ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/268800304

Online Passive Learning of Timed Automata for Cyber-Physical Production Systems

Conference Paper · July 2014

DOI: 10.1109/INDIN.2014.6945484

CITATIONS
31

READS
403

1 author:

Alexander Maier
Fraunhofer Institute for Optronics, System Technology and Image Exploitation IOSB
34 PUBLICATIONS **293** CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Project Informed Learning - Simulation-based Learning View project

Project Informed Learning - Hybrid Learning View project

All content following this page was uploaded by Alexander Maier on 13 March 2015.

The user has requested enhancement of the downloaded file.

Online Passive Learning of Timed Automata for Cyber-Physical Production Systems

Alexander Maier
Institute of Industrial Information Technologies
Lemgo, Germany
Email: alexander.maier@hs-owl.de

***Abstract*—Model-based approaches are very often used for diagnosis in production systems. And since the manual creation of behavior models is a tough task, many learning algorithms have been constructed for the automatic model identification. Most of them are tested and evaluated on artificial datasets on personal computers only. However, the implementation on cyber-physical production systems puts additional requirements on learning algorithms, for instance the real-time aspect or the usage of memory space. This paper analyzes the requirements on learning algorithms for cyber-physical production systems and presents an appropriate online learning algorithm, the Online Timed Automaton Learning Algorithm, OTALA. It is the first online passive learning algorithm for timed automata which in addition copes without negative learning examples. An analysis of the algorithm and comparison with offline learning algorithms completes this contribution.**

I. Motivation

The usage of model-based diagnosis approaches requires the existence of a behavior model. Often, these models are available as Statecharts or Finite Automata, which are created manually. But the automatic identification of Deterministic Finite Automata (DFA) is also an active research field. Several algorithms exist to learn DFAs from sampled data. Additionally algorithms for the identification of subclasses of the DFA as the Probabilistic Deterministic Finite Automaton (PDFA, e.g [1], [2]) or the Timed Automaton (TA, e.g. [3], [4]).

All of the existing algorithms work in an offline manner, i.e. the used data is stored in a database and is available for multiple usage during the learning process. However, the usage of learning algorithms in cyber-physical production systems puts additional requirements on the algorithms, for instance the real-time aspect. The data can be accessed only once. In one step this data sample has to be included into the model. Additionally, the algorithm has to work passively, i.e. the data have to be taken as they appear. This leads to the necessity of an online passive learning algorithm, which so far not exists.

To close this gap, this paper introduces an online learning algorithm for Timed Automata. To the best of our knowledge this is the first online passive learning algorithm for the identification of Timed Automata.

The rest of the paper is structured as follows: In section II a classification of learning algorithms is given and then some examples of learning algorithms for each type. After this, some requirements for learning algorithms in cyber-physical production systems are listed. Section III introduces the algorithm OTALA which is especially developed for the usage in cyber-physical production systems. In section IV the algorithm is evaluated and discussed from a practical point of view and an application scenario is given. Section V finally concludes the contribution and gives an outlook to future work.

II. Classification of Learning Algorithms for DFAs

Several formalisms exist to learn DFAs and its subclasses (e.g. Timed Automata). They can be roughly distinguished by following features:

- **Passive or active learning:** While passive learning algorithms have to cope with a given set of observations to learn the model, active learning algorithms can ask for additional observations, if needed. Active learning algorithms often imply a teacher-student relationship.
- **Offline or online learning:** Offline learning algorithms can apply preprocessing during the learning process since the data is stored and it is possible to access the data multiple times. Online learning is more real-time critical. Here, the algorithms are allowed to access the data only once. In one step this data sample has to be included into the model.
- **The usage of only positive or both positive and negative examples:** Some algorithms use both positive and negative data. In the context of grammar learning a positive example is a word accepted by the target language, on the contrary a negative example is not accepted. Within the scope of cyber-physical production systems it is not easy or even not possible to get or generate negative examples. Therefore, in this contribution we focus on algorithms, which learn based on positive examples only.
- **Allowing don't-care states or not:** So far, this criterion was not considered for the classification of learning algorithms. Algorithms which use incoming and outgoing events and event sequences for the state equivalence check usually have don't-care states since it may become possible to reach a state using different paths. To avoid don't-care states the algorithm should check the state information itself (e.g. the signal vector).

