WORKSHOP REPORT: LEARNING APPROACHES FOR HYBRID DYNAMICAL SYSTEMS

Swantje Plambeck[1], **Nemanja Hranisavljevic**[2], **Maximilian Schmidt**[1], **Kaja Balzereit**[3], **Aaron Bracht**[1], **Magnus Redeker**[4], **Negar Arabizadeh**[5], **Alexander Diedrich**[2], **Jens Eickmeier**[6], **Oliver Niggemann**[2], **and Goerschwin Fey**[1]
[1]Institute of Embedded Systems, Hamburg University of Technology, Hamburg, Germany
[2]Institute of Automation Technology, Helmut Schmidt University, Hamburg, Germany
[3]Bielefeld University of Applied Sciences and Arts, Bielefeld, Germany
[4]Fraunhofer IOSB-INA, Lemgo, Germany
[5]Karlsruhe Institute of Technology, Vision and Fusion Laboratory (IES), Karlsruhe, Germany
[6]SOLLICH KG, Bad Salzuflen, Germany

ABSTRACT

This report summarizes the workshop on "Learning Approaches for Hybrid Dynamical Systems", held at the 2025 Conference on Machine Learning for Cyber-Physical Systems (ML4CPS). The workshop aimed to strengthen collaboration and foster exchange between institutions engaged in research on model learning methods for hybrid CPSs.

The participating research groups approach the topic from diverse perspectives, for example, from an application perspective, from a tool perspective, or from a fundamental and formal perspective. Accordingly, this paper synthesizes the discussions from the workshop and presents an overview of key perspectives on several central topics, including the taxonomy of hybrid systems, current learning paradigms and techniques, and particularly representative use cases.

Keywords Cyber-Physical Systems, Hybrid Dynamical Systems, Model Learning, Model Inference, Hybrid Automata, Model Interoperability

1 Introduction

By integrating discrete and continuous dynamics, hybrid systems can effectively represent real-world systems in various domains, including production engineering [20], biological applications [4], or automotive technology [3]. As a result, they have emerged as a widely recognized and important modeling formalism.

The wide range of applications has led to the development of numerous modeling approaches and formalisms for describing hybrid systems. This diversity fosters open discussion and exchange within the community regarding differing perspectives on hybrid systems and their modeling. The workshop "Learning Approaches for Hybrid Dynamical Systems", held at the 2025 Conference on Machine Learning for Cyber-Physical Systems (ML4CPS), leveraged the expertise of individual research groups to bring together these diverse perspectives and approaches. This paper summarizes the key discussions from the workshop and offers an overview of current viewpoints as well as initial ideas for addressing open challenges in the field.

This article is organized around several key topics discussed during the workshop:

1. **Taxonomy of hybrid systems**: As a first step, a common taxonomy is developed to establish a shared understanding across different perspectives on hybrid systems. The taxonomy is structured around key aspects, including the representation of continuous dynamics, the representation of discrete dynamics, and the modeling of transitions between discrete modes.
2. **Learning of hybrid dynamical systems**: The second topic addresses the learning of hybrid dynamical systems, distinguishing different learning sub-steps and highlighting the substantial challenges posed by inherent complexity and the mixed nature of discrete and continuous dynamics.
3. **Real-world use cases**: An additional contribution is the presentation of two real-world use cases. For one, the observed data are publicly available, while the second – although the data are confidential – represents a complex, real-world CPS with several unresolved challenges.
4. **Integration into industry**: Finally, the discussion turns to asset administration shells, which serve as model characterizations that enable the integration of machine learning models into industrial practice.

2 Topics Covered

2.1 Taxonomy of Hybrid Dynamical Systems

Initial Framing and Understanding

This section explores modeling and learning approaches for hybrid dynamical systems, determining a fundamental taxonomy that represents existing methodologies and applications and offers insights into open topics and new directions in the research field. For this, the taxonomy covers different aspects of the learning process individually, which are the system, the learner, and the model. The system is the real-world system that is to be modeled, the learner is the algorithm that is used to learn the model, and the model is the abstract representation of the system.

