HYBRID ONLINE TIMED AUTOMATON LEARNING ALGORITHM FOR DISCRETE MANUFACTURING SYSTEMS

Simon Martens
Bielefeld University of Applied Sciences
simon.martens@hsbi.de

Alexander Maier
Bielefeld University of Applied Sciences
alexander.maier@hsbi.de

Alexander Wunn
Haver & Boecker OHG
a.wunn@haverboecker.com

ABSTRACT

This paper presents the Hybrid Online Timed Automaton Learning Algorithm (HyOTALA), a novel approach for anomaly detection in cyber-physical production systems (CPPS). It addresses the challenge of using hybrid data, combining sparse discrete and continuous signals, by learning a hybrid timed automaton model in an online setting. This model captures the dynamics of discrete manufacturing processes and provides significant advances in model identification for CPPS. The effectiveness of HyOTALA is demonstrated through a practical application, highlighting its potential to improve anomaly detection capabilities in industrial settings.

Keywords Cyber Physical Production Systems · Model Identification · Hybrid Timed Automata · Online · Unsupervised · Anomaly Detection

1 Introduction

Anomaly detection and predictive maintenance in cyber-physical production systems (CPPS) are important tasks in many industrial applications, as they can help identify and resolve problems in production processes. However, there are several challenges that need to be overcome to ensure a successful application [10].

When building models that can be used for anomaly detection, one of the biggest problems is that discrete PLC (programmable logic controller) data often goes unused.

Yet this data can provide valuable information about the health of a machine. Instead, the focus is on data from sensors, sometimes installed specifically for this purpose [25]. This may be due to a lack of awareness of methods tailored to anomaly detection in discrete production systems. Instead, classical time series algorithms are usually used, which do not always give the best results.

One reason why the use of discrete PLC data is crucial, is mode switching. Mode switching is responsible for continuous signal non-linearity and is highly correlated with discrete PLC data. An example is a gear change that results in an unstable motor speed. However, learning this dynamic is complicated without proper embedding of discrete signals. Even if discrete signals are embedded in a black box model, there is no guarantee that the characteristic will be learned correctly [11].

Another reason underpinning the importance of proper mode identification, is the temporal behaviour of a CPPS. In particular, explicit modelling of the transition time between modes is a powerful health indicator. A worn cutting tool, such as a blade, will take longer to cut through material. Or a worn conveyor belt will take longer to transport. Identifying these types of anomalies is low hanging fruit if discrete signals are properly processed.

A major challenge that we discuss in this paper is how to deal with hybrid data while ensuring that the above and other features are learned. A hybrid learning algorithm can handle sparse discrete data (e.g. digital valve states) and continuous data (e.g. analog engine speed).
To address this challenge, we've developed a new algorithm called the Hybrid Online Timed Automaton Learning Algorithm (HyOTALA). It learns a hybrid timed automaton model in an online setup.

In the paper, we'll first look at the current state of the art and explain the research gap in section 2. Then, in section 3, we'll take a deep dive into how HyOTALA works. After that, we'll evaluate the practical application of HyOTALA in section 4. Finally, we'll summarise our findings in section 5.

2 State of the Art

This section presents methods for identifying models that can be used for anomaly detection. While popular machine learning models may not always be the ideal choice, lesser-known explicit models offer a good alternative. This idea is further explored in the subsection 2.1. Further on, we will focus on (hybrid) timed automata as a formalism and explore methods for learning them. These methods will then be categorised according to the type of signals they can handle. Methods for identifying discrete models can be found in subsection 2.2, while methods for continuous and hybrid models, which are a combination of both types, are discussed in 2.3. Finally, the research gap of this paper is explained in 2.4.

2.1 Underutilization of Explicit Models

Modern machine learning methods show promising results on (research) data sets. Sequence-based models such as RNNs, LSTMs, or transformers are theoretically able to learn contextual, sequential, and temporal features of CPPS [17], [24].
However, there are practical limitations, such as the availability of large datasets for training complex models. It is also difficult to interpret and discuss black box models. In general, phenomenological models do not guarantee to learn system modes, sequential

and temporal behavior correctly. As a consequence, occurring point, context, and sequence anomalies can't be detected with certainty [31], [2].

