# Learning Behavior Models of Hybrid Systems Using Wavelets for Autonomous Jumps Detection

Asmir Vodenčarević
Knowledge-Based Systems Research Group, Department of Computer Science
International Graduate School Dynamic Intelligent Systems, University of Paderborn
33098 Paderborn, Germany
asmir.vodencarevic@upb.de

***Abstract*—Today's hybrid technical systems, such as production facilities, evolve rapidly. Therefore the efforts and resources needed to obtain their trustworthy behavior models are significantly increased. This forms the modeling bottleneck, as the model creation mostly relies on the expensive and durable manual modeling. Behavior models are of the greatest importance in monitoring, anomaly detection, and diagnosis applications.**

**To deal with this issue, the novel HyBUTLA algorithm was recently proposed for automated learning of behavior models from process data. Despite being the first hybrid automaton learning algorithm, it does not model the autonomous jumps (abrupt changes in process variables), and it suffers from long runtime due to the use of advanced machine learning methods.**

**In this paper the HyBUTLA algorithm is improved to account for the autonomous jumps by introducing the splitting step based on discrete wavelet transform. This significantly improves the algorithm runtime, since high function approximation accuracy can be reached by using rather simple methods such as multiple linear regression. The benefits that splitting brings are formally proved and demonstrated in a real-world production system.**

## I. Introduction

The success and variety of the model-based approaches to monitoring, anomaly detection, and diagnosis of today's complex technical systems have emphasized the need for having reliable behavior models. Modeling complex technical systems is in general a very hard task. These systems comprise always-increasing number of software-based components, embedded systems, and distributed architectures. They are characterized by a permanent interaction between the discrete supervisory controller and the continuous physical system, as shown in Fig. 1. Their interfaces are arrays of sensors, actuators, and industrial networks. The overall system behavior is hybrid in nature and therefore are such systems called hybrid systems [1]. The presence of dual discrete-continuous dynamics induces two types of changes in the system [2]:

1) Controlled jumps: Discrete supervisory controller emits discrete control signals that, by changing the state of the actuators, cause changes in the physical system.
2) Autonomous jumps: Continuous process (physical) variables change the state of the system by crossing thresholds or by abruptly changing dynamics due to some external influences (e.g. disturbances or operators).

While controlled jumps are naturally present in such systems, autonomous jumps are often artificially introduced in order to ease the modeling of complex non-linear dynamics [3].

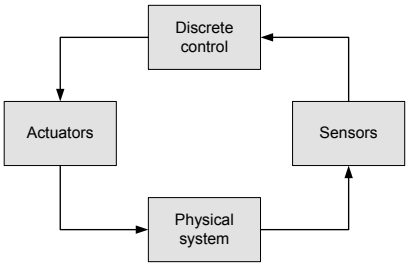



Fig. 1. Interacting discrete and continuous dynamics

On top of all complexity-related issues of modeling, behavior models are today mostly created manually in a costly and durable manner that relies highly on the expert knowledge. Manually created models also require manual updates. Such modeling bottleneck can be overcome by using the data-driven approaches from the learning theory [4] for learning the models automatically. Significant efforts have been made to learn behavior models in the modeling formalism of hybrid automaton [5]. The first hybrid automaton learning algorithm HyBUTLA (Hybrid Bottom-Up Timing Learning Algorithm, [6]) suffers however from long runtime due to the use of advanced machine learning methods for approximating continuous dynamics. In addition, it does not model the autonomous jumps in the system. The current HyBUTLA merging strategy depends on the type of used approximation functions.

In this paper, the HyBUTLA algorithm is extended with state splits, to account for the autonomous jumps in continuous dynamics. These jumps are identified by detecting abrupt changes in the signal using discrete wavelet transform [7]. In the formalism of hybrid automaton, it means that the transitions between the states could be triggered not only by the changes in control signals, but also by the changes in process variables. State split simplifies the learning of continuous dynamics by reducing amounts of data that need to be approximated within the states. This significantly improves the algorithm runtime and enables simpler methods, such as the multiple linear regression, to reach higher approximation accuracies. The benefits of this accessory are formally proved and evaluated in learning behavior models for a component of a small real process plant. Practical applicability of one such model was demonstrated in the anomaly detection application.