In the literature the terms *active* and *online* are often mixed up. The reason is that usually active learning algorithms work in an online manner and vice versa. Though, in this contribution, we introduce a learning algorithm which learns in an online and passive manner.

A. *Offline passive model learning*

Offline learning algorithms have to cope with a given set of observations. The general offline learning methodology, following the state merging method, is displayed in figure 1:

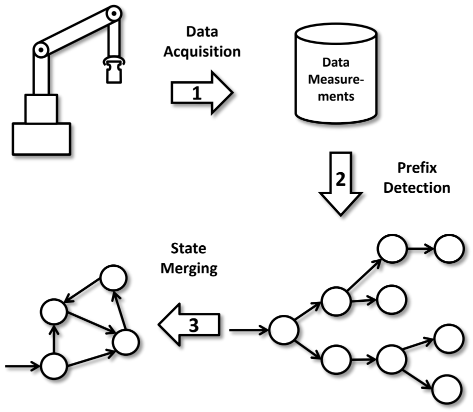

Fig. 1. The general offline learning methodology.

In **step (1)** the system is observed and the data are stored into a database.

In **step (2)** the prefix tree acceptor is created. Beginning with an initial state, with each discrete event a new state is created. The first observation cycle leads to a linked list. The following cycles again begin with the initial state. As far as the prefix is equal, the states and events are followed. When at some point an event is observed which is not an outgoing event from the current state, a new transition and a new state are created. The prefix tree acceptor stores the event paths in a dense form since every prefix is stored only once.

In **step (3)** each pairs of states are checked for compatibility. If a compatible pair of states is found, the states are merged. This is done to obtain a generalized automaton which abstracts the behavior of the observed system.

Several existing algorithms learn an automaton in an offline manner. They all proceed in the described steps but have different compatibility checks for state merging and different merging strategies.

Alergia [1] identifies a PDFA. For the compatibility check the Hoeffding bound [5] is used.

MDI (Minimal Divergence Inference) [2] also identifies a PDFA and uses a global merging criterion. It calculates the entropy of the current automaton and after a possible state merge. The state merge is rejected if the new automaton has a significantly different entropy and kept otherwise.

RTI+ (Real-Time Identification with only positive examples) [3] identifies a TA and is based on the red-blue framework [6]. Additionally to the state merging a splitting operation is introduced which splits a transition if the resulting subtrees are significantly different.

In contrast to the three aforementioned algorithms which work in a top-down manner, *BUTLA* (Bottom Up Timing Learning Algorithm) [4], which identifies a TA, uses the bottom-up strategy to search for compatible states in the prefix tree acceptor. This results in a speed increase. Similar to Alergia, the Hoeffding Bound is used to determine the compatibility of states.

B. *Online active model learning*

Online active learning is also often referred to as query learning. This implies a student-teacher relationship. The student (learner) can make a query to the teacher (oracle). This query can either be a question whether a certain sample is accepted by the target language (answer: yes or no) or whether the hypothesis is consistent with the target model (answer: yes or counterexample). Figure 2 illustrates the difference between active and passive learning according to [7].

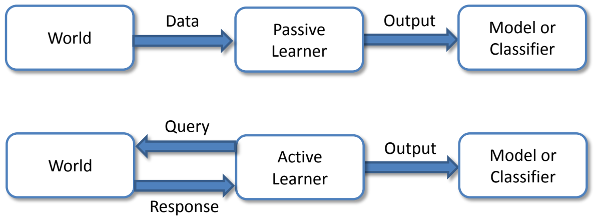

Fig. 2. Difference between active and passive learning [7].

