Improved Learning for Stochastic Timed Models by State-Merging Algorithms

Braham Lotfi Mediouni(✉), Ayoub Nouri, Marius Bozga, and Saddek Bensalem

Université Grenoble Alpes, CNRS, VERIMAG, 38000 Grenoble, France
braham-lotfi.mediouni@univ-grenoble-alpes.fr

Abstract. The construction of faithful system models for quantitative analysis, e.g., performance evaluation, is challenging due to the inherent systems' complexity and unknown operating conditions. To overcome such difficulties, we are interested in the automated construction of system models by learning from actual execution traces. We focus on the timing aspects of systems that are assumed to be of stochastic nature. In this context, we study a state-merging procedure for learning stochastic timed models and we propose several enhancements at the level of the learned model structure and the underlying algorithms. The results obtained on different examples show a significant improvement of timing accuracy of the learned models.

1 Introduction

A necessary condition for a successful system design is to rely on faithful models that reflect the actual system behavior. In spite of the long experience designers have on building system models, their construction remains a challenging task, especially with the increasing complexity of recent systems. For performance models, this is even harder because of the inherent complexity and the induced stochastic behavior that is usually combined with time constraints.

Machine Learning (ML) is an active field of research where new algorithms are constantly developed and improved in order to address new challenges and new classes of problems (see [13] for a recent survey). Such an approach allows to automatically build a model out of system observations, i.e., given a learning sample S, a ML algorithm infers an automaton that, in the limit[1], represents the language L of the actual system [2]. We believe that ML can be used to automatically build system models capturing performance aspects, especially the timing behavior and the stochastic evolution. Those system models may be useful for documenting legacy code, and for performing formal analyses in order to enhance the system performance, or to integrate new functionalities [11].

Despite the wide development of ML techniques, only few works were interested in learning stochastic timed models [9,10,12,14]. In this paper, we study the RTI+ algorithm [14] and we propose improvements that enhance its accuracy. This algorithm learns a sub-class of timed automata [1] augmented with

[1] By considering a sufficient number of observations [4].

C. Barrett et al. (Eds.): NFM 2017, LNCS 10227, pp. 178–193, 2017.
DOI: 10.1007/978-3-319-57288-8_13

probabilities, called Deterministic Real-Time Automata (DRTA). Given a timed learning sample S (traces of timestamped actions of the system), the algorithm starts by building a tree representation of S, called Augmented Prefix Tree Acceptor (APTA). Then, based on statistical tests, it performs state-merging and splitting operations until no more operations are possible. In this algorithm, clock constraints are captured as time intervals over transitions and are built in a coarse fashion. These time intervals are actually considered to be the largest possible, which makes the learning procedure converge faster. However, this introduces a lot of generalization in the built APTA, by allowing timing behaviors that are not part of the actual system language L. Furthermore, we identified that such behaviors cannot be refined during the learning process. The learned model is thus not accurate from a timing point of view.

In this work, we propose a more accurate learning procedure by investigating better compromises between the time generalization introduced in the APTA and its size (and consequently its learning time). We introduce three new APTA models representing different levels of time generalization; the first model is the exact representation of the learning sample, i.e., with no generalization, while the two others introduce some generalization which is less than the original RTI+. We implemented the new variants of the RTI+ algorithm and validated them on different examples. The obtained results show that the learned models are more accurate than the original implementation, albeit the learning time is generally higher.

Outline. In Sect. 2, we discuss some related works on learning stochastic timed models. Section 3 introduces notations and key definitions used in the rest of the paper. We recall the RTI+ learning algorithm and study underlying time representation issues in Sect. 4. In Sect. 5, we present the three improvements we propose for RTI+ and discuss them. Section 6 presents experiments and results of the improved algorithms. Conclusions are drawn in Sect. 7.

2 Related Works

In the literature, several algorithms have been proposed for automata learning [2,3,7,14], mostly in the deterministic case. In the last decades, an increasing interest has been shown for learning probabilistic models, partly due to the success of verification techniques such as probabilistic and statistical model checking [5,6]. Despite this development, only few works considered the problem of learning stochastic timed models [9,10,12,14].

Most of the algorithms proposed in this setting are based on the state-merging procedure made popular by the Alergia algorithm [3]. Moreover, many of them consider Continuous-Time Markov Chains (CTMCs) as the underlying model. For instance, in [12], an algorithm is proposed for model-checking black-box systems. More recently, the AAlergia algorithm [8], initially proposed for learning Discrete-Time Markov Chains and Markov Decision Processes, was extended to

learn CTMCs [9]. This work is an extension of Alergia to learn models having timed in/out actions.

Other algorithms such as RTI+ [14] and BUTLA [7] focus on learning timed automata augmented with probability distributions on discrete transitions and uniform probabilities over timing constraints. Both follow a state-merging procedure but consider different statistical tests for checking states compatibility.

In [10], authors focus on learning more general stochastic timed models, namely Generalized Semi-Markov Processes, following the same state-merging procedure. This algorithm relies on new statistical clocks estimators, in addition to the state compatibility criterion used in Alergia.

