FaMoS – Fast Model Learning for Hybrid Cyber-Physical Systems using Decision Trees

SWANTJE PLAMBECK, Institute of Embedded Systems, Hamburg University of Technology, Germany
AARON BRACHT, Institute of Embedded Systems, Hamburg University of Technology, Germany
NEMANJA HRANISAVLJEVIC, Institute of Automation Technology, Helmut Schmidt University, Germany
GOERSCHWIN FEY, Institute of Embedded Systems, Hamburg University of Technology, Germany

In the domain of cyber-physical systems, there is an increasing relevance of data-driven approaches for the learning of hybrid system dynamics. In particular, accurate models have been successfully abstracted from continuous (real-valued) traces and applied for various goals. However, industrial applications involving online modeling or rapid prototyping have two additional requirements: 1) runtime efficiency and 2) the interpretability of the approach and results.

This work adopts a common break down of this learning problem into four steps: 1) trace segmentation, 2) segment clustering, 3) characterization of the dynamics for each cluster (mode) and 4) learning of the overall model of mode transitions. Correspondingly, the bottlenecks in the state-of-the-art approaches are identified and discussed. Then, in a heuristic manner, interpretable and time-efficient algorithms for each of the steps are proposed giving a novel approach named FaMoS. The accuracy and runtime efficiency of the approach are evaluated for several system examples. FaMoS shows very short learning time, while the model's predictions of system dynamics are close to the ground truth behavior.

CCS Concepts: • **Computer systems organization** → *Embedded and cyber-physical systems*; • **Computing methodologies** → **Modeling methodologies**; *Classification and regression trees*; Online learning settings.

Additional Key Words and Phrases: Cyber-Physical Systems, Hybrid Dynamical Systems, Machine Learning, Decision Trees

ACM Reference Format:
Swantje Plambeck, Aaron Bracht, Nemanja Hranisavljevic, and Goerschwin Fey. 2024. FaMoS – Fast Model Learning for Hybrid Cyber-Physical Systems using Decision Trees. In *27th ACM International Conference on Hybrid Systems: Computation and Control (HSCC '24), May 14–16, 2024, Hong Kong, Hong Kong*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3641513.3650131

1 INTRODUCTION

Many digital systems of today, especially in the industrial domain and manufacturing, are Cyber-Physical Systems (CPS). CPS are a composite of embedded devices, actuators, sensors, and physical plants. All of these components are interdependent and influence each other over time. If available, abstracted dynamical models of the components' joint behavior can be powerful analysis tools, e.g., to detect faults in real-time or to verify system functionality.

However, creating or updating such dynamical models manually can be impracticable due to the following characteristics of CPS: 1) inner complexity, 2) emergent joint behavior of the components, and 3) frequent system changes [7, 29]. In this case, approaches for *data-driven modeling* are often considered. These are automated approaches which reconstruct the underlying system model from the observed traces of system behavior [31, 44, 45]. State-of-the-art machine learning techniques achieve convenient accuracy of the trained models which can be utilized for fault detection [17], diagnosis [11], health monitoring [44], or other tasks. However, for applications like rapid prototyping or online modeling in more conservative domains such as manufacturing [19, 27, 34], besides accuracy, three additional requirements are crucial:

(R1) Explainabillity of the learning process and the resulting model;
(R2) Good runtime efficiency (fast learning and application);
(R3) Small training dataset size.

Considering the existing solutions in model learning, this work aims at proposing an approach suitable to *R1-R3*. Typical CPS are often represented as *hybrid systems* combining discrete events and continuous signals [2, 10, 26]. This work adopts the common break down of the hybrid system learning problem into four steps presented in Fig. 1. The first step splits the collected data (observed traces) into segments by the detection of changepoints in signal dynamics (1). Afterwards, trace segments with similar dynamics are clustered together (2). Each of these clusters builds a mode whose dynamics must be characterized (3). Finally, an overall model is built from the results of the previous steps (4).

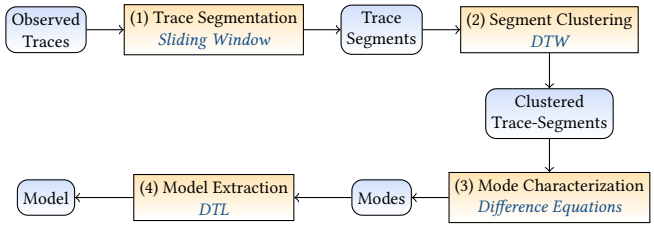


Fig. 1. General steps of hybrid system learning and in *italic* concrete algorithms used in FaMoS.

Correspondingly to this taxonomy, time-efficiency and interpretability bottlenecks of concrete implementations involved in five state-of-the-art approaches (presented in Section 2) are identified and discussed. Then, in a heuristic manner, suitable algorithms for each of the steps are proposed (refer to Fig. 1), giving a novel approach named FaMoS:

(1) In the segmentation step, an efficient sliding window approach is used;
(2) FaMoS uses a clustering strategy inspired by Dynamic Time Warping (DTW);
(3) Parameters of learned difference equations are used to characterize each mode;

HSCC '24, May 14–16, 2024, Hong Kong, Hong Kong
ACM ISBN 979-8-4007-0522-9/24/05
https://doi.org/10.1145/3641513.3650131

(4) Decision Tree Learning (DTL) is used to learn an overall model, providing a new paradigm for hybrid decision trees.

The evaluation of FaMoS performed on several example systems shows a promising trade-off between runtime efficiency and accuracy of the learned models. The empirical results specifically compare FaMoS to an open-source implementation of one existing approach for hybrid system learning (proposed in [45]).

In the following, we first discuss related work and point to bottlenecks of five existing learning approaches (Section 2). Section 3 provides the basics of decision trees and hybrid automata. Section 4 describes the complete learning approach of FaMoS in detail. Section 5 provides the empirical results of FaMoS. Finally, Section 6 discusses limitations and opportunities of FaMoS. A conclusion follows in Section 7.

2 RELATED WORK

For the learning of hybrid systems or hybrid automata which is often used as a synonym, convergence to the exact system representation as known, e.g., for discrete finite automata (DFA) [3, 16, 20], cannot be guaranteed. In fact, there exist negative results on the learnability and verifiability of hybrid systems [3, 20] which are based on the infinite state space of hybrid systems as well as on typical non-deterministic behavior. A hybrid automaton learned from the observations of a system is always an abstract representation. Therefore, the goal of learning strategies is to reduce the approximation or to trade off approximation against compactness, interpretability, or learning time. For example, there are frameworks considering the modeling of CPS for explainability [8] or diagnosis [32].

The diversity of existing approaches for learning CPS from observations is large. For example, neural networks have the advantage of producing highly accurate models due to their "universal approximation" property. Steude et al. [39] discuss the application of representation learning (Deep Learning (DL)) in the context of CPS. However, such approaches are typically not suitable if explainability is one of the requirements [4, 17]. At the same time, required dataset size and learning time are limitations for some applications. Other strategies to model system behavior include deterministic automata [38, 43], hybrid automata [10], and purely physical models, e.g., differential equations [12]. Moreover, in the field of switching dynamical systems the focus is on detecting switches in continuous dynamics, while modeling of complex mode transitions is less considered (e.g., see [21]). Finally, a universal applicability and accuracy guarantee for single learning algorithms does not exist in the literature [13]. For five existing learning approaches and for FaMoS, Table 1 lists the key algorithms used in the four learning steps (from Fig. 1). The list of learners is not exhaustive, but showcases the diversity in learning strategies and allows to relate FaMoS to the existing approaches.