Angluins $L*$ [8] is one of the first and most famous online active algorithms which learns a DFA. The learning operates as illustrated in figure 2.

Grinchtein introduced an online active learning algorithm for timed target languages identifying event-recording automata (ERA) [9]. The algorithm works similar to L*.

C. *Online passive model learning*

To the best of our knowledge so far no algorithm for the online passive learning of timed automata exists which in addition gets along with only positive examples.

D. *Requirements for model learning in cyber-physical production systems*

The implementation of learning algorithms on cyber-physical production systems as *a network of interacting elements with physical input and output instead of as standalone devices* [10], [11] puts several requirements on the algorithms which are mainly:

- **Economical usage of memory space:** The implementation of learning algorithms on cyber-physical systems requires an economical use of memory space. Due to the limited memory space the observations can not be stored (as it is required by offline learning algorithms) but rather every data sample has directly to be included into the model. This leads to the need of online learning algorithms.

- **Real-time capability:** The real-time capability is an important requirement for the learning in cyber-physical systems. Since the incoming data can not be stored, the implementation has to be able to handle the data stream in real-time, i.e. each data sample has to be included into the model before the next data sample arrives.
- **No expert knowledge:** Further, it is required that as little as possible expert knowledge is used. Offline learning algorithms still need some expert knowledge: at least the learning cycles have to be recorded. Here, the user of the learning algorithm usually has no knowledge about the amount of data which is needed to learn a correct automaton. The main advantage of our proposed algorithm is that it copes without expert knowledge. Additionally, the implementation can be operated by untrained personnel: The algorithm recognizes itself when the learning process is finished.
- **Active learning is not possible:** Learning in cyber-physical production systems cannot be performed using an active learning algorithm since there is no possibility to ask for samples. The data has to be taken as it comes. Additionally, there is no oracle (teacher) which can tell whether the hypothesis is correct and eventually return a counterexample.

III. Online Passive Learning of Timed Automata

In the previous section we have seen that offline learning algorithms for timed automata such as BUTLA [4] or RTI+ [3] have two main drawbacks:

1) Memory space is needed to store the observations.
2) At least for the recording and determining the number of observations some expert knowledge is needed.

To overcome these drawbacks in this section a passive online learning algorithm, the Online Timed Automaton Learning Algorithm (OTALA) is introduced. The main idea behind the learning algorithm is the assumption that each signal vector of the inputs/outputs represents a specific state in the automaton.

A. Timed Automaton Formalism

Alur introduced the formalism of Timed Automata [12]. Here, we modify the formalism for our purpose. According to Alur's definition the states hold no information about the value of certain signals (in cyber-physical production systems: actuators and sensors). Our formalism is modified to capture the signal vector as state information.

Definition 1: A **Timed Automaton** is a 4-tuple $A = (S, \Sigma, T, \delta)$, where

- S is a finite set of states. Each state $s \in S$ is a tuple $s = (id, \mathbf{u})$, where id is a current numbering and $\mathbf{u} = (io_1, io_2, ..., io_{|\Sigma|})^T$ is a vector, which indicates for each IO value, whether it is active ($io_n = 1$) or inactive ($io_n = 0$).
- Σ is the alphabet, the set of 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 transition timing constraint $\delta : T \rightarrow I$, where I is a set of intervals. δ always 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.

The modified Timed Automaton does not need an initial state as the original Timed Automaton does. Since the state is defined over the active/inactive IO values, the starting point is the state corresponding to the actual system's configuration. Further, due to the infinite operation of the system the set of final states is not needed either. Finally, the modified timed automaton does not need multiple clocks. Instead, only one clock is used which is reset with each firing of a transition, which results in relative time since the entering of states.

