Bridging the Gap Between Anomaly Detection and Runtime Verification: H-Classifiers

1st Hagen Heermann
Cyber-Physical Systems Chair
University of Kaiserslautern-Landau
Kaiserslautern, Germany
heermann@informatik.uni-kl.de

2rd Christoph Grimm
Cyber-Physical Systems Chair
University of Kaiserslautern-Landau
Kaiserslautern, Germany
grimm@informatik.uni-kl.de

***Abstract*—Runtime Verification (RV) and Anomaly Detection (AD) are crucial for ensuring the reliability of cyber-physical systems, but existing methods often suffer from high computational costs and lack of explainability. This paper presents a novel approach that integrates formal methods into anomaly detection, transforming complex system models into efficient classification tasks. By combining the strengths of RV and AD, our method significantly improves detection efficiency while providing explainability for failure causes. Our approach offers a promising solution for enhancing the safety and reliability of critical systems.**

***Index Terms*—Hybrid Systems, Runtime Monitoring, Consistency Checking, Reachability Analysis, Anomaly Detection**

I. Introduction

Ensuring the correctness and reliability of cyber-physical systems (CPS) is of paramount importance, particularly in safety-critical domains such as avionics, healthcare devices, and autonomous systems. These systems operate in dynamic and unpredictable environments, where even minor deviations can lead to catastrophic failures. To address these challenges, two key approaches have emerged as central techniques for ensuring system dependability: Runtime Verification (RV) and Anomaly Detection (AD).

Runtime Verification (RV) is a formal method that continuously monitors a system's execution in real time to ensure compliance with formally defined properties. RV checks the behavior of the system against a predefined set of specifications, providing immediate feedback and enabling corrective measures when violations occur. This approach is especially critical in domains where formal correctness guarantees are essential—such as autonomous vehicles and medical devices—ensuring continuous adherence to safety protocols and system reliability.

On the other hand, *Anomaly Detection (AD)* is a data-driven technique aimed at identifying deviations from expected system behavior, which may indicate potential errors, faults, or security threats. AD methods are typically categorized into model-based approaches, which use predefined statistical or machine learning models, and data-driven approaches, which rely on observed behavior without formal specifications. Although AD can be effective in detecting irregularities, it often struggles with explainability—understanding *why* an outlier was identified—and is susceptible to false positives and negatives due to the lack of formal guarantees. In particular, AD often flags deviations without offering clear insight into how these deviations relate to the underlying model or system dynamics, making it difficult for system operators to interpret and respond to detected anomalies.

This work was partially funded by the BMBF project KI4BoardNet No. 16ME0782 and the DFG project MoVe4SPS No. 530118585

In this work, we propose a novel framework that integrates formal methods from RV into the anomaly detection process, enhancing both the precision and explainability of the system. Our approach leverages techniques from [1], [2], transforming linear inequality systems into a more tractable space, thereby reducing the problem to verifying whether the inequality systems are satisfied by the observed system trajectory. This not only improves the precision of anomaly detection by grounding it in formal verification techniques but also significantly enhances explainability.

Specifically, by embedding a formal model into the anomaly detection process, our method can provide detailed explanations for why an anomaly was flagged. When an anomaly is detected, the system can indicate whether it adheres to the formal model or not. If the measurements conforms to the model, the system can specify which dynamics of the model it fits. However, if the anomaly does not adhere to the model, the system can highlight this deviation. This capability provides a clearer understanding of the system's behavior and helps in identifying which aspects of the formal model are not being met. This improved explainability supports more effective corrective actions and helps in discerning whether detected anomalies are due to model misfit or other factors, thus reducing the chances of false positives. By associating detected anomalies with formal model properties and dynamics, our approach enhances insight into the root causes of deviations, improving both system robustness and operator trust.

II. State of the Art

Runtime Verification (RV) is a formal technique that monitors system executions to ensure adherence to predefined specifications. For hybrid systems, RV often involves checking whether observed behaviors satisfy temporal logic formulas [3] or comply with mathematical models.

Property-based monitoring, for example, continuously verifies system behavior against a set of predefined properties [4].

978-3-9826741-0-0/DATE25/© 2025 EDAA

Alternatively, Signal Temporal Logic (STL) [5], [6] offers a formal language for specifying and monitoring temporal properties. More complex verification tasks often leverage formal logical systems and dedicated solvers, such as ModelPlex [7].

