mode)
- SP4: Guard and Jump Identification (find a switching condition explaining each change of mode)
- SP5: hybrid automaton Synthesis (generate the hybrid automaton from the previous result)

However, we noted two significant improvements that could be proposed for the hybrid system identification of CPS. One of them is about the lacking feature of online identification. A lot of previous works mention this feature as a possible improvement, but only a few have dealt with it [Soto et al. (2021)][Saberi et al. (2021)]. None to our knowledge has studied it for a hybrid automaton model respecting the properties of the model needed for Cyber-Physical Systems [Napoleone et al. (2020)].

The other improvement that can be made comes from the non-management of non-optimal decisions made while solving the different Sub-Problems. The point is that some results obtained by solving one sub-problem could help to solve the other sub-problems. For example, in the current framework, the mode classification sub-problem (SP2) is often solved after solving the change-point identification sub-problem (SP1). However, if a mode is isolated alone during the mode classification process (SP2), it is either a real singularity (a mode appearing only once in the training set), or it is due to a wrong decision performed during the change point identification process (SP1). This information could be transmitted back to the Change Point Identification sub-problem solver, which would propose, if necessary, a corrected change-point detection solution. The idea would be to develop retroactive loops between Sub-Problem solvers to improve the overall quality of the identification.

2.3 Problem statement

We noticed that most works dealing with hybrid system identification come from computer science or dynamical continuous system communities. They develop tools to solve sub-problems 1 to 4 and generally adapt an already existing automaton inference method to infer the hybrid automaton [Medhat et al. (2015)]. However, considering the future improvement that will be performed to solve sub-problem 1 to 4 (adaptive and online identification), no work in the literature proposes an adequate solution for sub-problem 5. Since our field of expertise is focused on Discrete Event Systems and Automaton Identification, the problem of hybrid automaton identification studied in this paper will only focus on developing an online adaptive solver of Sub-Problem 5. We will suppose that sub-problems 1 to 4 can be solved online and adaptively; the two main flaws identified in the previous section are corrected, so sub-problems 1 to 4 solvers can interact through a closed-loop communication process to adaptively update their solution. We define the online adaptive identification process as an identification process able to update its identification output (the model obtained by identification) according to a change in its input (the training set used for identification). The Problem of this paper is then how to solve the online and adaptive sub-problem identification 5 (hybrid automaton inference) with:

- as input of the method, the modification performed on one or several adaptive Guard/Jump/Flow Traces (for each trace, a Guard/jump/flow tuple can be changed, added, or removed). Since we aim at obtaining a model that can be simulated deterministically, we will suppose that the guard identified by sub-problems 1 to 4 solvers are exclusive (two guard predicates can't be verified at the same time, for the same valuation of variables)
- as Output of the method, a DHAIOS, the identified Deterministic Hybrid Automaton with Input and Output with Several potential initial states. The DHAIOS should also be able to represent by its structure the control cycle of the system it modelizes.

It is however important to notice that to our knowledge, such an adaptive online Guard/Jump/Flow Trace identification method doesn't exist yet in the literature. However several works propose methods solving Sub-Problems 1 to 4 [Lamrani et al. (2018)] [Saberi et al. (2021)] allowing to obtain with little adaptation, an equivalence of Guard/Jump/Flow Trace. Those results could still be used as input of the method we develop in this paper to perform online hybrid automaton inference.

3. GLOBAL FRAMEWORK BASED ON MOORE MACHINE IDENTIFICATION

In this section, we present the global framework of the proposed method and its link with the Moore Machine Identification Problem.

3.1 Global Framework

The method developed to solve Sub-Problem 5 is based on the rich literature about discrete event system identification [Estrada-Vargas et al. (2010)]. Several works about hybrid system identification propose a labeling procedure for guard, flow, or jump to manipulate them like events [Yang et al. (2021)]; some even use this labeling to infer Mealy Automaton [Medhat et al. (2015)]. Inspired by those works, the ideas and concepts of the proposed methods reside in a bijection between a Deterministic Moore Machine with several potential initial states (DMMS) and a DHAIOS associated with a newly developed online adaptive DMMS Identification method. The framework is presented (fig 1).