We could do better by limiting the degrees of freedom of the models and providing a fixed and simple structure based on domain knowledge (explicit modeling). There are several types of explicit models that have been learned and then used for anomaly detection. In [12] Bayesian networks learn to discover dependencies between sensors and actuators. Another formalism is Hidden Markov Models (HMM), which are used to detect anomalies in a SCADA system, as [27] shows.
Of all the available formalisms, it is easy to argue that a discrete automaton or a Petri net is definitely a natural representation of a CPPS, since PLC programmers regularly use it as a programming paradigm [19], [5]. A timed automaton takes a discrete automaton one step further. Basically, it is a state machine that also keeps track of the timing of transitions between states. This is a huge advantage, because it models the interaction of the PLC program with the environment. In general it is a good model of a CPPS [15]. Anomaly detection using a timed automaton can be done by traversing through the automaton and comparing observations with the learned model. A significant deviation indicates an anomaly. Since a timed automaton is close to the practical experience of domain experts, it enables human-in-the-loop model improvement and validation [4]. Timed automata are used in practice to model the behavior in CPPS in [6] or [28].

A hybrid timed automaton is a timed automaton with state-specific continuous signals that don't correspond to a specific state. This allows for joint and context-aware analysis. In practice, the information provided by the continuous signals is stored in a statistical or machine learning context-aware model. This approach goes in a similar direction as the emerging field of informed machine learning, which deals with the integration of expert knowledge into machine learning models [21], [1].
But the formalism has its limitations. It doesn't take into account all the nuances and complexities of every possible PLC program, especially those with advanced process control and dynamic patterns. Before applying the methods presented in this paper, a domain expert should confirm the applicability. In general, this can be done quite intuitively with hybrid automata, since the identified states of the automaton correspond to the states of the system and can therefore be validated by an expert.

2.2 Discrete Methods

There exists a number of methods for identifying discrete timed models, which can be roughly divided into offline and online methods. BUTLA [16] and RTI+ [29] are two such methods that learn a timed automaton and differ in the methodology of state merging. Both methods have in common that they work offline, i.e. measurements have to be stored in advance and are permanently available. OTALA is a method that learns a timed automaton in an online manner [14]. Other possibilities are the learning of Petri nets, dynamic Bayesian networks or continuous-time Markov chains.

2.3 Continuous and Offline Hybrid Methods

One group of algorithms enables joint timing analysis of continuous and hybrid system behavior by creating a discrete representation of the overall system behavior. They convert continuous signals into discrete signals. Of course, this also works in the absence of discrete signals. Finally, an offline discrete algorithm like BUTLA can be used as if there were only discrete signals, or HyBUTLA for offline identification of hybrid timed automata using the initial discrete and continuous signals [21]. Methods for

intelligent discretization are e.g. Boltzmann machines ([8]), deep belief nets ([9]), or self-organizing maps ([30]).

2.4 Research Gap: Online Hybrid Methods

To the best of our knowledge, there is no online algorithm for learning hybrid timed automata. Existing algorithms for learning hybrid timed automata are offline, meaning that they require all input-output traces to be available in advance for learning. Online learning is a requirement for many industrial use cases. Often industrial systems have limited resources and need to process data in a stream. There is an entire research area dedicated to machine learning on edge devices [22].

There is the online algorithm OTALA for discrete data, but there seems to be no method that includes continuous data. It might be possible to train an intelligent discretization model online in advance and combine it with OTALA. This approach seems inconvenient because it's not certain that OTALA's assumption that each discretized state corresponds to a unique machine state holds. In this paper we want to introduce HyOTALA, an online algorithm for learning hybrid timed automaton models.