Building upon the work from [1], [2], this paper employs symbolic simulation [8] for RV. Unlike traditional point-based simulation, symbolic simulation explores entire sets of possible states, enabling more comprehensive analysis.

While these approaches are effective, they are *static* in the sense that their verification capabilities do not improve over time. Recent research has explored the integration of machine learning with RV. For instance, $\mathcal{P}revent$ [9] represents a step in this direction. Generally, RV methods rely on computationally intensive solvers to determine system correctness. In general, the runtime verification approaches utilise larger solving tools to make verdicts on correctness. In contrast, Anomaly Detection (AD) works in a data driven manner. With this we mean that it does not rely on specifications but on measurements. It utilises this measured data as a reference for the typical behaviour and then identifies deviations. Anomaly detection methods often utilize statistical methods and machine learning techniques as a baseline for their classification. A well-known technique is the utilization of support vector machines (SVMs) as described in "New Support Vector Algorithms" [10]. A very important metric in many anomaly detection methods is the distance between two points in space. This distance metric is for example utilised in the k-nearest neighbors (k-NN) approach discussed in "Distance-Based Outlier Detection in Data Streams" [11]. Statistical methods evaluate data statistically and classify new data based on this information, exemplified in "Statistical Anomaly Detection with Sensor Networks" [12]. More training intensive approaches are deep learning approaches [13], for example auto-encoders [14] or hybrid models like VAE-LSTM [15]. All of these methods have in common that they are based on a simple classification function. This is demonstrated for the SVM case where the evaluation of $f(x) = sgn(\vec{w}^T\vec{x} + b)$, with sgn being the sign function, is enough for classification.

While explainability in RV is often inherent due to its reliance on formal methods, this is not always the case for AD. Given the increasing use of AD in safety-critical systems, ensuring the explainability of its results is paramount [16]. Several recent approaches have aimed to address this challenge [16]–[18], with practical applications already being demonstrated in fields such as the space industry [18]. This paper presents a novel approach that combines the rigor of RV with the efficiency of AD. By transforming system models into special decision diagrams and deriving inequality systems, we construct simple yet effective classifiers. This approach preserves the precision and explainability of RV while significantly reducing computational complexity. There are certain moves in a similar direction such as CORA [19], but to the best of our knowledge no similar approach exists.

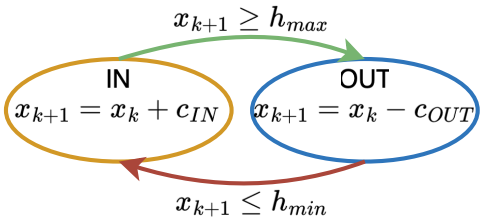

Fig. 1. Hybrid automaton model of a water tank system, where x_k represents the water level at time step k, constrained between the minimum level h_{min} and maximum level h_{max}. The guard conditions $x_{k+1} \geq h_{max}$ and $x_{k+1} \leq h_{min}$ decide for which continuous states to change to another discrete state.

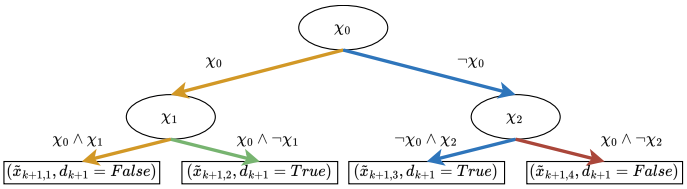

Fig. 2. During the development phase, the system model—specifically the hybrid automaton model defined in Fig. 1—is transformed into an Affine Arithmetic Cartesian Decision Diagram (AACDD) [1]. Each edge is labeled by a symbol χ, which acts as a placeholder for a specific linear inequality.

III. Approach

A. Hybrid Systems, Decision Diagrams and the Water Tank Example

To develop, illustrate and validate our method, we employ a water tank system as a case study. This system is modeled as a hybrid automaton with two discrete states (IN and OUT) and a single continuous state representing water height (Fig. 1). Water inflow occurs in the IN state until a maximum level is reached, triggering a transition to the OUT state where water is drained until a minimum level is attained. The system's simplicity and visualizability make it well-suited for demonstrating our approach.

