Timed Pattern Matching

Dogan Ulus[1], Thomas Ferrère[1], Eugene Asarin[2], and Oded Maler[1]

[1] VERIMAG, CNRS and the University of Grenoble-Alpes, France
[2] LIAFA, Université Paris Diderot / CNRS, Paris, France

Abstract. Given a timed regular expression and a dense-time Boolean signal we compute the set of all matches of the expression in the signal, that is, the set of all segments of the signal that satisfy the regular expression. The set of matches is viewed as a set of points in a two-dimensional space with each point indicating the beginning and end of a matching segment on the real time axis. Our procedure, which works by induction on the structure of the expression, is based on the following result that we prove in this paper: the set of all matches of a timed regular expression by a signal of finite variability and duration can be written as a finite union of zones.

1 Introduction

Pattern matching, the determination of all sub-sequences of a string of symbols that match some pre-specified pattern, is a fundamental operation in searching over texts and elsewhere. Pattern matching has been studied extensively for textual data starting from the 60s. The basic string matching algorithms for single words [KMP77,BM77] as well as specialized data structures such as suffix trees [Wei73] and later advancements can be found in [Ste94,CR02]. For more complex patterns, the classical regular expressions of [Kle56] is an adequate pattern description language supporting a minimal set of features (concatenation, alternation and repetition). It has been enhanced over the years by various features such as anchors, character sets and any-character [Fri06]. Pattern matching based on variants of regular expressions is implemented in many software tools ranging from the *grep* [Tho68] family to regular expression modules of modern programming languages, notably Perl and Python. Besides texts, pattern matching has important applications in Biology (DNA and protein searches) [AGM$^+$90] and in database querying (especially temporal databases [FRM94]).

In this work we introduce a quantitative-time variant of the pattern matching problem where discrete sequences are replaced by dense-time, discrete-valued signals. As a pattern specification formalism we use a variant of the *timed regular expressions* of [ACM02] which are expressively related, in terms of timed languages, to the *timed automata* of [AD94]. We provide a complete solution to the timed pattern matching problem defined as: find *all* sub-segments of the signal that match the expression. Note that a straightforward application of the classical translations of regular expressions to automata can be used to detect whether the *prefix* of a string matches the pattern. The classical algorithm of Thompson [Tho68] adapted the automaton construction for the matching context but still, the discrete case, finding *all* matches of an expression in a

string is considered a very difficult problem and is not part of the mainstream (some exceptions are [Pik87] and [Lau00]). One reason might be that without a symbolic representation, which is necessary for the timed case, the set of matches may be prohibitively large to represent.

In addition to the theoretical interest, we believe that the problem of finding patterns in real-time data has numerous applications in many domains. This particular work was triggered by assertion-based circuit (dynamic) verification which is the hardware equivalent of what is called runtime verification in software. This form of lightweight verification consists in monitoring simulation traces against temporal specifications. Monitoring procedures for *temporal logic* formulas are well-studied and have been extended successfully to real-time and analog signals [MN04,MNP08]. However, standard assertion languages used in the semi-conductor industry such as PSL [EF06,CVK04] and SVA [VR06,Spe06] combine temporal logic with regular expressions in a non trivial way. The results of this paper can be used to extend monitoring procedures toward such specification languages and their timed extensions such as the one proposed in [HL11].

To give an intuition of what we do, consider the expression $\varphi := \langle (p \wedge q) \cdot \bar{q} \cdot q \rangle_{[4,5]} \cdot \bar{p}$ whose verbal description is as follows. Inside a time window of a duration between 4 and 5 there exists an interval where both p and q are high, followed by q going down and up again; after that time window there is an interval where p is low. If we look at signals p and q plotted in Fig. 1-(a), we can see φ is matched by any time intervals $[t, t']$ such that $t \in [1, 2]$ and $t' \in [6, 7]$. Clearly, the number of such segments of the signal (the *matches*) is infinite and two of them, $[1.3, 6.8]$ and $[1.5, 7.0]$ are shown in Fig. 1-(b).

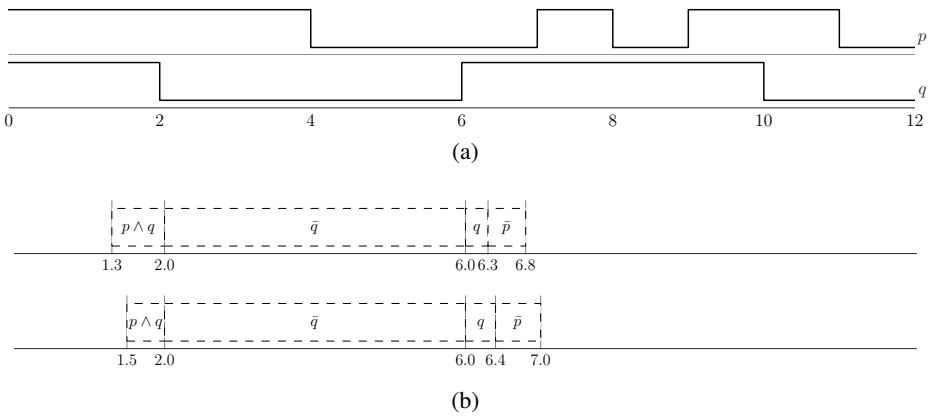


Fig. 1. (a) Boolean signals p and q; (b) Intervals $[1.3, 6.8]$ and $[1.5, 7.0]$ are two possible matches that satisfy φ over p and q.