Considering the output given by the online and adaptive resolution of Sub-Problems 1 to 4, we develop in a first part a labeling algorithm allowing us to perform a bijection between Guard/Jump/Flow Traces and a Moore Machine Input Output Sequences. We then develop an online adaptive DMMS Identification Method allowing us to generate on the fly a DMMS. We then apply a reverse mapping algorithm that transforms the DMMS into a DHAIOS.

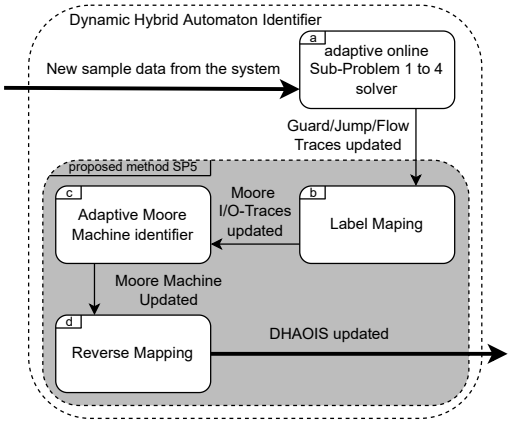

Fig. 1. DHAIOS identification Framework focused on the resolution of Sub-Problem 5

3.2 Moore Machine formalism

Definition 3. (Moore Machine). A Deterministic Moore Machine with multi-initial States is a tuple M of the form $M = (I, O, Q, q_0, \delta, \lambda)$, where:

- I is a finite set of input symbols.
- O is a finite set of output symbols.
- Q is a finite set of states.
- Q_0 is a non-empty set of initial states.
- $\delta : Q \times I \to Q$ is a transition function.
- λ is a output function, $\lambda : Q \to O$.

We also define $\delta^\star : Q \times I^\star \to Q$ as follows (for all set S, $S^\star$ denotes the set of all finite sequences over the set S ; $\epsilon \in S^\star$ denotes the empty sequence over S ; $w \cdot w'$ denotes the concatenation of two sequences $w, w' \in S^\star$. For $q \in Q$, $w \in I^\star$, and $a \in I$:

- $\delta^\star(q, \epsilon) = q$.
- $\delta^\star(q, w \cdot a) = \delta(\delta^\star(q, w), a)$.

We also define $\lambda^\star : Q \times I^\star \to O^\star$.

- $\lambda^\star(q, \epsilon) = \lambda(q)$
- $\lambda^\star(q, w \cdot a) = \lambda^\star(q, w) \cdot \lambda(\delta^\star(q, w \cdot a))$

We note $S_{I/O}$, the set of n Input/Output-traces (I/O-traces) S_k, with S_k a sequence of m_k I/O-pair $p_{k,l}$. I is the set of inputs and O is the set of outputs

$$S_{I/O} = \{S_1, ..., S_n\}, n \in \mathbb{N}$$
$$S_k = \{p_{k,1}, ..., p_{k,m_k}\}, m_k \in \mathbb{N}$$
$$p_{k,l} = (x, y),\ x \in I,\ y \in O$$

We name the tuple (k, l) the identifier of the I/O pair $p_{k,l}$, l symbolizes the position of the I/O pair in the k^{th} I/O-trace of $S_{I/O}$. We note $p_{k,l}(I)$ ($p_{k,l}(O)$) the input (the output) of a I/O-pair.

$$p_{k,l}(I) = x$$
$$p_{k,l}(O) = y$$

We call $\rho_i(S_k)$ ($\rho_O(S_k)$) the input (the output) sequence of S_k.

$$\rho_i(S_k) = (p_{k,1}(I), ..., p_{k,m_k}(I))$$
$$\rho_O(S_k) = (p_{k,1}(O), ..., p_{k,m_k}(O))$$

We define w, a I/O-word of size $|w| = z, z \in \mathbb{N}$ a sequence of z I/O-pair: $w = p_1 ... p_z$

3.3 Moore Machine Identification

The problem of learning Dynamical Moore Machines with Several potential initial states from Moore input-output traces is defined as follows [Giantamidis et al. (2021)]. Given an input alphabet I , an output alphabet O, and a set of Rtrain I/O-traces, called the training set, we want to synthesize automatically and dynamically a deterministic, complete, Moore machine $M = (I, O, Q, Q_0, \delta, \lambda)$, such that M is consistent with R_{train}, i.e., $\forall(\rho_I, \rho_O) \in R_{train}$: $\lambda^\star(\rho_I) = \rho_O$.(Rtrain is assumed to be itself consistent, in the sense it does not contain two different pairs with the same input word.)