3 Algorithm

The previous section discussed the need for an online hybrid timed automaton learning method. In this section, we present our solution, called the Hybrid Online Timed Automaton Learning Algorithm (HyOTALA). HyOTALA builds on OTALA, originally introduced in [14], and is capable of learning timed automata without continuous signals.

The former algorithm OTALA is briefly summarized in the first subsection 3.1 and will be used to explain HyOTALA. The main difference between OTALA and HyOTALA is the concept of a hybrid state, which is explained in subsection 3.2. Finally, subsection 3.3 will explain how to integrate hybrid states into OTALA, resulting in HyOTALA.

3.1 OTALA

OTALA is an online algorithm that can identify a timed automaton model of a CPPS using discrete PLC IO signals. It avoids the need to collect labeled anomaly samples because only a model of normal operation needs to be learned, it is only required that the recorded data originate from the normal process. In addition, OTALA has the advantage of being able to autonomously detect when the learning process has reached convergence.

As shown in algorithm 1, OTALA can be implemented using three data structures: an automaton, a state, and a transition.
The concept of an automaton state is encapsulated by the state class. It is characterized by an IO vector and a visit counter. The IO vector consists of discrete inputs and outputs and is a primary key for a state, because OTALA assumes that an IO vector corresponds to exactly one machine state. In practice, a domain expert must verify this assumption by selecting unambiguous signals that reflect the machine state. In addition, there should be one model for each machine module that operates concurrently.
Transitions between states are modeled using transition objects. Each transition is defined by its previous and next state, a visit counter, and a timing distribution. The original implementation of OTALA only keeps track of the minimum and maximum timing. However, experience has shown that a more robust representation is needed. Therefore, we propose to learn a timing probability distribution that better reflects the temporal behavior. If a normal distribution can be assumed, online parameter estimation

Algorithm 1 OTALA: Data structures

```
1: class STATE
2:     discreteIO : IO VECTOR
3:     visitCounter : INT
4: end class

5: class TRANSITION
6:     prevState : STATE
7:     nextState : STATE
8:     timingDistribution : ONLINE-PROBABILITY-DISTRIBUTION
9:     visitCounter : INT
10: end class

11: class AUTOMATON
12:     States : DICT                                  ▷ Key: Discrete IO
13:     Transitions : DICT                             ▷ Key: Previous and Next State
14:     NoChangeCounter : INT
15: end class
```

using Welford's method is a good option [3].
The heart of OTALA is the automaton class, which represents the entire automaton. It consists of a dictionary of states, indexed by their discrete IO vectors, and a dictionary of transitions, indexed by a combination of previous and next state IOs. This structure allows for efficient retrieval and updating of states and transitions during the learning process. In addition, the automaton also maintains a no-change counter, which is incremented if an observation doesn't change the automaton, or reset otherwise.

Algorithm 2 OTALA: Training loop

```
16: A ← initAutomaton()
17: while newObservationExists() and not isConverged(A) do
18:     t_t, d_t ← getNewObservation()                 ▷ time, discrete IO
19:     s_t : State ← getOrCreateState(A, d_t)
20:     visitState(s_t)                                ▷ counter
21:     if s_t ≠ s_{t−1} then
22:         tr : Transition ← getOrCreateTransition(A, s_t, s_{t−1})
23:         visitTransition(tr, t_t, t_{t−1})          ▷ counter, timing distribution
24:     end if
25:     trackConvergence(A)
26: end while
```

OTALA's primary objective is to construct an automaton that accurately represents the dynamics of a system based on discrete IO observations. The algorithm iteratively refines the automaton until it converges, indicating that a stable representation of the system's dynamics has been learned. This learning process is described in algorithm 2.