The approach is based on [1] and decomposed into two phases. In the initial phase, the hybrid automaton model is converted into a decision diagram. This conversion is demonstrated on the water tank example in Fig. 2. Where the colors depict which transition guards, from Fig. 1, belong to which edges in the decision diagram. The decision diagram encapsulates all the different discrete transition possibilities. For the continuous transitions, affine forms are utilized to encode the possible continuous transitions. Affine forms [20] are of the form $\tilde{x} = c + \sum a_i\epsilon_i$, with ϵ_i being the so-called noise symbols. These allow the modelling of value ranges for variables and linear correlation due to sharing of noise symbols. At the end of the conversion, the decision diagrams (AACDD [2]) features linear predicates in its internal nodes to encapsulate transition guards, and uses affine forms combined with discrete variables to describe both the continuous and discrete state spaces. Generally, in an AACDD the complete state of the system, discrete and continuous, is modeled. In this case the state for x_{k+1}. In the leafs there are tuples containing affine

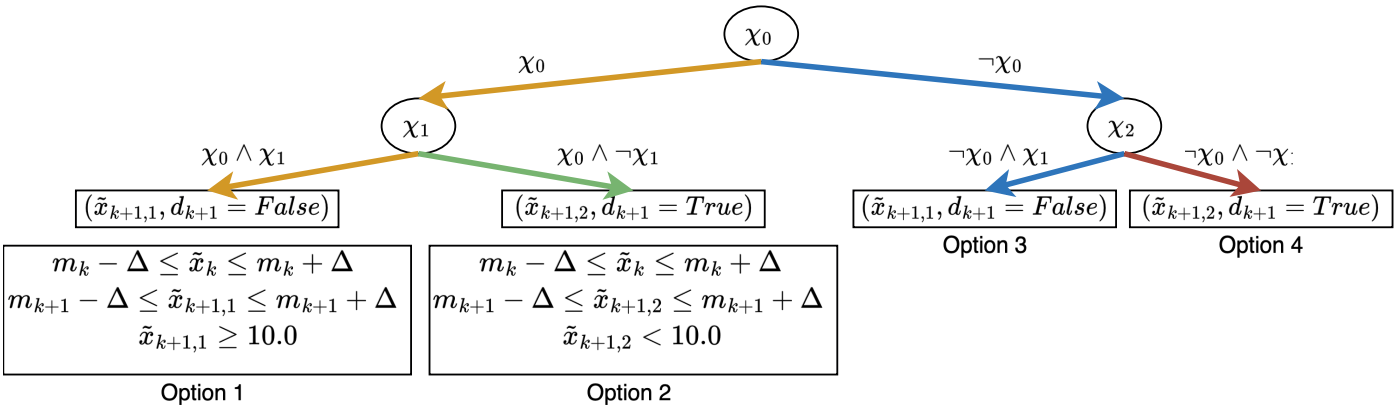

Fig. 3. At runtime, based on the measured transition (m_k, m_{k+1}), the decision diagram for the water tank system generates four distinct systems of inequalities. If any of these systems have a solution, the observed transition is considered consistent with the model.

forms for the value ranges of the continuous state variables and boolean values for the discrete state. In the internal nodes, labeled with χ, are either inequalities over the noise symbols or decision variables. During runtime, this decision diagrams is transformed, together with the measurements from the monitored system, into inequality systems —one inequality system per leaf of the decision diagrams. This process is visualized in Fig. 3. If any of these systems have a solution, the measured transition is classified as consistent with the model and not an anomaly. Otherwise, it is considered an anomaly. To prove the existence of a solution, a simplex linear programming solver is utilized. However, this dependency on a linear programming solver introduces a problematic knowledge gap regarding the timing of results. While certain bounds are guaranteed, they are not always existent. Additionally, there is a heavy reliance on solver tool making this approaches classification function complex algorithmic and space wise. To address this issue and improve runtime, we propose a novel approach based on the outlined method.

B. H-Classifier

The Affine Arithmetic Cartesian Decision Diagram introduces two types of consistencies for each leaf:

1) **Continuous Consistency:** This involved inequalities, as shown in Fig. 3, constraining the affine forms $\tilde{x}$ representing the system's continuous variables. Since these affine forms are dependent on noise symbols, the constraints applied to the space of these noise symbols.
2) **Discrete Consistency:** Here, inequalities over noise symbols arise from the path to the leaf, linked to the transition guards of the hybrid automaton.