Technically, our contribution is based on the following result. Let w be a Boolean signal defined over an interval $[0, d]$, let its restriction to the temporal interval $[t, t']$ be

denoted by $w[t,t']$ and let φ be a timed regular expression. Then, the set of matches for φ in w,

$$\mathcal{M}(\varphi, w) = \{(t,t') : w[t,t'] \in [\![\varphi]\!]\}$$

is a finite union of zones. Zones are a special class of convex polytopes definable by intersections of inequalities of the form $c_1 \leq x_i \leq c_2$ and $c_1 \leq x_i - x_j \leq c_2$. They are used extensively in the verification of timed automata and admit a data-structure (difference-bound matrices, DBM) on which various operations, including all those required by the recursive computation of $\mathcal{M}(\varphi, w)$, can be carried out efficiently.

The rest of the paper is organized as follows. Section 2 defines the syntax and semantics of timed regular expressions. Section 3 illustrates the zone-based decomposition of match-sets, states the main result that is the finiteness of such a decomposition, and proves it by showing the following lemma: for every signal w of finite variability and expression φ there exist some k such that $\mathcal{M}(\varphi^*, w) = \mathcal{M}(\varphi^{\leq k}, w)$. Section 4 gives more details about the implementation of our algorithm with a practical bound on the convergence to a fixed point for φ^*. Section 5 reports the performance of the algorithm on several families of examples and is followed by a discussion of future work.

2 Timed Regular Expressions over Signals

Signals are the dense-time analogues of sequences, functions from a time domain into a value domain. We will work with Boolean signals but the results can be easily extended to any discrete value domain.

Definition 1 (Boolean Signals). *Let $\mathbb{T} = [0,d]$ be a bounded interval of $\mathbb{R}_+$ and let m be a positive integer. A* Boolean signal *is a function $w : \mathbb{T} \to \mathbb{B}^m$.*

We use $w[t]$ to denote the value of the signal at time t. An interval $I \subseteq [0,d]$ is *uniform* with respect to w if $w[t] = w[t']$ for every $t, t' \in I$. A maximally uniform interval is a uniform interval such that any interval strictly containing it is not uniform. We focus on non-Zeno signals of bounded duration which thus have a finite number of maximally-uniform intervals. We use $w[t,t']$ to denote the segment of w on the interval $[t,t']$.
Remark: To keep the fluidity of the presentation we do not give too much attention to the issue of open/closed intervals in the definitions of signals and expressions. In fact, the semantics of timed regular expressions in [ACM02] is not based on total functions from $\mathbb{T}$ to the alphabet. An element from the underlying signal monoid is written as $w = p^5 \cdot \bar{p}^3$ which means 5 time of p followed by 3 time of $\bar{p}$. As a function, it is clear that $w[t] = 1$ when $t \in (0,5)$ and $w[t] = 0$ when $t \in (5,8)$. However concerning its value at points 0, 5 and 8 there are different schools of thought. One possibility is to let $w = 1$ at $[0,5)$, $w = 0$ at $[5,8)$ and undefined elsewhere. Another possibility is to consider the value of the signal be non-deterministic or undefined at boundary points. In this paper our regular expressions semantics simply ignore the value of signal w at boundary points.

As a pattern specification language we use a variant of the timed regular expression of [ACM02]. Such expressions admit, in addition to the standard *concatenation*,

union and *star*, also *intersection*, *time restriction* and *renaming* (we do not use the latter operator which was introduced to match the full expressive power of timed automata).

Definition 2 (Timed Regular Expressions). *The syntax of timed regular expressions is given by the grammar*

$$\varphi := \epsilon \mid p \mid \overline{p} \mid \varphi \cdot \varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \varphi^* \mid \langle \varphi \rangle_I$$

where $p \in \{p_1, \ldots, p_m\}$ is a propositional variable and I is an interval of $\mathbb{R}_+$ with integer endpoints.

We use an exponent notation with $\varphi^0 = \epsilon$, $\varphi^{k+1} = \varphi^k \cdot \varphi$, $\varphi^{<1} = \epsilon$ and $\varphi^{<k+1} = \varphi^{<k} \vee \varphi^k$. Note that the expressions used in the paper are more "symbolic" than in the standard "flat" theory of regular expression and closer in spirit to temporal logic where an atomic expression like p_i denotes (or can be viewed as a syntactic sugar for) the set of all tuples in $\mathbb{B}^m$ whose i^{th} coordinate is 1. The identity of signal w should always be clear from context, so that we use $p_i[t]$ as a shorthand for the value at time t of the projection of w on propositional variable p_i.

Typically, the semantics of real-time temporal logics such as MTL [Koy90] and MITL [AFH96] is expressed in terms of a satisfaction relation of the form $(w, t) \models \varphi$, indicating that the signal w satisfies φ *from* position t. For the past fragment of a temporal logic this is a statement about $w[0, t]$ while for the future fragment it is a statement about $w[t, d]$, see [MNP05]. For regular expressions we found necessary to parameterize the satisfaction relation by two time points $t \leq t'$, as concatenation requires equality between the *end* time of its left argument with the *begin* time of its right argument. In this setting we note $(w, t, t') \models \varphi$ the fact that $w[t, t']$ is part of the semantics of expression φ. Such a satisfaction relation also appears in extensions to real-time temporal logics such as freeze quantification which has been shown in [DBS12] to be monitorable by working directly in $\mathbb{R}^2$.

