Extended Timed Regular Expressions

Marco Muñiz, Marius Mikučionis, and Kim G. Larsen

Aalborg University, Denmark
{muniz,marius,kgl}@cs.aau.dk

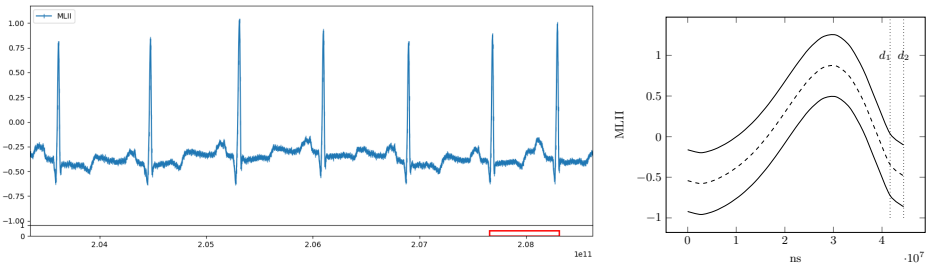
The behavior of complex organisms or systems is often stored as time series data. Time series data is valuable because it contains valuable information in the form of timed patterns. However, this patterns are difficult to formalize and to detect. We present Extended Timed Regular Expressions (ETRE) to express complex timed patterns which can be systematically and efficiently matched in large sets of time series data. We translate ETRE to Timed Automata (TA), where pattern matching is computed by reachability analysis in TA. We implement our theory using C++ in the new tool TimeRex. Our tool can be used for online (run time) or offline (post processing) pattern matching. We run extensive experiments on real data. We have been able to efficiently match a number of relevant patterns.

1 Introduction

Time series data is present everywhere, it can be produced by a Cyber Physical System (CPS), by sensors monitoring the human body, communication protocols, etc. As an example consider Figure 1a shows an example of time series data from an ECG from the PhysionNet MIT-BIH Arrhythmia Database [10]. Time series data is valuable because it contains valuable information in the form of timed patterns. These patterns can be used to classify the data, to do predictions, and when possible to control the behavior of a system. As example consider Figure 1a where it is of interest to find early occurrence of a QRS complex (peak) a sign of arrhythmia.

Regular Expressions (RE) is a fundamental formalism in Computer Science developed in formal language theory by Kleene [11] and formalizes the concept of a regular language. An equivalent – similarly fundamental – formalism is that of Finite State Automata. By now RE are commonly used to specify patterns in a text with supporting search and replace engines of word processors and text editors and with support in many programming languages. Timed patterns can be expressed in a number of formalisms including Timed Regular Expressions [3], Signal Temporal Logic [14,15], Linear Temporal Logic [18] , and Timed Automata [1] (TA).

This work introduces Extended Timed Regular Expressions (ETRE) which generalizes Timed Regular Expressions in two principles ways: events are value constraints (e.g. intervals) on one or more of the continuous signals present in the time-series, and sub-expressions may be subject to timing-constraints. Inspired on the equivalence notion between TRE and TA [3] we translate ETRE



(a) $\langle f^{\text{MLII}}_{0.38}\rangle_{[0.039s,0.045s]} \cdot \langle \Sigma^*\rangle_{[0.47s,0.67s]} \cdot \langle f^{\text{MLII}}_{0.38}\rangle_{[0.039s,0.045s]}$ (b) $\langle f^{\text{MLII}}_{0.38}\rangle_{[0.039s,0.045s]}$

Fig. 1: Pattern Matching ETRE to detect Arrhythmia. Red rectangle is a match.

to Extended TA (XTA) where ETRE pattern matching can be decided as reachability in XTA. Our translation carefully exploits the notion of urgency to allow for a discrete-time engine. We have implemented our theory using C++ in the new tool TimeRex. Our tool supports both online (run time) and offline pattern matching. Later in this paper we present an extensive experimental evaluation with encouraging results.

As an example consider Figure 1a which shows an ECG from the MIT-BIH arrhythmia database [10]. A QRS complex is a sophisticated temporal pattern with several points of inflection. Therefore, a high degree polynomial could be an initial approximation. Figure 1b describes a tube generated by the ETRE $\langle f^{\text{MLII}}_{\delta=0.38}\rangle_{[d_1=0.039s,d_2=0.045s]}$ where f is a seventh degree polynomial. A seventh degree polynomial is an approximation based on our experiments. It will correctly match on a one to one relation all QRS complex in the given data. Finally, the more complex ETRE $\langle f^{\text{MLII}}_{\delta=0.38}\rangle_{[0.039s,0.045s]} \cdot \langle \Sigma^*\rangle_{[0.04s,0.65s]} \cdot \langle f^{\text{MLII}}_{\delta=0.38}\rangle_{[0.039s,0.045s]}$ approximates a QRS complex followed by any value (Σ^*) for a period of in less than $0.65s$ and then followed by another QRS complex (a sign of arrhythmia). The red rectangle in Figure 1a shows a match of this ETRE in the signal from our tool TimeRex. This ETRE can be used to systematically detect all premature heart beats on the given signal.

Finally, the notions of ETRE and TA provide an unambiguous (explainable) formalism for the domain expert to specify important patterns (for matching) as well evaluate possibly learned patterns. Learning formulae is an active research area e.g. LTL learning [20], RE learning [13], parametric timed pattern matching [2,16].

Related Work ETRE are based on the seminal work for Timed Regular Expressions TRE [3]. ETRE syntax allows for arbitrary functions with a tolerance for noise. ETRE extend the alphabet by allowing tuples of real valued events. ETRE semantics align with those from TRE [3] (time event sequence semantics). In the following we compare our approach with the state of the art.

Timed Pattern Matching. The work in [19] introduces Timed Pattern Matching for TRE. The semantics of TRE in [19] are over boolean signals (signal semantics) and it is believed that the semantics do not coincide with the time event sequence semantics from [3]. The authors introduce the notion of match set. A match set contains the start and stop points of uncountably many matches. The authors show that for finite variability signals a match set can be computed and represented by a finite union of zones. The algorithms are based on operations on zones. The main differences with our approach are that ETRE semantics are the so called time event sequence semantics which align with the semantics from [3]. In addition, as input (corpus) we require finite timed words (discrete time series) with no finite variability constraints. As a result there is only finitely many sub-words which match the given ETRE. We do not compute a match set. We only compute the start and stop indexes of the first sub-word which matches. Further, our assumptions and construction allow for a discrete time engine implementation avoiding complex operations on zones.

Existing formalisms. Signal Temporal Logic (STL) [15,14] allows for predicates over real values. The semantics are over continuous time signals. ETRE are interpreted over time series data and arbitrary functions are supported by the syntax. Regular Expressions are widely used in industry. In the industrial context, we expect a smoother transition from RE to ETRE than from RE to STL. *Signal Regular Expressions (SRE).* The work in [5] introduces SRE together with qualitative and quantitative semantics. SRE allows to compare variables on signals against real numbers. The main differences with our approach are that SRE are interpreted in the signal semantics where as ETRE are interpreted using time event sequence semantics. Further the semantics for the Kleene star operator are different. In addition, ETRE support the use of functions. *Shape Expressions* [17,8] allows RE over parameterized signal shapes. Shape expressions translate to Shape Automata which act as recognisers. While Shape Expressions and ETRE aim at describing complex patterns, the underlying assumptions and techniques are different. A simple inspection could suggest that both formalisms complement. However, a formal comparison is be needed.

Online Monitoring Using Automata In [6] and [22,23] patterns are specified as Timed Automata. TA is then used for timed pattern matching on continuous signals. The underlying algorithms perform symbolic reachability using zones. On the contrary, our translation from ETRE yields a subset of TA for which a discrete engine implementation is possible. Further our construction enables efficient application of Partial Order Reduction.

Parametric Timed Pattern Matching The works in [2] and Parametric TRE [16] study parametric pattern matching. Here specifications allow parameters to cope with uncertainty. An extended match set is computed using reachability synthesis in parametric Timed Automata. In our case studies e.g. ECG for arrhythmia detection, identifying the right durations is challenging. We believe that this approach could complement nicely with ETRE for identifying adequate durations.

Skipping Techniques The work in [22] applies Boyer-Moore pattern matching method. In contrast to our work, the procedure in [22] requires pre-computing

the region graph. The same authors improve this result in [23], by using a more efficient skipping method and by replacing the region automaton by the zone graph. Our results ensure that we only need to construct a discrete transition system. Therefore, it should be possible to profit from such advanced techniques. As future work, we aim at including these techniques in our framework.

Quantitative Timed Pattern Matching The work in [5] introduces robust semantics for pattern matching of SRE. The semantics shows how robustly a signal satisfies (or violates) the given specification. The work in [21] presents and online method for computing robust semantics where the specification is given as a Timed Automata. Robust semantics for ETRE is interesting and relevant for many industrial applications.

2 Preliminaries

We apply our method to the theory of timed automata [1]. Our formal model is *extended timed automata* and it is an abstract representation of modeling formalism used in the tool UPPAAL [7]. *Clocks and Variables.* Let X be a set of *clocks*. A *clock valuation* is a function $\mu : X \to \mathbb{R}_{\geq 0}$. We use $\mathcal{V}(X)$ to denote the sets of all valuations for clocks in X. We use μ_{ini} to denote the valuation where all clocks in X are assign the value 0. Let V be a set of *variables*. A *variable valuation* is a function $\nu : V \to \mathbb{R}$ that maps variables to real numbers. We use $\mathcal{V}(V)$ to denote the set of all variable valuations. We use ν_{ini} to denote the valuation where all variables are assign the value 0. *Constraints.* The set $B(X)$ is the set of *clock constraints* generated by the grammar $\phi ::= x \bowtie c \mid \phi_1 \wedge \phi_2$, where $x \in X$, $\bowtie \in \{<, \leq, \geq, >\}$, and $c \in \mathbb{Q}$. The set $B(V)$ is a set of *Boolean variable constraints* over V. The set $B(X, V)$ of constraints comprises $B(X)$, $B(V)$, and conjunctions over clock and variable constraints. *Updates.* A *clock update* is of the form $x := 0$. A *variable update* is of the form $v := c$ where $c \in \mathbb{R}$. The set $U(X, V)$ of *updates* contains all finite, possibly empty sequences of clock and variable updates. We let $[\![r^\nu]\!] : \mathcal{V}(X) \cup \mathcal{V}(V) \to \mathcal{V}(X) \cup \mathcal{V}(V)$ be a map from valuations to valuations. We use $\mu[r]$ to denote the updated clock valuation $[\![r]\!](\mu)$. Analogously, for variable valuations. *Channels.* Given a set C of *channels*, the set $H(C)$ of synchronizations over channels is $\{h!, h?, \tau\}$ where $h \in C$, and τ represents an internal action.

Definition 1 (Extended Timed Automata XTA). *An* extended timed automaton $\mathcal{A}$ *is a tuple* $(L, L^u, L^f, l^{\mathsf{ini}}, X, V, H(C), E, I)$ *where:* L *is a set of locations,* $L^u \subseteq L$ *denotes the set of urgent locations,* $L^f \subseteq L$ *denotes the set of accepting locations,* $l^{\mathsf{ini}} \in L$ *is the initial location,* X *is a set of clocks,* V *is a set of variables,* $H(C)$ *is a set of channel synchronizations for set of channels* C*,* $E \subseteq L \times H(C) \times B(X) \times B(V) \times U(X, V) \times L$ *is a set of edges between locations with a channel expressions, a clock guard, a variable guard, an update set, and* $I : L \to B(X)$ *assigns clock invariants to locations.*

Definition 2 (Network of XTA). *A network* $\mathcal{N}$ *of XTA consists of a finite sequence* $\mathcal{A}_1, \ldots, \mathcal{A}_n$ *of XTA, where* $\mathcal{A}_i = (L_i, L_i^u, L_i^f, l_i^{\mathsf{ini}}, X_i, V_i, H(C)_i, E_i, I_i)$

for $1 \leq i \leq n$. *Locations are pairwise disjoint i.e.* $L_i \cap L_j = \emptyset$ *for* $1 \leq i, j \leq n$ *and* $i \neq j$. *The set of locations is* $L = \cup_{i=1}^{n} L_i$, *analogously for urgent* L^u *locations. The set of clocks is* $X = \bigcup_{i=1}^{n} X_i$ *and the set of variables is* $V = \cup_{i=1}^{n} V_i$. *The set of channel expressions is* $H(C) = \cup_{i=1}^{n} H(C)_i$. *The set of edges is* $E = \cup_{i=1}^{n} E_i$. *A location vector is a vector* $\vec{l} = (l_1, \ldots, l_n)$, *and* $\vec{l_0} = (l_1^0, \ldots, l_n^0)$ *is the initial location vector. The invariant function over location vectors is* $I(\vec{l}) = \bigwedge_i I_i(l_i)$.

We write $\vec{l}[l_i'/l_i]$ to denote the vector where the i-th element l_i of $\vec{l}$ is replaced by l_i'. We write $\vec{l}^i$ to denote the i-th element of $\vec{l}$. We write $\vec{l}_{\mathsf{ini}}$ to denote the vector where for all $1 \leq i \leq n$ we have $\vec{l}^i_{\mathsf{ini}} = l_i^{\mathsf{ini}}$.

Definition 3 (Semantics of a Network of XTA). *Let* $\mathcal{N} = \mathcal{A}_1, \ldots, \mathcal{A}_n$ *be a network of TA. Its semantics is defined as a* transition system $(S, s_0, \rightarrow)$, *where* $S \subseteq (L_1 \times \cdots \times L_n) \times \mathcal{V}(X) \times \mathcal{V}(V)$ *is the set of states comprising a location vector, a zone, and a variable valuation,* $s_{\mathsf{ini}} = (\vec{l}_{\mathsf{ini}}, \mu_{\mathsf{ini}}, \nu_{\mathsf{ini}})$ *is the initial state, and* $\rightarrow \subseteq S \times (\mathbb{R}_{\geq 0} \cup 2^E) \times S$ *is the transition relation defined by:*

- *Delay transition,* $(\vec{l}, \mu, \nu) \xrightarrow{d} (\vec{l}, \mu + d, \nu)$ *iff* $\vec{l}^i \notin L_i^u$ *for* $1 \leq i \leq n$, $d \in \mathbb{R}_{\geq 0}$ *and* $\mu + d' \models I(\vec{l})$ *holds for all* $d' \in [0, d]$.
- *Internal transition,* $(\vec{l}, \mu, \nu) \xrightarrow{\{e_i\}} (\vec{l}[l_i'/l_i], \mu', \nu')$ *iff exits* $e_i = (l_i, \tau, \phi, \psi, r, l_i') \in E_i$ *with* $\mu' = \mu[r]$, $\mu' \models I(\vec{l}[l_i'/l_i])$, $\nu' = \nu[r]$, *and* $\nu \models \psi$.
- *Broadcast transition,* $(\vec{l}, \mu, \nu) \xrightarrow{E'} (\vec{l'}, \mu', \nu')$ *iff* $E' = \{e, e_1, e_2, \ldots, e_m\} \subseteq E$, $|E'| > 1$, *and* E' *is such that* $e = (l, h!, \phi, \psi, r, l')$ *is a sender and for* $1 \leq i \leq m$ $e_i = (l_i, h?, \phi_i, \psi_i, r_i, l_i')$ *is a receiver. Where*
  - *edges* $e, e_1, e_2, \ldots, e_m$ *are from different components,*
  - $e_1, e_2, \ldots, e_m$ *are ordered according the component ordering* $\mathcal{A}_1, \ldots, \mathcal{A}_n$,
  - $\vec{l'} = \vec{l}[l/l'][l_1'/l_1] \ldots [l_m'/l_m]$,
  - $\mu \models \phi$ *and for* $1 \leq i \leq m$ $\mu \models \phi_i$, $\mu' = \mu[r][r_1] \ldots [r_m]$, $\mu' \models I(\vec{l'})$,
  - $\nu \models \psi$ *and for* $1 \leq i \leq m$ $\nu \models \psi_i$, $\nu' = \nu[r][r_1] \ldots [r_m]$.

Additional Notation In the following, we are given a network of TA $\mathcal{N} = \mathcal{A}_1, \ldots, \mathcal{A}_n$ with locations L, clocks X, variables V, edges E, and induced symbolic transition system $(S, s_0, \rightarrow)$. Given state $s = (\vec{l}, \mu, \nu) \in S$ we use $\vec{l}(s) = \vec{l}$, $\mu(s) = \mu$, $\nu(s) = \nu$ to denote the location vector, clock valuation, and variable valuation of s. A finite *run* ρ of $\mathcal{N}$ is a finite sequence $\rho = (\vec{l_0}, \mu_0, \nu_0) \xrightarrow{\lambda_0} (\vec{l_1}, \mu_1\nu_1) \xrightarrow{\lambda_1} \ldots \xrightarrow{\lambda_{n-1}} (\vec{l_n}, \mu_n, \nu_n)$. An *accepting run* is a run starting from the initial configuration s_{ini} and terminating at state s such that for all $1 \leq i \leq n$, $\vec{l}(s)^i \in L_i^f$ i.e. all locations in $\vec{l}$ are accepting locations. Network $\mathcal{N}$ is accepting if there exists an accepting run.