3 Background

Let Σ be a finite alphabet, Σ^* the set of words over Σ and ϵ the empty word. Let $\mathbb{T}$ be a time domain and $\mathbb{T}^*$ the set of time sequences over $\mathbb{T}$. In our work, we consider integer time values, i.e., $\mathbb{T} \subseteq \mathbb{N}$. For a set of clocks $\mathcal{C}$, let $CC(\mathcal{C})$ denote the set of clock constraints over $\mathcal{C}$. Let $\mathcal{I}$ be the intervals domain, where $I \in \mathcal{I}$ is an interval of the form $[a; b],\ a, b \in \mathbb{T}$ such that $a \leq b$, and represents the set of integer values between a and b. Let ω be an untimed word over Σ and τ a time sequence over $\mathbb{T}$. We write $\omega \leq \omega'$ (resp. $\tau \leq \tau'$) whenever ω (resp. τ) is a prefix of ω' (resp. τ'). We also write ωu (resp. τv) for the concatenation of ω (resp. τ) and $u \in \Sigma^*$ (resp. $v \in \mathbb{T}^*$). $|\omega|$ (resp. $|\tau|$) is the size of ω (resp. τ).

3.1 Deterministic Real-Time Automata (DRTA)

A Real-Time Automaton (RTA) is a timed automaton with a single clock that is systematically reset on every transition.

Definition 1 (Real-Time Automaton (RTA)). *An RTA is a tuple* $\mathcal{A} = \langle \Sigma, L, l_0, \mathcal{C}, T, inv \rangle$ *where: (1)* Σ *is the alphabet, (2)* L *is a finite set of locations, (3)* $l_0 \in L$ *is the initial location, (4)* $\mathcal{C} = \{c\}$ *contains a single clock, (5)* $T \subseteq L \times \Sigma \times CC(\mathcal{C}) \times \mathcal{C} \times L$ *is a set of edges with a systematic reset of the clock* c*, (6)* $inv : L \longrightarrow CC(\mathcal{C})$ *associates invariants to locations.*

For more convenience, transitions are denoted as $l \xrightarrow{\sigma, I} l'$, where $\sigma \in \Sigma$ and $I \in \mathcal{I}$ is a time interval including both transition guards and location invariants. For simplicity, we also omit the systematic reset.

An RTA is deterministic (DRTA) if, for each location l and a symbol σ, the timing constraints of any two transitions starting from l and labeled with σ are disjoint, i.e., $\forall l \in L, \forall \sigma \in \Sigma, \forall t_1, t_2 \in T,\ t_1 = \langle l, \sigma, I_1, l_1 \rangle$ and $t_2 = \langle l, \sigma, I_2, l_2 \rangle, (I_1 \cap I_2 \neq \emptyset) \Leftrightarrow (t_1 = t_2)$. A DRTA generates timed words over $\Sigma \times \mathbb{T}$. Each timed word is a sequence of timed symbols $(\omega, \tau) = (\sigma_1, \tau_1)(\sigma_2, \tau_2)...(\sigma_m, \tau_m)$, representing an untimed word ω together with a time sequence τ. A set of n timed words constitute a *learning sample* $S = \{(\omega, \tau)^i, i \in [1; n], \omega \in \Sigma^*, \tau \in \mathbb{T}^*\}$. We denote by $\mathbb{T}^S$ the set of time values appearing in S.

A Prefix Tree Acceptor (PTA) is a tree representation of the learning sample S where locations represent prefixes of untimed words in S. Timing information is captured in a PTA in form of intervals $I \in \mathcal{I}$ over transitions. This structure is called Augmented PTA (APTA). In the latter, each transition $l \xrightarrow{\sigma,I} l'$ is annotated with a frequency that represents the number of words in S having l' as a prefix. An APTA can be seen as an acyclic DRTA annotated with frequencies. Let $\mathcal{N} : T \longrightarrow \mathbb{N}$ be this annotation function. Given a DRTA $\mathcal{A}$, a pair $(\mathcal{A}, \mathcal{N})$ is an *annotated* DRTA, denoted $DRTA^+$.

3.2 Stochastic Interpretation of a DRTA

A DRTA starts at the initial location l_0 with probability 1. It moves from a location to another using transitions in T. At each location l, a transition is chosen among the set of available transitions T_l. Selecting a transition consists of choosing a timed symbol (σ_i, τ_i). A probabilistic strategy φ that associates a probability function to each location l over the set of transitions T_l is used to make this choice: $\varphi : L \times T \longrightarrow [0, 1]$, such that $\Sigma_{t \in T_l}\ \varphi(l, t) = 1, \forall l \in L$. For the chosen transition $l \xrightarrow{\sigma,I} l'$, the choice of the time value is done uniformly over the time interval I. Figure 1 shows an example where two transitions labeled A and B are possible from location 1. The strategy φ associates probability 0.6 to A and 0.4 to B. Then, uniform choices on the associated time intervals are performed.

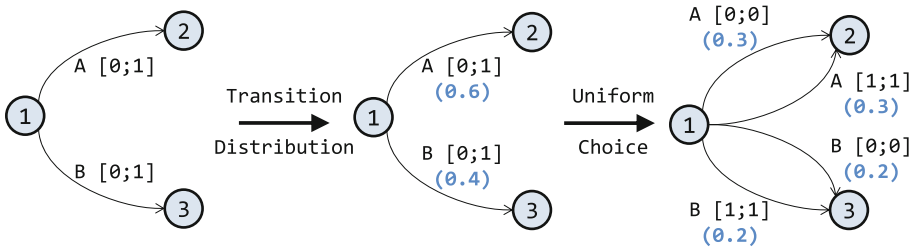


Fig. 1. Probabilistic strategy with uniform choice

4 The RTI+ Learning Procedure

RTI+ [14] is a state-merging algorithm for learning DRTA models from a sample of timed words. The algorithm first builds a PTA then reduces it by merging locations having similar behaviors, according to a given compatibility criterion. Compared to other state-merging algorithms, RTI+ relies on a time-split operation to identify the different timed behaviors and to split them into disjoint ones. The algorithm is able to learn a stochastic DRTA, i.e., a $DRTA^+$ where the strategy is obtained from the associated annotation function $\mathcal{N}$.