The learning process ends when either there are no more samples or *isConverged* in line 17 evaluates to true. The stability of the representation correlates with the number of recent adjustments. Thus, *isConverged* essentially checks whether the no change counter of the automaton exceeds a threshold.
In line 18 *getNewObservation* parses the new observation into a timestamp and a discrete IO vector. In the following line, *getOrCreateState* either creates a new state object or

retrieves the state object corresponding to the IO.
The if condition in line 21 checks if the state has changed. If there is a state change, a transition object is created or retrieved. Besides incrementing the visit counter, *visitTransition* (line 23) is responsible for fitting the time distribution.
Finally, line 25 tracks the convergence by maintaining the no change counter. The counter is reset when a new state or transition is created, or when the transition timing distribution changes significantly.

3.2 Hybrid State

Because discrete PLC IOs carry information such as conveyor on/off, an automaton state is highly correlated with the CPPS mode or context. Since continuous signals are usually highly mode dependent, such as the vibration of a conveyor, it makes sense to embed a state-aware continuous model into an OTALA state. A model embedded into a state results in a hybrid state (HyState) as shown in algorithm 3.

Algorithm 3 Hybrid State

```
1: class HYSTATE(STATE)
2:     contiuousModel : ONLINE-ML-MODEL
3:     metric : ONLINE-ML-METRIC
4: end class
```

Since a *HyState* is to be integrated into OTALA, the online properties must be preserved and the model must be learnable online. Here the term model also includes an online preprocessing pipeline. A *HyState* also needs an online metric such as an R2 score for two main reasons. First, it helps to measure the quality of the learned model. Second, OTALA measures the convergence of the learning process, and to extend it to hybrid states, the online metric can be used to estimate the convergence of each continuous model.

Choosing a model and a metric requires domain expertise. For inspiration and implementations of online machine learning methods, we recommend taking a look at River, a library for machine learning with streaming data [20]. Using a library like River allows you to easily build complex online pipelines, including steps like feature selection, scaling, and model training. As explained in 2, always keep in mind the computational and memory constraints and the higher computational demands of complex pipelines. Online ML models such as autoencoder neural networks for embedded devices are actively being researched [23], [7].
However, for most tasks, you don't need really complex models. One thing we have learned is that simple models work (surprisingly) well in HyOTALA. This is because the context is given and mode switches that introduce nonlinearity, like a gear shift, are handled by the timed automaton model.

A good starting point is to estimate joint or single distributions of each continuous signal and use them as the continuous model. This has the advantage that these models are easy to interpret and human-in-the-loop feedback can be used for model optimization and quality control [4].
Another option is linear models, which for example take an aggregation of recent observations as input and predict the next observation, also used in edge computing [26]. In proof-of-concept experiments, we have used Bayesian linear regression, which has the advantage of handling uncertainty [18]. It has also proven effective in other online state estimation tasks of CPS [13]. The model naturally simplifies the pipeline by eliminating

the need for preprocessing. It is also capable of modeling uncertainty in a system. In our internal proof-of-concept case study, it was able to model the system reasonably well.

3.3 HyOTALA: A Learning Algorithm for Hybrid Systems

As mentioned before, the main difference between OTALA and HyOTALA is the replacement of states by hybrid states. Almost everything else is the same. The training loop must also be adapted. Most of the steps to train the system, as shown in 2, remain the same. However, when there is new data (*getNewObservation*, 18), there is also a continuous signal vector, noted as c_t. The other change concerns the function *visitState* (20). The new implementation is shown in the algorithm 4.

Algorithm 4 HyOTALA: visitState

```
1: function VISITSTATE(s_t, c_t)          ▷ Input: State + Continuous Data
2:     incrementCounter(s_t)
3:     f_t = preprocess(c_t, other)       ▷ Create a feature vector
4:     learnOne(s_t, f_t)                 ▷ Adapt model of state s
5:     evaluate(s_t, f_t)                 ▷ Adapt metric of state s
6: end function
```

When a state is visited, the former OTALA implementation just incremented the visit counter. HyOTALA also updates the continuous model of the current state with the new continuous data (c_t). This involves a preprocessing pipeline that creates a feature vector f_t that is used for model training. Note that the creation of the feature vector can also take into account an old feature vector f_{t-1} or a signal vector c_{t-1} stored in an internal state. Finally, the online metric is updated by evaluating the current model.

