Time Robustness in MTL and Expressivity in Hybrid System Falsification

Takumi Akazaki[1,2(✉)] and Ichiro Hasuo[1]

[1] The University of Tokyo, Tokyo, Japan
`ultraredrays@is.s.u-tokyo.ac.jp`
[2] JSPS Research Fellow, Tokyo, Japan

Abstract. Building on the work by Fainekos and Pappas and the one by Donzé and Maler, we introduce **AvSTL**, an extension of metric interval temporal logic by *averaged* temporal operators. Its expressivity in capturing both space and time robustness helps solving *falsification* problems (searching for a critical path in hybrid system models); it does so by communicating a designer's intention more faithfully to the stochastic optimization engine employed in a falsification solver. We also introduce a sliding window-like algorithm that keeps the cost of computing truth/robustness values tractable.

1 Introduction

Model-Based Development of Hybrid Systems. The demand for quality assurance of *cyber-physical systems (CPS)* is ever-rising, now that computer-controlled artifacts—cars, aircrafts, and so on—serve diverse safety-critical tasks everywhere in our daily lives. In the industry practice of CPS design, deployment of *model-based development (MBD)* has become a norm. In MBD, (physical and costly) testing workbenches are replaced by (virtual and cheap) *mathematical models*; and this reduces by a great deal the cost of running a *development cycle*—design, implementation, evaluation, and redesign.

One of the distinctive features of CPS is that they are *hybrid systems* and combine discrete and continuous dynamics. For MBD of such systems the software *Simulink* has emerged as an industry standard. In Simulink a designer models a system using block diagrams—a formalism strongly influenced by *control theory*—and runs *simulation*, that is, numerical solution of the system's dynamics.

Falsification. The models of most real-world hybrid systems are believed to be beyond the reach of formal verification. While this is certainly the case with systems as big as a whole car, a single component of it (like automatic transmission or an engine controller) overwhelms the scalability of the state-of-art formal verification techniques, too.

What is worse, hybrid system models tend to have *black-box components*. An example is fuel combustion in an engine. Such chemical reactions are not easy to

model with ODEs, and are therefore commonly represented in a Simulink model by a *look-up table*—a big table of values obtained by physical measurements [18,19]. The lack of structure in a look-up table poses a challenge to formal verification: each entry of the table calls for separate treatment; and this easily leads to state-space explosion.

Under such circumstances, *falsification* by stochastic optimization has proved to be a viable approach to quality assurance [7,18,19]. The problem is formulated as follows:

The falsification problem

Given: a *model* $\mathcal{M}$ (a function from an input signal to an output signal), and
a *specification* φ (a temporal formula),

Answer: a *critical path*, that is, an input signal σ_{in} such that the output $\mathcal{M}(\sigma_{\text{in}})$ does not satisfy φ

$$\xrightarrow{\sigma_{\text{in}}}\boxed{\mathcal{M}}\xrightarrow[\not\models_{?}\varphi]{\mathcal{M}(\sigma_{\text{in}})}$$

Unlike *testing* or *monitoring*—where input σ_{in} is given and we check if $\mathcal{M}(\sigma_{\text{in}}) \models \varphi$—a falsification solver employs stochastic optimization techniques (like the Monte-Carlo ones) and iteratively searches for a falsifying input signal σ_{in}.

Falsification is a versatile tool in MBD of hybrid systems. It is capable of searching for counterexamples, hence revealing potential faults in the design. One can also take, as a specification φ, the negation $\neg\psi$ of a desirable property ψ; then successful falsification amounts to *synthesis* of an input signal that satisfies ψ. Stochastic optimization used in falsification typically does not rely on the internal structure of models, therefore the methodology is suited for models with black-box components. Falsification is fairly scalable, making it a realistic option in the industrial MBD scenarios; see e.g. [18,19].

The current work aims at enhancing falsification solvers, notable among which are S-TaLiRo [7] and BREACH [11]. An obvious way to do so is via improvement of stochastic optimization; see e.g. [24,26]. Here we take a different, logical approach.

Robustness in Metric Temporal Logics. Let us turn to a formalism in which a specification φ is expressed. *Metric interval temporal logic* (**MITL**) [6], and its adaptation *signal temporal logic* (**STL**) [23], are standard temporal logics for (continuous-time) signals. However their conventional semantics—where satisfaction is Boolean—is not suited for falsification by stochastic optimization. This is because a formula φ, no matter if it is *robustly* satisfied and *barely* satisfied, yields the same truth value ("true"), making it not amenable to hill climb-style optimization.

It is the introduction of *robust semantics* of **MITL** [16] that set off the idea of falsification by optimization. In robust semantics, a signal σ and a formula φ are assigned a continuous truth value $[\![\sigma, \varphi]\!] \in \mathbb{R}$ that designates how robustly the formula is satisfied. Such "robustness values" constitute a sound basis for stochastic optimization.

The original robust semantics in [16] is concerned with *space* robustness: for example, the truth values of $\diamondsuit_{[0,10]}(v \geq 80)$ ("the velocity reaches $80\,$km/h within $10\,$sec.") are 20 and 0, for the green and red signals on the right. Therefore space robustness is a "vertical margin" between a signal and a specification. An efficient algorithm is proposed in [12] for computing this notion of robustness.
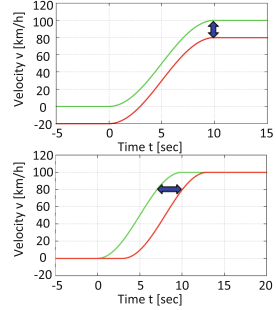
The notion of robustness is extended in [13] to take *time* robustness also into account. Consider the same specification $\diamondsuit_{[0,10]}(v \geq 80)$ against the green and red signals on the right. The green one is more robust since it reaches $80\,$km/h much earlier than the deadline $(10\,$s), while the red one barely makes the deadline.

The current work continues this line of work, with the slogan that *expressivity of temporal logic should help falsification*. With more expressivity, a designer's concerns that were previously ignored (much like time robustness was ignored in [16]) come to be reflected in the continuous truth value. The latter will in turn help stochastic optimization by giving additional "hints." We however are in a *trade-off* situation: the more expressive a logic is, the more expensive computation of truth values is in general.

Contributions. We aim at: a good balance in the last trade-off between expressivity and computational cost; and thereby enhancing falsification solvers by giving more "hints" to stochastic optimization procedures. Our technical contributions are threefold.

The Logic AvSTL. We introduce *averaged STL* (**AvSTL**); it is an extension of **STL** [23] by so-called *averaged temporal operators* like $\overline{\mathcal{U}}_I$ and $\overline{\diamondsuit}_I$. The (continuous) truth values of the new operators are defined by the average of truth values in a suitable interval. We show that this simple extension of **STL** successfully combines space and time robustness in [13,16]; and that its expressivity covers many common specifications (expeditiousness, persistence, deadline, etc.) encountered in the context of CPS.

