Efficiently Computable Distance-Based Robustness for a Practical Fragment of STL

Neha Rino, Mohammed Foughali, Eugene Asarin

Efficiently Computable Distance-Based Robustness for a Practical Fragment of STL

Neha Rino[1][0009−0001−3273−1954], Mohammed Foughali[2][0000−0002−5348−5465], and Eugene Asarin[2][0000−0001−7983−2202]

[1] Department of Computer Science, University of Warwick, Coventry, UK
[2] Université Paris Cité, CNRS, IRIF, Paris, France

Abstract. Quantitative monitoring mitigates two issues observed in exhaustive, qualitative verification approaches, namely the state-space explosion problem, and the rigidity of their binary verdicts. This is achieved through (i) analysing individual executions instead of building the whole state-space and (ii) providing a robustness measure instead of yes/no answers. In this paper, we consider real-time systems where executions and specifications are modelled as timed signals and Signal Temporal Logic (STL) formulae, respectively. We propose a new temporal robustness measure δ for STL, based on a new distance that we define over timed signals. In contrast with existing measures, δ provides a precise quantification of distances between the monitored signal and the boundary separating faulty and non-faulty executions w.r.t. an STL property. Thus, δ is suitable for a wide range of real-life perturbations, such as those affecting exclusively a particular time window within a signal. Though we prove that computing δ is NP-hard in general, we provide efficient algorithms for a practical fragment of STL. In particular, this fragment includes the key property of bounded response.

1 Introduction

Context & Motivation. A real-time system (RTS), e.g., a mobile robot or a self-driving car, is typically safety-critical: its failure may lead to catastrophic human, environmental or financial damages. Formally verifying that an RTS satisfies a *specification*, i.e., a set of real-time properties, is therefore crucial. In exhaustive qualitative approaches such as model checking, a *mathematical model* of the RTS, representing all its possible executions, is formally verified against real-time properties, formalised in a timed logic [3]. Despite their success with relatively large industrial applications (see e.g., [9]), these approaches suffer from two major drawbacks. First, the complexity of the underlying RTS often leads to state-space explosion [4]. Second, even if building the state space scales, the verification result is binary (the RTS satisfies/violates the properties), whereas more information is often needed (e.g., how close was the RTS to satisfying/violating a property?). Quantitative monitoring [14,16,5,12] is an alternative that tackles both limitations above through (i) analysing individual executions instead of building the whole state-space and (ii) providing a *robustness measure* instead

of yes/no answers. Distance-based robustness, exploiting the notion of distances between functions (e.g., the Fréchet distance [10], the Skorokhod distance [18], and Dynamic Time Warping [6]), gained significant popularity since Fainekos and Pappas [7] (see below), and is the focus of this paper[3]. Typically, in this context, the RTS executions are real-valued *timed signals* and properties are formalised in the Signal Temporal Logic (STL) [14]. The verification verdict is therefore a real number indicating "how far the execution at hand is from satisfying/failing a property". In the following, let ω be a real-valued timed signal representing an RTS execution, and φ an STL property.

Fainekos and Pappas [7] pioneered a notion of robustness based on the distance between ω and the boundary of the set of signals satisfying/violating φ. They focused on *spatial robustness* rather than *temporal robustness*, i.e., on perturbations that affect *what* happened in ω, rather than *when*. They noted that computing distances to measure such robustness is hard, and used therefore approximations. Several robustness measures incorporating different notions of temporal robustness were defined later on, in particular in [5] and [17]. Temporal robustness measures in [5] and [17] are based on a notion of distance corresponding to the overall deviation of ω from φ, through projecting ω on predicates over φ. These measures are efficiently computable but relatively rigid: they cannot capture some real-life temporal perturbations, e.g., when a particular section of ω slows down or speeds up independently of the rest of ω (more in Sect. 5). There is therefore a lack of a notion of efficiently computable temporal robustness, based on precise distances between ω and the set of signals satisfying/violating φ.

Contributions & Outline. In this paper, we show that by focusing on Boolean (timed) signals, obtained from real-valued ones, and restricting STL to a set of practical properties, we can define a precise, efficiently computable robustness measure. We first define a new distance d, that captures precisely how far a Boolean signal ω (i.e., a function from a dense-time interval to a finite set) is from another Boolean signal, for which we provide a linear algorithm. Inspired by the *Hausdorff distance* and its extension to timed words [2], two signals are d apart if d is the least amount of time within which any value in one signal has a matching value in the other signal. Thus, d accounts for perturbations such as stretching/shrinking constant-valued segments within ω by some amount of time, which corresponds to a temporal perturbation that is restricted to a particular time window within the execution. Using d, we define a new temporal robustness measure δ, as the distance of ω to the boundary between the language of a property and its complement. While we prove that, in general, the computation of δ is NP-hard, we provide efficient algorithms to achieve it for a practical fragment of STL, linear in the product of signal and formula sizes, including the bounded response property. We therefore provide the first temporal robustness measure on Boolean signals that is both precise and efficiently computable on a practical fragment of STL.

[3] Verification approaches where the verdict is still binary but the underlying model of the RTS is robust, e.g., [11], are out of this paper's scope.

The rest of this paper is organised as follows. We introduce notions and formalisms used in this paper, namely signals and STL in Sect. 2. Then, the core of our contribution is detailed. First, we present our distance d, and an efficient linear algorithm to compute it (Sect. 3). Afterwards, we present our temporal robustness measure δ, results on its complexity, as well as efficient algorithms to compute it for a fragment of STL (Sect. 4). We then discuss related work in Sect. 5, and wrap up with concluding remarks (Sect. 6). Due to space constraints, most proofs and algorithms are provided in the Appendix.

2 Preliminaries

2.1 Boolean signals

To define signals, we first set up some notation. We say that a function f defined over the domain $[0,T] \subset \mathbb{R}$ is piecewise constant (resp. piecewise linear) if there exists a finite collection of connected subsets of $[0,T]$, namely $\{S_i\}_{i \leq N}$ such that $\bigcup_{i \leq N} S_i = [0,T]$ and $\forall i \leq N$, f when restricted to S_i is constant (resp. linear).

A function f over $[0,T] \subset \mathbb{R}$ is called *càdlàg* if, for every element t of its domain, f has both a left limit and a right limit, and the latter equals $f(t)$:

$$\forall t \in [0,T] : \exists \lim_{t' \to t^-} f(t') \wedge f(t) = \lim_{t' \to t^+} f(t').$$

Definition 1 (Signals). *An h-dimensional Boolean timed signal is a piecewise constant càdlàg function $\omega : [0,T] \to \{0,1\}^h$.*

The domain of ω is $[0,T]$, the *range* of ω is $\{y \in \{0,1\}^h | \exists x \in [0,T] \ \ \omega(x) = y\}$, and the inverse image of a value $v \in \{0,1\}^h$ under ω is $\omega^{-1}(v) = \{t | \omega(t) = v\}$.

Note that ω in Def. 1 can be obtained through applying the predicates $p_1?, \ldots, p_h?$ (over a set of propositional variables $P = \{p_1, \ldots p_h\}$) to a *real-valued signal* and transforming to càdlàg (see e.g., [14]). Accordingly, ω represents the behaviour of each $p_i \in P$ over time: $p_i(t) = \pi_i(\omega(t))$ for any $i \in \{1, \ldots h\}, t \in [0,T]$, where π_i is the projection onto the i^{th} component of ω. Hereafter, we refer to Boolean timed signals simply as "signals".

Given a signal ω over a domain $[0,T]$, we define the restriction of the signal between $[a,b]$ to be the signal $\omega|_a^b$ over the domain $[0, b-a]$ such that $\omega|_a^b(t) = \omega(t+a)$. We also define the concatenation of two signals ω_1 over domain $[0,T_1]$ and ω_2 over $[0,T_2]$ as a signal $\omega_1 \cdot \omega_2$ over the domain $[0, T_1+T_2]$, such that

$$\forall\, t \in [0,T_1),\ \omega_1 \cdot \omega_2(t) = \omega_1(t) \text{ and } \forall t \in [T_1, T_1+T_2],\ \omega_1 \cdot \omega_2(t) = \omega_2(t-T_1).$$

2.2 Signal Temporal Logic

To express timed properties overs signals, we use Signal Temporal Logic (STL)[14]. Note that, since we consider Boolean signals (that can be obtained from real-valued ones, Sect. 2.1), the definitions we provide for STL coincide with those of MITL[1].

Definition 2 (Syntax). *An STL formula φ has the following form :*

$$\varphi \coloneqq \top \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \, \mathcal{U}_{[a,b]} \, \varphi,$$

where $p \in P$, $a, b \in \mathbb{Q}_{\geq 0}$ and $a \leq b$.

Other modalities like $\Diamond_{[a,b]}$ and $\Box_{[a,b]}$ are introduced in the standard manner:

$$\Diamond_{[a,b]}\varphi \coloneqq \top \, \mathcal{U}_{[a,b]} \, \varphi \text{ and } \Box_{[a,b]}\varphi \coloneqq \neg\Diamond_{[a,b]}\neg\varphi.$$

Definition 3 (Qualitative semantics). *The qualitative semantics of STL is given by the function χ defined below.*

$$\begin{aligned}
\chi(\omega, \top, t) &= 1;\\
\chi(\omega, p_i, t) &= \pi_i(\omega(t));\\
\chi(\omega, \neg\varphi, t) &= 1 - \chi(\omega, \varphi, t);\\
\chi(\omega, \varphi_1 \vee \varphi_2, t) &= \max(\chi(\omega, \varphi_1, t), \chi(\omega, \varphi_2, t));\\
\chi(\omega, \varphi_1 \, \mathcal{U}_{[a,b]} \, \varphi_2, t) &= \max_{t' \in [t+a, t+b]} \{\min(\min_{t'' \in [t,t']} (\chi(\omega, \varphi_1, t'')), \chi(\omega, \varphi_2, t'))\}.
\end{aligned}$$

A signal ω satisfies a formula ϕ, denoted $\omega \vDash \varphi$, iff $\chi(\omega, \varphi, 0) = 1$.

3 The Distance d

We will now define our distance d between two signals. Informally, d is the least amount of time within which any value in one signal has a matching value in the other signal. Our distance d is based on the Hausdorff distance d_H, a standard distance over sets of points X and Y:

$$d_H(X, Y) = \max\{\sup_{x \in X} \inf_{y \in Y} |x - y|, \sup_{y \in Y} \inf_{x \in X} |x - y|\}.$$

Definition 4 (Distance d). *Given two signals (of the same dimension) $\mathbf{s} : [0, T_1] \to \{0,1\}^h$, $\mathbf{r} : [0, T_2] \to \{0,1\}^h$ we define the directed distance:*

$$\overrightarrow{d}((\mathbf{s}, \mathbf{r}) = \sup_{x \in [0, T_1]} \inf_{\mathbf{r}(y) = \mathbf{s}(x)} |x - y|.$$

Distance d over the same signals is defined by symmetrising $\overrightarrow{d}$:

$$d(\mathbf{s}, \mathbf{r}) = \max(\overrightarrow{d}(\mathbf{s}, \mathbf{r}), \overrightarrow{d}(\mathbf{r}, \mathbf{s})).$$

Therefore, d is computed using the Hausdorff distances between the sets of intervals where the signals have the same value. Note that d is inspired by the distance on timed words in [2], which we call d_{ABD} (after its authors). However, an intuitive encoding of signals into timed words shows that d and d_{ABD} are incomparable, and consequently that d_{ABD} is not suitable for signals (Sect. 5).

Proposition 1. *For any fixed dimension h, d is a metric (with finite or infinite values) on h-dimensional signals.*

3.1 Algorithm to compute d efficiently

Computational settings The complexity of our algorithms is measured under the computational model of a multi-tape Turing machine.

Given a signal $\omega : [0, T] \to \{0, 1\}^h$, we define a data structure to store and manipulate ω. We use a list of triples representing the segments of ω, i.e., $\{(0 = t_0, v_0, t_1), (t_1, v_1, t_2), \ldots, (t_{n-1}, v_{n-1}, t_n = T)\}$ where ω takes the value v_j on the interval $[t_j, t_{j+1})$ ($[t_j, t_{j+1}]$ if $j = n-1$). The dimension of the signal (and thus of each value v_j, for all $0 \leq j < n$) equals h, so by definition each v_j can be represented by a binary string of length h by simply listing its value on each component.

As per standard practice, the points of discontinuity of any signal (i.e., the set of points where its value changes, $\{t_0, t_1, \ldots t_n\}$) are rational numbers. Given a signal ω, the numbers, $\{t_1, \ldots t_n\}$ are written as fractions, $\{\frac{p'_1}{q_1}, \ldots \frac{p'_n}{q_n}\}$ such that $\forall i \leq n$ p'_i and q_i are coprime. Let $q = lcm(\{q_1, q_2, \ldots q_n\})$, and for all $i \leq n$, rewrite t_i as the equivalent fraction $\frac{p_i}{q}$. Let the least number of bits required to write any element of the set $\{p_1, p_2, \ldots p_n, q\}$ be b. We say accordingly that ω has size (n, h, b). The bit size of such a representation is $\mathcal{O}(n(b + h))$. Fig. 1 illustrates an example of a two-dimensional signal ω. Its representation and size following the data structure above are given in the caption.

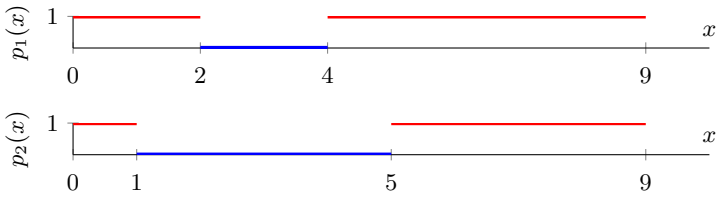

Fig. 1. A 2-dimensional signal ω with representation $\{(0, 11, 1), (1, 10, 2), (2, 00, 4), (4, 10, 5), (5, 11, 9)\}$, size $(5, 2, 4)$.

Proposition 2. *A signal ω of size (n, h, b) satisfies* $|\text{Range}(\omega)| \leq \min(n, 2^h)$.

In order to efficiently implement the distance algorithm, we perform a preprocessing step that turns our signal representation into a convenient list of lists of intervals. Each interval of the signal is assigned to its list based on the value of the signal on the interval (Alg. 1). Note that, in Alg. 1, we use parenthesis to denote the triples "interval lower bound, value, interval upper bound"; the specifities on whether the interval is closed or open at each of its ends are implicitly driven by the càdlàg nature of the signal.

For each signal s, we compute the preimage $PreIm_v$, the set of intervals that evaluate to a vector v' that has v as a prefix. We do so inductively, starting with the 0-dimensional vector $\perp$, whose preimage is the whole signal. At the i^{th} step,

Algorithm 1 Partitioning the signals: RangeSort($\mathbf{s}$)

Let $\mathbf{s} = \{(t_0, v_0, t_1), (t_1, v_1, t_2), \dots (t_{n-1}, v_{n-1}, t_n)\}$
$\{PreIm_\perp\} \leftarrow L_0 = \{(t_0, v_0, t_1), (t_1, v_1, t_2), \dots (t_{n-1} v_{n-1}, t_n)\}$
$i \leftarrow 0$
while $i \leq h$ **do**
    $L_i \leftarrow \emptyset$
    for all $v \in L_{i-1}$ **do**
        $PreIm_{v0} \leftarrow \{(t_j, v_j, t_{j+1}) | v0 \text{ is a prefix of } v_j\}$
        $PreIm_{v1} \leftarrow \{(t_j, v_j, t_{j+1}) | v1 \text{ is a prefix of } v_j\}$
        $L_i \leftarrow L_i \cup \{PreIm_{v0}, PreIm_{v1}\}$
return L_h

we start with a list $L_{i-1} = \{PreIm_{v_1}, PreIm_{v_2}, \dots PreIm_{v_r}\}$ where each v_j is an $(i-1)$-dimensional Boolean vector. Given L_{i-1}, we compute L_i by reading through each $PreIm_{v_j}$ once, comparing the i^{th} component of their vector values, and separating the tuples of $PreIm_{v_j}$ into two sets, $PreIm_{v_j 0}$ and $PreIm_{v_j 1}$ respectively, which are then inserted into L_i. This procedure terminates within h stages, with each step taking at most $\mathcal{O}(n(h+b))$ time.

Efficient computation of d

Theorem 1. *Given two signals ω and ω' with size parameters (n, h, b), $d(\omega, \omega')$ can be computed in $\mathcal{O}(hn(h+b))$ (equivalently $\mathcal{O}(h(|\omega| + |\omega'|))$ time.*

As previously noted, our distance d can be written in terms of the Hausdorff distance between real sets:

$$\begin{aligned}
\overrightarrow{d}(\mathbf{s}, \mathbf{r}) &= \sup_{x \in [0, T_1]} \inf_{\mathbf{r}(y) = s(x)} |x - y| \\
&= \max_{v \in \text{Range}(\mathbf{s})} \{ \sup_{\mathbf{s}(x) = v} \inf_{\mathbf{r}(y) = v} |x - y| \} \\
&= \max_{v \in \text{Range}(s)} \overrightarrow{d_H}(\mathbf{s}^{-1}\{v\}, \mathbf{r}^{-1}\{v\}).
\end{aligned}$$

Let $\overline{S}$ the topological closure of set S, and note that $\overrightarrow{d_H}(s^{-1}(\{v\}), r^{-1}(\{v\})) = \overrightarrow{d_H}(\overline{s^{-1}(\{v\})}, \overline{r^{-1}(\{v\})})$. Our overall Alg. 2 is based on the above observations. The scheme is:

- partition the two signals into lists of intervals, corresponding to $\overline{\mathbf{s}^{-1}(v)}$ and $\overline{\mathbf{r}^{-1}(v)}$ for each $v \in \text{Range}(\mathbf{s})$;
- compute the Hausdorff distance over such unions of intervals, for each v;
- maximise over all $v \in Range(s)$.

The first step is done in $\mathcal{O}(hn(h+b))$ time, and the third in $\mathcal{O}(nb)$ time. Next, we show how to achieve $\mathcal{O}(nb)$-time complexity for the second step as well.