A key aspect that has not been thoroughly explored is the impact of the ratio of continuous to discrete signals on the algorithm's quality. We suggest to maintain a balanced ratio of continuous to discrete signals. A rule of thumb indicates that the ratio of continuous to discrete signals (conti/discrete) should not exceed 0.5.

4 Evaluation

The proposed learning algorithm is applied to real data extracted from a Haver & Boecker OHG packaging machine. Haver & Boecker OHG is a machine manufacturer that specializes in building packaging machines that pack bulk materials into bags. A packaging machine can consist of several filling modules. As an example, a filling module is shown in Figure 1. The main components of a filling module are the weighing unit and the impeller that feeds the product into a bag. The impeller is not visible in this illustration because it is located behind the filling tube. Each filling module follows a cyclical pattern of placing the bag on the filling tube, taring the weight, filling the bag, sealing the bag, and releasing the bag. For simplicity, only one filling module is considered in this section. This is also done to avoid possible parallel processes. The data set includes 71 binary and 2 continuous characteristics from each PLC cycle. Error messages from the machine are also included in the data set.

The training data was reviewed by domain experts for anomalies and errors. A time period was chosen where the machine was continuously packing and no errors occurred. Each continuous feature is modeled by a separate model. Initial modeling efforts revealed a major problem in learning the continuous models. This problem concerns the states during the filling process. In some cases, a binary feature may experience a rising edge

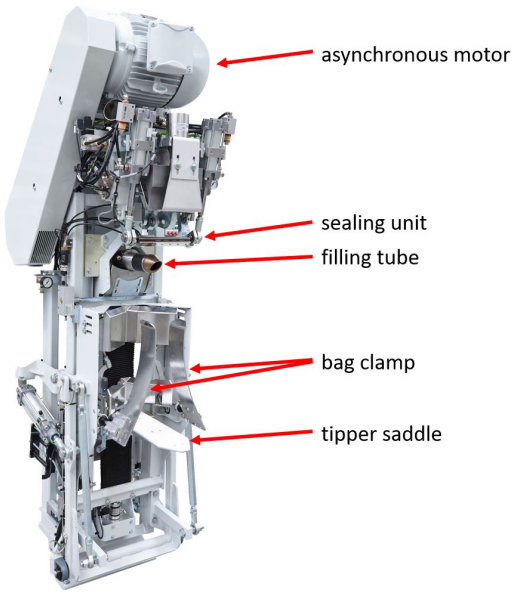

Figure 1: A Haver & Boecker OHG Filling Module

for a few PLC cycles, causing a brief exit from the current state during filling. This can make it difficult to model the continuous features as a function of time because the values of the continuous feature increase during the brief exit. The difficulty is that when the state is re-entered, the value of the continuous feature is higher than when the state was first entered. Figure 2 illustrates the problem. Fill 1 is a fill where this phenomenon did not occur. However, the phenomenon is observed in Fill 2. It is clear that the value of the signal is significantly larger when the state is re-entered. To solve this problem, domain experts have recommended removing certain binary features. Therefore, the state is no longer left for a short time.

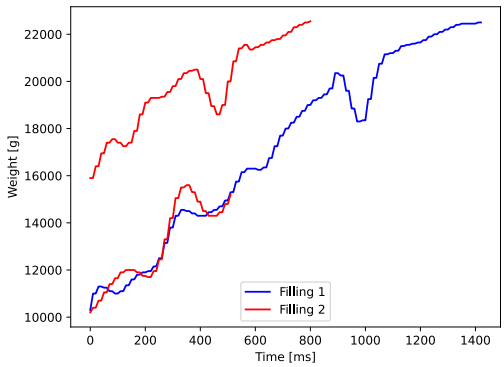

Figure 2: Continuous Signals in one Hybrid State

Using HyOTALA results in an automaton with about 45 states and 60 transitions, which is shown in Figure 3. The cyclic behavior mentioned above is easily recognizable. The results were shown to several domain experts, mostly PLC programmers. They were able to identify different stages of the packaging machine based on the provided automaton. Even on closer inspection by the domain experts, this automaton is considered an appropriate abstraction of the machine's behavior.