An Algorithm for Computing AvSTL Robustness. It is natural to expect that nonlocal temporal operators—like $\mathcal{U}_I$, $\diamondsuit_I$ and their averaged variants—incur a big performance penalty in computing truth values. For **STL** (without averaged modalities) an efficient algorithm is proposed in [12]; it employs the idea of the *sliding window minimum algorithm* [22] and achieves complexity that is linear with respect to the size of an input signal (measured by the number of timestamps).

We show that, under mild and realistic assumptions, the same idea as in [12] can be successfully employed to compute **AvSTL** truth values with linear complexity.

Enhancing S-TaLiRo: Implementation and Experiments. We use S-TaLiRo and demonstrate that our logic **AvSTL** indeed achieves a reasonable balance between expressivity and computational cost. We present our prototype implementation: it takes S-TaLiRo and lets the above algorithm (called the

AvSTL *evaluator*) replace TaLiRo, S-TaLiRo's original engine for computing **STL** truth values (see Fig. 7 in Sect. 4).

For its evaluation, we pick some benchmark models $\mathcal{M}$ and **STL** specifications φ—they are mostly automotive examples from [18]—and compare performance between:

– our prototype, run for $\mathcal{M}$ and the original **STL** specification φ,[1] and
– our prototype, run for $\mathcal{M}$ and a *refinement* of φ given as an **AvSTL** formula.

For benchmarks of a certain class we observe substantial performance improvement: sometimes the latter is several times faster; and in some benchmarks we even see the latter succeed in falsification while the former fails to do so.

Related Work. Besides those which are discussed in the above and the below, a closely related work is [2] (its abstract appeared in [3]). There a notion of *conformance* between two models $\mathcal{M}_1$, $\mathcal{M}_2$ is defined; and it is much like (an arity-2 variation of) combination of space and time robustness. Its use in falsification and comparison with the current approach is future work.

Organization of the Paper. In Sect. 2 we introduce the logic **AvSTL**: its syntax, semantics, some basic properties and examples of temporal specifications expressible in it. In Sect. 3, building on [12], an algorithm for computing **AvSTL** truth values is introduced and its complexity is studied. The algorithm is implemented and used to enhance a falsification solver S-TaLiRo, in Sect. 4, where experiment results are presented and discussed.

We used colors in some figures for clarity. Consult the electronic edition in case the colors are unavailable. Most of the proofs are deferred to an appendix in the extended version [4], where the other appendices are found, too.

2 Averaged Signal Temporal Logic AvSTL

We introduce *averaged STL* (**AvSTL**). It is essentially an extension of **MITL** [6] and **STL** [23] with so-called *averaged* temporal operators. We describe its syntax and its semantics (that is inspired by robust semantics in [13,16]). We also exemplify the expressivity of the logic, by encoding common temporal specifications like expeditiousness, persistence and deadline. Finally we will discuss the relationship to the previous robustness notions [13,16] for STL.

2.1 Syntax

We let $\equiv$ stand for the syntactic equality. We let $\mathbb{R}$ denote the set of real numbers, with $\mathbb{R}_{\geq 0}$ and $\mathbb{R}_{\leq 0}$ denoting its obvious subsets. We also fix the set **Var** of variables, each of which stands for a physical quantity (velocity, temperature, etc.).

---

[1] This is the control case of our experiments. We do not use S-TaLiRo itself, because we would like to disregard the potential disadvantage caused by the communication between the **AvSTL** evaluator (the additional component) and S-TaLiRo. We note that the **AvSTL** evaluator is capable of evaluating **STL** formulas, too.

Table 1. Definition of positive and negative robustness

$$[\![\sigma, \top]\!]^+ \triangleq \infty$$
$$[\![\sigma, \bot]\!]^+ \triangleq 0$$
$$[\![\sigma, x < r]\!]^+ \triangleq 0 \sqcup (r - \sigma(0)(x))$$
$$[\![\sigma, x \leq r]\!]^+ \triangleq 0 \sqcup (r - \sigma(0)(x))$$
$$[\![\sigma, x \geq r]\!]^+ \triangleq 0 \sqcup (\sigma(0)(x) - r)$$
$$[\![\sigma, x > r]\!]^+ \triangleq 0 \sqcup (\sigma(0)(x) - r)$$
$$[\![\sigma, \neg\varphi]\!]^+ \triangleq -[\![\sigma, \varphi]\!]^-$$
$$[\![\sigma, \varphi_1 \vee \varphi_2]\!]^+ \triangleq [\![\sigma, \varphi_1]\!]^+ \sqcup [\![\sigma, \varphi_2]\!]^+$$
$$[\![\sigma, \varphi_1 \wedge \varphi_2]\!]^+ \triangleq [\![\sigma, \varphi_1]\!]^+ \sqcap [\![\sigma, \varphi_2]\!]^+$$

$$[\![\sigma, \varphi_1 \, \mathcal{U}_I \, \varphi_2]\!]^+ \triangleq \bigsqcup_{t \in I}([\![\sigma^t, \varphi_2]\!]^+ \sqcap \bigsqcap_{t' \in [0,t)}[\![\sigma^{t'}, \varphi_1]\!]^+)$$
$$[\![\sigma, \varphi_1 \, \mathcal{R}_I \, \varphi_2]\!]^+ \triangleq \bigsqcap_{t \in I}([\![\sigma^t, \varphi_2]\!]^+ \sqcup \bigsqcup_{t' \in [0,t)}[\![\sigma^{t'}, \varphi_1]\!]^+)$$
$$[\![\sigma, \varphi_1 \, \overline{\mathcal{U}}_I \, \varphi_2]\!]^+ \triangleq$$
$$\begin{cases} \dfrac{1}{b-a}\displaystyle\int_a^b [\![\sigma, \varphi_1 \, \mathcal{U}_{I \cap [0,\tau]} \, \varphi_2]\!]^+ d\tau & (I \text{ is bounded}) \\ [\![\sigma, \varphi_1 \, \mathcal{U}_I \, \varphi_2]\!]^+ & (I \text{ is unbounded}) \end{cases}$$
$$[\![\sigma, \varphi_1 \, \overline{\mathcal{R}}_I \, \varphi_2]\!]^+ \triangleq$$
$$\begin{cases} \dfrac{1}{b-a}\displaystyle\int_a^b [\![\sigma, \varphi_1 \, \mathcal{R}_{I \cap [0,\tau]} \, \varphi_2]\!]^+ d\tau & (I \text{ is bounded}) \\ [\![\sigma, \varphi_1 \, \mathcal{R}_I \, \varphi_2]\!]^+ & (I \text{ is unbounded}) \end{cases}$$