4.1 Building the APTA

The timed learning sample is represented as an APTA where all the time intervals span over $\mathbb{T}$. Initially, the built APTA only contains a root node consisting of the empty word ϵ. RTI+ proceeds by adding a location in the tree for each prefix of untimed words in S. Then, a transition labeled with σ is created from location l to location l' if the prefix of l' is obtained by concatenating the prefix of l and symbol σ. Finally, transitions are augmented with the largest time constraint $[0; max(\mathbb{T}^S)]$, where $max(\mathbb{T}^S) = Max\{a \in \mathbb{T}^S\}$. The annotation function $\mathcal{N}$ is built at the same time and represents transitions frequencies. In this work, we denote this construction as *generalized-bound APTA*.

Definition 2 (Generalized-bound APTA). *A generalized-bound APTA is a* $DRTA^+$ $(\langle \Sigma, L, l_0, \mathcal{C}, T, inv\rangle, \mathcal{N})$ *where:*

- $L = \{\omega \in \Sigma^* \mid \exists(\omega', \tau') \in S, \omega \leq \omega'\}$, $l_0 = \epsilon$,
- T *contains transitions of the form* $t = \langle \omega, \sigma, [0; max(\mathbb{T}^S)], \omega' \rangle$ *s.t.* $\omega\sigma = \omega'$, *and* $\mathcal{N}(t) = |\{(\omega' u, \tau) \in S, u \in \Sigma^*\}|$.

4.2 The Learning Process

The learning process aims to identify the $DRTA^+$ that represents the target language while reducing the size of the initial APTA. At each iteration, the algorithm first tries to identify the timing behavior of the system, by using time-split operations. The second step consists of merging compatible locations that show similar stochastic and timed behaviors. Locations that are not involved in merge or split operations are marked as belonging to the final model (promote operation). The algorithm proceeds by initially marking the root of the APTA as part of the final model and its successors as candidate locations. The latter will be considered for time-split, merge or promote operations.

Time-Split Operation. For a given transition $t = \langle l, \sigma, [a; b], l' \rangle \in T$, splitting t at a specific time value $c \in [a; b]$ consists of replacing t by two transitions t_1 and t_2 with disjoint time intervals $[a; c]$ and $[c+1; b]$, respectively. This operation alters the subtree of l' such that the corresponding timed words that used to trigger transition t with time values in $[a; c]$ (resp. $[c + 1; b]$) are reassigned to the subtree pointed by transition t_1 (resp. t_2).[2]

Merge Operation. Given a marked location l (belongs to the final model) and a candidate location l', this operation is performed by first redirecting the transitions targetting l', to l and then by folding the subtree of l' on l (see footnote 2).

[2] For further details, see: http://www-verimag.imag.fr/~nouri/drta-learning/Appendice.pdf.

4.3 Compatibility Evaluation

The compatibility criterion used in RTI+ is the Likelihood Ratio (LR) test. Intuitively, this criterion measures a distance between two hypotheses with respect to specific observations. In our case, the considered hypotheses are two $DRTA^+$ models: H with m transitions and H' with m' transitions ($m < m'$), where H is the model after a merge operation (resp. before a split) and H' is the model before a merge (resp. after a split). The observations are the traces of S.

We define the likelihood function that estimates how S is likely to be generated by each model (H or H'). It represents the product of the probability to generate each timed word in S[3]. Note that the i^{th} timed word $(\omega, \tau)^i$ in S corresponds to a unique path p^i in the $DRTA^+$. The probability to generate $(\omega, \tau)^i$ is the product of the probabilities of each transition in $p^i = s_1 s_2 ... s_{|\omega|}$:

$$f((\omega, \tau)^i, H) = \prod_{j=1}^{|\omega|-1} \pi((\omega_j, \tau_j)^i, s_j)$$

Where $\pi((\omega_j, \tau_j)^i, s_j)$ corresponds to the probability to transit from the location s_j to s_{j+1} in H with the j^{th} timed symbol $(\omega_j, \tau_j)^i$. Given a learning sample S of size n, the likelihood function of H is $Likelihood(S, H) = \prod_{i=1}^{n} f((\omega, \tau)^i, H)$. The likelihood ratio is then computed as follows

$$LR = \frac{Likelihood(S, H)}{Likelihood(S, H')}$$

Let Y be a random variable following a χ^2 distribution with $m' - m$ degrees of freedom, i.e., $Y \rightsquigarrow \chi^2(m' - m)$. Then, $y = -2ln(LR)$ is asymptotically χ^2 distributed. In order to evaluate the probability to obtain y or more extreme values, we compute the p-value $pv = P(Y \geq y)$. If $pv < 0.05$, then we conclude that H and H' are significantly different, with 95% confidence.

The compatibility criterion concludes that a time-split operation is accepted whenever it identifies a new timing behavior, that is, the model after split is significantly different from the model before split ($pv < 0.05$). In constrast, a merge is rejected whenever the model after merge is significantly different from the model before merge since the merged locations are supposed to have similar stochastic and timed behaviors.

4.4 Shortcomings

The RTI+ algorithm relies on the generalized-bound APTA as initial representation of the learning sample S. As pointed out before, this kind of APTA augments an untimed PTA with the largest possible time intervals, without considering the values that concretely appear in S. This introduces an *initial generalization* that leads the APTA to accept words that do not actually belong to S and might

[3] Since the timed words in S are generated independently.

not belong to the target language L. In Example 1, we show that this initial generalization cannot be refined later in the learning process. More concretely, we observe that the time intervals that do not appear in S are not isolated and removed from the $DRTA^+$.