4. AUTOMATON INFERENCE ALGORITHMS

As explained in the General Framework, the Resolution of the Sub-Problem 5 is divided into three steps, the Label Mapping, the Moore Identification method, and The reverse mapping from Moore Machine to DHAIOS.

We will note the identified DMMS: ${}^i\text{DMMS} = ({}^iI, {}^iO, {}^iQ, {}^iQ_0, {}^i\lambda, {}^i\delta)$ and the identified DHAIOS: ${}^i\text{DHAIOS} = ({}^iL, {}^iX_i, {}^iX_o, {}^iFlow, {}^iS, {}^iGuard, {}^iJump, {}^iL_{init}, {}^iX_{init})$

4.1 Label-Mapping

Considering a Guard/Jump/Flow Trace, ${}^iG$ is the set of Guards predicates in the Guard/Jump/Flow Trace, ${}^iF$ the set of flows predicate and ${}^iJ$ the set of jumps predicate. Each time a new Guard/Jump/Flow tuple is observed, the pair of flow and jump is mapped to a label, the set of those label is ${}^iO$. The Guard is mapped to a label, the set of those label is ${}^iI$. It Results in two bijective function, the Input Bijection to Guard (IBG) and the Output Bijection to Jump and Flow (OBJF):

$$IBG : {}^iI \to {}^iG$$
$$OBJF : {}^iO \to {}^iJ \times {}^iF$$

This mapping operation allows us to transform each Guard/Jump/Flow tuples into a Moore I/O-pair. For a given $x \in^i I$, we note G(x) the guard associated with x by the IBG. For a given $y \in^i O$, we note J(y) the jump associated with y by the OBJF and F(y) the flow associated with y by the OBJF.

4.2 Moore Identification

Considering the characteristics the identified hybrid automaton need to have, the Moore Machine identification method developed need to:

- identify a deterministic model.
- identify a model that is updated online adaptively depending on how the I/O-traces change.
- Highlight the control cycle behavior of the system.

The idea to overcome the online adaptive and deterministic issue consists in mapping every element of the Moore I/O trace to a location of the identified Moore Machine. Doing so, each time an element of the I/O trace change,

only the location or transition associated with it can be modified. To respect the determinism condition, we will also link each I/O pair to a history of antecedent I/O pairs, ensuring that for the given Input of the I/O pair, the model has no other option than generating the output of the I/O pair. Since several I/O traces will be assigned to the same location, it should highlight the control cycle behavior of the system. The General Framework of the Dynamic Deterministic Moore Machine Identification method is presented in fig 2.

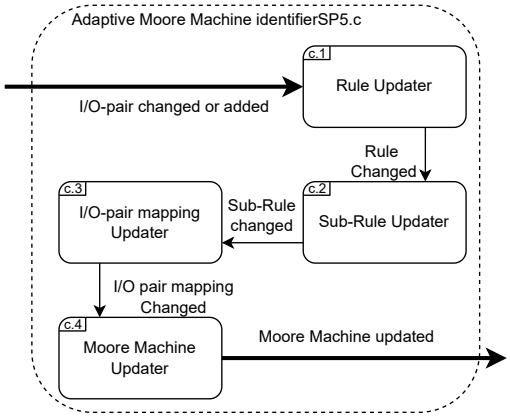

Fig. 2. Online Adaptive Moore Machine Identification Framework

The Rule Updater

The first operation of the method is called the Rule Updater, After observing a new or modified IO pair $p_{k,l} = (x, y)$, if necessary, the rule $r_{k,l}$ associated with the I/O pair is modified or added. The rule gives the minimal (in size) history sequence of I/O pair finishing by $p_{k,l}$ ensuring that, in any I/O traces used for the identification, the input x occurring after the history of the rule generates the output y.