978-1-4673-0311-8/12/$31.00 ©2012 IEEE

## II. State of the art

There are many approaches to manual modeling of complex hybrid systems that use various formalisms such as Bond graphs, Kalman and particle filters, Bayesian and Petri networks, and finite automata. An overview can be found in [8]. However, there have been a few attempts toward learning such models automatically. Noticeable work is done in learning finite automata models from data.

The well-known formalism capable of representing discrete-event systems is called deterministic finite automaton (DFA). Several algorithms exist for learning DFA models in both online (additional data for learning can be used during runtime) and offline (only prerecorded data can be used) manner. The most famous ones are Angluin's L* [9], Gold's algorithm [10], ALERGIA [11], and MDI [12]. The last two ones use the state merging approach for learning probabilistic DFA. Here the learning examples are represented as the paths from the initial state to one of the final states, where the common prefixes are combined in a tree-like structure called the prefix tree acceptor. Then the states found to be similar with respect to some criteria are merged. The expressiveness of DFA was extended by adding the time information to the events that trigger transitions between the states. In this way a timed automaton (TA) is obtained [13]. In [14] Verwer presented the first TA learning algorithms, namely ID_1-DTA and RTI+, which use a top-down state merging order. The BUTLA algorithm for learning TAs with a bottom-up merging order is given in [15]. By adding differential (or other) functions that approximate continuous dynamics inside the states, a formalism of hybrid automaton (HA) is obtained [5]. First attempts to learn HA models were given in [16], but this approach is incapable of including discrete signals. In [6] the HyBUTLA algorithm that can learn both discrete and continuous elements of the hybrid automaton is given. This algorithm needs to be able to take into account the autonomous jumps in continuous signals.

There has been a lot of work done in the previous two decades on abrupt change detection, which can be used for finding the autonomous jumps automatically. This work resulted in the linear model-based approaches (see e.g. [17]), the model-free approaches (support vector machines [18]), and the non-parametric approaches (Discrete Fourier Transform [19] and Discrete Wavelet Transform [20]). For more detailed classification see [21]. These approaches have found a wide range of applications, including: quality control, seismic data processing, vibration monitoring of mechanical systems, and image processing [17]. Discrete Wavelet Transform (DWT) is particularly used for signal segmentation, especially in fault detection and diagnosis (see e.g. [21], [22], [23]). Since HyBUTLA algorithm takes no a priori knowledge about the types of existing signals, model-based approaches cannot be applied. Due to the speed, effectiveness, and popularity of DWT in analyzing both sinusoidal and impulse, high-frequency as well as low-frequency signal components, it has been the natural choice for this model-learning task.

## III. Learning behavior models

The overall system model consists of a *parallelism model* of its structure, and behavior models for its components. The parallelism model is a set of interconnected components that show strictly sequential behavior. There is no algorithm to learn it automatically, but it is usually derived in the plant planning and design phases [6]. In addition, signal values are normally logged during runtime of the plant. Having this in mind, we tried to answer the following general question: *Given a system's structure and its measurements, how can behavior models for its components be automatically learned?*

### A. Modeling Formalism

The modeling formalism found to be most suitable for representing hybrid systems is the hybrid automaton [5]. It is capable of modeling both discrete and continuous dynamics, together with the timing and probabilistic information. In the following, its formal definition is given.