B. Learning Algorithm

The key question for automata learning algorithms is about the compatibility of states. Offline algorithms use the event sequences of the postfixes to determine the compatibility of states. However, learning the automaton in an online manner, we have to refer to other information. Therefore, the learning algorithm OTALA is mainly based on the following assumption:

Assumption 1: Each state in the observed cyber-physical production system can be described by the signal vector of the inputs/outputs and corresponds to one state in the final automaton.

This assumption is necessary, since during runtime the compatibility of states has to be determined and, especially at the beginning, no information about event sequences is available.

A similar idea is taken up by [13]. They also provide a learning algorithm, which however can not be used in an online manner. Following this assumption, the emergence of don't-care states is avoided.

The algorithm OTALA works as follows (see also figure 3 and algorithm in figure 4): For each observed event it is checked whether the configuration $\mathbf{u}(t)$ has been observed before and is stored in a state s (line 3). If such a state does not exist it is created with a corresponding transition (lines 13 - 17), otherwise it is checked whether a transition from the current to the new state exits. If a corresponding transition exists, the timing information (e.g. interval with the minimum and maximum time stamp) is updated (lines 5 - 6) otherwise a new transition is created (lines 7 - 9). After each step it is checked whether the learning converged to the final automaton (lines 18 - 20) and if it converged, the identified automaton is returned.

C. Convergence Criterion

Offline learning algorithms suffer from the point of detection of learning convergence. The data have to be acquired before learning the automaton. The needed amount of data is unknown beforehand. Thus, often too little data is acquired or,

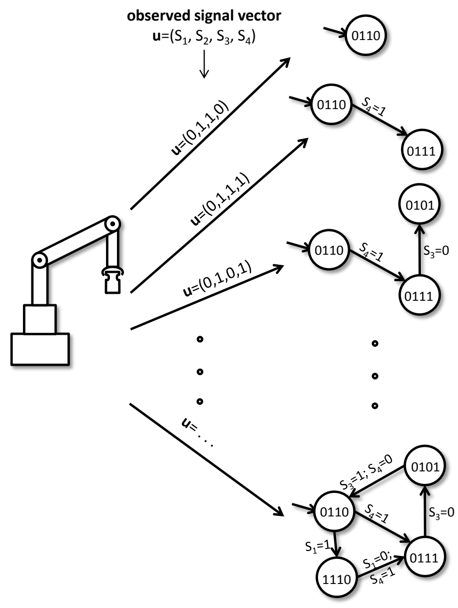

Fig. 3. The OTALA learning methodology.

```
Algorithm OTALA (Σ, U):
Given: Observations $\mathbf{U} = \{\mathbf{u}_0, \ldots, \mathbf{u}_{n-1}\}$ where $\mathbf{u}_i$ is one IO vector
Result: Timed Automaton $\mathcal{A}$

(1)  while newObservationExists()
(2)    for all $s \in S$
(3)      if $\mathbf{u}(t) == \mathbf{u}(s)$
(4)        stateExists = $true$
(5)        if transitionExits()
(6)          adaptTimingInformation(t)
(7)        else
(8)          createNewTransition($currentState, t(j), s$)
(9)        end if
(10)     end if
(11)     currentState = $s$
(12)   end for
(13)   if !stateExists
(14)     $s_{new}$ = createNewState()
(15)     createNewTransition($currentState, t(j), s_{new}$)
(16)     currentState = $s_{new}$
(17)   end if
(18)   if learningConverged()
(19)     return $\mathcal{A}$
(20)   end if
(21) end while
(22) return $\mathcal{A}$
```

Fig. 4. Online timed automata learning algorithm OTALA.

to be on the safe side, more data is acquired then actually is needed. A special feature of the OTALA algorithm is that it autonomously recognizes when the learning process is finished.

The learning progress can be measured based on the following characteristics:

1) **Number of states:** During the learning process the number of states is continuously growing. Whenever a new state is observed, which is not available in the model, it is added to the model.
2) **Number of transitions:** Additionally, the number of transitions has to be considered. Even when no states are added anymore, still transitions between states can be added.
3) **Changing (enlarging) the time bounds:** Finally, the changing of the time bounds has to be considered. Even when no states and transitions are added to the model anymore, still the time bounds (the minimum and maximum time value for each transition) can be changing.