Definition 3 (Semantics). *The satisfaction relation $\models$ of a timed regular expression φ by a signal w, relative to start time t and end time $t' \geq t$ is defined as follows:*

$$\begin{array}{ll}
(w,t,t') \models \epsilon & \leftrightarrow t = t' \\
(w,t,t') \models p & \leftrightarrow t < t' \text{ and } \forall t''.\, t < t'' < t' \rightarrow p[t''] = 1 \\
(w,t,t') \models \overline{p} & \leftrightarrow t < t' \text{ and } \forall t''.\, t < t'' < t' \rightarrow p[t''] = 0 \\
(w,t,t') \models \varphi \cdot \psi & \leftrightarrow \exists t''.\, (w,t,t'') \models \varphi \text{ and } (w,t'',t') \models \psi \\
(w,t,t') \models \varphi \vee \psi & \leftrightarrow (w,t,t') \models \varphi \text{ or } (w,t,t') \models \psi \\
(w,t,t') \models \varphi \wedge \psi & \leftrightarrow (w,t,t') \models \varphi \text{ and } (w,t,t') \models \psi \\
(w,t,t') \models \varphi^* & \leftrightarrow \exists k \geq 0.\, (w,t,t') \models \varphi^k \\
(w,t,t') \models \langle \varphi \rangle_I & \leftrightarrow t' - t \in I \text{ and } (w,t,t') \models \varphi
\end{array}$$

The set of segments of w that match an expression φ is captured by the match-set.

Definition 4 (Match-Set). *For any signal w and expression φ, we let*

$$\mathcal{M}(\varphi, w) := \{(t, t') \in \mathbb{T} \times \mathbb{T} : (w, t, t') \models \varphi\}$$

Geometrically speaking, match-sets are subsets of $[0, d] \times [0, d]$ confined to the upper triangle defined by $t \leq t'$. In the sequel we show constructively that for every timed regular expression φ and a finite-variability signal w, the match-set can be written as a finite union of zones.

3 Match-Sets and Zones

Zones constitute a restricted class of convex polyhedra defined by orthogonal constraints $c \prec t_i$ and difference constraints $c \prec t_i - t_j$ with $\prec \in \{<, \leq, \geq, >\}$. They are used extensively to represent clock values in the analysis of timed automata. In this paper we use them to represent absolute time values in a match-set. While the constants for the diagonal constraints may be taken as integers, those of the orthogonal constraints have fractional parts inherited from the time stamps of events in w. Such a zone z also lies in the upper quadrant of $\mathbb{R}^2$ and not below the diagonal. Consequently we let $\pi_1(z)$, $\pi_2(z)$ and $\delta(z)$ be its vertical, horizontal and diagonal projections; these are intervals with respective endpoints noted $\pi_1^-(z)$, $\pi_1^+(z)$, $\pi_2^-(z)$, $\pi_2^+(z)$, $\delta^-(z)$, and $\delta^+(z)$. Typically we have

$$(t_1, t_2) \in z \quad \leftrightarrow \quad \begin{cases} \pi_1^-(z) \leq \quad t_1 \quad \leq \pi_1^+(z) \\ \pi_2^-(z) \leq \quad t_2 \quad \leq \pi_2^+(z) \\ \delta^-(z) \leq t_2 - t_1 \leq \delta^+(z) \end{cases}$$

with all constraints being tight. Note that under this representation diagonal constraints may no longer be integers. Due to the positive duration constraint for atomic predicates we also have to consider zones that are partly open, with the same canonical representation yet featuring strict inequalities.

Below we explain how the match-set of an expression is inductively constructed and prove that the outcome is always a finite union of zones. Since operations $\wedge$, $\langle . \rangle_I$ and $\cdot$ distribute over union, it is sufficient to prove closure of zones under their associated match-set operations and the closure of unions of zones will immediately follow. The closure is almost immediate for all cases except φ^* where we will compute the match-set by a finite number of concatenation. The convergence to a fixed point $\mathcal{M}(w, \varphi^{\leq k+1}) = \mathcal{M}(w, \varphi^{\leq k})$ can be intuitively understood due the finite number of zones definable using a finite number of constants in the constraints. One may show that non-redundant constraints, as opposed to tight ones are either integers or have a fractional part equal to that of some event in w. However a bound obtained from an argument along these lines would be overly pessimistic and we will use other proofs to compute better bounds on the number of iterations.

Empty word Just notice that the match-set of ϵ is the diagonal zone

$$\mathcal{M}(\epsilon, w) = \{(t, t') \in \mathbb{T} \times \mathbb{T} : t = t'\}.$$

Literals When φ is a literal (p or $\bar{p}$) the match-set is a disjoint union of triangles touching the diagonal whose number depends on the number of switching points of the projection of w on p, see Fig. 2-(a).

Boolean Operations The match-sets for disjunction and conjunction satisfy

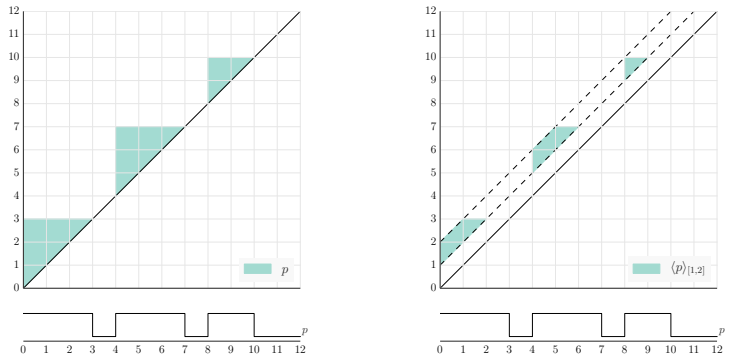
$$\mathcal{M}(\varphi \vee \psi, w) = \mathcal{M}(\varphi, w) \cup \mathcal{M}(\psi, w) \quad \text{and} \quad \mathcal{M}(\varphi \wedge \psi, w) = \mathcal{M}(\varphi, w) \cap \mathcal{M}(\psi, w)$$

and finite unions of zones are closed under Boolean operations.

Time Restriction The match-set of the time restriction of an expression is obtained by intersecting the match-set with the corresponding diagonal band, that is,

$$\mathcal{M}(\langle\varphi\rangle_I, w) = \mathcal{M}(\varphi) \cap \{(t, t') : t' - t \in I\}.$$

This is just an intersection with a zone and the result remains a union of zones, see Fig. 2-(b).