Trace segmentation: HAutLearn uses a simple and fast approach based on extrema of the first derivative of observed signals. HyBUTLA applies a more sophisticated wavelet analysis to quickly detect abrupt changes in the signals. These abrupt changes are than treated as additional events for the discrete part of the automata learning algorithm. Another approach, DENTA, applies neural-network based representation learning hoping to detect changes easily in the learned latent space using simple thresholding. POSEHAD applies a sliding window approach with discrepancy analysis while DyClee-Based uses a dynamic clustering approach.

Segment clustering: In HAutLearn, clustering is performed using linear matrix inequality, which is adapted in FaMoS and prepended with a DTW-based clustering. On the other hand, HyBUTLA implicitly clusters the segments during the PTA merging procedure (described below) using the parameters of simple continuous models learned from segments. DENTA performs clustering in an implicit way as well, by fitting a probabilistic model and discovering latent discrete units which can explain the clusters in the observed signals. POSEHAD uses DTW and similarty indices while DyClee-Based applies a second stage of dynamic clustering.

Mode characterization: HAutLearn uses ordinary differential equations (ODE) to characterize each mode. In HyBUTLA, mode characterization can be performed only after the transition model is learned. Once a timed automaton is created, for each mode a separate continuous model is learned. This can be a simple recurrent network, or even just a range of values for each signal, for each mode of the automaton. DENTA does not explicitly characterize each mode. This is implicitly achieved by the deep network as each mode defines a posterior distribution of the visible units given the latent units of the probabilistic model. POSEHAD uses polynomials, while the DyClee-based approach uses the established support vector regression and random forest regression.

Overall model extraction: Learning of the overall transition model depends on the previous steps for each of the five approaches. HAutLearn tries to identify the guard conditions of a discrete state (mode) by means of linear matrix inequality and automata learning via a Prefix Tree Acceptor (PTA). Similarly, HyBUTLA constructs an automaton via PTA using an intensive iterative process of comparing different subtrees in the PTA. Although, the rule for comparing the subtrees is given in an analytical way, the resulting model is not simple to interpret. The reason is the iterative process disabling the possibility to directly explain a learned mode. Mode characterization is performed after learning of the discrete part is finished. DENTA, on the other hand, relies on the features extracted using the deep neural network and therefore applies a very fast and interpretable algorithm to learn mode transitions. The whole approach must still be considered non-efficient in terms of runtime and less interpretable due to the DL step beforehand. POSEHAD, similarly to HAutLearn tries to identify jump conditions for each discrete state by analyzing confidence levels and automata learning. In contrast to the other approaches, the DyClee-based approach uses Heterogeneous Petri Nets to capture the overall behavior of the system.

FaMoS uses decision trees to represent the temporal behavior of the system, i.e., the transitions between system modes. Even though decision trees are usually used for classification, there exists also other work considering decision trees on temporal data. In Lucena-Sanchez et al. [25], feature selection strategies for decision and regression tree learning on time-series data are discussed. In another approach, decision trees on temporal data are learned via interval logic [36]. Limitations of using decision trees in comparison to DFAs are formalized and evaluated in [33]. Della Monica et al. [15] use decision trees on temporal logic to classify time series

Table 1. Comparison of learning strategies for hybrid systems. The key runtime and interpretabillty bottlenecks for each method are given bold.

Step	HAutLearn [45]	HyBUTLA [31]	DENTA [17]	POSEHAD [34]	DyClee-Based [44]	FaMoS
Trace Segmentation	Extrema of the first derivative of observed signals	WA with manually defined thresholds	Thresholds on features extracted using **DL network of RBM**	Sliding-window with discrepancy	Dynamic-Clustering (1. stage) [5]	Sliding-window with distance
Segment Clustering	**LMI**	Parameters of continuous models used in AL state merging	Simple clustering using thresholds in the latent space	Similarity indices from DTW comparisons	**Dynamic-Clustering (2. stage)**	DTW-inspired pre-clustering and LMI
Mode Characterization	ODEs	Linear Regression and NN	Probability function	Polynomials	SVR & RFR	ARX model
Model Extraction	Guard Conditions from LIs and AL via PTA	**AL via PTA**	Simple AL in the latent space	**Mining of jump conditions via confidence levels and AL**	HtPN	DTL

Abbreviations: Wavelet Analysis (WA), Restricted Boltzmann Machine (RBM), Deep Learning (DL), Linear (Matrix) Inequality (LMI / LI), Dynamic Time Warping (DTW), Neural Networks (NN), Decision Tree Learning (DTL), Prefix-Tree Acceptor (PTA), Heterogeneous Petri Nets (HtPN), Support Vector Regression (SVR), Random Forest Regression (RFR), Automata Learning (AL)

data. Furthermore, automata learning strategies internally use tree structures for intermediate representations [18].

3 PRELIMINARIES

This section shortly introduces the basics and notation of decision trees, hybrid automata, and observations on the system.

3.1 Decision Tree Learning

DTL is a lightweight machine learning technique used for classification and regression. A decision tree [37] is a tree $T = (V, E)$ with vertices V and edges E that represents a classifier $d : \mathcal{X} \to C$. The set $\mathcal{X}$ consists of vectors of feature values $\mathbf{f} = [f_1, ..., f_m]$, while C is a set of classes.

A common decision tree learning algorithm is the CART algorithm which results in a binary decision tree [24].

3.2 Hybrid System

We use the well-known formalization of hybrid systems [10] for CPS which we formalize by a 5-tuple $(X, Q, \Sigma, \mathcal{F}, \mathcal{T})$ where

- $X = \{x_1, x_2, ..., x_n\}$ is the set of variables,
- Q is the set of system modes,
- $\mathcal{F}$ defines the flow, i.e., change of the variables X within each system mode,
- Σ is a set of events leading to transitions between system modes. Each event is a set of intervals on the domains of the variables in X. The event is considered active if all of the variables are in their given interval,
- $\mathcal{T} : Q \times \Sigma \to Q$ defines transitions between system modes Σ. A transition is triggered if the corresponding event $\sigma \in \Sigma$ is active.

The variables in X are input variables $i_1, ..., i_k$ and output variables $o_1, ..., o_l$ of the system. We also refer to derivatives of the system variables as $X^{(d)} = \{x_1^{(d)}, x_2^{(d)}, ..., x_n^{(d)}\}$.

For data aggregation, the hybrid system is observed over time. The learning data consists of time-discrete traces

$$\mathbf{t} = [X[\tau], X[\tau + 1], X[\tau + 2], \dots].$$

We assume an identical, fixed sampling rate for all variables. A *signal* is the 1-dimensional observation of a single variable.

Further characteristics which are typical, benign characteristics are *continuous differentiable* flow functions $\mathcal{F}$ on all variables which usually applies to physical influences. Furthermore, a system is to be designed *robustly*, when small deviations in inputs or initial values lead only to small deviations from the nominal system behavior [22, 40].