We note $Hist(w)$, the history of a word w as the word preceding the last I/O-pair of w:

$$w = p_0, ..., p_d, d \in \mathbb{N}$$
$$Hist(w) = p_0, ..., p_{d-1}$$

We define the Rule $r_{k,l}$ as the word of the minimal size of trace S_k, ending by I/O-pair $p_{k,l} = (x, y)$ such as in any set of traces of $S_{I/O}$, the history of $r_{k,l}$ can't be followed by the I/O-pair (x, y_b) with $y_b \neq y$. The history of $r_{k,l}$ represents a deterministic condition on the occurrence of output y after observing input x.

$$r_{k,l} = min_{|w|}(\{w \in W_{prev}(k, l-1)_{\setminus \Omega(k,l)}\} \cdot p_{k,l}$$
$$\text{with: } \begin{cases} \Omega(k,l) = \bigcup_{i,j \in IE(k,l)} W_{Prev}(i, j-1)\}) \\ IE(k,l) = \{i, j \in \mathbb{N}^2, p_{i,j}(I) = x, p_{i,j}(O) \neq y\} \\ (x, y) = p_{k,l} \end{cases}$$

$W_{Prev}(k, l)$ is the set of world in S_k, ending by $p_{k,l}$:

$$W_{prev}(k, l) = \{w = p_{k,l-i}...p_{k,l}, i \in [0, ..., l-1]\}$$

The Sub-Rule Updater

The problem with the previously defined Rule is that if we want to respect a history for a given rule $r_{k,l}$ with $|r_{k,l}| = n$, we then at least need the previous I/O-pair to be consistent with the history of $r_{k,l}$. If $|r_{k,l-1}| < n - 1$, the rule $r_{k,l-1}$ is not a history long enough to ensure rule $|r_{k,l}|$, it is not consistent with it. We then need to define the set of history imposed on an I/O-pair by the rules of its succeeding I/O-pairs, that is the Sub-Rule.

We note w_{-1} the last I/O-pair of a word w:

$$w = p_0, ..., p_d, d \in \mathbb{N}$$
$$w_{-1} = p_{d-1}$$

We define $sr_{k,l}$, the Sub-Rule of an I/O-pair $p_{k,l}$ as the set of words ending by $p_{k,l}$ that are Sub-Word of a rule of an I/O pair succeeding $p_{k,l}$.

$$sr_{k,l} = \{w \in \bigcup_{i \geq l} SW(r_{k,i}), w_{-1} = p_{k,l}\}$$

with SW(w), the Sub-Word set of a word w, the set of words included in w and starting by the first I/O-pair of w:

$$SW(w) = \{w_i, i \in [0, ..., z-1]\}$$
$$\text{with: } \begin{cases} w = (x_1, y_1)...(x_z, y_z), z = |w| \\ w_i = (x_1, y_1)...(x_{z-i}, y_{z-i}) \end{cases}$$

The I/O-pair mapping updater

The next step is the I/O-pair mapping updater; it updates the mapping operated between a given I/O-pair and a word respecting the rules and sub rules of this I/O pair.

We define the Word to I/O-Pair Mapping (Wpm) as a function that associates the I/O-pair $p_{k,l}$ to the longer word among the rule and sub-rules of $p_{k,l}$ and consistent with the I/O-pairs before $p_{k,l}$.

$$Wpm(k, l) = max_{|w|}(\{w \in W_{prev}(k, l) \cap SR\})$$
$$SR = (sr_{k,l} \cup \{r_{k,l}\})$$

The Moore Machine Updater

The method's final step is the construction of the DMMS from the WPM elaborated during the previous step. The Moore Machine construction is executed in 5 steps:

- We define WM as the set of mapped words:
  $WM = \{Wpm(k, l), k \in [1, ..., |S_{I/O}|], l \in [1, ..., |S_k|]\}$
- For each word wm in WM, a location is created, with, as associated output the last output of the last IO-pair of wm:
  $${}^iQ = \{q_{wm}, wm \in WM\}$$
  $${}^i\lambda : q_{wm} \rightarrow wm_{-1}(O)$$
- For a given $wm \in WM$, if it exists an I/O-pair $p_{k,l}$, such as WPM($p_{k,l}$)=wm, then, if $p_{k,l+1}$ exists, a transition is created from the location associated to Wpm($p_{k,l}$) to the location associated to Wpm($p_{k,l+1}$), there can't be more than one transition going from WPM($p_{k,l}$) to WPM($p_{k,l+1}$). We define T, the set of transitions tuples:
  $$T = \{(q_{wm1}, q_{wm2}), \exists (k, l) \in (N)^2, Wpm(k, l) = wm1, Wpm(k, l+1) = wm2\}$$
- All transition is associated with the input of the last I/O-pair of the wm associated with its target location:

$$\forall (q_{wm1}, q_{wm2}) \in T$$
$$^{i}\delta : q_{wm1} \times wm2_{-1}(I) \rightarrow q_{wm2}$$

- The potential initial states are the states that are mapped to the first I/O pair of each I/O-sequences:

$$^{i}Q_0 = \{q_{wm}, wm = Wpm(k,1), k \in [1, ..., |S_{I/O}|]\}$$

The result of this 4^{th} step is then the identified Moore Machine: $^{i}DMMS = \{^{i}I, ^{i}O, ^{i}Q, ^{i}Q_0, ^{i}\lambda, ^{i}\delta\}$. A reduction algorithm can then be used if needed to reduce the model's size.

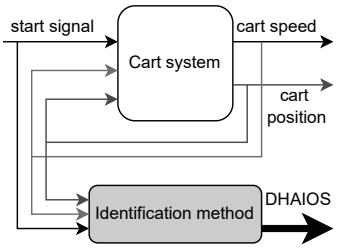

Fig. 3. Cart system identification process

4.3 Converting Moore Machine into DHAIOS

We finally need to define the reverse mapping converting the Moore machine back into a hybrid automaton. We define:

- $^{i}X_i$ the set of variables observed from the cyber-physical system considered as input.
- $^{i}X_o$ the set of variables observed from the cyber-physical system considered as output.

^{i}L is constructed from the set of states of the DMMS:

$$^{i}L = \{l_{wm}, qwm \in {}^{i}Q\}$$

The set flow function $^{i}Flow$ is constructed from the output function of the DMMS:

$$^{i}Flow : l_{qwm} \rightarrow F(\delta(qwm))$$

The set of switch ^{i}S is constructed from the transition function of the DMMS:

$$^{i}S = \{(l_{qwm1}, l_{qwm2}), {}^{i}\delta(q_{wm1}, wm2_{-1}(I)) = q_{wm2}\}$$

The guard function $^{i}Guard$ is constructed from the transition function of DMMS:

$$^{i}Guard : s \rightarrow G(wm2_{-1}(I)), s = (l_{qwm1}, l_{qwm2}) \in iS$$
$$^{i}\delta(q_{wm1}, wm2_{-1}(I)) = q_{wm2}$$

The Jump function $^{i}Jump$ is constructed from the output function of the DMMS:

$$^{i}Jump : s \rightarrow J(^{i}\delta(qwm2)), s = (l_{qwm1}, l_{qwm2}) \in iS$$

The potential states of initial location $^{i}L_{init}$ is constructed from the set of initial states:

$$^{i}L_{init} = \{l_{qwm}, qwm \in {}^{i}Q_0\}$$

The output function of the DMMS, applied to the potential initial states, gives the initial values of the variables:

$$^{i}X_{init} : l_{qwm} \rightarrow J(\delta(qwm)), l_{qwm} \in {}^{i}L_0$$

The identified hybrid automaton is then $^{i}DHAIOS$:

$$(^{i}L, ^{i}X_i, ^{i}X_o, ^{i}Flow, ^{i}S, ^{i}Guard, ^{i}Jump, ^{i}L_{init}, ^{i}X_{init})$$

Using an adapted Moore machine minimization method, the DHAIOS obtained can then be minimized if needed.