A state $s \in S$ is *zero time* if it can not delay, denoted by $\mathsf{zt}(s)$ and defined by $\mathsf{zt}(s)$ iff $\forall s' \in S, \lambda \in \mathbb{R}_{\geq 0} \cup 2^E$. $s \xrightarrow{\lambda} s' \implies \lambda \in 2^E$. We write $s_0 \rightarrow^* s_n$ iff exists run $s_0 \xrightarrow{\lambda_0} s_1 \xrightarrow{\lambda_1} s_1 \ldots s_n$. We write $s_0 \xrightarrow[\mathsf{zt}]{}^* s_n$ iff exists run $s_0 \xrightarrow{\lambda_0} s_1 \xrightarrow{\lambda_1} s_1 \ldots s_n$ such that $\lambda_i \in 2^E$ and if $n > 0$ we have $\mathsf{zt}(s_i)$ for $0 \leq i < n$.

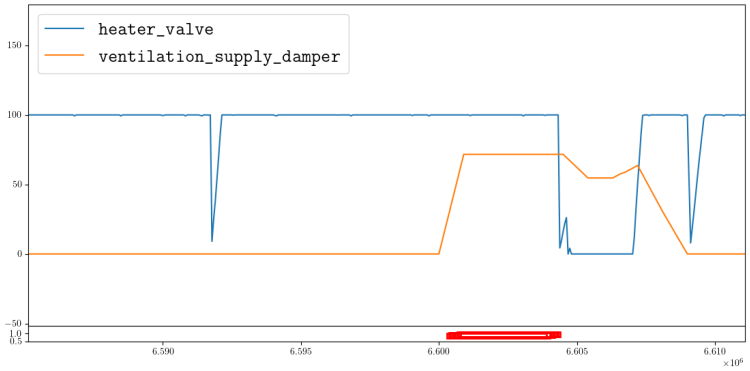


Fig. 2: Heating and ventilation in Thomas Manns vej 23. in Aalborg University. Red rectangles at the bottom indicates the matches of the ETRE $\langle(80^{\text{heat}}_{20})^*\rangle_{[3600s,7200s]} \cap \langle(60^{\text{vent}}_{40})^*\rangle_{[3600s,7200s]}$. This are instances where the heater and the ventilator are ON at the same time for a period between 1 to 2 hours.

Time-additivity. For any $d_1, d_2 \in \mathbb{R}_{\geq 0}$, $s, s' \in S$ it holds that $s \xrightarrow{d_1} \cdot \xrightarrow{d_2} s'$ iff $s \xrightarrow{d_1+d_2} s'$ Given any run we can use time-additivity to produce a run which consists of alternations of delays and discrete transitions. We denote such runs as *runs closed under time-additivity.*

3 Extended Timed Regular Expressions (ETRE)

Our intention is to extend the applicability of Timed Regular Expressions to the context of time series data and to industrial applications involving large magnitudes of data. First we use timed words to formalize time series data. Then we define ETRE by keeping in mind the nature of time series data and computational efficiency. We describe timed words and Extended Timed Regular Expressions in the spirit of [4].

3.1 Timed Words

A monoid is a triple $(M, \cdot, \epsilon)$ where M is a set, $\cdot$ is an associative binary operation on M and ϵ is the identity element of M satisfying $\epsilon \cdot m = m \cdot \epsilon = m$ for every $m \in M$. Time passage is described by the *time monoid* $(\mathbb{R}_{\geq 0}, +, 0)$ of positive real numbers under addition. Events are described by the *event monoid* $(\Sigma, +, \mathbf{0})$ of vectors $\Sigma \subseteq \mathbb{R}^n$ under vector addition. As described in [4] the *time-event monoid* $\mathcal{T} = (\mathbb{R}_{\geq 0} \uplus \Sigma, \cdot, \varepsilon)$ is obtained as the free product of the time and event monoids, where $\cdot$ is concatenation and ε is the empty word. A *timed word* $w = t_0 \cdot a_0 \cdot t_1 \cdot a_1 \cdot \ldots$, is an element of the time-event monoid with $t_i \in \mathbb{R}_{>0}$ and $a_i \in \Sigma$ for $i > 0$. Given timed word w its duration is given by $\mathsf{dur}(w) \in \mathbb{R}_{\geq 0}$.

Figure 1a shows a finite timed word on a single dimension i.e. $\Sigma = \mathbb{R}^1$. Figure 2 shows a finite timed word on two dimensions. Dimensions correspond to the valve position for the heating and the ventilation of a room in a building at Aalborg University. Our goal is to define and match complex patterns in time series data, for this we define Extended Timed Regular Expressions.

Definition 4 (Extended Timed Regular Expressions (ETRE)). *The set $\mathcal{E}$ of ETRE is given by the following grammar:*

$$\varphi ::= \phi \mid \varphi \cap \varphi$$
$$\phi ::= \varepsilon \mid c^i_\delta \mid \Sigma \mid \langle f^i_\delta \rangle_J \mid \phi \cup \phi \mid \phi \cdot \phi \mid \phi^+ \mid \phi^* \mid \langle \phi \rangle_J$$

where $c \in \mathbb{R}$, $0 \leq i < n$, $\delta \in \mathbb{R}_{\geq 0}$, J is an integer-bounded interval, and f is a real-valued computable function.

Definition 4 is similar to that given in [4]. The main differences include, events are vectors in $\mathbb{R}^n$, we include functions, and intersections are only allowed at the top level. Restriction on intersections is to avoid the expensive automata product construction (allowing nesting of intersections would not affect our theoretical results). Instead, intersections will be translated to networks of XTA which will be explored on the fly.

Definition 5 (ETRE semantics). *The semantics of a regular expression φ is given by a set of elements of the time-event monoid. Formally, $[\![]\!] : \mathcal{E} \to 2^{\mathcal{T}}$.*

$$\begin{aligned}
[\![\varepsilon]\!] &= \{\varepsilon\} \\
[\![c^i_\delta]\!] &= \{r \cdot (a^0, \ldots, a^n) \mid r \in \mathbb{R}_{\geq 0}, a^i \in [c-\delta, c+\delta], a^k \in \mathbb{R} \text{ for } k \neq i\} \\
[\![\Sigma]\!] &= \{r \cdot (a^0, \ldots, a^n) \mid r \in \mathbb{R}_{\geq 0}, a^i \in \mathbb{R}\} \\
[\![\langle f^i_\delta \rangle_{[d_1,d_2]}]\!] &= \{w \mid w = t_0 \cdot a_0 \cdot \ldots \cdot t_m \cdot a_m \text{ with } \mathsf{dur}(w_{0:m-1}) < d_1, \\
&\qquad \mathsf{dur}(w) \in [d_1, d_2], \text{and for } 0 \leq j \leq m \text{ and } k \neq i \text{ we have} \\
&\qquad 0 < t_j, a^k_j \in \mathbb{R}, a^i_j \in [f(\mathsf{dur}(w_{0:j})) - \delta, f(\mathsf{dur}(w_{0:j})) + \delta]\} \\
[\![\phi_1 \cdot \phi_2]\!] &= [\![\phi_1]\!] \cdot [\![\phi_2]\!] \\
[\![\phi_1 \cup \phi_2]\!] &= [\![\phi_1]\!] \cup [\![\phi_2]\!] \\
[\![\langle \phi \rangle_J]\!] &= [\![\phi]\!] \cap \{w \mid \mathsf{dur}(w) \in J\} \\
[\![\phi^*]\!] &= \bigcup_{i=0}^{\infty} ([\![\underbrace{\phi \cdot \ldots \cdot \phi}_{i \text{ times}}]\!]) \\
[\![\phi^+]\!] &= \phi \cdot \phi^* \\
[\![\varphi_1 \cap \varphi_2]\!] &= [\![\varphi_1]\!] \cap [\![\varphi_2]\!]
\end{aligned}$$

As an example consider the ETRE $\langle f^{\text{MLII}}_{\delta=0.38} \rangle_{[d_1=0.039s, d_2=0.045s]}$ we use MLII instead of 1 for readability. The function f is a 7^{th} degree polynomial as illustrated by the dashed line in Figure 1b. The semantics of this expression is the set of timed words inside the tube induced by f and δ with duration in the interval $[d_1, d_2]$. The semantics of the more complex expression $\langle f^{\text{MLII}}_{0.38} \rangle_{[0.039s,0.045s]} \cdot \langle \Sigma^* \rangle_{[0.47s,0.67s]} \cdot \langle f^{\text{MLII}}_{0.38} \rangle_{[0.039s,0.045s]}$ is the set of timed words in $\langle f^{\text{MLII}}_{0.38} \rangle_{[0.039s,0.045s]}$ concatenated with any word (Σ^*) with duration in $[0.47s, 0.67s]$ followed by timed words in $\langle f^{\text{MLII}}_{0.38} \rangle_{[0.039s,0.045s]}$. This set of timed words hint to a QRS complex followed by another QRS complex too early, a sign of arrhythmia.

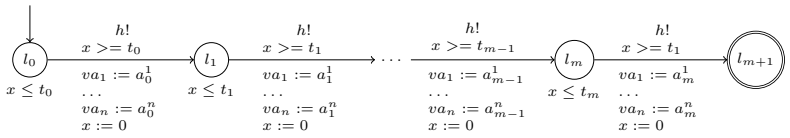


Fig. 3: XTA $\mathcal{A}_w$ for timed word $w = t_0 \cdot a_0 \cdot t_1 \cdot a_1 \cdot t_2 \cdot a_2 \cdot \ldots t_m \cdot a_m$

As another example consider ETRE $\langle(80^{\text{heat}}_{20})^*\rangle_{[1h,2h]} \cap \langle(60^{\text{vent}}_{40})^*\rangle_{[1h,2h]}$ from Figure 2. The semantics of the ETRE $\langle(80^{\text{heat}}_{20})^*\rangle_{[1h,2h]}$ is the set of all timed words with value 80 ± 20 in the dimension heat with duration between 1 and 2 hours. This ETRE indicate that the heater has been ON for the given duration. An intersection with the set $\langle(60^{\text{vent}}_{40})^*\rangle_{[1h,2h]}$ indicates that both heater and ventilation have been ON for the duration. Note that computing matches for every conjunt independently, will require to merge the results while satisfying the time constraints. Table 1 shows a number of ETRE used in our case studies.

4 Timed Word Membership for ETRE

The work in [4] shows the language equivalence between Timed Regular Expressions and Timed Automata. The expressiveness of ETRE goes beyond that of Timed Automata. However, we are interested in matching patterns in finite time series data (finite timed words) i.e. given a finite timed word w with non-zero delays and ETRE φ we need to decide if $w \in [\![\varphi]\!]$. We show that we can decide this problem by translating w and φ to a network of XTA, and then checking if the resulting network is accepting.

Definition 6 (XTA for a word). *Given non-empty finite word $w = t_0 \cdot a_0 \cdot t_1 \cdot a_1 \cdot \ldots t_m \cdot a_m$ with $t_i > 0$ the corresponding XTA is given by $\mathcal{A}_w = (L, \emptyset, L^f, l_0, \{\hat{x}, x\}, \{va_1, \ldots, va_n\}, H(C), E, I)$ where: $L = \{l_0, \ldots, l_{m+1}\}$, $C = \{h\}$, $E = \{(l_j, h!, x \geq t_j, \mathsf{true}, [va_0 := a^0_j, \ldots va_n := a^n_j, x := 0], l_{j+1}) \mid 0 \leq j \leq m\}$, $I(l_j) = x \leq t_j$ for $0 \leq j \leq m$ and $I(l_{m+1}) = \mathsf{true}$.*

Figure 3 describes $\mathcal{A}_w$. The automaton for the empty word ε is the automaton consisting of a unique location which is initial and accepting. Given ETRE φ the next step is to construct a network of XTA $\mathcal{N}_\varphi$ which accepts and produces finite words in φ. Note that $\mathcal{A}_w$ will generate inputs on which $\mathcal{N}_\varphi$ will synchronize. We carefully design $\mathcal{N}_\varphi$ to be input enabled by using urgent locations.

Definition 7 (Network of XTA for an ETRE). *Given ETRE φ the corresponding network $\mathcal{N}_\varphi = \mathcal{A}_1, \ldots, \mathcal{A}_n$ with channel $C = \{h\}$ is defined inductively as follows. The network for $\varphi \equiv \phi$ consist of a single automaton given by:*

- *$\phi \equiv \varepsilon$ then $\mathcal{A}_\varepsilon = (\{l_0, l_1\}, \{l_0\}, \{l_1\}, l_0, \emptyset, \emptyset, H(C), E, I)$ where $E = \{(l_0, \tau, \mathsf{true}, \mathsf{true}, [], l_1)\}$ and $I(l) = \mathsf{true}$ for $l \in \{l_0, l_1\}$.*

- $\phi \equiv c^i_\delta$ *then* $\mathcal{A}_\phi = (L, \{l_1\}, \{l_2\}, l_0, \emptyset, \{va_i\}, H(C), E, I)$ *where* $L = \{l_0, l_1, l_2\}$, $E = \{(l_0, h?, \mathsf{true}, \mathsf{true}, l_1), (l_1, \tau, \mathsf{true}, c - \delta \leq va_i \leq c + \delta, [], l_2)\}$ *and* $I(l) = \mathsf{true}$ *for* $l \in \{l_0, l_1, l_2\}$.
- $\phi \equiv \Sigma$ *then* $\mathcal{A}_\phi = (L, \{l_1\}, \{l_2\}, l_0, \emptyset, \emptyset, H(C), E, I)$ *where* $L = \{l_0, l_1, l_2\}$, $E = \{(l_0, h?, \mathsf{true}, \mathsf{true}, l_1), (l_1, \tau, \mathsf{true}, \mathsf{true}, [], l_2)\}$ *and* $I(l) = \mathsf{true}$ *for* $l \in \{l_0, l_1, l_2\}$.
- $\phi \equiv \langle f^i_\delta \rangle_{[d_1,d_2]}$ *then* $\mathcal{A}_\phi = (L, \{l_1\}, \{l_2\}, l_0, \{x\}, \{va_i, \delta, f, d_1, d_2\}, H(C), E, I)$, $L = \{l_0, l_1, l_2\}$, $I(l) = \mathsf{true}$ *for* $l \in L$, *and* $E = \{ (l_0, h?, \mathsf{true}, \mathsf{true}, [], l_1), (l_1, \tau, x < d_1, f(x) - \delta \leq va_i \leq f(x) + \delta, [], l_0), (l_1, \tau, d_1 \leq x \leq d_2, f(x) - \delta \leq va_i \leq f(x) + \delta, [], l_2)\}$.
- $\phi \equiv \phi_1 \cdot \phi_2$ *then* $\mathcal{A}_\phi = (L, L^u, L^f_2, l^{\mathsf{ini}}_1, X_1 \cup X_2, V_1 \cup V_2, H(C), E' \cup E_2, I_1 \cup I_2)$ *where* E' *is obtained from* E_1 *by replacing every edge of the form* $(l, \alpha, \phi, \psi, r, l')$ *with* $l' \in L^f_1$ *by an edge* $(l, \alpha, \phi, \psi, r', l^{\mathsf{ini}}_2)$ *where* r' *appends to* r *reset* $x := 0$ *for every clock* $x \in X_2$, $L = (L_1 \setminus L^f_1) \cup L_2$, *and* $L^u = (L^u_1 \setminus L^f_1) \cup L^u_2$.
- $\phi \equiv \phi_1 \cup \phi_2$ *then* $\mathcal{A}_\phi = (L_1 \cup L_2 \cup \{l\}, L^u_1 \cup L^u_2 \cup \{l\}, L^f_1 \cup L^f_2, l, X_1 \cup X_2, V_1 \cup V_2, H(C), E' \cup E_1 \cup E_2, I)$ *where* $E' = \{(l, \tau, \mathsf{true}, \mathsf{true}, [], l^{\mathsf{ini}}_i) \mid 1 \leq i \leq 2\}$ *and* $I = I_1 \cup I_2 \cup \{(l, \mathsf{true})\}$.
- $\phi \equiv \phi^+_1$ *then* $\mathcal{A}_\phi = (L_1, L^u_1, L^f_1, l^{\mathsf{ini}}_1, X_1, V_1, H(C), E_1 \cup E', I_1)$ *where* E' *is obtained by adding for every edge* $(l', \alpha, \phi, \psi, r, l_f)$ *in* E_1 *with* $l_f \in L^f_1$ *an edge of the form* $(l', \alpha, \phi, \psi, r', l^{\mathsf{ini}}_1)$. *Where* r' *appends to* r *reset* $x := 0$ *for every clock* $x \in X_1$.
- $\phi \equiv \phi^*_1$ *then* $\mathcal{A}_{\varepsilon \cup \phi^+_1}$ *results of the union of* $\varepsilon \cup \phi^+_1$.
- $\phi \equiv \langle \phi_1 \rangle_{[d_1,d_2]}$ *then* $\mathcal{A}_\phi = (L_1, L^u_1, L^f_1, l^{\mathsf{ini}}_1, X_1 \cup \{x\}, V_1, H(C), E, I)$ *where* E *is obtained from* E_1 *by replacing every edge of the form* $(l', \alpha, \phi, \psi, r, l_f) \in E_1$ *where* $l_f \in L^f_1$ *with an edge* $(l', \alpha, \phi \wedge d_1 \leq x \leq d_2, \psi, r, l_f)$.