The learning can be considered as finished, when no more states and transitions are added to the model and when the time bounds of the transitions do not change anymore. In practice, this is not easy to decide, since of course we cannot look into the future and therefore we cannot know if one of the values will change in the future.

So we have to decide based on the past, whether learning is finished. If for a *certain amount of time* nothing changed, the learning process can be considered as finished. The question here is: What does ”certain amount of time” mean?

Figure 5 shows a typical model convergence curve, where the number of changes (states, transitions, timing) is plotted against the running number of incoming events.

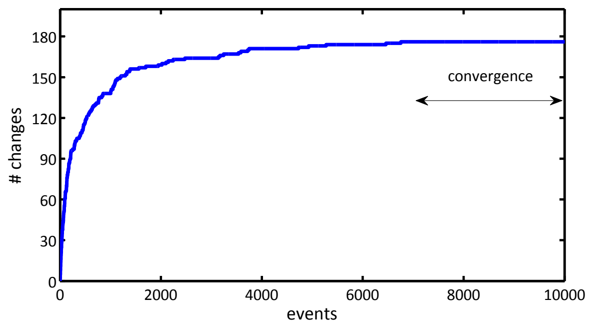

Fig. 5. The model changing curve and learning convergence for a test data set.

A trivial yet efficient possibility is to consider the last n_{conv} samples and check whether the model has been changed concerning the characteristics in the aforementioned enumeration. The value n_{conv} can be chosen arbitrarily, a good choice is dependent on the observed process. However, setting $n_{conv} = 1000$ is empirically in most cases a good choice.

Another method takes the variable number of states into account. For this, the number of states is multiplied with a factor f_{conv} which also can be chosen arbitrarily. $f_{conv} \approx 50$ in most cases is a good choice. Hence, the learning converged if the last $n_{conv} = |S| \cdot f_{conv}$ events didn't contain any changes. Indeed, beginning with the learning process, the value of n_{conv} has to be set to an initial value, e.g. $n_{conv} = 200$ such that the learning does not abort too early as long as the number of states is small.

D. Runtime Analysis

The reader may has noticed that the runtime is exponential. Applying assumption 1, the number of possible states is given by the number of possible combinations of each input and output signals: $|S| = 2^{|IO|}$. In each learning step, all possible states have to be checked for equivalence. Therefore, the worst case runtime for each incoming event is in $\mathcal{O}(2^{|IO|})$.

To obtain a model with the worst case number of states we used the drunken sailor simulation (a random walk) to generate a model such that each state in a discrete state space will be visited at some point in time. Figure 6 shows the runtime behavior of the model learning of the drunken sailor simulation with 10 signals. The total number of states is $|S| = 2^{10} = 1024$. The growing number of states is shown in Figure 7.

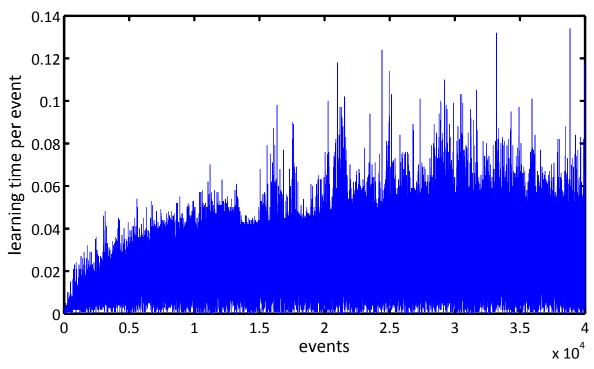

Fig. 6. Runtime behavior of OTALA.