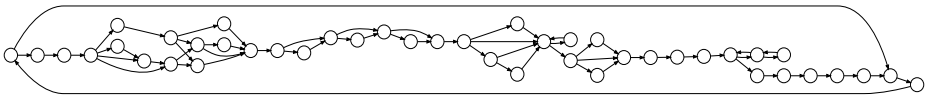

Figure 3: Anonymized Hybrid Timed Automaton for a Packaging Process

Figure 4 compares the total number of changes to the number of PLC cycles. In this figure, only the changes resulting from the addition of a new state or transition are counted. It is obvious that most states and transitions are learned after only one fill. This paper only discusses the convergence of the binary part. In addition, future considerations should include the convergence of the continuous models and the time distributions of the transitions.

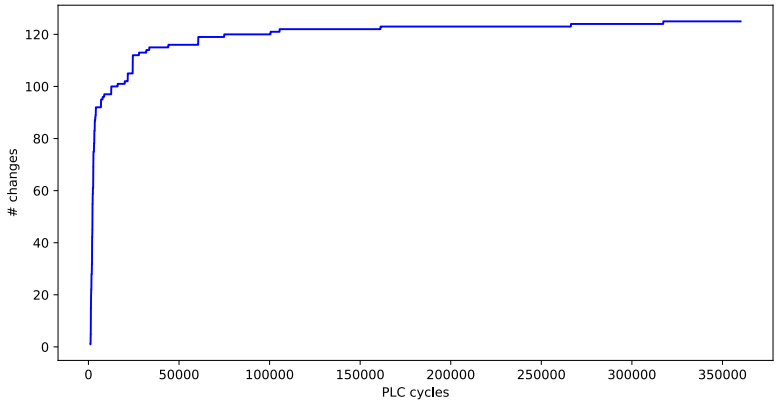

Figure 4: Convergence

Two anomalies are used to further determine the learning performance of the resulting automaton. The first case is a typical anomaly that occurs in packaging machines: Overweight or underweight of the filled bag. Underweight or overweight bags can occur for several reasons. For example, the product flow is not optimal and the product flows at different speeds during filling. Another reason could be an improperly parameterized packaging machine. The filling of an overweight bag is considered as an example. The set of binary states observed during filling is compared with the set of binary states of the machine. This comparison shows that 4 states were observed during the filling process that are not represented in the learned automaton. This kind of observation is called a binary anomaly. The same comparison was made with the filling of an underweight bag. This comparison also shows unknown states that are not contained in the learned automaton.

A second type of anomaly is considered that focuses on the learning performance of continuous models. Unlike the previous example, which considers binary models. When the bag is discharged after filling and sealing, it falls onto the conveyor belt below. Under unfortunate circumstances, the bag may tilt as it falls onto the conveyor belt. As a result, the following bags cannot be transported any further and a deadlock occurs. In the current configuration of the packaging machine, there are no sensors installed in this area to detect this error. In this case, the observed continuous signals of a particular state are compared with the corresponding learned continuous models. Immediately after the bag is tilted in the machine, unusual deviations are observed in some states before the following filling. This result is plausible and can be explained by domain experts. The reason for the unusually high weight measurement before filling is due to the tilted bag pressing on the machine's weighing unit. However, the weighing unit tares and no further anomalies are detected during the subsequent filling process. We can therefore show that, at least in this case, several continuous anomalies are detected, even though no sensors are explicitly installed for this case.