The network for $\varphi \equiv \varphi_1 \cap \varphi_2$ *consists of the automata in the network for* φ_1 *and in the network for* φ_2.

Figure 4 illustrates our construction. Note that some locations can delay "wait" for inputs in channel h. We call such locations *input locations*.

Definition 8 (Input Locations). *Given network with locations* L *and edges* E. *Location* l *is an* input location *iff exists edge* $(l, h?, \phi, \psi, r, l')$ *in* E.

The network $\mathcal{N}_\varphi$ has a number of syntactic structural invariants. These invariants are key to our results, proofs, and algorithms.

Lemma 1. *Given ETRE* φ *and the induced network* $\mathcal{N}_\varphi = \mathcal{A}_1, \ldots, \mathcal{A}_n$. *Then the network* $\mathcal{N}_\varphi$ *has the following structural invariants:*

1. *Initial locations are not accepting locations.*
2. *Accepting locations have no outgoing edges.*
3. *Input locations have a unique outgoing edge* e. *The destination location of* e *is not an accepting or input location.*
4. *For any location* l. *Location* l *is not urgent iff* l *is accepting or* l *is an input location.*

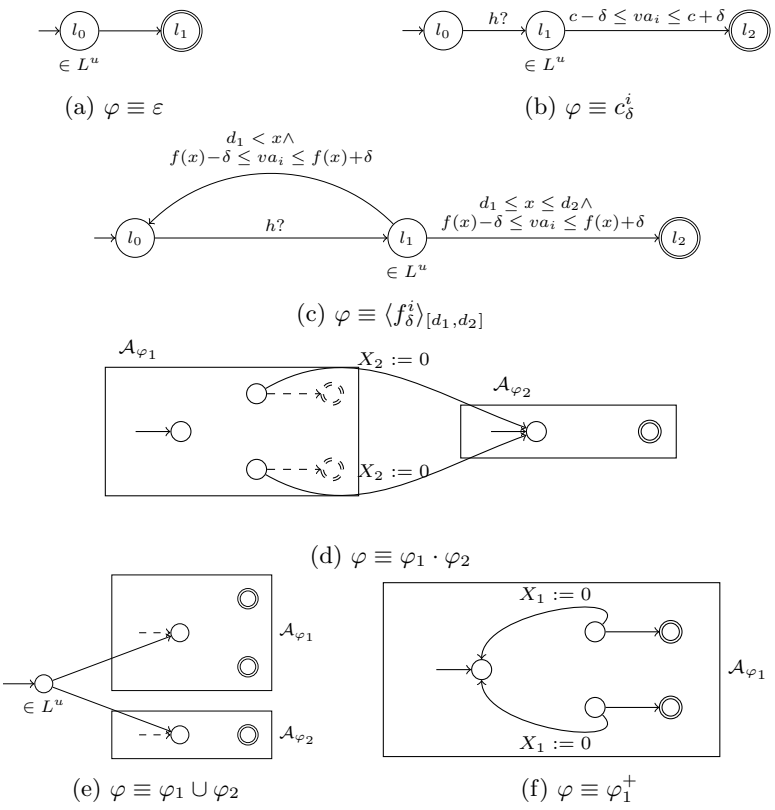


Fig. 4: XTA for ETRE

Given a finite timed word w Lemma 1 implies that accepting runs of the network including $\mathcal{A}_w$ and the automata in $\mathcal{N}_\varphi$ have a particular form. Informally, $\mathcal{A}_w$ delays and send an input, then $\mathcal{N}_\varphi$ consumes the input and urgently (a sequence $\xrightarrow[\mathsf{zt}]{}^*$) checks if the corresponding prefix of w is inside $[\![\varphi]\!]$.

Lemma 2. *Given finite timed word $w = t_0 \cdot a_0 \cdot t_1 \cdot a_1 \cdot \ldots \cdot t_m \cdot a_m$, ETRE φ, induced network $\mathcal{A}_w, \mathcal{A}_1, \ldots, \mathcal{A}_n$, and closed under time-additivity accepting run ρ. Then ρ is of the form:*

$$s_{\mathsf{ini}} \xrightarrow[\mathsf{zt}]{}^* s_0 \xrightarrow{t_0} \cdot \xrightarrow{E_{a_0}} s'_0 \xrightarrow[\mathsf{zt}]{}^* s_1 \xrightarrow{t_1} \cdot \xrightarrow{E_{a_1}} s'_1 \xrightarrow[\mathsf{zt}]{}^* \ldots \rightarrow^* s_m \xrightarrow{t_m} \cdot \xrightarrow{E_{a_m}} s'_m \xrightarrow[\mathsf{zt}]{}^* s_{m+1}$$

where $E_{a_i} \subseteq 2^E$ for $0 \leq i \leq m$ is the set of edges participating in the broadcast induced by the edge going from l_i to l_{i+1} in $\mathcal{A}_w$.

Lemma 2 implies that for accepting runs, the only possible delays (closed under time-additivity) are the ones in $\mathcal{A}_w$. This will allow us to implement an efficient discrete-time engine avoiding the expensive difference-bound matrices (DBM) operations. Note, that Lemma 2 indicates that there can be urgent

behavior from the initial state before automaton $\mathcal{A}_w$ has updated the shared variables va and sent a synchronization. If an operation e.g. concatenation of $\varphi_1 \cdot \varphi_2$ is applied, the variable valuation after executing $\mathcal{A}_{\varphi_1}$ will differ from the initial valuation for $\mathcal{A}_{\varphi_2}$ and could potentially block $\mathcal{A}_{\varphi_2}$. The following lemma ensures that the initial urgent behavior can not be blocked by variable guards.

Lemma 3. *Let* $s_{\mathsf{ini}} \xrightarrow[\mathsf{zt}]{u} s_0$ *be a prefix of an accepting run with* $u \in (E_1 \cup \cdots \cup E_n)^*$. *For any edge* $u_i = (l, \alpha, \phi, \psi, r, l')$ *in* u *the variable guard* ψ *is* true.

In the following we are given a finite timed word w and ETRE φ. We use $\mathcal{N}_\varphi^w$ to denote the network $\mathcal{A}_w, \mathcal{A}_1, \ldots, \mathcal{A}_n$.

Lemma 4. *Given finite timed word* w. *If* $w \in [\![\varphi]\!]$ *then* $\mathcal{N}_\varphi^w$ *is accepting.*

Lemma 5. *Given finite timed word* w. *If* $\mathcal{N}_\varphi^w$ *is accepting then* $w \in [\![\varphi]\!]$.

The proofs are quite technical and long. Detailed proofs for all the lemmas and theorems can be found in the extended version of this paper. Informally, for the above lemmas. We apply Lemma 2 and use structural induction on φ. For the case when $\varphi \equiv \phi$ we use induction on the length of w (accepting run ρ respectively). For the case when $\varphi \equiv \varphi_1 \cap \varphi_2$ we use the structural I.H. together with the fact that after receiving an input from $\mathcal{A}_w$, the actions in automata φ_1, φ_2 are urgent and commute (only read shared variables). The following theorem indicates that we can perform a reachability analysis in $\mathcal{N}_\varphi^w$ to decide if $w \in [\![\varphi]\!]$.

Theorem 1. *Given finite timed word* w *then* $w \in [\![\varphi]\!]$ *iff* $\mathcal{N}_\varphi^w$ *is accepting.*

5 Timed Pattern Matching with ETRE

Our goal is to efficiently match complex timed patterns described as an ETRE in time series data. An additional step is to transform a given time series into a timed word. This can be easily done by replacing the time stamps in the data by the induced delays. We require that the delays are not 0. Note that in [19] timed pattern matching returns a match set with uncountably many points where the expression matches. In contrast our algorithm will only return the start (and stop) indexes of the first sub-word that matches the expression. Our pattern matching method is described in Algorithm 1 presented in the spirit of [9]. For illustration purposes and clarity Line 1 ask to compute $\mathcal{A}_w$. As expected our implementation will not construct $\mathcal{A}_w$ but only inject the events and delays from w. Our implementation supports injection of events online (run time) and off-line for e.g. post processing.

Our method exploits the form of the runs as described in Lemma 2. The lemma indicates that there is an alternation among non-urgent and urgent behavior. For this reason, after delaying and injecting an event, Algorithm 1 calls Algorithm 2 for urgent exploration in Line 8 (or if the initial state is urgent Line 2). Algorithm 2, is the well known reachability algorithm for model checking. An invariant of this algorithm is that all the states in the waiting list W_{zt} are

Algorithm 1 Timed Pattern Matching for ETRE using XTA

Input Word $w = t_0 \cdot a_0 \dots t_m \cdot a_m$, ETRE φ.
Output First position k with $w_{0:k} \in [\![\varphi]\!]$, or $\perp$ if no match exists.

1: compute $\mathcal{A}_w, \mathcal{A}_1, \dots, \mathcal{A}_n$ from φ using Definition 7
2: **if** $\mathsf{zt}(s_{\mathsf{ini}})$ **then** $W :=$ *UrgentExploration*(s_{ini})
3: **else** $W := \{s_{\mathsf{ini}}\}$
4: **for** $k = 0$ to m **do**
5: $\quad W' := \emptyset$
6: $\quad$ **for all** $s \in W$ **do**
7: $\quad\quad$ **if** s is accepting for $\mathcal{A}_1, \dots, \mathcal{A}_n$ **then return** k
8: $\quad\quad$ **if** exists s' with $s \xrightarrow{t_k} \cdot \xrightarrow{E'} s'$ **then** $W' := W' \cup$ *UrgentExploration*(s')
9: $\quad W := W'$
10: **if** exists $s \in W$ s.t. s is accepting for $\mathcal{A}_1, \dots, \mathcal{A}_n$ **then return** m
11: **return** $\perp$

Algorithm 2 *UrgentExploration*(s)

Input urgent state s
Output a pair with a set with non urgent states and a boolean indicating if an accepting state was found

1: $W_{\mathsf{zt}} := \{s\}$, $P := \emptyset$, $W_{\neg\mathsf{zt}} := \emptyset$
2: **while** $W_{\mathsf{zt}} \neq \emptyset$ **do**
3: $\quad$ pick $s \in W_{\mathsf{zt}}$, $W_{\mathsf{zt}} := W_{\mathsf{zt}} \setminus \{s\}$
4: $\quad$ **if** $s \in P$ **then continue**
5: $\quad P := P \cup \{s\}$
6: $\quad$ compute stubborn set $\mathsf{St}(s)$
7: $\quad Succs := \{s' \mid \exists i \in \{1, \dots, n\}, e \in E_i . s \xrightarrow{e} s' \text{ and } \{e\} \in \mathsf{St}(s)\}$
8: $\quad$ **for all** $s \in Succs$ **do**
9: $\quad\quad$ **if** $\mathsf{zt}(s)$ **then** $W_{\mathsf{zt}} := W_{\mathsf{zt}} \cup \{s\}$
10: $\quad\quad$ **else** $W_{\neg\mathsf{zt}} := W_{\neg\mathsf{zt}} \cup \{s\}$
11: **return** $W_{\neg\mathsf{zt}}$

urgent. In addition, since automata $\mathcal{A}_1, \dots, \mathcal{A}_n$ do not share clocks and only read shared variables (they are independent) it is an ideal scenario for application of urgent partial order reduction techniques [12] Line 6.

The intuition for the correctness of our method is as follows. Algorithm 1 simulates w in the network $\mathcal{N}_\varphi$. If Line 7 returns k then all locations in $\mathcal{N}_\varphi$ are accepting. From the algorithm execution we can construct an accepting run indicating that $\mathcal{N}_\varphi^{w_{[0:k]}}$ is accepting. By Lemma 5 we have that $w_{[0:k]} \in [\![\varphi]\!]$. In the case where $w_{[0:k]} \in [\![\varphi]\!]$ for the first k. By Lemma 4, we have that exits accepting run ρ in $\mathcal{A}_{w_{[0:k]}}, \mathcal{A}_1, \dots, \mathcal{A}_n$. By Lemma 2 we have the form of ρ and we can easily see that the algorithm can execute ρ (or equivalent if POR is applied).

Theorem 2 (Total Correctness). *Given $w = t_0 \cdot a_0 \dots t_m \cdot a_m$ and ETRE φ. Algorithm 1 terminates and if Algorithm 1 returns $k \geq 0$ then $w_{[0:k]} \in [\![\varphi]\!]$ otherwise $w \notin [\![\varphi]\!]$.*

Table 1: ETRE expressions for case studies.

Property	**ETRE**
$\mathsf{qrs\text{-}const}_\delta$	1_δ^{MLII}
$\mathsf{qrs\text{-}const\text{-}verify}_\delta$	$\mathsf{qrs\text{-}const}_\delta \cap (1_{0.1}^{\text{annot}} \cup -0.5_{0.1}^{\text{annot}})$
$\mathsf{early\text{-}const}_\delta$	$\mathsf{qrs\text{-}const}_\delta \cdot \langle \Sigma^* \rangle_{[0.04s,0.65s]} \cdot \mathsf{qrs\text{-}const}_\delta$
early-const-verify	$\mathsf{early\text{-}const}_\delta \cap \langle \Sigma^* \rangle_{[0.47s,0.77s]} \cdot 1.0_{0.1}^{\text{annot}} \cdot \langle \Sigma^* \rangle_{[0,03s]}$
$\mathsf{qrs\text{-}func}_\delta$	$\langle f_\delta^{\text{MLII}} \rangle_{[0.039s,0.045s]}$
$\mathsf{qrs\text{-}func\text{-}verify}_\delta$	$\mathsf{qrs\text{-}func}_\delta \cap (\langle \Sigma^* \rangle_{[0s,0.45s]} \cdot 1_{0.1}^{\text{annot}} \cup -0.5_{0.1}^{\text{annot}} \cdot \langle \Sigma^* \rangle_{[0,03s]})$
$\mathsf{early\text{-}func}_\delta$	$\mathsf{qrs\text{-}func}_\delta \cdot \langle \Sigma^* \rangle_{[0.47s,0.67s]} \cdot \mathsf{qrs\text{-}func}_\delta$
$\mathsf{early\text{-}func\text{-}verify}_\delta$	$\mathsf{early\text{-}func}_\delta \cap \langle \Sigma^* \rangle_{[0.47s,0.77s]} \cdot 1_{0.1}^{\text{annot}} \cdot \langle \Sigma^* \rangle_{[0s,0.03s]}$
$\mathsf{sto\text{-}gates\text{-}closed}_\delta$	$300_{270}^{\text{havn}} \cap 77_\delta^{\text{port}}$
$\mathsf{sto\text{-}safe\text{-}fjord\text{-}ub}_\delta$	50_δ^{fjord}
sto-close-diff	$(-200_{100}^{\text{diff}} \cup 200_{100}^{\text{diff}}) \cap 51_{50}^{\text{port}}$
$\mathsf{sto\text{-}water\text{-}levels}_\delta$	$0_{25}^{\text{diff}} \cap 0_\delta^{\text{port}}$
$\mathsf{tmv\text{-}energy}_J$	$\langle (80_{20}^{\text{heat}})^* \rangle_J \cap \langle (60_{40}^{\text{vent}})^* \rangle_J$
$\mathsf{tmv\text{-}solar}_J$	$\langle (f(t) = 268.2 + 0.1t)_{100}^{\text{solar}} \rangle_J$

6 Evaluation

We have implemented our theory using C++ in the tool TimeRex. Our tool supports both online (run time) and offline pattern matching. The online version uses C++20 coroutines and ranges to compose lazy reading from a JSON-formatted stream of records, parsing and matching, and a buffer to store the passed records for a limited backtracking. Our tool includes a simple python GUI to enter ETRE and visualize the matches in the corresponding data. We run our experiments using a HPC cluster with AMD EPYC 9334 CPUs. For every experiment we allocate a single CPU with 10GB or RAM. Table 2 gives information on the time series data for each case study.

