Using Behavior Models for Anomaly Detection in Hybrid Systems

Asmir Vodenčarević, Hans Kleine Büning
Knowledge-Based Systems Research Group
University of Paderborn
33098 Paderborn, Germany
{asmir.vodencarevic, kbcsl}@upb.de

Oliver Niggemann, Alexander Maier
inIT – Institut Industrial IT
Hochschule Ostwestfalen-Lippe
32657 Lemgo, Germany
{oliver.niggemann, alexander.maier}@hs-owl.de

***Abstract*—The importance of safety and reliability in today's real-world complex hybrid systems, such as process plants, led to the development of various anomaly detection and diagnosis techniques. Model-based approaches established themselves among the most successful ones in the field. However, they depend on a model of a system, which usually needs to be derived manually. Manual modeling requires a lot of efforts and resources.**

This paper gives a procedure for anomaly detection in hybrid systems that uses automatically generated behavior models. The model is learned from logged system's measurements in a hybrid automaton framework. The presented anomaly detection algorithm utilizes the model to predict the system behavior, and to compare it with the observed behavior in an online manner. Alarms are raised whenever a discrepancy is found between these two. The effectiveness of this approach is demonstrated in detecting several types of anomalies in a real-world running production system.

***Keywords*—hybrid system, behavior model, anomaly detection, learning hybrid automata**

I. Introduction

Hybrid systems are dynamical systems that consist of discrete controls embedded in the continuous environments ([1]). Due to their large representation in the real-world, especially in the process and the automotive industry, the significance of hybrid systems cannot be overemphasized. The major hybrid system characteristic is a permanent interaction between the discrete control signals (given by computer controller) and the continuous process variables (that represent the physical plant). This interaction is performed through the set of sensors, actuators, and communication networks.

Safety and reliability of hybrid systems are of the great importance. Ensuring a high level of these properties is becoming increasingly hard, as the hybrid systems tend to become more complex due to the employment of the small, embedded components and distributed architectures. Systems such as complex process plants are usually accompanied by nonlinearities, different types of noises and external disturbances. Moreover, they can consist of various components, i.e. hydraulic, pneumatic, electric, mechanical, and chemical (e.g. chemical plant). Having such complexity makes hybrid systems very susceptible to anomalies. The anomaly is considered to be any significant deviation of the system's behavior from the expected one. In order to compare these behaviors, a model of the typical (correct) behavior is required. If such model is available, it is used by the anomaly detection algorithm, together with both the input and the output signals of the system. This is shown in figure 1. Based on provided inputs, the model internally computes (estimates) the expected outputs, which are then compared with the real, observed output values. Due to the dual nature of hybrid system's dynamics, the anomaly detection procedure needs to include the following:

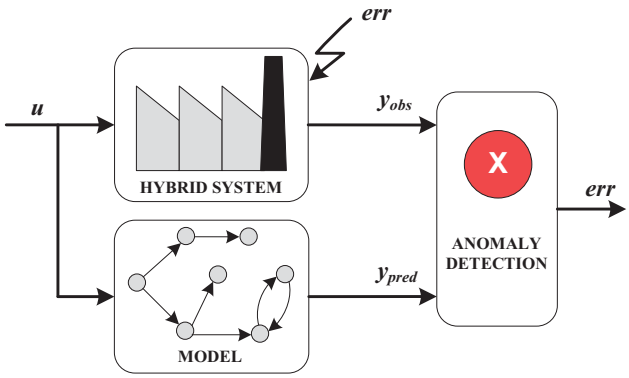

Fig. 1. Model-based anomaly detection

1) Monitoring of control signals: The correctness of observed sequences and timings of control signals has to be constantly checked.
2) Monitoring of process variables: The values of the measurable process variables need to be checked with regard to some acceptable (predefined) tolerances.

When a discrepancy is found in either of these two cases, an alarm is raised signalizing a detected anomaly.

Model-based anomaly detection techniques belong to a wider field of model-based diagnosis ([2], [3]). Model-based approaches are also used in other applications, e.g. in optimization, testing and design. Although anomaly detection presents a big challenge by itself, the major obstacle of all model-based approaches remains the so-called "modeling bottleneck". The model of a system is required, which is in most cases hardly obtainable. Manual modeling has to deal with several serious problems:

978-1-4577-0746-9/11/$26.00 ©2011 IEEE

- Expert knowledge: The sufficient knowledge about the system needs to be available. The system's variables and their mathematical relations have to be known.
- Internal variables: In order to infer the laws of system's dynamics, the additional measurements of some internal variables are often required. Usually, some system's internals are not observable.
- Resources: Manual modeling requires both human and budget resources.
- Model updates: Manually created models must be manually updated. In order to preserve the model adequacy and accuracy, updates are needed whenever something is changed in the system.

A promising alternative to manual modeling are the data-driven approaches that come from the field of learning theory ([4]). In this paper behavior models are identified using the HyBUTLA algorithm (presented in [5]). It is, to the best of our knowledge, the first algorithm capable of automatically learning models of hybrid systems in the modeling formalism of hybrid automata ([6]). This paper further presents the anomaly detection algorithm ANODA, which utilizes such automatically learned hybrid automaton models for finding discrepancies between the observed and the predicted behavior. In the given example, the behavior model is identified for a component of a real-world small process plant. Comprehensive results are then given for six different fault types that this model was able to detect.