**Definition 1.** *(Hybrid Automaton) A 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 set of final states.*
- $\Sigma$ *is the alphabet comprising all relevant events.*
- $T \subseteq S \times \Sigma \times S$ *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* $Num : T \to \mathbb{N}$ *counts the number of observations that used the transition.*
- *A set of transition timing constraints* $\Delta$ *with the elements* $\delta : T \to I$*, where* $I$ *is the set of intervals. Time evolution is recorded by the set of clocks* $X$*.*
- *A set of functions* $\Theta$ *with elements* $\theta_s : \mathbb{R}^n \to \mathbb{R}^m, \forall s \in S, n, m \in \mathbb{N}$*. I.e.* $\theta_s$ *is the function computing output signal value changes within state* $s$*. For learning these functions, input signal values and time can be used.*

A hybrid automaton example is given in Fig. 2. According to Definition 1, it follows: $s_1, s_2, s_3 \in S$, $a, b \in \Sigma$, $\delta_1, \delta_2 \in \Delta$, and $\theta_{s_1}, \theta_{s_2}, \theta_{s_3} \in \Theta$. Assume the system is in state $s_1$ where its continuous dynamics is defined by function $\theta_{s_1}$. If an event $a$ is observed at the time instance that satisfies constraint $\delta_1$, a transition occurs to the state $s_2$. In the recorded plant history, such a transition occurred 8 times, giving $Num(s_1, a, s_2) = 8$. For the other transition $Num(s_1, b, s_3) = 2$, thus the corresponding transition probabilities are 0.8 and 0.2 respectively.

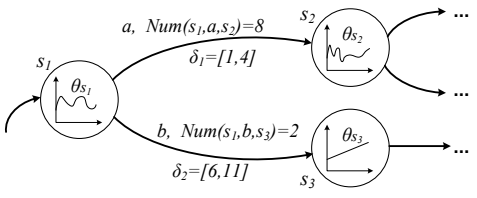



Fig. 2. A hybrid automaton example

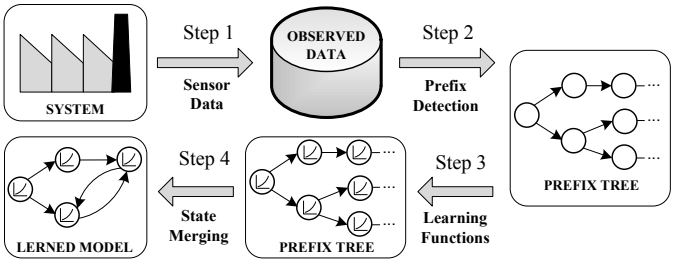



Fig. 3. An approach to learning hybrid automaton

Comparing to the original formalism given in [5], only a single clock ($|X| = 1$) is allowed and it uses relative timing. In addition, arbitrary functions are used for function approximation, instead of strictly differential equations.

### B. The HyBUTLA Learning Algorithm

The HyBUTLA algorithm learns the hybrid automaton behavior model using an approach illustrated in Fig. 3:

**Step (1)**: Process data are logged in the performed running cycles of a technical system. The events $\Sigma$ that trigger transitions between the states (see Definition 1) are defined as the changes in discrete signals only.

**Step (2)**: By combining common events of different cycles, a tree-like structure called the *prefix tree acceptor* is created. Every cycle is represented as a path from the initial state to one of the final states, where several cycles can share parts of the paths.

**Step (3)**: $\theta_s$ functions that approximate the continuous output signal are learned for every state $s \in S$.

**Step (4)**: Similar states are merged to make the automaton more general. To that aim, the compatibility between the states is checked using the bottom-up order. For a pair of states, similarity is checked in: the probabilities of the in-coming transitions, the probability density functions of the in-coming transitions' timings, and the function approximations. When all criteria are satisfied, two states are merged, and the new function is learned for approximating the output signal of the resulting state.