System to Model: Abstraction and Preprocessing

Often, a system under learning is considered a black box, and the learning process aims to derive a detailed representation and understanding of the inner workings of the system. We might assume that the system is a hybrid automaton and search for the corresponding hybrid automaton model. This representation is usually an abstraction of the real-world system. Such an abstraction involves, for example, the following aspects:

- Discretization: While the system is continuous, the model or part of the model might be discrete [21], [22]. For hybrid automata, these discrete parts are the modes and transitions. Techniques like Variational Autoencoders (VAE) [18, 21], Boltzmann machines, Principal Component Analysis (PCA), Clustering [26], and Self-Organizing Maps (SOMs) [21], are possible options for performing a discretization.
- Unobservable Variables: The system might have unobservable variables that influence the output of the system. These variables should be considered in the model.
- Inputs & Outputs: The signals of the system need to be mapped to the inputs and outputs of the model or defined as such. In many cases, this mapping is straightforward, but in some cases, the searched model might require a different mapping.
- Modeling of Subsystems: The model might only consider parts of the system. This also includes leaving out real-world effects such as noise or other disturbances.
- Maturity of the Model: The model might be more or less mature or efficient. Maturity quantifies, for example, the ability of the system to estimate unobservable states or to

steer the output signal (through the input signal) to reach and stabilize at its desired value, even in the presence of noise and disturbances [2].

- Further Preprocessing: Any kind of preprocessing, such as de-noising, filtering, and feature extraction, contributes to an abstraction between the system and the model [16].

While there exist various techniques to perform these abstractions and examples of their applications, there are still several new methods as well as combinations of existing methods to be explored. A relevant aspect discussed in this regard is the re-usability of models across different applications. From a domain-overlapping perspective, this can be the unification of models that serve specific purposes, e.g., all predictive models, or all models that are used for testing should be based on similar modeling paradigms.

Considering a fixed domain and application, model re-usability can be achieved by altering the inputs and outputs of the model. Thus, instead of considering a model as a fixed direction for the transfer of signals, the model is a flexible tool where all external signals can serve both for input and output. This is a challenging approach, especially since the behavioral representation of the model is invertible.

A further challenge are models for explanation. These models might need to be altered on the basis of the explanations sought. This goes beyond the mere alteration of inputs and outputs, as the representation of inputs and outputs of the model might need to be adapted as well.

System to Learner: Complexity Dimensions

The relation between system and model is further influenced by the learner, which usually defines which type of model is learned, e.g., a hybrid automaton, a neural network, or a decision tree. The learner is the interface between the system and the model. Thus, the learner on the input side must match the complexity of the system and the provided learning data, and on the output side must provide a model that is able to represent the system. We first discuss the complexity dimensions of the system and then continue with the type of model that is learned.

The complexity of a system is hard to quantify as it is influenced by various factors [6]. The following list gives an overview of the most important dimensions that influence the complexity of a system, while we consider a hybrid automaton representation of the system:

- Number of Variables: The number of variables that are considered in the system, including inputs, outputs, and states.
- Set of Possible Flow Functions: The set of possible flow functions that can be used in the system. This set can be restricted, e.g., to linear differential equations or polynomial functions. A further subdimension is, for example, the order of the differential equation or polynomial function.
- Number of Modes: The number of modes that are considered in the system. This can be a single model or a multi-model structure. Often, there is a trade-off between the number of modes and the complexity of the flow functions. If the flow functions are simple, more modes are needed to represent the system. In case of complex flow functions that might represent jump conditions, this could aggregate multiple modes in a single one [23].
- Conditions and Jumps: The conditions and jumps in the transition events. Transition events can be defined in various ways, e.g., as a condition on the state variables or as a stochastic event. The complexity of the system is influenced by the number of conditions and jumps that are considered, but also by the complexity of these conditions and jumps.
- Stochasticity: As mentioned with the transition events, these might be stochastic. In practice, such determinism might be caused by noise, uncertainty, or unobservable variables. Often, a stochastic system is considered more complex because the behavior is harder to predict. However, introducing stochasticity can also simplify the model, as it might be easier to represent the system with fewer modes, simpler flow functions,

or fewer conditions. A maturity quantification of the model could give insights in the ability to achieve modeling goals under stochastic conditions [2].