II. State of the art

A. Model-Based Diagnosis of Hybrid Systems

Like all running entities, hybrid systems are also prone to faults. These can be classified into permanent and non-permanent, based on the duration of their effects ([7]). Permanent faults cause the system to remain in a faulty state (e.g. broken pump), while non-permanent ones disappear after some time (e.g. some external disturbance disappears). Based on a location of the occurrence, faults can appear in continuous parts (e.g. pump not working with nominal power) and discrete parts (controller hardware and software errors) of the system ([8]). Faults can also be abrupt (with immediate effects) or subtle (causing slow system degradation). Particularly difficult task is diagnosing multiple and interactive faults, especially in the cases of error propagation ([9]).

To deal with these issues, many anomaly detection and isolation techniques have been developed. One approach, aiming at diagnosing sensor failures, is called hardware redundancy ([10]). Redundant sensors are used, and the anomaly is detected when discrepancy is found in their measurements. This approach is very limited and expensive. Another model-free approach is based on the expert systems ([7]). Domain knowledge is used to compile the diagnosis rules, which enumerate symptom-fault relationships. This approach presumes the existence of the expert knowledge, which is usually hard and costly to obtain. The issues with such heuristic approaches are their task-dependency, the manual update of diagnosis rules, and the difficult tracking of interacting faults ([11]).

To address the shortcomings of these approaches, model-based diagnosis (MBD) techniques have been introduced (see [2], [3]). MBD exploits the analytical redundancy between the model and the sensor data (the system). Here the model's output is used as a reference to the output of a system (as shown in figure 1). MBD has the following pluses: it is generally applicable, fast enough for medium-sized problems, easily maintainable, directly supports reusability, and has a well-founded theory ([12]).

Significant work has been done in applying MBD to discrete event systems, i.e. the systems characterized by discrete changes of state values caused by the asynchronous discrete events. Here discrete-event models such as finite state automata, timed automata or Petri networks are used (see e.g. [13], [14], [15], [16]). Similarly, a lot of research is conducted in the MBD of strictly continuous systems using continuous models (e.g. bond graphs, parameter estimation etc.). For an overview of these approaches check [17]. One of the most used modeling formalisms in MBD of hybrid systems is the hybrid automaton ([6]), which is formally defined in section III-A. Since hybrid systems are the subject of both computer science and engineering, the methods from both areas can be found. Kalman filter-based observers are often used in MBD (e.g. [8], [18], [19] and [20]). Given measurements that contain Gaussian white noise with zero mean, Kalman filter predicts the output signal values and estimates their uncertainties. Another popular approach is particle filtering (see e.g. [21]). In contrast to Kalman filter, particle filter does not presume Gaussian noise distribution, and given a larger number of samples, it has much higher accuracy. Bayes approaches to estimation of monitored output variables also exist (e.g. [18]).

All mentioned approaches to model-based diagnosis of hybrid systems have been more or less successful in detecting and isolating anomalies by estimating the current state of the system. This estimation is compared with the observation, which is then associated with normal or abnormal behavior. Common diagnosis problems, like diagnosing multiple faults, modeling nonlinear dynamics, dealing with noisy data, and having erroneous observations are in many cases avoided by imposing idealized assumptions ([22]). However, the major drawback in applying these approaches to the real-world systems is that they all assume the system's model is already given (or manually created), so they only deal with the estimation of its parameters. In reality the system is usually hard to model manually, but a lot of system logs (sensor measurements) are available. In this case the machine learning approaches to automatic model generation are much more promising. The targets of these approaches are to learn both the model structure and its parameters from observations, but also to find an effective way to incorporate domain knowledge into the model learning algorithm, as it does not make sense to learn what is already known about the system.

Since many diagnosis approaches use various automata models, we give their overview together with the state of the art automata learning algorithms in the following section.

B. Learning Behavior Models

There exist a number of automata formalisms that can model different types of behaviors. Strict discrete behavior of the discrete-event systems is modeled as a deterministic finite automaton (DFA). Discrete behavior with important timing information can be represented using timed automaton ([23]). Continuous dynamics is usually modeled using differential equations ([24]). Since hybrid systems contain all these behavioral aspects, one of their most suitable modeling formalisms is hybrid automaton ([6]). It essentially represents timed automaton capable of describing continuous dynamics by a set of differential (or arbitrary) equations added to the automaton states. Check [25] for a detailed overview of the various automata formalisms.

In general, the automata learning algorithms can use both positive examples (i.e. measurements from normal system behavior) and negative examples (coming from a faulty behavior). In the real production systems the number of available negative examples is always significantly smaller, thus the algorithms capable of learning behavior models using dominantly (or only) positive examples have wider applicability.