Example 1. Let us consider the following learning sample $S = \{$(A,5)(B,5); (A,4)(A,3); (A,3)(B,5); (B,1)(A,5); (B,3)(B,5); (B,5)(A,1)$\}$. The left-hand model in Fig. 2 presents the initial $DRTA^+$ (H) of S on which we evaluate a time-split operation. The latter is expected to identify the empty interval $[0;2]$ on transition $t = \langle 1, A, [0;5], 2\rangle$, since no timed word in S takes this transition with time values in $[0;2]$. The right-hand figure represents the model assuming a split of transition t at time value 2 (H'). The LR test returns $pv = 1$ which leads to reject the time-split operation, and hence, the empty interval $[0;2]$ is not identified during the learning process.

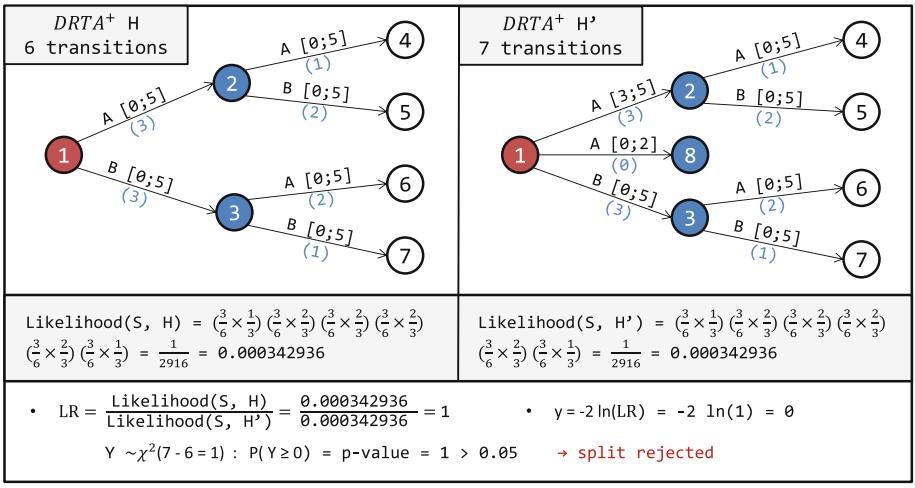


Fig. 2. Identifying empty intervals with time-split operation

The generalized-bound APTA introduces empty time intervals that cannot be removed during the learning process. To overcome this issue, we propose, in the next section, new representations of the learning sample S.

5 Learning More Accurate Models

A faithful representation of the learning sample consists of building an APTA that accepts only words in S by taking into account the time values. This can be done at different granularities, which results on different tradeoffs between the introduced initial generalization and the APTA size. We propose three different APTA models denoted unfolded, constructive-bound and tightened-bound APTAs.

5.1 Unfolded APTA

This APTA model fits perfectly the traces in S, that is, accepts exactly the timed words in S. Hence, it does not introduce any initial generalization. To build such a model, we need to consider both symbols and time values. The APTA initially contains the empty word. Locations are added for every timed prefix and corresponding transitions are created such that each transition only accepts a single time value, i.e., time intervals are equalities of the form $[a; a] \in \mathcal{I}$.

Definition 3 (Unfolded APTA). *An unfolded APTA is a DRTA$^+$ where:*

- $L = \{(\omega, \tau) \in \Sigma^* \times \mathbb{T}^* | \ \exists (\omega', \tau') \in S, \omega \leq \omega', \tau \leq \tau'\}$,
- T *contains transitions of the form* $t = \langle (\omega, \tau), \sigma, [a; a], (\omega', \tau') \rangle$ *such that: (1)* $\omega\sigma = \omega'$, *(2)* $\tau a = \tau'$, *and* $\mathcal{N}(t) = |\{(\omega' u, \tau' v) \in S, u \in \Sigma^*, v \in \mathbb{T}^*\}|$.

5.2 Constructive-Bound APTA

A more compact representation of S compared to the unfolded APTA can be obtained by reducing the size of the initial APTA. At each location, a reduction of the number of transitions is performed by grouping all the contiguous time values for the same symbol into a single transition where the time interval $\boldsymbol{I}$ is the union of the different time intervals.

Definition 4 (Constructive-bound APTA). *A constructive-bound APTA is a DRTA$^+$ where:*

- $L = \{(\omega, \{I_i\}_{i=1}^{|\omega|}), \omega \in \Sigma^*, I_i \in \mathcal{I} \mid \forall i \in [1; |\omega|], \forall c \in I_i, \exists (\omega', \tau') \in S, \forall j < i, \tau'_j \in I_j, c = \tau'_i, \omega \leq \omega'\} \cup \{l_0 = (\epsilon, \emptyset)\}$,
- T *contains transitions of the form* $t = \langle (\omega, \{I_i\}_{i=1}^{|\omega|}), \sigma, \boldsymbol{I}, (\omega', \{I'_i\}_{i=1}^{|\omega'|}) \rangle$ *such that: (1)* $\omega\sigma = \omega'$, *(2)* $\forall i \in [1; |\omega|], I_i = I'_i$ *and* $I'_{|\omega'|} = I'_{|\omega|+1} = \boldsymbol{I}$, *and* $\mathcal{N}(t) = |\{(\omega' u, \tau' v) \in S, \forall i \in [1; |\omega'|], \tau'_i \in I'_i, u \in \Sigma^*, v \in \mathbb{T}^*\}|$.