5 Conclusion

The main lessons learned from the practical application of HyOTALA are that the integration of domain knowledge during data preprocessing, training data selection and training of HyOTALA has a significant impact. It avoids the need to collect labeled anomaly samples because only a model of normal operation needs to be learned.
The transparency and interpretability of the algorithm was particularly helpful during the various modeling tests during training. In addition, the algorithm seems intuitive to humans, and most domain experts with no previous experience in machine learning were able to grasp the algorithm quickly. This aspect is particularly important because it promotes the acceptance of this algorithm within the company. The modeling results seem to be understandable so far. Another finding is that a short change from one state to another can be counterproductive in the case of continuously increasing or decreasing continuous values. Especially when the continuous processes are modeled as a function of time.

Going forward, the availability of more publicly available datasets with hybrid signals containing both continuous and discrete elements would be helpful. Such datasets are essential for rigorously testing and refining HyOTALA and similar algorithms, thereby improving their applicability to real-world hybrid scenarios. In addition, conducting more extensive experiments with HyOTALA could provide deeper insights into the effectiveness of online pipelines in different contexts. This would be particularly beneficial for newcomers to HyOTALA, providing them with a solid foundation and clear guidelines for implementation.
One of the most promising applications of HyOTALA is in anomaly detection, as suggested in the evaluation section 4. Further research should focus on developing a detailed framework and best practices for the use of hybrid timed automata in practical settings. In conclusion, while HyOTALA represents a significant advancement in the field of machine learning for CPPS, its full potential can only be realized through continued exploration, refinement, and application.

References

[1] V. Cobilean, H. S. Mavikumbure, C. S. Wickramasinghe, D. L. Marino, and M. Manic. Informed deep learning for anomaly detection in cyber-physical systems.

In *2023 IEEE International Conference on Industrial Technology (ICIT)*, pages 1–7. IEEE, 2023.

[2] A. A. Cook, G. Misirli, and Z. Fan. Anomaly detection for iot time-series data: A survey. *IEEE Internet of Things Journal*, 7:6481–6494, 2020.

[3] A. A. Efanov, S. A. Ivliev, and A. G. Shagraev. Welford's algorithm for weighted statistics. In *2021 3rd International Youth Conference on Radio Electronics, Electrical and Power Engineering (REEPE)*, pages 1–5. IEEE, 2021.

[4] C. Emmanouilidis, S. Waschull, J. A. Bokhorst, and J. C. Wortmann. Human in the ai loop in production environments. In *Advances in Production Management Systems. Artificial Intelligence for Sustainable and Resilient Production Systems: IFIP WG 5.7 International Conference, APMS 2021, Nantes, France, September 5–9, 2021, Proceedings, Part IV*, pages 331–342. Springer, 2021.

[5] G. Frey and L. Litz. Formal methods in plc programming. In *Smc 2000 conference proceedings. 2000 ieee international conference on systems, man and cybernetics.'cybernetics evolving to systems, humans, organizations, and their complex interactions'(cat. no. 0*, volume 4, pages 2431–2436. IEEE, 2000.

[6] C. Gerking, D. Schubert, and E. Bodden. Model checking the information flow security of real-time systems. In *Engineering Secure Software and Systems: 10th International Symposium, ESSoS 2018, Paris, France, June 26-27, 2018, Proceedings 10*, pages 27–43. Springer, 2018.

[7] T. L. Hayes and C. Kanan. Online continual learning for embedded devices. *arXiv preprint arXiv:2203.10681*, 2022.

[8] 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.

[9] N. Hranisavljevic, O. Niggemann, and A. Maier. A novel anomaly detection algorithm for hybrid production systems based on deep learning and timed automata. *arXiv preprint arXiv:2010.15415*, 2020.

[10] P. Kamat and R. Sugandhi. Anomaly detection for predictive maintenance in industry 4.0-a survey. In *E3S web of conferences*, volume 170, page 02007. EDP Sciences, 2020.

[11] C. Krupitzer, T. Wagenhals, M. Züfle, V. Lesch, D. Schäfer, A. Mozaffarin, J. Edinger, C. Becker, and S. Kounev. A survey on predictive maintenance for industry 4.0. *arXiv preprint arXiv:2002.08224*, 2020.