One of the most popular DFA learning algorithms that use both types of examples is Angluin's L* ([26]). It runs in an online manner, which means that it can request additional learning samples during its runtime. In contrast, the offline algorithms can rely only on the given datasets, already recorded in the system's past. The example of the offline DFA learning algorithm is given by Gold ([27]). Algorithms ALERGIA ([28]) and MDI ([29]) learn probabilistic DFAs in an offline manner, but using only positive examples. Common for these algorithms is that they first build a so-called prefix tree, i.e. the tree structure that represents every example as a path from the initial state, to one of the final states. Once having a prefix tree, these algorithms proceed by checking compatibility between the states, using different compatibility criteria. The pairs of the states that are found to be compatible are merged.

Several timed automata learning algorithms that use only positive or both positive and negative examples, are given by Verwer in [30]. Besides of being the first algorithms for learning timing information, they also introduce another novelty - the splitting step. The states, whose division creates the automaton subtrees that are significantly different (with regard to some criteria), are split.

Although hybrid automata have huge significance in modeling hybrid systems, there is a serious lack in the hybrid automata learning algorithms. Despite the dominant use of hybrid automata in technical disciplines, the first attempts toward learning them automatically surprisingly came from the biological application. The algorithm given in [31] is learning the dynamics of the *action potentials*, i.e. the electrical signals of certain types of cells in the living organisms. However, this algorithm does not account for discrete signals. In [5] we presented the HyBUTLA algorithm, that is to the best of our knowledge the first hybrid automata learning algorithm that can model both discrete and continuous dynamics.

III. Automated learning of behavior models

Generally speaking, every hybrid system can be seen as a set of interconnected components that work in parallel and often asynchronously, where each individual component comprises strictly sequential behavior. Such components are defined by a parallelism model of the system's structure ([5]). There are no algorithms for identifying this model automatically. However, parallelism model is usually generated using an expert knowledge in the system's (plant's) planning and design phases, and it is based on standards such as AutomationML (standard for factory systems, see [32]). The target of the algorithm given in this section is the automatic learning of behavior models for the hybrid system's components (i.e. the behavior functions b_C defined below). It is assumed that the measurements of control and process variables (data for learning) are already given. In the following, the formal definition of a component is given.

Definition 1. *(Component) A component C is defined by a behavior function $b_C : \mathbb{R}^{m+1} \rightarrow \mathbb{R}^n, n, m \in \mathbb{N}$. b_C is a function over m input variables and over time and it returns n output variables.*

A. Formal Framework

As stated in the previous sections, hybrid automata are chosen as a modeling formalism for learning behavior models of hybrid systems. The expressiveness of this formalism enables the modeling of both discrete and continuous dynamics, with the inclusion of timing and probabilistic information (i.e. modeling the times and the probabilities of signal changes). Furthermore, hybrid automata are easily visualized and understood. The formal definition is given in the following.

Definition 2. *(Hybrid Automaton) The hybrid automaton (HA) is a tuple $A = (S, s_0, F, \Sigma, T, \Delta, Num, \Theta)$, where*

- *S is a finite set of states, $s_0 \in S$ is the initial state, and $F \subseteq S$ is a distinguished set of final states,*
- *Σ is the alphabet. For a component, Σ equals the set of events that trigger transitions.*
- *$T \subseteq S \times \Sigma \times S$ gives the set of transitions. E.g. for a transition $\langle s, a, s' \rangle$, $s, s' \in S$ are the source and destination state and, $a \in \Sigma$ is the trigger event.*
- *A function $Num : T \rightarrow \mathbb{N}$ counts the number of observations in which a transition has been used in the system's past. Num can be used to compute transition probabilities p for a transition $\langle v, a, w \rangle \in T$: $p(v, a, w) = \frac{Num(v,a,w)}{\sum_{(v,b,w') \in T, b \in \Sigma, w' \in S} Num(v,b,w')}$.*
- *A set of transition timing constraints Δ with the elements $\delta : T \rightarrow I$, where I is the set of intervals. δ refers to the time spent since the last event occurred. It is expressed as a time range or as a probability density function (PDF), i.e. as probability over time. Continuous time evolution is recorded by the set of clocks X.*
- *A set of functions Θ with elements $\theta_s : \mathbb{R}^n \rightarrow \mathbb{R}^m, \forall s \in S, n, m \in \mathbb{N}$. I.e. θ_s is the function computing signal value changes within state s. For learning these functions, all other networks signal values and time can be used.*

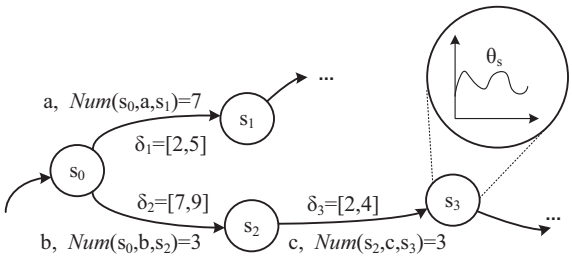

Fig. 2. A hybrid automaton example

It is important noting that comparing to the original definition ([6]), the formalism is here simplified and accommodated to the learning task. The automaton presented here has only one clock counting time between the events, thus only the relative timing is used (the clock is always reset with the appearance of the new event). Every transition is associated with a probability, and furthermore arbitrary functions are allowed to approximate the continuous behavior, in contrast to differential equations only. An example of hybrid automaton is illustrated in figure 2. The following elements can be recognized with regard to definition 2: $s_0, s_1, s_2, s_3 \in S$, $a, b, c \in \Sigma$, $\delta_1, \delta_2, \delta_3 \in \Delta$, and $\theta_s \in \Theta$.