6.1 Arrhythmia

We consider the PhysionNet MIT-BIH Arrhythmia Database [10] [1]. We analyze the data for patient 100 which contains 650000 samples and 2274 annotations. Annotations classify ventricular myocardial depolarization called the *QRS complex* as normal (value -0.5) and arrhythmia (value 1). From the 2274 annotations

[1] `https://physionet.org/content/mitdb/1.0.0/`

33 are classified as arrhythmia. We use ETRE to approximate the QRS complex, and arrhythmia. Table 1 presents the ETRE we use for matching and Table 2a presents our results. We briefly describe the properties:

1. $\mathsf{qrs\text{-}const}_\delta$ approximates a QRS complex using the constant 1 (a peak) on the dimension MLII.
2. $\mathsf{qrs\text{-}const\text{-}verify}_\delta$ verifies $\mathsf{qrs\text{-}const}_\delta$ by checking (the second conjunct) that the corresponding annotation is either normal -0.5 or arrhythmia 1.
3. $\mathsf{early\text{-}const}_\delta$ approximates arrhythmia by matching a peak $\mathsf{qrs\text{-}const}_\delta$ followed by any value (Σ) in the given interval and then followed by another peak.
4. early-const-verify verifies $\mathsf{early\text{-}const}_\delta$ by checking that during the duration of $\mathsf{early\text{-}const}_\delta$ the value 1 appears in the annotation.
5. $\mathsf{qrs\text{-}func}_\delta$ approximates a QRS complex using ETRE $\langle f_\delta^{\mathrm{MLII}} \rangle_{[0.039s,0.045s]}$ where function f is a 7^{th} degree polynomial illustrated in Figure 1b.
6. $\mathsf{qrs\text{-}func\text{-}verify}_\delta$ verifies $\mathsf{qrs\text{-}func}_\delta$ by checking that during the duration of $\mathsf{qrs\text{-}func\text{-}verify}_\delta$ values -0.5 or 1 appear in the annotation signal.
7. $\mathsf{early\text{-}func}_\delta$ approximates arrhythmia by matching a QRS complex (using $\mathsf{qrs\text{-}func}_\delta$) followed by any sequence of values with a duration between 0.47 to 0.67 seconds and then followed by another QRS complex. Figure 1a shows a match for this expression when $\delta = 0.38$.
8. $\mathsf{early\text{-}func\text{-}verify}_\delta$ verifies $\mathsf{early\text{-}func}_\delta$ by checking that during the duration of $\mathsf{early\text{-}func}_\delta$ the value 1 appears in the annotation.

Table 2a presents our results of matching the ETRE described above for different δ values. We observe that execution times are below 2 minutes (we consider it fast given the number of samples), even in the presence of Σ^* which might cause a quadratic number of calls (in the number of samples) to Algorithm 1.

Consider $\mathsf{early\text{-}const}_{0.4}$ with 177 matches of which 155 were also annotated $\mathsf{qrs\text{-}const\text{-}verify}_{0.4}$. Remember that there is a total of 33 arrhythmia annotations. A visual inspection shows that there are repetitions (an arrhythmia detected more than once). We also observe that 1 arrhythmia annotation is not matched and few QRS complex are matched but not annotated as arrhythmia. We obtain a similar result with expression early-func. However, with few more false positives in a period where the heart rate of the patient is slightly higher. Note that the duration of the unconstrained time word (Σ^*) in early-const an in early-func is different and has a big impact in the matches. We report that finding the appropriate ETRE and their parameters such as δ or duration is involved and can greatly affect the number of matches.

6.2 Storm Surge Barriers

The data comes from sea water gates provided by Danish Coastal Authorities[2] through Storm_Safe project[3]. The gate installation consists of 14 identical gates at Hvide Sande harbor controlling the flow of water between North Sea and Ringkøbing fjord, with the following objectives:

[2] https://kyst.dk/hav-og-anlaeg/maalinger-og-data/
[3] https://www.interregnorthsea.eu/stormsafe

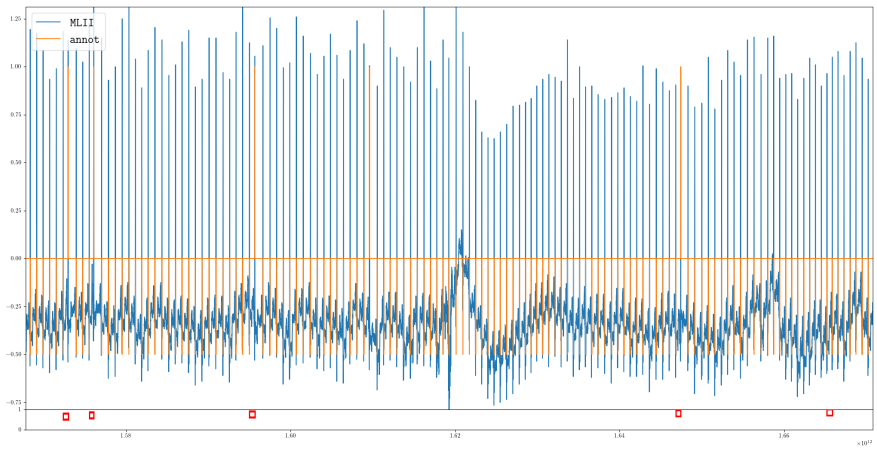


Fig. 5: early-const$_{0.4}$ missing one arrhythmia and matching one non annotated.

1. Protect the fjord coasts from North Sea storms: all gates must be closed during storms. Query for finding counter examples: sto-gates-closed$_\delta$.
2. Protect the fjord coasts from (precipitation and melting snow) floods by letting the fjord water out into North Sea and maintain the water levels below 25cm. Counter example query: sto-safe-fjord-ub$_\delta$.
3. Protect the installation and harbor navigation from strong water currents by allowing the gates to open only if the difference between sea and fjord water levels is less than 100cm. Counter example query: sto-close-diff.
4. Support the fish migration by opening all gates when the sea and fjord levels are the same. Counter example query: sto-water-levels$_\delta$.

The prepared data consists of a time series of: timestamp, North Sea water level, Ringkøbing fjord level (both at Hvide Sande harbor), state of each gate (0cm means fully closed, 550cm means fully open) and a reason behind the gate control command. The water levels are sampled every 10 minutes from a 4 minute running average and the gate state changes according to the control commands.

The algorithm has detected tens of thousands of requirement violations over the 14 years worth of data (see Table 2b). The violations were minor, stemming from underestimating the storm magnitude (e.g. sea water levels exceeding the fjord levels just a little above 1m, and/or operator closing the last fish water passage right after the sea levels were reported above the limit), making the tool useful for both local operator alerts and monitoring transparency. Even though the log files contain millions of records (hundreds of megabytes) the memory consumption stays very limited (up to 5.7MB), which makes the tool suitable

for embedded platforms, responsible for safe and autonomous operations close to the physical gates.

6.3 Smart Building

As example we consider the time series data from the AAU Civil Engineering building (Thomas Manns Vej 23, Aalborg). The building is highly automated and provides interfaces for automatic control of e.g. blinds, heaters, ventilation, etc. In spite of the buildings high level of automation, current controllers are not collaborative and energy consumption or user comfort can be improved.

1. $\mathsf{tmv\text{-}energy}_J$ identifies instances where the heater and the ventilation (cooling) are open (about 80% or 60%) simultaneously for a duration in the interval J seconds.
2. $\mathsf{tmv\text{-}solar}_J$ matches when the solar radiation is increasing according to a first degree polynomial for a duration in the interval J, this information can be used to control the automatic blinds to improve user comfort.

Table 2c shows the number of matches, time and memory usage for both ETRE. There are 23360 data points, memory usage is below 6Mb and computing times below 3 seconds. Figure 2 shows some matches for $\mathsf{tmv\text{-}energy}_J$. For every sample, our implementation only returns the start and stop indexes of the first match. However, we observe that many consecutive matches are found. While not incorrect it can be redundant. In deed, it would be interesting to apply parametric techniques e.g. [2,16] to find the maximal interval J where both the heating and the cooling are active.

7 Discussion

We presented Extended Timed Regular Expressions (ETRE) which extend TRE to the context of time series data. Events are vectors over signals. We present a sound and complete translation for finite words from ETRE to Timed Automata. We have implemented our approach in the tool TimeRex. We conduct experiments on 3 case studies with real world time series data. Our experiments are encouraging with fast execution times for complex ETRE in large time series data. We observe that finding the appropriate ETRE which contains several parameters e.g. for arrhythmia detection is complex task and requires domain expertise. Future work includes learning ETRE expressions, studying complement of ETRE expressions, and studying automata based optimizations.

Acknowledgments. We thank MD. Ernesto Barrientos and DVM. Gonzalo Malaga for their feedback on ECG and arrhythmia detection. We are also grateful to Ole Skovsgaard Daniel and the rest of DCA team for explaining storm barrier data and requirements. We thank the reviewers for their thorough and constructive comments and suggestions.

Table 2: ETRE experiments. Time in seconds. Memory usage for all experiments with mean 5.5MB and standard deviation 0.158MB

(a) Arrhythmia.

δ	**ETRE**	**Matches**	**Time**	**ETRE**	**Matches**	**Time**
0.32	qrs-const$_\delta$	9545	1.7	qrs-const-verify$_\delta$	2253	2.8
0.34		9918	1.7		2263	2.8
0.36		10274	1.8		2265	2.8
0.38		10591	1.8		2268	2.8
0.40		10875	1.8		2271	2.8
0.32	early-const$_\delta$	153	6.9	early-const-verify$_\delta$	136	20.3
0.34		158	7.2		140	20.9
0.36		165	7.3		145	21.5
0.38		169	7.5		148	22.1
0.40		177	7.9		155	23.9
0.32	qrs-func$_\delta$	2409	30.5	qrs-func-verify$_\delta$	2409	109.4
0.34		2681	33.1		2681	118.6
0.36		2930	35.1		2930	126.7
0.38		3189	37.0		3189	133.4
0.40		3469	37.9		3469	136.1
0.32	early-func$_\delta$	94	41.2	early-func-verify$_\delta$	29	63.7
0.34		105	46.1		34	69.9
0.36		115	50.9		39	77.9
0.38		128	54.0		40	84.3
0.40		143	57.8		44	92.1

(b) Storm Surge Barriers.

δ	**ETRE**	**Match**	**Time**
73	sto-gates-closed	39473	7.7
74		41049	7.5
75		144246	7.7
76		147210	7.7
20	sto-safe-fjord-ub	41429	7.2
21		45668	7.3
22		50116	7.3
23		55152	7.5
24		61829	7.2
25		68631	7.3
100	sto-close-diff	3074	8.9
0	sto-water-levels	72937	7.7
1		76733	7.7
2		214435	8.0
3		219446	8.0

(c) Thomas Manns Vej 23.

J in s.	**ETRE**	**Match**	**Time**
[1800, 3600]	tmv-energy$_J$	172	2.7
[3600, 5400]		81	2.7
[5400, 7200]		43	2.7
[7200, 9000]		13	2.7
[300, 600]	tmv-solar$_J$	8142	2.5
[600, 900]		5648	2.7
[900, 1200]		3845	2.9
[1200, 1500]		2619	3.0

(d) Time series data for case studies.

Data	**Dimensions**	**Size,MB**	**Samples**	**Annotations**
Arrhythmia	3	60	650000	2274
Storm-safe	19	50	764572	–
TMV 23	13	23	23360	–

References

1. Alur, R., Dill, D.L.: A theory of timed automata. Theoretical Computer Science 126(2), 183–235 (1994)
2. André, É., Hasuo, I., Waga, M.: Offline timed pattern matching under uncertainty. In: 23rd International Conference on Engineering of Complex Computer Systems, ICECCS 2018, Melbourne, Australia, December 12-14, 2018. pp. 10–20. IEEE Computer Society (2018), https://doi.org/10.1109/ICECCS2018.2018.00010
3. Asarin, E., Caspi, P., Maler, O.: A kleene theorem for timed automata. In: Proceedings of the 12th Annual IEEE Symposium on Logic in Computer Science. pp. 160–. LICS '97, IEEE Computer Society, Washington, DC, USA (1997), http://portal.acm.org/citation.cfm?id=788019.788856
4. Asarin, E., Caspi, P., Maler, O.: Timed regular expressions. Journal of the ACM 49(2), 172–206 (2002)
5. Bakhirkin, A., Ferrère, T., Maler, O., Ulus, D.: On the quantitative semantics of regular expressions over real-valued signals. In: Abate, A., Geeraerts, G. (eds.) Formal Modeling and Analysis of Timed Systems. pp. 189–206. Springer International Publishing, Cham (2017)
6. Bakhirkin, A., Ferrère, T., Nickovic, D., Maler, O., Asarin, E.: Online timed pattern matching using automata. In: Jansen, D.N., Prabhakar, P. (eds.) Formal Modeling and Analysis of Timed Systems. pp. 215–232. Springer International Publishing, Cham (2018)
7. Behrmann, G., David, A., Larsen, K.G.: A tutorial on UPPAAL. In: Bernardo, M., Corradini, F. (eds.) SFM-RT 2004. pp. 200–236. No. 3185 in LNCS, Springer (2004)
8. Dejan Ničković, X.Q.: Shape expressions for specifying and extracting signal features. Runtime Verification. RV 2019. Lecture Notes in Computer Science 11757, https://par.nsf.gov/biblio/10199912
9. Esparza, J., Blondin, M.: Automata Theory: An Algorithmic Approach. MIT Press (2023), https://books.google.dk/books?id=SP2nEAAAQBAJ
10. Goldberger, A., Amaral, L., Glass, L., Havlin, S., Hausdorg, J., Ivanov, P., Mark, R., Mietus, J., Moody, G., Peng, C.K., Stanley, H., Physiobank, P.: Components of a new research resource for complex physiologic signals. PhysioNet 101 (01 2000)
11. Goyvaerts, J., Levithan, S.: Regular Expressions Cookbook - Detailed Solutions in Eight Programming Languages, Second Edition. O'Reilly (2012), http://www.oreilly.de/catalog/9781449319434/index.html
12. Larsen, K., Mikučionis, M., Muniz, M., Srba, J.: Urgent partial order reduction for extended timed automata. In: Proceedings of the 18th International Symposium on Automated Technology for Verification and Analysis (ATVA'20). LNCS, vol. 12302, p. 179–195. Springer-Verlag (2020)
13. Lee, M., So, S., Oh, H.: Synthesizing regular expressions from examples for introductory automata assignments. In: Proceedings of the 2016 ACM SIGPLAN International Conference on Generative Programming: Concepts and Experiences. p. 70–80. GPCE 2016, Association for Computing Machinery, New York, NY, USA (2016), https://doi.org/10.1145/2993236.2993244
14. Maler, O., Nickovic, D.: Monitoring temporal properties of continuous signals. In: Lakhnech, Y., Yovine, S. (eds.) Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems. pp. 152–166. Springer Berlin Heidelberg, Berlin, Heidelberg (2004)
15. Maler, O., Nickovic, D., Pnueli, A.: From mitl to timed automata. In: Formal Modeling and Analysis of Timed Systems: 4th International Conference, FORMATS

2006, Paris, France, September 25-27, 2006. Proceedings 4. pp. 274–289. Springer (2006)
16. Mambakam, A., Asarin, E., Basset, N., Dang, T.: Pattern matching and parameter identification for parametric timed regular expressions. In: Proceedings of the 26th ACM International Conference on Hybrid Systems: Computation and Control. HSCC '23, Association for Computing Machinery, New York, NY, USA (2023), https://doi.org/10.1145/3575870.3587115
17. Ničković, D., Qin, X., Ferrère, T., Mateis, C., Deshmukh, J.: Shape expressions for specifying and extracting signal features. In: Runtime Verification: 19th International Conference, RV 2019, Porto, Portugal, October 8–11, 2019, Proceedings. p. 292–309. Springer-Verlag, Berlin, Heidelberg (2019), https://doi.org/10.1007/978-3-030-32079-9_17
18. Pnueli, A.: The temporal logic of programs. In: 18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 31 October - 1 November 1977. pp. 46–57. IEEE Computer Society (1977), https://doi.org/10.1109/SFCS.1977.32
19. Ulus, D., Ferrère, T., Asarin, E., Maler, O.: Timed pattern matching. In: Legay, A., Bozga, M. (eds.) Formal Modeling and Analysis of Timed Systems - 12th International Conference, FORMATS 2014, Florence, Italy, September 8-10, 2014. Proceedings. Lecture Notes in Computer Science, vol. 8711, pp. 222–236. Springer (2014), https://doi.org/10.1007/978-3-319-10512-3_16
20. Valizadeh, M., Fijalkow, N., Berger, M.: LTL learning on gpus. In: Gurfinkel, A., Ganesh, V. (eds.) Computer Aided Verification - 36th International Conference, CAV 2024, Montreal, QC, Canada, July 24-27, 2024, Proceedings, Part III. Lecture Notes in Computer Science, vol. 14683, pp. 209–231. Springer (2024), https://doi.org/10.1007/978-3-031-65633-0_10
21. Waga, M.: Online quantitative timed pattern matching with semiring-valued weighted automata. In: André, É., Stoelinga, M. (eds.) Formal Modeling and Analysis of Timed Systems. pp. 3–22. Springer International Publishing, Cham (2019)
22. Waga, M., Akazaki, T., Hasuo, I.: A boyer-moore type algorithm for timed pattern matching. In: Fränzle, M., Markey, N. (eds.) Formal Modeling and Analysis of Timed Systems. pp. 121–139. Springer International Publishing, Cham (2016)
23. Waga, M., Hasuo, I., Suenaga, K.: Efficient online timed pattern matching by automata-based skipping. In: Abate, A., Geeraerts, G. (eds.) Formal Modeling and Analysis of Timed Systems. pp. 224–243. Springer International Publishing, Cham (2017)

A Proofs

Proof (Lemma 1). Given word w, ETRE φ and induced network $\mathcal{A}_w, \mathcal{A}_1, \ldots, \mathcal{A}_n$, note that the Lemma only applies to $\mathcal{A}_i$ with $i \in \{1, \ldots, n\}$. We continue by structural induction on φ. Base case: $\varphi \equiv \varepsilon$, $\varphi \equiv c^i_\delta$, $\varphi \equiv \Sigma$, $\varphi \equiv \langle f^i_\delta \rangle_{[d_1,d_2]}$ with induced network $\mathcal{A}_w, \mathcal{A}_\varphi$. A simple insepction in $\mathcal{A}_\varphi$ can show that it has the desired structural invariants. Inductive step.