(a) (b)

Fig. 2. (a) The match-set of a signal with respect to an atomic expression; (b) The effect of time restriction.

Concatenation Viewing $\mathcal{M}$ as a binary relation, the match-set for concatenation is nothing but a relational composition of the corresponding match-sets:

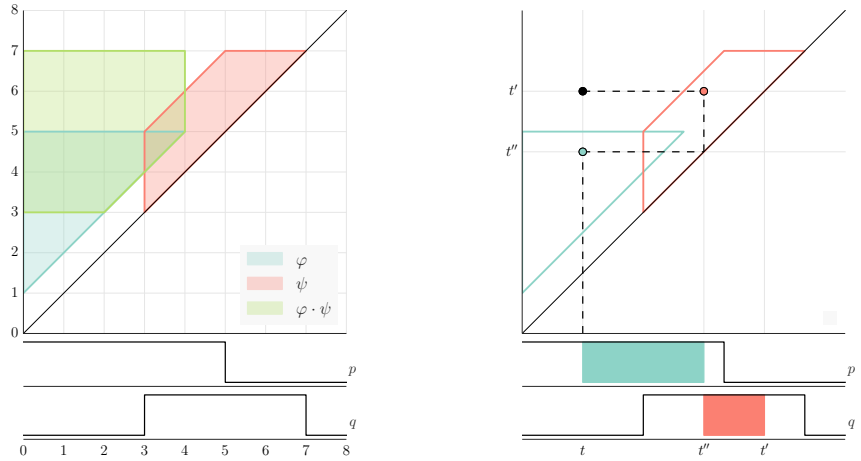
$$\mathcal{M}(\varphi \cdot \psi, w) = \mathcal{M}(\varphi, w) \circ \mathcal{M}(\psi, w)$$

as illustrated in Fig. 3-(a).

Lemma 1. *The composition of two zones is a zone.*

Proof. Let $F[t, t']$ and $G[t, t']$ be conjunctions of difference constraints defining zones z and z' respectively. Their composition $z'' = z \circ z'$ is defined by the formula $H := \exists t''.F[t, t''] \wedge G[t'', t']$. Eliminating t'' from H using the Fourier-Motzkin procedure we get an equivalent, quantifier-free formula H' which is also a conjunction of difference constraints and hence z'' is a zone.

Geometrically speaking, $z \circ z'$ can be seen as inverse-projecting z and z' into a 3-dimensional space with axes labeled by t, t' and t'', intersecting these two sets and projecting back on the plane t, t'. Another intuition in dimension 2 is given Fig. 3-(b).



(a) (b)

Fig. 3. (a) Match-sets of expressions $\varphi := \langle p \rangle_{[1,\infty]}$, $\psi := \langle q \rangle_{[0,2]}$ and $\varphi \cdot \psi$; (b) A point $(t, t') \in z \circ z'$ corresponds to a path from t to t' via some $(t, t'') \in z$, t'' on the diagonal, and $(t'', t') \in z'$

Star We prove that the match-set of the star can be computed by a finite number of concatenations. An interval $[t, t']$ is said to be *unitary* with respect to w if $t' - t < 1$ and w is constant throughout its interior (t, t'). The following simple property of unitary intervals can be proved by a straightforward structural induction on φ.

Lemma 2. *Let $[t, t']$ be a unitary interval with respect to w. For all intervals $[r, r'] \subseteq [t, t']$ we have $(w, r, r') \models \varphi$ if and only if $(w, t, t') \models \varphi$.*

Let $\sigma(w)$ be the least k such that w can be covered by k unitary intervals, that is, there exists a sequence of intervals $[0, t_1], [t_1, t_2], ..., [t_{k-1}, d]$, all unitary with respect to w. A key property of $k = \sigma(w)$ is the following.

Lemma 3. *For any $n > 2k + 1$ if $(w, t, t') \models \varphi^n$ then $(w, t, t') \models \varphi^{n-1}$.*

Proof. Let $[0, t_1], [t_1, t_2], ..., [t_{k-1}, d]$ be a sequence of unitary intervals with respect to w. If $(w, t, t') \models \varphi^n$ then there exists a sequence of time points $t = r_0 \leq r_1 \leq \cdots \leq r_n = t'$ such that for any $i \in 1..n$

$$(w, r_{i-1}, r_i) \models \varphi \tag{1}$$

When $n > 2k + 1$, by the pigeonhole principle, among time points $r_0, \ldots, r_n$ there are three consecutive points, denoted by r_{i-1}, r_i, r_{i+1}, within the same unitary interval $[t_{j-1}, t_j]$ of w. By Lemma 2 it holds that $(w, r_{i-1}, r_{i+1}) \models \varphi$, thus the time point r_i can be excluded from $r_0, \ldots, r_n$ still preserving (1). Hence $(w, t, t') \models \varphi^{n-1}$.

Corollary 1. *For any expression φ and any signal w with $\sigma(w) = k$ it holds that $\mathcal{M}(\varphi^*, w) = \mathcal{M}(\varphi^{\leq 2k+1}, w)$.*

From all this we conclude:

Theorem 1 (Match-Sets and Unions of Zones). *Given a finite variability signal w and a timed regular expression φ, $\mathcal{M}(\varphi, w)$ is a finite union of zones.*

Fig. 4 demonstrates the whole process of computing matches by zones for the expression $\langle (p \wedge q) \cdot \bar{q} \cdot q \rangle_{[4,5]} \cdot \bar{p}$ and signal of Fig. 1-(a) from the introduction. The result is indeed the rectangle $[1, 2] \times [6, 7]$.