B. The Learning Algorithm

Behavior models are identified using an offline HyBUTLA algorithm for learning hybrid automata (presented in [5]). Similar to ALERGIA and MDI algorithms (see section II-B), HyBUTLA first builds a prefix tree acceptor (PTA) from the given observations, and then merges the compatible states using specific compatibility criteria.

At first the sensor measurements of discrete and continuous variables are taken from a running system. Data is recorded in the several examples, i.e. the running cycles of a system. In the current state of the algorithm, the events Σ that trigger the state transitions (see definition 2) are defined as the changes in discrete variables. This is illustrated in figure 3, where the changes in two control signals trigger the transitions between the states. Transition timing is the timing of corresponding signal change relative to the timing of previous signal change (only the relative timing is used).

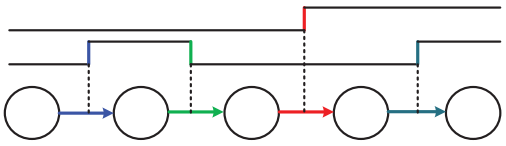

Fig. 3. Identification of the hybrid states

The HyBUTLA learning algorithm is given in figure 4. First, the common events of different examples are combined, yielding the prefix tree acceptor (line 1). An illustrative example of the PTA created using four positive examples is given in figure 5. Relevant continuous signals are approximated in every state using regression methods such as multiple linear regression or neural networks (line 2).

Given:
(1) Hybrid system's component C, its events Σ
(2) Measurements $\mathcal{S} = \{\mathbf{S}_0, \ldots, \mathbf{S}_{n-1}\}$ where $\mathbf{S}_i = (\Sigma \times \mathbb{R})^p, p \in \mathbb{N}$ is one sequence of p events over time (i.e. one measurement or one scenario)
Result: C's function b_C defined by an automaton
(1) Build prefix tree $A = (S, s_0, F, \Sigma, T, \delta, Num, \Theta)$ based on $\mathcal{S}$. A is a hybrid automaton according to definition 2.
(2) $\forall s \in S$ learn $\theta_s \in \Theta$ using continuous data from $\mathcal{S}$.
(3) **for all** $v, w \in S$ in a bottom-up order **do**
(4) **if** compatible(v, w) **then**
(5) merge(v, w)
(6) **end if**
(7) **end for**

Fig. 4. Hybrid automata identification algorithm HyBUTLA.

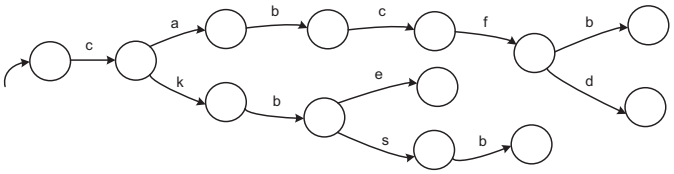

Fig. 5. An example of a prefix tree acceptor

In the real-world hybrid systems such as process plants, the number of the obtained states can be very large. In order to reduce it, the merging step is introduced. The state compatibility is checked using a bottom-up order, in a breadth-first-like manner (lines 3-4). The compatibility criterion consists of the following three checks:

1) Transition probabilities: The probabilities of the in-coming transition events are tested for a pair of states, which are candidates for merging. This test is implemented as the Hoeffding Bound (see [28]).
2) Transition timings: The transition timings can be expressed as the probability density functions - PDFs (see definition 2). It is checked if PDFs of two in-coming transitions with the same event are overlapping (i.e. if their timings belong to the same time cluster).
3) Function approximations: The similarity of the functions that approximate continuous behavior in two states is checked. This test naturally depends on the type of the used functions. For example, when the multiple linear regression is used, it is checked if the difference between the corresponding regression coefficients is not larger than some predefined threshold (e.g. 10%).

When all three criteria are satisfied for two states, they are merged (line 5). For the newly created state, the new functions that approximate continuous dynamics are learned using the portions of continuous data from both previous states.

IV. Anomaly detection in hybrid systems

Automatically generated behavior models are used for model-based anomaly detection in hybrid systems. Generally speaking, the purpose of the anomaly detection is to find the anomalous (i.e. unlike, unexpected) objects, usually referred to as the outliers. To this aim, several formal definitions are given.

Definition 3. *(Path Through the Automaton) Let* $A = (S, s_0, F, \Sigma, T, \Delta, Num, \Theta)$ *be a hybrid automaton, according to definition 2. A path P through the automaton is defined as a sequence of transitions* $P \subseteq T$.