The final dimensions of this list of complexity dimensions suggest that the handling of noise and uncertainty is a particular challenge in the learning process. Especially for continuous dynamic, a complete coverage of the state space on final learning data sets is impossible. Thus, the model must be able to generalize from the data that is available, but at the same time, it must be able to handle the noise or ambiguity that is present in the system. In the optimal case, the model is able to distinguish between noise and real system behavior and to provide uncertainty estimates for the model itself.

A complexity dimension that goes beyond single-system models is multi-model structures. Again, a multi-model structure is often considered more complex, as it requires the coordination of multiple models. Nevertheless, a multi-model structure can also simplify the representation of the system, as it allows for a more modular representation of the system. This can be especially useful if the system is composed of multiple subsystems that can be represented by different models. These models might even be of different types to specifically adapt to the subsystem that is modeled.

Learner to Model: Expressiveness of the Result

In model learning for hybrid dynamical systems, the learner produces a model that captures the behavior of the underlying system. Typically, the type of model is predetermined by the learning method. Depending on the learning strategy, the model can be learned directly from the data or inferred from the observed behavior of the system [23].

Learning directly from data constructs model components such as system modes, continuous dynamics, and transitions based on input-output traces or time-series data. In contrast, inferring a model from behavior involves reasoning about system structure and dynamics based on observed properties or execution traces, often guided by prior knowledge or structural constraints [23].

Suppose that the system is represented by a hybrid automaton, the learning process can take several forms. The hybrid automaton may be learned explicitly: its discrete modes, flow functions, and transition logic are directly extracted from the data. Alternatively, the model may be only partially explicit. For example, the mode structure might be identified from the data, while the continuous dynamics (i.e., flow functions) are approximated using regression-based methods such as neural networks or symbolic regression [19]. This introduces a dependency on the abstraction between the physical system and its model representation. When the predefined model class restricts the allowable flow functions, it may necessitate approximating or abstracting the original dynamics to fit the target model.

A further approach involves using purely data-driven regression models to capture system behavior without constructing an explicit hybrid automaton. In such cases, methods with high representational capacity—such as neural networks, kernel methods, or support vector machines—can effectively approximate system trajectories, even if the resulting model lacks an interpretable hybrid structure.

The degree to which a model is explicit, partially explicit, or inferred has significant implications for its expressiveness, interpretability, and explainability. When the learned model mirrors the hybrid automaton structure of the true system, it can offer insights into system dynamics, such as physical laws and variable dependencies—features that are valuable for applications like system diagnosis. In contrast, non-explicit models are often simpler to train and may offer advantages in terms of computational efficiency.

Explicit learning of hybrid automata also offers structural modularity: the model consists of three interconnected elements: (1) discrete modes, (2) continuous flow functions, and (3) transition relations. This separation allows different learning techniques to be applied to each component, offering high flexibility. The benefits of this modular pipeline approach are discussed further in Section 2.2.

Finally, the learner and the selected model class also define the model complexity dimensions. Some of these dimensions are externally defined (e.g., the class of admissible flow functions),

while others must be inferred during learning (e.g., the number of modes). The learner must be capable of adapting to the true complexity of the system, which can be particularly challenging if the complexity is unknown a priori or lies outside the expressiveness of the chosen model class.