Algorithm 2 Overall Distance Computation: $\mathsf{Dist}(\mathbf{s}, \mathbf{r})$

Let $\mathbf{s}, \mathbf{r}$ be two signals
$(\mathbf{s}, \mathbf{r}) \leftarrow \mathsf{RangeSort}(\mathbf{s}, \mathbf{r})$
$d \leftarrow 0$
if Range($\mathbf{s}$) $\neq$ Range($\mathbf{r}$) **then**
  return ∞
for all $v \in$ Range($\mathbf{s}$) **do**
  $d \leftarrow \max(d, \mathsf{HDist}(\overline{\mathbf{s}^{-1}(v)}, \overline{\mathbf{r}^{-1}(v)}), \mathsf{HDist}(\overline{\mathbf{r}^{-1}(v)}, \overline{\mathbf{s}^{-1}(v)}))$
return d

Computing Hausdorff distance over real intervals Suppose we seek to compute $\overrightarrow{d_H}(S, R)$ where S and R are unions of disjoint, increasing intervals in $\mathbb{R}$, i.e., $S = \bigsqcup_{1 \le i \le n} [ls_i, us_i]$ and $R = \bigsqcup_{1 \le j \le m} [lr_j, ur_j]$.

A useful auxiliary function in this setting is the notion we define below as *close*. close : $\mathbb{R} \times \mathcal{P}(\mathbb{R}) \to \mathbb{R}$ takes a point x in $\mathbb{R}$ and a closed subset S of $\mathbb{R}$, and returns the closest possible point in S to x, i.e.,

$$\text{close}(x, S) = \arg\min_{y \in S} |x - y|.$$

This allows us to rewrite the directional Hausdorff distance as

$$\overrightarrow{d_H}(S, R) = \sup_{x \in S} |x - \text{close}(x, R)|.$$

We introduce two lemmas regarding the directional Hausdorff distance in some simpler cases:

Lemma 1. *Given two real intervals $s = [ls, us]$ and $r = [lr, ur]$, the directional Hausdorff distance between these is given by $\overrightarrow{d_H}(s, r) = \max(0, lr - ls, us - ur)$.*

Proof. Consider distance to r overall as a function of each point, i.e., the function $dr : \mathbb{R} \to \mathbb{R}_{\ge 0}$ such that $\forall t \in \mathbb{R}$, $dr(t) = \overrightarrow{d_H}(\{t\}, r)$. For a fixed r, dr is a real convex function, so it can only attain a maximum over s at one of its endpoints.

Hence, for all $ls < t < us$, $\overrightarrow{d_H}(\{t\}, r) \le \max(\overrightarrow{d_H}(\{ls\}, r), \overrightarrow{d_H}(\{us\}, r))$, so we have reduced the problem to computing $dr(ls)$ and $dr(us)$.

We proceed by considering four cases, corresponding to the relative positions of the endpoints of s and r.

Case 1 : $lr \le ls \le us \le ur$**.**
In this case, clearly $s \subseteq r$, so $\overrightarrow{d_H}(s, r) = 0$.
Case 2 : $lr \le ls < ur < us$**.**
In this case, $dr(ls) = 0$ and $dr(us) = us - ur$.
Case 3 : $ls < lr < us \le ur$**.**
In this case, $dr(ls) = lr - ls$ and $dr(us) = 0$.
Case 4 : $ls < lr < ur < us$**.**
In this case, $dr(ls) = lr - ls$ and $dr(us) = us - ur$.

Overall, we can conclude that,

$$\overrightarrow{d_H}(s,r) = \sup_{t \in s} \overrightarrow{d_H}(\{t\}, r) = \max(dr(ls), dr(us)) = \max(0, lr - ls, us - ur).$$

Example 1. We consider two signals s and r presented on Fig. 2.

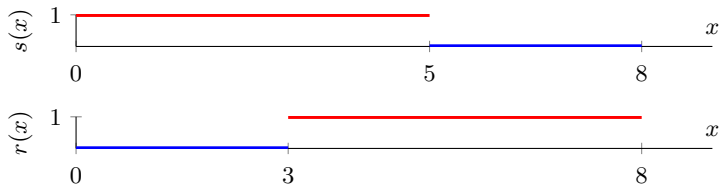

Fig. 2. Two one-dimensional signals

Let us compute $\overrightarrow{d}(s,r)$ using the auxiliary function *close*(). For all points x belonging to $[0,3]$, their value in s is one, and the closest point in r with the same value is $\text{close}(x, r^{-1}(1)) = 3$. For $x \in [3,5]$, signals s and r have the same value, i.e., $\text{close}(x, r^{-1}(1)) = x$; this segment does not contribute to the overall distance. For $x \in [5,8]$, $s(x)$ equals zero and the nearest point with the same value in r is also 3, i.e., $\text{close}(x, r^{-1}(1)) = 3$. Therefore:

$$\overrightarrow{d}(s,r) = \max(\sup_{x \in [0,3]}(|x-3|), 0, \sup_{x \in [5,8]}(|x-3|)) = 5$$

Similarly, we can compute $\overrightarrow{d}(r,s) = \max(\sup_{x\in[0,3]}(|x-5|), 0, \sup_{x\in[5,8]}(|x-5|)) = 5$ and finally $d(s,r) = \max(\overrightarrow{d}(s,r), \overrightarrow{d}(r,s)) = 5$.

As this illustrates, close() allows us to refine the domain of s until each segment either has a single closest point or has an identical valued segment in r. We implement close() using the easier to manipulate $\text{closeInt} : S \to [m]$, denoting the closest interval to a point: $\text{closeInt}(x) = \min\{j | \text{close}(x, R) \in R_j\}$.

Lemma 2. *Given any point $t \in \mathbb{R}$ and a set of closed intervals in increasing order $[lr_1, ur_1] \ldots [lr_m, ur_m]$, we define the bucket sequence $\{r_i\}_{i \le m}$:*

$$r_0 = -\infty, \forall\, i \in [m-1] : r_i = \frac{lr_{i+1} + ur_i}{2}, r_m = \infty.$$

We have the following equivalence: $\text{closeInt}(t) = i \iff r_{i-1} \le t \le r_i$.

Proof. The bucket sequence is named such because it divides the real line into m buckets. Because these buckets cover all of $\mathbb{R}$, it is clear that $\forall t \in \mathbb{R}, \exists j \le m, r_{j-1} < t \le r_j$.

We seek to prove that $\text{closeInt}(t) = i \iff i = j$ as defined above.

We begin by remarking that $\forall t \in \mathbb{R}$, if $\text{closeInt}(t) = i$ then for all $j \neq i$, we know $t \notin [lr_j, ur_j]$ as then $\text{closeInt}(t) = j$.

Now, we consider the following two cases based on the relative positions of the closest segment to t and the bucket it belongs to.

Case 1 : $\text{closeInt}(t) = i > j$ Since this implies that the i-bucket is to the right of the j-bucket, we infer that

$$r_{i-1} - t = r_{i-1} - r_j + r_j - t \geq r_j - t$$

Now, using the above and Lem. 1 we can conclude that

$$\overrightarrow{d_H}(t, [lr_i, ur_i]) \geq lr_i - t \geq r_{i-1} - t \geq r_j - t \geq \overrightarrow{d_H}(t, [lr_j, ur_j])$$

Case 2 : $\text{closeInt}(t) = i < j$ Since this implies that the i-bucket is to the left of the j-bucket, we infer that

$$t - r_i = t - r_{j-1} + r_{j-1} - r_i \geq t - r_{j-1}$$

Now, using the above and Lem. 1 we can conclude that

$$\overrightarrow{d_H}(t, [lr_i, ur_i]) \geq t - ur_i \geq t - r_i \geq t - r_{j-1} \geq \overrightarrow{d_H}(t, [lr_j, ur_j])$$

Both these cases result in a contradiction, hence $i = j$, the closest interval is the one in the bucket of t.

Algorithm 3 Hausdorff Distance Computation: $\mathsf{HDist}(S, R)$

Let $S = \bigsqcup_{1 \leq i \leq n} [ls_i, us_i]$ and $R = \bigsqcup_{1 \leq j \leq m} [lr_j, ur_j]$
$i, j, \leftarrow 1, l \leftarrow ls_1$, $B \leftarrow \emptyset$, $S' \leftarrow \emptyset$, $B[0] \leftarrow -\infty$
for all $i < m$ **do**
    $B[i] \leftarrow \frac{lr_{i+1} + ur_i}{2}$
$B[m] \leftarrow +\infty$, $i \leftarrow 0$
while $i \leq n$ **do**
    while $j \leq m$ **do**
        if $B_j \leq l$ **then**
            $j \leftarrow j + 1$
        else if $us_i \leq B_j$ **then**
            $S' \leftarrow (l, us_i, j)$, $i \leftarrow i + 1$, $l \leftarrow ls_i$
        else
            $S' \leftarrow (l, B_j, j)$, $j \leftarrow j + 1$, $l \leftarrow ur_j$
for all $(l, u, j) \in S'$ **do**
    $d \leftarrow \max(d, lr_j - l, u - ur_j)$
return d

Using Lem. 1 and Lem. 2, Alg. 3 efficiently computes $\overrightarrow{d_H}$. In brief, we sort all the points in S into buckets between r_i and r_{i+1}, resulting in at most $m + n$ segments in S. The distance of each such segment to the appropriate segment in R is computed in $\mathcal{O}(b)$ time.

4 Distance-based Temporal Robustness δ

Our temporal robustness δ is based on the distance between a signal and the language of a formula. We first recall the standard definitions in this setting. Def. 6 is the temporal version of robustness in [7].

Definition 5 (Point-to-set distance). *Given a metric d, a point ω and a set S, we define $d(\omega, S) = \inf_{\omega' \in S} d(\omega, \omega')$. As a special case, $d(\omega, \emptyset) = \infty$.*

Definition 6 (Temporal robustness). *Given a signal ω and an STL formula φ, the distance-based notion of temporal robustness, δ, is defined as*

$$\delta(\omega, \varphi) = \begin{cases} d(\omega, \mathcal{L}(\neg\varphi)), & \text{if } \omega \vDash \varphi; \\ -d(\omega, \mathcal{L}(\varphi)), & \text{if } \omega \nvDash \varphi. \end{cases}$$

Applying these definitions to our distance d, our robustness measure δ shows how far a signal ω is to the closest boundary point of the language of a property φ. If $\omega \nvDash \varphi$, then $-\delta(\omega, \varphi)$ is the minimum distance between ω and any signal ω' that satisfies φ (i.e., in any such ω', there is a value where ω and ω' are mismatched by at least $|\delta(\omega, \varphi)|$). Conversely, if $\omega \vDash \varphi$, then $\delta(\omega, \varphi)$ is the minimum distance ω has from any signal ω' violating φ.

We first argue that computing δ is intractable in general, as we prove that even estimating whether it is finite given a one-dimensional signal is NP-hard (Sect. 4.1). Then, we identify a tractable fragment of STL, including the bounded response property, for which we provide linear algorithms (in the product of the signal and formula sizes) to compute δ (Sect. 4.2).

4.1 Hardness of computing δ

Theorem 2. *Given a one-dimensional signal ω and an STL formula φ, deciding whether $\delta(\omega, \varphi)$ is finite or not is NP-hard.*

Proof. (sketch) The proof of NP-hardness proceeds via a reduction from CNF-SAT. Given ψ, a propositional logic formula in conjunctive normal form with variables $\{x_1, \ldots, x_n\}$, we first check whether it is satisfied by assigning all the variables the value 0, or all 1. If this is not the case, we map formula ψ to an STL formula ϕ over a one-dimensional signal ω, replacing each x_i by $\Box_{[i-0.5,i]} p$.

We claim that ϕ is satisfiable iff ψ is. Indeed, any satisfying assignment v of ψ can be mapped to a one-dimensional signal on $[0, n]$ having $\omega(t) = v(q_i)$ for $t \in [n-1, n)$, which satisfies φ. In the other direction, given a signal ω satisfying φ, we can build a satisfying assignment for ψ taking $v(x_i) = \chi(\omega, \Box_{[i-0.5,i]} p, 0)$.

To finish the proof, we consider a one-dimensional signal ω_0, taking both values 0 and 1. By the previous, if ψ is satisfiable then $\delta(\omega_0, \varphi)$ is finite, otherwise infinite, this concludes the reduction.

For example, in order to check whether $\psi = (x_1 \vee x_2) \wedge (\neg x_1)$ is satisfiable, we consider the STL formula $\varphi = (\Box_{[0.5,1]} p \vee \Box_{[1.5,2]} p) \wedge (\Box_{[0.5,1]} \neg p)$.

4.2 A tractable fragment of STL

We show in the following that computing $\delta(\omega, \varphi, t)$ with φ a formula in STL_r, a restricted fragment of STL, has linear complexity in the product of the signal and formula sizes (i.e. $\mathcal{O}(|\varphi| \cdot |\omega|)$).

STL_r, albeit restrictive, contains practically relevant properties such as bounded response. The latter has in general the form $\square(B_1 \implies \Diamond_{[0,b]} B_2)$, which means that every request (specified by propositional formula B_1) is granted (as specified by B_2) within b time units. This non trivial property, with nested constrained liveness and safety modalities, is crucial in practice [8], [15], [13].

Before introducing STL_r, we first define the *domain* of an STL formula as an over-approximation of the time intervals that determine its satisfiability. The addition operator over intervals $X, Y \subseteq \mathbb{R}$ is the classical one i.e., $X + Y \coloneqq \{x + y \,|\, x \in X \,, y \in Y\}$.

Definition 7 (Domain of a formula). *Given an STL formula φ, we define its domain, $dom(\varphi)$, inductively as follows :*

$$\begin{aligned}
\mathrm{dom}(\top) &= \emptyset; \\
\mathrm{dom}(p) &= [0, 0]; \\
\mathrm{dom}(\neg\varphi) &= \mathrm{dom}(\varphi); \\
\mathrm{dom}(\varphi_1 \vee \varphi_2) &= \mathrm{dom}(\varphi_1) \cup \mathrm{dom}(\varphi_2); \\
\mathrm{dom}(\varphi_1 \,\mathcal{U}_{[a,b]}\, \varphi_2) &= (\mathrm{dom}(\varphi_1) + [0, b]) \cup (\mathrm{dom}(\varphi_2) + [a, b]).
\end{aligned}$$

Using Def. 7, the domain of an STL formula φ can be computed inductively on the structure of φ, in time $\mathcal{O}(|\varphi|)$. Informally, the satisfaction of φ by a signal only depends on its values in $\mathrm{dom}(\varphi)$.

Proposition 3. *If $\omega_1|_{\mathrm{dom}(\varphi)} = \omega_2|_{\mathrm{dom}(\varphi)}$ then $\omega_1 \models \varphi \Leftrightarrow \omega_2 \models \varphi$.*

Definition 8 (STL_r). *Given the set of atomic Boolean propositions P, STL_r is the set of all φ defined using the following grammar:*

$$\begin{aligned}
&B \coloneqq p \in P | B \vee B | \neg B; \\
&\varphi \coloneqq B \,|\, \square_{[a,b]} B \,|\, B \,\mathcal{U}_{[a,b]}\, B \,|\, \square(B \implies \Diamond_{[0,b]} B) | \neg\varphi | \varphi_1 \vee \varphi_2,
\end{aligned}$$

whenever $|\,\mathrm{dom}(\varphi_1) \cap \mathrm{dom}(\varphi_2)| \leq 1$.

Algorithms for computing δ on STL_r

Theorem 3. *Given a signal ω with size parameters (n, h, b) and a property $\varphi \in STL_r$, computing $\delta(\omega, \varphi)$ is in $\mathcal{O}(n(h + b) \cdot |\varphi|)$-time.*

We prove this theorem by providing an algorithm that achieves the claimed complexity, by induction over the structure of φ using several linear-time algorithms. The top-level procedure is given in Alg. 4.

Algorithm 4 Computing $\delta(\omega, \varphi)$ with $\varphi \in \mathrm{STL}_r$

Let $\omega : [0, T] \to \{0, 1\}^d$, sign $\leftarrow -1$
if $(\varphi = \varphi_1 \vee \varphi_2)$ **then**
  return $\max(\delta(\omega, \varphi_1), \delta(\omega, \varphi_2))$
else if $(\varphi = \neg\varphi')$ **then**
  return $-(\delta(\omega, \varphi'))$
if $\varphi = B, \Box_{[a,b]} B, B_1 \,\mathcal{U}_{[a,b]}\, B_2, \Box(B_1 \implies \Diamond_{[0,b]} B_2)$ **then**
  if $\varphi = B, \Box_{[a,b]} B$ **then**
    $(\omega, \varphi) \leftarrow \mathsf{colour}(\omega, \varphi, B)$
  else
    $(\omega, \varphi) \leftarrow \mathsf{colour}(\omega, \varphi, B_1, B_2)$
  if $\omega \vDash \varphi$ **then**
    sign $\leftarrow +1$, $\varphi \leftarrow \neg\varphi$
  return sign$\cdot d(\omega, \varphi)$ ▷ *call the function corresponding to* φ

The function Alg. 6 is called explicitly by Alg. 4 if φ uses some Boolean combination of propositional variables (i.e., as we have defined it, B). Alg. 6 corresponds to a preprocessing step, that we call "colouring" ω, through a change of variables (both in ω and φ), considering each maximal Boolean subformula as a new propositional variable. This substitution trick is sound, i.e., the distance to the original language of φ is the same as the distance of the coloured signal to the simplified formula. The last line of the main algorithm (Alg. 4) calls a function corresponding to one of the algorithms Algs. 16, 5 to 7 and 13 to 15 in order to compute $d(\omega, \varphi)$ depending on the syntax of φ. In the following, we give a high-level presentation of the simplest (resp. hardest) case, i.e., when φ is a proposition (resp. bounded response), in which Alg. 5 (resp. Alg. 16) is called. The pseudocode for the remaining algorithms (called by Alg. 4) can be found in the Appendix, alongside proofs of correctness and complexity for all algorithms.