Definition 4. *(Observation) An observation of the hybrid system is defined as a tuple* $o = (a, t, \boldsymbol{u}, y)$*, where:*

- $a \in \Sigma$ *is the trigger event in the system*
- t *is a relative time value (relative to the last control signal change)*
- $\boldsymbol{u}$ *is the vector of the continuous signals' input values that are used for predicting the output*
- y *is the observed value of the monitored output signal, which is compared with the predicted output.*

Various causes of the anomalies exist. For example, every considered dataset contains so-called natural (normal) variation. The differences that may be detected between the objects of interest can fall into this variability. Anomaly detection algorithms often need to be adjusted to account for this phenomenon. Another issue is related to the observation's accuracy. Imperfections in sensors, measurement noise, external disturbances or a human factor can often insert errors in data, which can later on be recognized as anomalies. Such errors should be eliminated in the data preprocessing stage. We are however interested in detecting those anomalies, that are caused by the faulty operating conditions (process) appearing in a hybrid system during its runtime. These are defined by a statistician Douglas Hawkins ([33]).

Definition 5. *(Hawkins' Definition of an Outlier) An outlier is an observation that differs so much from other observations as to arouse suspicion that it was generated by a different mechanism.*

The usage of the learned model in detecting such conditions was already shown in figure 1. In figure 6 the anomaly detection algorithm ANODA is given. It runs in an online manner, which means that it receives observations periodically (e.g. every second) from the running system. Three types of anomalies can be detected using the ANODA algorithm:

Wrong control sequence: This anomaly appears when the sequence of control signals (events) is incorrect. More concretely, it occurs if there is no automaton transition from the current state, which should be triggered by the observed event (i.e. the probability of triggering a non-existing transition is 0). For example, while heating some raw material in a container, the succeeding event should be 'heater off', which is triggered when a certain temperature is reached. If the event 'empty container' occurs instead, the wrong control sequence is detected. This is illustrated in figure 7, where the anomaly is represented by a non-existing path through the automaton, i.e. $P_{automaton} \neq P_{observed}$.

Given:
(1) Hybrid Automaton (HA) $A = (S, s_0, F, \Sigma, T, \Delta, Num, \Theta)$ (according to definition 2)
(2) $o = (a, t, \boldsymbol{u}, y)$ is an observation according to definition 4. ANODA algorithm receives such observations periodically.
(3) ε is a predefined alarm threshold for monitoring continuous output signal (e.g. 5% or 10%)
(4) $[a, b], a > 0$ is a measurement range for monitored output variable y. This range us used as a reference for discrepancy calculation.
(5) $s_{curr} := s_0$, at first, the current state is the initial state
Result: detected anomaly (if there exists one), otherwise "OK"

```
(1)  T' = {e = (s_curr, a, *) ∈ T}, s_curr ∈ S, a ∈ Σ, where * is an arbitrary element
(2)  if T' = ∅
(3)    return anomaly: unknown event
(4)  else
(5)    while isNotEmpty(T') do
(6)      if t ∈ δ(e = (s_curr, a, s_new)) then
(7)        s_curr := s_new
(8)        y_p = θ_{s_new}(u)
(9)        diff = (y_p − y) * 100/(b − a)
(10)       if y = 0 and y_p ≠ 0 then
(11)         return anomaly: zero value of the signal
(12)       else if diff ≥ ε then
(13)         return anomaly: signal drop
(14)       else if diff ≤ −ε then
(15)         return anomaly: signal jump
(16)       else
(17)         return OK
(18)       end if
(19)     else if t < min(δ(e)) then
(20)       return anomaly: event too early
(21)     else if t > max(δ(e)) then
(22)       return anomaly: event too late
(23)     end if
(24)   end while
(25) end if
```

Fig. 6. Anomaly detection algorithm ANODA

For every observation, ANODA algorithm first defines a subset $T' \subseteq T$ with all transitions that can be triggered by the observed event a from the current state s_{curr} (line 1). If there are no such transitions, an anomaly "unknown event" is detected (lines 2-3).

Wrong event timing: The timing of each occurring transition is compared with learned timing constraints (i.e. time intervals δ, see definition 2). Wrong event timing is detected when the event comes before or after the corresponding time interval. Depending on the application, wrong event timing can be reported in various ways. For example, if the event

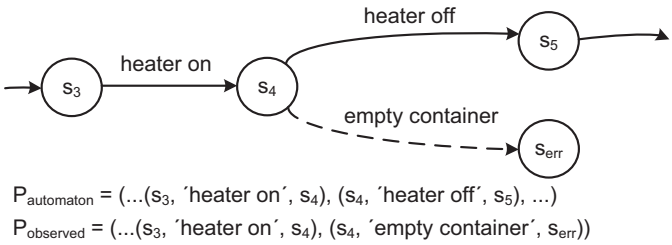

Fig. 7. Wrong control sequence example

timing is "very close" to the interval borders (e.g. less than a second earlier or later), only a warning could be signaled. However, if the event timing is "far away" from the allowed time interval, an alarm could be raised.

If T' is not an empty set, ANODA algorithm iterates through its transitions (line 5) and checks if there exists the one whose time interval δ includes observed event time t (line 6). When such transition is not found, the anomaly "event too early" or "event too late" is signaled (lines 19-20 or 21-22, respectively).

Process variable out of range: The third anomaly type that can be detected using hybrid automaton model is related to the continuous output variables. The thresholds (tolerances) are defined for all monitored variables. An anomaly is detected whenever the difference between predicted and observed values exceeds a given threshold.