In its current state, the HyBUTLA algorithm suffers from the following drawbacks:

1) Neglected autonomous jumps: Although continuous signals in the real systems often have sudden changes, these are not taken into account by the algorithm. New states are derived solely on the basis of controlled jumps.
2) Insufficient runtime performance: In order to adequately approximate non-linear functions that can change abruptly, the advanced and time-consuming regression methods are often needed.
3) State merging dependence on approximation functions: The third criteria for state compatibility must be defined differently based on the type of used $\Theta$ approximation functions. E.g. if multiple linear regression is used, it is checked if the corresponding regression coefficients of functions from the two states are similar enough.

## IV. Modeling Autonomous Jumps

To overcome stated issues, an extension to the HyBUTLA algorithm is given here: the inclusion of the autonomous jumps. When these jumps are found, the states are split at their locations, making it faster and easier to approximate the continuous dynamics. Higher accuracy rates can be achieved with simpler regression methods. In addition, the function similarity is excluded from the state merging criteria. The illustration of the split idea is depicted in Fig. 4. It shows one controlled jump that triggers the transition, and one autonomous jump that splits the state. For automatic detection of autonomous jumps, discrete wavelet transform is used.

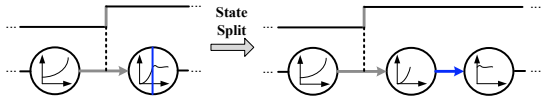



Fig. 4. The illustration of the split idea

### A. Discrete Wavelet Transform

In the classical Fourier analysis, the signal is transformed from the time domain into the frequency domain. Fourier transform breaks the signal into the sum of sinusoids with different frequencies. It shows the amplitude-frequency relationship, but the time information is lost. Therefore, it cannot be seen at what point some specific event happened. In contrast, the Wavelet transform gives this information, by representing the signal as the sum of scaled and shifted versions of the so-called *mother wavelet*. Mother wavelet and all of its scaled and shifted versions represent the waveforms of limited length and zero average value. The continuous wavelet transform (CWT) is defined as the sum over all time of the signal $f(t)$ multiplied by scaled and shifted versions of the mother wavelet function $\psi(t)$. It is calculated as follows [21]:

$$CWT(\tau, s) = \frac{1}{\sqrt{|s|}} \int_{-\infty}^{\infty} f(t)\psi^* \left( \frac{t-\tau}{s} \right) dt, \qquad (1)$$

where $\psi^*(t)$ is a complex conjugate of the mother wavelet, and $\tau, s \in \mathbb{R}, s \neq 0$ are the shifting and the scaling parameters, respectively. $\tau$ relates to the time, as it translates the wavelet over the signal. Scale parameter $s$ relates to the frequency, as it expands or compresses the wavelet. Low scale compresses the wavelet (increasing wavelet's frequency), making it suitable for detecting the sudden changes in the signal (i.e. the autonomous jumps). High $s$ values stretch the wavelet (decreasing wavelet's frequency), enabling the insight into the slow changes of the signal. Transform coefficients $CWT$ indicate the similarity between the wavelet and the analyzed signal. Higher correlation is registered at larger coefficients.

The calculation of CWT over all time at all scales and positions is very computationally expensive. When it is calculated using limited number of scales and positions that are based on the powers of two, the much faster and just as accurate discrete wavelet transform (DWT) is obtained [7], [20], [21].

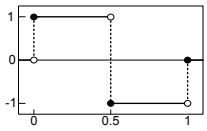



Fig. 5. The Haar wavelet $\psi$ function

There are a number of wavelets developed over years for different applications. The oldest and simplest is the *Haar wavelet* (also known as Daubechies 1 and db1, [7]). It is discontinuous (see Fig. 5), and thus is often used for detecting abrupt changes in the signal.