Propositions Alg. 5 pertains to the simplest case, i.e., where φ is a proposition p. We claim that the distance can be found as the leftmost point of the first interval where ω takes some boolean value a satisfying p. Indeed, any trace ω' that satisfies p at 0 is at least at distance t from ω, which implies $d(\omega, p) \geq t$. On the other hand, taking the witness trace ω' coinciding with ω everywhere except a small interval $[0, \varepsilon]$ where it equals a, we get a distance $d(\omega, \omega') = t$ with $\omega' \vDash p$, hence $d(\omega, p) \leq t$.

Algorithm 5 $d(\omega, \varphi)$ where $\varphi = p$

Let $\omega = (0, t_1, a_1)(t_1, t_2, a_2) \dots (t_{n-1}, t_n, a_n)$ where $\forall\, i \in [n] : a_i \in \{0, 1\}$.
$i \leftarrow \min\{k : a_k \vDash p\}$
return $d = t_{i-1}$

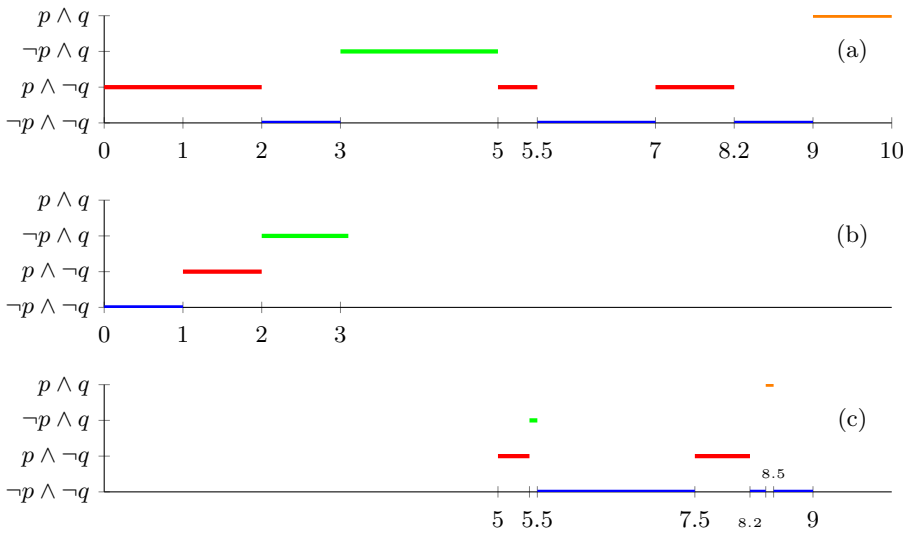

Fig. 3. a signal ω (a); its approximation satisfying φ_1 on $[0,3]$ (b) and on $[5,9]$ (c).

Bounded response Let $\varphi_b = \Box(\mathrm{p} \implies \Diamond_{[0,b]}\mathrm{q})$ with propositions p (modelling a "call") and q (modelling a "response"), and let us focus on a signal ω such that $\omega \not\models \varphi$. In order to isolate where the calls without response live in the signal, we informally describe below a partition of the signal using neighbourhoods. No $(\neg p \wedge \neg q)$ or $(\neg p \wedge q)$-valued segment has a call that needs a response, and any $(p \wedge q)$-valued segment provides each call with a response instantaneously. Hence, there must be a $(p \wedge \neg q)$-valued segment, whose left endpoint is more than b away from the nearest q-satisfying segment to its right, as otherwise every call in the signal has been satisfied. We call each section of the signal with such $(p \wedge \neg q)$-valued segment up to its nearest response a *region of fault*. Regions of fault come in three types, initial (like $[0,t]$, so the 0 point does not satisfy q, middle (a segment that contains a fault, and has no q points, except for at both ends), or final (of the form $[t,T]$ where $[0,T]$ was the domain, such that the T does not satisfy q).

By the following lemma, we see that we can compute the distances of different regions of fault independently, and then maximise over all of them to obtain the overall distance.

Lemma 3 (Decomposition lemma). *Given a signal $\omega \not\models \varphi$ over the domain $[0,T]$, consider the following decomposition of the domain: $t_0 = 0 \leq t_1 < \cdots \leq t_{2n} = T$, where $\forall i \in [n], (t_{2i}, t_{2i+1})$ contains no regions of fault and $[t_{2i+1}, t_{2i+2}]$ is a region of fault. Given such a decomposition, $d(\omega, \varphi) = \max_{1 \leq i \leq n} d(\omega|_{t_{2i-1}}^{t_{2i}}, \varphi)$.*

To illustrate, consider ω in Fig. 3(a) and φ_1 (φ_b with $b = 1$). Here we use red segments for calls without immediate responses, while green and orange segments

Algorithm 16 $d(\omega, \varphi)$ where $\varphi = \Box(p \implies \Diamond_{[0,b]} q)$

Let $\omega = (0, t_1, a_1)(t_1, t_2, a_2) \dots (t_{n-1}, t_n, a_n)$ where $\forall\, i \in [n] : a_i \in \{0, 1\}$.
$R \leftarrow \emptyset$, $i \leftarrow 1$, $u, v \leftarrow 0$
for all $(i \leq n)$ **do**
  if $\omega(t_i) \not\models q$ **then**
    $v \leftarrow t_i$
  else
    if $u < v$ **then**
      $R \leftarrow R \cup [u, v]$
    $u \leftarrow t_i$
  $i \leftarrow i + 1$
$R \leftarrow R \cup [u, T]$, $i \leftarrow 1$
if $(R = \{[0, t_n\}\})$ **then**
  $d \leftarrow \infty$
else
  for all $[u, v] \in R$ **do**
    if $(u = 0) \wedge \omega(0) \not\models q$ **then** ▷ *Initial ROF*
      $l \leftarrow v - \min\{t \in [u, v] \mid \omega(t) \models p\}$
      if $\exists\, t \in [u, v]$, $\omega(t) = \neg p \wedge \neg q$ **then**
        $c \leftarrow -\delta(\omega, \Box_{[v-l, v-l+(\frac{l-b}{2})]} \neg p)$
        $d \leftarrow \max(d, (\frac{l-b}{2}), c)$
      else
        $d \leftarrow \max(d, (l - b))$
    else if $(\omega(u) \models q) \wedge (\omega(v) \models q)$ **then** ▷ *Middle ROF*
      $l \leftarrow v - u$
      $m \leftarrow \max\{t \in [u, v] \mid (t \leq \frac{l-b}{2}) \wedge (\omega(t) = p \wedge \neg q)\}$
      $m' \leftarrow \min\{t \in [u, v] \mid (t \geq \frac{l-b}{2}) \wedge (\omega(t) = p \wedge \neg q)\}$
      $c \leftarrow \max\{r \in \mathbb{R}_{\geq 0} \mid \omega \models \Box_{[\frac{l-b}{2} - r, \frac{l-b}{2} + r]}\, p \wedge \neg q\}$
      $d \leftarrow \max(d, 0.5 \times \max(m, l - b - m', c))$
    else ▷ *Final ROF*
      $l \leftarrow \max\{t \in [u, v] \mid \omega(t) \models p\} - u$
      if $\exists\, t \in [u, v]$, $\omega(t) = \neg p \wedge \neg q$ **then**
        $c \leftarrow -\delta(\omega, \Box_{[v-(\frac{l}{2}), v]} \neg p)$
        $d \leftarrow \max(d, c, (\frac{l}{2}))$
      else
        $d \leftarrow \max(d, l)$
return d

are responses. The initial red segment makes $[0, 3]$ a region of fault as the call at 0 has no response. Given this, locally speaking, the closest signal satisfying bounded response is given on Fig. 3(b). This gets rid of (resp. retains) as much as necessary of the red segment i.e., in the interval $[0, 1)$ (resp. $[1, 2)$) in order to satisfy φ. The overall distance to the language here is 1.

The second region of fault corresponds to $[5, 9]$, as the red segments starting at 5 and 7 constitute calls that do not always receive timely responses. The closest signal here, locally, is given on Fig. 3(c). This signal provides thin *spikes*

of responses at 5.5 and at 8.5, thereby satisfying all the calls in the signal. Notice how the spike is green at 5.5 but orange at 8.5, this is to obtain the closest signal to ω in this region that satisfies φ_1: the closest response to point 5.5 (resp. 8.5) in ω is at its left (resp. right) i.e., the green response up to 5 (resp. orange response from 9). The signal in Fig. 3(c) also minimally shifts the red segment that used to start at 7 to now start at 7.5, providing an overall distance of 0.5.

By gluing these two together, we find a signal that satisfies bounded response globally, and is at the minimum possible distance of 1 from the original signal. Alg. 16 separates the domain into regions of fault (lines 3-11), then depending on the type of fault (initial, middle or final), finds closest signals for the regions of the fault. It then efficiently computes the distance incurred by each region, maximises over all regions, and returns the result. The efficiency lies in the fact that in a given region of fault, irrespective of how many segments it contains, the overall distance is governed by a constant number of parameters.

5 Related Work

The literature on quantitative monitoring and distance-based robustness is abundant. We compare our contributions with the closest works to ours.

From a conceptual point of view, our distance d is heavily inspired by d_{ABD}, a distance on time-event sequences, proposed by Asarin, Basset and Degorre[2]. One could think that time-event sequences and timed Boolean signals are two formalisations of the same phenomenon and that d and d_{ABD}, both Hausdorff like, should therefore coincide. However, an intuitive encoding of Boolean signals into timed-event sequences shows that d and d_{ABD} are actually incomparable. The proof is detailed in the Appendix. In other words, although these distances look quite similar, they in fact measure very different aspects of timed behaviours.

As for robustness, the most relevant semantics for our setting was [7], which defined the spatial robustness measure ρ, computed inductively on the structure of the formula of interest. The authors however did not study temporal robustness. Using similar inductive principles, a temporal robustness measure θ was defined in [5], which was used in conjunction with ρ to yield space-time robustness. Another temporal robustness measure η was later defined by [17]. Both θ and η are more generic than our δ, in the sense that they cover all STL, with fully inductive procedures in the case of θ. However, they have two main disadvantages compared to δ. First, they are both restrictive in terms of the perturbations one permits on the signal. For instance, neither is able to capture perturbations affecting a particular time window in the signal. Second, our δ, providing a precise distance between a signal and the boundary between the sets of signals satisfying/violating the property of interest, is more precise than θ and η, both relying on an overall deviation between a signal and a formula. This results in e.g., a large satisfaction (positive robustness) measure for a signal that can easily violate the property of interest under a very small perturbation. We give examples regarding both points above in the Appendix, and prove that θ

and η are incomparable with δ. In particular, we prove that θ and η are not suitable as upper (or lower) bounds for our δ.

6 Conclusion

In this article, we define a distance d over the space of multi-dimensional timed Boolean signals and use it to develop a robustness measure δ for evaluating signals with respect to STL specifications. We provide efficient algorithms for computing this distance between signals, linear in the number of intervals in the signals. We provide a lower bound for the complexity of computing δ for general STL formulae, by showing that even the problem of estimating whether it is finite is NP-hard. Despite this, we provide as the main result efficient algorithms, with linear complexity in the product of the signal and formula sizes, to compute δ for a class of practical STL formulae, notably including the bounded response property. This means that in a number of real-world instances, δ is efficiently computable and can assist in the analysis and repair of RTSs, contributing to the ongoing effort of quantitative verification for STL specifications.

The main direction for future work consists in extending our efficient algorithms to a larger class of specifications and to real-valued signals. Implementation and experimental evaluation of the approach over real-life case studies, and comparison to other robustness measures is of primary interest. It would be also interesting to obtain an exact complexity of computing δ for general STL formulas, and explore fixed-parameter tractability of the problem.

Acknowledgements. This work was partly supported by Agence Nationale de Recherche and Japan Science and Technology Agency under Grant CyPhAI (`https://www.cyphai.io`). Foughali was partly supported under the program "Investissement d'Avenir" launched by the French Government and implemented by ANR, with the reference "ANR-18-IdEx-000" as part of its program "Emergence". Most of this work was realised during Rino's stay at IRIF, Université Paris Cité. Rino was supported by the IDEX scholarship, ENS Paris-Saclay, the Feuer International Scholarship in Artificial Intelligence by University of Warwick alumnus Jonathan Feuer, and the Centre for Discrete Mathematics and its Applications (DIMAP) and Department of Computer Science, University of Warwick.

References

1. Alur, R., Feder, T., Henzinger, T.A.: The benefits of relaxing punctuality. J. ACM **43**(1), 116–146 (1996). `https://doi.org/10.1145/227595.227602`
2. Asarin, E., Basset, N., Degorre, A.: Distance on timed words and applications. In: International Conference Formal Modeling and Analysis of Timed Systems (FORMATS). pp. 199–214. Springer (2018). `https://doi.org/10.1007/978-3-030-00151-3_12`

3. Bouyer, P.: Model-checking timed temporal logics. Electronic notes in theoretical computer science **231**, 323–341 (2009). https://doi.org/10.1016/j.entcs.2009.02.044
4. Clarke, E.M., Klieber, W., Novácek, M., Zuliani, P.: Model checking and the state explosion problem. In: Tools for Practical Software Verification, LASER. pp. 1–30. Springer (2011). https://doi.org/10.1007/978-3-642-35746-6_1
5. Donzé, A., Maler, O.: Robust satisfaction of temporal logic over real-valued signals. In: International Conference Formal Modeling and Analysis of Timed Systems (FORMATS). pp. 92–106. Springer (2010). https://doi.org/10.1007/978-3-642-15297-9_9
6. Efrat, A., Fan, Q., Venkatasubramanian, S.: Curve matching, time warping, and light fields: New algorithms for computing similarity between curves. Journal of Mathematical Imaging and Vision **27**, 203–216 (2007). https://doi.org/10.1007/s10851-006-0647-0
7. Fainekos, G.E., Pappas, G.J.: Robustness of temporal logic specifications for continuous-time signals. Theor. Comput. Sci. **410**(42), 4262–4291 (2009). https://doi.org/10.1016/j.tcs.2009.06.021
8. Foughali, M., Bensalem, S., Combaz, J., Ingrand, F.: Runtime verification of timed properties in autonomous robots. In: International Conference on Formal Methods and Models for System Design (MEMOCODE). pp. 1–12. IEEE (2020). https://doi.org/10.1109/MEMOCODE51338.2020.9315156
9. Foughali, M., Hladik, P., Zuepke, A.: Compositional verification of embedded real-time systems. J. Syst. Archit. **142**, 102928 (2023). https://doi.org/10.1016/j.sysarc.2023.102928
10. Fréchet, M.M.: Sur quelques points du calcul fonctionnel. Rendiconti del Circolo Matematico di Palermo (1884-1940) **22**(1), 1–72 (1906). https://doi.org/10.1007/BF03018603
11. Gupta, V., Henzinger, T.A., Jagadeesan, R.: Robust timed automata. In: International Workshop on Hybrid and Real-Time Systems (HART). pp. 331–345. Springer (1997). https://doi.org/10.1007/BFb0014706
12. Henzinger, T.A.: Quantitative monitoring of software. In: Software Verification. pp. 3–6. Springer (2022). https://doi.org/10.1007/978-3-030-95561-8_1
13. Henzinger, T.A., Manna, Z., Pnueli, A.: An interleaving model for real-time. In: Next Decade in Information Technology: Jerusalem Conference on Information Technology. pp. 717–730 (1990). https://doi.org/10.1109/JCIT.1990.128356
14. Maler, O., Nickovic, D.: Monitoring temporal properties of continuous signals. In: International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems (FTRTFT). vol. 3253, pp. 152–166. Springer (2004). https://doi.org/10.1007/978-3-540-30206-3_12
15. Maler, O., Nickovic, D., Pnueli, A.: On synthesizing controllers from bounded-response properties. In: International Conference on Computer Aided Verification (CAV). pp. 95–107 (2007). https://doi.org/10.1007/978-3-540-73368-3_12
16. Rizk, A., Batt, G., Fages, F., Soliman, S.: On a continuous degree of satisfaction of temporal logic formulae with applications to systems biology. In: International Conference on Computational Methods in Systems Biology (CMSB). pp. 251–268. Springer (2008). https://doi.org/10.1007/978-3-540-88562-7_19
17. Rodionova, A., Lindemann, L., Morari, M., Pappas, G.J.: Temporal robustness of temporal logic specifications: Analysis and control design. ACM Trans. Embed. Comput. Syst. **22**(1), 13:1–13:44 (2023). https://doi.org/10.1145/3550072
18. Skorokhod, A.V.: Limit theorems for stochastic processes. Theory of Probability & Its Applications **1**(3), 261–290 (1956). https://doi.org/10.1137/1101022

Appendix

A Distance for Boolean Timed Signals

A.1 Properties of d

Proposition 1. *For any fixed dimension h, d is a metric (with finite or infinite values) on h-dimensional signals.*

Proof. We prove that d has the three properties of a metric.

1. Symmetry: d is symmetric by definition.

2. Two signals s and r are 0 apart iff $s = r$:
($\Rightarrow$) By contradiction. Let s and r be two signals s.t. $d(s,r) = 0$ (and therefore $d(r,s) = 0$, by symmetry). This implies that s and r have the same domain $[0,T]$ and that for all $t \in [0,T]$, $\inf_{r(t')-s(t)}\{|t-t'|\} = 0$. It follows that for every $v \in \mathit{Range}(s)$, every point of $s^{-1}(v)$ is a limit point of $r^{-1}(v)$. As both r and s are piecewise constant and càdlàg, the only limit points of $r^{-1}(v)$, for some $v \in \mathit{Range}(s)$, are the right endpoints of constant segments. For s and r to be inequal, there must exist such limit point t that does not belong to $r^{-1}(v)$. Suppose t exists. If t belongs to $s^{-1}(v)$, then as s is càdlàg, we know that the right-limit at t exists and is equal to t, so there is an open neighbourhood to its right where the value of s is v, let ϵ be the radius of this neighbourhood. Let $t' = \min\{t'' | t'' \geq t, \wedge r(t'') = v\}$. Knowing that $r(t) \neq v$, we deduce that $t' - t = \delta > 0$. Consider now $t_s = t + 0.5 \times \min(\epsilon, \delta)$. We see that $s(t_s) = v$ but there are no points t_r such that $r(t_r) = v \wedge |t_r - t_s| \leq 0.25 \times \max(\epsilon, \delta)$, which contradicts $d(s,r) = 0$. It follows that every point of $r^{-1}(v)$ belongs to $s^{-1}(v)$ and vice versa, for all $v \in \mathit{Range}(s) = \mathit{Range}(r)$, which boils down to the equality $s = r$.
($\Leftarrow$) Trivial ($d(s,s) = 0$ for all signals s).