In the scope of this paper, we model the dynamics $\mathcal{F}$ of every system mode by AutoRegressive models with eXogenous inputs – ARX models [23]. The dynamics of each system mode are represented by a time-discrete LTI model using a *difference equation*. This state equation is defined as follows:

$$\mathbf{y}[\tau + 1] = \mathbf{A}\mathbf{y}[\tau] + \mathbf{B}\mathbf{u}. \tag{1}$$

The matrices $\mathbf{A}$ and $\mathbf{B}$ are system matrix and input matrix, respectively, while the vectors $\mathbf{y}$ and $\mathbf{u}$ are state and input vector, respectively. The input vector represents the input variables $i_1, \dots, i_k$ while the state vector considers current and past output variables $o_1, \dots, o_l$ forming a higher order difference equation [14]:

$$\mathbf{y}[\tau] = \tag{2}$$
$$\begin{bmatrix} o_1[\tau] & \cdots & o_l[\tau] & \cdots & o_1[\tau + k_c] & \cdots & o_n[\tau + k_c] \end{bmatrix}^T.$$

The order of the difference equation is given as $k_c + 1$ using the parameter k_c.

4 MODELING PROCEDURE

In this section, we provide the details of FaMoS' learning process, i.e. its four steps shown in Fig. 2.

4.1 Trace Segmentation

The goal of trace segmentation is the identification of a change in the dynamics of the system behavior. In FaMoS, the trace segmentation is realized with a sliding window approach [42]. For every time point within a trace, the immediate past and immediate future are windows sliding over the whole trace. The deviation of these two windows yields a metric for the change in system behavior. The Euclidean distance between the windows is used as a deviation metric. Thus, for a signal $\mathbf{t}_x$ of variable x and length N the deviation between immediate future and immediate past for time point τ is calculated as follows

$$\begin{aligned}\text{deviation}_{\mathbf{t}_x}[\tau] = \sum_{k=0}^{w-1} |(\mathbf{t}_x[\tau - w + k] - \mathbf{t}_x[\tau - w]) \\ - (\mathbf{t}_x[\tau + 1 + k] - \mathbf{t}_x[\tau + 1])|, \end{aligned} \tag{3}$$

where w gives the window width and is a parameter of the trace segmentation. The subtraction of $\mathbf{t}_x[\tau - w]$ and $\mathbf{t}_x[\tau + 1]$, respectively, aligns both windows to an initial value of 0. Evaluating the deviation for all time points $\tau \in [w, N - w]$, we have a time-discrete deviation function. Established algorithms to detect local maxima (peak detection) are applied on the deviation function to extract the changepoints. Thus, large deviations between immediate future and immediate past are considered as changes in the dynamic system behavior. Especially for differentiable flow functions (see Section 3.2), the sliding window approach provides an efficient strategy to detect dynamic changes on the observed traces.

As the dynamic behavior might be defined by higher order difference equations, the sliding-window trace segmentation is applied to the derivatives of the observed variables as well. The maximum number of derivatives d_{max} to be considered is a parameter of FaMoS. The trace segmentation is carried out on every signal separately. Finally, all changepoints are collected. Changepoints which are within a distance of w_{size} are combined into a single changepoint. Traces $\mathbf{t}$ are then split into segments $\mathbf{\Delta t}$ at the changepoints.

4.2 Segment Clustering

Segments of similar behavior are detected in a segment clustering step. Afterwards, every cluster represents a specific dynamic behavior, i.e., a mode of the hybrid system. Instead of explicitly evaluating the dynamics for each trace to analyse their similarity, FaMoS performs clustering using a DTW-inspired comparison which is a fast and intuitive similarity check.

Fig. 2 shows the steps of this clustering process. In Step (1), the derivative d_c, considered for clustering, is selected. Especially for hybrid systems whose behavior is defined by higher order dynamics, the geometric similarity in derivatives is a better indication than the trace itself. The degree of the derivative d_c is a parameter of FaMoS. In the next step, all pairs of segments are created for a pairwise comparison (2). Afterwards, a DTW-inspired comparison is done for every signal on the variables x_1 to x_n (3).

DTW is a technique to compare the similarity of two vectors. We use a DTW-inspired procedure in FaMoS to determine the similarities of segments (4). Fig. 3 illustrates this adaptation of common DTW [28]. Common DTW can be applied to segments of different lengths to identify similar dynamics under differing sampling rates. Here, we assume a fixed sampling rate, thus, comparing segments of identical lengths. To bring two segments to the same length, we can align them either at the start – cutting of the end of the longer segment – or align them at the end – cutting of the beginning of the longer segment. Both approaches are executed as shown in Fig. 3 and forwarded to a DTW comparison.

The DTW comparison of two segments $\mathbf{\Delta t_1}$ and $\mathbf{\Delta t_2}$ of length N yields a distance value dist^{DTW} and a diagonality value diag^{DTW} which results from an alignment mapping. Two segments are considered similar if the distance and the alignment values are low [34]. We combine diagonality and distance into a single similarity metric using a weighted sum:

$$\begin{aligned}\text{s}^{\text{DTW}}(\mathbf{\Delta t_1}, \mathbf{\Delta t_2}) = & w_{\text{diag}} \cdot \text{diag}^{DTW}(\mathbf{\Delta t_1}, \mathbf{\Delta t_2}) \\ & + w_{\text{dist}} \cdot \frac{1}{L} \cdot \text{dist}^{\text{DTW}}(\mathbf{\Delta t_1}, \mathbf{\Delta t_2}), \end{aligned} \tag{4}$$

where $w_{\text{diag}} + w_{\text{dist}} = 1$ holds. FaMoS uses $w_{\text{diag}} = w_{\text{dist}} = 0.5$. The distance metric is additionally normalized to the length L of DTW's alignment. After evaluation of the similarity $\tilde{s}$ and $\hat{s}$ for the start- and end-aligned segments, the minimum of the two similarities is used as the final similarity s of the two segments as shown in Fig. 3.

For every signal, a similarity matrix is constructed (5). Clusters are found by defining thresholds ρ_i. Based on these thresholds, we declare for every row i in the similarity matrix all segments j with $s_{i,j} \leq \rho_i$ to come from the same cluster as segment i and provide cluster-IDs for all clusters found (6). Especially when considering robust ARX models, traces coming from the same dynamics are efficiently identified with this strategy.

Example 4.1. For an example set of six trace segments, the following similarity matrix results:

$$S = \begin{pmatrix} 0 & 0.33 & 0.33 & 0.07 & 0.18 & 0.33 \\ 0.33 & 0 & 0.05 & 0.17 & < 0.01 & 0.05 \\ 0.33 & 0.05 & 0 & 0.25 & 0.03 & < 0.01 \\ 0.07 & 0.17 & 0.25 & 0 & 0.15 & 0.11 \\ 0.18 & < 0.01 & 0.03 & 0.15 & 0 & 0.03 \\ 0.33 & 0.05 & < 0.01 & 0.11 & 0.03 & 0 \end{pmatrix}$$

Small values in the similarity matrix indicate similar segments. Thus, the diagonal is zero. Furthermore, the segments 2 and 5 can be considered similar as well as 3 and 6, because of their small similarity metric. Choosing $\rho_i = 0.1$ for all rows, the following clusters result: $\{1, 4\}, \{2, 3, 5, 6\}$. We label these clusters with ID 'a' for the first and 'b' for the second cluster.