2.2 Learning Process for Hybrid Systems

Initial Framing and Understanding

Learning hybrid automata is a challenging task due to the interplay between discrete and continuous dynamics. In the literature, the learning process is often structured into modular steps to incrementally build a comprehensive model of the system. Figure 1 illustrates a commonly adopted learning pipeline, which consists of the following steps:

1. **Trace Segmentation**
   Identification of mode switches in observed system trajectories.
2. **Segment Clustering**
   Grouping of segments that exhibit similar dynamic behavior, assumed to correspond to identical dynamical modes.
3. **Mode Characterization**
   Derivation of continuous flow functions that describe the behavior within each identified mode.
4. **Transition Identification**
   Determination of transition conditions or guard predicates that govern the switching behavior between modes.
5. **Model Inference**
   Integration of the identified components—modes, transitions, and flow functions—into a unified hybrid automaton model.

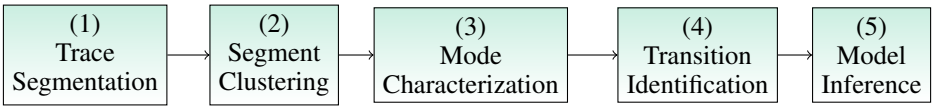

Figure 1: Standard learning process for hybrid automata

This modular formulation allows flexibility in selecting different techniques at each stage, as summarized in Table 1. Some methods—such as Self-Organizing Maps or Decision Trees—can be employed at multiple steps, promoting tighter integration and efficient information flow across the learning pipeline. To select a suitable technique, maturity quantification is a possible base to compare the efficiency of different techniques used in the learning process (such as their ability to steer the output signal within a specific error margin, estimate unobservable variables with a certain precision, or ultimately achieve the desired product quality after implementing the learned model) [2].

Although this structure is widely adopted, it implicitly assumes a linear and decoupled progression through the steps. However, this assumption is not without limitations. In particular, the segmentation of traces is often performed without knowledge of the actual system dynamics, which contradicts the very rationale for identifying distinct modes—namely, to represent distinct types of dynamical behavior that cannot be captured by a single flow function.

Current approaches frequently use purely statistical or geometric criteria for segmentation, such as Euclidean distance or changepoint detection, which may not align with the true dynamical boundaries of the system. This can lead to an unnecessary large number of segments and modes, resulting in models that are more complex than required and potentially harder to interpret, generalize, or verify.

Trace Segmentation	Euclidean Distance, Self-Organizing Maps, Changepoint Detection, . . .
Segment Clustering	Dynamic Time Warping, Self-Organizing Maps, Clustering, . . .
Mode Characterization	Linear Regression, Self-Organizing Maps, Linear Matrix Inequalities, . . .
Transition Identification	Condition Mining, Rule-Based Systems, Decision Trees, . . .
Model Inference	Automata Learning, Decision Trees, Petri Net Construction, . . .

Table 1: Techniques Applied in the Learning Process

Revisiting the Learning Pipeline: Toward Model-Guided Segmentation

A key insight is that multiple modes are only necessary when a single flow function cannot sufficiently capture the observed behavior. Therefore, we argue that trace segmentation should not be an independent, upstream step but rather be informed by the process of mode characterization. That is, segmentation decisions should be governed by the expressiveness of the modeling technique used to characterize each mode.

Concretely, instead of predefining mode boundaries, one could attempt to model the entire trace using a candidate flow function (e.g., linear regression, neural networks, etc.). Where the modeling error becomes unacceptably high—indicating that the current function cannot represent the underlying behavior—a segmentation point is introduced. This approach couples segmentation with model expressiveness, leading to models that introduce new modes only when necessary.

This shift from purely data-driven segmentation to model-informed segmentation would reduce unnecessary model complexity. Moreover, it opens the possibility for iterative refinement, where segmentation and mode characterization co-evolve through successive approximations, improving both segmentation accuracy and overall model quality.

Implications and Future Directions

This perspective challenges the rigid step-by-step pipeline by advocating for integrated and feedback-driven learning processes. For example, segment clustering can be refined based on insights from mode characterization; similarly, transition conditions might inform whether two clusters should be merged or split. Such iterative and adaptive processes stand in contrast to the traditional linear approach and better reflect the intertwined nature of discrete and continuous system behavior.