3. Triangle inequality: We prove that for any three arbitrary signals x, y, z, the following holds :

$$d(x,z) \leq d(x,y) + d(y,z)$$

The first case is when $d(x,z) = \infty$. Here, we can easily deduce that one of $d(x,y), d(y,z)$ must also be infinite as their ranges must differ. The above inequality therefore holds.

Otherwise, if $d(x,z)$, $d(x,y)$ and $d(y,z)$ are all finite, then the ranges of the three signals coincide. We can therefore fall back on the above inequality using the triangle inequality for Hausdorff distance :

$$\begin{aligned} d(x,z) = \max_v d_H(x^{-1}(v), z^{-1}(v)) \leq \\ \max_v \left(d_H(x^{-1}(v), y^{-1}(v)) + d_H(y^{-1}(v), z^{-1}(v))\right) \leq \\ \max_v d_H(x^{-1}(v), y^{-1}(v)) + \max_v d_H(y^{-1}(v), z^{-1}(v)) = \\ d(x,y) + d(y,z), \end{aligned}$$

where v takes all values in the common range of the signals.

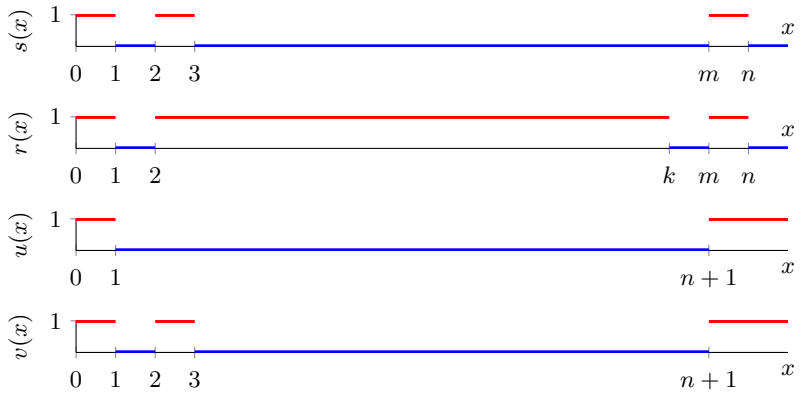

Fig. 4. Metrics on signals and timed words are incomparable

Proposition 4. *The distance d_{ABD} [2] is incomparable with d even up to constant factors, i.e. for any constant K there exist two couples of one-dimensional signals s, r and u, v such that*

$$d(s,r) \geq K \cdot dw(s^{\updownarrow}, r^{\updownarrow}) \text{ and } dw(u^{\updownarrow}, v^{\updownarrow}) \geq K \cdot d(u,v).$$

Proof. To prove the above proposition, we first recall that a timed word over an alphabet Σ is a sequence $(a_1,t_1)(a_2,t_2)\dots(a_n,t_n)$ with $a_i \in \Sigma$ (events) and $0 \leq t_1 \leq t_2 \leq \cdots \leq t_n$ real-valued timestamps. There is a standard way of encoding a (one-dimensional) signal $s : [0,T] \to \{0,1\}$ as a timed word $s^{\updownarrow}$ over the alphabet $\{\uparrow,\downarrow\}$ (rising and falling edges). The word $s^{\updownarrow}$ contains a letter $(\uparrow,t)$ whenever s switches from 0 to 1 at time t, and a letter $(\downarrow,t)$ whenever s switches from 1 to 0. We also use special letters b_0 and b_1 to denote whether the signal starts off at 0 or 1 respectively.

We begin by proving the first assertion,

$$\forall K \in \mathbb{N},\ \exists s, r,\ d(s,r) \geq K \cdot dw(s^{\updownarrow}, r^{\updownarrow})$$

To do so, consider the following two signals, s and r as defined by Fig. 4. Let $s^{\updownarrow}$ and $r^{\updownarrow}$ be their timed word encodings. Let us set $k = 2i+2$, $m = 2i+3$, $n = 2i+4$. Our timed word counterparts for the same are $s^{\updownarrow} = (b_1,0)(\downarrow,1)(\uparrow,2)(\downarrow,3)(\uparrow,2i+3)(\downarrow,2i+4)$ and $r^{\updownarrow} = (b_1,0)(\downarrow,1)(\uparrow,2)(\downarrow,2i+2)(\uparrow,2i+3)(\downarrow,2i+4)$. Computing our signal distance yields $d(s,r) = i$. On the other hand, computing the corresponding distance on the timed words yields $d_{ABD}(s^{\updownarrow}, r^{\updownarrow}) = 2$. Hence for each constant K, setting i to be $3K$ in this example yields $d(s,r) \geq K \cdot d_{ABD}(s^{\updownarrow}, r^{\updownarrow})$.

Now, let us prove by example the second assertion,

$$\forall K \in \mathbb{N},\ \exists u, v,\ dw(u^{\updownarrow}, v^{\updownarrow}) \geq K \cdot d(u,v).$$

To prove existence, consider the signals u, v defined on the same figure and their timed word encodings $u^{\updownarrow}, v^{\updownarrow}$. Our timed word counterparts for the same are $u^{\updownarrow} = (b_1, 0)(\downarrow, 1)(\uparrow, n+1)$ and $v^{\updownarrow} = (b_1, 0)(\downarrow, 1)(\uparrow, 2)(\downarrow, 3)(\uparrow, n+1)$.

Computing our signal distance yields $d(u, v) = 2$. On the other hand, computing the corresponding distance on the timed words yields $d_{ABD}(u^{\updownarrow}, v^{\updownarrow}) = n - 1$. Hence for each constant K, setting n to be $2K + 2$ in this example yields $d_{ABD}(u^{\updownarrow}, v^{\updownarrow}) \geq K \cdot d(u, v)$.

A.2 Algorithm for computing d efficiently

Theorem 1. *Given two signals ω and ω' with size parameters (n, h, b), $d(\omega, \omega')$ can be computed in $\mathcal{O}(hn(h+b))$ (equivalently $\mathcal{O}(h(|\omega| + |\omega'|))$ time.*

Proof. Alg. 2 first partitions the two signals, then for each vector in their range, performs the Hausdorff computation on the vector's preimages. The partitioning step is simple, so the correctness of Alg. 2 lies in the correctness of Alg. 3. Alg. 3 partitions S into S' such that each segment in S' has its associated bucket, and computes the distance segment by segment. By Lemmas 1 and 2, this procedure is correct, hence Alg. 3 is correct, hence, Alg. 2 is correct.

As for complexity, Alg. 3 takes at most $\mathcal{O}(|S| + |R|)$ steps. This is because the for loop takes $|R|$ steps and the while loop takes at most $\mathcal{O}(|S| + |R|)$ steps (either i or j are incremented each pass).

The complexity of Alg. 2 is similarly $\mathcal{O}(|s| + |r|)$. This is because, for each element of the range, it calls Alg. 3 which takes at most $\mathcal{O}(|S|+|R|)$ steps for each preimage pair S, R. So overall, Alg. 2 takes the sum of the number of segments in the preimage of each vector, summed over all vectors, which is exactly the sum of the sizes of the signal. Accounting for the preprocessing step, this makes the overall algorithm take $\mathcal{O}(hn(h+b))$ time.

B Distance-based Temporal Robustness

B.1 Comparing δ with other Temporal Robustness Measures

In this subsection, we compare our temporal robustness measures with others present in the literature, both intuitively and by incomparability.

Let us first define these robustness measures: the synchronous (θ) [5] and asynchronous (η) [17].

Definition 9 (Two measures of temporal robustness). *given a signal ω and an STL formula φ, the directed robustness measures theta$^{\pm}$ and $\eta^{\pm}$ are*

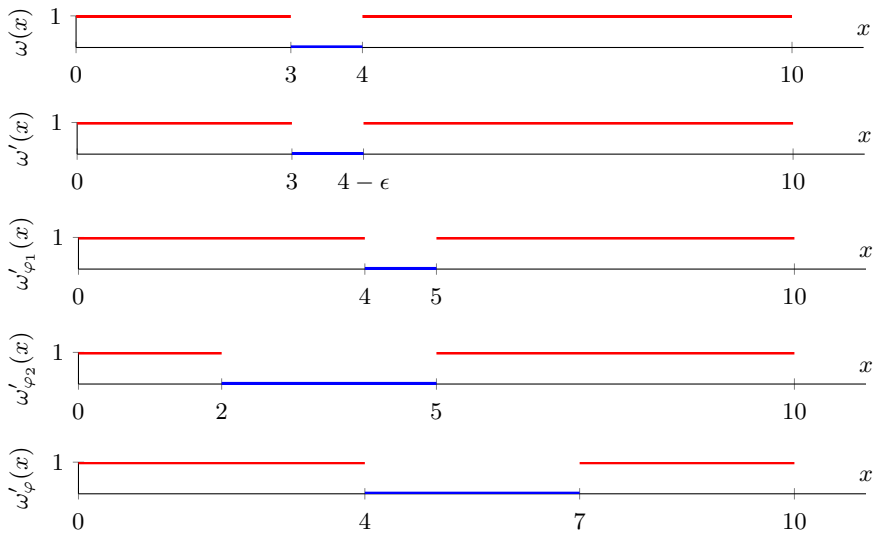

Fig. 5. Comparing robustness measures.

defined inductively as follows :

$$\theta^{\pm}_{p}(\omega,t) = \chi_p(\omega,t)\cdot \sup\{\tau \geq 0 | \forall t' \in \{t\} \pm [0,\tau]\, \chi_p(\omega,t) = \chi_p(\omega,t')\};$$
$$\theta^{\pm}_{\neg\varphi}(\omega,t) = -\theta^{\pm}_{\neg\varphi}(\omega,t);$$
$$\theta^{\pm}_{\varphi_1\vee\varphi_2}(\omega,t) = \max\{\theta^{\pm}_{\varphi_1}(\omega,t), \theta^{\pm}_{\varphi_2}(\omega,t)\};$$
$$\theta^{\pm}_{\varphi_1\ \mathcal{U}_{[a,b]}\ \varphi_2}(\omega,t) = \sup_{t'\in t+[a,b]}\{\min(\inf_{t''\in[t,t']}\{\theta^{\pm}_{\varphi_1}(\omega,t'')\}, \theta^{\pm}_{\varphi_2}(\omega,t'))\},\ \textit{and}$$
$$\eta^{\pm}_{\varphi}(\omega,t) = \chi_\varphi(\omega,t)\cdot \sup\{\tau \geq 0 | \forall t' \in \{t\} \pm [0,\tau] \chi_\varphi(\omega,t) = \chi_\varphi(\omega,t')\}.$$

Their symmetric counterparts are $\theta_\varphi(\omega,t) = \chi_\varphi(\omega,t)\cdot\min\{|\theta^{-}_{\varphi}(\omega,t)|, |\theta^{+}_{\varphi}(\omega,t)|\}$, *and similarly for* η.

We start with two examples that informally demonstrate cases where δ seems to be better at capturing how much a signal needs to be perturbed to start satisfying/failing a formula.

Example 2. Say our specification is the STL formula $\varphi = \Diamond\Box_{[0,1]}\neg p$, and we wish to examine the robustness of the one-dimensional signal ω given on Fig. 5. By the definition above, calculations result in $\eta_\varphi(\omega, 1.5) = 1.5$, but signal ω', very close to ω, does not satisfy φ. This suggests that ω is not very robust even when subject to arbitrarily small perturbations. On the other hand, $\delta(\omega,\varphi) = 0$, which seems to agree with how precarious ω's satisfaction is.

Example 3. Similarly, consider the STL formula $\varphi = \varphi_1 \wedge \varphi_2$ where $\varphi_1 = p\,\mathcal{U}_{[4,5]}\,\neg p$ and $\varphi_2 = \Diamond(\neg p\,\mathcal{U}_{[3,4]}\,p)$. When one computes θ for the initial signal ω, we first see that $\theta_\varphi(\omega,t) = \min\{\theta_{\varphi_1}(\omega,t), \theta_{\varphi_2}(\omega,t)\}$.

Fig. 6. A counterexample

On the one hand, $\theta_{\varphi_1}(\omega, t)$ gives -1, and $\theta_{\varphi_2}(\omega, t)$ comes out to be -1 as well. This suggests that satisfying both φ_1 and φ_1 would be about as difficult since the definition of θ implies that $\theta_{\varphi}(\omega, t) = -1$.

On the other hand, if we study the signal itself, satisfying just φ_1 or just φ_2 can be done using signals ω'_{φ_1} or just ω'_{φ_2} on the same figure. But, in order to satisfy both at once, the closest signal necessarily differs from the original in the entire domain between $[3, 7]$, (see ω'_{φ} on Fig. 5), which intuitively should result in a robustness measure of ω w.r.t. ϕ of at least -2. Meanwhile, $\delta(\omega, \varphi) = -3$ which seems to reflect how satisfying the conjuction is harder than just satisfying either subformula.

Hence, these examples show that the above robustness measures do not capture some reasonable types of perturbations, thereby missing signals that are in some sense very close to a given signal. They also show that these measures are sometimes too optimistic, assuming that satisfying two properties is as hard as satisfying the harder of the two, which in practice does not always hold, since satisfying both at the same time can be much harder than just satisfying either one.

Proposition 5. *$|\delta|$ is incomparable with both $|\eta|$ and $|\theta|$.*

Proof. By incomparable, we mean that none of the following inequalities hold for all signals and formulae :

$$\delta(\omega, \varphi) \geq \theta(\omega, \varphi)$$

$$\delta(\omega, \varphi) \leq \theta(\omega, \varphi)$$
$$\delta(\omega, \varphi) \geq \eta(\omega, \varphi)$$
$$\delta(\omega, \varphi) \leq \eta(\omega, \varphi)$$

We prove this by providing four examples of cases where each of these inequalities fail.

Case 1 : $\exists \omega, \varphi,\ \delta(\omega, \varphi) < \eta(\omega, \varphi)$**.**

The example on Fig. 5 already demonstrated a signal ω and a formula φ such that $|\delta(\omega, \varphi, t)| < |\eta(\omega, \varphi, t)|$.

Case 2 : $\exists \omega, \varphi,\ \delta(\omega, \varphi) > \eta(\omega, \varphi)$**.**

For the other direction, consider $\varphi = \Box\neg p \vee p\, \mathcal{U}_{[7,9]}\ \neg p$, $t = 1.5$ and the signal ω on Fig. 6.

$\eta(\omega, \varphi, 1.5) = -1.5$, as the signal when restricted to $[3, 9]$ satisfies φ. On the other hand, $\delta(\omega, \varphi, t) = 4$ as the only way for a signal that is at a finite distance of ω to satisfy φ is to satisfy $p\, \mathcal{U}_{[7,9]}\, \neg p$. Hence, the two measures of robustness $|\delta|$ and $|\eta|$ are incomparable. **Case 3 :**$\exists \omega, \varphi,\ \delta(\omega, \varphi) > \theta(\omega, \varphi)$**.**

As for comparison with θ, similarly, we note that the example on Fig. 5 shows a case where $|\delta(\omega, \varphi, t)| > |\theta(\omega, \varphi, t)|$.

Case 4 :$\exists \omega, \varphi,\ \delta(\omega, \varphi) < \theta(\omega, \varphi)$**.**

For the other direction, consider $\varphi = \Box\neg p \vee p\, \mathcal{U}_{[7,9]}\, \neg p$, $t = 1.5$ and the signal ω as in the previous proof. It is known that $|\theta(\omega, \varphi, t)| \leq |\eta(\omega, \varphi, t)|$ (Theorem 4.10, [17]), so since $|\eta(\omega, \varphi, t)| < |\delta(\omega, \varphi, t)|$, we have that in this example

$$|\theta(\omega, \varphi, t)| < |\delta(\omega, \varphi, t)|.$$

B.2 Hardness of computing δ

Theorem 3. *Given a signal ω with size parameters (n, h, b) and a property $\varphi \in STL_r$, computing $\delta(\omega, \varphi)$ is in $\mathcal{O}(n(h + b) \cdot |\varphi|)$-time.*

In order to do so, we analyse the functions that Alg. 4 (also reproduced here) calls.

Algorithm 4 Computing $\delta(\omega, \varphi)$ where $\varphi \in \mathrm{STL}_r$

```
Let ω : [0, T] → {0, 1}^d, sign ← −1
if (φ = φ1 ∨ φ2) ∧ (dom(φ1) ∩ dom(φ2) ≠ ∅) then
    return max(δ(ω, φ1), δ(ω, φ2))
else if (φ = ¬φ′) ∧ (|φ′| < |φ|) then
    return −(δ(ω, φ′))
if φ = B, □_[a,b] B, B1 U_[a,b] B2, □(B1 ⟹ ◇_[0,b] B2) then
    if φ = B, □_[a,b] B then
        (ω, φ) ← colour(ω, φ, B)
    else
        (ω, φ) ← colour(ω, φ, B1, B2)
    if ω ⊨ φ then
        sign ← +1, φ ← ¬φ
    return sign·d(ω, φ)
```

We begin with the colouring algorithm, Alg. 6.