### B. The State Split

When two states are merged, their belonging datasets for learning a new $\theta$ function are combined. In [6], the simplest method used for learning was the multiple linear regression with linear terms (MLR-LT). The goodness of fit was expressed as the coefficient of determination $R^2(\theta)$, which shows how good the learned $\theta$ function approximates the observed data $y_j$. It is defined as follows:

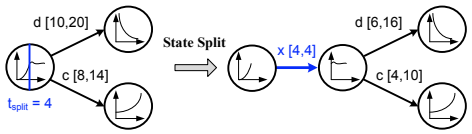
$$R^2(\theta) = 1 - \frac{SS_{error}}{SS_{total}} = 1 - \frac{\sum\limits_j (y_j - \theta_j)^2}{\sum\limits_j (y_j - \overline{y})^2}, \qquad (2)$$

where $SS_{error}$ is the sum of squares of errors, $SS_{total}$ is the total sum of squares, $\theta_j$ are values of learned $\theta$ function at $j$ points, and $\overline{y}$ is the mean value of observed data $y_j$.

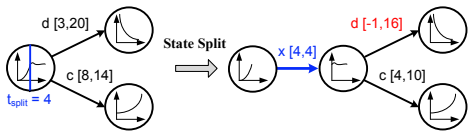
Ones the state merging is completed, the state split can be performed. First, only those states whose signal approximation has the $R^2$ value lower than some threshold (e.g. 90%), are selected as the candidates for splitting. Discrete wavelet transform of these signals is performed using the Haar wavelet. The location (time) $t_{split}$ of the highest DWT coefficient indicates the highest abrupt change of the signal (autonomous jump), and thus is a potential splitting position. In order to perform the split, following conditions have to hold:

1) Positive resulting transition timings: State split changes the time intervals $\delta$ of that state's out-going transitions (refer to Definition 1). When the state is split at the moment $t_{split}$, this time must be subtracted from all $\delta$ intervals of the out-going transitions. If all resulting intervals are strictly positive, the split is allowed (see Fig. 6(a)). In case any resulting interval includes zero or negative values, the split will not be performed (see Fig. 6(b), problematic interval is colored red).
2) Increased overall $R^2$ value: State split makes sense only if it ensures the global increase in the function approximation accuracy. Theorem 1 in Section IV-C gives and proves this condition.

When these conditions are fulfilled, the state split is performed. Otherwise, the next highest DWT coefficient is taken, and the tests are repeated for its $t_{split}$. This process goes until the split is finally performed, or the state is declared to be unsplittable. Once all states have been recursively evaluated (including those already obtained by splitting) the procedure is stopped and the new automaton is obtained.




(a) Positive transition timings




(b) Negative transition timing

Fig. 6. Effects of state split on transition timings

### C. The Global Accuracy Increase

State split that divides the state $s_k \in S$ into $s'_k$ and $s''_k$, has to ensure both the local and global increase of the model's function approximation accuracy. The local increase is obtained when the $R^2$ of both resulting states is higher than the $R^2$ of the state being split, i.e. it has to hold:

$$R^2(\theta_{s'_k}) > R^2(\theta_{s_k}) \; and \; R^2(\theta_{s''_k}) > R^2(\theta_{s_k}). \qquad (3)$$

The global increase of the function approximation accuracy is evaluated by calculating and comparing the total $R^2$ averaged over the automaton states, before and after the split. Unfortunately, the condition for ensuring the local increase given by (3) does not ensure the global increase as well. For illustration purposes, the following example is given.

**Example 1.** *Let a hybrid automaton $A$ be given according to Definition 1, with $n = 6$ states: $S = \{s_0, s_1, s_2, s_3, s_4, s_5\}$. Let the function approximation accuracies of the $\Theta$ functions of these states be: $R^2(\theta_{s_0}) = 0.1, R^2(\theta_{s_1}) = 0.2, R^2(\theta_{s_2}) = 0.3, R^2(\theta_{s_3}) = 0.4, R^2(\theta_{s_4}) = 0.5, R^2(\theta_{s_5}) = 0.6$. Its average coefficient of determination is calculated according to:*

$$R^2_A = \frac{\sum\limits_{i=0}^{n-1} R^2(\theta_{s_i})}{n} \qquad (4)$$

*and for this example equals $R^2_A = 0.35$. Now let us assume that the state $s_2$ has been split, resulting in the states $s'_2$ and $s''_2$ with the respective function approximation accuracies $R^2(\theta_{s'_2}) = 0.31$ and $R^2(\theta_{s''_2}) = 0.32$. The obtained automaton $A_{split}$ now has seven states: $S_{split} = \{s_0, s_1, s'_2, s''_2, s_3, s_4, s_5\}$, and its average coefficient of determination is $R^2_{A_{split}} = 0.347$. Obviously the condition given by (3) ensures the local, but not the global accuracy increase.*