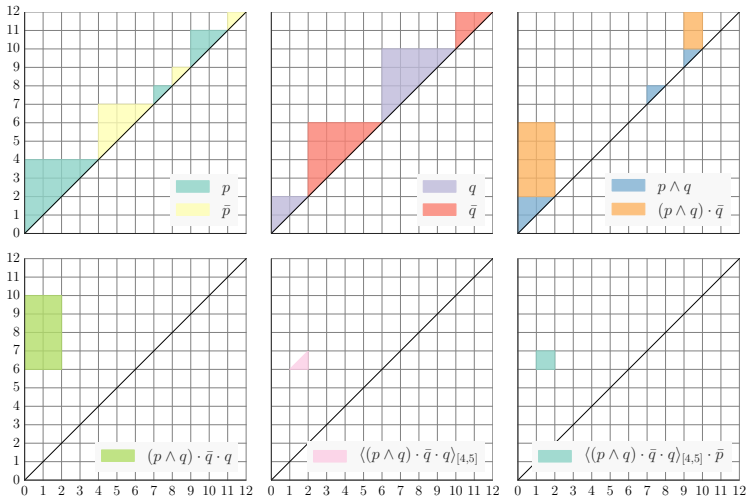


Fig. 4. The match-sets for all sub-expressions of $\langle (p \wedge q) \cdot \bar{q} \cdot q \rangle_{[4,5]} \cdot \bar{p}$ in the signal of Fig. 1-(a)

4 Computation

4.1 Algorithms and implementation

The proof of Theorem 1 gives us a procedure for computing match-sets. This procedure, sketched in Algorithm 1, recursively calls a subroutine COMBINE that takes as arguments the topmost operator of the expression $\bullet \in \{\cdot, \vee, \wedge, ^*, \langle\,\rangle_I\}$ along with the match-set(s) of the subexpression(s) and applies the operation corresponding to the specific operator $\bullet$. All the operations on individual zones, including composition of two zones, see Lemma 1, are realized using calls to the zone library of the tool IF [BGM02] from our *Python* implementation.

Algorithm 1 $\text{ZONES}(\varphi, w)$

select (φ)
case $\epsilon, p, \overline{p}$**:**
  $Z_\varphi := \text{ATOM}(\varphi, w)$
case $\bullet\, \psi$**:**
  $Z_\psi := \text{ZONES}(\psi, w)$
  $Z_\varphi := \text{COMBINE}(\bullet, Z_\psi)$
case $\psi_1 \bullet \psi_2$**:**
  $Z_{\psi_1} := \text{ZONES}(\psi_1, w)$
  $Z_{\psi_2} := \text{ZONES}(\psi_2, w)$
  $Z_\varphi := \text{COMBINE}(\bullet, Z_{\psi_1}, Z_{\psi_2})$
end select
return Z_φ

Our algorithm intensively performs various binary operations over sets of zones, that is, $Z \bullet Z' = \{z \bullet z' : z \in Z, z' \in Z'\}$. In addition to the operations defined in the expressions, in various stages we eliminate redundancy by checking *pairwise inclusion* of zones covering a match-set. For this we define a special *filtering* operation by $\downarrow Z = \{z \in Z : \forall z' \in Z.\, z \not\subset z'\}$, which consists in removing from Z all zones strictly included in other zones in Z. We define the general pairwise inclusion test $\sqsubseteq$ by $Z \sqsubseteq Z' \leftrightarrow \forall z \in Z, \exists z' \in Z'.\, z \subseteq z'$, that is each zone in Z is included in a zone of Z'. Note that the filtering Z is just taking the smallest $Z' \subseteq Z$ such that $Z \sqsubseteq Z'$.

A straightforward implementation of a binary operation on sets of zones with n elements will need $\mathcal{O}(n^2)$ operations. In practice, many of these operations yield an empty set and can be avoided by exploiting inherent ordering between zones. Such operations are very similar to the spatial join operation, studied extensively for spatial databases, see [JS07]. Spatial joins are usually performed using a *filter-and-refine* approach to avoid redundant operations, where two-dimensional objects in Euclidean space are first approximated by their *minimum bounding boxes* in a filtering stage and then actual operations are performed on filtered sets of objects. We implement this idea through the *plane-sweep* algorithm, used as is for intersection and filtering, and specialized for concatenation.

For intersection, Algorithm 2 keeps Z and Z' sorted according to π_1^-. It maintains two active lists Y and Y' consisting of candidates for intersection. Elements are successively moved to the active lists and are removed from them when it is clear they will not participate in further non-empty intersections. This happens for $z \in Y$ such that $\pi_1^+(z) < \pi_1^-(z')$ for every $z' \in Y'$ and vice versa. The filtering operation $\downarrow$ is performed using a similar algorithm. For concatenation, that is computing $Z'' = Z \circ Z'$, observe that $z \circ z' \neq \emptyset$ iff $\pi_2(z) \cap \pi_1(z') \neq \emptyset$. Hence we can apply an algorithm similar to Algorithm 2 where Z is sorted according to π_2^- and Z' is sorted according to π_1^-.

For the star operation, we tried two different approaches. In the *incremental* approach we compose the input set Z with an accumulating set Y initialized to Z. In the *squaring* approach we compose the accumulating set Y with itself. The squaring approach is more efficient for sets of zones that converge slowly to a fixpoint. Algorithm 3 depicts our implementation of this approach. In order not to compose the same