$$[\![\sigma, \top]\!]^- \triangleq 0$$
$$[\![\sigma, \bot]\!]^- \triangleq -\infty$$
$$[\![\sigma, x < r]\!]^- \triangleq 0 \sqcap (r - \sigma(0)(x))$$
$$[\![\sigma, x \leq r]\!]^- \triangleq 0 \sqcap (r - \sigma(0)(x))$$
$$[\![\sigma, x \geq r]\!]^- \triangleq 0 \sqcap (\sigma(0)(x) - r)$$
$$[\![\sigma, x > r]\!]^- \triangleq 0 \sqcap (\sigma(0)(x) - r)$$
$$[\![\sigma, \neg\varphi]\!]^- \triangleq -[\![\sigma, \varphi]\!]^+$$
$$[\![\sigma, \varphi_1 \vee \varphi_2]\!]^- \triangleq [\![\sigma, \varphi_1]\!]^- \sqcup [\![\sigma, \varphi_2]\!]^-$$
$$[\![\sigma, \varphi_1 \wedge \varphi_2]\!]^- \triangleq [\![\sigma, \varphi_1]\!]^- \sqcap [\![\sigma, \varphi_2]\!]^-$$

$$[\![\sigma, \varphi_1 \, \mathcal{U}_I \, \varphi_2]\!]^- \triangleq \bigsqcup_{t \in I}([\![\sigma^t, \varphi_2]\!]^- \sqcap \bigsqcap_{t' \in [0,t)}[\![\sigma^{t'}, \varphi_1]\!]^-)$$
$$[\![\sigma, \varphi_1 \, \mathcal{R}_I \, \varphi_2]\!]^- \triangleq \bigsqcap_{t \in I}([\![\sigma^t, \varphi_2]\!]^- \sqcup \bigsqcup_{t' \in [0,t)}[\![\sigma^{t'}, \varphi_1]\!]^-)$$
$$[\![\sigma, \varphi_1 \, \overline{\mathcal{U}}_I \, \varphi_2]\!]^- \triangleq$$
$$\begin{cases} \dfrac{1}{b-a}\displaystyle\int_a^b [\![\sigma, \varphi_1 \, \mathcal{U}_{I \cap [0,\tau]} \, \varphi_2]\!]^- d\tau & (I \text{ is bounded}) \\ [\![\sigma, \varphi_1 \, \mathcal{U}_I \, \varphi_2]\!]^- & (I \text{ is unbounded}) \end{cases}$$
$$[\![\sigma, \varphi_1 \, \overline{\mathcal{R}}_I \, \varphi_2]\!]^- \triangleq$$
$$\begin{cases} \dfrac{1}{b-a}\displaystyle\int_a^b [\![\sigma, \varphi_1 \, \mathcal{R}_{I \cap [0,\tau]} \, \varphi_2]\!]^- d\tau & (I \text{ is bounded}) \\ [\![\sigma, \varphi_1 \, \mathcal{R}_I \, \varphi_2]\!]^- & (I \text{ is unbounded}) \end{cases}$$

Definition 2.1. (Syntax). In **AvSTL**, the set **AP** of *atomic propositions* and the set **Fml** of *formulas* are defined as follows.

$$\mathbf{AP} \ni l ::= x < r \mid x \leq r \mid x \geq r \mid x > r \quad \text{where } x \in \mathbf{Var}, r \in \mathbb{R}$$
$$\mathbf{Fml} \ni \varphi ::= \top \mid \bot \mid l \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \varphi \mathcal{U}_I \varphi \mid \varphi \overline{\mathcal{U}}_I \varphi \mid \varphi \mathcal{R}_I \varphi \mid \varphi \overline{\mathcal{R}}_I \varphi$$

Here I is a closed non-singular interval in $\mathbb{R}_{\geq 0}$, i.e. $I = [a, b]$ or $[a, \infty)$ where $a < b$. The overlined operator $\overline{\mathcal{U}}_I$ is called the *averaged-until* operator.

We introduce the following connectives as abbreviations, as usual: $\varphi_1 \to \varphi_2 \equiv (\neg\varphi_1) \vee \varphi_2$, $\Diamond_I \varphi \equiv \top \mathcal{U}_I \varphi$, $\Box_I \varphi \equiv \bot \mathcal{R}_I \varphi$, $\overline{\Diamond}_I \varphi \equiv \top \overline{\mathcal{U}}_I \varphi$ and $\overline{\Box}_I \varphi \equiv \bot \overline{\mathcal{R}}_I \varphi$. We omit subscripts I for temporal operators if $I = [0, \infty)$. The operators $\overline{\mathcal{R}}_I$, $\overline{\Diamond}_I$ and $\overline{\Box}_I$ are called the *averaged-release*, *averaged-eventually* and *averaged-henceforth* operators, respectively. We say a formula φ is *averaging-free* if it does not contain any averaged temporal operators.

2.2   Robust Semantics

AvSTL formulas, much like **STL** formulas in [13,16], are interpreted over (real-valued, continuous-time) *signals*. The latter stand for trajectories of hybrid systems.

Definition 2.2 (Signal). A *signal* over **Var** is a function $\sigma \colon \mathbb{R}_{\geq 0} \to (\mathbb{R}^{\mathbf{Var}})$; it is therefore a bunch of physical quantities indexed by a continuous notion of time.

For a signal σ and $t \in \mathbb{R}_{\geq 0}$, σ^t denotes the *t-shift* of σ, that is, $\sigma^t(t') \triangleq \sigma(t + t')$.

The interpretation of a formula φ over a signal σ is given by two different "truth values," namely *positive* and *negative robustness*. They are denoted by $[\![\sigma, \varphi]\!]^+$ and $[\![\sigma, \varphi]\!]^-$, respectively.

We will always have $[\![\sigma, \varphi]\!]^+ \geq 0$ and $[\![\sigma, \varphi]\!]^- \leq 0$. We will also see that, for averaging-free φ, it is never the case that $[\![\sigma, \varphi]\!]^+ > 0$ and $[\![\sigma, \varphi]\!]^- < 0$ hold at the same time. See the figure on the right for an example, where a sine-like (black) curve is a signal σ. The blue and red curves stand for the positive and negative robustness, of the formula $x \geq 0$ over the (t-shifted) signal σ^t, respectively.
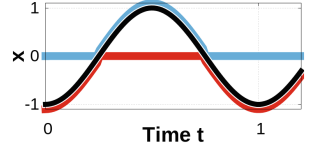
Definition 2.3 (Positive/Negative Robustness). Let $\sigma \colon \mathbb{R}_{\geq 0} \to \mathbb{R}^{\mathbf{Var}}$ be a signal and φ be an **AvSTL** formula. We define the *positive robustness* $[\![\sigma, \varphi]\!]^+ \in \mathbb{R}_{\geq 0} \cup \{\infty\}$ and the *negative robustness* $[\![\sigma, \varphi]\!]^- \in \mathbb{R}_{\leq 0} \cup \{-\infty\}$ by mutual induction, as shown in Table 1. Here $\sqcap$ and $\sqcup$ denote infimums and supremums of real numbers, respectively.