Future work should explore algorithmic frameworks that implement this vision—integrating model expressiveness into early-stage segmentation, and promoting end-to-end learning pipelines with dynamic feedback loops between stages. By making segmentation contingent on representational limits, we expect to achieve models that are both simpler and more faithful to the underlying system dynamics.

2.3 Use cases and datasets

Initial Framing and Understanding

Two specific use cases were examined to stimulate discussion, highlight key challenges, and explore preliminary solution approaches.

Conveyor System from SF-OWL

The conveyor system is part of the high-rack storage system from SmartFactoryOWL, Lemgo, Germany. It consists of six belt conveyors (the state as in year 2016 when the data were recorded) [5]. The dataset is available online at `https://www.kaggle.com/datasets/inIT-OWL/high-storage-system-data-for-energy-optimization`.

The system consists of four horizontal conveyors: LH1, LH2, RH1 and RH2 and two vertical conveyors: LV and RV, which move LH2 and RH2, respectively. While the motion of vertical conveyors is controlled to achieve the set-point vertical position, horizontal conveyors stop (and run) based on the binary presence sensors detecting the horizontal position of an item. There are four operations (movements) that the conveyor system can perform (see Figure 2).

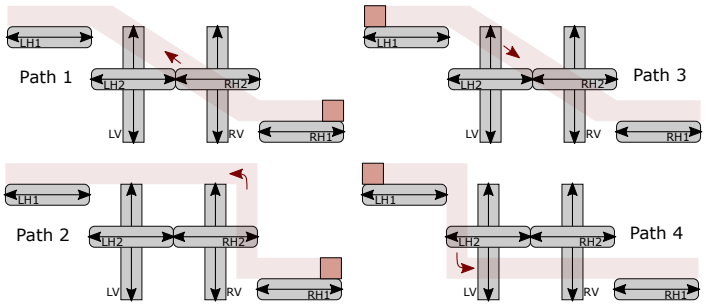

Figure 2: The conveyor system from SmartFactoryOWL.

This is a hybrid system consisting of 6 discrete events and 24 continuous variables. Event variables: the electrical drives controlling the conveyors receive from the PLC different commands which can take one of the two (in case of vertical conveyors) or three values (in case of the horizontal conveyors). The 24 continuous variables represent position, power, voltage, and current of each of the six conveyors. In total, 49830 observations in two files (sequences) are recorded during periods of 16.3 and 35.5 minutes, respectively. In both files, the system was alternating the following sequence of operations: Path1, Path3, Path2, Path4, Path1, Path3,...

Columbus ECLSS dataset – project KISS ([1])

The Environmental Control and Life Support System (ECLSS) is a CPS on board the COLUMBUS module of the International Space Station (ISS). ECLSS is tasked with maintaining habitable conditions for astronauts through a number of valves, heat exchangers, pipes, etc. and multiple control loops. At the same time, it transmits telemetry data to an Earth-based location at one second intervals to facilitate operational support. The support provided by a human operator can react to different situations and, if necessary, can reconfigure the system in order to continue the system's functionality. As this process is currently based on the manually defined rules and the operator's understanding of the system state based on the telemetry data, an AI-based assistance could be highly beneficial.

The reconfiguration problem requires an algorithmic solution that can be executed at some point in time and return a set of recommended reconfiguration possibilities. The input to the reconfiguration algorithm is:

- a set of configuration constraints that constrain a subset of variables to have specific values (for example, a variable must be 0); and,
- observed data before the reconfiguration is triggered (e.g., during last 15 minutes).

The historical data to train the system model (available in the *KISS* project) is a substantial telemetry data set that covers observations of several hundreds of variables over 6 years, from the beginning of 2018 to the end of 2023.

The identified challenges are the following.