Algorithm 2 COMBINE$(\wedge, Z, Z')$ *assume Z, Z' sorted by π_1^-*

$Y := Y' := Z'' := \emptyset$
while $Z \neq \emptyset \vee Z' \neq \emptyset$ **do**
  $z :=$ first(Z); $\ell := \pi_1^-(z)$
  $z' :=$ first(Z'); $\ell' := \pi_1^-(z')$
  if $\ell < \ell'$ **then**
    Move z from Z to Y
    $Y' := \{z' \in Y' : \pi_1^+(z') \geq \ell\}$
    foreach $z' \in Y'$ **loop**
      $z'' := z \cap z'$
      $Z'' := Z'' \cup \{z''\}$
    end loop
  else
    Move z' from Z' to Y'
    $Y := \{z \in Y : \pi_1^+(z) \geq \ell'\}$
    foreach $z \in Y$ **loop**
      $z'' := z \cap z'$
      $Z'' := Z'' \cup \{z''\}$
    end loop
  end if
end while
return $\downarrow Z''$

sequence of zones twice, we maintain two sets X_k and Y_k such that at the end of iteration k we have $\cup X_k = (\cup Z)^{2^k}$ and $\cup Y_k = (\cup Z)^{<2^k}$. According to Corollary 1, we may stop at the first k such that $2^k \geq 2 \cdot \sigma(w) + 1$; however a fixpoint can be reached in less iterations. For performance reasons we use the pairwise inclusion test $X_k \sqsubseteq Y_k$ and give in the sequel an upper-bound on the number of iterations needed until this condition is met.

Algorithm 3 COMBINE$(^*, Z)$

$Y := Z$
$X :=$ COMBINE$(\cdot, Z, Z)$
while $X \not\sqsubseteq Y$ **do**
  $Y :=\downarrow(\, Y \cup X \cup$ COMBINE$(\cdot, X, Y)\,)$
  $X :=$ COMBINE$(\cdot, X, X)$
end while
return $Y \cup \{\varepsilon\}$

4.2 A bound on the number of iterations

We show that for an input set of zones Z produced by our matching procedure, having $|Z|$ elements and covering $d = \lceil \max\{\pi_2^+(z') - \pi_1^-(z) : z, z' \in Z\} \rceil$ time units, the pairwise inclusion test is met before $k = \log(|Z| + d)$ iterations.

A sequence of zones $z_1, \ldots, z_n$ is said to be *redundant* if there exists $1 \leq i < j \leq n$ with $z_1 \circ \cdots \circ z_j \subseteq z_1 \circ \cdots \circ z_i$. Note that the star algorithm eliminates redundant sequences as for any such sequence $z_1, \ldots, z_n$ we have by transitivity $z_1 \circ \cdots \circ z_n \subseteq z_1 \circ \cdots \circ z_i \circ z_{j+1} \circ \cdots \circ z_n$. We first see that in a non-redundant sequence the maximal duration never decreases.

Lemma 4. *For any z, z' such that $z \circ z' \not\subseteq z$ we have $\delta^+(z \circ z') \geq \delta^+(z)$.*

Proof. The propagation of difference constraints gives us $\delta^+(z \circ z') = \min\{\delta^+(z) + \delta^+(z'), \pi_2^+(z') - \pi_1^-(z)\}$. Suppose $\delta^+(z \circ z') < \delta^+(z)$ and show $z \circ z' \subseteq z$. First note that $\pi_1(z \circ z') \subseteq \pi_1(z)$. By hypothesis $\pi_2^+(z') - \pi_1^-(z) < \delta^+(z)$, yet $\delta^+(z) \leq \pi_2^+(z) - \pi_1^-(z)$ so that $\pi_2^+(z') < \pi_2^+(z)$. This implies that $\pi_2(z \circ z') \subseteq \pi_2(z)$. Finally the hypothesis $\delta^+(z \circ z') < \delta^+(z)$ gives us $\delta(z \circ z') \subseteq \delta(z)$.

We call *repeated* a position i in the sequence $z_1, \ldots, z_n$ such that there exists $j > i$ with $z_i = z_j$. Now when appending a zone that may be repeated, the maximal duration increases by the corresponding amount.

Lemma 5. *For any z, z' such that there exists z'' with $z \circ z' \circ z'' \circ z' \not\subseteq z \circ z'$ we have $\delta^+(z \circ z') = \delta^+(z) + \delta^+(z')$.*

Proof. Suppose $\delta^+(z \circ z') < \delta^+(z) + \delta^+(z')$, take z'' a zone and show $z \circ z' \circ z'' \circ z' \subseteq z \circ z'$. Similarly to the proof of Lemma 4 it is sufficient to show that π_2^+ and δ^+ do not increase. On the one hand $\pi_2^+(z \circ z' \circ z'' \circ z') \leq \pi_2^+(z') = \pi_2^+(z \circ z')$, and on the other hand $\delta^+(z \circ z' \circ z'' \circ z') \leq \pi_2^+(z') - \pi_1^-(z) = \delta^+(z \circ z')$.

By a straightforward induction on the expression one may show that a zone z' such that $\delta^+(z') < 1$ always verifies $z' = \pi_1(z') \times \pi_2(z') \cap \{(t, t') : t' - t > 0\}$. Thus if such a zone z' was repeated it would make the corresponding sequence redundant; under the conditions of Lemma 5 we indeed have $\delta^+(z \circ z') \geq \delta^+(z) + 1$.

Theorem 2. *Let Z be a set of zones covering d time units; Algorithm 3 stops within $k = \log(|Z| + d)$ iterations.*