- $\varphi \equiv \langle \varphi_1 \rangle_{[d_1,d_2]}$. With induced network $\mathcal{A}_w, \mathcal{A}_\varphi$ where $\mathcal{A}_\varphi$ is obtained from $\mathcal{A}_{\varphi_1}$. By I.H. $\mathcal{A}_{\varphi_1}$ has the desired properties. Note that by Definition 7 locations, urgent locations, and accepting locations in $\mathcal{A}_\varphi$ are the same as in $\mathcal{A}_{\varphi_1}$. Further no edges were added or removed.
- $\varphi \equiv \varphi_1 \cdot \varphi_2$. With induced network $\mathcal{A}_w, \mathcal{A}_\varphi$ where $\mathcal{A}_\varphi$ is obtained from $\mathcal{A}_{\varphi_1}, \mathcal{A}_{\varphi_2}$. By I.H. $\mathcal{A}_{\varphi_1}$, $\mathcal{A}_{\varphi_2}$ respectively L_1, L_2 have the desired structural properties. Lemma 1 (1), clearly $l_1^{\mathsf{ini}} \notin L_2^f$. Lemma 1 (2), holds because $L^f = L_2^f$ and by I.H. L_2^f has no outgoing edges. Lemma 1 (3), holds because $E = E' \cup E_2$ where E' has been obtained from E_1 by redirecting the destination of some edges to l_2^{ini}. This will not add outgoing edges for any location and by I.H. $l_2^{\mathsf{ini}} \notin L_2^f$. Note that since the redirected edges where pointint to a location in L_1^{ini}. By I.H. the source of the edges where not input locations. Lemma 1 (4). ($\Rightarrow$ direction) Note that by Definition 7 we have $L \subseteq L_1 \cup L_2$ because L_1^f was removed. By I.H. $L_1 \cup L_2$ have the desired property and every edge leading to a location in L_1^f has been redirected to l_2^{ini}. We continue by contradiction. Assume $l \in L$ and $l \notin L^u$ and l is not an input location or $l \notin L_f^2$. By I.H. l was either an input location or $l \in L_f^2$. Note that redirecting edges did not change (created, removed) input locations. In both cases we get a contradiction. Therefore we can conclude that $\mathcal{A}_\varphi$ has the desired property. ($\Leftarrow$ direction) Holds from the I.H. and the fact that no location was changed fron non urgent to urgent.
- $\varphi \equiv \varphi_1 \cup \varphi_2$. With induced network $\mathcal{A}_w, \mathcal{A}_\varphi$ where $\mathcal{A}_\varphi$ is obtained from $\mathcal{A}_{\varphi_1}, \mathcal{A}_{\varphi_2}$. By I.H. $\mathcal{A}_{\varphi_1}$, $\mathcal{A}_{\varphi_2}$ have the desired structural properties. property. Definition 7 we have $L = \{l\} \cup L_1 \cup L_2$, $E \supseteq E_1 \cup E_2$, $l \in L^u$. Lemma 1 (1), holds because the new initial location $l \notin L_1^f \cup L_2^f$. Lemma 1 (2) (3), holds because l is nor accepting or input location. Lemma 1 (4), because of the I.H. we only need to consider the new added location l.($\Rightarrow$ direction) trivially holds because $l \in L^u$. ($\Leftarrow$ direction) trivially holds because l is not accepting nor an input location.
- $\varphi \equiv \varphi_1 \cap \varphi_2$. With induced network $\mathcal{A}_w, \mathcal{A}_{\varphi_1}, \mathcal{A}_{\varphi_2}$. By I.H. $\mathcal{A}_{\varphi_1}$, $\mathcal{A}_{\varphi_2}$ have the desired structural properties.
- $\varphi \equiv \varphi_1^+$ then $\mathcal{N}_\varphi^w = \mathcal{A}_w, \mathcal{A}_\varphi$. Lemma 1 (1), by I.H. we have $l_1^{\mathsf{ini}} \notin L_1^f$. Lemma 1 (2), By I.H. we have that L_1^f has no outgoing edges. Further, we have $L^f = L_1^f$ and the source locations of the added edges are not in L_1^f (otherwise L_1^f would have outgoing edges). Lemma 1 (3), we have $E = E_1 \cup E'$ where E' is obtained by adding for every edge $(l', \alpha, \phi, \psi, r, l_f)$ in E_1 with $l_f \in L_1^f$ an edge of the form $(l', \alpha, \phi, \psi, r', l_1^{\mathsf{ini}})$. By I.H. we have that l' is not

an input location (because it leads to accepting location l_f). Therefore, the new added edges do not affect any input location in $L = L_1$. In particular the added edges have no synchronizations (otherwise the source will be an input location). Lemma 1 (4), holds by I.H. and because $L^u = L_1^u$, $L^f = L_1^f$, and the added edges did not modify or creaded any input locations in $L = L_1$.
- $\varphi \equiv \varphi_1^*$. We have $\mathcal{N}_\varphi^w = \mathcal{A}_w, \mathcal{A}_{\varepsilon \cup \varphi_1^+}$ where $\mathcal{A}_{\varepsilon \cup \varphi_1^+}$ is the union of $\mathcal{A}_\varepsilon$ with $\mathcal{A}_{\varphi^1}$ which has been proven above. □

Corollary 1. *Given finite timed word $w = t_0 \cdot a_0 \cdot \ldots \cdot t_m \cdot a_m$, ETRE φ, induced network $\mathcal{N}_\varphi^w$, and closed under time-additivy accepting run $s_0 \xrightarrow{\lambda_0} s_1 \xrightarrow{\lambda_1} \ldots \xrightarrow{\lambda_{n-1}} s_n$. If $d_0 \ldots d_k$ are the non-zero delays obtained from $\lambda_0 \ldots \lambda_{n-1}$ by removing discrete transitions i.e. $\lambda_i \in 2^E$. Then $k = m$ and for $i \in \{0, \ldots, k\}$ we have $t_i = d_i$.*

Proof (Corollary 1). Let $\rho = s_0 \xrightarrow{\lambda_0} s_1 \xrightarrow{\lambda_1} \ldots \xrightarrow{\lambda_{n-1}} s_n$. We continue by induction on the length n of ρ. Base case $|\rho| = 1$. Then $\rho = s$, s is accepting and there are no transitions. Inductive step $|\rho| = n$. Case $\lambda_0 \in 2^E$ then by the induction hypothesis $d_0, \ldots, d_k$ are the delays in $\lambda_1, \ldots, \lambda_{n-1}$ and $t_i = d_i$, $k = m$ for $i \leq k$. Case $\lambda_0 > 0$, note that $\mathcal{A}_w$ is at location l_0. We need to consider the following cases:

- Case $\lambda_0 > t_0$. Then the invariant $x \leq t_0$ of l_0 will be violated.
- Case $\lambda_0 < t_0$. Then since ρ can reach the accepting location. The guard $x \geq t_0$ of the only outgoing edge has to be satisfied. Therefore we must have for some j with $v = \lambda_0, \ldots, \lambda_j$ induces delays $d'_0, \ldots, d'_{k'}$ with $\Sigma_{i=0}^{k'} d' = t_0$. Assume there exist a discrete transition $\lambda_i \in 2^E$ with $i < j$. By Lemma 1 since it is possible to delay, every component is at an input or accepting location. Accepting locations have no outgoing edges and λ_i can not be a syncronization because $\mathcal{A}_w$ is at l_0. Therefore v is a sequence of pure delays. But this contradicts the assumption that ρ is closed unter timed additivity. Thus this case is not possible.
- Case $\lambda_0 = t_0$. Then we have $d_0 = t_0$ and by I.H. we have $d_1, \ldots, d_k$ are the delays in $\lambda_1, \ldots, \lambda_{n-1}$, $t_i = d_k$ for $0 < i \leq k$, and $m - 1 = k - 1$. Therefore we have $d_0, \ldots, d_k$ are the delays in $\lambda_0, \ldots, \lambda_{n-1}$ and $m = k$.

Proof (Lemma 2). We expand ρ using w showing that it has the desired form. First let s_0 to be the first state in ρ having a delay that is $s_{\mathsf{ini}} \rightarrow^* s_0 \xrightarrow{\lambda_0} s$. Consider the following cases:

- $s_{\mathsf{ini}} = s_0$ then it trivially holds that $s_{\mathsf{ini}} \xrightarrow[\mathsf{zt}]{}^* s_0 \xrightarrow{\lambda_0} s$
- $s_{\mathsf{ini}} \neq s_0$ then we need to show $s_{\mathsf{ini}} \xrightarrow{e_0} s_1'' \xrightarrow{e_1} \ldots s_2'' \xrightarrow{e_2} \ldots \xrightarrow{e_m} s_0$ with $\mathsf{zt}(q)$ for $q \in \{s_{\mathsf{ini}}, s_1'', \ldots, s_m''\}$. Clearly at s_{ini} automaton $\mathcal{A}_w$ can not execute any edge without doing a delay. Therefore, $e_i \in E_j$ for $0 \leq i \leq m$ and $1 \leq j \leq n$ i.e. e_i belongs to some automaton $\mathcal{A}_j$ for φ. Since accepting locations have not outgoing egdes and input locations require synchorinzation from $\mathcal{A}_w$. By using Lemma 1 (4), we can conclude that the source locations for any e_i $0 \leq i \leq m$ are urgent. Therefore we can conclude $s_{\mathsf{ini}} \xrightarrow[\mathsf{zt}]{}^* s_0 \xrightarrow{\lambda_0} s$.

At this point ρ has been expanded to $s_{\mathsf{ini}} \xrightarrow{\mathsf{zt}}{}^{*} s_0 \xrightarrow{\lambda_0} s$. We need to show that $\lambda_0 = t_0$. Consider the following cases:

- Case $\lambda_0 > t_0$. Then the invariant $x \leq t_0$ of l_0 will be violated.
- Case $\lambda_0 < t_0$. Then since ρ can reach the accepting location. The guard $x \geq t_0$ of the only outgoing edge has to be satisfied. Therefore we must have for some j with $v = \lambda_0, \ldots, \lambda_j$ induces delays $d'_0, \ldots, d'_{k'}$ with $\Sigma_{i=0}^{k'} d' = t_0$. Assume there exist a discrete transition $\lambda_i \in 2^E$ with $i < j$. By Lemma 1 since it is possible to delay, every component is at an input or accepting location. Accepting locations have no outgoing edges and λ_i can not be a syncronization because $\mathcal{A}_w$ is at l_0. Therefore v is a sequence of pure delays. But this contradicts the assumption that ρ is closed unter timed additivity. Thus this case is not possible.
- Case $\lambda_0 = t_0$. As desired an ρ has prefix $s_{\mathsf{ini}} \xrightarrow{\mathsf{zt}}{}^{*} s_0 \xrightarrow{t_0} s$

At this point we have $s_{\mathsf{ini}} \xrightarrow{\mathsf{zt}}{}^{*} s_0 \xrightarrow{t_0} s$ and $\mathcal{A}_w$ is at location l_0 with enabled broadcast synchronization edge from l_0 to l_1. By semantics of XTA we have $s_{\mathsf{ini}} \xrightarrow{\mathsf{zt}}{}^{*} s_0 \xrightarrow{t_0} \cdot \xrightarrow{E_{a_0}} s'_0$. Note that since it was possible to delay t_0 by Lemma 1 automata for φ are at input or accepting locations. If all locations in s'_0 are accepting we are done. Otherwise, some $\mathcal{A}_i$ was at an input location and synchronized in E_{a_0} and s'_0 has a location destination of an input location say $l \in L_i$ by Lemma 1 that location is not an input location and not accepting, and $l \in L_i^u$. Therefore, we can conclude $\mathsf{zt}(s'_0)$. Therefore we can produce $s_{\mathsf{ini}} \xrightarrow{\mathsf{zt}}{}^{*} s_0 \xrightarrow{t_0} \cdot \xrightarrow{E_{a_0}} s'_0 \xrightarrow{\mathsf{zt}} s''_0$. In particular, the same reasoning as the case as when $s_{\mathsf{ini}} \neq s_0$ and we need to show $s_{\mathsf{ini}} \xrightarrow{\mathsf{zt}}{}^{*} s_0$ can be used to conclude $s_{\mathsf{ini}} \xrightarrow{\mathsf{zt}}{}^{*} s_0 \xrightarrow{t_0} \cdot \xrightarrow{E_{a_0}} s'_0 \xrightarrow{\mathsf{zt}}{}^{*} s_1$. The above described steps show how to expand ρ for $t_0 \cdot a_0$. We can repeat the previously described steps for $t_1 \cdot a_1 \cdot \ldots t_m \cdot a_m$. □

Proof (Lemma 4). Given φ, the size of φ is given by number of sub-terms in φ. We continue by induction the size $k = |\varphi|$. By assumption $w \in [\![\varphi]\!]$. $k = 1$

- $\varphi \equiv \varepsilon$. We have $\mathcal{N}_\varepsilon^w = \mathcal{A}_w, \mathcal{A}_\varepsilon = \mathcal{A}_1, \mathcal{A}_2$ where $\mathcal{A}_\varepsilon$ has unique accepting location, and $\mathcal{A}_\varepsilon$ can execute the unique edge $e \in E_\varepsilon$ since it has no constraints. Therefore $\mathcal{N}_\varepsilon^w$ can reach execute the accepting run $(l_1^{\mathsf{ini}}, l_2^{\mathsf{ini}}) \xrightarrow{e} (l_1^{\mathsf{ini}}, l)$ with $l_1^{\mathsf{ini}} \in L_1^f$ and $l \in L_2^f$.
- $\varphi \equiv c_\delta^i$. Since $w \in [\![\varphi]\!]$ by Definition 5 we have $w \in \{r \cdot (a^0, \ldots, a^n) \mid r \in \mathbb{R}_{\geq 0}, c - \delta \leq a^i \leq c + \delta$ and $a^k \in \mathbb{R}$ for $k \neq i\}$. Thus w is of the form $r \cdot a$ with $r > 0$. By Definition 7 we have $\mathcal{N}_\varphi^w = \mathcal{A}_w, \mathcal{A}_\varphi = \mathcal{A}_1, \mathcal{A}_2$. With $C = \{h\}$, $\mathcal{A}_w = (L_1, \emptyset, l_1^1, l_0^1, \{\hat{x}, x\}, \{va_1, \ldots, va_n\}, H(C), E_1, I_1)$ with $L_1 = \{l_0^1, l_1^1\}$, $I(l_0^1) = x \leq r$, and $E_1 = \{\ e_0^1 = (l_0^1, h!, x \geq r, \mathsf{true}, [va_0 := a_j^0, \ldots va_n := a_j^n, x := 0], l_1^1)$. Further $\mathcal{A}_\varphi = (L_2, \{l_1^2\}, \{l_2^2\}, l_0^2, \emptyset, \emptyset, H(C), E_2, I_2)$ where $L_2 = \{l_0^2, l_1^2, l_2^2\}$, $E_2 = \{e_0^2 = (l_0^2, h?, \mathsf{true}, \mathsf{true}, l_1^2), e_1^2 = (l_1^2, \tau, \mathsf{true}, c - \delta \leq va_i \leq c + \delta, [], l_2^2)\}$ and $I_2(l) = \mathsf{true}$ for $l \in \{l_0^2, l_1^2, l_2^2\}$. Then $\mathcal{N}_\varphi^w$ can execute the following accepting run:

delay r $((l_0^1, l_0^2), \mu_{\mathsf{ini}}, \nu_{\mathsf{ini}}) \xrightarrow{r} ((l_0^1, l_0^2), \mu_{\mathsf{ini}} + r, \nu_{\mathsf{ini}})$

broadcast $\xrightarrow{\{e_0^1, e_0^2\}} ((l_1^1, l_1^2), \hat{x} = r \wedge x = 0, va_0 = a^0 \wedge \cdots \wedge va_n = a^n)$

accept $\xrightarrow{\{e_1^2\}} ((l_1^1, l_2^2), \hat{x} = r \wedge x = 0, va_0 = a^0 \wedge \cdots \wedge va_n = a^n)$

Note that guard $c - \delta \leq va_i \leq c + \delta$ of e_1^2 is satisfied because $va_i = a^i$ and $c - \delta \leq a^i \leq c + \delta$.