The definition in Table 1 is much like the one for **STL** [12,13],[2] except for the averaged modalities on which a detailed account follows shortly. Conjunctions and disjunctions are interpreted by infimums and supremums, in a straightforward manner.

Figure 1 illustrates the semantics of averaged-temporal operators—the novelty of our logic **AvSTL**. Specifically, the black line designates a signal σ whose only variable is x; and we consider the "averaged-eventually" formula $\overline{\Diamond}_{[0,1]}(x \geq 0)$. For this formula, the definition in Table 1 specializes to:

$$[\![\sigma, \overline{\Diamond}_{[0,1]}(x \geq 0)]\!]^+ = \int_0^1 \left(\bigsqcup_{\tau' \in [0,\tau]} 0 \sqcup \sigma(\tau')(x) \right) d\tau, \quad \text{and} \quad [\![\sigma, \overline{\Diamond}_{[0,1]}(x \geq 0)]\!]^- = \int_0^1 \left(\bigsqcup_{\tau' \in [0,\tau]} 0 \sqcap \sigma(\tau')(x) \right) d\tau.$$

These values obviously coincide with the sizes of the blue and red areas in Fig. 1, respectively. Through this "area" illustration of the averaged-eventually operator we see that: the sooner φ is true, the more (positively) robust $\overline{\Diamond}_I \varphi$ is. It is also clear from Fig. 1 that our semantics captures space robustness too: the bigger a vertical margin is, the bigger an area is.
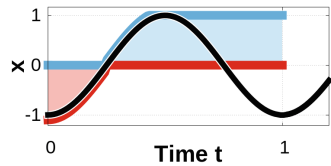
Fig. 1. The positive and negative robustness of $\overline{\Diamond}_{[0,1]}(x \geq 0)$ at $t = 0$.

Remark 2.4. Presence of averaged temporal operators forces separation of two robustness measures (positive and negative). Assume otherwise, i.e. that we have one robustness measure that can take both positive and negative values; then

---

[2] There is no distinction between strict inequalities ($<$) and non-strict ones ($\leq$). This is inevitable in the current robustness framework. This is also the case with **STL** in [12,13].

robustness that floats between positive and negative values over time can "cancel out" after an average is taken. This leads to the failure of *soundness* (see Propositions 2.9 and 2.10; also [13,16]), and then a positive robustness value no longer witnesses the Boolean truth of (the qualitative variant of) the formula. This is not convenient in the application to falsification.

2.3   Basic Properties of AvSTL

Lemma 2.5 (Temporal Monotonicity). *Let $0 \leq t_0 < t \leq t'$. The following hold.*

$$\llbracket \sigma, \varphi_1 \, \mathcal{U}_{[t_0,t]} \, \varphi_2 \rrbracket^+ \leq \llbracket \sigma, \varphi_1 \, \mathcal{U}_{[t_0,t']} \, \varphi_2 \rrbracket^+ \quad \llbracket \sigma, \varphi_1 \, \mathcal{U}_{[t_0,t]} \, \varphi_2 \rrbracket^- \leq \llbracket \sigma, \varphi_1 \, \mathcal{U}_{[t_0,t']} \, \varphi_2 \rrbracket^-$$
$$\llbracket \sigma, \varphi_1 \, \mathcal{R}_{[t_0,t]} \, \varphi_2 \rrbracket^+ \geq \llbracket \sigma, \varphi_1 \, \mathcal{R}_{[t_0,t']} \, \varphi_2 \rrbracket^+ \quad \llbracket \sigma, \varphi_1 \, \mathcal{R}_{[t_0,t]} \, \varphi_2 \rrbracket^- \geq \llbracket \sigma, \varphi_1 \, \mathcal{R}_{[t_0,t']} \, \varphi_2 \rrbracket^-$$

The inequalities hold also for the averaged temporal operators.   □

We can now see well-definedness of Definition 2.3: we need that the integrals are defined; and the lemma shows that the integrated functions are monotone, hence Riemann integrable.

In Definition 2.3, the definitions for averaged operators with an infinite endpoint (like $\overline{\mathcal{U}}_{[0,\infty)}\varphi$) are given in terms of non-averaged operators. This is so that their well-definedness is immediate; the following lemma justifies those definitions.

Lemma 2.6. *For any $t_0 \in \mathbb{R}_{\geq 0}$, $\llbracket \sigma, \varphi_1 \, \mathcal{U}_{[t_0,\infty)} \, \varphi_2 \rrbracket^+ = \lim_{t \to \infty} \llbracket \sigma, \varphi_1 \, \overline{\mathcal{U}}_{[t_0,t]} \, \varphi_2 \rrbracket^+$. The same is true if we replace $\llbracket \_ \rrbracket^+$ with $\llbracket \_ \rrbracket^-$, and if we replace $\mathcal{U}$ with $\mathcal{R}$.*   □
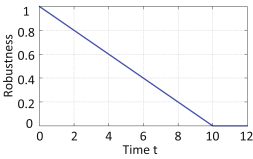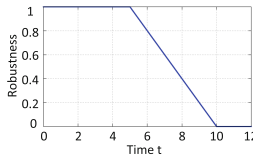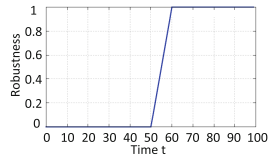
Fig. 2. Expeditiousness   **Fig. 3.** Deadline   **Fig. 4.** Persistence

2.4   Common Temporal Specifications Expressed in AvSTL

Here we shall exemplify the expressivity of **AvSTL**, by encoding typical temporal specifications encountered in the model-based development of cyber-physical systems.

Remark 2.7. In what follows we sometimes use *propositional variables* such as `airbag` and gear_i. For example, gear_2 is a shorthand for the atomic formula $x_{\text{gear}_2} \geq 0$ in **AvSTL**, where the variable x_{gear_2} is assumed to take a discrete value (1 or −1).

Expeditiousness $(\overline{\diamondsuit}_I \varphi)$**.** Consider the following informal specification: *after heavy braking, the airbag must operate within 10 ms.* Its formalization in **STL** is straightforward by the formula $\Box(\texttt{heavyBraking} \rightarrow \diamondsuit_{[0,10]} \texttt{airbag})$. However, an airbag that operates after 1 ms. is naturally more desirable than one that operates after 9.99 ms. The **STL** formula fails to discriminate between these two airbags.