In Definition 4, each location corresponds to a subset of timed words that have a common untimed prefix where each symbol (of the prefix) apprears with contiguous time values. A location is labeled by the given untimed prefix ω and the sequence of intervals $\{I_i\}_{i=1}^{|\omega|}$ corresponding to each symbol of ω. I_i is the interval grouping the contiguous time values for the symbol σ_i of ω. All time values of these intervals are present in at least one timed word in S. A transition is added between locations l and l' such that: (1) the concatenation of the untimed prefix relative to l and symbol σ produces the untimed prefix relative to l', and (2) adding $\boldsymbol{I}$ to the interval sequence of l gives the interval sequence of l'.

5.3 Tightened-Bound APTA

The minimal size of APTA is obtained by allowing the minimal number of transitions from each location. This minimal number is obtained by assigning at most one transition for each symbol of Σ. The initial generalization is reduced

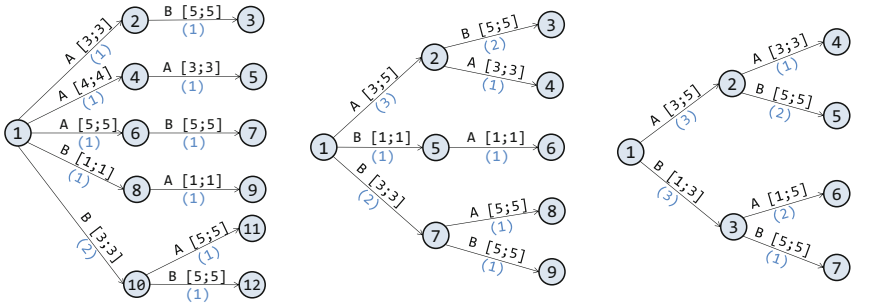
(compared to the generalized-bound APTA) by identifying the minimum (resp. the maximum) time value t_{min} (resp. t_{max}) among all the time values for each location l and symbol σ. Then, a single transition is created from l with symbol σ and a time interval $[t_{min}; t_{max}]$. We call this APTA model a *tightened-bound* APTA. It has the same structure as the generalized-bound APTA but with tighter bounds. The time interval $[t_{min}; t_{max}]$ of each transition is computed locally depending on the corresponding timed words in S.

Definition 5 (Tightened-bound APTA). *A tightened-bound APTA is a* $DRTA^+$ *where:*

- $L = \{\omega \in \Sigma^* \mid \exists(\omega', \tau') \in S, \omega \leq \omega'\}$, $l_0 = \epsilon$,
- *T contains transitions of the form* $t = \langle \omega, \sigma, [t_{min}; t_{max}], \omega' \rangle$ *such that: (1)* $\omega\sigma = \omega'$, *(2)* $t_{min} = Min\{\tau_{|\omega'|} \mid (\omega' u, \tau) \in S\}$, *(3)* $t_{max} = Max\{\tau_{|\omega'|} \mid (\omega' u, \tau) \in S\}$, *and* $\mathcal{N}(t) = |\{(\omega' u, \tau) \in S\}|$, *where* $u \in \Sigma^*$.

5.4 Evaluation

In this section, we discuss the proposed APTA models with respect to their ability to faithfully represent S and to their size. We consider the following sample S = {(A,5)(B,5); (A,4)(A,3); (A,3)(B,5); (B,3)(A,5); (B,3)(B,5); (B,1)(A,1)}. Figure 3 depicts the three types of APTAs representing S.



(a) Unfolded (b) Constructive-bound (c) Tightened-bound

Fig. 3. The three APTA models for the sample S

Initial Generalization. The unfolded APTA does not introduce any initial generalization (Fig. 3a). The constructive-bound APTA is a more compact representation of S compared to the unfolded APTA with less generalization than the generalized-bound APTA. Some generalization is introduced due to the possible combination of grouped time values. In other words, the time values of the time intervals appear in S, but the language generated by the APTA overapproximates S since it accepts more time sequences. For instance, in Fig. 3b, the timed word (A,3)(A,3) is accepted although not in S. This is due to the combination of time values coming from the timed words (A,4)(A,3) and (A,3)(B,5).

The tightened-bound APTA introduces two kinds of generalization. The first is due to the combination of grouped time values, as for the constructive-bound APTA. The second one is caused by the presence of empty intervals. An example is given in Fig. 3c where transitting from the root is possible using the timed symbol (B,2) which is not in S. This latter generalization is similar to the one we pointed out for the generalized-bound APTA albeit with more restrictive time intervals, since the empty intervals $[0; t_{min} - 1]$ and $[t_{max} + 1; max(\mathbb{T}^S)]$ are initially removed. The relationship between these models and the generalization they introduce is summarized in Fig. 4.

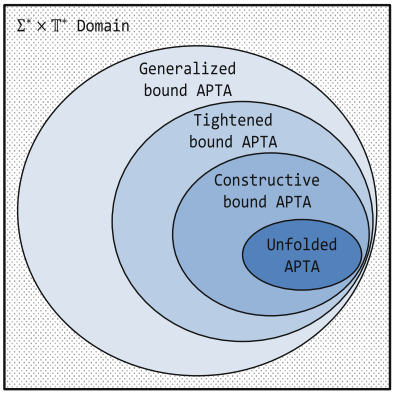


Fig. 4. Generalization introduced by the different *APTAs*

APTA Size. In terms of the size of initial representation, the unfolded APTA is the largest. The APTA size depends on the size of Σ and $\mathbb{T}^S$. The worst case is encountered when all the traces in S are of the same length N and when S contains all the combinations of symbols and time values. The resulting complete tree, in this case, represents the upper bound on the exponential number of locations and can be expressed as

$$MaxSize_{(unfolded)} = \frac{1 - (|\Sigma| \times |\mathbb{T}^S|)^{N+1}}{1 - (|\Sigma| \times |\mathbb{T}^S|)}$$