FaMoS uses the following ρ_i:

$$\rho_i = \alpha \cdot \min_{j \neq i} s_{i,j},$$

where α is a similarity threshold factor. By choosing $\alpha > 1$, every segment is similar to at least one other segment. In addition, we limit ρ_i to an interval $[\rho_{min}, \rho_{max}]$ to avoid clustering too many or too few segments:

$$\rho_i = \min(\max(\alpha \cdot \min_{j \neq i} s_{i,j}, \rho_{min}), \rho_{max}). \tag{5}$$

Having found clusters on all signals, we combine the vector of cluster IDs from all signals to a global cluster in every time step (7). This is illustrated in Fig. 4 for the signals $\mathbf{\Delta t}_{x_1}, \mathbf{\Delta t}_{x_2}, \mathbf{\Delta t}_{x_3}$. Every global cluster, finally, represents a mode of the hybrid system.

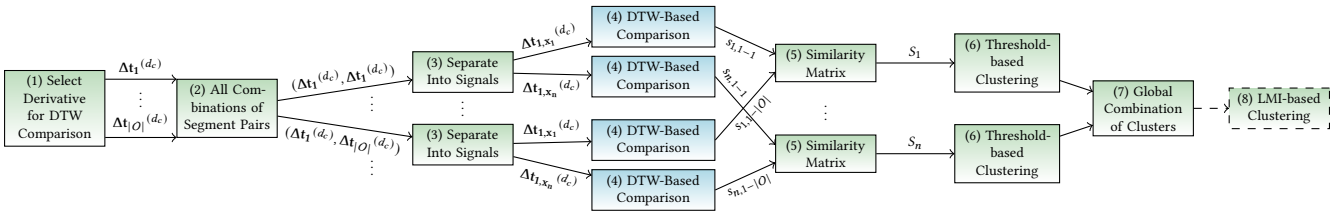


Fig. 2. Segment Clustering in FaMoS

Fig. 3. DTW Adaptation

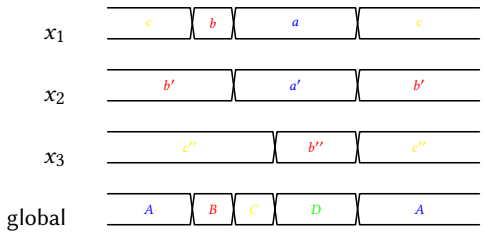


Fig. 4. Logic Combination to Global Clusters

FaMoS provides an additional optional Step (8) in the clustering process which does a refined clustering. So far, the clustering depends on the choice of the parameter d_c and the thresholds ρ_i. In case that an intensive parametrization is not possible or a chosen parametrization does not lead to the required performance, the refined clustering improves the DTW-inspired preliminary clustering. With unoptimized parametrization, not all similarities might be detected, meaning that there are multiple clusters which actually refer to the same system mode. To identify those similar clusters, FaMoS performs a comparison of the solution spaces of difference equations, i.e., a direct comparison on the segment's dynamics. This is done with a Linear Matrix Inequality (LMI) formulation as used in [45]. A pair of preliminary clusters (after DTW-inspired clustering) is compared using LMI. The two clusters are combined to one cluster, if the solution space of their dynamics within a threshold σ is overlapping. The DTW-inspired clustering is very fast while intuitively checking segment similarity. Nevertheless, clustering with the DTW-inspired approach depends on the choice of the parameters such as d_c and the thresholds ρ_i. LMI compares the actual dynamics of two segments and allows to cluster previously undetected similarities. The solution of the LMI formulation is computationally intensive and time consuming. The proposed combination of the two approaches reduces the problem size for LMI. Thus, overall a moderate or small clustering effort is achieved.

4.3 Mode Characterization

After reconstructing the system modes by clustering, we characterize the dynamics of each mode, i.e., define the flow relation $\mathcal{F}$ which characterizes each system mode with a difference equation. To achieve this, the matrices $\mathbf{A}$ and $\mathbf{B}$ from Equation 1 are identified. The matrices are found by solving the following problem for every cluster

$$\min \|\mathbf{Y}_O[\tau+1] - \mathcal{A}\mathbf{Y}_O[\tau]\| \tag{6}$$

where $\mathbf{Y}_O[\tau]$ is the observation matrix on the set $O = \{\Delta\mathbf{t_1}, \ldots, \Delta\mathbf{t_m}\}$ of trace segments of the considered cluster:

$$\mathbf{Y}_O[\tau] = \tag{7}$$

$$\begin{bmatrix} \mathbf{y}[\tau^0_{\Delta\mathbf{t}_1}] \cdots \mathbf{y}[\tau^{|\Delta\mathbf{t}_1|-1}_{\Delta\mathbf{t}_1}] \cdots \mathbf{y}[\tau^0_{\Delta\mathbf{t}_m}] \cdots \mathbf{y}[\tau^{|\Delta\mathbf{t}_m|-1}_{\Delta\mathbf{t}_m}] \\ \mathbf{u}[\tau^0_{\Delta\mathbf{t}_1}] \cdots \mathbf{u}[\tau^{|\Delta\mathbf{t}_1|-1}_{\Delta\mathbf{t}_1}] \cdots \mathbf{u}[\tau^0_{\Delta\mathbf{t}_m}] \cdots \mathbf{u}[\tau^{|\Delta\mathbf{t}_m|-1}_{\Delta\mathbf{t}_m}] \end{bmatrix}, \tag{8}$$

The matrix $\mathcal{A} = [\mathbf{A}, \mathbf{B}]$ finds the state and input matrix of the state equation, respectively.

4.4 Overall Model Extraction

The final step in FaMoS is model extraction with DTL. As introduced in Section 3.1, decision trees usually represent a classification function. Here, the learned decision tree represents rules for changing or staying in a system mode and defines a new modeling paradigm for hybrid systems.

Definition 4.2. Hybrid Decision Tree: Following the formalism of hybrid systems, we define a hybrid decision tree as a 5-tuple $(X, V_H, E_H, Q, \mathcal{F})$ where

- $X = \{x_1, x_2, ..., x_n\}$ is a set of variables,
- V_H, E_H are a set of vertices and edges, respectively, which form a tree,
- Q is a set of system modes, and
- $\mathcal{F}$ defines the flow, i.e., change of the variables X within each system mode.

The tree V_H, E_H represents a classification function $d_H : Q \times X \to Q$ which encodes rules for the transition between modes of the system. Using this function, the hybrid decision tree predicts the upcoming system behavior based on the current mode and variable values.

The construction of the hybrid decision tree is done with common DTL algorithms. The following feature vector and label are used in DTL for FaMoS:

$$\mathbf{f} = [q[\tau],\ x_i[\tau]\ \forall i \in [n]\ ] \tag{9}$$
$$c = q[\tau+1],$$

where

- $q[\tau]$ gives the current mode of the system at time τ,
- $x_i[\tau]$ $\forall i \in [n]$ gives the value of all system variables at time τ.
- $q[\tau+1]$ gives the system mode in the next time step $\tau+1$.

Using this feature vector, the learned decision tree has all information needed to construct transitions of the hybrid system, i.e., two states $q[\tau]$ and $q[\tau+1]$ as well as the variable values to identify events. By this, the decision tree learns transitions in form of compact classification rules. Every leaf in the decision tree represents a system mode and is associated with its respective flow function found during mode characterization.