Such *expeditiousness* ("as soon as possible") requirements are more adequately modeled in **AvSTL**, using the averaged-eventually modality $\overline{\diamondsuit}_I$. See Fig. 2, where the horizontal axis is for time t. The vertical axis in the figure stands for the positive robustness value $[\![\sigma_t, \overline{\diamondsuit}_{[0,10]} \texttt{airbag}]\!]^+$ of the formula $\overline{\diamondsuit}_{[0,10]} \texttt{airbag}$, where σ_t is a signal in which airbag operates (i.e. $x_{\texttt{airbag}}$ becomes from -1 to 1) at time t. We see that the formula successfully distinguishes an early-bird airbag from a lazy one.

Therefore the **AvSTL** formula $\Box(\texttt{heavyBraking} \rightarrow \overline{\diamondsuit}_{[0,10]} \texttt{airbag})$ formalizes a (refined) informal specification that: after heavy braking, the airbag must operate within 10 ms; *but the sooner the better*. It is not hard to expect that the latter is more faithful to the designer's intention than the original informal specification.

Deadline $(\diamondsuit_{[0,T]} \varphi \vee \overline{\diamondsuit}_{[T,T+\delta]} \varphi)$**.** The expeditiousness-type requirement that we have discussed is sometimes too strict. Let us consider the following scenario: there is a deadline set at time T and arrival by then is rewarded no matter how late; and then there is a deadline extension by time δ and arrival between the deadline and the extended one is rewarded too, but with certain deduction.

Such a *deadline* specification is expressed in **AvSTL** by the formula $\diamondsuit_{[0,T]} \varphi \vee \overline{\diamondsuit}_{[T,T+\delta]} \varphi$, combining non-averaged and averaged eventually modalities. See Fig. 3, where the positive robustness of the formula $(\diamondsuit_{[0,5]} \texttt{airbag}) \vee (\overline{\diamondsuit}_{[5,5+5]} \texttt{airbag})$ is plotted, for the same signals σ_t as before (i.e. in σ_t the airbag operates at time t).

Persistence $(\Box_{[0,T]} \varphi \wedge \overline{\Box}_{[T,T+\delta]} \varphi)$**.** *Persistence* ("for as long as possible") specifications are dual to deadline ones and expressed by a formula $\Box_{[0,T]} \varphi \wedge \overline{\Box}_{[T,T+\delta]} \varphi$. An example is the following informal specification on automatic transmission: *when a gear shifts into first, it never shifts into any other gear for the coming 50 ms.* A likely intention behind it is to prevent mechanical wear of gears that is caused by frequent gear shifts. In this case the following specification would be more faithful to the intention: when a gear shifts into first, it never shifts into any other gear for the coming 50 ms., *and preferably for longer*. This is formalized by the formula $\Box(\texttt{shiftIntoGear}_1 \rightarrow \Box_{[0,50]} \texttt{gear}_1 \wedge \overline{\Box}_{[50,50+\delta]} \texttt{gear}_1)$.

For illustration, Fig. 4 plots the positive robustness of $\Box_{[0,50]} \texttt{gear}_1 \wedge \overline{\Box}_{[50,60]} \texttt{gear}_1$ for signals σ'_t, where $\texttt{gear}_1$ is true in σ'_t from time 0 to t, and is false afterwards.

Other Temporal Specifications. Expressivity of **AvSTL** goes beyond the three examples that we have seen—especially after the extension of the language

with *time-reversed* averaged temporal operators. The reversal of time here corresponds to the symmetry between *left* and *right* time robustness in [13]. Such an extension of **AvSTL** enables us to express specifications like *punctuality* ("no sooner, no later") and *periodicity*. The details will be reported in another venue.

2.5   Soundness of Refinements from STL to AvSTL

In Sect. 2.4 we have seen some scenarios where an **STL** specification is *refined* into an **AvSTL** one so that it more faithfully reflects the designer's intention. The following two are prototypical:

- **(◇-refinement)** the refinement of $\Diamond_I \varphi$ ("eventually φ, within I") into $\overline{\Diamond}_I \varphi$ ("eventually φ within I, but as soon as possible"); and
- **(□-refinement)** the refinement of $\Box_{[a,b]} \varphi$ ("always φ throughout $[a,b]$") into $\Box_{[a,b]} \varphi \wedge \overline{\Box}_{[b,b+\delta]} \varphi$ ("always φ throughout $[a,b]$, and desirably also in $[b,b+\delta]$").

The following *soundness* results guarantee validity of the use of these refinements in falsification problems. *Completeness*, in a suitable sense, holds too.

Definition 2.8. A *positive context* is an **AvSTL** formula with a hole $[\,]$ at a positive position. Formally, the set of positive contexts is defined as follows:

$$\mathcal{C} ::= [\,] \mid \mathcal{C} \vee \varphi \mid \varphi \vee \mathcal{C} \mid \mathcal{C} \wedge \varphi \mid \varphi \wedge \mathcal{C} \mid \mathcal{C} \, \mathcal{U}_I \, \varphi \mid \varphi \, \mathcal{U}_I \, \mathcal{C} \mid \mathcal{C} \, \overline{\mathcal{U}}_I \, \varphi \mid \varphi \, \overline{\mathcal{U}}_I \, \mathcal{C}$$
$$\mid \mathcal{C} \, \mathcal{R}_I \, \varphi \mid \varphi \, \mathcal{R}_I \, \mathcal{C} \mid \mathcal{C} \, \overline{\mathcal{R}}_I \, \varphi \mid \varphi \, \overline{\mathcal{R}}_I \, \mathcal{C} \quad \text{where } \varphi \text{ is an } \mathbf{AvSTL} \text{ formula.}$$

For a positive context $\mathcal{C}$ and an **AvSTL** formula ψ, $\mathcal{C}[\psi]$ denotes the formula obtained by substitution of ψ for the hole $[\,]$ in $\mathcal{C}$.

Proposition 2.9 (Soundness and Completeness of ◇-Refinement). *Let $\mathcal{C}$ be a positive context. Then $[\![\sigma, \mathcal{C}[\overline{\Diamond}_{[a,b]}\varphi]]\!]^+ > 0$ implies $[\![\sigma, \mathcal{C}[\Diamond_{[a,b]}\varphi]]\!]^+ > 0$. Moreover, for any b' such that $b' < b$, $[\![\sigma, \mathcal{C}[\Diamond_{[a,b']}\varphi]]\!]^+ > 0$ implies $[\![\sigma, \mathcal{C}[\overline{\Diamond}_{[a,b]}\varphi]]\!]^+ > 0$.* □

Proposition 2.10 (Soundness and Completeness of □-Refinement). *Let $\mathcal{C}$ be a positive context. Then $[\![\sigma, \mathcal{C}[\Box_{[a,b]}\varphi \wedge \overline{\Box}_{[b,b+\delta]}\varphi]]\!]^+ > 0$ implies $[\![\sigma, \mathcal{C}[\Box_{[a,b]}\varphi]]\!]^+ > 0$. Moreover, for any $b' > b$, $[\![\sigma, \mathcal{C}[\Box_{[a,b']}\varphi]]\!]^+ > 0$ implies $[\![\sigma, \mathcal{C}[\Box_{[a,b]}\varphi \wedge \overline{\Box}_{[b,b+\delta]}\varphi]]\!]^+ > 0$.* □

2.6   Relationship to Previous Robustness Notions

Our logic **AvSTL** captures *space robustness* [16]—the first robustness notion proposed for **MITL/STL**, see Sect. 1—because the averaging-free fragment of **AvSTL** coincides with **STL** and its space robust semantics, modulo the separation of positive and negative robustness (Remark 2.4).