This maximum number of locations is reduced in the constructive-bound APTA by grouping contiguous time values. However, this improvement is meaningless in the case where all the time values are disjoint. For a given interval $[0; max(\mathbb{T}^S)]$, the maximum number of disjoint intervals is encountered when all the time values are disjoint and is equal to $(max(\mathbb{T}^S) + 1)\ div\ 2$. The worst case number of locations of a complete tree of depth $N + 1$ is

$$MaxSize_{(constructive)} = \frac{1 - (|\Sigma| \times ((max(\mathbb{T}^S) + 1)\ div\ 2))^{N+1}}{1 - (|\Sigma| \times ((max(\mathbb{T}^S) + 1)\ div\ 2))}$$

The number of locations, in this case, is highly dependent on the size of Σ and less on the size of $\mathbb{T}^S$. This latter can be removed by allowing only one interval for each symbol at each location. This is the case of the generalized-bound APTA and the tightened-bound APTA which return the minimum number of locations.

6 Experiments

In this section, we evaluate the learned model according to its ability to accept the words belonging to L and to reject the others. This gives insight into how

accurate the learned model is. A *C++* implementation of the proposed algorithms and the considered examples can be found in http://www-verimag.imag.fr/~nouri/drta-learning. The same page also contains additional materials such as algorithms and formal definitions of the elementary operations, in addition to a discussion about the proposed models' accuracy.

6.1 Evaluation Procedure

The accuracy of the learned model can be quantified using two metrics: the precision and the recall. The precision is calculated as the proportion of words that are correctly recognized (true positives) in the learned model H' over all the words recognized by H', while the recall represents the proportion of words that are correctly recognized in H' over all the words recognized by the initial model H. The precision and the recall can be combined in a single metric called F1 score. A high F1 score corresponds to a high precision and recall, and conversely.

$$Precision(H', H) = \frac{|\{(\omega, \tau) \in \Sigma^* \times \mathbb{T}^* \mid (\omega, \tau) \in \mathcal{L}(H) \wedge (\omega, \tau) \in \mathcal{L}(H')\}|}{|\mathcal{L}(H')|}$$

$$Recall(H', H) = \frac{|\{(\omega, \tau) \in \Sigma^* \times \mathbb{T}^* \mid (\omega, \tau) \in \mathcal{L}(H) \wedge (\omega, \tau) \in \mathcal{L}(H')\}|}{|\mathcal{L}(H)|}$$

$$F1\_score(H', H) = 2 \times \frac{prec(H', H) \times recall(H', H)}{prec(H', H) + recall(H', H)}$$

Based on these metrics, we distinguish four degrees of generalization for the learned models (see Fig. 5):

1. The maximum F1 score is obtained when the exact target language $L(H)$ is learned.
2. A precision of 1 and a recall strictly lower than 1 characterize an under-approximation, i.e., the learned model H' recognizes a subset of words of $L(H)$.
3. A recall of 1 and a precision strictly lower than 1 characterize an over-approximation, i.e., the learned model H' accepts all the words of $L(H)$ in addition to extra words not in $L(H)$.
4. A precision and a recall strictly lower than 1 characterize a cross-approximation, i.e., $L(H')$ contains only a subset of words in $L(H)$ plus additional words not in $L(H)$.

Our experimental setup shown in Fig. 6, consists of three modules responsible for trace generation, model learning and model evaluation. Since we are trying to evaluate how accurate the learning algorithm is, the initial model H, designed as a $DRTA^+$, is only known by the trace generator and the model evaluator, while the model learner has to guess it. The trace generator produces a timed learning sample S and a test sample. The latter contains timed traces that do not appear in S. This sample is used to evaluate the learned model with respect to new traces that were not used during the learning phase.

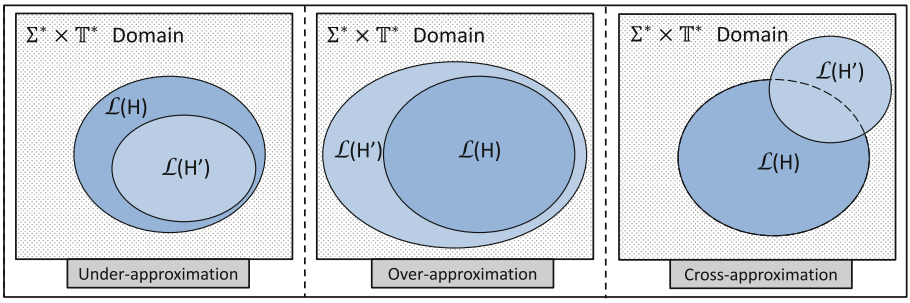


Fig. 5. Degrees of generalization of the learned language $L(H')$ with respect to the target language $L(H)$

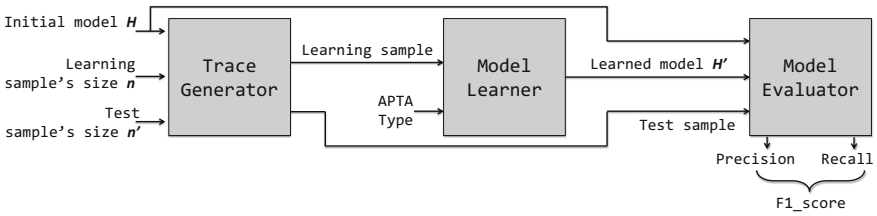


Fig. 6. Experimental setup to validate the improved learning procedure

6.2 Benchmarks

We run our experimental setup on three examples, namely, *Periodic A*, *Periodic A-B* and *CSMA/CD communication medium model*.