- Reconfiguration: The workshop explored the unique needs associated with the application of the hybrid system model in reconfiguration scenarios.
- Within the dataset, configurations are not uniformly represented, with the system predominantly functioning in a limited set of common configurations.
- Many valid configurations are absent from the dataset, for example, redundancy scenarios that are never used. Key question: Can we infer the behavior of the never-occurring modes?
- The system appears to be considerably complex due to numerous variables, hidden variables, and the extensive interaction between its discrete and continuous components [6].
- A very interesting challenge would be the creation of a common model that can be used for anomaly detection, diagnosis, and system reconfiguration. Currently, the KISS project proposes three distinct models for these tasks [14, 15, 17, 25].

2.4 Interoperable ML datasets and applications

The concept of the Asset Administration Shell (AAS) plays a pivotal role in the realm of Industrie 4.0 (I4.0), which is characterized by the intelligent interconnection of machines and processes through information and communication technology, [10, 12, 13]. As traditional value chains evolve into flexible, dynamic, and globally connected value networks, AAS facilitates the implementation of digital twins and ensures interoperability among the solutions of various suppliers.

A specific submodel within the AAS framework is the Artificial Intelligence (AI) Dataset submodel, [7], which is designed to manage and standardize information pertaining to datasets used in AI applications. The AI Dataset submodel finds its utility in various use cases, particularly in documenting datasets to simplify their reuse by providing standardized access to crucial meta-information, such as the conditions under which the data was generated. Moreover, the submodel enriches the management of the dataset by documenting static parameters such as mean values and data formats, as well as dynamic parameters such as environmental conditions affecting data collection, like humidity and temperature. In terms of functionality, the AI Dataset submodel allows data scientists and AI experts to access comprehensive dataset information and select suitable data for AI training based on provided metrics. It also supports the investigation of data-related issues and facilitates easy communication between users and creators through detailed contact information.

Beyond, the Artificial Intelligence Model Nameplate submodel provides a standardized structure for documenting AI models, facilitating efficient management and reuse of trained models while ensuring clear communication of model properties among stakeholders, [9], the Artificial Intelligence Deployment submodel focuses on the standardized deployment of AI models, particularly in industrial settings, enabling easier usage and management of the AI lifecycle, [8], and within the Capability submodel the capabilities and requirements of AI services can be described, [11].

For example, the integration of data-driven services in manufacturing environments is increasingly essential to improve efficiency and optimize operations. As shop floors become populated with several thousand machines, managing these machines and their corresponding value creation processes becomes overwhelmingly complex for humans. This complexity makes it challenging to track which machines are benefiting from existing data-driven services, how these services can be extended to additional machines, and whether different data-driven services can be combined to create greater value for specific machines.

To address these challenges effectively, a data-driven service can be equipped with an AAS that specifies its capabilities, the data required to utilize those capabilities, and the data or value that the service generates for machines, processes, or products. As depicted in Figure 3, an automatization of the matching of machines and data-driven services enables manufacturers to make informed decisions without being overwhelmed by the sheer volume of machines and services [24]. Ultimately, machines and data-driven services equipped with interoperable AASs