B.3 Colouring the signal

Recall that every h-dimensional signal is a function $s : [0, T] \to \{0, 1\}^{|P|}$. The distance we have defined over signals treats every vector in the range of the signals separately, and computes the Hausdorff distance on each such level before taking the overall maximum. Each vector in the range of the signal, however, represents an assignment for all the propositional variables in P, and one can

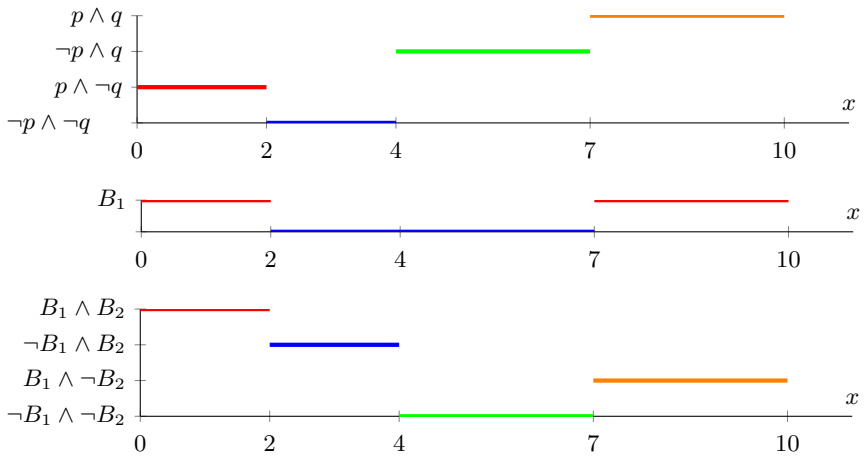

Fig. 7. A signal over $\{p, q\}$; its colourings over $B_1 = p$ and over $B_1 = p, B_2 = \neg q$

evaluate B on each such assignment. Hence, for every level of the signal in the flattened form, either B is true or false, so we can colour each of those segments either red or blue respectively. In this manner, if there is just one B in the question, we can effectively view the d dimensional signal with potentially 2^d different values, as a one-dimensional signal. On the other hand, if there are two such terms to consider, B_1 and B_2, then we consider all four possible combinations for the two, namely using a separate colour for each segment that satisfies $B_1 \wedge B_2$, $B_1 \wedge \neg B_2$, $\neg B_1 \wedge B_2$ and $\neg B_1 \wedge \neg B_2$. Note that no assignment can satisfy two distinct colours, and the combinations are cumulatively exhaustive, hence each level gets exactly one colour.

Example 4. Fig. 7 represents a two-dimensional signal and two colourings thereof.

Clearly, the colouring procedure takes time linear in the size of the signal times the size of B each time, which is $\mathcal{O}(|\omega| \cdot |\varphi|)$

The purpose of this precomputation is to simplify the δ computation moving forward, as we can now effectively use the recoloured version, and treat various combinations of the relevant Boolean terms B almost like atomic propositions. This is sound, as the distance from the original signal to the language of the formula is the same as distance of the recoloured signal to the language of the formula over the space of signals that use B_i as the propositional variables. For our purposes, we only need how to colour for two terms at a time, as our fragment does not necessitate more.

Lemma 4 (Colouring Lemma). *Consider a signal $\omega : [0, T] \to \{0, 1\}^{|P|}$, and an STL_r formula φ that contains $n > 0$ maximal Boolean combinations of propositional variables $B_1, \ldots B_n$. Denote the coloured, 2^n-leveled version of ω one obtains using Alg. 6 by $\pi(\omega) : [0, T] \to \{0, 1\}^n$. Take n fresh propositional*

Algorithm 6 Colouring Algorithm

Let $\omega : [0,T] \to \{0,1\}^{|P|}$ be the signal
Let $\{B_1, \dots B_n\}$ be Boolean combinations of variables in P.
$R \leftarrow \text{Range}(\omega)$
$\forall a \in \{0,1\}^n \, S_a \leftarrow \emptyset$ ▷ *These store preimages of values in* ω'
for all $v \in R$ **do**
    $\forall i \in [n]\,, b_i \leftarrow (v \vDash B_i)$
    $S_{b_1 \dots b_n} \leftarrow \omega^{-1}(v)$
$\forall a \in \{0,1\}^n\,, \forall\, t \in S_a : \omega'(t) = a$
$\varphi' \leftarrow \varphi[\forall i \in [n],\, B_i \mapsto p_i]$
return (ω', φ')

variables $p'_1, \dots p'_n$*, and consider the simplified version of* φ *that rewrites each* B_i *with the proposition* p_i*. Call this simplified formula* $c(\varphi)$*, i.e.,* $c(\varphi) = \varphi[\forall i \in [n] : B_i \mapsto p_i]$*. Then, we have that*

$$\delta(\omega, \varphi) = \delta(\pi(\omega'), c(\varphi')).$$

Proof. Let the map $\pi : \{0,1\}^{|P|} \to \{0,1\}^{|B|}$ that takes vectors $v \in \{0,1\}^{|P|}$ to their valuations under each B_i, i.e.,

$$\pi(v)_i = 1 \iff v \vDash B_i.$$

The colouring map composes π after the original signal, i.e. $\mathsf{colour}(\omega) = \omega \circ \pi = \pi(\omega)$.

We seek to prove that

$$\delta(\omega, \varphi) = \delta(\pi(\omega'), c(\varphi)).$$

In order to do so, we first prove that $sign(\delta(\omega, \varphi, t)) = sign(\delta(\pi(\omega'), c(\varphi), t))$.

And then we prove that $d(\omega, \varphi) = d(\pi(\omega'), c(\varphi))$.

1. Checking that the signs of $\delta(\omega, \varphi)$ and $\delta(\pi(\omega'), c(\varphi))$ match.

To do so, we show that if $\omega \vDash \varphi \iff \pi(\omega) \vdash c(\varphi)$, by induction on the structure of φ.

Base case : $\varphi = B$**.** In this case, the claim clearly holds.

Inductive hypothesis : Assume for all subformulae ψ of ϕ, the claim holds.

Case 1 :$\varphi = \neg\varphi_1$**.**

In this case, since $\omega \vDash \varphi_1 \iff \pi(\omega) \vdash c(\varphi_1)$ the claim holds.

Case 2 :$\varphi = \varphi_1 \vee \varphi_2$**.**

In this case, since $\omega \vDash \varphi_1 \iff \pi(\omega) \vdash c(\varphi_1)$ and $\omega \vDash \varphi_2 \iff \pi(\omega) \vdash \varphi'_2$, if either one is true the coloured result is true, and if neither hold, then neither can the coloured result.

Case 3 :$\varphi = B_1 \, \mathcal{U}_{[a,b]} \, B_2$**.**

In this case, there exists a point $t \in [a,b]$ such that $\forall t' \in [0,t]\, \omega(t') \vDash B_1$ and $\omega(t) \vDash B_2$. This implies that $\forall t' \in [0,t]\, \pi(\omega)(t') \vDash p_1$ and $\pi(\omega)(t) \vDash p_2$ implying that $\pi(\omega) \vDash p_1 \, \mathcal{U}_{[a,b]} \, p_2$.

For the other direction, similarly if there exists a point $t \in [a, b]$ such that $\forall t' \in [0, t]\, \pi(\omega)(t') \vDash p_1$, then $\forall t' \in [0, t]\, \omega(t') \vDash B_1$ as that is exactly when $\pi(\omega) \vDash B_1$, and $\pi(\omega)(t) \vDash p_2$ implies that $\omega(t) \vDash B_2$, so $\omega \vDash B_1\, \mathcal{U}_{[a,b]}\, B_2 \iff \pi(\omega) \vDash p_1\, \mathcal{U}_{[a,b]}\, p_2$.

This concludes the induction, showing that for all signals ω and STL_r formulae φ, the result holds.

2. The distance from a signal to a language is also preserved : $d(\omega, \varphi) = d(\pi(\omega'), c(\varphi))$.

In this section, we assume without loss of generality that $\omega \nvDash \varphi$. In order to prove the above, we first try to show that

$$d(\omega, \varphi) \geq d(\pi(\omega), c(\varphi)).$$

($\geq$) Consider any two signals $\omega, \omega' : [0, T] \to \{0, 1\}^{|P|}$, let $d(\omega, \omega') = x$. Then, for every $t \in [0, T]$, $\exists t'$ such that $|t - t'| \leq x$ and $\omega(t) = \omega'(t')$ and vice versa. But, $\omega(t) = \omega'(t') \implies \pi(\omega(t)) = \pi(\omega'(t'))$, so we can conclude that $d(\pi(\omega), \pi(\omega')) \leq x$, i.e., $d(\pi(\omega), \pi(\omega')) \leq d(\omega, \omega')$.

This means that for any pair of signals, colouring them cannot make them further apart. Now, consider a signal ω and a formula φ. By definition, the following two equations hold.

$$d(\omega, \varphi) = \inf_{\beta \in \mathcal{L}(\varphi)} (d(\omega, \beta))$$
$$d(\pi(\omega), c(\varphi)) = \inf_{\gamma \in \mathcal{L}(c(\varphi))} (d(\omega, \gamma))$$

From the fact that colouring preserves the sign of δ, we know that $\pi(\mathcal{L}(\varphi)) = \mathcal{L}(c(\varphi))$.

But, by the previous observation, we know that

$$\forall \beta \in \mathcal{L}(\varphi),\, d(\omega, \beta) \geq d(\pi(\omega), \pi(\beta))$$
$$\implies d(\omega, \varphi) \geq d(\pi(\omega), c(\varphi)).$$

($\leq$) Secondly, suppose $d(\pi(\omega), c(\varphi)) = x$. Then, for every $\epsilon \geq 0$, consider a signal $\mu : [0, T] \mapsto \{0, 1\}^{|B|}$ such that $\mu \in \mathcal{L}(c(\varphi))$ and $d(\pi(\omega)) \leq x + \epsilon$. We will define a signal ω' with values in $\{0, 1\}^{|P|}$ that is meant to represent the preimage of μ that is closest to ω. We define this signal, $\omega' : [0, T] \mapsto \{0, 1\}^{|P|}$ as follows :

$$\forall t \in [0, T],\, \omega'(t) = \underset{v |\, \pi(v) = \pi(\mu(t))}{\arg\min} \{\overrightarrow{d_H}(\{t\}, \omega^{-1}(v))\}.$$

Next, we seek to show that $d(\omega, \varphi) \leq x + \epsilon$. We will now show that ω' is a satisfying signal in $\mathcal{L}(\varphi)$, such that $d(\omega, \omega') \leq x + \epsilon$.

We first show that

$$\overrightarrow{d_H}(\omega, \omega') \leq x + \epsilon.$$

For all $t \in [0, T]$, if $\omega(t) = v$, then there exist (possibly) several t' such that $|t - t'| \leq x + \epsilon$, and $\mu(t') = \pi(v)$. Consider the timestamp t' that minimises

$|t - t'|$. Then, $\omega'(t') = v$. This is because if $\omega'(t') = v' \neq v$, then we know that $\pi(v') = \pi(v)$ but $\overrightarrow{d_H}(\{t'\}, \omega^{-1}(v)) \geq \overrightarrow{d_H}(\{t'\}, \omega^{-1}(v'))$, which contradicts the fact that t' was the closest to t.

We now show that

$$\overrightarrow{d_H}(\omega', \omega) \leq x + \epsilon.$$

On the other hand, if $t \in [0, T]$, if $\omega'(t) = v$, then there exist (possibly) several t' such that $|t - t'| \leq x + \epsilon$, and $\pi(\omega(t')) = \pi(v)$. Consider the timestamp t' that minimises $|t - t'|$. Then, $\omega(t') = v$ by definition.

Hence, we conclude that

$$\forall \epsilon,\ \exists \omega' \in \mathcal{L}(\varphi),\ d(\omega, \omega') \leq x + \epsilon$$

$$\implies d(\omega, \mathcal{L}(\varphi)) \leq x.$$

Hence, $d(\omega, \varphi) \leq d(\pi(\omega), c(\varphi))$, and hence, we see that the distance to the language is preserved.

As both the sign and the distance are preserved, δ is preserved under colouring.

Now that we have this lemma, we can conduct the preprocessing step and focus on computing δ for the following, much simpler set of cases. Note that in Alg. 4 we only ever run alg:colour with up to 2 Boolean terms, so even though in general for n Boolean terms the running time of this algorithm is $\mathcal{O}(2^n|\varphi| \cdot |\omega|)$, n is never greater than 2 in STL_r, so each call takes time $\mathcal{O}(|\varphi| \cdot |\omega|)$.

B.4 Auxiliary Constructions

Before we proceed with these algorithms for computing δ on STL_r, we take a moment to study a few constructions on signals that will assist in computing δ. We will argue that given a signal with n segments, each of these constructions can be computed in $\mathcal{O}(n)$-time, so we can use them in our computations.

Spikes. Given a signal ω and a formula φ such that $\omega \not\models \varphi$, when computing $d(\omega, \mathcal{L}(\varphi))$, often there will not exist a single signal $\omega' \in \mathcal{L}(\varphi)$ such that $d(\omega, \omega') = d(\omega, \mathcal{L}(\varphi))$. Instead, the witness to the fact that the distance between ω and $\mathcal{L}(\varphi)$ is at most $d(\omega, \mathcal{L}(\varphi))$ is a limit point of the language, which is to say a sequence of signals whose distances to ω tend to $d(\omega, \mathcal{L}(\varphi))$. The simplest type of such limit phenomena in our metric space is what we call a *spiked signal*, an object that resembles an existing signal exactly, except for its value at one time instant, where it has a spike (or a segment of length almost zero). This object is the limit of a sequence of signals that only differ around vanishingly small neighbourhoods of one fixed time instant.

Definition 10 (Spike). *Given a signal $\omega : [0, T]$, a time instant $t_s \in [0, T]$, and a value $v \neq \omega(t_s)$. We define Spike(ω, t_s, v) to be a sequence of signals $\sigma = \{s_i\}_{i \in \mathbb{N}}$ over the same domain such that*

$$\forall i \in \mathbb{N}, s_i(t) = \begin{cases} v & |t - t_s| \leq \frac{1}{2^i}; \\ \omega(t) & \text{otherwise.} \end{cases}$$

Using the following lemma, whenever we want to construct a limit point that differs only at one time instant from a given signal, we can simply use the spiked version of the signal, and we know how to compute distances to this limiting sequence.

Lemma 5. *For any two signals ω, α over the domain $[0,T]$ such that $Range(\omega) = Range(\alpha)$, a time instant $t_s \in [0,T]$, and value $v \neq \omega(t_s)$, it holds that*

$$d(\alpha, Spike(\omega, t_s, v)) = \max(\min_{\alpha(t)=v} |t - t_s|, d(\omega, \alpha)).$$

Proof. Begin by noting that if the value of the spike, v, is not in the range of α, then the above statement is true as both sides evaluate to infinity. Henceforth, we assume that $v \in Range(\alpha)$.

Let $\omega_s = Spike(\omega, t_s, v)$. We wish to prove that

$$d(\omega_s, \alpha) = \max(d(\omega, \alpha), \min_{\alpha(t)=v} |t - t_s|)$$

We first remark that given two subsets S, R of $\mathbb{R}$, the Hausdorff distance between sets is the distance between the closures of the sets, i.e.

$$d_H(S, R) = d_H(\overline{S}, \overline{R})$$

and that hence removing a limit point p of S will not change the Hausdorff distance, i.e.,

$$d_H(S, R) = d_H(S - \{p\}, R)$$

Secondly, if one adds a point p to S, then using the properties of the supremum operator, the distance increases, if at all, by exactly the distance of this new point to R, i.e.

$$d_H(S \cup \{p\}, R) = \max(d_H(\{p\}, R), d_H(S, R))$$

Now, we notice that by definition,

$$d(\omega_s, \alpha) = \max_{u \in Range(\omega_s)} \sup_{\omega_s(x)=u} \inf_{\alpha(y)=u} |x - y|$$

and

$$d(\omega, \alpha) = \max_{u \in Range(\omega_s)} \sup_{\omega(x)=u} \inf_{\alpha(y)=u} |x - y|$$

Let $\omega(t_s) = v_0$. The above two equations only differ on the component of the maximum taken over the vectors v and v_0, so we will focus on these. We also note that since $\omega^{-1}(v_0) = \omega_s^{-1}(v_0) - \{t_s\}$,

$$d_H(\omega^{-1}(v_0), \alpha^{-1}(v_0)) = d_H(\omega_s^{-1}(v_0), \alpha^{-1}(v_0))$$

In addition, since $\omega_s^{-1}(v) = \omega^{-1}(v) \cup t_s$,

$$d_H(\omega_s^{-1}(v), \alpha^{-1}(v)) = \max(d_H(\{t_s\}, \alpha^{-1}(v)))d_H(\omega_s^{-1}(v), \alpha^{-1}(v))$$

Hence,

$$d(\omega_s, \alpha) = \max(d(\omega, \alpha), \min_{\alpha(t)=v} |t - t_s|)$$

Clearly we can add multiple spikes to a signal so long as we make sure we place them at distinct time instants, as the above argument only cares about a small enough neighbourhood around each spike.

Other operations are defined in Table 1.

Operation name	Notation	Definition
Next Value Match	$\mathsf{Next}(\omega, v)$	$g(t) = \min\{t' \geq t \mid \omega(t') = v\}$
Previous Value Match	$\mathsf{Prev}(\omega, v)$	$g(t) = \max\{t' \leq t \mid \omega(t') = v\}$
Current Value Segment	$\mathsf{CurrSeg}(\omega, v)$	$g(t) = \mathsf{Next}(\omega, \neg v)(t) - \mathsf{Prev}(\omega, \neg v)(t)$
Constant Function	$\mathsf{Const}(a)$	$g(t) = a$
Time Dilation	$\mathsf{TDilate}(f, a)$	$g(t) = f\left(\frac{t}{a}\right)$
Space Dilation	$\mathsf{SDilate}(f, a)$	$g(t) = \frac{f(t)}{a}$
Time Translation	$\mathsf{TTranslate}(f, a)$	$g(t) = f(t - a)$
Space Translation	$\mathsf{STranslate}(f, a)$	$g(t) = f(t) + a$
Identity Difference	$\mathsf{IdDiff}(f)$	$g(t) = t - f(t)$
Maximum	$\mathsf{Max}(f, g)$	$g(t) = \max(f(t), g(t))$
Minimum	$\mathsf{Min}(f, g)$	$g(t) = \min(f(t), g(t))$
Anti-value	$\mathsf{Anti}(f)$	$g(t) = -f(t)$

Table 1. Definition of operations. Given a signal $\omega : [0, T] \to \{0, 1\}^h$, a function $f : [0, T] \to \mathbb{R}$, a value $v \in \{0, 1\}^h$, and $a \in \mathbb{R}$, the operations produce a function $g : [0, T] \to \mathbb{R}$ as defined in the third column.