A training set $\mathcal{L}$ for DTL is created from the results of the previous steps. The result of mode characterization are traces which are separated into segments and clustered. For every cluster a system mode defined by a difference equation describing the flow function $\mathcal{F}$ is known. From this information, we extract the feature vector and class label as defined above for every sampling point τ in the collected traces. The learning set created from the traces is then handed over to the DTL algorithm for model extraction. The resulting hybrid decision tree is used as the final hybrid model of the system.

4.5 Limitations & Opportunities from Parametrization

Table 2 lists the parameters of FaMoS which have been introduced in the previous sections. The sampling period T should be chosen according to known rules for signal processing and allow for a clear identification of dynamics. For trace segmentation and clustering in FaMoS, a low number of samples per segment is often sufficient. For peak detection with the sliding window segmentation, the window w should cover a reasonable range around a changepoint, thus w should be similar to the expected length of segments. For clustering, the impact of parametrization is reduced by using the refined LMI-based clustering on top of DTW-based clustering. For mode characterization, the order of the difference equation defined by k_c in Equation 1 has to be chosen. Higher order equations can also model lower order dynamics, thus, putting an upper bound is sufficient.

The parametrization allows an adaptation to various different systems. Furthermore, FaMoS' changepoint detection is independent from the actual system dynamics and, thus, universally applicable. A drawback of generalization on the other hand is less specificity. DTW comparisons, for example, are more qualitative than LMI solutions which directly compare solution spaces of dynamics.

For this work, we characterize dynamics by difference equations. Nevertheless, our generalized segmentation and DTW-based clustering is independent of the characterization of dynamics which allows for an easy adaptation to other dynamic representations.

5 EXPERIMENTS

5.1 Experiment Process & Parametrization

In this section, we present an empirical evaluation of the FaMoS framework to quantify the accuracy and the execution time and compare to the state-of-the-art algorithm HAutLearn. All benchmarks are carried out on the same computer (Intel i7-8650U 1.9GHz, 8 GB Memory, charging). The FaMoS framework is implemented in MATLAB R2023b [41]. The source code and examples for FaMoS used for this paper are publicly available in [9].

All parameters of the learning process are provided in Table 2. Parameters of the systems are the sampling time T and the number of signals n consisting of k input and l output signals. The maximal considered derivative d_{max}, the window size w, and the distance under which two changepoints are merged w_{size} are the parameters for the trace segmentation. In the segment clustering step, HAutLearn uses LMI-based clustering while FaMoS uses DTW-based clustering with an option for refined clustering with LMI. In segment clustering, HAutLearn has the parameters σ (tolerance) and the maximal considered time-difference k_c in the state space vector. FaMoS has the parameters α, ρ_{min}, and ρ_{max}.

Model extraction is done with DTL in FaMoS while HAutLearn uses a PTA-based approach. HAutLearn's model extraction has the parameters η, λ and γ which are introduced in [45]. No parameters are required for model extraction in FaMoS.

To find the parameters for our empirical evaluation, one parameter is varied while the remaining parameters are kept constant. This loop over all parameters is repeated until a local optimum is reached.

5.2 Adaptations of HAutLearn

In our experimental evaluation, we compare to the HAutLearn framework. Some adaptations and improvements are done to allow the evaluation of all examples. Originally, HAutLearn is designed to model only difference equations of degree one which are transformed to continuous differential equations in the final model. We extend HAutLearn to allow the inclusion of further differences in the state vector $\mathbf{y}$, such that higher order hybrid systems are learned. Furthermore, HAutLearn expresses event conditions in form of linear inequalities either as

$$\sum_{i=0}^{n} a_i \cdot x_i + 1 < 0 \text{ or as } \sum_{i=0}^{n} a_i \cdot x_i + 1 > 0\,.$$

In our experiments, relaxing these expressions to

$$\sum_{i=0}^{n} a_i \cdot x_i + 1 < \psi \text{ and } \sum_{i=0}^{n} a_i \cdot x_i + 1 > -\psi\,,$$

improves the model accuracy significantly. Thus, ψ is another parameter of HAutLearn.

5.3 Examples

For the empirical evaluation, five systems are used. For each system, nine traces with roughly 2000 samples each are generated using different initial conditions. One trace is used for the evaluation while up to eight traces are used for training. The values in the traces are normalized to $[0, 1]$ making the results of all examples comparable. The first set of examples: a Simple Heating System (SimpHeatSys) and a Buck Converter (BuckConv) [6] are introduced in [45]. Furthermore, a Complex Tank System (ComplexSys) models an irrigation system consisting of three tanks each equipped with a pump and associated with different leakage rates. The hybrid system is designed to reduce the peak power consumption of the pumps while no tank runs empty. The time while more than one pump

Table 2. Parameters of FaMoS

Step	Parameter	Description
Simulation / Data Aggregation	T	Sampling period
Simulation / Data Aggregation	n	Number of variables
Trace Segmentation	w	Window width
Trace Segmentation	w_{size}	Minimum distance of changepoints
Trace Segmentation	d_{max}	Maximum order of derivative
Clustering	d_c	Order of derivative
Clustering	$\alpha, \rho_{min}, \rho_{max}$	Similarity thresholds
Clustering	σ	LMI tolerance
State Equation	k_c	Order of difference Equations

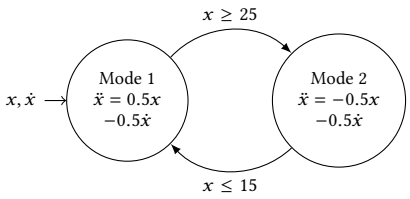


Fig. 5. Two State Hybrid Automaton

is running is minimized, by activating pumps only if a water level threshold for any tank is underrun. This example represents a larger hybrid system with eight states and three output variables. The Variable Heating System (VarHeatSys) is derived from the Simple Heating System by introducing an input variable. This input variable is a switch that increases the average temperature and heating power. Finally, a Two State Hybrid Automaton (TwoState) as shown in Fig. 5 represents a synthetic example of a hybrid system with dynamics of degree two.

The parameters determined using the above procedure are provided in Table 3 for all examples. The parameters $w = w_{size} = 10$ and $\lambda = 0.1$ are used for all examples.

5.4 Evaluation of Segment Clustering

For all presented results, changepoints are detected correctly in the trace segmentation step, thus, for the evaluation the generated clusters are compared to the ground truth represented by the system modes. For this comparison, we find a ground truth mode for every cluster. This is the most frequent ground truth mode observed for all segments of that cluster. A clustering is rated false whenever the mapping of a segment to a system mode deviates from the true system mode.

Traversals of a mode within one sampling step – which happens if multiple events are active at a time – are considered as one transition, i.e., modes visited for one sampling point are excluded.

The results of the evaluation of segment clustering for FaMoS (FaM) and HAutLearn (HAut) are provided in the left part of Table 4. A timeout of 30 minutes is set for each example; '–' indicates that a timeout happened. FaMoS with pure DTW-based clustering finishes within 30 minutes for all examples while HAutLearn finishes before the timeout only for two of five examples, the Simple Heating System and the Variable Heating System. In all examples, except the Variable Heating System, FaMoS's DTW-based approach achieves perfect clustering. For the Variable Heating System, segments of identical dynamics are not always similar according to DTW-based comparisons and, thus, are represented by different clusters. These clusters are merged using the LMI-based refinement. This demonstrates the utility of refining clusters using LMIs.