In our novel approach, we shift emphasis and replace the concept of continuous consistency by using the joint range $\langle \tilde{x}_k, \tilde{x}_{k+1} \rangle$. The joint range of affine forms refers to the set of all possible pairs of values that two affine forms can simultaneously take based on a shared input space. This joint range precisely defines points in the transition space represented by these two affine forms. It allows us to define a function $G : \mathbb{R}^m \to \mathbb{R}^n$, which maps every possible assignment in the noise symbol space to one in the transition space while preserving the continuous consistency requirements.

To ensure that discrete consistency is maintained, we need to transform the inequality system $A\epsilon \leq 0$, which defines the path constraints over the noise symbols, into the transition space using the Generator matrix G. This transformation ensures that the points in the transition space, already consistent with the continuous consistency condition via G, also satisfy the discrete consistency conditions derived from the path constraints.

We express the inequality system for these path constraints as $A\vec{\epsilon} \leq \vec{b}$. Additionally, for each noise symbol, we introduce two bounding inequalities, $\epsilon \leq 1$ and $\epsilon \geq -1$, which restrict the noise symbol values to those consistent with the joint range. These additional constraints help to ensure that the transformed system only represents points within the joint range, thus encoding continuous consistency established by the transformation G. Any inequality system involving affine forms over noise symbols can be brought into this form. Now, we perform the following transformation using G:

$$\mathbf{A}(\mathbf{G}^{+}(\vec{x} - \vec{c})) \leq \vec{b}, \quad \mathbf{A}'\vec{x} \leq \vec{b}', \tag{1}$$

where $\mathbf{A}' = \mathbf{A}\mathbf{G}^{+}$ and $\vec{b}' = \vec{b} + \mathbf{A}'\vec{c}$.

After performing the transformation, we obtain a new inequality system $\mathbf{A}'$, and for simplicity, we omit $\vec{b}'$ in further discussion. This new system operates within the transition space. Since the Generator matrix G is generally not square ($\mathbb{R}^{m \times n}$), we rely on the Moore-Penrose pseudoinverse [21] to handle the transformation. The pseudoinverse allows us to invert G in a generalized way, even when it is non-square, ensuring that the transformation remains valid.

All points in the transition space that satisfy this new inequality system $\mathbf{A}'$ will also satisfy the consistency conditions for the associated leaf. This means that, given a measurement vector $\vec{m}$, we can check whether it is consistent with the leaf's dynamics by evaluating whether it satisfies the condition $\mathbf{A}'\vec{m} \leq \vec{b}'$. If this inequality holds, it confirms that the measurement aligns with both continuous and discrete consistency for that specific leaf.

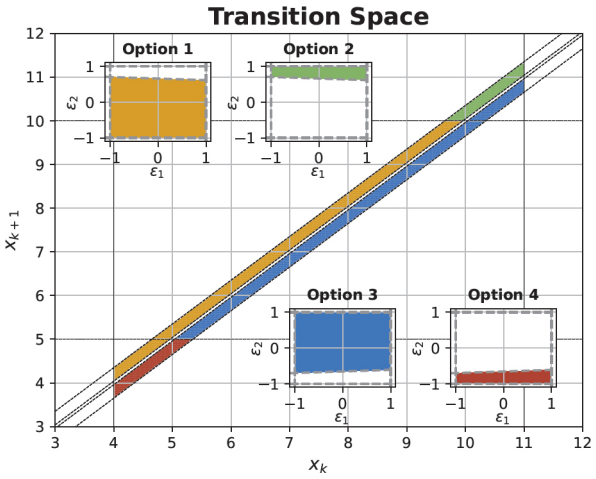

Fig. 4. Fig. 3 presents inequality system regions in the assignment space (here insets labeled Option 1-4) derived from the decision diagram's leaf nodes. The corresponding geometric regions in the transformed transition space serve as classifiers for our proposed method. Colors in the figure align with the respective options in Fig. 3

.

As the AACDD serve as a reference, we need to check consistency for every leaf. Thus, for each leaf, we construct its respective inequality system $\mathbf{A}'_j$, with k representing the total number of leaves. The acceptance strategy is that if the measurement vector $\vec{m}$ is consistent with any of the dynamics across the leaves, we accept it as consistent with our model. This can be formalized in the following logical equation:

$$\bigvee_{i=1}^{k} \mathbf{A}'_i \vec{m} \leq \vec{b}_i. \tag{2}$$

We define this entire process as our H-classifier. The term "H-classifier" stems from the fact that we are determining whether the measurement vector lies within a polytop in the transition space. This polytop is represented in half-space form [22], as it is defined by inequalities that split the space into two halves. The division and the relationship between the noise symbol space and the transition space are shown in Fig. 4 for the water tank example.