5. CASE STUDY

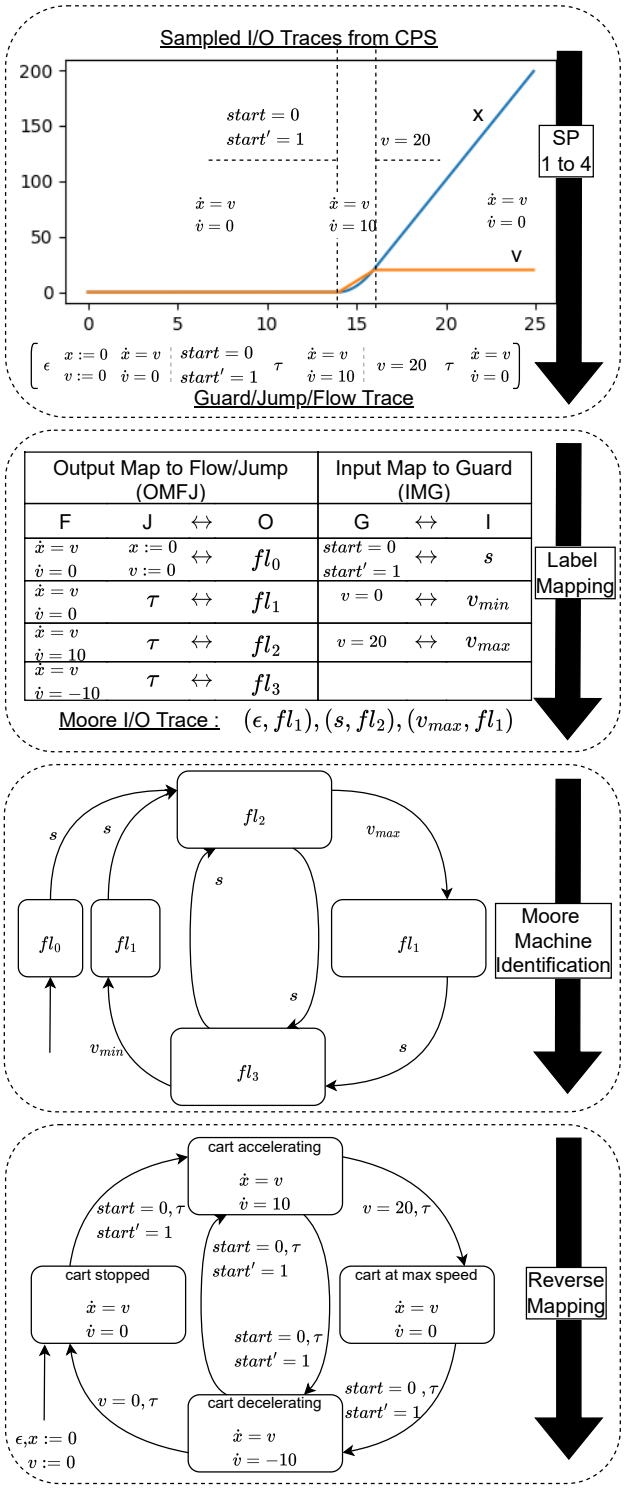

Fig. 4. Dynamic hybrid automaton identification example

The case study developed here is a simple system consisting of a cart controlled by an operator. The operator can click on a button to start or stop the cart. If the cart is "started", it accelerates until it reaches a maximum speed

(v=20). When it is "stopped", it decelerates until reaching zero speed. The identification process of this system is illustrated in fig 3. A signal is sent to the system by an operator (start signal) to start or stop the cart. The variable speed and position are two output variables of the systems. Fig 4 illustrates the four steps of the identification method:

- In the first step, the sampled traces of signal coming from the system is transformed into a dynamical Guard/Jump/Flow Trace
- In the second step, a mapping table is created, each flow and jump is associated with an Output alphabet, and all guards are associated with an Input alphabet; this allows to transform the Guard/Jump/Flow trace into a Moore Sequence.
- In the Third Step, the Moore Machine Identification method is applied
- Finally, the reverse mapping method is applied, transforming the Moore machine into a hybrid automaton

6. CONCLUSION AND FUTURE WORK

Based on a constructed bijection between a Moore Machine and a hybrid automaton, we succeeded in developing an Adaptive Online Moore Machine inference method solving the online adaptive hybrid automaton inference problem. The proposed method can infer a deterministic hybrid automaton from Guard/jump/Flow traces without prior knowledge of the system's initial state. The inferred automaton also displays control cycle behaviors of the identified system. Moreover, the method can locally update the inferred hybrid automaton if any Guard/jump/Flow traces used as input is extended, reduced, modified, added, or deleted. Future work will address a real case implementation and mathematical proof of the method's performance. We noticed that the method could still be improved, especially by modifying the rules and sub-rules generation properties, reducing the inferred automaton's size even more, thus increasing the pertinence of displayed control cycle behaviors. It is also important to notice that the full potential of this method can not be exploited yet with the actual methods solving hybrid system identification Sub-Problem SP 1 to 4. Other improvements must then be proposed to solve those 4 sub-problems adaptively.