Periodic A is a synthetic periodic task A that executes for 1 to 3 time units in a period of 5 time units. The goal of this benchmark is to check if the algorithm is able to learn the periodicity and the duration of a single task. Two less constrained variants of this model are also considered. In both of them, we remove the periodicity of the task by setting a predefined waiting time of 5 time units after the task A finishes. In the first variant, called aperiodic contiguous-time (**ap_cont A**), the execution time of the task A can take contiguous time values in $[1; 3]$. In the second one, called aperiodic disjoint (**ap_dis A**), the execution time takes the disjoint time values 0, 2 and 4. Our goal is to check if the algorithm is able to detect the unused time values 1 and 3.

Periodic A-B consists of two sequential tasks A and B, taking execution time values, respectively, in intervals [1;3] and [1;2] with a periodicity of 5 time units. In this example, the learning algorithm is faced with dependencies between clock constraints for the task A, the task B and their periodicities, which is a more complex setting.

CSMA/CD communication Medium Model is a media access control protocol for single-channel networks that solves data collision problems. We focus on the **CSMA/CD** communication medium model for a 2-station network presented in [5]. Figure 7 represents the underlying **CSMA/CD** communication medium model where λ represents the propagation time. We assume that t_{max} is the maximum time elapsing between two consecutive events.

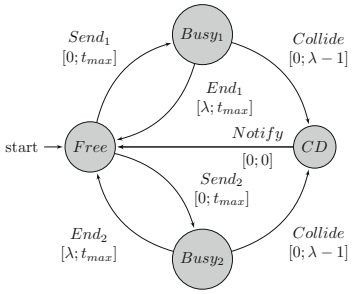


Fig. 7. CSMA/CD communication medium model for a 2-station network

6.3 Results

Experiments have been done on the described examples using a learning sample of size 200 and a test sample of size 1000.

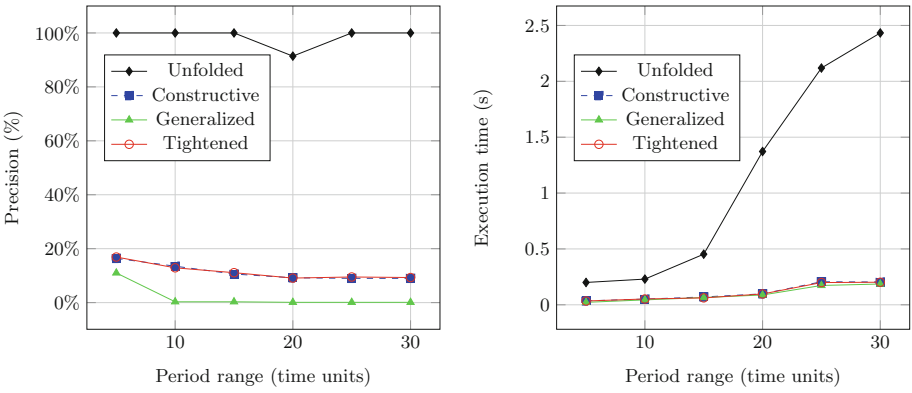
The Synthetic Examples. Table 1 summerizes the results for **periodic A** (and variants) and **periodic A-B**. Since all the learned models have a 100% recall, only the precision is discussed in the sequel. The obtained results show that the original RTI+ learns an over-approximating model with a poor precision for all the considered examples. In contrast, as shown by the F1 score, the exact model is learned using the unfolded APTA for **periodic A** and its variants. Both the constructive and the tightened-bound APTAs do not learn the exact **periodic A** model (although more accurate than RTI+). They actually fail to identify the periodicity of task A. For **ap_cont A**, the constructive and the tightened-bound APTAs learn the exact model. However, for **ap_dis A**, the constructive-bound approach learns the exact model, while the tightened-bound one returns a model with a low precision since it does not detect the unused time values 1 and 3.

For the **periodic A-B** example, none of the variants was able to learn an accurate model: the obtained precision is at most 2.27% (using the constructive-bound APTA). They all fail to capture dependencies over clock constraints. Nevertheless, the precision is still better than the original RTI+ (0.18%).

Figure 8 shows the impact of bigger time periods in the **periodic A** example on the quality of the learned model (Fig. 8a) and the learning time (Fig. 8b). We observe that increasing the period makes it more difficult to learn accurate models; Increasing the period decreases the precision as shown in Fig. 8a and increases the learning time as shown in Fig. 8b. For instance, the original RTI+ with generalized-bound APTA is quite fast but its precision tends to zero. Using constructive and tightened-bound APTAs improves the precision with a similar learning time. Finally, relying on the unfolded APTA produces very precise models but induces an important learning time when the period exceeds 15 time units.

Table 1. Accuracy results for the synthetic benchmarks with the four APTAs

Benchmark		Periodic A	Ap_dis A	Ap_cont A	Periodic AB
Generalized-bound	Precision	11%	0.8%	0.6%	0.18%
	Recall	100%	100%	100%	100%
	F1 score	0.1982	0.0159	0.0119	0.0036
Unfolded	Precision	100%	100%	100%	1.97%
	Recall	100%	100%	100%	100%
	F1 score	1.0000	1.0000	1.0000	0.0386
Constructive-bound	Precision	16.4%	100%	100%	2.27%
	Recall	100%	100%	100%	100%
	F1 score	0.2818	1.0000	1.0000	0.0444
Tightened-bound	Precision	16.9%	3.01%	100%	2.18%
	Recall	100%	100%	100%	100%
	F1 score	0.2891	0.0584	1.0000	0.0427