Proof. We first show that any non-redundant sequence of zones $z_1, \ldots, z_n$ with m repetitions verifies $\delta^+(z_1 \circ \cdots \circ z_n) \geq m$.
Let i be a position in the sequence. If z_i is repeated there exists $j > i$ with $z_i = z_j$. Factoring the composition of $z_1, \ldots, z_j$ into $(z_1 \circ \cdots \circ z_{i-1}) \circ z_i \circ (z_{i+1} \circ \cdots \circ z_{j-1}) \circ z_j$ we see by Lemma 5 that $\delta^+(z_1 \circ \cdots \circ z_i) = \delta^+(z_1 \circ \cdots \circ z_{i-1}) + \delta^+(z_i)$ and in particular $\delta^+(z_1 \circ \cdots \circ z_i) \geq \delta^+(z_1 \circ \cdots \circ z_{i-1}) + 1$, maximal duration increases of 1. Else z_i is not repeated and by Lemma 4 we ensure $\delta^+(z_1 \circ \cdots \circ z_i) \geq \delta^+(z_1 \circ \cdots \circ z_{i-1})$, maximal duration does not decrease. With m repeated zones the sequence $z_1, \ldots, z_n$ has maximal duration of at least m.
Now suppose the algorithm reaches iteration k, and take x a zone in X_k. There exists a sequence $z_1, \ldots, z_n$ such that $x = z_1 \circ \cdots \circ z_n$ with $n = 2^k \geq |Z| - d$. If such a sequence was non-redundant it would have at least $n - |Z|$ repeated zones, and maximal duration $\delta^+(z_1 \circ \cdots \circ z_n) \geq n - |Z| \geq d > \pi_2^+(z_n) - \pi_1^-(z_1)$ which is impossible. Therefore it is redundant so that there exists $y \in Y_k$ with $x \subseteq y$; we have shown $X_k \sqsubseteq Y_k$ thus if iteration k is reached the algorithm stops.

5 Experimentation

We test our implementation on several examples, focusing on performance aspects and investigating the sensitivity of our algorithms to various parameters such as signal variability, signal length, and the magnitude of time units in the expression.

Example: Performance of the concatenation In this example we define an expression $\varphi := p \cdot q$ on random Boolean signals p and q with length $\mathcal{L}$. We draw a number $\mathcal{E}$ of switching points from a uniform distribution between 0 and $\mathcal{L}$, and we define the variability of the resulting signal as $\mathcal{V} = \mathcal{E}/\mathcal{L}$. In Fig. 5-(a), we evaluate expression φ over such signals w with fixed variability and of increasing length to measure the execution time. This corresponds to the usual situation when monitoring various executions of the same system: there the algorithm performs in linear time with respect to signal length. Note that we use similar algorithms for concatenation, union, intersection and time-restriction operations; therefore, one can extend performance behavior of concatenation to others.

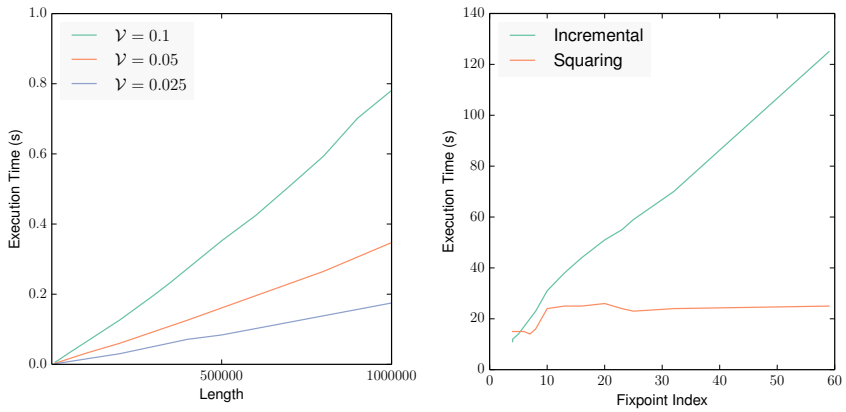


Fig. 5. (a) Performance of concatenation operation with respect to signal length; (b) Performance comparison of star algorithms with respect to fixpoint index

Example: Performance of the star In this example we use a family of expressions $(\langle p \cdot q \rangle_{[0,r]})^*$ on randomly generated Boolean signals p and q of fixed length and variability, taken as large numbers in order to stress our algorithms. Reducing r in the expression increases the fixpoint index n, defined as the minimum sequence length such that all longer sequences are included in a sequence of at most this length. We can compare incremental and squaring star algorithms as n changes and we plot the

performance results in Fig. 5-(b). We see that the squaring performs better than incremental algorithm except cases where n is small. This can be explained by the fact that in the squaring approach, the effect of filtering is multiplied in the following sense. Every sequence that we discard may have appeared in several factorizations of longer sequences; squaring will reuse sequences as subsequences in many places which the incremental approach does not do. Another effect that we observe is that the number of zones covering sequences of length n does not explode but rather seem to stay constant over iterations. This is illustrated by the linearity of execution time with respect to fixpoint index in the incremental approach.

Example: A complex expression In this example we keep expression φ fixed while modifying the signal. We have two Boolean signals p and q, and define an expression which is satisfied when they oscillate rapidly together for some amount of time:

$$\varphi := \langle (\langle p \cdot \bar{p} \rangle_{[0,10]})^* \wedge (\langle q \cdot \bar{q} \rangle_{[0,10]})^* \rangle_{[80,\infty]}$$

We generate input signals by segments, with a segment length of 400 time units. For each segment, we draw switching points in time using an exponential distribution over the segment to provide less switching at the end thus favoring the stabilization case. We tested the expression φ with varying signal length and variability. The results are depicted in Table 1. We also report the number of maximally uniform intervals in the signal $|w|$ along with the number of zones found $|Z_\varphi|$. These results are consistent with simpler examples, indicating that one can monitor complex expressions without facing a blow-up in computation time.

Table 1. Evaluation time of the matchset construction of φ as a function of the variability ($\mathcal{V}$) and length ($\mathcal{L}$) of input signal w.

$\mathcal{V}$	$\mathcal{L}$	$\lvert w\rvert$	$\lvert Z_\varphi\rvert$	Time (s)
0.025	40000	1893	0	0.08
0.025	80000	3825	0	0.17
0.025	160000	7642	0	0.37
0.05	40000	3654	0	0.27
0.05	80000	7305	0	0.60
0.05	160000	14614	0	1.27
0.075	40000	5131	1	0.64
0.075	80000	10476	4	1.40
0.075	160000	21200	5	2.88
0.1	40000	6715	10	1.35
0.1	80000	13306	23	2.73
0.1	160000	26652	47	5.83

6 Future work

We can already consider several direct or indirect extensions to the work presented here.