The relationship to *space-time robustness* proposed in [13] is interesting. In [13] they combine time and space robustness in the following way: for each time t and each space robustness value $c > 0$, *(right) time robustness relative to c*, denoted by $\theta_c^+(\varphi, \sigma, t)$, is defined by "how long after time t the formula φ maintains space robustness c." See the figure on the right, where the space-time robustness $\theta_c^+(x \geq 0, \sigma, 0)$ is depicted.
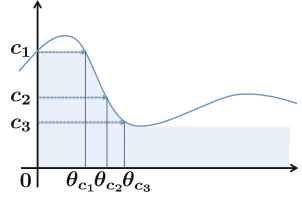
After all, space-time robustness in [13] is a function from c to $\theta_c^+(\varphi, \sigma, t)$; and one would like some real number as its characteristic. A natural choice of such is the *area* surrounded by the graph of the function (the shaded area in the figure), and it is computed in the same way as *Lebesgue integration*, as the figure suggests.

What corresponds in our **AvSTL** framework to this "area" characteristic value is the robustness of the formula $\overline{\Box}_{[0,\infty)}(x \geq 0)$ computed by Riemann integration (here we have to ignore the normalizing factor $\frac{1}{b-a}$ in Table 1). Therefore, very roughly speaking: our "averaged" robustness is a real-number characteristic value of the space-time robustness in [13]; and the correspondence is via the equivalence between Riemann and Lebesgue integration.

3   A Sliding-Window Algorithm for AvSTL Robustness

We shall present an algorithm for computing **AvSTL** robustness. It turns out that the presence of averaged modalities like $\overline{\Diamond}_I$—with an apparent nonlocal nature—does not incur severe computational overhead, at least for formulas in which averaged modalities are not nested. The algorithm is an adaptation of the one in [12] for **STL** robustness; the latter in turn relies on the *sliding window minimum algorithm* [22]. The algorithm's time complexity is linear with respect to the number of timestamps in the input signal; it exhibits a practical speed, too, as we will see later in Sect. 4.

Firstly we fix the class of signals to be considered.

Definition 3.1 (Finitely   Piecewise-Constant/Piecewise-Linear   Signal). A 1-dimensional signal $\sigma \colon \mathbb{R}_{\geq 0} \to \mathbb{R}$ is *finitely piecewise-constant (FPC)* if it arises from a finite sequence $\big[\,(t_0, r_0), (t_1, r_1), \ldots, (t_n, r_n)\,\big]$ of timestamped values, via the correspondence $\sigma(t) = r_i$ (for $t \in [t_i, t_{i+1})$). Here $0 = t_0 < \cdots < t_n$, $r_i \in \mathbb{R}$, and t_{n+1} is deemed to be ∞.

Similarly, a 1-dimensional signal $\sigma \colon \mathbb{R}_{\geq 0} \to \mathbb{R}$ is *finitely piecewise-linear (FPL)* if it is identified with a finite sequence $\big[\,(t_0, r_0, q_0), \ldots, (t_n, r_n, q_n)\,\big]$ of timestamped values, via the correspondence $\sigma(t) = r_i + q_i(t - t_i)$ (for $t \in [t_i, t_{i+1})$). Here $q_i \in R$ is the slope of σ in the interval $[t_i, t_{i+1}]$.

The definitions obviously extend to many-dimensional signals $\sigma \colon \mathbb{R}_{\geq 0} \to \mathbb{R}^{\mathbf{Var}}$.

We shall follow [12,13] and measure an algorithm's complexity in terms of the number of timestamps (n in the above); the latter is identified with the *size* of a signal.

Definition 3.2 (Robustness Signal $[\varphi]_\sigma$). Let $\sigma : \mathbb{R}_{\geq 0} \to \mathbb{R}^{\mathbf{Var}}$ be a signal, and φ be an **AvSTL** formula. The *positive robustness signal* of φ over σ is the signal $[\varphi]_\sigma^+ : \mathbb{R}_{\geq 0} \to \mathbb{R}$ defined by: $[\varphi]_\sigma^+(t) \triangleq [\![\sigma^t, \varphi]\!]^+$. Recall that $\sigma^t(t') = \sigma(t + t')$ is the t-shift of σ (Definition 2.2). The *negative robustness signal* $[\varphi]_\sigma^-$ is defined in the same way.

An averaged modality turns a piecewise-constant signal into a piecewise-linear one.

Lemma 3.3

1. *Let φ be an averaging-free **AvSTL** formula. If a signal σ is finitely piecewise-constant (or piecewise-linear), then so is $[\varphi]_\sigma^+$.*
2. *Let φ be an **AvSTL** formula without nested averaged modalities. If a signal σ is finitely piecewise-constant, then $[\varphi]_\sigma^+$ is finitely piecewise-linear.*

The above holds for the negative robustness signal $[\varphi]_\sigma^-$, too.

Proof. Straightforward by the induction on the construction of formulas.   □

Our algorithm for computing **AvSTL** robustness $[\![\sigma, \varphi]\!]$ will be focused on: (1) a finitely piecewise-constant input signal σ; and (2) an **AvSTL** formula φ where averaged modalities are not nested. In what follows, for presentation, we use the (non-averaged and averaged) eventually modalities $\diamondsuit_I, \overline{\diamondsuit}_I$ in describing algorithms. Adaptation to other modalities is not hard; for complex formulas, we compute the robustness signal $[\varphi]_\sigma$ by induction on φ.

3.1   Donzé et al.'s Algorithm for STL Robustness

We start with reviewing the algorithm [12] for **STL** robustness. Our algorithm for **AvSTL** robustness relies on it in two ways: 1) the procedures for averaged modalities like $\overline{\diamondsuit}_I$ derive from those for non-averaged modalities in [12]; and (2) we use the algorithm in [12] itself for the non-averaged fragment of **AvSTL**.

Remark 3.4. The algorithm in [12] computes the **STL** robustness $[\![\sigma, \varphi]\!]$ for a finitely piecewise-*linear* signal σ. We need this feature e.g. for computing robustness of the formula $\square(\texttt{heavyBraking} \to \overline{\diamondsuit}_{[0,10]}\texttt{airbag})$: note that, by Lemma 3.3, the robustness signal for $\overline{\diamondsuit}_{[0,10]}\texttt{airbag}$ is piecewise-linear even if the input signal is piecewise-constant.