In comparison to [1], a similar inequality system was constructed for each leaf, as shown in Fig. 3. However, in that approach, the inequality systems were bounded by measurement values and had to be rebuilt from scratch for each new measurement. Moreover, since these systems were over the noise symbol space, they could not be directly evaluated with the measurement vector, requiring a simplex solver for solution verification.

In addition to just simply being able to classify into, "fits the model" or "does not fit the model", we can give additional information. We can precisely state which specific parts of the dynamics the observed transitions fits to.

IV. Performance Evaluation of the H-Classifier Approach

To evaluate our proposed method, we conducted comparative experiments against a One-Class SVM and the results from the approach [1], [2] this work is based on. In the result tables we will denote the approach this work is based on as "Formal Method". We generated eight synthetic datasets, four for each system (water tank and $\Sigma - \Delta$ model). These datasets were categorized as follows:

1) **Correct (C) Datasets**: Trajectories were sampled from nominal simulation runs within specified operating conditions.
2) **Small Parameter Excursion (SPE) Datasets**: Trajectories were generated with slight parameter deviations beyond allowed limits.
3) **Large Parameter Excursion (LPE) Datasets**: Trajectories were produced with significant parameter variations exceeding acceptable bounds.
4) **Noise (N) Datasets**: Trajectories were obtained from nominal simulations with added uniform noise.

First, we will review the results for each system before discussing the general results, starting with the water tank system. Since the transition space of the water tank model is two-dimensional, we can plot the classification results. In Fig. 5, the left side shows the classification results using the new approach, while the right side shows the results of the One-Class SVM (OCSVM) classification. For the presented results we utilised the OCSVM implementation provided by the scikit-learn library [23]. The underlying polytopes are also plotted in the results of the new approach, demonstrating how transitions inside the polytopes are classified as correct, while those outside are classified as outliers. The classification results for the water tank system are presented in Tab. I. The novel approach provides results nearly similar to the original approach. Although the accuracy is very similar to the results of the approach this work is based on, there is a significant improvement in the speed of classification, as shown in Tab. II. Despite the promising results for the water tank system, there are some issues with the novel approach when applied to the $\Sigma - \Delta$ model. The classification results, shown in Tab. III, indicate that the novel approach is less precise than the original approach because it accepts more transitions as inliers. This issue likely arises from two factors: the inequality system not being accurately transformed into the transition space, and the introduction of numerical errors due to the use of the Moore-Penrose matrix in the transformation process. While some accuracy is lost, there is a significant gain in classification speed, as demonstrated in the timing results in Tab. IV. When taking real time response requirements into consideration such as 40 to 100ms or 20 to 50ms for fighter jets [24], then the classification speeds are in the realm of real time capability.

Overall, we can conclude that the presented approach trades off some accuracy for a considerable increase in speed. In the previous section we displayed how we changed the initial classification approach. We adjusted it to be more akin to classical anomaly detection methods. For example, we aligned

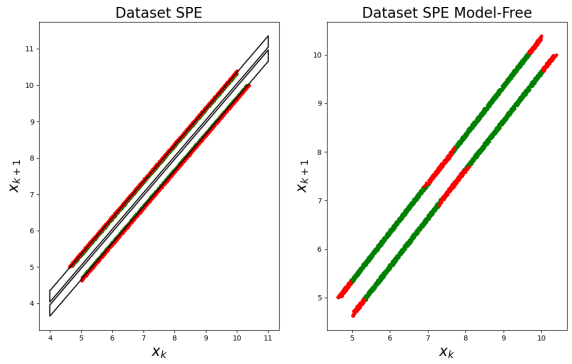

Fig. 5. Classification results of the H-Classifier (left panel) and One-Class SVM classification results (right panel) for the **SPE** data set. In the H-Classifier results the polytopes (black lines) are added. Note that the results are very close to the boarder and the red outlier dot overlap the green inlier.

it with techniques such as one-class support vector machines. In the results we can see that this has as expected the result to make the classification process faster in comparison to the more formal approach of [1], [2] as well as the more classical anomaly detection approaches.

TABLE I
INLIER/OUTLIER PREDICTION RESULTS FOR THE DIFFERENT DATA SETS OF THE WATER TANK. THE Δ USED FOR THE "FORMAL METHOD" APPROACH RESULTS WAS 0.001. THE NUMBER OF DATA VECTORS IN EACH DATA SET WAS 99000.