The global increase of the model's function approximation accuracy is guaranteed under the following condition:

$$R^2(\theta_{s'_k}) + R^2(\theta_{s''_k}) > R^2(\theta_{s_k}) + R^2_A. \quad (5)$$

**Theorem 1.** *Let $A = (S, s_0, F, \Sigma, T, \Delta, Num, \Theta)$ be the hybrid automaton with $|S| = n$ states. Its total coefficient of determination averaged over the states is given by (4). The split of a state $s_k \in S$ into the states $s'_k$ and $s''_k$ brings the global increase of the function approximation accuracy iff for the two resulting states $s'_k$ and $s''_k$ the condition (5) holds.*

*Proof:* By splitting the state $s_k \in S$, a new automaton $A_{split}$ is obtained with the number of states $|S_{split}| = n + 1$. The global increase of the function approximation accuracy is ensured if the following inequality holds:

$$R^2_{A_{split}} > R^2_A. \quad (6)$$

We will prove the *if-part* of the theorem, i.e. we prove (6) assuming (5). The condition given by (5) can be rewritten as:

$$R^2(\theta_{s'_k}) + R^2(\theta_{s''_k}) - R^2(\theta_{s_k}) - R^2_A > 0.$$

Now, we add and subtract the $R^2$ values of the other states (unaffected by split):

$$\sum_{i=0}^{k-1} R^2(\theta_{s_i}) + R^2(\theta_{s'_k}) + R^2(\theta_{s''_k}) + \sum_{i=k+1}^{n-1} R^2(\theta_{s_i}) - \sum_{i=0}^{k-1} R^2(\theta_{s_i}) - R^2(\theta_{s_k}) - \sum_{i=k+1}^{n-1} R^2(\theta_{s_i}) - R^2_A > 0.$$

The first line represents the total coefficient of determination averaged over the states for the automaton $A_{split}$ according to (4), and multiplied by $(n + 1)$. It follows:

$$(n+1)R^2_{A_{split}} - \left[\sum_{i=0}^{n-1} R^2(\theta_{s_i}) + R^2_A\right] = (n+1)R^2_{A_{split}} - (n+1)R^2_A > 0.$$

After dividing the inequality by $(n + 1)$, we finally get:

$$R^2_{A_{split}} - R^2_A > 0. \quad (7)$$

The expressions given by (6) and (7) are equal, thus the *if-part* of the theorem is proven. Using the same methodology, it is now easy to prove the *only-if-part* of the theorem: prove (5) assuming (6). This completes the proof. ■

To illustrate the global increase of the model's function approximation accuracy under the validity of condition given by (5), we give the following example.

**Example 2.** *Assuming the same setting as in example 1, we see that the condition (5) did not hold:*

$$R^2(\theta_{s'_2}) + R^2(\theta_{s''_2}) = 0.63 \not> R^2(\theta_{s_2}) + R^2_A = 0.65.$$

*In case the split of the state $s_2$ would result in the states $s'_2$ and $s''_2$ with the respective function approximation accuracies $R^2(\theta_{s'_2}) = 0.36$ and $R^2(\theta_{s''_2}) = 0.32$, the condition (5) would be satisfied, and the total average coefficient of determination of the resulting automaton would be $R^2_{A_{split}} = 0.354 > R^2_A$.*

## V. Case study

The state split is evaluated at the same small process plant as in [6], using the same dataset. This exemplary system known as the Lemgo Model Factory produces bulk goods. The learning algorithm uses logs from 12 production cycles (with average length 191 samples) comprising 6 continuous and 6 discrete signals. The modeled component is the popping machine. Like in [6], the main goal is to obtain the smallest automaton with the highest function approximation accuracy.