Consider computing the robustness signal $[\diamondsuit_{[a,b]}\varphi]_\sigma$, assuming that the signal $[\varphi]_\sigma$ is already given.[3] The task calls for finding the supremum of $[\varphi]_\sigma(\tau)$ over

---

[3] In the rest of Sect. 3.1, for simplicity of presentation, we assume that $[\varphi]_\sigma$ is piecewise-constant. We note that the algorithm in [12] nevertheless extends to piecewise-linear $[\varphi]_\sigma$.
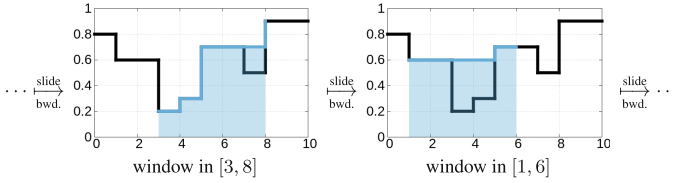
Fig. 5. A sliding window for computing $[\Diamond_{[0,5]}(x \geq 0)]_\sigma^+$; the black line is the signal σ
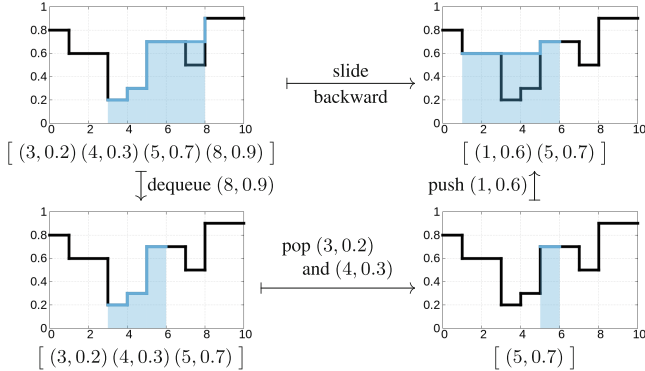

Fig. 6. Use of stackqueues and their operations, in the sliding window algorithm

$\tau \in [t + a, t + b]$; and this must be done for each t. Naively doing so leads to quadratic complexity.

Instead Donzé et al. in [12] employ a *sliding window* of size $b-a$ and let it scan the signal $[\varphi]_\sigma$ from right to left. The scan happens once for all, hence achieving linear complexity. See Fig. 5, where we take $[\Diamond_{[0,5]}(x \geq 0)]_\sigma^+$ as an example, and the blue shaded area designates the position of the sliding window. The window slides from $[3, 8]$ to the closest position to the left where its left-endpoint hits a new timestamped value of $[\varphi]_\sigma$, namely $[1, 6]$.

It is enough to know the shape of the blue (partial) signal in Fig. 5, at each position of the window. The blue signal denotes the (black) signal σ's local supremum within the window; more precisely, it denotes

the value of the signal $[\![\sigma^t, \Diamond_{[0,\tau]}(x \geq 0)]\!]^+$ at time $t + \tau$, where $\tau \in [0, 5]$ and t is the leftmost position of the window. We can immediately read off the signal $[\Diamond_{[0,5]}(x \geq 0)]_\sigma^+$ from the blue signals: the former is the latter's value at the rightmost position of the window.

The keys in the algorithms in [12,22] lie in:

- use of the *stackqueue* data structure (depicted above on the right) for the purpose of representing the blue (partial) signal in Fig. 5; and
- use of the operations *push*, *pop* and *dequeue* for updating the blue signal.

See Fig. 6, where each entry of a stackqueue is a timestamped value (t, r). We see that the slide of the window, from top-left to top-right in Fig. 6, is expressed by

---
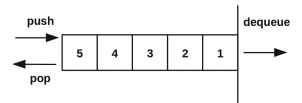
Algorithm 1. An algorithm for computing $[\overline{\Diamond}_{[a,b]}\varphi]_\sigma$

Require: An FPC signal $[\varphi]_\sigma$ given as a sequence $(t_0, r_0), \ldots, (t_n, r_n)$
Ensure: The FPL signal $[\overline{\Diamond}_{[a,b]}\varphi]_\sigma$

  $t_{\mathsf{temp}} := t_n - a$;
  $F := \big[\, (t_{\mathsf{temp}} + a, [\varphi]_\sigma(t_{\mathsf{temp}} + a)) \,\big]$; $\triangleright$ F is the FPC signal $\tau \mapsto [\![\sigma^t, \Diamond_{[a,\tau]}\varphi]\!]$
  $s := (b - a) \cdot [\varphi]_\sigma(t_{\mathsf{temp}} + a)$; $\triangleright$ The area of F
  $G := \big[\, (t_{\mathsf{temp}}, s/(b-a), 0) \,\big]$; $\triangleright$ The FPC signal $[\overline{\Diamond}_{[a,b]}\varphi]_\sigma$
  while $t_{\mathsf{temp}} \geq 0$ **do**
    $t_{\mathsf{old}} := t_{\mathsf{temp}}$;
    $t_{\mathsf{temp}} :=$ the greatest t such that $t < t_{\mathsf{old}} \wedge \big(\exists t_i.\, t + a = t_i \vee \exists (t', r') \in F.\, t + b = t'\big)$;
    $\mathsf{Deq} := \{(t, r) \in F \mid t > t_{\mathsf{temp}} + b\}$; $F := F \setminus \mathsf{Deq}$; $\triangleright$ Dequeue old elements in F
    $\mathsf{Pop} := \{(t, r) \in F \mid r \leq [\varphi]_\sigma(t_{\mathsf{temp}} + a)\}$; $F := F \setminus \mathsf{Pop}$; $\triangleright$ Pop small elements in F
    $t_{\mathsf{Pop}} := \min\{t \mid (t, r) \in F \ \text{or} \ t = t_{\mathsf{temp}} + b\}$;
    $F := \big[(t_{\mathsf{temp}} + a, [\varphi]_\sigma(t_{\mathsf{temp}} + a))\big] \cup F$ $\triangleright$ Push the left endpoint of the window to F
    $r_{\mathsf{left}} := \min\{r \mid (t, r) \in F\}$;
    $r_{\mathsf{right}} := \max\{r \mid (t, r) \in F\}$;
    $s := s - (t_{\mathsf{old}} - t_{\mathsf{temp}}) \cdot r_{\mathsf{right}} - \mathsf{area}(\mathsf{Pop}) + (t_{\mathsf{Pop}} - (t_{\mathsf{temp}} + a)) \cdot r_{\mathsf{left}}$
    $G := \{(t_{\mathsf{temp}}, s/(b-a), r_{\mathsf{right}} - r_{\mathsf{left}})\} \cup G$
  end while

---

dequeue, pop and then push operations to stackqueues (in Fig. 6: from top-left to bottom-left, bottom-right and then top-right). Pseudocode for the algorithm can be found in [4, Appendix A.1].

