Passive Learning of Timed Automata from Logs (Student Abstract)

Lénaïg Cornanguer[*]
Inria, Univ Rennes, CNRS, IRISA
F-35000 Rennes
lenaig.cornanguer@irisa.fr

Abstract

We propose a novel algorithm to passively learn deterministic Timed Automata from events sequences associated with the delay occurring between them. This algorithm produces models that are more specific than State-of-the-Art algorithms and that has a better identification of the temporal constraints applying on the systems.

Introduction

Many systems save a history of the events occurring within it or linked to it. These timestamped events sequences are named logs and the main purposes of logging are to track the evolution of a process or to identify the cause when a problem is occurring. If each logging sequence informs us about one specific execution of the system, looking at the whole set of logs can give us an insight of the global functioning of the system. When the behavior of a system is time-dependant, this functioning can be complex, and obtaining its behavior model helps to understand it. One behavior model is the Timed Automaton (TA). Traditionally used to model electronic circuits and ensure their safety, TA are now used to model biological ecosystems, software, or industrial systems. The first strength of this modeling is its expressiveness: in addition to the classical automata expressiveness, TA support temporal constraints which can be verified with temporal logics such as TCTL (Alur, Courcoubetis, and Dill 1990). This expressiveness has lead to the development of numerous tools based on TA. Some of these tools were created to verify properties (e.g. safety specifications) of the represented system thanks to timed logical expressions. Other ones allow the generation of test case scenarios with the controller synthesis. Finally, TA's formalism can be translated into a state-transition diagram, which makes it easily interpretable.

Given timestamped event sequences, learning a TA modelling the sytem's behavior is a recent issue. We can distinguish two learning approaches, the active and the passive one. While active approaches require to query the system, passive approaches can learn TA only with labeled sequences. The existing algorithms that passively learn TA tend to lack precision due to their management of the time constraints or to produce too complex models, which reduce the interpretability.

We propose a novel algorithm named TAG that passively learns deterministic TA from timestamped event sequences and use a new strategy to obtain a good trade-off between the size of the models and their precision.

Background and State-of-the-Art

Timed Automata

A Timed Automaton is a data structure to model the functioning of dynamic systems whose state is conditioned by the occurrence of discrete events at specific moments. The notion of time is supported by temporal variables named clocks that can be reset during the transition from one state to another (e.g., the clock "$c0$" on the transition from S1 to S0 in Figure 1) and whose value conditions the transition from a state to another (e.g., "$c0 < 5$" on the same transition as before). Formally, a Timed Automaton A is a 6-uple $A = (Q, \Sigma, C, E, q_0, F)$ where Q is a finite set of states, Σ is a finite set of events or symbols, C is a finite set of clocks, E is a finite set of transitions, $q_0 \in Q$ is the initial state and $F \subseteq Q$ is a finite set of final states. A transition $e \in E$ of A is a 5-uple $e = (q, a, g, r, q')$ where q and q' are respectively the source state and the destination state of e, g is a guard i.e. a constraint on the value of a clock, and r is the set of clocks being reset on the transition.

Timed Automata Passive Learning Algorithms

Learning a TA is a NP-hard problem (Verwer, de Weerdt, and Witteveen 2010). To reduce the complexity of the task, the existing approaches use approximations or learn a subclass of Timed Automata.

GenProgTA (Tappler et al. 2018) is an algorithm that learns TA from positive timestamped event sequences based on genetic programming. This algorithm requires the user to give a lot of parameters such as the number of clocks or the relevant guards.

Timed k-Tail (Pastore, Micucci, and Mariani 2017) is an algorithm that learns TA from a positive set of operations and their duration. For each state of the initialized automaton, it compares the set of possible trajectories of k transitions starting from this state. This set is called the k-future

[*]Supervisors: C. Largouët, L. Rozé and A. Termier.

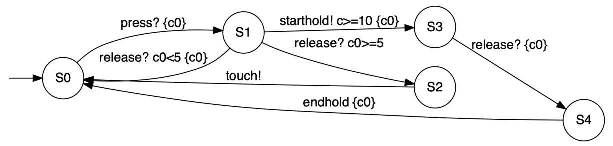

Figure 1: The model TA from GenProgTA used to generate the timestamped sequences of events.

of the state and the states with the same k-future are merged. One of the drawbacks of Timed k-Tail is that it learns an automaton that isn't deterministic in contrast to the algorithm we propose.