Longest or shortest match. There often exists several matches beginning at a given time or position. In that case string matching programs enforce the *greedy* policy of returning the longest match (containing all other matches), while hardware specification languages enforce the *lazy* policy of returning the shortest match (earliest violation). Both features are straightforward to implement within our framework.

Online matching. In the context of dynamic verification, it is useful to monitor a property during simulation so as to possibly stop early in case a violation. Making our algorithm work online would require to re-order the operations on zones according to some notion of time. The duration-restriction operator may also enjoy specific treatment so as to cancel matches passed the maximum duration.

New operators. Temporal operators can be useful to specify the intent of the regular expression; for instance a safety property should be monitored according to the semantics of a temporal *always* operator. This feature is available in hardware specification languages PSL/SVA, where a regular expression may be evaluated in the context of an arbitary temporal logic formula. Negation may also be introduced not as a Boolean operation on signals, but as a regular expression primitive. Looking for *absence* of a match requires to compute the complement of a matchset, which may be expensive to compute.

Events and delays. Standard assertions languages only handle events or actions. This also was the model of the timed regular expressions of [ACM02]. In our signals framework, we would like to handle events such as rise and fall of signals. Events only induce finitely many matches and can be represented by punctual zones. In the setting of events-based regular expressions, specification languages have concatenation operators allowing to wait a (non-deterministic) number of discrete time units between events; the generalization of SVA proposed in [HL11] shows that this concept easily translates in the real-time setting.

References

[ACM02] Eugene Asarin, Paul Caspi, and Oded Maler. Timed regular expressions. *Journal of the ACM (JACM)*, 49(2):172–206, 2002.

[AD94] Rajeev Alur and David L Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[AFH96] Rajeev Alur, Tomás Feder, and Thomas A Henzinger. The benefits of relaxing punctuality. *Journal of the ACM (JACM)*, 43(1):116–146, 1996.

[AGM+90] Stephen Altschul, Warren Gish, Webb Miller, Eugene Myers, and David Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, pages 403–410, 1990.

[BGM02] Marius Bozga, Susanne Graf, and Laurent Mounier. IF-2.0: A validation environment for component-based real-time systems. In *CAV*, pages 343–348, 2002.

[BM77] Robert Stephen Boyer and J Strother Moore. A fast string searching algorithm. *Communications of the ACM*, 1977.

[CR02] Maxime Crochemore and Wojciech Rytter. *Jewels of Stringology*. World Scientific, 2002.

[CVK04] Ben Cohen, Srinivasan Venkataramanan, and Ajeetha Kumari. *Using PSL/Sugar for formal and dynamic verification: Guide to Property Specification Language for Assertion-based Verification*. VhdlCohen Publishing, 2004.

[DBS12] Petr Dluhos, Lubos Brim, and David Safránek. On expressing and monitoring oscillatory dynamics. In *HSB*, pages 73–87, 2012.

[EF06] Cindy Eisner and Dana Fisman. *A practical introduction to PSL*. Springer, 2006.

[Fri06] Jeffrey Friedl. *Mastering regular expressions*. O'Reilly Media, Inc., 2006.

[FRM94] Christos Faloutsos, Mudumbai Ranganathan, and Yannis Manolopoulos. Fast subsequence matching in time-series databases. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, 1994.

[HL11] John Havlicek and Scott Little. Realtime regular expressions for analog and mixed-signal assertions. In *FMCAD*, pages 155–162, 2011.

[JS07] Edwin H Jacox and Hanan Samet. Spatial join techniques. *ACM Transactions on Database Systems (TODS)*, pages 0–70, 2007.

[Kle56] Stephen Cole Kleene. Representation of events in nerve nets and finite automata. *Automata Studies*, 1956.

[KMP77] Donald E Knuth, James H Morris, Jr, and Vaughan R Pratt. Fast pattern matching in strings. *SIAM Journal on Computing*, pages 323–350, 1977.

[Koy90] Ron Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4):255–299, 1990.

[Lau00] Ville Laurikari. NFAs with tagged transitions, their conversion to deterministic automata and application to regular expressions. In *Proceecedings of the symposium on String Processing and Information Retrieval (SPIRE)*, pages 181–187, 2000.

[MN04] Oded Maler and Dejan Nickovic. Monitoring temporal properties of continuous signals. In *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems (FORMATS/FTRTFT)*, pages 152–166. 2004.

[MNP05] Oded Maler, Dejan Nickovic, and Amir Pnueli. Real time temporal logic: Past, present, future. In *Proceedings of the Conference on Formal Modeling and Analysis of Timed Systems (FORMATS)*, pages 2–16, 2005.

[MNP08] Oded Maler, Dejan Ničković, and Amir Pnueli. Checking temporal properties of discrete, timed and continuous behaviors. In *Pillars of Computer Science*, pages 475–505. 2008.

[Pik87] Rob Pike. The text editor sam. *Software: Practice and Experience*, 17(11):813–845, 1987.

[Spe06] Chris Spear. *SystemVerilog for Verification*. Springer, 2006.

[Ste94] Graham A Stephen. *String searching algorithms*. World Scientific, 1994.

[Tho68] Ken Thompson. Programming techniques: Regular expression search algorithm. *Communications of the ACM*, pages 419–422, 1968.

[VR06] Srikanth Vijayaraghavan and Meyyappan Ramanathan. *A practical guide for SystemVerilog assertions*. Springer, 2006.

[Wei73] Peter Weiner. Linear pattern matching algorithms. *Switching and Automata Theory*, 1973.