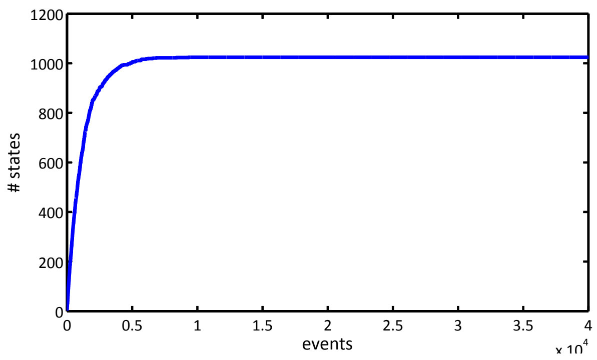

Fig. 7. Convergence behavior of OTALA.

In practice however, the systems show a better behavior. Often signals are only switched on and off together in a pair or group (e.g. two conveyor belts to transport something) or combinations of signal groupings can be precluded (e.g. the filling of a container can only be started if the container is not full). For each excluded combination possibility the number of possible states reduces by half. This reduces the runtime of the learning process.

The runtime also can be reduced, if more memory space is used. For each possible state, memory space is allocated and the state ID corresponds to the decimal representation of the IO-vector. During runtime of the algorithm, it is not necessary to iterate over all possible states, just the IO-vector has to be converted into a decimal number and the corresponding memory space has to be addressed. The drawback of this method is the high amount of needed memory space and in usual cases, it is not used at all, but since the number of IOs is known beforehand, the needed amount of memory space can be predetermined.

The difference between the space and time optimized version of OTALA can be seen in figure 8. Here, the cumulative runtime over all incoming events is used. Both show a rather linear runtime behavior, the time optimized version runs about 10 times faster. The time optimized allocates the maximum amount of space at the beginning, even if it is not needed, whereas the space optimized version only allocates memory space when it is necessary.

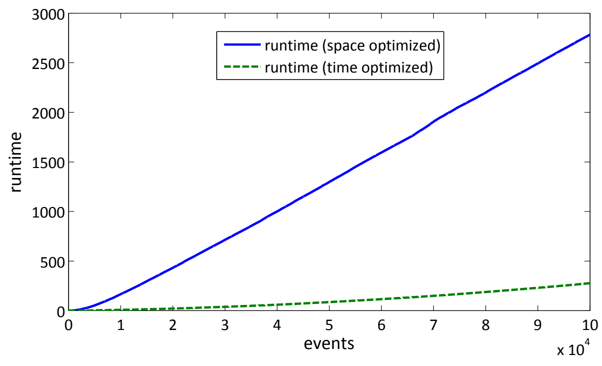

Fig. 8. Difference between the space and time optimized version of OTALA.

IV. Evaluation and Discussion

In this section the application point of view is used to evaluate the usability of OTALA. For this, representative for other offline learning algorithms, the offline learning algorithm BUTLA (see section II-A) is used for comparison. For a comprehensive evaluation of BUTLA with other algorithms, please refer to [14].

A. Online vs. offline learning algorithm

The learning time of online and offline learning algorithms cannot directly be compared. As showed before, the learning time of OTALA is exponential in the number of IOs, while the runtime of BUTLA is cubic in the number of states in the prefix tree acceptor (the number of states in the prefix tree acceptor grows with the amount of observations).

BUTLA works best if the learning data is separated into cycles such that the prefix tree acceptor is highly condensed. However, this often cannot be guaranteed for cyber-physical production systems. In that case the prefix tree acceptor can consist of far more than one hundred thousand of states in a linked list. And since the learning time is cubic in the size of the prefix tree acceptor, this would lead to a high computation time.