(a) Precision of the learned model (b) Learning time

Fig. 8. Impact of varying the task A period on the precision/the learning time

The CSMA/CD Example. Table 2 summerizes the experiments done on **CSMA/CD**. On the one hand, one can notice that RTI+, like in the previous cases, learns an over-approximating model with a poor precision (6.20%) but in a short time (∼6 s). Moreover, the generalized-bound APTA, initially having 2373 locations, is reduced to a final model with only 4 locations which represents a high reduction. On the other hand, the proposed APTAs produce significantly different models that cross-approximate the original **CSMA/CD**. For instance, the tightened-bound APTA learns a very precise model (93.70%). However, the model is obtained in more than 8 hours and has 370 locations. Using the constructive-bound APTA gives a model with less precision (85.80%) in a lower execution time (∼3 h). Finally, the unfolded APTA gives a model with

Table 2. Experimental results for **CSMA/CD** using the four APTA models

APTA type	Precision	Recall	F1 score	Time	APTA size	DRTA size	Reduction
Generalized	6.20%	100.00%	0.1168	∼6 s	2373	4	99.83%
Unfolded	49.40%	96.70%	0.6539	∼9 min	3586	19	99.47%
Constructive	85.80%	52.00%	0.6475	∼3 h	2652	207	92.19%
Tightened	93.70%	49.90%	0.6512	∼8 h	2373	370	84.41%

a 49.40% precision and a 96.70% recall, which corresponds to the best F1 score (0.6539). Furthermore, compared to constructive and tightened-bound APTAs, the learning time for the unfolded APTA is lower (∼9 min). Hence, we conclude that, for this example, the unfolded APTA provides a good tradeoff between accuracy and learning time.

7 Conclusion

In this work, we proposed different variants of the RTI+ algorithm for learning models with both stochastic and timed behaviors. We formally defined three APTA models with different levels of generalization and representation sizes. We validated our proposal by performing different experiments that showed that using the new APTA variants provides more accurate models regarding the time behaviors. However, we observed that a higher learning time is generally required, depending on the desired accuracy. In the future, we are planning to improve our algorithms to better handle models with dependencies over clock constraints such as in the **periodic A-B** example. We are investigating a new compatibility criterion that takes into account such dependencies and that is able to isolate empty time intervals.

References

1. Alur, R., Dill, D.L.: A theory of timed automata. Theoret. Comput. Sci. **126**(2), 183–235 (1994)
2. Angluin, D.: Learning regular sets from queries and counterexamples. Inf. Comput. **75**(2), 87–106 (1987)
3. Carrasco, R.C., Oncina, J.: Learning stochastic regular grammars by means of a state merging method. In: Carrasco, R.C., Oncina, J. (eds.) ICGI 1994. LNCS, vol. 862, pp. 139–152. Springer, Heidelberg (1994). doi:10.1007/3-540-58473-0_144
4. De la Higuera, C.: Grammatical Inference: Learning Automata and Grammars. Cambridge University Press, Cambridge (2010)
5. Kwiatkowska, M., Norman, G., Sproston, J., Wang, F.: Symbolic model checking for probabilistic timed automata. Inf. Comput. **205**(7), 1027–1077 (2007)
6. Legay, A., Delahaye, B., Bensalem, S.: Statistical model checking: an overview. In: Barringer, H., Falcone, Y., Finkbeiner, B., Havelund, K., Lee, I., Pace, G., Roşu, G., Sokolsky, O., Tillmann, N. (eds.) RV 2010. LNCS, vol. 6418, pp. 122–135. Springer, Heidelberg (2010). doi:10.1007/978-3-642-16612-9_11

7. Maier, A., Vodencarevic, A., Niggemann, O., Just, R., Jaeger, M.: Anomaly detection in production plants using timed automata. In: 8th International Conference on Informatics in Control, Automation and Robotics. pp. 363–369 (2011)
8. Mao, H., Chen, Y., Jaeger, M., Nielsen, T.D., Larsen, K.G., Nielsen, B.: Learning probabilistic automata for model checking. In: 2011 Eighth International Conference on Quantitative Evaluation of Systems (QEST), pp. 111–120. IEEE (2011)
9. Mao, H., Chen, Y., Jaeger, M., Nielsen, T.D., Larsen, K.G., Nielsen, B.: Learning deterministic probabilistic automata from a model checking perspective. Mach. Learn. **105**(2), 255–299 (2016)
10. de Matos Pedro, A., Crocker, P.A., de Sousa, S.M.: Learning stochastic timed automata from sample executions. In: Margaria, T., Steffen, B. (eds.) ISoLA 2012. LNCS, vol. 7609, pp. 508–523. Springer, Heidelberg (2012). doi:10.1007/978-3-642-34026-0_38
11. Nouri, A., Bozga, M., Molnos, A., Legay, A., Bensalem, S.: ASTROLABE: a rigorous approach for system-level performance modeling and analysis. ACM Trans. Embed. Comput. Syst. **15**(2), 31:1–31:26 (2016)
12. Sen, K., Viswanathan, M., Agha, G.: Learning continuous time markov chains from sample executions. In: Proceedings of the First International Conference on The Quantitative Evaluation of Systems, pp. 146–155. QEST 2004, IEEE Computer Society, Washington, DC (2004)
13. Verwer, S.E., Eyraud, R., De La Higuera, C.: PAutomaC: a probabilistic automata and hidden markov models learning competition. Mach. Learn. **96**(1–2), 129–154 (2014)
14. Verwer, S.E.: Efficient identification of timed automata: theory and practice. Ph.D. thesis, TU Delft, Delft University of Technology (2010)