Given a signal (or a piece-wise linear function $f : [0, T] \to \mathbb{R}$, henceforth known as a *plot*) with n segments, it is easy to see that each of the operations in *Table* 1 aside from Max or Min can be computed in $\mathcal{O}(n)$-time, and returns a signal (respectively, a plot) with at most n segments. Given two plots with m and n segments respectively, Max and Min can be computed in $\mathcal{O}(m + n)$-time, and returns a plot with at most $m + n$ segments.

With the notation developed in Table 1, we can define a generalisation of the function close defined in Sect. 3 for subsets of $\mathbb{R}$ to signals. $\mathsf{Closest}(\omega, t, \alpha)$ is the closest point to t in α that matches in value to $\omega(t)$. If no singular point exists, the limit of a sequence of points that match is returned.

$$\mathsf{Closest}(\omega, t, \alpha) = \begin{cases} \mathsf{Prev}(\alpha, \omega(t))(t) & t - \mathsf{Prev}(\alpha, \omega(t))(t) < \mathsf{Next}(\alpha, \omega(t))(t) - t \\ \mathsf{Next}(\alpha, \omega(t))(t) & \textit{otherwise} \end{cases}$$

Now, we focus on computing δ for STL_r formulae. As seen in Appendix B.3, we can assume these signals and formulae have already been coloured, so the problem has been reduced to computing $d(\omega, \varphi)$ where ω is at most 2-dimensional, $\omega \not\models \varphi$ and φ is assumed without loss of generality to be one of the following seven formulae:

$$\{p, \Diamond_{[a,b]} p, \Box_{[a,b]} p, p\, \mathcal{U}_{[a,b]}\, q, \neg(p\, \mathcal{U}_{[a,b]}\, q), G(p \Rightarrow \Diamond_{[0,b]} q), \neg(G(p \Rightarrow \Diamond_{[0,b]} q))\}.$$

We provide individual algorithms for computing $d(\omega, \varphi)$, in each case, prove their correctness, and demonstrate that they also run in $\mathcal{O}(|\omega|)$ - time.

Before we embark on proving the correctness of individual algorithms, we provide here a blueprint of the general logic present in all of these proofs, for easier parsing.

Blueprint for proving that the result of an algorithm $d(\omega, \varphi) \geq Res$. There are two elements to any such proof, broadly.

1. The first step is to show that $\forall \beta \in \mathcal{L}(\varphi)$, $d(\omega, \beta) \geq Res$. This ensures that the overall distance $d(\omega, \varphi) \geq Res$.
2. The second step is ensuring that Res is a feasible distance to ω from either a signal or a limiting sequence of signals $\alpha \in \bar{(}\mathcal{L}(\varphi))$. This ensures that the overall distance $d(\omega, \varphi) \geq Res$ since

$$d(\omega, \varphi) = \inf_{\beta \in \mathcal{L}(\varphi)} (d(\omega, \beta)) \leq d(\omega, \alpha) = Res.$$

With these two steps satisfied, we can conclude that $d(\omega, \varphi) \geq Res$. All the proofs of correctness from here on out follow this broad structure.

B.5 Propositions, Finally and Globally

In the next three subsections, let $\omega = \{(0, a_1, t_1)(t_1, a_2, t_2) \dots (t_{n-1}, a_n, t_n = T)\}$ where $\forall\, i \in [n] : a_i \in \{0, 1\}$ or $\forall\, i \in [n] : a_i \in \{0, 1\}^2$ as is applicable.

Algorithm 5 $\mathsf{d}(\omega, \varphi)$ where $\varphi = p(a)$

1: $l \leftarrow \mathsf{IdDiff}(\mathsf{Prev}(\omega, p))(a)$
2: $r \leftarrow \mathsf{Anti}(\mathsf{IdDiff}(\mathsf{Next}(\omega, p)))(a)$
3: **return** $d = \min(l, r)$

Lemma 6. *Given a signal $\omega : [0, T] \to \{0, 1\}$ and a formula $\varphi = p$ such that $\omega \not\models \varphi$, Alg. 5 computes $d(\omega, \mathcal{L}(\varphi))$, and runs in $\mathcal{O}(n)$ time.*

Proof. In order to prove that $d(\omega, p) = \min(\mathsf{Next}(\omega, p)(a) - a, a - \mathsf{Prev}(\omega, p)(a))$, we first prove that $d(\omega, p) \geq \min(\mathsf{Next}(\omega, p)(a) - a, a - \mathsf{Prev}(\omega, p)(a))$, and then prove that there exists an element or a limit point of $\mathcal{L}(\varphi)$ such that its distance to ω is $\min(\mathsf{Next}(\omega, p)(a) - a, a - \mathsf{Prev}(\omega, p)(a))$.

1. Given a signal ω and a time instant $a \in [0, T]$ such that $\omega(a) \neq 1$, consider any signal α such that $\alpha \vDash p(a)$. $\mathsf{Closest}(\alpha, a, \omega)$ must either be the next p match after a, or the previous one. Hence, $\forall \alpha \in \mathcal{L}(\varphi)$, $d(\omega, \alpha) \geq \min(\mathsf{Next}(\omega, p)(a) - a, a - \mathsf{Prev}(\omega, p)(a))$

$$d(\omega, p) \geq \min(\mathsf{Next}(\omega, p)(a) - a, a - \mathsf{Prev}(\omega, p)(a))$$

2. $\omega_s = Spike(\omega, a, p)$ is a limit point of the set of signals satisfying $p(a)$, and $d(\omega, \omega_s) = \min(\mathsf{Next}(\omega, p)(a) - a, a - \mathsf{Prev}(\omega, p)(a))$

$$\implies d(\omega, p) \leq \min(\mathsf{Next}(\omega, p)(a) - a, a - \mathsf{Prev}(\omega, p)(a))$$

Hence, we can conclude that

$$d(\omega, p) = \min(\mathsf{Next}(\omega, p)(a) - a, a - \mathsf{Prev}(\omega, p)(a)).$$

Algorithm 6 $\mathsf{d}(\omega, \varphi)$ where $\varphi = \Diamond_{[a,b]} p$

1: $l \leftarrow \mathsf{IdDiff}(\mathsf{Prev}(\omega, p))(a)$
2: $r \leftarrow \mathsf{Anti}(\mathsf{IdDiff}(\mathsf{Next}(\omega, p)))(b)$
3: **return** $d = \min(l, r)$

Lemma 7. *Given a signal $\omega : [0, T] \to \{0, 1\}$ and a formula $\varphi = \Diamond_{[a,b]} p$ such that $\omega \not\models \varphi$, Alg. 6 computes $d(\omega, \mathcal{L}(\varphi))$, and runs in $\mathcal{O}(n)$ time.*

Proof. In order to prove that $d(\omega, p) = \min(\mathsf{Next}(\omega, p)(b) - b, a - \mathsf{Prev}(\omega, p)(a))$, we first prove that $d(\omega, p) \geq \min(\mathsf{Next}(\omega, p)(b) - b, a - \mathsf{Prev}(\omega, p)(a))$, and then prove that there exists an element or a limit point of $\mathcal{L}(\varphi)$ such that its distance to ω is $\min(\mathsf{Next}(\omega, p)(b) - b, a - \mathsf{Prev}(\omega, p)(a))$.

1. Given a signal ω and an interval $[a, b] \subseteq [0, T]$ such that $\forall t \in [a, b]\ \ \omega(t) \neq 1$, consider any signal α such that $\alpha \vDash \Diamond_{[a,b]} p$.
   $\mathsf{Closest}(\alpha, a, \omega)$ must the previous p-value in ω, and similarly $\mathsf{Closest}(\alpha, b, \omega)$ must the next match for p in ω .
   Hence, $\forall \alpha \in \mathcal{L}(\varphi)$, $d(\omega, \alpha) \geq \min(\mathsf{Next}(\omega, p)(b) - b, a - \mathsf{Prev}(\omega, p)(a))$

   $$d(\omega, p) \geq \min(\mathsf{Next}(\omega, p)(b) - b, a - \mathsf{Prev}(\omega, p)(a))$$

2. Without loss of generality let $\mathsf{Next}(\omega, p)(b) - b \geq a - \mathsf{Prev}(\omega, p)(a)$.
   $\omega_s = Spike(\omega, a, p)$ is a limit point of the set of signals satisfying $\Diamond_{[a,b]} p$, and $d(\omega, \omega_s) = \min(\mathsf{Next}(\omega, p)(b) - b, a - \mathsf{Prev}(\omega, p)(a))$, so

   $$d(\omega, p) \leq \min(\mathsf{Next}(\omega, p)(b) - b, a - \mathsf{Prev}(\omega, p)(a)).$$

   Hence, we can conclude that

   $$d(\omega, p) = \min(\mathsf{Next}(\omega, p)(b) - b, a - \mathsf{Prev}(\omega, p)(a)).$$

Lemma 8. *Given a signal $\omega : [0, T] \to \{0, 1\}$ and a formula $\varphi = \Box_{[a,b]} p$ such that $\omega \not\models \varphi$, Alg. 7 computes $d(\omega, \mathcal{L}(\varphi))$, and runs in $\mathcal{O}(n)$ time.*

Proof. In order to prove that $d(\omega, p) = \max(lc, rc, lm, rm)$, we first prove that $d(\omega, p) \geq \max(lc, rc, lm, rm)$, and then prove that there exists an element or a limit point of $\mathcal{L}(\varphi)$ such that its distance to ω is $\max(lc, rc, lm, rm)$.

Algorithm 7 $\mathsf{d}(\omega, \varphi)$ where $\varphi = \Box_{[a,b]} p$

$lm \leftarrow \mathsf{Prev}(\omega, \neg p)\left(\frac{a+b}{2}\right) - a$
$rm \leftarrow b - \mathsf{Next}(\omega, \neg p)\left(\frac{a+b}{2}\right)$
$lc \leftarrow \min(\mathsf{Next}(\omega, p)(a) - a, \frac{\mathsf{CurrSeg}(\omega, \neg p)(a)}{2})$
$rc \leftarrow \min(b - \mathsf{Prev}(\omega, p)(b), \frac{\mathsf{CurrSeg}(\omega, \neg p)(b)}{2})$
return $d = \min(lm, rm, lc, rc)$

1. Since $\omega \not\models \varphi$, we know there exists some point t_0 in $[a, b]$ that does not satisfy p. $\forall \alpha \models \varphi$, the closer t_0 is to the center of the interval $[a, b]$, the greater $d(\omega, \alpha)$ will be, as $\mathsf{Closest}(\omega, t_0, \alpha)$ must be outside $[a, b]$. If we pick the t_0 to be the point in $[a, b]$ that is as close to $\frac{a+b}{2}$ while not satisfying p, then $|\mathsf{Closest}(\omega, t_0, \alpha) - t|$ is exactly $\max(lm, rm)$.
   In addition, suppose there is a segment $[u, \neg p, v)$ in ω such that $v - u = c$, and this segment contains exactly one endpoint of the interval, suppose without loss of generality it is incident on a. In α, $[a, v)$ evaluates to p, and the closest for any of these points is either v or u. Calculating the distance incurred in this segment, we find
   $$\overrightarrow{d_H}([a, v], \omega^{-1}(p)) \geq \min\left(\frac{c}{2}, v - a\right)$$
   as either all the points find their closest at v, or they split appropriately between u and v. This is computed by lc and rc.
   Hence, $\forall \alpha \in \mathcal{L}(\varphi)$, $d(\omega, \alpha) \geq \max(lc, rc, lm, rm)$
   $$d(\omega, p) \geq \max(lc, rc, lm, rm)$$
2. Consider the constant signal $\mathsf{Const}(p)$ over the domain $[0, T]$.
   Given these, we construct the signal $\omega_1 = \omega|_0^a \cdot \mathsf{Const}(p)|_a^b \cdot \omega|_b^D$. We then add spikes of value $\neg p$ at a and b. Let the result of these operations be α.
   We claim that $d(\omega, \alpha) = \max(lc, rc, lm, rm)$ as defined in the algorithm. For every timestamp outside $[a, b]$, there are instantaneous matches in either signal. For ω, the timestamps that don't have instantaneous matches are those in $[a, b]$ with value $\neg p$. Their closest points are either a or b (at the spikes) in α, and this maximises precisely at $\max(lm, rm)$. For α, the timestamps that don't have instantaneous matches are the ones that used to have value $\neg p$ in ω, in $[a, b]$, and the spikes. The distance incurred by the spikes is not more than $\max(lm, rm)$. For every segment $[u, v)$ with value $\neg p$ in ω between $[a, b]$, if the segment was wholly inside $[a, b]$, it is either to the left of $\mathsf{Prev}(\omega, \neg p)\left(\frac{a+b}{2}\right)$ or to the right of $\mathsf{Next}(\omega, \neg p)\left(\frac{a+b}{2}\right)$. Either ways, the distance such a segment incurs at worst, is $\frac{v-u}{2}$ which is less than lm or rm, as is applicable. Lastly, if the segment is not wholly within $[a, b]$, then it contains either a or b, and the distance it incurs to its closest points is exactly what is accounted for with $\max(lc, rc)$. Hence, $d(\omega, \alpha)$ is exactly $\max(lc, rc, lm, rm)$, where α is a limit point of the language.
   $$\implies d(\omega, p) \leq \max(lc, rc, lm, rm)$$

Hence, we can conclude that

$$d(\omega, p) = \max(lc, rc, lm, rm).$$

For the rest of the fragment, we will often wish to know the cost of ensuring that a subformula φ holds for a constant length segment, but not be sure where said segment of time begins or ends. This introduces the idea of calculating $d(\omega, \varphi(t))$ where t is a parameter that determines the domain of the formula $\varphi = p, F, G$ as the case may be. In the settings that will be relevant, the result of $d(\omega, \varphi(t))$ is a plot, which we can calculate using our predefined auxiliary constructions, still in linear time in the size of the input signal. Below, we present some algorithms for the same.

Algorithm 8 $\mathsf{d}(\omega, \varphi)$ where $\varphi = p(t)$

1: $LPlot \leftarrow \mathsf{IdDiff}(\mathsf{Prev}(\omega, p))$
2: $RPlot \leftarrow \mathsf{Anti}(\mathsf{IdDiff}(\mathsf{Next}(\omega, p)))$
3: **return** $d = \mathsf{Min}(LPlot, RPlot)$

Lemma 9. *Given a signal $\omega : [0, T] \to \{0, 1\}$ and a formula $\varphi = p$ such that $\omega \not\models \varphi$, Alg. 8 computes the function $d(\omega, \mathcal{L}(\varphi))$, and runs in $\mathcal{O}(n)$ time.*

Proof. At each value of t, this algorithm computes

$$\min(\mathsf{Next}(\omega, p)(t) - t, t - \mathsf{Prev}(\omega, p)(t))$$

in agreement with the value that Alg. 5 yields. Clearly this takes $\mathcal{O}(n)$ time, and the resulting plot has $\mathcal{O}(n)$ segments since the set of segments of Prev and Next have the same endpoints.

Algorithm 9 $\mathsf{d}(\omega, \varphi)$ where $\varphi = \Diamond_{[t+a,t+b]} p$

1: $\omega_a \leftarrow \mathsf{TTranslate}(\omega, -a)$
2: $LPlot \leftarrow \mathsf{IdDiff}(\mathsf{Prev}(\omega_a, p))$

3: $\omega_b \leftarrow \mathsf{TTranslate}(\omega, -b)$
4: $RPlot \leftarrow \mathsf{Anti}(\mathsf{IdDiff}(\mathsf{Next}(\omega_b, p)))$

5: **return** $d = \mathsf{Min}(LPlot, Rplot)$

Lemma 10. *Given a signal $\omega : [0, T] \to \{0, 1\}$ and a formula $\varphi = \Diamond_{[t+a,t+b]} p$ such that $\omega \not\models \varphi$, Alg. 9 computes the function $d(\omega, \mathcal{L}(\varphi))$, and runs in $\mathcal{O}(n)$ time.*

Proof. At each value of t, this algorithm computes

$$\min(\mathsf{Next}(\omega, p)(t+b) - t + b, t + a - \mathsf{Prev}(\omega, p)(t+a))$$

in agreement with the value that Alg. 6 yields. Clearly this takes $\mathcal{O}(n)$ time, and the resulting plot has $\mathcal{O}(n)$ segments since each of Prev and Next have n segments.

In case only one endpoint of F's domain varies, the corresponding plot ($LPlot$ or $RPlot$) would just receive the constant value (l or r respectively) computed in Alg. 6.