FaMoS' clustering has a faster execution time than HAutLearn. Apart from this execution time, additional time is needed to identify suitable parameters for the clustering step. For FaMoS more parameters have to be identified compared to HAutLearn. Still, the additional time needed for parameter tuning in FaMoS is less impactful than the slower iteration rate in HAutLearn.

Overall, FaMoS's DTW-based approach provides accurate results with very low execution time. Furthermore, the optional refinement using LMIs is especially powerful if similarities of segments in DTW-based comparison are not sufficient for a valid clustering.

5.5 Mode Prediction Performance for Anomaly Detection

A learned model of a hybrid system can be used for anomaly detection by comparing a trace predicted by the model, to the actual behavior of the system. To achieve a good anomaly detection accuracy the prediction has to be close to the nominal trace. We evaluate HAutLearn and FaMoS for a usage in this scenario by determining the conformance degree (T,J,τ_c,ϵ) of the predicted trace to a ground truth trace as formally introduced in [1]. T is the overall length of both traces, J is the number of changepoints of both traces, τ_c is the maximal timeshift between two associated changepoints and ϵ is the maximal deviation of trace values but allowing timeshifts up to τ_c. Both frameworks use the learned dynamics of the system modes to generate the trace prediction. FaMoS predicts system mode changes by applying the decision function of the learned decision tree model on the feature vector of a sample. HAutLearn predicts the next mode from the active event conditions and the current system mode using the learned hybrid automaton.

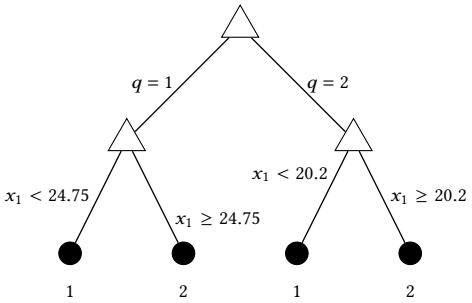
To provide some insights into the efficiency of both training methods the minimum required training set size relative to the number of all training samples needed to achieve the best prediction performance are provided in the right part of Table 4. For all examples except the Variable Heating System, the required training set size for the best prediction is lower for FaMoS than for HAutLearn. Furthermore, FaMoS achieves a similar or better prediction performance

Table 3. Parameter Valuations for all Examples

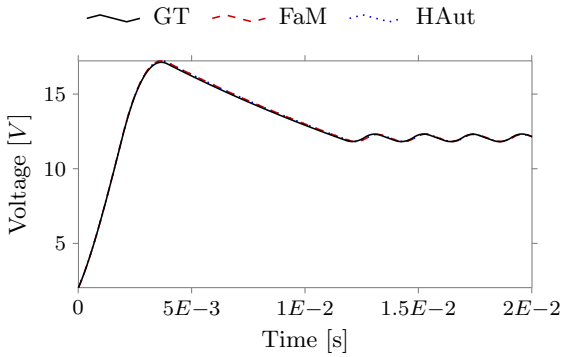
Example	General			Seg.	Clustering					Model Extraction		
	T in s	l	k	d_{max}	σ	k_c	α	$\rho_{min/max}$	d_c	η	γ	ψ
Simple Heating System	1E-2	1	0	3	1.5E-3	0	2.5	0.01, 0.1	0	10^5	10	2.7E-3
Variable Heating System	1E-2	1	1	3	2.5E-4	0	2.5	0.01, 0.1	0	10^5	10	2.7E-3
Two State Hybrid Automaton	1E-2	1	0	3	1.0E-7	1	7.5	0.05, 0.1	1	10^5	5	3.7E-3
Buck Converter	1E-5	2	0	1	2.0E-6	0	2.5	0.03, 1.0	0	10^5	10	2.0E-4
Complex Tank System	1E-2	3	0	3	4.0E-8	0	2.5	0.01, 1.0	0	10^5	10	0.0

Table 4. Evaluation Results

Example	Time Clustering			False Clusterings			Best Predictions						Min. Train Set		Learning Time	
	DTW FaM	DTW-LMI FaM	LMI HAut	DTW FaM	DTW-LMI FaM	LMI HAut	T	J	FaM τ_c	FaM ϵ	HAut τ_c	HAut ϵ	for best Prediction FaM	for best Prediction HAut	FaM	HAut
SimpHeatSys	15.1s	16.2s	384.2s	0	0	1	19.85	44	0.06	0.002	0.08	0.002	9%	14%	0.01s	5.3s
VarHeatSys	6.1s	6.4s	337.4s	64	0	29	19.98	23	0.02	0.007	0.02	0.007	52%	38%	0.01s	7.6s
TwoState	1.1s	127.9s	> 1800s	0	0	–	19.98	7	0.02	0.007	0.06	0.018	25%	27%	0.01s	5.0s
BuckConv	5.9s	31.0s	> 1800s	0	0	–	0.02	16	1.3E-4	0.001	1.3E-4	0.002	28%	34%	0.01s	10.0s
ComplexSys	4.2s	194.8s	> 1800s	0	0	–	19.98	18	0.11	0.013	2.98	0.187	52%	100%	0.01s	22.0s



(a) Learned Decision Tree of Simple Heating System



(b) Predicted Traces against Ground Truth of Buck Converter

Fig. 6. Comparison of FaMoS and HAutLearn Performance

in our examples. FaMoS' learning time is roughly 10ms for all examples. Fig. 6b shows the voltage variable of the predicted traces of both frameworks against the Ground Truth (GT) trace for the buck converter example. Both frameworks generate predictions that are close to the ground truth for this complex, practical example. Finally, Fig. 6a demonstrates the interpretability of the learned DT of the Simple Heating System. The thresholds associated with system mode changes are easily extractable.

Overall, these evaluation results show that FaMoS is a fast (R1) and accurate strategy for learning interpretable models (R2) of hybrid systems using small training datasets (R3), thus, fulfilling the previously defined requirements. FaMoS outperforms the HAutLearn framework in runtime and at least matches the HAutLearn framework in prediction accuracy for the presented systems.

6 DISCUSSION

The experimental evaluation in Section 5 shows the success of FaMoS in learning different types of hybrid systems with one or multiple variables and different dynamics. FaMoS offers further opportunities for scalability, because the sliding-window segmentation and the DTW-inspired clustering are easily parallelizable for multiple variables and derivatives.

Apart from adaptability, the main feature of FaMoS visible from the empirical results is its time-efficiency. In trace segmentation, FaMoS uses the sliding window approach which runs in $O(w)$, where w is the window width which is usually small. This sliding window evaluation is done for all sampling points in the traces. For clustering, DTW-inspired comparisons are used which have a complexity of $O(N^2)$, where N is the length of the shorter one of the two compared segments. Finally, DTL is used in model learning. Using the CART algorithm, the runtime is $O(n \cdot M \cdot \log(M))$, where

n is the number of features (being a constant for a given system) and M is the number of samples in a learning set [35].

FaMoS itself is not yet well adapted to real world data, especially very low sampling rates and noise. Derivatives of variables are calculated from the original traces which is only possible for no or limited noise. Currently, this limitation can be overcome by filtering signals or by observing derivatives directly.

Overall, FaMoS is a performant and accurate modeling strategy, supplementing existing learning strategies by a time-efficient alternative.

6.1 Comparison of Existing Strategies

The existing approaches provided in Table 1 showcase the diversity in model learners for hybrid systems. All of these approaches have different trade-offs between attributes like generalization, specificity, accuracy, complexity, learning time, interpretability, or flexibility.