Assuming correctness of the observed event and its timing, the newly entered state s_{new} is marked as the current state (line 7 in figure 6), and the expected output value y_p is estimated (line 8). Then, its difference from the observed value y is calculated in relation to the given variable measurement range $b - a$ (line 9). The following types of anomalies can be detected: "zero value of the signal" (lines 10-11), "signal drop" (lines 12-13), and "signal jump" (lines 14-15). If none of them is signaled, ANODA algorithm returns "OK" indicating that the given observation is anomaly-free. This algorithm can be easily extended to monitor multiple output variables.

V. An Example: A Small Process Plant

A. Lemgo Model Factory

In order to give an example of using behavior models for anomaly detection in hybrid systems, the Lemgo Model Factory (LMF) is used. This hybrid system represents a plant for storing, transporting, processing and packing bulk material (corn). It has a modular design, i.e. it consists of eight modules, namely: a storage system, a few transportation systems, a weighting station, a bottle filling mechanism, a production facility, a product packing system, a bearing robot, and a lid robot. These modules comprise a number of components including: distributed PLCs, communication buses (e.g. PROFINET), a popping machine, several conveyor belts etc. Such components are identified by manually created parallelism model, mentioned in section III. Approximately 250 various discrete and continuous signals can be recorded. The Lemgo Model Factory is shown in figure 8.

Fig. 8. The Lemgo Model Factory

In this example, we focus our attention on the popping machine. Here 12 variables were recorded, including 6 discrete (binary) control signals, and 6 continuous process variables. Data sampling rate was 1 Hz. The popping machine receives a raw material (corn), and activates the heater that heats the air. Hot air is then blown toward the raw material container. Popcorn is obtained as a result of this processing.

Described process represents one production example. For learning the popping machine behavior model, we used data recorded in 12 production examples that are classified as positive (taken under normal operating conditions). Once the model was learned using HyBUTLA algorithm (see figure 4), additional test (negative) examples are given to the anomaly detection algorithm ANODA (see figure 6) that was able to detect the anomalies with significant accuracy.

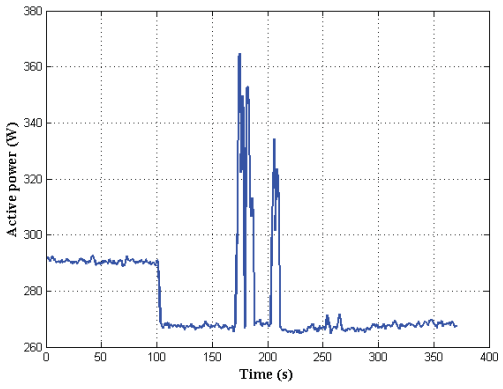

Fig. 9. Popping machine output signal

Five out of six continuous variables were identified as predictors (inputs) and one variable, namely the active power of the popping machine's heater, represents the output. Its measurement range is 0-1150W. The model predicts the outputs, while the anomaly detection algorithm ANODA detects their discrepancies from the observed signal values. Figure 9 gives the time diagram of the output signal, taken in one of the recorded production examples (with the mean value and the standard deviation 276.97 $\pm$ 16.51 W).

B. Learned Behavior Models

HyBUTLA algorithm given in section III was used to build a popping machine hybrid automaton model. Initially, the prefix tree acceptor (PTA) was constructed, and than the compatible states were merged. The key goals were to obtain the smallest automaton possible (to reduce the PTA size as much as possible), and to approximate state continuous dynamics with the highest accuracy possible. To this aim, we used both multiple linear regression (MLR) and neural networks (NN), as regression methods. For evaluating the approximation accuracy, the coefficient of determination R^2 was used. R^2 expresses the percent of data variability that is accounted for by the regression line.

Model	Size(states)	Reduction(%)	MLR R^2(%)	NN R^2(%)
PTA	52	0	68.6	99.5
HA	19	63.5	67.2	98.8

TABLE I
PROPERTIES OF PTA AND LEARNED BEHAVIOR MODEL

Although several models were created by varying the similarity threshold for multiple linear regression coefficients (see [5] for more results with alike output signal), here we present only the final model (HA), used by ANODA algorithm. The properties of both PTA and this HA model are summarized in table I. It can be seen that by merging the states, the size of the PTA was reduced by over 63%. The table also reports the output signal's coefficients of determination (R^2), averaged over all states. Due to the high nonlinearity of the approximated signal (see figure 9), multiple linear regression was unable to achieve high R^2 values. However, the neural network was able to make much better approximations.

C. Experimental Results

Using ANODA algorithm (given in figure 6), the anomaly detection was performed targeting the following faults:

1) Zero value of the signal (f_1)
2) Signal drop by more than 10% (negative offset onto the signal, f_2)
3) Signal jump by more than 10% (positive offset onto the signal, f_3)
4) Unknown control event occurred (f_4)
5) Control event occurred too early (f_5)
6) Control event occurred too late (f_6)

The faults f_1, f_2 and f_3 are detected when the process (continuous) output variable is out of range. Faults f_4, f_5 and f_6 occur in control (discrete) signals.