Algorithm 10 $\mathsf{d}(\omega, \varphi(\mathsf{t}))$ where $\varphi = \Box_{[t+a,t+b]}p$

1: $\omega_m \leftarrow \mathsf{TTranslate}(\omega, -\frac{a+b}{2})$
2: $MPrev \leftarrow \mathsf{Prev}(\omega_m, \neg p)$
3: $LmPlot \leftarrow \mathsf{STranslate}(\mathsf{Anti}(\mathsf{IdDiff}(MPrev)), \frac{b-a}{2})$

4: $MNext \leftarrow \mathsf{Next}(\omega_m, \neg p)$
5: $RmPlot \leftarrow \mathsf{STranslate}(\mathsf{IdDiff}(MNext), \frac{b-a}{2})$

6: $\omega_a \leftarrow \mathsf{TTranslate}(\omega, -a)$
7: $ANext \leftarrow \mathsf{Next}(\omega_a, p)$
8: $ACurr \leftarrow \mathsf{SDilate}(\mathsf{CurrSeg}(\omega_a, \neg p), 2)$
9: $LcPlot \leftarrow \mathsf{Min}(\mathsf{Anti}(\mathsf{IdDiff}(ANext), ACurr))$

10: $\omega_b \leftarrow \mathsf{TTranslate}(\omega, -b)$
11: $BPrev \leftarrow \mathsf{Prev}(\omega_b, p)$
12: $BCurr \leftarrow \mathsf{SDilate}(\mathsf{CurrSeg}(\omega_b, \neg p), 2)$
13: $LcPlot \leftarrow \mathsf{Min}(\mathsf{IdDiff}(BPrev), BCurr)$

14: **return** $DPlot = \mathsf{Min}(LmPlot, RmPlot, LcPlot, RcPlot)$

Lemma 11. *Given a signal $\omega : [0, T] \to \{0, 1\}$ and a formula $\varphi = \Box_{[t+a,t+b]}p$ such that $\omega \not\models \varphi$, Alg. 10 computes the function $d(\omega, \mathcal{L}(\varphi))$, and runs in $\mathcal{O}(n)$ time.*

Proof. $\forall t$, we wish to prove this algorithm computes $d(\omega, \mathcal{L}(\Box_{[t+a,t+b]}p))$.

At the end of line 3, $LmPlot$ computes

$$\mathsf{Prev}(\omega, \neg p)\left(t + \left(\frac{a+b}{2}\right)\right) - \left(t + \left(\frac{a+b}{2}\right)\right) - \left(\frac{b-a}{2}\right) = \mathsf{Prev}(\omega, \neg p)\left(t + \left(\frac{a+b}{2}\right)\right) - (t+a)$$

Which agrees with lm as computed in Alg. 7.

Similarly, $RmPlot$ computes

$$\left(t + \left(\frac{a+b}{2}\right)\right) - \mathsf{Next}(\omega, \neg p)\left(t + \left(\frac{a+b}{2}\right)\right) - \left(\frac{b-a}{2}\right) = (t+b) - \mathsf{Next}(\omega, \neg p)\left(t + \left(\frac{a+b}{2}\right)\right)$$

LcPlot computes

$$\min\left(\frac{\mathsf{CurrSeg}(\omega,\neg p)(t+a)}{2}, \mathsf{Next}(\omega,p)(t+a)-(t+a)\right)$$

RcPlot computes

$$\min\left(\frac{\mathsf{CurrSeg}(\omega,\neg p)(t+b)}{2}, (t+b)-\mathsf{Prev}(\omega,p)(t+b)\right)$$

Each of these agrees with rm, lc and rc as computed in Alg. 7, so the overall algorithm is correct.

Clearly this takes $\mathcal{O}(n)$ time, and the resulting plot has $\mathcal{O}(n)$ segments since each of Prev and Next have n segments, and the operations at most double the number of segments.

Algorithm 11 $\mathsf{d}(\omega, \varphi(\mathsf{t}))$ where $\varphi = \Box_{[a,t+b]}p$

1: $\omega_m \leftarrow \mathsf{TTranslate}(\mathsf{TDilate}(\omega, 2), -(a+b))$
2: $MPrev \leftarrow \mathsf{Prev}(\omega_m, \neg p)$
3: $MPrev \leftarrow \mathsf{STranslate}(MPrev, a+b)$
4: $MPrev \leftarrow \mathsf{SDilate}(MPrev, 2)$
5: $LmPlot \leftarrow \mathsf{STranslate}(MPrev, -a)$

6: $MNext \leftarrow \mathsf{Next}(\omega_m, \neg p)$
7: $MNext \leftarrow \mathsf{STranslate}(MNext, a+b)$
8: $MNext \leftarrow \mathsf{SDilate}(MNext, 2)$
9: $RmPlot \leftarrow \mathsf{STranslate}(\mathsf{IdDiff}(MNext), b)$

10: $LcPlot \leftarrow \mathsf{Const}(\min(\mathsf{Next}(\omega, p, a) - a, \frac{\mathsf{CurrSeg}(\omega,\neg p,a)}{2}))$

11: $\omega_b \leftarrow \mathsf{TTranslate}(\omega, -b)$
12: $BPrev \leftarrow \mathsf{Prev}(\omega_b, p)$
13: $BCurr \leftarrow \mathsf{SDilate}(\mathsf{CurrSeg}(\omega_b, \neg p), 2)$
14: $LcPlot \leftarrow \mathsf{Min}(\mathsf{IdDiff}(BPrev), BCurr)$

15: **return** $DPlot = \mathsf{Min}(LmPlot, RmPlot, LcPlot, RcPlot)$

Lemma 12. *Given a signal $\omega : [0,T] \to \{0,1\}$ and a formula $\varphi = \Box_{[a,t+b]}p$ such that $\omega \not\models \varphi$, Alg. 11 computes the function $d(\omega, \mathcal{L}(\varphi))$, and runs in $\mathcal{O}(n)$ time.*

Proof. $\forall t$, we wish to prove this algorithm computes $d(\omega, \mathcal{L}(\Box_{[a,t+b]}p))$.

At the end of line 5, $LmPlot$ should compute

$$\mathsf{Prev}(\omega,\neg p)\left(\frac{t+a+b}{2}\right) - a$$

in order to agree with lm as computed in Alg. 7, substituting $t+b$ for b. This holds if and only if at the end of line 4, $MPrev$ stores

$$\mathsf{Prev}(\omega, \neg p)\left(\frac{t+a+b}{2}\right)$$

This is true because at the end of line 4, $MPrev$ stores

$$\frac{(\mathsf{Prev}(\omega_m, \neg p)) + (a+b)}{2}$$

where

$$\omega_m(t) = \omega\left(\frac{t+a+b}{2}\right)$$

$$\mathsf{Prev}(\omega_m, \neg p) = \max\{t' | (t' < t) \wedge (\omega_m(t) = \neg p)\} = \max\{t' | (t' < t) \wedge (\omega_m\left(\frac{t+a+b}{2}\right) = \neg p)\} =$$

$$\max\{t' | (t' < 2t'' - (a+b)) \wedge (\omega(t'') = \neg p)\} = 2\max\{t' | (t' < t'') \wedge (\omega(t'') = \neg p)\} - (a+b)$$

Which is precisely the result of lines 1-4.

Similarly, in accordance with Alg. 7 $RmPlot$ computes

$$t + b - \mathsf{Next}(\omega, \neg p)\left(\frac{t+a+b}{2}\right)$$

Which is true because by an analogous argument,

$$\mathsf{Next}(\omega, \neg p)\left(\frac{t+a+b}{2}\right) = \frac{(\mathsf{Next}(\omega_m, \neg p)) + (a+b)}{2}$$

Exactly as in Alg. 7, $LcPlot$ computes

$$\min\left(\frac{\mathsf{CurrSeg}(\omega, \neg p)(a)}{2}, \mathsf{Next}(\omega, p)(a) - (a)\right)$$

And exactly as in Alg. 10, $RcPlot$ computes

$$\min\left(\frac{\mathsf{CurrSeg}(\omega, \neg p)(t+b)}{2}, (t+b) - \mathsf{Prev}(\omega, p)(t+b)\right)$$

Each of these agrees with rm, lc and rc as computed in Alg. 7, so the overall algorithm is correct.

Clearly this takes $\mathcal{O}(n)$ time, and the resulting plot has $\mathcal{O}(n)$ segments since each of Prev and Next have n segments, and the operations at most double the number of segments.

Lemma 13. *Given a signal $\omega : [0,T] \to \{0,1\}$ and a formula $\varphi = \Box_{[t+a,b]}p$ such that $\omega \not\models \varphi$, Alg. 12 computes the function $d(\omega, \mathcal{L}(\varphi))$, and runs in $\mathcal{O}(n)$ time.*

Algorithm 12 $\mathsf{d}(\omega, \varphi(\mathsf{t})$ where $\varphi = \Box_{[t+a,b]}p$

1: $\omega_m \leftarrow \mathsf{TTranslate}(\mathsf{TDilate}(\omega, 2), -(a+b))$
2: $MPrev \leftarrow \mathsf{Prev}(\omega_m, \neg p)$
3: $MPrev \leftarrow \mathsf{STranslate}(MPrev, a+b)$
4: $MPrev \leftarrow \mathsf{SDilate}(MPrev, 2)$
5: $LmPlot \leftarrow \mathsf{STranslate}(\mathsf{Anti}(\mathsf{IdDiff}(MPrev), -a))$

6: $MNext \leftarrow \mathsf{Next}(\omega_m, \neg p)$
7: $MNext \leftarrow \mathsf{STranslate}(MNext, a+b)$
8: $MNext \leftarrow \mathsf{SDilate}(MNext, 2)$
9: $RmPlot \leftarrow \mathsf{STranslate}(-(MNext), b)$

10: $\omega_a \leftarrow \mathsf{TTranslate}(\omega, -a)$
11: $ANext \leftarrow \mathsf{Next}(\omega_a, p)$
12: $ACurr \leftarrow \mathsf{SDilate}(\mathsf{CurrSeg}(\omega_a, \neg p), 2)$
13: $LcPlot \leftarrow \mathsf{Min}(\mathsf{Anti}(\mathsf{IdDiff}(ANext), ACurr))$

14: $RcPlot \leftarrow Const(\min(b - \mathsf{Prev}(\omega, p, b), \frac{\mathsf{CurrSeg}(\omega, \neg p, b)}{2}))$

15: **return** $DPlot = \mathsf{Min}(LmPlot, RmPlot, LcPlot, RcPlot)$

Proof. $\forall t$, we wish to prove this algorithm computes $d(\omega, \mathcal{L}(\Box_{[t+a,b]}p))$.

At the end of line 5, $LmPlot$ should compute

$$\mathsf{Prev}(\omega, \neg p)\left(\frac{t+a+b}{2}\right) - (t+a)$$

in order to agree with lm as computed in Alg. 7, substituting $t+b$ for b. This is true because as seen in the proof of correctness for Alg. 11 at the end of line 4, $MPrev$ stores

$$\mathsf{Prev}(\omega, \neg p)\left(\frac{t+a+b}{2}\right)$$

Similarly, in accordance with Alg. 7 $RmPlot$ computes

$$t + b - \mathsf{Next}(\omega, \neg p)\left(\frac{t+a+b}{2}\right)$$

Which is true because

$$\mathsf{Next}(\omega, \neg p)\left(\frac{t+a+b}{2}\right) = \frac{(\mathsf{Next}(\omega_m, \neg p)) + (a+b)}{2}$$

Exactly as in Alg. 10, $LcPlot$ computes

$$\min\left(\frac{\mathsf{CurrSeg}(\omega, \neg p)(t+a)}{2}, \mathsf{Next}(\omega, p)(t+a) - (t+a)\right)$$

And exactly as in Alg. 7, *RcPlot* computes

$$\min\left(\frac{\mathsf{CurrSeg}(\omega, \neg p)(b)}{2}, b - \mathsf{Prev}(\omega, p)(b)\right)$$

Each of these agrees with rm, lc and rc as computed in Alg. 7, so the overall algorithm is correct.

Clearly this takes $\mathcal{O}(n)$ time, and the resulting plot has $\mathcal{O}(n)$ segments since each of Prev and Next have n segments, and the operations at most double the number of segments.

B.6 Until, Not Until, and Not Bounded Reponse

In this subsection, we compute the distance of a signal to the languages of the formulae $p\,\mathcal{U}_{[a,b]}\,q$ $\neg p\,\mathcal{U}_{[a,b]}\,q$ and $\neg(\Box(p \Rightarrow \Diamond_{[0,b]}q)$, the negation of the bounded response property.

We group these together due to the similarity in the technique we use to compute the robustness. We compute the distance by first rewriting these languages in terms of equivalent but easier to compute formulae, and then computing distances to those formulae using the variable domain algorithms introduced in the previous subsection.

Algorithm 13 $\mathsf{d}(\omega, \varphi)$ where $\varphi = p\,\mathcal{U}_{[a,b]}\,q$

1: $QPlot \leftarrow \mathsf{d}(\omega, \mathsf{q(t)})$
2: $GPlot \leftarrow \mathsf{d}(\omega, \Box_{[0,t]}\mathsf{p})$
3: $DPlot \leftarrow \mathsf{Max}(QPlot, GPlot)$

4: **return** $d = \min_{t\in[a,b]} DPlot$

Lemma 14. *Given a signal $\omega : [0,T] \to \{0,1\}$ and a formula $\varphi = p\,\mathcal{U}_{[a,b]}\,q$ such that $\omega \not\models \varphi$, Alg. 13 computes the function $d(\omega, \mathcal{L}(\varphi))$, and runs in $\mathcal{O}(n)$ time.*

Proof. In order to prove that $d(\omega, p) = \min_{t\in[a,b]} DPlot$, we first prove that $d(\omega, p) \geq \min_{t\in[a,b]} DPlot$, and then prove that there exists an element or a limit point of $\mathcal{L}(\varphi)$ such that its distance to ω is $\min_{t\in[a,b]} DPlot$.

1. We know that $\omega \not\models \varphi$. Let $t_u = \arg\min_{t\in[a,b]} DPlot$.
   Consider any signal $\alpha \in \mathcal{L}(\varphi)$. Since $\alpha \models p\,\mathcal{U}_{[a,b]}\,q$, by definition we know that $\exists t \in [a,b]$ $(\alpha \in \mathcal{L}(q(t) \wedge \Box_{[0,t)}p))$.
   We now note that the result of the algorithm $\min_{t\in[a,b]} DPlot = \min_{t\in[a,b]} d(\omega, q(t) \wedge \Box_{[0,t]}p)))$ by definition.
   Distance to the set $\Box_{[0,t]}p$, or $\Box_{[0,t)}p$ is identical, so $\forall\alpha \in \mathcal{L}(\varphi)$ $\min_{t\in[a,b]} DPlot \leq d(\omega, \alpha)$.

Hence, $\forall \alpha \in \mathcal{L}(\varphi)$, $d(\omega, \alpha) \geq \min_{t \in [a,b]} DPlot$

$$d(\omega, p) \geq \min_{t \in [a,b]} DPlot$$

2. As for feasibility, we see that a signal α that is identical to ω's nearest signal in $\Box_{[0,t_u)} p$ except for a q value spike at t_u has the property $(\alpha \vDash \varphi) \wedge d(\omega, \alpha) \leq \min_{t \in [a,b]} DPlot$.

$$\implies d(\omega, p) \leq \min_{t \in [a,b]} DPlot$$

Hence, we can conclude that

$$d(\omega, p) = \min_{t \in [a,b]} DPlot.$$

It uses four linear time operations, and hence overall runs in $\mathcal{O}(n)$ time.

Algorithm 14 $\mathsf{d}(\omega, \varphi)$ where $\varphi = \neg p \, \mathcal{U}_{[a,b]} \, q$

1: $PQPlot \leftarrow \mathsf{d}(\omega, (\neg \mathsf{p} \wedge \neg \mathsf{q})(\mathsf{t}))$
2: $GPlot \leftarrow \mathsf{d}(\omega, \Box_{[\mathsf{a},\mathsf{t}]} \neg \mathsf{q})$
3: $DPlot \leftarrow \mathsf{Max}(PQPlot, GPlot)$

4: $d \leftarrow \min(d(\omega, \Box_{[a,b]} \neg q), d(\omega, \Diamond_{[0,a]} \neg p))$
5: **return** $d = \min(d, \min_{t \in [a,b]} DPlot)$

Lemma 15. *Given a signal $\omega : [0, T] \to \{0, 1\}$ and a formula $\varphi = \neg p \, \mathcal{U}_{[a,b]} \, q$ such that $\omega \nvDash \varphi$, Alg. 14 computes the function $d(\omega, \mathcal{L}(\varphi))$, and runs in $\mathcal{O}(n)$ time.*

Proof. In order to prove that $d(\omega, p) = d$, we first prove that $d(\omega, p) \geq d$, and then prove that there exists an element or a limit point of $\mathcal{L}(\varphi)$ such that its distance to ω is d.

1. Consider any signal $\alpha \in \mathcal{L}(\varphi)$. Since $\alpha \vDash \neg(p \, \mathcal{U}_{[a,b]} \, q)$, by definition we know that either $\alpha \vDash \Diamond_{[0,a]} \neg p$, or $\alpha \vDash \Box[a, b] \neg q$, or $\exists t \in [a, b]$ $(\alpha \in \mathcal{L}((\neg p \wedge \neg q)(t) \wedge \Box_{[a,t)} \neg q))$. This is because in order to violate $p \, \mathcal{U}_{[a,b]} \, q$, one of three things must happen.
   The first possibility is that there is no q in the interval $[a, b]$.
   The second, is that there was a moment where $\neg p$ held in the initial "necessary" interval $[0, a]$.
   The third, is that if both the first and the second possibilities did not occur, then it must be true that before the first q in $[a, b]$ (i.e. at some moment t when $\Box_{[a,t)} \neg q$ was true, p did not hold (i.e., $(\neg p \wedge \neg q)(t)$).

Whichever of the three possibilities α falls into, due to lines 1-3 it is clear that $d \leq d(\omega, \alpha)$
Hence, $\forall \alpha \in \mathcal{L}(\varphi)$, $d(\omega, \alpha) \geq d$

$$d(\omega, p) \geq d$$

2. Now, we seek to prove feasibility. Note that the first two possibilities are formulae with effectively disjoint domains, and the third need only be considered within in the complement of the union of the first two. Hence, depending on which term is lower, we can find a witness signal α such that $d(\omega, \alpha) = d$ by either gluing together the witness signals for the first two cases, or using the witness for the third.

$$\implies d(\omega, p) \leq d$$

Hence, we can conclude that

$$d(\omega, p) = d.$$

Algorithm 15 $\mathsf{d}(\omega, \varphi)$ where $\varphi = \neg(\Box(p \Rightarrow \Diamond_{[0,b]} q))$

1: $PPlot \leftarrow \mathsf{d}(\omega, \mathsf{p(t)})$
2: $GPlot \leftarrow \mathsf{d}(\omega, \Box_{[t+0,t+b]} \neg \mathsf{q})$
3: $DPlot \leftarrow \mathsf{Max}(PPlot, GPlot)$

4: **return** $d = \min_{t \in [0,T]} DPlot$

Lemma 16. *Given a signal $\omega : [0, T] \to \{0, 1\}$ and a formula $\varphi = \neg(\Box(p \Rightarrow \Diamond_{[0,b]} q))$ such that $\omega \not\models \varphi$, Alg. 15 computes the function $d(\omega, \mathcal{L}(\varphi))$, and runs in $\mathcal{O}(n)$ time.*

Proof. In order to prove that $d(\omega, p) = \min_{t \in [0,T]} DPlot$, we first prove that $d(\omega, p) \geq \min_{t \in [0,T]} DPlot$, and then prove that there exists an element or a limit point of $\mathcal{L}(\varphi)$ such that its distance to ω is $\min_{t \in [0,T]} DPlot$.