FaMoS focuses on the requirements (R1) explainability, (R2) runtime efficiency, and (R3) learnability on small dataset sizes and, thus, builds a modeling strategy from algorithms respecting these requirements. Table 1 marks the bottlenecks of existing strategies with respect to the requirements in bold. For HAutLearn [45] and HyBUTLA [31], especially runtime limitations in the segment clustering and model extraction step, respectively, exist. Also POSEHEAD [34] and the DyClee-based [44] approach require comparably high runtime for model extraction and segment clustering, respectively. DyClee lacks interpretability, e.g., due to the usage of support vector regression for mode characterization. Interpretability is also a bottleneck of DENTA [17] which uses a deep network for trace segmentation. FaMoS' segmentation and clustering do not require large data set sizes and have a low runtime. The characterization of dynamics by difference equations achieves well-interpretable ARX models. Finally, DTL results in well-interpretable models and has fast learning algorithms.

In addition to the requirements (R1) to (R3), there are further trade-offs regarding generalization, flexibility, and accuracy. In trace segmentation, FaMoS, HAutLearn, and POSEHAD use very intuitive and flexible approaches, but HAutLearn's extremum-strategy does not generalize well for higher order dynamics. The wavelet analysis used in HyBUTLA is slightly less intuitive than extrema or a sliding-window approach, but still well interpretable and offers even better generalization. DENTA and the DyClee-based approach use more powerful segmentation strategies, thus, having good generalization, but larger computational efforts.

In segment clustering, LMI is an optimal strategy for dynamics that can be formulated as linear matrix equations, thus achieving high accuracy, but low generalization to other dynamics. HyBUTLA, DENTA and DyClee's clustering is interdependent with their segmentation strategies, thus loosing flexibility. DTW and DTW-inspired clustering used in POSEHEAD and FaMoS promises lowest learning time and complexity.

For mode characterization, mathematical models such as linear regression, polynomials, difference equations, or trees, used in all learners are well interpretable and have similar learning time. Machine learning strategies used in HyBUTLA and the DyClee-Based learner are less interpretable and often require more learning time.

FaMoS constructs a decision tree model, while the DyClee-Based approach builds heterogeneous petri nets and all other learners build hybrid automata structures. All types of models are well interpretable. Decision tree models offer further flexibility in model size and generalization on limited learning data.

7 CONCLUSION

Abstracted dynamical models of CPS are powerful tools for tasks like monitoring, testing, and verification. In this paper, we propose the data-driven model learning strategy FaMoS for hybrid system modeling. FaMoS provides a time-efficient and intuitive modeling procedure applicable for industrial applications involving online modeling or rapid prototyping. The learning process adapts a sliding window trace segmentation and DTW-inspired clustering. Furthermore, a new model extraction strategy and modeling paradigm with decision trees is implemented. We relate our approach to existing strategies for modeling of hybrid systems and discuss trade-offs in runtime efficiency, interpretability, generalization, and accuracy. In an empirical evaluation, FaMoS shows high runtime efficiency. The learned models are interpretable and have good accuracy also on limited learning data. Thus, FaMoS provides a competitive learning strategy which adds an alternative modeling procedure to the scope of model learning for hybrid systems targeting applications with limitations in runtime and requirements for interpretability of learned models.

ACKNOWLEDGMENTS

This research is partially funded by the two sources BMBF project AGenC no. 01IS22047A and (K)ISS [30] as part of dtec.bw® - Digitization and Technology Research Center of the Federal Armed Forces of Germany (BMVg) which we gratefully acknowledge. dtec.bw is funded by the European Union – NextGenerationEU.

REFERENCES

[1] Houssam Abbas, Hans Mittelmann, and Georgios Fainekos. 2014. Formal property verification in a conformance testing framework. In *2014 Twelfth ACM/IEEE Conference on Formal Methods and Models for Codesign (MEMOCODE)*. IEEE, New York, 155–164.

[2] Rajeev Alur. 2015. *Principles of Cyber-Physical Systems.* Technical Report. MIT Press.

[3] R. Alur, T.A. Henzinger, G. Lafferriere, and G.J. Pappas. 2000. Discrete abstractions of hybrid systems. *Proc. IEEE* 88, 7 (2000), 971–984. https://doi.org/10.1109/5.871304

[4] Fin Hendrik Bahnsen and Goerschwin Fey. 2019. Local Monitoring of Embedded Applications and Devices using Artificial Neural Networks. In *Euromicro Conference on Digital System Design (DSD)*. IEEE, New York, 485–491. https://doi.org/10.1109/DSD.2019.00076

[5] Nathalie Barbosa Roa, Louise Travé-Massuyès, and Victor H. Grisales-Palacio. 2019. DyClee: Dynamic clustering for tracking evolving environments. *Pattern Recognition* 94 (2019), 162–186.

[6] Omar Ali Beg, Houssam Abbas, Taylor T Johnson, and Ali Davoudi. 2017. Model validation of pwm dc–dc converters. *IEEE Transactions on Industrial Electronics* 64, 9 (2017), 7049–7059.

[7] Jürgen Beyerer and Oliver Niggemann. 2018. Machine Learning in Automation. *At-Automatisierungstechnik* 66, 4 (2018), 281–282. https://doi.org/10.1515/auto-2018-0036

[8] Mathias Blumreiter, Joel Greenyer, Francisco Javier Chiyah Garcia, Verena Klös, Maike Schwammberger, Christoph Sommer, Andreas Vogelsang, and Andreas Wortmann. 2019. Towards Self-Explainable Cyber-Physical Systems. *CoRR* abs/1908.04698 (2019), 6 pages.

[9] Aaron Bracht, Swantje Plambeck, and Goerschwin Fey. 2024. TUHH-IES/FaMoS-DT: v0.1. https://doi.org/10.5281/zenodo.10657936

[10] Michael S. Branicky. 2005. *Introduction to Hybrid Systems*. Birkhäuser Boston, Boston, MA, 91–116.

[11] Andreas Bunte, Benno Stein, and Oliver Niggemann. 2019. Model-Based Diagnosis for Cyber-Physical Production Systems Based on Machine Learning and Residual-Based Diagnosis Models. In *Conference on Artificial Intelligence (AAAI)*. QQQI, Hawaii, USA, 2727–2735.

[12] François E. Cellier. 1991. *Continuous system modeling*. Springer, New York.

[13] Fabio Cremona, Marten Lohstroh, David Broman, Edward A. Lee, and Michael Masin. 2019. Hybrid co-simulation: it's about time. *Software & Systems Modeling* 18 (2019), 1655–1679. Issue 3.

[14] Paul Cull, Mary Flahive, and Robby Robson. 2005. *Matrix Difference Equations*. Springer, New York, Chapter 7, 392.

[15] Dario Della Monica, Giovanni Pagliarini, Guido Sciavicco, and Ionel Eduard Stan. 2023. Decision Trees with a Modal Flavor. In *Advances in Artificial Intelligence (AIxIA)*, Agostino Dovier, Angelo Montanari, and Andrea Orlandini (Eds.). Springer-Verlag, Berlin, Heidelberg, 47–59.

[16] Mark Gold. 1967. Language identification in the limit. *Information and Control* 10, 5 (1967), 447–474.