- $\varphi \equiv \Sigma$. Similar as the case c_δ^i, but condition is weaker since there is no variable guard in the edge going to accepting in $\mathcal{A}_\varphi$.
- $\varphi \equiv \langle f_\delta^i \rangle_{[d_1,d_2]}$. Since $w \in [\![\varphi]\!]$ by Definition 5 we have

$$w \in \{w \mid w = t_0 \cdot a_0 \cdot \ldots \cdot t_m \cdot a_m \text{ with } \mathsf{dur}(w_{0:m-1}) < d_1, \\ \mathsf{dur}(w) \in [d_1, d_2], t_j > 0, a_j^k \in \mathbb{R} \text{ for } k \neq i, \text{and} \\ a_j^i \in [f_\delta^i(\mathsf{dur}(w_{0:j})) - \delta, f_\delta^i(\mathsf{dur}(w_{0:j})) + \delta]\}$$

By Definition 7 we have $\mathcal{N}_\varphi^w = \mathcal{A}_w, \mathcal{A}_\varphi = \mathcal{A}_1, \mathcal{A}_2$. With $C = \{h\}$, $\mathcal{A}_w$ as given by Definition 6 but renaming edges by its component e.g. edge e_4 to e_4^1 and $\mathcal{A}_\varphi$ as given by Definition 7 $\mathcal{A}_\varphi = (L, \{l_1\}, \{l_2\}, l_0, \{x_2\}, \emptyset, H(C), E, I)$ with $L = \{l_0, l_1, l_2\}$, $I(l) = \mathsf{true}$ for $l \in \{l_0, l_1, l_2\}$ and $E = \{ e_0^2 = (l_0, h?, \mathsf{true}, \mathsf{true}, [], l_1), e_1^2 = (l_1, \tau, x_2 < d_1, f(x_2) - \delta \leq va_i \leq f(x_2) + \delta, [], l_0), e_2^2 = (l_1, \tau, d_1 \leq x_2 \leq d_2, f(x_2) - \delta \leq va_i \leq f(x_2) + \delta, [], l_2) \}$. We continue by induction on the length $m = |w|$ of w. As I.H. let $s_{\mathsf{ini}} \rightarrow^* s$ with $s = ((l_m, l_2), \hat{x} = \mathsf{dur}(w_{0:m}) = x_2 \wedge x = 0, va = a_m)$.
Case $|w| = 1$ then $w = t_0 \cdot a_0$ then the following run is a witness

delay t_0 $((l_0^1, l_0^2), \mu_{\mathsf{ini}}, \nu_{\mathsf{ini}}) \xrightarrow{t_0} ((l_0^1, l_0^2), \mu_{\mathsf{ini}} + t_0, \nu_{\mathsf{ini}})$

broadcast $\xrightarrow{\{e_1^1, e_0^2\}} ((l_1^1, l_1^2), \hat{x} = x_2 = t_0 \wedge x = 0, va = a_0)$

accept $\xrightarrow{\{e_2^2\}} ((l_1^1, l_2^2), \hat{x} = x_2 = t_0 \wedge x = 0, va = a_0)$
since $\mathsf{dur}(w) = t_0 = x_2$ and $\mathsf{dur}(w) \in [d_1, d_2]$ by assumption.

Case $|w| = m$. By I.H. we have $\rho \equiv s_{\mathsf{ini}} \rightarrow^* s$ with $s = ((l_{m-1}, l_2), \hat{x} = \mathsf{dur}(w_{0:m-1}) = x_2 \wedge x = 0, va = a_{m-1})$. By assumption $\mathsf{dur}(w_{0:m-1}) < d_1$ then the guard $x_2 < d_1$ of edge e_1^2 is satisfied and there is a transition $s \xrightarrow{\{e_1^2\}} s'$ with $s' = ((l_{m-1}, l_0^2), \hat{x} = \mathsf{dur}(w_{0:m-1}) = x_2 \wedge x = 0, va = a_{m-1})$. The next steps are analogous to the base case.

Inductive step $|\varphi| = k > 1$.

- $\varphi \equiv \langle \varphi_1 \rangle_{[d_1,d_2]}$ then $w \in [\![\varphi]\!] \cap \{a \mid \mathsf{dur}(a) \in [d_1, d_2]\}$. Then by I.H we have exists accepting run ρ_1 in $\mathcal{A}_w, \mathcal{A}_{\varphi_1} = \mathcal{A}_1, \mathcal{A}_2$ with $\rho_1 = s_0 \rightarrow^* s' = ((l_f^1, l'), \mu, \nu) \xrightarrow{e_f} ((l_f^1, l_f^2), \mu_1, \nu_1)$ with $l^2 \in L_2^f$. By Definition 7 $\mathcal{A}_\varphi$ is obtained from $\mathcal{A}_{\varphi_1} = \mathcal{A}_2$ as $\mathcal{A}_\varphi = (L_2, L_2^u, L_2^f, l_2^{\mathsf{ini}}, X_2 \cup \{x\}, V_2, H(C), E, I)$ where E is obtained from E_2 by replacing for every edge of the form $(l', \alpha, \phi, \psi, r, l_f) \in E_2$ with $l_f \in L_2^f$ a new edge $(l', \alpha, \phi \wedge d_1 \leq x \leq d_2, \psi, r, l)$, and $I = I_1 \cup \{(l, \mathsf{true})\}$. Then edge e_f in execution ρ_1 has been replaced by edge $e'_f = (l', \alpha, \phi \wedge d_1 \leq x \leq d_2, \psi, r, l_f^2)$ with $\mu \models \phi$ and $\nu \models \psi$. Since we have that $\mu(x) = dur(w)$ and $d_1 \leq dur(w) \leq d_2$ the edge e'_f is enabled at s' and we can produce an accepting run from ρ_1 as follows $s_0 \rightarrow^* s' \xrightarrow{e'_f} ((l_f^1, l_f^2), \mu_1, \nu_1)$.

- $\varphi \equiv \varphi_1 \cdot \varphi_2$. Then $w \in [\![\varphi_1]\!] \cdot [\![\varphi_2]\!]$. Which implies $w = w_1 \cdot w_2$ with $w_1 \in [\![\varphi_1]\!]$ and $w_2 \in [\![\varphi_2]\!]$. By Definition 7 we have $\mathcal{N}_\varphi^w = \mathcal{A}_w, \mathcal{A}_\varphi$. With $C = \{h\}$. $\mathcal{A}_w$ is as given by Definition 6 from $w = w_1 \cdot w_2$. By I.H. we have exists accepting run ρ_1 in $\mathcal{A}_{w_1}, \mathcal{A}_{\varphi_1}$ and ρ_2 in $\mathcal{A}_{w_2}, \mathcal{A}_{\varphi_2}$. We construct accepting run ρ in $\mathcal{A}_w, \mathcal{A}_\varphi$ from ρ_1 and ρ_2. Note that if $\varphi_i \equiv \varepsilon$ then the run can be trivially extended. Therefore, assume $\varphi_i \neq \varepsilon$. Further, let $\mathcal{A}_{\varphi_i}$ be $\mathcal{A}_i$ with L_i etc. Then let $w = \underbrace{t_0 \cdot a_0 \cdot \ldots \cdot t_i \cdot a_i}_{w_1} \cdot \underbrace{t_{i+1} \cdot a_{i+1} \cdot \ldots \cdot t_m \cdot a_m}_{w_2}$ and let $\rho_1 = ((l_0, l_1^{\mathsf{ini}}), \mu_{\mathsf{ini}}^1, \nu_{\mathsf{ini}}^1) \xrightarrow{u}{}^* ((l_{i+1}, l^1), \mu^1, \nu^1) \xrightarrow{e_f} ((l_{i+1}, l_1^f), \mu^{1'}, \nu^{1'})$ and let $\rho_2 = ((l_{i+1}, l_2^{\mathsf{ini}}), \mu_{\mathsf{ini}}^2, \nu_{\mathsf{ini}}^2) \xrightarrow{v}{}^* ((l^f, l^f), \mu^2, \nu^2)$. By Definition 7 edge e_f has added edge $e = (l^1, \alpha, \phi, \psi, r, l_{\mathsf{ini}}^2)$ with r resetting all clocks in X_2 and clearly $\mu^1 \models \phi$ and $\nu^1 \models \psi$. Then ρ can be constructed from the delay and edge transitions in u, v in ρ_1, ρ_2 as follows: $\rho = ((l_0, l^{\mathsf{ini}}), \mu_{\mathsf{ini}}, \nu_{\mathsf{ini}}) \xrightarrow{u}{}^* ((l_{i+1}, l^1), \mu, \nu) \xrightarrow{e} ((l_{i+1}, l_2^{\mathsf{ini}}), \mu', \nu') \xrightarrow{v}{}^* s^f$ where locations in s^f are accepting, $\mu'|_{\mathcal{A}_2} = \mu_{\mathsf{ini}}^2$ because edge e resets all clocks in X_2. In addition Lemma 3 ensures that there is no variable guard until a broadcast synchronization has occured.
- $\varphi \equiv \varphi_1 \cup \varphi_2$. Then $w \in [\![\varphi_1]\!] \cup [\![\varphi_2]\!]$. Definition 7 $\mathcal{N}_\varphi^w = \mathcal{A}_w, \mathcal{A}_\varphi$ with $\mathcal{A}_\varphi = (L_1 \cup L_2 \cup \{l\}, L_1^u \cup L_2^u \cup \{l\}, L_1^f \cup L_2^f, l, X_1 \cup X_2, V_1 \cup V_2, H(C), E' \cup E_1 \cup E_2, I_1 \cup I_2)$ where $E' = \{(l, \tau, \mathsf{true}, \mathsf{true}, [], l_i^{\mathsf{ini}}) \mid 1 \leq i \leq 2\}$. W.l.o.g. let $w \in [\![\varphi_1]\!]$. By I.H. we have that $\mathcal{A}_w, \mathcal{A}_{\varphi_1}$ have an accepting run $\rho_1 = s_{\mathsf{ini}}^1 \xrightarrow{u}{}^* s_f^1$. Definition 7 $\mathcal{A}_\varphi$ has added edge $e = ((l, \tau, \mathsf{true}, \mathsf{true}, [], l_1^{\mathsf{ini}}))$. Then accepting run ρ can be constructed from ρ_1 as follows:

$$((l_{\mathsf{ini}}, l), \mu_{\mathsf{ini}}, \nu_{\mathsf{ini}}) \xrightarrow{e} ((l_{\mathsf{ini}}, l_1^{\mathsf{ini}}), \mu_{\mathsf{ini}}, \nu_{\mathsf{ini}}) \xrightarrow{u}{}^* s_f$$

  where locations in s_f are accepting locations.
- $\varphi \equiv \varphi_1 \cap \varphi_2$. Then $w \in [\![\varphi_1]\!] \cap [\![\varphi_2]\!]$. Definition 7 $\mathcal{N}_\varphi^w = \mathcal{A}_w, \mathcal{A}_{\varphi_1}, \mathcal{A}_{\varphi_2}$. By I.H. we have accepting runs ρ^i in $\mathcal{A}_w, \mathcal{A}_{\varphi_i}$. We construct accepting run ρ for $\mathcal{N}_\varphi^w$ from ρ^1 and ρ^2. We continue by lexicographic induction, let $S = \{0, \ldots, |\rho^1|\} \times \{0, \ldots, |\rho^2|\}$ and $\preccurlyeq \subseteq S \times S$. As I.H. have that for any (i, j) such that $\rho_0^1 \rightarrow^* \rho_i^1$, $\rho_0^2 \rightarrow^* \rho_j^2$ with $\rho_i^1 = ((l^0, l^1, \mu^1, \nu^1))$, $\rho_j^2 = ((l^0, l^2, \mu^2, \nu^2))$ such that $\mu^1(\hat{x}) = \mu^2(\hat{x})$ there exists run $\rho \equiv s_{\mathsf{ini}} \rightarrow^* ((l^0, l^1, l^2), \mu, \nu)$ in $\mathcal{N}_\varphi^w$ such that $\mu|_{\mathcal{A}_w, \mathcal{A}_1} = \mu^1$, $\mu|_{\mathcal{A}_w, \mathcal{A}_2} = \mu^2$, $\nu|_{\mathcal{A}_w, \mathcal{A}_1} = \nu^1$ and $\nu|_{\mathcal{A}_w, \mathcal{A}_2} = \nu^2$.
  We show case (i, j). From Lemma 2 we have that ρ_i^1 is of the form

$$s_{\mathsf{ini}}^1 \xrightarrow{\mathsf{zt}}{}^* s_0^1 \xrightarrow{t_0} \cdot \xrightarrow{E_{a_0}} s_0^{1'} \xrightarrow{\mathsf{zt}}{}^* s_1 \xrightarrow{t_1} \cdot \xrightarrow{E_{a_1}} s_1^{1'} \xrightarrow{\mathsf{zt}}{}^* \ldots \rightarrow^* s_i^1 = ((l^0, l^1), \mu^1, \nu^1)$$

  and ρ^2 is of the form

$$s_{\mathsf{ini}}^2 \xrightarrow{\mathsf{zt}}{}^* s_0^2 \xrightarrow{t_0} \cdot \xrightarrow{E_{a_0}} s_0^{2'} \xrightarrow{\mathsf{zt}}{}^* s_1 \xrightarrow{t_1} \cdot \xrightarrow{E_{a_1}} s_1^{2'} \xrightarrow{\mathsf{zt}}{}^* \ldots \rightarrow^* s_j^2 = ((l^0, l^2), \mu^2, \nu^2)$$

  Let $(i', j') \preccurlyeq (i, j)$, consider the case with $i' + 1 = i$ and $j' = j$ (the case for $j' < j$ is symmetric). By I.H. we have exists $s_{\mathsf{ini}}^1 \rightarrow^* s_{i'}^1 = ((l^0, l_{i'}^1, \mu_{i'}^1, \nu_{i'}^1))$ in $\mathcal{A}_w, \mathcal{A}_{\varphi_1}$, and $\rho \equiv s_{\mathsf{ini}} \rightarrow^* \rho_{i',j} = ((l^0, l_{i'}^1, l^2), \mu, \nu)$ in $\mathcal{A}_w, \mathcal{A}_{\varphi_1}, \mathcal{A}_{\varphi_2}$ with $\mu|_{\mathcal{A}_w, \mathcal{A}_1} = \mu_{i'}^1$, $\mu|_{\mathcal{A}_w, \mathcal{A}_2} = \mu^2$, $\nu|_{\mathcal{A}_w, \mathcal{A}_1} = \nu_{i'}^1$ and $\nu|_{\mathcal{A}_w, \mathcal{A}_2} = \nu^2$. As a consequence of Lemma 2 and the form of ρ_1 we need to consider the following cases for $\rho_{i'}^1 \xrightarrow{\lambda} \rho_i^1$.