Method	Data Set	Inliers	Outliers
Formal Method	**C**	99000	0
Formal Method	**SPE**	72567	26433
Formal Method	**LPE**	51579	47421
Formal Method	**N**	71574	27426
OC-SVM	**C**	89103	9897
OC-SVM	**SPE**	84047	14953
OC-SVM	**LPE**	62515	36485
OC-SVM	**N**	85942	13058
H-Classifier	**C**	99000	0
H-Classifier	**SPE**	71973	27027
H-Classifier	**LPE**	51480	47520
H-Classifier	**N**	70696	28304

TABLE II
TIMING RESULTS FOR THE WATER TANK FOR THE DIFFERENT DATA SETS.

Method	Data Set	Overall	Per
Formal Method	**C**	0.925s	$9\mu s$
Formal Method	**SPE**	0.74s	$7\mu s$
Formal Method	**LPE**	0.821s	$8\mu s$
Formal Method	**N**	0.748s	$8\mu s$
OC-SVM	**C**	22.056 s	$200\mu s$
OC-SVM	**SPE**	22.059 s	$200\mu s$
OC-SVM	**LPE**	22.075 s	$200\mu s$
OC-SVM	**N**	22.072 s	$200\mu s$
H-classifier	**C**	0.431 s	$4\mu s$
H-classifier	**SPE**	0.546 s	$5\mu s$
H-classifier	**LPE**	0.627 s	$6\mu s$
H-classifier	**N**	0.578 s	$5\mu s$

TABLE III
INLIER/OUTLIER PREDICTION RESULTS FOR THE DIFFERENT DATA SETS OF THE $\Sigma - \Delta$ MODEL. THE Δ USED FOR THE "FORMAL METHOD" APPROACH RESULTS WAS 0.001. THE NUMBER OF DATA VECTORS IN EACH DATA SET WAS 100000. HE Δ USED FOR THE H-CLASSIFIER WAS 0.5

Method	Data Set	Inliers	Outliers
Formal Method	**C**	100000	0
Formal Method	**SPE**	98552	1448
Formal Method	**LPE**	25431	74569
Formal Method	**N**	97615	2385
OC-SVM	**C**	90000	10000
OC-SVM	**SPE**	84831	15169
OC-SVM	**LPE**	38738	61262
OC-SVM	**N**	86867	13133
H-Classifier	**C**	99985	15
H-Classifier	**SPE**	99904	96
H-Classifier	**LPE**	34597	65403
H-Classifier	**N**	99336	664

TABLE IV
TIMING RESULTS FOR THE DIFFERENT DATA SETS OF THE $\Sigma - \Delta$ MODEL.

Method	Data Set	Overall	Per
Formal Method	**C**	5.561 s	$55\mu s$
Formal Method	**SPE**	5.607 s	$56\mu s$
Formal Method	**LPE**	9.375 s	$93\mu s$
Formal Method	**N**	5.76 s	$57\mu s$
OC-SVM	**C**	25.578 s	$255\mu s$
OC-SVM	**SPE**	25.589 s	$256\mu s$
OC-SVM	**LPE**	25.353 s	$253\mu s$
OC-SVM	**N**	25.574 s	$256\mu s$
H-classifier	**C**	0.577 s	$6\mu s$
H-classifier	**SPE**	0.646 s	$6\mu s$
H-classifier	**LPE**	0.819 s	$8\mu s$
H-classifier	**N**	0.661 s	$7\mu s$

A. Real Water Tank

We have seen the results for the synthetically generated data sets. The question is the applicability of the approach to the real world. For this we did some testing on a measurement of a real-world test water tank system. This setup had a water tank with two pumps that were either pumping water into the tank or out. The underlying data stems from a measurement of the system for over an hour with a measurement frequency of 1Hz. In Fig. 6, the top most graph, the resulting trajectory can be seen. Clearly visible are the measurement inaccuracies that come with real world data. As done for the previous data sets, we transform this trajectory into the transition space. We can see the result of this transformation in Fig. 6 center, 6 bottom. In this depiction we can again clearly see the number of outliers due to measurement errors and other disturbances of the system. We applied as for the previous data sets the classification with the original approach, the AD approach which here is again a one class svm and the novel H-classifier approach. The results of the classification can be seen in Tab. V.