[17] Nemanja Hranisavljevic, Alexander Maier, and Oliver Niggemann. 2020. Discretization of hybrid CPPS data into timed automaton using restricted Boltzmann machines. *Engineering Applications of Artificial Intelligence* 95 (2020), 103826. https://doi.org/10.1016/j.engappai.2020.103826

[18] Malte Isberner, Falk Howar, and Bernhard Steffen. 2014. The TTT Algorithm: A Redundancy-Free Approach to Active Automata Learning. In *Runtime Verification*. Springer International Publishing, Cham, 307–322.

[19] Pranav Srinivas Kumar, William Emfinger, and Gabor Karsai. 2015. A testbed to simulate and analyze resilient cyber-physical systems. In *2015 International Symposium on Rapid System Prototyping (RSP)*. IEEE, New York, 97–103. https://doi.org/10.1109/RSP.2015.7416553

[20] Edward A. Lee. 2016. Fundamental Limits of Cyber-Physical Systems Modeling. *ACM Transactions on Cyber-Physical Systems* 1, 1 (2016), 26 pages. https://doi.org/10.1145/2912149

[21] Scott Linderman, Matthew Johnson, Andrew Miller, Ryan Adams, David Blei, and Liam Paninski. 2017. Bayesian Learning and Inference in Recurrent Switching Linear Dynamical Systems. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 54)*, Aarti Singh and Jerry Zhu (Eds.). PMLR, Palermo, 914–922. https://proceedings.mlr.press/v54/linderman17a.html

[22] Changliu Liu, Tomer Arnon, Christopher Lazarus, Clark W. Barrett, and Mykel J. Kochenderfer. 2019. Algorithms for Verifying Deep Neural Networks. *CoRR* abs/1903.06758 (2019), 161 pages. arXiv:1903.06758

[23] Lennart Ljung. 1986. *System Identification: Theory for the User*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

[24] Wei-Yin Loh. 2011. Classification and regression trees. *WIREs Data Mining and Knowledge Discovery* 1, 1 (2011), 14–23. https://doi.org/10.1002/widm.8

[25] Estrella Lucena-Sanchez, Guido Sciavicco, and Ionel Eduard Stan. 2020. Symbolic Learning with Interval Temporal Logic: theCase of Regression. In *Workshop on Artificial Intelligence and Formal Verification, Logics, Automata and Synthesis (OVERLAY)*. CEUR, Bozen Bolzano, 5–10.

[26] J. Lunze and F. Lamnabhi-Lagarrigue. 2009. *Handbook of Hybrid Systems Control: Theory, Tools, Applications*. Cambridge University Press, Cambridge. https://books.google.de/books?id=pPLRV3ehMVIC

[27] Alexander Maier. 2014. Online Passive Learning of Timed Automata for Cyber-Physical Production Systems. In *International Conference on Industrial Informatics (INDIN)*. IEEE, Porto Alegre, Brazil, 60–66.

[28] Meinard Müller. 2007. *Dynamic Time Warping*. Springer-Verlag, Heidelberg, Chapter 4, 69–84.

[29] Oliver Niggemann and Volker Lohweg. 2015. On the Diagnosis of Cyber-Physical Production Systems - State-of-the-Art and Research Agenda. In *Twenty-Ninth Conference on Artificial Intelligence (AAAI-15)*. AAAI, Austin, Texas, USA, 8 pages.

[30] Oliver Niggemann and Stephan Myschik. 2022. DTEC - (K)ISS – Künstliche Intelligenz für die Diagnose der ISS. https://dtecbw.de/home/forschung/hsu/projekt-kiss.

[31] Oliver Niggemann, Benno Stein, Asmir Vodencarevic, Alexander Maier, and Hans Kleine Büning. 2012. Learning Behavior Models for Hybrid Timed Systems. *AAAI Conference on Artificial Intelligence* 26, 1 (2012), 1083–1090. https://doi.org/10.1609/aaai.v26i1.8296

[32] Oliver Niggemann, Stefan Windmann, Soeren Volgmann, Andreas Bunte, and Benno Stein. 2014. Using Learned Models for the Root Cause Analysis of Cyber-Physical Production Systems. In *DX*. Research Gate, Graz, 8 pages.

[33] Swantje Plambeck, Lutz Schammer, and Gorschwin Fey. 2022. On the Viability of Decision Trees for Learning Models of Systems. In *Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, New York, 696–701. https://doi.org/10.1109/ASP-DAC52403.2022.9712579

[34] Iman Saberi, Fathiyeh Faghih, and Farzad Sobhi Bavil. 2021. A Passive Online Technique for Learning Hybrid Automata from Input/Output Traces. *ACM Transactions on Embedded Computing Systems* 22, 1 (2021), 1–24. https://doi.org/10.1145/3556543

[35] Habiba Muhammad Sani, Ci Lei, and Daniel Neagu. 2018. Computational Complexity Analysis of Decision Tree Algorithms. In *International Conference on Artificial Intelligence*. Springer International Publishing, Cham, 191–197.

[36] Guido Sciavicco and Ionel Eduard Stan. 2020. Knowledge Extraction with Interval Temporal Logic Decision Trees. In *International Symposium on Temporal Representation and Reasoning (TIME)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 9:1–9:16.

[37] Shai Shalev-Shwartz and Shai Ben-David. 2014. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, Cambridge. https://doi.org/10.1017/CBO9781107298019

[38] Bernhard Steffen, Falk Howar, and Maik Merten. 2011. *Introduction to Active Automata Learning from a Practical Perspective*. Springer, Berlin, Heidelberg, 256–296. https://doi.org/10.1007/978-3-642-21455-4_8

[39] Henrik Steude, Alexander Windmann, and Oliver Niggemann. 2022. Learning Physical Concepts in CPS: A Case Study with a Three-Tank System. *IFAC-PapersOnLine* 55, 6 (2022), 15–22. https://doi.org/10.1016/j.ifacol.2022.07.099

[40] Paulo Tabuada, Sina Yamac Caliskan, Matthias Rungger, and Rupak Majumdar. 2014. Towards Robustness for Cyber-Physical Systems. *IEEE Trans. Automat. Control* 59, 12 (2014), 3151–3163. https://doi.org/10.1109/TAC.2014.2351632

[41] The MathWorks Inc. 2023. MATLAB (R2023b).

[42] Charles Truong, Laurent Oudre, and Nicolas Vayatis. 2020. Selective review of offline change point detection methods. *Signal Processing* 167, 4 (2020), 20 pages. https://doi.org/10.1016/j.sigpro.2019.107299

[43] Henning Urbat and Lutz Schröder. 2020. Automata Learning: An Algebraic Approach. In *ACM/IEEE Symposium on Logic in Computer Science*. ACM, New York, 900–914. https://doi.org/10.1145/3373718.3394775

[44] Amaury Vignolles, Elodie Chanthery, and Pauline Ribot. 2022. Hybrid Model Learning for System Health Monitoring. In *Symposium on Fault Detection, Supervision and Safety for Technical Processes (SAFEPROCESS)*. Science Direct, Pafos, 7–14.

[45] Xiaodong Yang, Omar Ali Beg, Matthew Kenigsberg, and Taylor T. Johnson. 2022. A Framework for Identification and Validation of Affine Hybrid Automata from Input-Output Traces. *ACM Transactions on Cyber-Physical Systems* 6, 2 (2022), 1–24. https://doi.org/10.1145/3470455