RTI+ (Verwer, de Weerdt, and Witteveen 2010) is an algorithm that learns Probabilistic Deterministic Real-Time Automata (PDRTA) from only positive timestamped event sequences. A PDRTA is a TA where the notions of clock and guards are reduced to intervals of acceptable delays between two successive events. It is also deterministic. RTI+ introduces the notion of transition division which permits to model the influence of time on the target state for the same event. However, its management of the temporal constraints leads to models not sensitive enough.

Contribution

We propose a novel algorithm named TAG for Timed Automata Generator, its strategy is in two phases.

After having initialized an automaton with an input sample of positive timestamped event sequences, it reduces its size by merging the states having the same k-future, like Timed k-Tail. However, if the merge induces a determinism issue, it also merges the problematic transitions and states until a deterministic solution.

The second phase attempts to capture the temporal dynamic of the system by realizing transition divisions. If the event occurring after a transition differs in function to the time value of passage, the transition and its target state are split into two paths. During this second phase, it also realizes state merging if needed, but always by favoring the transition divisions.

It finally returns a PDRTA consistent with the input sample, concise in terms of state number but capturing the temporal constraints of the system.

Experimentation

Experimental Setup

In order to compare our new algorithm with algorithms of the State-of-the-Art, we choose to use an example of GenProgTA (Figure 1). We generated a set of 1000 event sequences with the delay occurring between two events consistent with the model TA. 70% of this set was used to learn a TA with each algorithm. The 30% remaining were used to test the models by checking if they were accepted by it and compute the True Positive Rate (TPR). We also generated 300 sequences inconsistent with the model automaton to compute the False Positive Rate (FPR).

Algorithm	TPR	FPR	State number
GenProgTA	0.94	0.00	5
RTI+	1.00	0.22	6
TAG	1.00	0.00	5

Table 1: Validation scores of the learned models.

Results

We did not compute the TPR and the FPR of the TA learned by Timed k-Tail because it is not deterministic. This TA has 11 states. The TPR, FPR, and the state number of the TA learned by the other algorithms are presented in Table 1. The model learned by GenProgTA did not recognize all the sequences of the positive set of validation due to an error on a guard. However, it rejected all the sequences of the negative one on the contrary to RTI+'s model which accepted 66 negative sequences. The cause is a bad management of the temporal constraints leading to a model with guards too large. Finally, our algorithm TAG produced the exact same automaton as the target and therefore correctly accepts or rejects the sequences of the validation set.

Conclusion and Future Work

Learning passively TA from logs can be really interesting to model systems without having to know its functioning or having to interact with it.

We propose a novel algorithm that passively learns TA and achieves a trade-off between a model that would be too sensitive to the input data and a model that wouldn't be specific enough and accept wrong behaviors.

The next step of our work is to improve the validation of the resultant TA. A study about the distances between TA has to be done to consolidate the validation. An other step is the formal verification of the algorithm. Then, we would like to delete progressively the constraints on the learned automaton to obtain models fully exploitable by the model checking or controller synthesis tools.

References

Alur, R.; Courcoubetis, C.; and Dill, D. 1990. Model-checking for real-time systems. In *[1990] Proceedings. Fifth Annual IEEE Symposium on Logic in Computer Science*, 414–425.

Pastore, F.; Micucci, D.; and Mariani, L. 2017. Timed k-tail: Automatic inference of timed automata. In *2017 IEEE International conference on software testing, verification and validation (ICST)*, 401–411.

Tappler, M.; Aichernig, B. K.; Larsen, K. G.; and Lorber, F. 2018. Learning Timed Automata via Genetic Programming. *arXiv:1808.07744 [cs]* .

Verwer, S.; de Weerdt, M.; and Witteveen, C. 2010. A likelihood-ratio test for identifying probabilistic deterministic real-time automata from positive data. In *International Colloquium on Grammatical Inference*, 203–216. Springer.