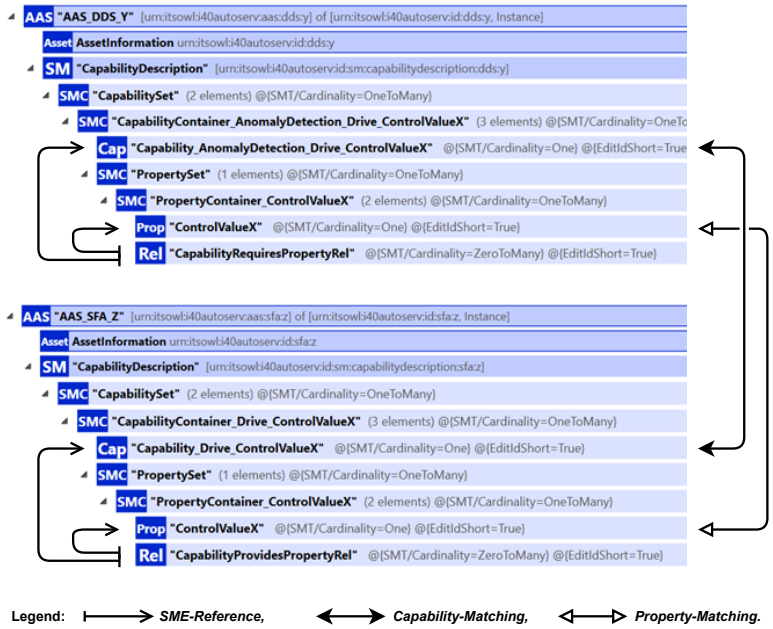

Figure 3: The matching of data-driven services (DDSs) and shop-floor assets (SFAs) like machines: Matching of their capabilities as well as required and provided properties based on AAS-specified capability descriptions. © Fraunhofer IOSB-INA.

not only enhance operational efficiency but also drive greater value creation on the shop-floor in a future-proof manner.

3 Conclusions and Outlook

In this paper, we dicuss the challenges and opportunities in learning hybrid dynamical systems. The development of learning approaches for hybrid dynamical systems is a crucial area of research. By properly addressing abstraction techniques, model complexities, and learning processes, we significantly enhance the understanding and application of these systems. We present a taxonomy that categorizes different modeling approaches and highlights the importance of understanding the underlying characteristics of the system, the learner, and the model. We also propose a comprehensive learning procedure for hybrid systems, which consists of five steps: trace segmentation, segment clustering, mode characterization, transition identification, and model inference. This modular approach allows for the application of various techniques at each step, enabling efficient information transfer and iterative refinement of the model. Further, we introduce a real-world data set from a complex Cyber-Physical System (CPS) and outline the learning process, emphasizing the need for flexibility and adaptability in extracting information from data. We discuss asset administration shells as a tool to put machine learning models into practice. With this report of the workshop on ”Learning Approaches for Hybrid Dynamical Systems” we, thus, present a diverse view on the current state of the art as well as open challenges and opportunities.

Acknowledgments

The authors thank the participants of the workshop for their valuable contributions and discussions. In particular, we would like to thank the organizers of the ML4CPS conference for providing a platform for this workshop.

References

[1] DTEC - (K)ISS – Künstliche Intelligenz für die Diagnose der ISS, 2022.

[2] N. Arabizadeh, J. Pfrommer, and J. Beyerer. How to quantify the maturity of production processes. In *ML4CPS–Machine Learning for Cyber-Physical Systems*. UB HSU, 2025.

[3] M. S. Branicky. Introduction to hybrid systems. In *Handbook of networked and embedded control systems*, pages 91–116. Springer, 2005.

[4] R. Ghosh and C. Tomlin. Symbolic reachable set computation of piecewise affine hybrid automata and its application to biological modelling: Delta-notch protein signalling. *IEE Proceedings-Systems Biology*, 1(1):170–183, 2004.

[5] N. Hranisavljevic, A. Maier, and O. Niggemann. Discretization of hybrid CPPS data into timed automaton using restricted Boltzmann machines. *Engineering Applications of Artificial Intelligence*, 95:103826, 2020.

[6] N. Hranisavljevic, T. Westermann, S. Plambeck, H. S. Steude, G. Benndorf, and O. Niggemann. A model learning perspective on the complexity of cyber-physical systems. In *Conference on Machine Learning for Engineering (ML4CPS)*, 2025.

[7] Industrial Digital Twin Association e. V. Artificial Intelligence Dataset Submodel - IDTA Number: 02058.

[8] Industrial Digital Twin Association e. V. Artificial Intelligence Deployment Submodel - IDTA Number: 02059.

[9] Industrial Digital Twin Association e. V. Artificial Intelligence Model Nameplate Submodel - IDTA Number: 02060.

[10] Industrial Digital Twin Association e. V. Asset Administration Shell Reading Guide.

[11] Industrial Digital Twin Association e. V. Capability Description Submodel - IDTA Number: 02020.