- $\lambda \in \mathbb{R}_{>0}$. That is $\lambda = t_k$ for some $k \in \{0, \ldots, m\}$. We have that $\mu^1_{i'}(\hat{x}) = \mu^2_j(\hat{x})$ and $\mu^1_{i'}(x) = \mu^2_j(x)$. If $l_2 \notin L^u_2$ then it is possible to delay for $\mathcal{A}_w, \mathcal{A}_2$ and we have $\rho \equiv s_{\mathsf{ini}} \rightarrow^* \rho_{i',j} \xrightarrow{t_k} \rho_{i,j}$ with $\mu(\rho_{i,j})|_{\mathcal{A}_w,\mathcal{A}_1} = \mu(\rho_i)$ (similar for $\mathcal{A}_w, \mathcal{A}_2$) as desired. If $l_2 \in L^u_2$, then since $\mathcal{A}_w$ is accepting and the form of the run we have $\rho_2 \equiv \rho^2_{\mathsf{ini}} \rightarrow^* \rho^2_j \xrightarrow{v}_{\mathsf{zt}} \rho^2_k \xrightarrow{t_k} \rho^2_{k-1}$. Where $v \in (2^{E_2})^*$ is a sequence of edges in E_2. That is $\mathcal{A}_2$ executed a number of edges without any delays. By construction of $\mathcal{A}_1$ and $\mathcal{A}_2$ we have that $X(\mathcal{A}_1) \cap X(\mathcal{A}_2) = \emptyset$ and $V(\mathcal{A}_1) \cap V(\mathcal{A}_2) = \emptyset$, thus $\mathcal{A}_1$ can not disable any edge in v. We can extend the accepting run as follows: $\rho \equiv s_{\mathsf{ini}} \rightarrow^* \rho_{i',j} \xrightarrow{v}_{\mathsf{zt}} \rho_k \xrightarrow{t_k} \rho_{i,j}$ as which agrees on the valuations as desired.
- $\lambda \in 2^E$.
  - Case $\lambda = e \in E_1$ is a discrete transition in $\mathcal{A}_1$. Then we have $\rho^1_{\mathsf{ini}} \rightarrow^* \rho^1_{i'} \xrightarrow{e} \rho^1_i$. Since by I.H. we have $\mu(\rho_{i',j})|_{\mathcal{A}_w,\mathcal{A}_1} = \mu(\rho^1_{i'})$, and $\mu(\rho^1_{i'})$ satisfies the guards of e (similar for variable valuations) we can extend the accepting run as follows: $\rho \equiv s_{\mathsf{ini}} \rightarrow^* \rho_{i',j} \xrightarrow{e} \rho_{i,j}$. By construction of $\mathcal{A}_1$ and $\mathcal{A}_2$ we have that $X(\mathcal{A}_1) \cap X(\mathcal{A}_2) = \emptyset$ and $V(\mathcal{A}_1) \cap V(\mathcal{A}_2) = \emptyset$. Therefore, edge e has can not modify any clock or variable in $\mathcal{A}_2$ and we have $\mu(\rho_{i,j})|_{\mathcal{A}_w,\mathcal{A}_2} = \mu(\rho^2_j)$ (similar for $\mathcal{A}_w, \mathcal{A}_1$ and variable valuations).
  - Case $\lambda = E_{a_k}$ with $E_{a_k} = \{e_w, e_1\}$ i.e. a broadcast transition. If $l_2 \notin L^u_2$ then by Lemma 1 (4) l is either an accepting or an input location. If l_2 is accepting then we can just extend the run $\rho \equiv s_{\mathsf{ini}} \rightarrow^* \rho_{i',j} \xrightarrow{\{e_w,e_1\}} \rho_{i,j}$. If l_2 is an input location, then by definition of input location and Lemma 1 (3) we have unique outgoing edge e_2 from l_2 such that $\rho^2_{\mathsf{ini}} \rightarrow^* \rho^2_j \xrightarrow{\{e_w,e_2\}} (l'_0, l'_2, \mu^{2'}, \nu^{2'})$. Since e_1 and e_2 can not disable each other we have $\rho \equiv s_{\mathsf{ini}} \rightarrow^* \rho_{i',j} \xrightarrow{\{e_w,e_1,e_2\}} ((l'_0, l_1, l'_2), \mu', \nu')$ and since e_2 can not modify any clock or variable in $\mathcal{A}_1$ we have $\mu'|_{\mathcal{A}_w,\mathcal{A}_1} = \mu^1$ (analogously for e_1). Finally, note that $\nu'|_{\mathcal{A}_w,\mathcal{A}_1} = \nu^1$ and $\nu'|_{\mathcal{A}_w,\mathcal{A}_2} = \nu^{2'}$

– $\varphi \equiv \varphi^+_1$. Then $w \in [\![\varphi^+_1]\!] \equiv \underbrace{[\![\varphi_1]\!] \cdot \ldots \cdot [\![\varphi_1]\!]}_{i>0 \text{ times}}$. We continue by induction on i.

Base case $i = 1$. By I.H. of the structural induction we have $w \in [\![\varphi_1]\!]$.
Case $i + 1$. We have $u \cdot v = w \in \underbrace{[\![\varphi_1]\!] \cdot \ldots \cdot [\![\varphi_1]\!]}_{i+1 \text{ times}}$ and By I.H. we have $u \in \underbrace{[\![\varphi_1]\!] \cdot \ldots \cdot [\![\varphi_1]\!]}_{i \text{ times}}$ with accepting run ρ in $\mathcal{A}_u, \mathcal{A}_{\varphi^+_1}$ of the form $\rho_0 \rightarrow^* \rho_k \xrightarrow{e'} \rho_i$ induced by u. By Definition 7 we have that edge $e' = (l', \alpha, \phi, \psi, r, l_f)$ in $E_{\varphi^+_1}$ with $l_f \in L^f_{\varphi^+_1}$ induced an added edge of the form $e = (l', \alpha, \phi, \psi, r', l^{\mathsf{ini}}_1)$. Note that by structural I.H. we have $v \in [\![\varphi_1]\!]$ with accepting run ρ^1 in $\mathcal{A}_v, \mathcal{A}_{\varphi_1}$ with ρ^1 of the form $\rho^1_{\mathsf{ini}} \xrightarrow{v'} \rho^1_f$ where $v' \in (2^E \cup \mathbb{R}_{>0})^*$ is the sequence of transitions induced by word v. We can now extend ρ to accept $u \cdot v$ with

$\rho_0 \to^* \rho_k \xrightarrow{e} ((l_u, l^1_{\mathsf{ini}}), \mu, \nu) \xrightarrow{v'} s_f$. Note that edge e resets all clocks in $\mathcal{A}_{\varphi_1}$. Therefore $\mu(x) = \mu(\rho^1_0)(x)$ for $x \in X_{\varphi_1}$. Thus clock guards for edges in v' are satisfied. Lemma 3 ensures that edges in v' have no variable guards until a broadcast has been received. Ensuring that v' can be executed starting at $\nu \neq \rho^1_{\mathsf{ini}} = \nu_{\mathsf{ini}}$. □

Proof (Lemma 5). Given finite timed word $w = t_0 \cdot a_0 \cdot t_1 \cdot a_1 \cdot t_2 \cdot a_2 \cdot \ldots t_m \cdot a_m$. If $\mathcal{N}^w_\varphi$ is accepting then $w \in [\![\varphi]\!]$. Since $\mathcal{N}^w_\varphi$ is accepting we have exist an accepting run ρ We continue by induction the size $k = |\varphi|$. $k = 1$

- $\varphi \equiv \varepsilon$. Then clearly $\varepsilon \in [\![\varepsilon]\!]$. If $w \neq \varepsilon$, then it is not possible to perform a broadcast synchronization and $\mathcal{A}_w$ is stuck in a non-acepting location. A contradiction.
- $\varphi \equiv c^i_\delta$. From Lemma 2 ρ is of the form $s_{\mathsf{ini}} \xrightarrow{\mathsf{zt}}{}^* s_0 \xrightarrow{t_0} \cdot \xrightarrow{E_{a_0}} s'_0 \xrightarrow{\mathsf{zt}}{}^* s_f$ In particular because of Definition 7 we have the concrete run $s_{\mathsf{ini}} \xrightarrow{t_0} \cdot \xrightarrow{\{e_w, e^1_0\}} s_1 \xrightarrow{\{e^1_1\}} s_f$ where all locations in s_f are accepting. e_w is the only edge in $\mathcal{A}_w$, e^1_0 is the input edge in $\mathcal{A}_\varphi$ and $e^1_1 = (l_1, \tau, \mathsf{true}, c - \delta \leq va_i \leq c + \delta, [], l_2)$ in $\mathcal{A}_\varphi$. Since $a_0 = va$ we have that $t_0 \cdot a_0 \in [\![c^i_\delta]\!]$.
- $\varphi \equiv \Sigma$. Trivial, similar as the case c^i_δ.
- $\varphi \equiv \langle f^i_\delta \rangle_{[d_1,d_2]}$. By Definition 7 we have $\mathcal{A}_1, \mathcal{A}_2$ with $\mathcal{A}_2$ having edges $E = \{ e^2_0 = (l_0, h?, \mathsf{true}, \mathsf{true}, [], l_1), e^2_1 = (l_1, \tau, x_2 < d_1, f(x_2) - \delta \leq va_i \leq f(x_2) + \delta, [], l_0), e^2_2 = (l_1, \tau, d_1 \leq x_2 \leq d_2, f(x_2) - \delta \leq va_i \leq f(x_2) + \delta, [], l_2) \}$. By Lemma 2 ρ is of the form

$$s_{\mathsf{ini}} \xrightarrow{\mathsf{zt}}{}^* s_0 \xrightarrow{t_0} \cdot \xrightarrow{E_{a_0}} s'_0 \xrightarrow{\{e^2_1\}}_{\mathsf{zt}} s_1 \xrightarrow{t_1} \cdot \xrightarrow{E_{a_1}} s'_1 \ldots \to^* s_m \xrightarrow{t_m} \cdot \xrightarrow{E_{a_m}} s'_m \xrightarrow{e^2_2}_{\mathsf{zt}} s_{m+1}$$

  Note that clock x_2 is never reseted. Then, because of the clock guard from $x_2 < d_1$ from e^2_1 we have that $\mathsf{dur}(w_{0:m-1}) < d_1$ and because of the clock guard $d_1 \leq x_2 \leq d_2$ we have $\mathsf{dur}(w_{0:m}) \in [d_1, d_2]$. Finally, after every synchronization E_{a_j} for $0 \leq j \leq m$ we have $va = a_j$ and the guard $f(x_2) - \delta \leq va_i \leq f(x_2) + \delta$ in e^2_1 and e^2_2 ensure $a^i_j \in [f^i_\delta(\mathsf{dur}(w_{0:j})) - \delta, f^i_\delta(\mathsf{dur}(w_{0:j})) + \delta$. Therefore, $w \in [\![\langle f^i_\delta \rangle_{[d_1,d_2]}]\!]$.

Inductive step $|\varphi| = k > 1$

- $\varphi \equiv \langle \varphi_1 \rangle_{[d_1,d_2]}$. By assumption we have exists accepting run ρ in $\mathcal{A}_w, \mathcal{A}_\varphi$. By Lemma 2 we have that ρ has the shape

$$s_{\mathsf{ini}} \xrightarrow{\mathsf{zt}}{}^* s_0 \xrightarrow{t_0} \cdot \xrightarrow{E_{a_0}} s'_0 \xrightarrow{\mathsf{zt}}{}^* \ldots \to^* s_m \xrightarrow{t_m} \cdot \xrightarrow{E_{a_m}} s'_m \xrightarrow{\mathsf{zt}}{}^* s_m \xrightarrow{e'}_{\mathsf{zt}} s_{m+1}$$

  Where $e' = (l', \alpha, \phi \wedge d_1 \leq x \leq d_2, \psi, r, l_f)$ has been obtained from $e = (l', \alpha, \phi, \psi, r, l_f) \in E_{\varphi_1}$ and $l_f \in L^f_{\varphi_1}$. Now, $\mathcal{A}_\varphi$ has been obtained from $\mathcal{A}_{\varphi_1}$ by adding clock x. We have that $\Sigma^m_{i=0} t_i = \mu(s_m)(x)$ and $\mu(s_m) \models \phi \wedge d_1 \leq x \leq d_2$. Therefore for the weaker guard of e we have $\mu(s_m) \models \phi$ so we can produce accepting run ρ' in $\mathcal{A}_w, \mathcal{A}_{\varphi_1}$.

$$s_{\mathsf{ini}} \xrightarrow{\mathsf{zt}}{}^* s_0 \xrightarrow{t_0} \cdot \xrightarrow{E_{a_0}} s'_0 \xrightarrow{\mathsf{zt}}{}^* \ldots \to^* s_m \xrightarrow{t_m} \cdot \xrightarrow{E_{a_m}} s'_m \xrightarrow{\mathsf{zt}}{}^* s_m \xrightarrow{e}_{\mathsf{zt}} s_{m+1}$$

  And by I.H. we have and $w \in [\![\varphi_1]\!]$. Since $\mu(s_m) \models \phi \wedge d_1 \leq x \leq d_2$ we have $\mathsf{dur}(w) \in [d_1, d_2]$. Therefore, $w \in [\![\varphi]\!] \cap \{w \mid \mathsf{dur}(w) \in [d_1, d_2]\}$.

- $\varphi \equiv \varphi_1 \cdot \varphi_2$. By assumption we have exists accepting run ρ in $\mathcal{A}_w, \mathcal{A}_\varphi$. In the following let $w = u \cdot v$ thus $u = t_0 \cdot a_0 \cdot \ldots t_k \cdot a_k$ and $v = t_{k+1} \cdot a_{k+1} \cdot \ldots t_m \cdot a_m$ By Lemma 2 we have that ρ has the shape

$$s_{\mathsf{ini}} \xrightarrow{\mathsf{zt}}{}^* s_0 \xrightarrow{t_0} \cdot \xrightarrow{E_{a_0}} s'_0 \xrightarrow{\mathsf{zt}}{}^* \ldots \rightarrow^* s_k \xrightarrow{t_k} \cdot \xrightarrow{E_{a_k}} s'_k \xrightarrow{\mathsf{zt}}{}^* \cdot \xrightarrow{e'} s_{k+1}$$
$$\xrightarrow{\mathsf{zt}}{}^* s'_{k+1} \xrightarrow{t_{k+1}} \cdot \xrightarrow{E_{a_{k+1}}} s''_{k+1} \xrightarrow{\mathsf{zt}}{}^* \ldots \rightarrow^* s_m \xrightarrow{t_m} \cdot \xrightarrow{E_{a_m}} s'_m \xrightarrow{\mathsf{zt}}{}^* s_{m+1}$$

  Note that all edges in the path from $s_{\mathsf{ini}} \rightarrow s_{k+1}$ are in $\mathcal{A}_u, \mathcal{A}_{\varphi_1}$ and all edges in the path from $s_{k+1} \rightarrow s_{m+1}$ are in $\mathcal{A}_v, \mathcal{A}_{\varphi_2}$. In particular, edge $e' = (l', \alpha, \phi, \psi, r, l_f)$ has been obtained from $e = (l', \alpha, \phi, \psi, r, l_f) \in E_{\varphi_1}$, $l_f \in L^f_{\varphi_1}$ and r resets all clocks and variables for $\mathcal{A}_v, \mathcal{A}_{\varphi_2}$. Therefore, we can construct the following accepting run ρ^1 from ρ for $\mathcal{A}_u, \mathcal{A}_{\varphi_1}$

$$s^1_{\mathsf{ini}} \xrightarrow{\mathsf{zt}}{}^* s^1_0 \xrightarrow{t_0} \cdot \xrightarrow{E_{a_0}} s^{1'}_0 \xrightarrow{\mathsf{zt}}{}^* \ldots \rightarrow^* s^1_k \xrightarrow{t_k} \cdot \xrightarrow{E_{a_k}} s^{1'}_k \xrightarrow{\mathsf{zt}}{}^* s^1_k \xrightarrow{e} s^1_{k+1}$$

  where $\rho^1_i = \rho_i|_{\mathcal{A}_u, \mathcal{A}_{\varphi_1}}$ for $i < |\rho^1|$ and accepting run ρ^2 for $\mathcal{A}_v, \mathcal{A}_{\varphi_2}$

$$s^2_{k+1} \xrightarrow{\mathsf{zt}}{}^* s^{2'}_{k+1} \xrightarrow{t_{k+1}} \cdot \xrightarrow{E_{a_{k+1}}} s^{2''}_{k+1} \xrightarrow{\mathsf{zt}}{}^* \ldots \rightarrow^* s^2_m \xrightarrow{t_m} \cdot \xrightarrow{E_{a_m}} s^{2'}_m \xrightarrow{\mathsf{zt}}{}^* s^2_{m+1}$$