A more graphical representation of the classification results for the H-classifier can be seen in Fig. 6 center. We see very similar results for the H-Classifier and the original approach. Since the graphical results look very similar we omitted to represent them here. This similarity is due to the fact that they fundamentally base on the very same things, the polytopes

TABLE V
CLASSIFICATION RESULTS OF THE REAL WORLD WATER TANK SYSTEM.

Method	Inliers	Outliers
Formal Method	3537	868
OC-SVM	3851	554
H-Classifier	3498	907

defined by the leafs of the decision diagram (AACDD).

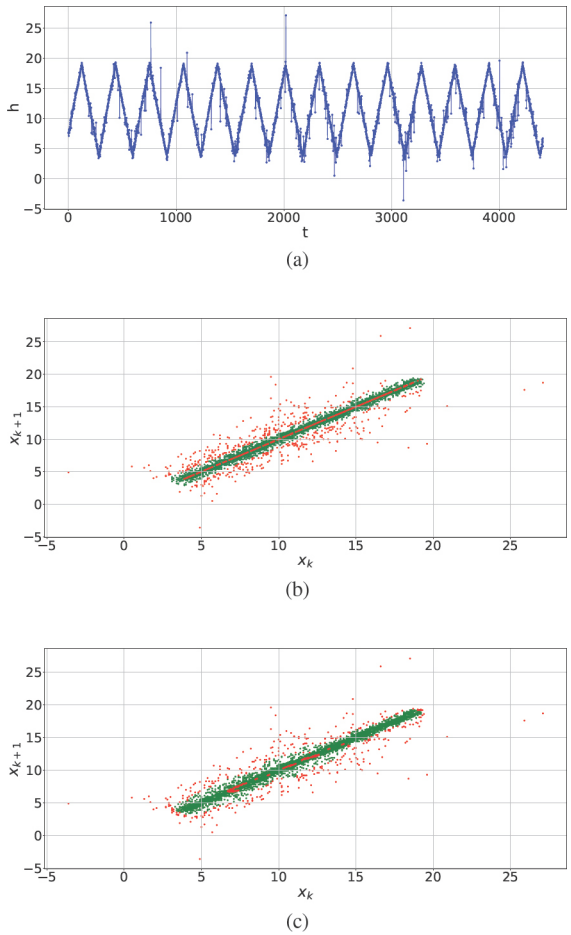

Fig. 6. Panel (a) illustrates a measurement trajectory of the real water tank system. Outliers, potentially caused by measurement errors or external disturbances, are clearly visible. In panel (b) the figure depicts the classification results for the water tank problem obtained using the H-Classifier. The classifier accurately identifies all outliers attributed to measurement errors and external disturbances. In constrast panel (c) shows the classification results of the one class support vector machine. Similarly to the H-Classifier the OC-SVM filters out most of the large outliers but it is from a visual analysis less strict than the H-Classifier.

In Fig. 6, bottom, the results of the one class svm classification can be seen. As visible in the results table and in visible comparing both figures, the one class SVM is more tolerant than the other approaches. All the results presented were more or less in the same margin of classification into inliers or outliers. The question that can be asked is which results are more ”correct”. While we cannot answer this question here, we want to give an interesting final observation. In the original and the H-Classifier approach presented in this paper we can actually explain why we accept or reject an observed transition. We can clearly state that when we accept a transition it is due to the fact that it adheres to our dynamics and also the properties defined for our system. We can even go so far as to give bounds on the parameters and discrete states the system has to be in. This ability to explain the result is lacking in most AD approaches.

V. CONCLUSION

In conclusion, the primary objective of developing a process that integrates formal specification, characteristic of a runtime verification (RV) approach, while ensuring the efficiency of the classification process, typical of an anomaly detection (AD) approach, was largely achieved. The results indicate that, although the method did not fully align with the intended correctness criteria, it demonstrated a close alignment with its foundational approach. Therefore, this goal was only partially met. However, the efficiency target was fully realized, with the classification process significantly outperforming other approaches in terms of evaluation speed. Overall, the H-Classifier successfully accomplished most of its intended goals, representing a strong balance between correctness and efficiency in this context.

REFERENCES

[1] H. Heermann and C. Grimm, “Runtime verification of hybrid systems with affine arithmetic decision diagrams,” in *Methoden und Beschreibungssprachen zur Modellierung und Verifikation von Schaltungen und Systemen*, ITG-Fb. 309: MBMV 2023, 2023. isbn: 978-3-8007-6065-7.