[12] Industrial Digital Twin Association e. V. Specification of the Asset Administration Shell Part 1: Metamodel – IDTA Number: 01001-3-0-2.

[13] Industrial Digital Twin Association e. V. Specification of the Asset Administration Shell Part 2: Application Programming Interfaces – IDTA Number: 01002-3-0-3.

[14] B. Kelm, K. Balzereit, L. Moddemann, S. Myschik, and O. Niggemann. Application of a Model-based Reconfiguration Approach for the ISS COLUMBUS Environmental Control and Life Support System (ECLSS). In *33rd International Workshop on Principle of Diagnosis – DX 2022*, Sept. 2022.

[15] B. Kelm, K. Balzereit, S. Myschik, and O. Niggemann. Qualitative Graph-based Reconfiguration for Serially Constrained Hybrid Systems. In *34th International Workshop on Principles of Diagnosis (DX'23)*, Sept. 2023.

[16] M. Knitt, S. Plambeck, J. C. Wieck, J. Kohlisch, S. Balduin, E. M. Veith, J. Schyga, J. Hinckeldeyn, G. Fey, and J. Kreutzfeldt. Towards the automatic generation of models for prediction, monitoring, and testing of cyber-physical systems. In *2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–4, 2023.

[17] L. Moddemann, H. Steude, A. Diedrich, and O. Niggemann. *Discret2Di - Deep Learning based Discretization for Model-based Diagnosis*. Nov. 2023.

[18] L. Moddemann, H. S. Steude, A. Diedrich, I. Pill, and O. Niggemann. Extracting knowledge using machine learning for anomaly detection and root-cause diagnosis. In *2024 IEEE 29th International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8, 2024.

[19] O. Niggemann, B. Stein, A. Vodencarevic, A. Maier, and H. Kleine Büning. Learning behavior models for hybrid timed systems. *Proceedings of the AAAI Conference on Artificial Intelligence*, 26(1):1083–1090, Sep. 2021.

[20] O. Niggemann, B. Stein, A. Vodenčarević, A. Maier, and H. Kleine Büning. Learning Behavior Models for Hybrid Timed Systems. In *Twenty-Sixth Conference on Artificial Intelligence (AAAI-12)*, pages 1083–1090, Toronto, Ontario, Canada, 2012.

[21] P. J. Overlöper, L. Moddemann, N. Hranisavljevic, A. Windmann, and O. Niggemann. Discretization of CPS Time Series with Neural Networks. In *IEEE International Conference on Emerging Technologies and Factory Automation*, Padova, 2024.

[22] S. Plambeck and G. Fey. Decision tree models of continuous systems. In *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8, 2022.

[23] S. Plambeck, M. Schmidt, A. Subias, L. Travé-Massuyès, and G. Fey. Usability of Symbolic Regression for Hybrid System Identification - System Classes and Parameters. In I. Pill, A. Natan, and F. Wotawa, editors, *35th International Conference on Principles of Diagnosis and Resilient Systems (DX 2024)*, volume 125 of *Open Access Series in Informatics (OASIcs)*, pages 30:1–30:14, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[24] M. Redeker, D. Quirin, R. Schroeder, T. Klausmann, A. Löwen, A. Wollbrink, H. Stichweh, S. Althoff, A. Bender, W. Sextro, and M. Hesse. Towards a One-Stop-Shop Solution for the Application of Data-Driven Value-Adding Services in Production. In *IEEE ETFA*, 2024.

[25] H. S. Steude, C. Geier, L. Moddemann, M. Creutzenberg, J. Pfeifer, S. Turk, and O. Niggemann. End-to-end MLOps integration: a case study with ISS telemetry data. In *ML4CPS – Machine Learning for Cyber-Physical Systems*. Helmut-Schmidt-Universität Hamburg, 2024.

[26] A. Vignolles, E. Chanthery, and P. Ribot. Hybrid model learning for system health monitoring. *IFAC-PapersOnLine*, 55(6):7–14, 2022.