The properties of prefix tree acceptor ($PTA$) and the two learned models are given in Table I. The first learned model, marked as $A_{min}$, is the smallest automaton obtained by merging $PTA$ states. This model was exposed to the state split to produce the second model $A_{split}$. For three models ($PTA$, $A_{min}$, and $A_{split}$), Table I gives: a number of states, a size reduction (in relation to $PTA$ size), the average $R^2$ obtained using MLR-LT, and the model learning times. In contrast to the problem setting given in [6] (no expert knowledge about I/O relations), output signal is known here in the learning phase. It is the active power of the popping machine's heater measured in a range of $0 - 3.25\ kW$. The average $R^2$ is here reported only for this signal (in contrast to [6] where it was averaged over all approximated signals).

TABLE I
Properties of learned behavior models and learning times

| Automaton ID | $PTA$ | $A_{min}$ | $A_{split}$ |
|---|---|---|---|
| Size (states) | 52 | 19 | 35 |
| Reduction (%) | 0 | 63.5 | 32.7 |
| MLR-LT $R^2$ (%) | 78 | 73 | 91 |
| MLR-LT time (s) | 2.63 | 6.14 | 24.1 |

$A_{split}$ model has significantly higher $R^2$ comparing to $A_{min}$ (18% in the absolute value). This is however paid with the increase in model size. Furthermore, the state split based on discrete wavelet transform reduces the function learning complexity, thus enables learning of models with relatively high accuracy in very short time. Comparing to the time needed to train neural networks reported in [6], this approach is around 17 times faster. It is interesting to note that $A_{split}$ has higher $R^2$ than $PTA$.

Table I presents only the most accurate model obtained with state split (due to space restrictions). However, during splitting a series of models with different sizes and accuracies was created. These are shown in Fig. 7. The models from Table I are denoted at the bottom. As the number of states obtained by splitting increases, the model reduction (in relation to $PTA$ size) decreases. However, the accuracy ($R^2$) increases. This trade-off enables to select the model with appropriate size-accuracy ratio, depending on the application area.

Practical potential of $A_{split}$ model is evaluated at the same exemplary system in a task of anomaly detection. As in [6], the following types of anomalies are targeted: signal zero value ($f_1$), signal drop by 10% ($f_2$), signal jump by 10% ($f_3$), unknown control event ($f_4$), and wrong control event timing ($f_5$). For each anomaly type, 100 runs were done, and their

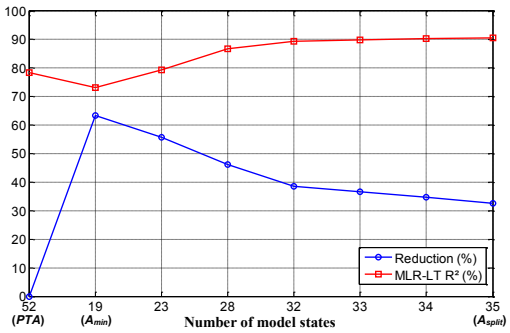



Fig. 7. Trade-off between model size and accuracy

average error rates are given in Table II. There were 20% of anomalous samples in the test cycle. All anomalies in discrete parts ($f_4$,$f_5$) were successfully detected, while anomalies in continuous parts had low error rates. These rates are slightly better than those given in [6], where neural network models were used for prediction.