In contrast, the runtime of OTALA is only based on the number of IOs and not on the amount of data. Therefore, the cumulated learning time over the samples is rather linear (as can also be seen in figure 8). In cyber-physical production

TABLE I. COMPARISON OF OTALA WITH OFFLINE LEARNING ALGORITHM (E.G. BUTLA)

	OTALA	offline
not possible to use expert knowledge	+	-
limited memory space	+	-
amount of needed learning data is not known beforehand	+	-
observations can not be separated into cycles automatically	+	-
small number of signals	+	+
data can be separated into cycles	+	+
huge number of signals	-	+
states do not correspond to signal vectors	-	+

systems, the number of IOs is known beforehand. Therefore, the needed learning time can roughly be predicted. Additionally, in most cases, the number of IOs is limited to a small number. This makes OTALA applicable, despite the exponential runtime. If more space can be used, the runtime further can be reduced.

Furthermore, OTALA is able to stop the learning process if the identified model converged to the target model, such that not all data have to be considered for learning.

Summarized, comparing online (OTALA) and offline (BUTLA) learning algorithms, the following recommendations can be given:

Use an online learning algorithm (e.g. OTALA), when:

- memory space is limited, such that the observations cannot be stored,
- it is not possible to use expert knowledge at all,
- the number of needed learning samples and the convergence of learning progress is not known beforehand or
- the observations can not be separated into cycles which form a dense prefix tree acceptor.

On the contrary, use an offline learning algorithm (e.g. BUTLA), when:

- the number of signals is huge,
- the states of the target model can not be described by signal vectors and equal events can arise subsequently or
- the learning data can be separated into cycles.

Table I summarizes the comparison of the online learning algorithm OTALA with the offline learning algorithm BUTLA.

B. Use Case

For the evaluation of the learning algorithm we used an experimental production plant in our institute, the smart factory (see figure 9). It is an exemplary plant which uses different techniques to transport and store bulk material, to fill it in bottles and finally use for production.

The identified models have been used for anomaly detection. For this, the behavior of the running production system

Fig. 9. Smart Factory as exemplary plant for experiments.

is compared with the prediction of the identified model. A difference hints at an anomaly [4].

For the evaluation, a part of the smart factory has been used. It comprises 10 IOs (6 inputs, 4 outputs). In a large-scale experiment in which the amount of data was extended by simulation, 5000 production cycles have been generated. An expert of the smart factory guessed that 200 production cycles should be enough to identify a correct behavior model. To keep the scenario realistic, only the first 200 samples were used by BUTLA to learn a model. OTALA needed 500 samples to converge. Both OTALA and BUTLA identified a model with 15 states.

Then, 3000 positive and 100 negative samples where used for anomaly detection. The result is given in the confusion matrix (according to [15]) in Table II.

- True Positives (TP) are samples, which are really anomalous and detected as such by the anomaly detection algorithm.
- False Positives (FP) are samples that are free of anomalies, but recognized as anomalous.
- False Negatives (FN) samples contain anomalies, which are not detected by the algorithm.
- True Negatives (TN) are really normal samples that are recognized as such by the algorithm.

The accuracy is calculated as:

$$Accuracy = \frac{f_{TP} + f_{TN}}{f_{TP} + f_{TN} + f_{FP} + f_{FN}} \quad (1)$$

TABLE II. CONFUSION MATRIX FOR EXPERIMENT IN THE SMART FACTORY.

	true positive f_{TP}	true negative f_{TN}	false positive f_{FP}	false negative f_{FN}	Accuracy
OTALA	100%	98%	2%	0%	99%
BUTLA	100%	76%	24%	0%	88%

It can be seen, that all inserted errors could be found with both identified models (100% true positives and 0% false negative). However, since BUTLA used too little learning examples, the value for false positives is very high (24%). This clarifies the problem that is not known in advance how many samples are needed for learning. The number of false positives for OTALA is smaller (2%) since the learning converged and only some outliers have not been used for learning.

In the experiment we used only 200 samples for learning the model with BUTLA, because this number was guessed by an expert. This amount of data was not sufficient to converge, and therefore, the accuracy is worse than using OTALA (which learned until convergence). However, using the same number of learning samples for BUTLA, it reaches the same accuracy as OTALA.