[2] H. Heermann, J. Koch, and C. Grimm, “Reliable and real-time anomaly detection for safety-relevant systems,” in *Proceedings of DVCON 2024*, 2024. Accepted for presentation.

[3] Z. Manna and A. Pnueli, *Temporal verification of reactive systems: safety*. Springer Science & Business Media, 2012.

[4] R. Caldas, J. A. P. García, M. Schiopu, P. Pelliccione, G. Rodrigues, and T. Berger, “Runtime verification and field testing for ros-based robotic systems,” 2024.

[5] O. Maler and D. Nickovic, “Monitoring temporal properties of continuous signals,” in *International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems*, pp. 152–166, Springer, 2004.

[6] O. Maler, “Some thoughts on runtime verification,” in *Runtime Verification: 16th International Conference, RV 2016, Madrid, Spain, September 23–30, 2016, Proceedings 7*, pp. 3–14, Springer, 2016. doi: 10.1007/978-3-319-46982-9_1.

[7] S. Mitsch and A. Platzer, “Modelplex: Verified runtime validation of verified cyber-physical system models,” *Formal Methods in System Design*, vol. 49, pp. 33–74, 2016.

[8] V. Herdt, H. M. Le, D. Große, and R. Drechsler, “Compiled symbolic simulation for systemc,” in *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–8, 2016.

[9] R. Babaee, A. Gurfinkel, and S. Fischmeister, “Prevent: A Predictive Run-Time Verification Framework Using Statistical Learning,” in *Software Engineering and Formal Methods* (E. B. Johnsen and I. Schaefer, eds.), (Cham), pp. 205–220, Springer International Publishing, 2018.

[10] B. Schölkopf, A. J. Smola, R. C. Williamson, and P. L. Bartlett, “New Support Vector Algorithms,” *Neural Computation*, vol. 12, pp. 1207–1245, 05 2000.

[11] L. Tran, L. Fan, and C. Shahabi, “Distance-based outlier detection in data streams,” *Proc. VLDB Endow.*, vol. 9, p. 1089–1100, aug 2016.

[12] I. C. Paschalidis and Y. Chen, “Statistical anomaly detection with sensor networks,” *ACM Trans. Sen. Netw.*, vol. 7, sep 2010.

[13] Z. Z. Darban, G. I. Webb, S. Pan, C. C. Aggarwal, and M. Salehi, “Deep learning for time series anomaly detection: A survey,” 2024.

[14] C. Zhou and R. C. Paffenroth, “Anomaly detection with robust deep autoencoders,” in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD ’17, (New York, NY, USA), p. 665–674, Association for Computing Machinery, 2017.

[15] S. Lin, R. Clark, R. Birke, S. Schönborn, N. Trigoni, and S. Roberts, "Anomaly detection for time series using vae-lstm hybrid model," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4322–4326, 2020.

[16] J. Sipple and A. Youssef, "A general-purpose method for applying explainable ai for anomaly detection," in *Foundations of Intelligent Systems* (M. Ceci, S. Flesca, E. Masciari, G. Manco, and Z. W. Raś, eds.), (Cham), pp. 162–174, Springer International Publishing, 2022.

[17] Z. Li, Y. Zhu, and M. van Leeuwen, "A survey on explainable anomaly detection," 2023.

[18] S. Cuéllar, M. Santos, F. Alonso, E. Fabregas, and G. Farias, "Explainable anomaly detection in spacecraft telemetry," *Engineering Applications of Artificial Intelligence*, vol. 133, p. 108083, 2024.

[19] M. Althoff, "Checking and establishing reachset conformance in cora 2023," in *Proc. of 10th International Workshop on Applied Verification of Continuous and Hybrid Systems*, 2023. doi: 10.29007/5v1g.

[20] M. V. A. Andrade, J. L. D. Comba, and J. Stolfi, "Affine arithmetic," in *INTERVAL'94*, INTERVAL'94, 1994.

[21] R. Penrose, "A generalized inverse for matrices," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 51, pp. 406 – 413, 1955.

[22] M. Althoff, O. Stursberg, and M. Buss, "Verification of uncertain embedded systems by computing reachable sets based on zonotopes," *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 5125–5130, 2008. 17th IFAC World Congress.

[23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[24] J. Lala and R. Harper, "Architectural principles for safety-critical real-time applications," *Proceedings of the IEEE*, vol. 82, no. 1, pp. 25–40, 1994.