  Note that $s_{k+1}|_{\mathcal{A}_v, \mathcal{A}_{\varphi_2}} = s^2_{k+1} = s^2_{\mathsf{ini}}$ because edge e' resets all clocks and variables for $\mathcal{A}_v, \mathcal{A}_{\varphi_2}$. In particular we have $\rho^2_i = \rho_{k+1+i}|_{\mathcal{A}_v, \mathcal{A}_{\varphi_2}}$ for $0 \leq i < |\rho^2|$. Since ρ^1 and ρ^2 are accepting by I.H. we have $u \in [\![\varphi_1]\!]$ and $v \in [\![\varphi_2]\!]$. Therefore, $w = u \cdot v \in [\![\varphi_1]\!] \cdot [\![\varphi_2]\!]$.
- $\varphi \equiv \varphi_1 \cup \varphi_2$. By assumption we have exists accepting run ρ in $\mathcal{A}_w, \mathcal{A}_\varphi$ (with edges E) with $\rho_0 \xrightarrow{e} \rho_1 \xrightarrow{u}{}^* \rho_f$ where $u \in (\mathbb{R}_{>0} \cup 2^E)^*$. W.l.o.g. let $e = (l, \tau, \mathsf{true}, \mathsf{true}, [], l^{\mathsf{ini}}_1)$ then we can construct accepting run ρ^1 for $\mathcal{A}_w, \mathcal{A}_{\varphi_1}$ from ρ as follows: $\rho^1 \equiv \rho_1|_{\mathcal{A}_w, \mathcal{A}_{\varphi_1}} \xrightarrow{u}{}^* \rho_f|_{\mathcal{A}_w, \mathcal{A}_{\varphi_1}}$. Since ρ^1 is accepting in $\mathcal{A}_w, \mathcal{A}_{\varphi_1}$. Note that $\rho_1|_{\mathcal{A}_w, \mathcal{A}_{\varphi_1}} = s^1_{\mathsf{ini}}$. By I.H. we have $w \in [\![\varphi_1]\!]$ as desired.
- $\varphi \equiv \varphi_1 \cap \varphi_2$. By assumption we have exists accepting run ρ in $\mathcal{A}_w, \mathcal{A}_{\varphi_1}, \mathcal{A}_{\varphi_2}$. We show that for any ρ_i with $0 \leq i \leq |\rho|$ there exist ρ^1 and j such that $\rho_i|_{\mathcal{A}_w, \mathcal{A}_{\varphi_1}} = \rho^1_j$. Base case $i = 0$ then clearly $\rho_0|_{\mathcal{A}_w, \mathcal{A}_{\varphi_1}} = \rho^1_0 = s^1_{\mathsf{ini}}$. Inductive step $i + 1$. We have $\rho \rightarrow^* \rho_i = ((l^0, l^1, l^2), \mu, \nu) \xrightarrow{\lambda_i} ((l^{0'}, l^{1'}, l^{2'}), \mu', \nu')$. By I.H. we have $\rho^1 \rightarrow^* \rho^1_j$ s.t. $\rho_i|_{\mathcal{A}_w, \mathcal{A}_{\varphi_1}} = \rho^1_j$. We have the following cases for λ_i
  - $\lambda_i \in \mathbb{R}_{>0}$ then clearly $\mu + \lambda_i$ models the invariants $I(l^0)$ and $I(l^1)$. Therefore we have $\rho^1 \rightarrow^* \rho^1_j \xrightarrow{\lambda_i} \rho^1_{j+1}$ with $\rho_{i+1}|_{\mathcal{A}_w, \mathcal{A}_{\varphi_1}} = \rho^1_{j+1}$.
  - $\lambda_i \in E_1$ we have $\rho^1 \rightarrow^* \rho^1_j \xrightarrow{\lambda_i} \rho^1_{j+1}$ with $\rho_{i+1}|_{\mathcal{A}_w, \mathcal{A}_{\varphi_1}} = \rho^1_{j+1}$.
  - $\lambda_i \in E_2$ then there is no need to extend ρ^1 and note that since $\mathcal{A}_2$ does not share variables or clocks with $\mathcal{A}_1$ and only reads variables from $\mathcal{A}_w$ we have $\rho_{i+1}|_{\mathcal{A}_w, \mathcal{A}_{\varphi_1}} = \rho_i|_{\mathcal{A}_w, \mathcal{A}_{\varphi_1}} = \rho^1_j$.
  - $\lambda_i \subseteq \{e^0, e^1, e^2\}$ is a broad cast synchronization with $e^k \in E_k$ for $0 \leq k \leq 2$. Note that e_0 needs to be in λ_i otherwise there is no sender. Note that the case where e_2 is not participating is subsumed with this case. Also note that e_1 and e_2 can not enable or disable each other. We have $\rho^1 \rightarrow^* \rho^1_j \xrightarrow{\{e^0, e^1\}} \rho^1_{j+1}$ with $\rho_{i+1}|_{\mathcal{A}_w, \mathcal{A}_{\varphi_1}} = \rho^1_{j+1}$.

Note that the case for accepting run ρ^2 is symmetric. It follows that we can use ρ to construct accepting runs ρ^i for $\mathcal{A}_w, \mathcal{A}_{\varphi_i}$ for $1 \leq i \leq 2$. By I.H. we have $w \in [\![\varphi_1]\!]$ and $w \in [\![\varphi_2]\!]$. Therefore, $w \in [\![\varphi_1 \cap \varphi_2]\!]$.

- $\varphi \equiv \varphi_1^+$. Let $w = u_1 \cdot \ldots \cdot u_i$ for $1 \leq i$. By assumption there is accepting run ρ in $\mathcal{A}_w, \mathcal{A}_\varphi$ induced by w of the following shape, $\rho_0 \rightarrow^* \rho_1 \xrightarrow{e_1} \rho_{u_1} \rightarrow^* \rho_2 \xrightarrow{e_2} \rho_{u_2} \ldots \rightarrow^* \rho_2 \xrightarrow{e_i} \rho_{u_i}$ where every $e_k \in E_\varphi$ with $e_k = (l', \alpha, \phi, \psi, r', l_{\mathsf{ini}})$ has been obtained from $e'_k = (l', \alpha, \phi, \psi, r, l_f) \in E_{\varphi_1}$ with $l_f \in L^f_{\varphi_1}$. Lets consider the prefix induced by u_1 i.e. $\rho_0 \rightarrow^* \rho_1 \xrightarrow{e_1} \rho_{u_1} = ((l_{|u_1|}, l^1_{\mathsf{ini}}), \mu, \nu)$ then the run $\rho_0 \rightarrow^* \rho_1 \xrightarrow{e'_1} ((l_{|u_1|}, l_f), \mu', \nu')$ is accepting in $\mathcal{A}_{u_1}, \mathcal{A}_{\varphi_1}$. By I.H. we have $u_1 \in [\![\varphi_1]\!]$. We continue by showing how to construct accepting run for u_2 $\mathcal{A}_{u_2}, \mathcal{A}_{\varphi_1}$. From $\rho_{u_1} \xrightarrow{v}{}^* \rho_2 \xrightarrow{e_2} \rho_{u_2}$ construct $((l_0, l^1_{\mathsf{ini}}), \mu_{\mathsf{ini}}, \nu_{\mathsf{ini}}) \xrightarrow{v}{}^* \rho'_2 \xrightarrow{e'_2} \rho'_{u_2}$. Note that $\mu(x) = \mu_{\mathsf{ini}}(x)$ for $x \in X_{\varphi_1}$ because e_1 resets all clocks in X_{φ_1}. However, this is not the case for ν. Lemma 3 ensures that there will be no variable guards until the next synchronization from $\mathcal{A}_{u_2}$ which make variable valuations agree. Therefore, we can execute the transitions in v. By I.H. we have $u_2 \in [\![\varphi_1]\!]$. We can repeat this argument i times to obtain $u_1 \in [\![\varphi_1]\!], \ldots, u_i \in [\![\varphi_1]\!]$ and $w = u_1 \cdot \ldots \cdot u_i \in \underbrace{[\![\varphi_1]\!] \cdot \ldots \cdot [\![\varphi_1]\!]}_{i \text{ times}}$. □

Proof (Theorem 2).

- Termination. We are given a finite word w of size m and a discrete state space (time is discrete). Note that Algorithm 2 is the standard fix point computation for model checking reachability properties. The only difference is that instead of returning that a property was satisfied, it collects and returns accepting states. Therefore Algorithm 2 terminates and returns a finite number of states. Finally, Algorithm 1 will iterate at most $\mathcal{O}(m * |W|)$ where $|W|$ is the biggest size of a list returned by Algorithm 2.
- If Algorithm 1 returns $k \geq 0$ then $w_{[0:k]} \in [\![\varphi]\!]$. The the algorithm traverses a witness accepting run ρ. It is easy to conclude that the form of ρ agrees with Lemma 2. Since ρ is accepting, Lemma 5 ensures that $w \in [\![\varphi]\!]$.
- If $w_{[0:k]} \in [\![\varphi]\!]$ then Algorithm 1 returns $k \geq 0$. Let $u = w_{0:k}$ By Lemma 4 we have that exits accepting run ρ in $\mathcal{A}_u, \mathcal{A}_1, \ldots, \mathcal{A}_n$. By Lemma 2 we have that ρ is of the form

$$s_{\mathsf{ini}} \xrightarrow[\mathsf{zt}]{}{}^* s_0 \xrightarrow{t_0} \cdot \xrightarrow{E_{a_0}} s'_0 \xrightarrow[\mathsf{zt}]{}{}^* s_1 \xrightarrow{t_1} \cdot \xrightarrow{E_{a_1}} s'_1 \xrightarrow[\mathsf{zt}]{}{}^* \ldots \rightarrow^* s_k \xrightarrow{t_k} \cdot \xrightarrow{E_{a_k}} s'_k \xrightarrow[\mathsf{zt}]{}{}^* s_f$$

  We continue by induction on the length i of ρ. As I.H. have that Algorithm 1 can find state ρ_i. For simplicity let us consider the case where Urgent Partial Order Reduction (UPOR) is disabled. i.e. for every state s we have $\mathsf{En}(s) \subseteq \mathsf{St}(s)$. However, note that the results will hold because UPOR preserves states which can delay and accepting states can delay.
  - Base case $i = 0$. Then $\rho_0 = s_{\mathsf{ini}}$ and we are done.
  - Inductive step $i + 1$. By I.H. we have that Algorithm 1 can reach state $\rho_0 \rightarrow^* \rho_i$. Need to show $\rho_i \xrightarrow{\lambda_i} \rho_{i+1}$.

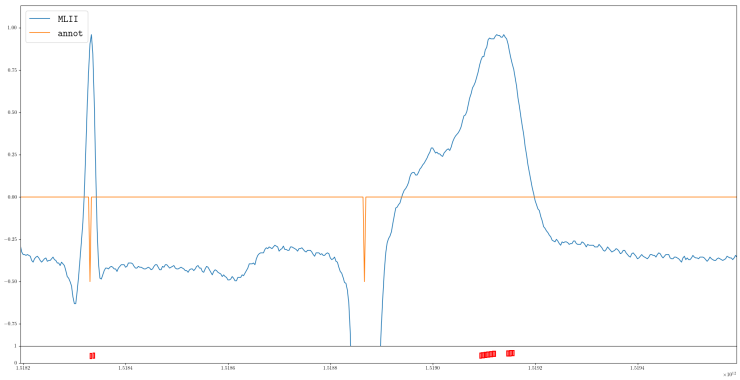


Fig. 6: Multiple matches using ETRE qrs-const

* $\lambda_i \in \mathbb{R}_{>0}$. Then Algorithm 1 Line 8 is executed and it can reach ρ_{i+1}.
* $\lambda_i \in 2^E$ and $|\lambda_i| > 1$ with $1 \leq j \leq n$. Then Algorithm 1 Line 8 is executed and it can reach ρ_{i+1}.
* $\lambda_i \in E_i$ for $1 \leq i \leq n$. Then ρ_i is urgent and Algorithm 2 Line 7 can reach ρ_{i+1}.

Since s_f is accepting and it can be reached, at k-th iteration Algorithm 1 Line 7 returns value k. □

B Complement for Case Studies

The following table shows the data used for our evaluation section.

Data	Dimensions	Size,MB	Samples	Annotations
Arrhythmia	3	60	650000	2274
Storm-safe	19	50	764572	–
TMV 23	13	23	23360	–

Table 3: Data

B.1 Arrhythmia

Figure 7 shows the automaton corresponding to the ETRE $\mathsf{qrs\text{-}func}_\delta \cdot \langle \Sigma^* \rangle_{[0.47s,0.67s]} \cdot \mathsf{qrs\text{-}func}_\delta$. Locations 0 to 2 correspond to $\mathsf{qrs\text{-}func}_\delta$ locations 6 to 8 correspond

to qrs-func$_\delta$. At location 2 the construction for $\langle \Sigma^* \rangle_{[0.47s,0.67s]}$ starts. This is the union of the empty word locations 3 and 6 with Σ^+ constraint to interval $[0.47s, 0.67s]$ locations 4,5 and 6.

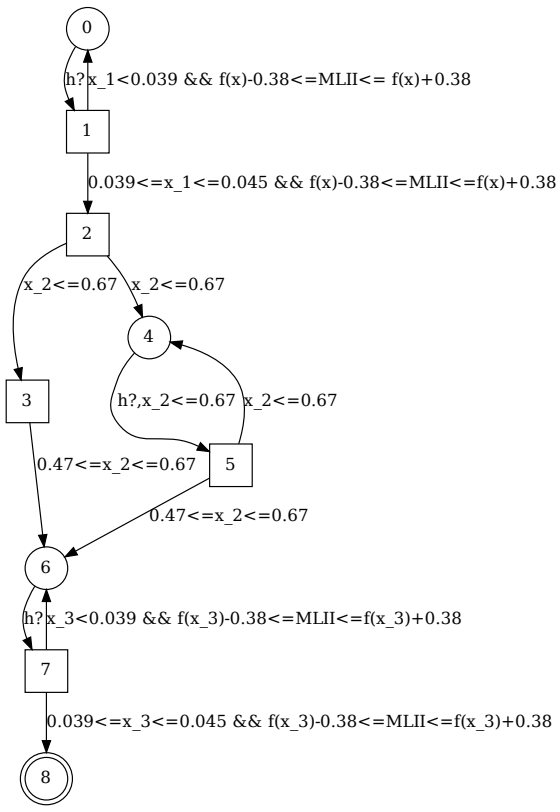


Fig. 7: Automaton for ETRE early-func$_{0.38}$. Square locations denote urgent locations.

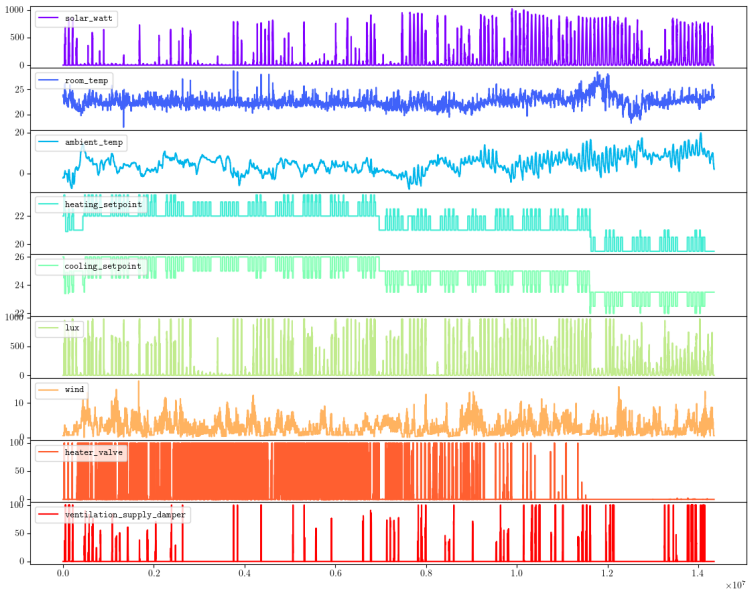
B.2 Storm Surge Barriers

Figure 8 shows a match for sto-close-diff. A violation to the requirement that the gates should be open only if the difference on the levels is below 100.

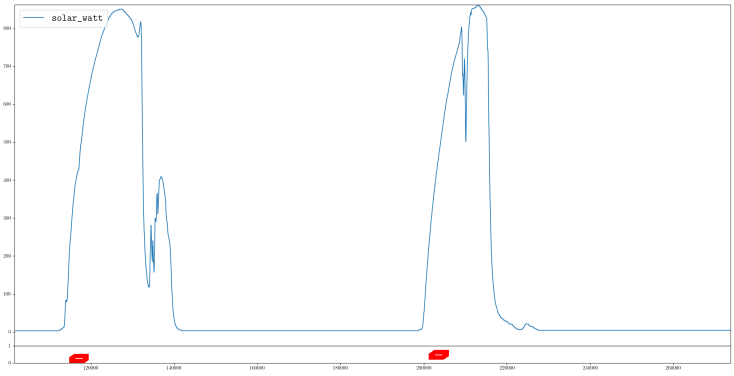
Fig. 8: Storm Surge Barriers. Match for ETRE sto-close-diff. Matching a violation when a gate is open (port_sum) when the difference in water levels is bigger than 100.

B.3 Smart Buildings

Figures 9a shows the data collected from from 2024-11-20 to 2025-05-04. Figure 9b shows matches using a first degree polynomial to detect increase in the solar watt signal.



(a) Data from building TMV 23. from 2024-11-20 to 2025-05-04 in seconds.



(b) Matching solar watt is increasing within 45min. using first degree polynomial.

Fig. 9: Pattern Matching ETRE for building TMV 23.