It can be seen, that both algorithms produce good results, but it reveals the weakness of BUTLA: It is not possible to determine how many learning samples are required to learn a correct model, whereas OTALA is able to recognize convergence autonomously and therefore uses the minimum number of learning samples only. Therefore, OTALA is especially suited for the case if only little knowledge about the process is available.

The exponential runtime of OTALA means that it is not suited for systems with a high amount of IOs. Nevertheless, according to [4] and [13] it is not useful to learn one overall model for the whole system but rather for subsets of the IOs. That means, that only those signals are used to learn a model, which belong to the same subprocess. This again reduces the number of used signals and makes OTALA applicable.

V. Conclusion

In this contribution we presented OTALA, the first online passive learning algorithm for timed automata. A special focus lies on the applicability for the learning of the timing behavior of cyber-physical production systems. The algorithm only needs positive examples, which means that no observations of faulty behavior are needed, which are hardly producible in technical systems.

A possible application might look like this: OTALA is used to learn a behavior model of the production system. The learning itself runs without expert knowledge, as well as the detection of convergence. After finishing the learning process, it can be switched automatically to anomaly detection. Here, an algorithm as proposed in [4] can be used.

We showed that, although the worst case runtime is exponential, due to the limited number of IOs, OTALA is still applicable for the learning of timed behavior in cyber-physical production systems. The runtime further can be reduced by using more memory space. Since the number of IOs is known beforehand, the runtime per event and the overall memory consumption can be predetermined.

So far, OTALA works with discrete signals only, i.e. it learns a discrete Timed Automaton. In future work, the algorithm will be extended to the learning of hybrid Timed Automata including continuous signals into the states.

Acknowledgment

The research for this paper is funded within the AVA project by the "Bundesministerium für Bildung und Forschung" in Germany.

References

[1] R. C. Carrasco and J. Oncina, "Learning stochastic regular grammars by means of a state merging method," in *GRAMMATICAL INFERENCE AND APPLICATIONS*. Springer-Verlag, 1994, pp. 139–152.

[2] 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.

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

[4] A. Maier, O. Niggemann, A. Vodenčarević, R. Just, and M. Jaeger, "Anomaly detection in production plants using timed automata," in *8th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*. Noordwijkerhout, The Netherlands, Jul 2011.

[5] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *Journal of the American Statistical Association*, vol. 58, no. 301, pp. pp. 13–30, 1963.

[6] K. Lang, B. Pearlmutter, and R. Price, "Results of the abbadingo one dfa learning competition and a new evidence-driven state merging algorithm," 1998.

[7] S. Tong, "Active learning: Theory and applications," Ph.D. dissertation, Stanford University, Stanford, CA, USA, 2001.

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

[9] O. Grinchtein, B. Jonsson, and M. Leucker, "Learning of event-recording automata," in *In 6th International Workshop on Verification of Infinite-State Systems, volume 138/4 of Electronic Notes in Theoretical Computer Science*. Springer, 2004, pp. 379–395.

[10] E. Lee, "Cyber physical systems: Design challenges," in *Object Oriented Real-Time Distributed Computing (ISORC), 2008 11th IEEE International Symposium on*, 2008, pp. 363–369.

[11] R. R. Rajkumar, I. Lee, L. Sha, and J. Stankovic, "Cyber-physical systems: The next computing revolution," in *Proceedings of the 47th Design Automation Conference*, ser. DAC '10. New York, NY, USA: ACM, 2010, pp. 731–736.

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

[13] M. Roth, S. Schneider, J.-J. Lesage, and L. Litz, "Fault detection and isolation in manufacturing systems with an identified discrete event model," *Int. J. Systems Science*, vol. 43, no. 10, pp. 1826–1841, 2012.

[14] A. Vodenčarević, A. Maier, and O. Niggemann, "Evaluating learning algorithms for stochastic finite automata," in *2nd International Conference on Pattern Recognition Applications and Methods (ICPRAM 2013); Barcelona, Spain*, Feb 2013.

[15] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining, (First Edition)*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2005.

View publication stats