In order to evaluate the results, we used the ratios of the numbers of correctly and incorrect predicted samples. These numbers are defined in a table called a confusion matrix. Table II shows the confusion matrix with counts of true positive, false positive, false negative and true negative samples. Based on these counts, the following performance metrics are calculated (all expressed as a percentage):

Actual values	Predicted values: Positive	Predicted values: Negative
Positive	True Positive (TP)	False Negative (FN)
Negative	False Positive (FP)	True Negative (TN)

TABLE II
CONFUSION MATRIX

- Sensitivity (true positive rate, $TP/(TP+FN)$): It is important to know the portion of truly identified anomalies. Sensitivity represents the probability that the algorithm will recognize the anomaly when the anomaly is truly present in the given sample.
- Specificity (true negative rate, $TN/(TN+FP)$): This measure gives the probability that the algorithm will recognize the sample as being normal, when the anomaly is truly absent.
- Accuracy ($(TP+TN)/(TP+TN+FP+FN)$): For model comparison reasons, it is usually convenient to have a single number that summarizes the capabilities of the anomaly detection algorithm. Accuracy represents the number of correct predictions in relation to the total number of predictions.

Table III shows the average sensitivity, specificity and accuracy calculated over 100 runs that were performed for each of the six faults.

Performance metrics	Faults f_1	f_2	f_3	f_4	f_5	f_6
Sensitivity(%)	100	97.12	98.29	100	100	100
Specificity(%)	100	92.98	93.95	100	100	100
Accuracy(%)	100	93.82	94.83	100	100	100

TABLE III
PERFORMANCE METRICS FOR THE SIX FAULTS

It can be seen that the faults in discrete signals (f_4, f_5 and f_6) are always detected. This is because of the deterministic way in which these faults are checked, i.e. it is tested can the observed transition really be triggered in the current automaton state, and does the observed transition timing fall into the learned time interval. However, using hard thresholds for the timing can be very sensitive, especially in the situations when the observed timing is very close to the interval, but not contained by it (e.g. observed timing $t = 23$, learned interval $\delta = [7, 22]$). For such cases, some sort of a warning could be triggered, instead of raising the alarm.

Anomaly detection in the continuous signal is more challenging, as it depends on the online prediction capabilities of the model (i.e. output value needs to be predicted at every second based on the vector of inputs). Nevertheless, judging by the obtained results (performance metrics for f_1, f_2 and f_3 in table III), and since no preprocessing was used, it can be concluded that automatically learned behavior models can be used for anomaly detection in hybrid systems, with the acceptable detection rates.

VI. Conclusion

This paper presented the usability of behavior models for anomaly detection in hybrid systems, such as the real-world process plants. Many model-based approaches to anomaly detection and diagnosis in general exist, but one common thread is always present: the "modeling bottleneck". Due to the obstacles of manual modeling, behavior models in this paper are derived automatically from recorded system observations, using an offline HyBUTLA algorithm. This algorithm generates the models by learning hybrid automata. To the best of our knowledge, this is the first such algorithm.

Learned behavior model is used to detect the anomalies in a component of a real small production hybrid system. This model is provided as an input to the anomaly detection algorithm ANODA, together with the observations taken from the running system periodically, at every second (i.e. this algorithm works in an online manner). For this particular example, ANODA algorithm required in the worst case 21.3 ms to classify obtained observation (7.7 ms in the average case). All anomalies in discrete control signals are successfully detected. Discrepancies of the continuous output variable from its tolerance are detected with the accuracies ranging from 93% (acceptable) to 100% (excellent). Both portions of correctly identified actual positive (sensitivity), and truly identified actual negative samples (specificity) are always over 92%. These results confirm a significant potential of automatically learned behavior models in the anomaly detection applications.

In the future work, the model learning algorithm will be extended to enable generation of the new states based on threshold crossings of the continuous variables (autonomous jumps). By introducing such modeling simplification, the modeling of complex nonlinearities in the system continuous behavior should be eased ([34]). Further, the complexity of learning behavior models in the hybrid automaton framework will be analyzed.

References

[1] M. S. Branicky, "Introduction to hybrid systems," in *Handbook of Networked and Embedded Control Systems*, 2005, pp. 91–116.

[2] J. de Kleer and B. C. Williams, "Diagnosing Multiple Faults," *Artificial Intelligence*, vol. 32, no. 1, pp. 97–130, 1987.

[3] R. Reiter, "A Theory of Diagnosis from First Principles," *Artificial Intelligence*, vol. 32, no. 1, pp. 57–95, 1987.

[4] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: data mining, inference and prediction*, 2nd ed. Springer, 2008.

[5] A. Vodenčarević, H. Kleine Büning, O. Niggemann, and A. Maier, "Identifying behavior models for process plants," in *16th IEEE International Conference on Emerging Technologies and Factory Automation ETFA'2011*, Toulouse, France, September 2011.

[6] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P. h. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine, "The algorithmic analysis of hybrid systems," *Theoretical Computer Science*, vol. 138, pp. 3–34, 1995.

[7] R. Mohammadi, "Fault diagnosis of hybrid systems with applications to gas turbine engines," Ph.D. dissertation, Concordia University, Montreal, Canada, 2009.