1. Consider any signal $\alpha \in \mathcal{L}(\varphi)$. Since $\alpha \vDash \neg(\Box(p \Rightarrow \Diamond_{[0,b]} q)$, by definition we know that

$$\exists t \;\; \alpha \vDash p(t) \wedge \Box_{[t,t+b]} \neg q$$

Due to lines 1 and 2 we see that $d = \min_{t \in [0,T]} DPlot \leq \mathsf{Max}(PPlot(t), GPlot(t)) \leq d(\omega, \alpha)$.
Hence, $\forall \alpha \in \mathcal{L}(\varphi)$, $d(\omega, \alpha) \geq \min_{t \in [0,T]} DPlot$

$$d(\omega, p) \geq \min_{t \in [0,T]} DPlot$$

2. Now, we seek to prove feasibility. Once again, the two conditions are formulae with effectively disjoint domains, so we can find a witness signal α such that $d(\omega, \alpha) = \min_{t\in[0,T]} DPlot$ by gluing together the witness signals for the two conditions.

$$\implies d(\omega, p) \leq \min_{t\in[0,T]} DPlot$$

Hence, we can conclude that

$$d(\omega, p) = \min_{t\in[0,T]} DPlot.$$

B.7 Bounded Response

Now, we focus on computing δ for the bounded response property, a very useful STL specification that promises that whenever a request of some sort (p) appears, the system will grant it a response (q) within an appropriate time interval.

Hence, we can present again the algorithm for computing δ for the bounded response property.

Before we prove the correctness of Alg. 16, we take a moment to set up some notation and prove some lemmas about the properties of the algorithm.

Let us denote by $Res(\omega, b, val)$ the signal that greedily places a thin spike at every multiple of b, with value val.

Lemma 17. *When the entire domain $[0, T]$ of ω is a region of fault, one of the following cases occur*

1. *It is an initial region of fault, and there is a point with value $\neg p \wedge \neg q$ in $[T - l, T - l + \frac{l-b}{2}]$, in which case $d(\omega, \varphi) = \max(c, \frac{l-b}{2})$.*
2. *It is an initial region of fault, and there is no point with value $\neg p \wedge \neg q$ in that range, in which case $d(\omega, \varphi) = l - b$.*
3. *It is a middle region of fault, in which case $d(\omega, \varphi) = \frac{1}{2} \cdot \max(m, l - b - m', c)$*
4. *It is an final region of fault, and there is a point with value $\neg p \wedge \neg q$, in which case $d(\omega, \varphi) = \max(c, \frac{l}{2})$.*
5. *It is an final region of fault, and there is no point with value $\neg p \wedge \neg q$, in which case $d(\omega, \varphi) = l$.*

Where l, m, m' and c are as defined in the algorithm.

Moreover, as calculated above, for any $\omega_i \cdot \omega \cdot \omega_f$, i.e. an extension of ω, and any other signal $\omega'' \vDash \varphi$ over the domain $[0, T]$, $d(\omega'', \omega_i \cdot \omega \cdot \omega_f) \leq d(\omega, \varphi)$.

Proof. We proceed case by case, as indicated by the lemma.

1. In order to prove that $d(\omega, p) = \max(c, \frac{l-b}{2})$, we first prove that $d(\omega, p) \geq \max(c, \frac{l-b}{2})$, and then prove that there exists an element or a limit point of $\mathcal{L}(\varphi)$ such that its distance to ω is $\max(c, \frac{l-b}{2})$.

Algorithm 16 $d(\omega, \varphi)$ where $\varphi = \Box(p \Rightarrow \Diamond_{[0,b]} q)$

Let $\omega = (0, t_1, a_1)(t_1, t_2, a_2) \dots (t_{n-1}, t_n, a_n)$ where $\forall\, i \in [n] : a_i \in \{0, 1\}$.
$R \leftarrow \emptyset$, $i \leftarrow 1$, $u, v \leftarrow 0$
for all $(i \leq n)$ **do**
    if $\omega(t_i) \nvDash q$ **then**
        $v \leftarrow t_i$
    else
        if $u < v$ **then**
            $R \leftarrow R \cup [u, v]$
        $u \leftarrow t_i$
    $i \leftarrow i + 1$
$R \leftarrow R \cup [u, T]$, $i \leftarrow 1$
if $(R = \{[0, t_n\}\})$ **then**
    $d \leftarrow \infty$
else
    for all $[u, v] \in R$ **do**
        if $(u = 0) \wedge \omega(0) \nvDash q$ **then** ▷ *Initial ROF*
            $l \leftarrow v - \min\{t \in [u, v] \,|\, \omega(t) \vDash p\}$
            if $\exists\, t \in [u, v]$, $\omega(t) = \neg p \wedge \neg q$ **then**
                $c \leftarrow -\delta(\omega, \Box_{[v-l, v-l+\left(\frac{l-b}{2}\right)]} \neg p)$
                $d \leftarrow \max(d, \left(\frac{l-b}{2}\right), c)$
            else
                $d \leftarrow \max(d, (l - b))$
        else if $(\omega(u) \vDash q) \wedge (\omega(v) \vDash q)$ **then** ▷ *Middle ROF*
            $l \leftarrow v - u$
            $m \leftarrow \max\{t \in [u, v] \,|\, (t \leq \frac{l-b}{2}) \wedge (\omega(t) = p \wedge \neg q)\}$
            $m' \leftarrow \min\{t \in [u, v] \,|\, (t \geq \frac{l-b}{2}) \wedge (\omega(t) = p \wedge \neg q)\}$
            $c \leftarrow \max\{r \in \mathbb{R}_{\geq 0} \,|\, \omega \vDash \Box_{[\frac{l-b}{2} - r, \frac{l-b}{2} + r]}\, p \wedge \neg q\}$
            $d \leftarrow \max(d, 0.5 \times \max(m + a, l - b - m', c))$
        else ▷ *Final ROF*
            $l \leftarrow \max\{t \in [u, v] \,|\, \omega(t) \vDash p\} - u$
            if $\exists\, t \in [u, v]$, $\omega(t) = \neg p \wedge \neg q$ **then**
                $c \leftarrow -\delta(\omega, \Box_{[v-\left(\frac{l-a}{2}\right), v]} \neg p)$
                $d \leftarrow \max(d, c, \left(\frac{l-a}{2}\right))$
            else
                $d \leftarrow \max(d, l)$
return d

(a) In this case, consider any $\omega' \vDash \varphi$, and let $t = \min\{t|\omega'(t) \vDash q\}$.
Since $\omega' \vDash \varphi$, we know that for all $t' \in [0, t-b]$ $\omega'(t') \nvDash p$. Hence, the closest match for the value at $T-l$ in ω is after $t-b$. On the other hand, the closest match for the value of ω' at t is at T. Hence,

$$d(\omega, \omega') \geq \max\{t - b - T + l, T - t\} \geq \frac{l-b}{2} = m.$$

Moreover, for any segment in $[T-l, T-l+\frac{l-b}{2}]$ with value $p \wedge \neg q$, the cost of erasing them all is at least c.
This holds even if ω were extended left and right.
Hence, $\forall \alpha \in \mathcal{L}(\varphi)$, $d(\omega, \alpha) \geq \max(c, \frac{l-b}{2})$

$$d(\omega, p) \geq \max(c, \frac{l-b}{2})$$

(b) In order to achieve $\max(c, \frac{l-b}{2})$, consider the signal $(\neg p \wedge \neg q)|_0^m \cdot \omega|_m^{m+b} \cdot (\omega(T))|_{m+b}^T$. This is both a limit point of the language of φ, and also is at the required distance from ω.

$$\implies d(\omega, p) \leq \max(c, \frac{l-b}{2})$$

Hence, we can conclude that

$$d(\omega, p) = \max(c, \frac{l-b}{2}).$$

2. In order to prove that $d(\omega, p) = l - b$, we first prove that $d(\omega, p) \geq l - b$, and then prove that there exists an element or a limit point of $\mathcal{L}(\varphi)$ such that its distance to ω is $l - b$.
   (a) In this case, consider any $\alpha \vDash \varphi$, and let $t = \min\{t|\alpha(t) \vDash q\}$. If $t - b > 0$ then the value of 0 must not be $p \wedge \neg q$, and any such value's closest is at the very least at T, so the distance is already maximal. Hence, it is more efficient to let $t - b \leq 0$, and so that the closest(T) is not too far, $t = b$. This holds even if ω were extended left and right.
   Hence, $\forall \alpha \in \mathcal{L}(\varphi)$, $d(\omega, \alpha) \geq l - b$

$$d(\omega, p) \geq l - b$$

   (b) In order to achieve this, consider the signal $(\neg p \wedge \neg q)|_0^b \cdot (\omega(T))|_b^T$. This is both a limit point of the language of φ, and also is at the required distance from ω.

$$\implies d(\omega, p) \leq l - b$$

   Hence, we can conclude that

$$d(\omega, p) = l - b.$$

3. In order to prove that $d(\omega, p) = \frac{1}{2} \cdot \max(m, l-b-m', c)$, we first prove that $d(\omega, p) \geq \frac{1}{2} \cdot \max(m, l-b-m', c)$, and then prove that there exists an element or a limit point of $\mathcal{L}(\varphi)$ such that its distance to ω is $\frac{1}{2} \cdot \max(m, l-b-m', c)$.
   (a) In this case, consider any $\alpha \vDash \varphi$, and let m, m' and c be defined as in lines 25-27.
   We first focus on the case where $c = 0$. Without loss of generality, suppose $m \geq l-b-m'$, set m' to $l-b-m$ instead.
   As in previous cases, we consider the timestamp t in ω such that $t = \max\{t \leq m | \alpha(t) \vDash p\}$. Closest point to m in α is hence t, and the closest point to t in ω is 0, so $d(\omega, \varphi) \geq \frac{m}{2}$. This holds even if ω were extended left and right, and a similar argument shows the lower bound for the m' case. .
   On the other hand, suppose $c > 0$. In this case, $m = m' = \frac{l-b}{2}$. If $c \leq \frac{l-b}{2}$ then the above cases argument remains sufficient. On the other hand, if c is larger, then consider the midpoint of c, call it t_c. Since c is large, we can see that $t_c \in [\frac{m}{2}, \frac{l-b+m}{2}]$, which means if it remains valued $p \wedge \neg q$ then its response would incur more than $\frac{c}{2}$ distance to its closest in ω, and if it is valued anything else, its own closest incurs at least $\frac{c}{2}$ distance, hence $d(\omega, \varphi) \geq \frac{c}{2}$ All these lowerbounds hold even if ω were extended left and right.
   Hence, $\forall \alpha \in \mathcal{L}(\varphi)$, $d(\omega, \alpha) \geq \frac{1}{2} \cdot \max(m, l-b-m', c)$

$$d(\omega, p) \geq \frac{1}{2} \cdot \max(m, l-b-m', c)$$

   (b) In order to achieve this when $c = 0$, consider the signal $Res(\omega|_0^m, b, \omega(0)) \cdot (\neg p \wedge \neg q)|_m^{m'} \cdot Res(\omega|_{m'}^T, b, \omega(T))$. This is both a limit point of the language of φ, and also is at the required distance from ω.
   In order to achieve this when $c \neq 0$ we consider the same witness signal as in the previous case if there are any points valued $\neg p \wedge \neg q$ in this region of fault. If not, let $m = \frac{c}{2}$ and $m' = T - \frac{c}{2}$, we consider the signal $\omega' = (\omega(0))|_0^m \cdot (p \wedge \neg q)|_m^{m'} \cdot (\omega(T))|_{m'}^T$, and we find that it achieves the required distance.

$$\implies d(\omega, p) \leq \frac{1}{2} \cdot \max(m, l-b-m', c)$$

   Hence, we can conclude that

$$d(\omega, p) = \frac{1}{2} \cdot \max(m, l-b-m', c).$$

4. In order to prove that $d(\omega, p) = \max(c, \frac{l}{2})$, we first prove that $d(\omega, p) \geq \max(c, \frac{l}{2})$, and then prove that there exists an element or a limit point of $\mathcal{L}(\varphi)$ such that its distance to ω is $\max(c, \frac{l}{2})$.
   (a) In this case, consider any $\alpha \vDash \varphi$, and let $t = \max\{t | \alpha(t) \vDash q\}$.

Since $\alpha \vDash \varphi$, we know that for all $t' \in [t, T]$ $\alpha(t') \nvDash p$. Hence, the closest match for the value for t in ω is 0. On the other hand, the closest match for the value of ω at l is at t. Hence,

$$d(\omega, \alpha) \geq \frac{l}{2} = m.$$

Moreover, for any segment in $[\frac{l}{2}, l]$ with value $p \wedge \neg q$, the cost of erasing them all is at least c.
This holds even if ω were extended left and right.
Hence, $\forall \alpha \in \mathcal{L}(\varphi)$, $d(\omega, \alpha) \geq \max(c, \frac{l}{2})$

$$d(\omega, p) \geq \max(c, \frac{l}{2})$$

(b) In order to achieve this distance,let $m = \frac{l}{2}$, consider the signal $\omega|_0^m \cdot (\neg p \wedge \neg q)|_m^T$ with a spike of value $\omega(0)$ at m. This is both a limit point of the language of φ, and also is at the required distance from ω.

$$\implies d(\omega, p) \leq \max(c, \frac{l}{2})$$

Hence, we can conclude that

$$d(\omega, p) = \max(c, \frac{l}{2}).$$

5. In order to prove that $d(\omega, p) = l$, we first prove that $d(\omega, p) \geq l$, and then prove that there exists an element or a limit point of $\mathcal{L}(\varphi)$ such that its distance to ω is l.
   (a) In this case, consider any $\alpha \vDash \varphi$, and let $t = \min\{t | \alpha(t) \vDash q\}$. If $t < l$ then the value of l must not be $p \wedge \neg q$, and any such value's closest is at the very least at 0, so the distance is already maximal. Hence, it is more efficient to let $t = l$. This holds even if ω were extended left and right.
   Hence, $\forall \alpha \in \mathcal{L}(\varphi)$, $d(\omega, \alpha) \geq l$

$$d(\omega, p) \geq l$$

   (b) In order to achieve this, consider the signal $Res(\omega, b, \omega(0))$. This is both a limit point of the language of φ, and also is at the required distance from ω.

$$\implies d(\omega, p) \leq l$$

   Hence, we can conclude that

$$d(\omega, p) = l.$$

Lemma 3 (Decomposition lemma). *Given a signal $\omega \nvDash \varphi$ over the domain $[0, T]$, consider the following decomposition of the domain: $t_0 = 0 \leq t_1 < \cdots \leq t_{2n} = T$, where $\forall i \in [n], (t_{2i}, t_{2i+1})$ contains no regions of fault and $[t_{2i+1}, t_{2i+2}]$ is a region of fault. Given such a decomposition, $d(\omega, \varphi) = \max_{1 \leq i \leq n} d(\omega|_{t_{2i-1}}^{t_{2i}}, \varphi)$.*

Proof. For all $i \in [n]$ let $\omega_i = \omega|_{t_{2i-1}}^{t_{2i}}$. Suppose there existed a signal $\omega'' \vDash \varphi$ such that $d(\omega'', \omega) < \max_{i \in [n]} d(\omega_i, \varphi) = d$. Then, let ω_k be the region of fault with the greatest distance, i.e., $d(\omega_k, \varphi) = d$. Let ω_k'' be ω'' restricted to the same domain as ω_k. We know that $d(\omega_k'', \omega) \leq d(\omega'', \omega) < d$.

But, by Lem. 17, we know that ω_k is locally optimal for ω even though the domain of ω is extended. Hence, this provides a contradiction.

Lemma 18. *Given a signal $\omega : [0, T] \to \{0, 1\}^2$ and a formula $\varphi = \square(p \Rightarrow \Diamond_{[0,b]} q)$ such that $\omega \nvDash \varphi$, Alg. 16 computes $d(\omega, \mathcal{L}(\varphi))$, and runs in $\mathcal{O}(n)$ time.*

Proof. Given Lem. 3, we see that splitting the signal into regions of fault to compute distance individually is sound, and Lem. 17 showed the correctness of the distance computation on each region of fault, so overall the algorithm is correct.

As for its complexity, it runs in linear time as each local computation is linear in the size of its region of fault, and the overall computation as a result runs in time linear in the number of segments in the entire signal.

Now, we are ready to prove Thm. 3.

Theorem 3. *Given a signal ω with size parameters (n, h, b) and a property $\varphi \in STL_r$, computing $\delta(\omega, \varphi)$ is in $\mathcal{O}(n(h + b) \cdot |\varphi|)$-time.*

Proof. The fact that $\delta(\omega, \neg\varphi) = -\delta(\omega, \varphi)$ follows from the definition of δ, and when domains are disjoint, as we have seen, witness signals only differ from signals within their domains, so $d(\omega, \varphi \vee \varphi') = \max(d(\omega, \varphi), d(\omega, \varphi'))$ since we can construct a non-conflicting combined witness signal.

Aside from these facts, it follows that Alg. 4 is correct because of Lemmas 4, 6 to 8, 14 to 16 and 18.

As for overall complexity, since each subroutine runs in $\mathcal{O}(|\varphi| \cdot |\omega|)$ and only 2 non-constant time algorithms are run per instance, the overall procedure has the same running time.