REFERENCES

Brusaferri, A., Matteucci, M., Portolani, P., Spinelli, S., and Vitali, A. (2020). Hybrid system identification using a mixture of NARX experts with LASSO-based feature selection. In *2020 7th International Conference on Control, Decision and Information Technologies (CoDIT)*, 545–550. IEEE, Prague, Czech Republic.

de Souza, R., Moreira, M., and Lesage, J.J. (2020). A timed model for discrete event system identification and fault detection. *IFAC-PapersOnLine*, 53(2), 808–813.

Estrada-Vargas, A.P., López-Mellado, E., and Lesage, J.J. (2010). A Comparative Analysis of Recent Identification Approaches for Discrete-Event Systems. *Mathematical Problems in Engineering*, Vol.2010. Publisher: Hindawi.

Giantamidis, G., Tripakis, S., and Basagiannis, S. (2021). Learning Moore machines from input–output traces. *International Journal on Software Tools for Technology Transfer*, 23(1), 1–29.

Lamrani, I., Banerjee, A., and Gupta, S.K.S. (2018). HyMn: Mining linear hybrid automata from input output traces of cyber-physical systems. In *2018 IEEE Industrial Cyber-Physical Systems (ICPS)*, 264–269. St. Petersburg.

Lauer, F. and Bloch, G. (2019). *Hybrid System Identification: Theory and Algorithms for Learning Switching Models*, volume 478 of *Lecture Notes in Control and Information Sciences*. Springer International Publishing.

Liu, L. and Hanssens, D. (1982). Identification of multiple-input transfer function models. *Communications in Statistics - Theory and Methods*, 11(3), 297–314.

Medhat, R., Ramesh, S., Bonakdarpour, B., and Fischmeister, S. (2015). A framework for mining hybrid automata from input/output traces. In *2015 International Conference on Embedded Software (EMSOFT)*, 177–186. IEEE, Amsterdam, Netherlands.

Napoleone, A., Macchi, M., and Pozzetti, A. (2020). A review on the characteristics of cyber-physical systems for the future smart factories. *Journal of Manufacturing Systems*, 54, 305–335.

Oztemel, E. and Gursev, S. (2020). Literature review of Industry 4.0 and related technologies. *Journal of Intelligent Manufacturing*, 31(1), 127–182.

Qi, Q., Tao, F., Hu, T., Anwer, N., Liu, A., Wei, Y., Wang, L., and Nee, A. (2021). Enabling technologies and tools for digital twin. *Journal of Manufacturing Systems*, 58, 3–21.

Roth, M., Lesage, J.J., and Litz, L. (2011). The concept of residuals for fault localization in discrete event systems. *Control Engineering Practice*, 19(9), 978–988.

Saberi, I., Faghih, F., and Bavil, F.S. (2021). A Passive Online Technique for Learning Hybrid Automata from Input/Output Traces. *arXiv:2101.07053 [cs]*.

Soto, M.G., Henzinger, T.A., and Schilling, C. (2021). Synthesis of hybrid automata with affine dynamics from time-series data. In *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control*, 1–11. ACM, Nashville Tennessee.

Wonham, W.M., Cai, K., and Rudie, K. (2017). Supervisory Control of Discrete-Event Systems: A Brief History – 1980-2015. *IFAC-PapersOnLine*, 50(1), 1791–1797.

Yang, X., Beg, O.A., Kenigsberg, M., and Johnson, T.T. (2021). A Framework for Identification and Validation of Affine Hybrid Automata from Input-Output Traces. *ACM Transactions on Cyber-Physical Systems*.