[8] S. Narasimhan and G. Biswas, "Model-based diagnosis of hybrid systems," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 37, no. 3, pp. 348 –361, May 2007.

[9] C. Baydar and K. Saitou, "Prediction and diagnosis of propagated errors in assembly systems using virtual factories," *Application Brief, ASME Journal of Computers and Information Science in Engineering*, vol. 1, pp. 261–265, 2001.

[10] R. Isermann, *Fault-Diagnosis Systems: An Introduction from Fault Detection to Fault Tolerance*, 1st ed. Springer, 2006.

[11] M. Sanseverino and F. Cascio, "Model-based diagnosis for automotive repair," *IEEE Expert*, vol. 12, no. 6, pp. 33 –37, nov/dec 1997.

[12] B. Peischl and F. Wotawa, "Model-based diagnosis or reasoning from first principles," *IEEE Intelligent Systems*, vol. 18, no. 3, pp. 32–37, 2003.

[13] S. Hashtrudi Zad, R. Kwong, and W. Wonham, "Fault diagnosis in discrete-event systems: framework and model reduction," *Automatic Control, IEEE Transactions on*, vol. 48, no. 7, pp. 1199 – 1212, 2003.

[14] S. Tripakis, "Fault diagnosis for timed automata," in *FTRTFT*, 2002, pp. 205–224.

[15] J. Lunze, J. Schröder, and P. Supavatanakul, "Diagnosis of discrete event systems: the method and an example," in *Proceedings of the Workshop on Principles of Diagnosis, DX'01*, Via Lattea, Italy, 2001, pp. 111–118.

[16] P. Supavatanakul, J. Lunze, V. Puig, and J. Quevedo, "Diagnosis of timed automata: Theory and application to the damadics actuator benchmark problem," *Control Engineering Practice*, vol. 14, no. 6, pp. 609–619, June 2006.

[17] S. Narasimhan, "Model-based diagnosis of hybrid systems," Ph.D. dissertation, Vanderbilt University, Faculty of the Graduate School, Nashville, TN, USA, 2002.

[18] F. Zhao, X. D. Koutsoukos, H. W. Haussecker, J. Reich, and P. Cheung, "Monitoring and fault diagnosis of hybrid systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 35, no. 6, pp. 1225–1240, 2005.

[19] M. W. Hofbaur and B. C. Williams, "Mode estimation of probabilistic hybrid systems," in *In Intl. Conf. on Hybrid Systems: Computation and Control*. Springer Verlag, 2002, pp. 253–266.

[20] B. C. Williams, M. M. Henry, and M. M. Henry, "Model-based estimation of probabilistic hybrid automata," Tech. Rep., 2002.

[21] M. Wang and R. Dearden, "Detecting and Learning Unknown Fault States in Hybrid Diagnosis," in *Proceedings of the 20th International Workshop on Principles of Diagnosis, DX09*, Stockholm, Sweden, 2009, pp. 19–26.

[22] P. Struss and B. Ertl, "Diagnosis of bottling plants - first success and challenges," in *20th International Workshop on Principles of Diagnosis, Stockholm*, Stockholm, Sweden, 2009.

[23] R. Alur and D. Dill, "A theory of timed automata," *Theoretical Computer Science*, vol. vol. 126, pp. 183–235, 1994.

[24] F. E. Cellier, *Continuous System Modeling*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1991.

[25] B. Kumar, O. Niggemann, and J. Jasperneite, "Statistical models of network traffic," in *International Conference on Computer, Electrical and Systems Science*. Cape Town, South Africa, Jan 2010.

[26] D. Angluin, "Learning regular sets from queries and counterexamples." *Inf. Comp.*, pp. 75(2):87–106, 1987.

[27] E. M. Gold, "Complexity of automaton identification from given data," *Information and Control*, vol. 37, no. 3, pp. 302–320, 1978.

[28] R. C. Carrasco and J. Oncina, "Learning deterministic regular grammars from stochastic samples in polynomial time," in *RAIRO (Theoretical Informatics and Applications)*, 1999, p. 33(1):120.

[29] F. Thollard, P. Dupont, and C. de la Higuera, "Probabilistic DFA inference using Kullback-Leibler divergence and minimality," in *Proc. 17th International Conf. on Machine Learning*. Morgan Kaufmann, 2000, pp. 975–982.

[30] S. Verwer, "Efficient identification of timed automata: Theory and practice," Ph.D. dissertation, Delft University of Technology, 2010.

[31] R. Grosu, S. Mitra, P. Ye, E. Entcheva, I. V. Ramakrishnan, and S. A. Smolka, "Learning cycle-linear hybrid automata for excitable cells," in *Proc. of HSCC07, the 10th International Conference on Hybrid Systems: Computation and Control, volume 4416 of LNCS*. Springer Verlag, 2007, pp. 245–258.

[32] AutomationML, www.automationml.org, 2010.

[33] D. M. Hawkins, *Identification of outliers*. London; New York: Chapman and Hall, 1980.

[34] P. J. Mosterman and G. Biswas, "Towards procedures for systematically deriving hybrid models of complex systems," in *Hybrid Systems*, 2000, pp. 324–337.