TABLE II
ERROR RATES FOR THE FIVE ANOMALY TYPES

| Anomaly Type | $f_1$ | $f_2$ | $f_3$ | $f_4$ | $f_5$ |
|---|---|---|---|---|---|
| Error Rate (%) | 0 | 4.5 | 4.3 | 0 | 0 |

## VI. CONCLUSION

This paper presented the next step in overcoming the modeling bottleneck by learning behavior models of hybrid systems from recorded process data. The first hybrid automaton learning algorithm HyBUTLA [6] is here extended to model the autonomous jumps in continuous signals. The proposed state split uses discrete wavelet transform to detect the abrupt changes in the approximated output signal. Based on detected change locations (timings), the corresponding states with low function approximation accuracy are split and the functions for the new states are learned. The state split enabled the simpler and faster regression methods to reach high accuracies. Moreover, the state merging criteria does not depend on the similarity of the approximation functions any more.

The gain of the state split is formally proved and a case study has been conducted in a real small process plant. Obtained results show that the state split increases the model accuracy, but also that the trade-off exists between accuracy and the model size. Comparing to the previous approach [6] where neural networks were used, the model learning process is now much faster. Practical usability of new models was demonstrated in the anomaly detection. Obtained low error rates confirm their significant potential in this application area. The future work will bring more experimental results on algorithm scalability, as well as complexity analysis of learning hybrid automata from data.

## REFERENCES


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

[2] S. Narasimhan and G. Biswas, "An approach to model-based diagnosis of hybrid systems," in *Proceedings of the 5th International Workshop on Hybrid Systems: Computation and Control*, ser. HSCC '02, 2002, pp. 308–322.

[3] P. J. Mosterman and G. Biswas, "Towards procedures for systematically deriving hybrid models of complex systems," in *Proceedings of the 3rd International Workshop on Hybrid Systems: Computation and Control*, ser. HSCC '00, 2000, pp. 324–337.

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

[5] 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.

[6] A. Vodenčarević, H. Kleine Büning, O. Niggemann, and A. Maier, "Identifying behavior models for process plants," in *Proc. of the 16th IEEE International Conf. on Emerging Technologies and Factory Automation ETFA'2011*, Toulouse, France, September 2011, pp. 937–944.

[7] I. Daubechies, *Ten lectures on wavelets*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 1992.

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

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

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

[11] R. C. Carrasco and J. Oncina, "Learning deterministic regular grammars from stochastic samples in polynomial time," in *RAIRO (Theoretical Informatics and Applications)*, 1999, pp. 33(1):1–20.

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

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

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

[15] A. Maier, O. Niggemann, A. Vodenčarević, R. Just, and M. Jäger, "Anomaly detection in production plants using timed automata," in *Proc. of the 8th International Conf. on Informatics in Control, Automation and Robotics (ICINCO)*. Noordwijkerhout, The Netherlands, July 2011, pp. 363–369.

[16] 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.

[17] M. Basseville and I. V. Nikiforov, *Detection of abrupt changes: theory and application*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1993.

[18] F. Desobry and M. Davy, "Support vector-based online detection of abrupt changes," in *Proc. of the IEEE ICASSP, Hong Kong*, 2003, pp. 872–875.

[19] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, 3rd ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2009.

[20] V. Alarcon-Aquino and J. A. Barria, "Change detection in time series using the maximal overlap discrete wavelet transform," in *Proc. of the Latin American applied research*, vol. 39, 2009, pp. 145–152.

[21] A. Ukil and R. Živanović, "Detection of abrupt changes in power system fault analysis: A comparative study," in *Proc. of the Southern African University Power Engineering Conference*, Johannesburg, South Africa, 2005.

[22] H. Q. Zhang and Y. Yan, "A wavelet-based approach to abrupt fault detection and diagnosis of sensors," *Instrumentation and Measurement, IEEE Transactions on*, vol. 50, no. 5, pp. 1389 –1396, oct 2001.

[23] H. Guo, J. Crossman, Y. Murphey, and M. Coleman, "Automotive signal diagnostics using wavelets and machine learning," *Vehicular Technology, IEEE Transactions on*, vol. 49, no. 5, pp. 1650 –1662, sep 2000.