[12] Q. Lin, S. Adepu, S. Verwer, and A. Mathur. Tabor: A graphical model-based approach for anomaly detection in industrial control systems. In *Proceedings of the 2018 on asia conference on computer and communications security*, pages 525–536, 2018.

[13] Z. Lyu, G. Wang, and C. Tan. A novel bayesian multivariate linear regression model for online state-of-health estimation of lithium-ion battery using multiple health indicators. *Microelectronics Reliability*, 131:114500, 2022.

[14] A. Maier. Online passive learning of timed automata for cyber-physical production systems. In *2014 12th IEEE International Conference on Industrial Informatics (INDIN)*, pages 60–66, 2014.

[15] A. Maier, O. Niggemann, and J. Eickmeyer. On the learning of timing behavior for anomaly detection in cyber-physical production systems. In *DX*, pages 217–224, 2015.

[16] A. Maier, O. Niggemann, R. Just, M. Jäger, and A. Vodenčarević. Anomaly detection in production plants using timed automata-automated learning of models from observations. In *International Conference on Informatics in Control, Automation and Robotics*, volume 2, pages 363–369. SciTePress, 2011.

[17] W. Marfo, D. K. Tosh, and S. V. Moore. Condition monitoring and anomaly detection in cyber-physical systems. In *2022 17th Annual System of Systems Engineering Conference (SOSE)*, pages 106–111. IEEE, 2022.

[18] A. P. Mathur and N. O. Tippenhauer. Swat: A water treatment testbed for research and training on ics security. In *2016 international workshop on cyber-physical systems for smart water networks (CySWater)*, pages 31–36. IEEE, 2016.

[19] O. Modrlák and M. Sbieschni. Sequence control of a drilling machine by simatic s7-300. 2019.

[20] J. Montiel, M. Halford, S. M. Mastelini, G. Bolmier, R. Sourty, R. Vaysse, A. Zouitine, H. M. Gomes, J. Read, T. Abdessalem, et al. River: machine learning for streaming data in python. 2021.

[21] O. Niggemann, B. Stein, A. Vodencarevic, A. Maier, and H. K. Büning. Learning behavior models for hybrid timed systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 26, pages 1083–1090, 2012.

[22] P. P. Ray. A review on tinyml: State-of-the-art and prospects. *Journal of King Saud University-Computer and Information Sciences*, 34(4):1595–1623, 2022.

[23] H. Ren, D. Anicic, and T. A. Runkler. Tinyol: Tinyml with online-learning on microcontrollers. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2021.

[24] O. Serradilla, E. Zugasti, and U. Zurutuza. Deep learning models for predictive maintenance: a survey, comparison, challenges and prospect. *ArXiv*, abs/2010.03207, 2020.

[25] D. Shetve, I. Raju, R. V. Prasad, R. Trestian, H. Nguyen, and H. Venkataraman. Adaptive n-step technique for real-time anomaly detection in smart manufacturing. *2022 IEEE 5th International Conference on Industrial Cyber-Physical Systems (ICPS)*, pages 01–06, 2022.

[26] H. Sivan, M. Gabel, and A. Schuster. Online linear models for edge computing. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2019, Würzburg, Germany, September 16–20, 2019, Proceedings, Part I*, pages 645–661. Springer, 2020.

[27] K. Stefanidis and A. Voyiatzis. An hmm-based anomaly detection approach for scada systems. pages 85–99, 2016.

[28] G. Sugumar and A. Mathur. Assessment of a method for detecting process anomalies using digital-twinning. In *2019 15th European Dependable Computing Conference (EDCC)*, pages 119–126. IEEE, 2019.

[29] S. Verwer. Efficient identification of timed automata: Theory and practice. 2010.

[30] A. von Birgelen and O. Niggemann. Using self-organizing maps to learn hybrid timed automata in absence of discrete events. In *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8. IEEE, 2017.

[31] Y. Wu, H. Dai, and H. Tang. Graph neural networks for anomaly detection in industrial internet of things. *IEEE Internet of Things Journal*, 9:9214–9231, 2021.