3.2 An Algorithm for AvSTL Robustness

It turns out that the last algorithm is readily applicable to computing **AvSTL** robustness. Consider an averaged-eventually formula $\overline{\Diamond}_{[a,b]}\varphi$ as an example. What we have to compute is the size of the shaded areas in Fig. 5 (see also Fig. 1); and the shape of the blue signals in Fig. 5 carry just enough information to do so.

Pseudocode for the adaptation of the previous algorithm (in Sect. 3.1) to $\overline{\Diamond}_{[a,b]}\varphi$ is found in Algorithm 1. Its complexity is linear with respect to the number n of the timestamp values that represent the signal $[\varphi]_\sigma$.

An algorithm for the averaged-henceforth formula $[\overline{\Box}_{[a,b]}\varphi]_\sigma$ is similar. Extensions to averaged-until and averaged-release operators are possible, too; they use doubly-linked lists in place of stackqueues. See [4, Appendix A.2] for more details. Combining with the algorithm in Sect. 3.1 to deal with non-averaged temporal operators, we have the following complexity result. The complexity is the same as for **STL** [12].

Theorem 3.5. *Let φ be an **AvSTL** formula in which averaged modalities are not nested. Let σ be a finitely piecewise-constant signal. Then there exists an algorithm to compute $[\![\sigma, \varphi]\!]^+$ with time-complexity in $\mathcal{O}(d^{|\varphi|}|\varphi||\sigma|)$ for some constant d.*

The same is true for the negative robustness $[\![\sigma, \varphi]\!]^-$. $\square$

Remark 3.6. The reason for our restriction to finitely piecewise-constant input signals is hinted in Remark 3.4; let us further elaborate on it. There the averaged modality $\overline{\diamondsuit}_{[0,10]}$ turns a piecewise-constant signal into a piecewise-linear one (Lemma 3.3); and then the additional Boolean connectives and non-averaged modalities (outside $\overline{\diamondsuit}_{[0,10]}$) are taken care of by the algorithm in [12], one that is restricted to piecewise-linear input.

It is not methodologically hard to extend this workflow to piecewise-*polynomial* input signals (hence to nested averaged modalities as well). Such an extension however calls for computing local suprema of polynomials, as well as their intersections—tasks that are drastically easier with affine functions. We therefore expect the extension to piecewise-polynomial signals to be computationally much more expensive.

4   Enhanced Falsification: Implementation and Experiments

We claim that our logic **AvSTL** achieves a good balance between expressivity—that communicates a designer's intention more faithfully to a falsification solver—and computational cost, thus contributing to the model-based development of cyber-physical systems. In this section we present our implementation that combines: (1) S-TaLiRo [7], one of the state-of-art falsification solvers that relies on robust MTL semantics and stochastic optimization; and (2) the **AvSTL** *evaluator*, an implementation of the algorithm in Sect. 3.2. Our experiments are on automotive examples of falsification problems; the results indicate that (refinement of specifications by) **AvSTL** brings considerable performance improvement.
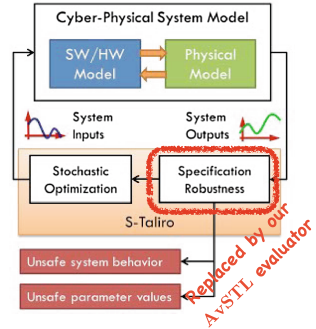
Fig. 7. An overview of S-TaLiRo (from [1]), with our modification

Implementation. S-TaLiRo [7] is "a Matlab toolbox that searches for trajectories of minimal robustness in Simulink/Stateflow" [1]. Recall the formalization of a falsification problem (Sect. 1). S-TaLiRo's input is: (1) a model $\mathcal{M}$ that is a Simulink/Stateflow model; and (2) a specification φ that is an **STL** formula.

S-TaLiRo employs stochastic simulation in the following *S-TaLiRo loop*:

1. Choose an input signal σ_{in} randomly.
2. Compute the output signal $\mathcal{M}(\sigma_{\mathsf{in}})$ with Simulink.
3. Compute the robustness $[\![\mathcal{M}(\sigma_{\mathsf{in}}), \varphi]\!]$.
4. If the robustness is ≤ 0 then return σ_{in} as a critical path. Otherwise choose a new σ_{in} (hopefully with a smaller robustness) and go back to Step 2.

Our modification of S-TaLiRo consists of: 1) changing the specification formalism from **STL** to **AvSTL** (with the hope that the robustness $[\![\mathcal{M}(\sigma_{\mathsf{in}}), \varphi]\!]^{+}$ carries

Table 2. Experiment results. Time is in seconds. The "Succ." columns show how many trials succeeded among the designated number of trials; the "Iter." columns show the average number of iterations of the S-TaLiRo loop, executed in each trial (max. 1000); and the "Time" columns show the average time that each trial took. For the last two we also show the average over *successful* trials.

Problem 1. Falsification means finding an input signal that keeps the engine speed ω below 2000 rpm, for T seconds. The bigger T is, the harder the problem is. We applied $\diamond$-refinement.

Problem 1	$T = 20$			$T = 30$			$T = 40$		
Specification to be falsified	Succ. /**100**	Iter. (Succ.)	Time (Succ.)	Succ. /**100**	Iter. (Succ.)	Time (Succ.)	Succ. /**100**	Iter. (Succ.)	Time (Succ.)
$\diamond_{[0,T]}(\omega \geq 2000)$	100	128.8	20.2	81	440.9	82.5	32	834.3	162.9
		128.8	20.2		309.7	59.0		482.2	94.4
$\overline{\diamond}_{[0,T]}(\omega \geq 2000)$	100	123.9	22.9	98	249.8	46.1	81	539.6	110.9
		123.9	22.9		234.5	43.4		431.6	89.2

Problem 2. Falsification means finding an input signal that keeps ω within a range of 3500–4500 rpm for T consecutive seconds, at a certain stage. We applied $\diamond$-refinement.

Problem 2	$T = 10$		
Specification to be falsified	Succ. /**100**	Iter. (Succ.)	Time (Succ.)
$\Box\diamond_{[0,T]}(\omega \leq 3500 \vee \omega \geq 4500)$	45	625.4	209.1
		167.7	56.1
$\Box\overline{\diamond}_{[0,T]}(\omega \leq 3500 \vee \omega \geq 4500)$	74	442.0	154.3
		245.9	86.6

Problem 3. Falsification means finding an input signal that shifts the gear into the fourth within T seconds. The smaller T is, the harder the problem is. Here $\mathtt{gear}_4$ is a propositional variable. We applied $\Box$-refinement.

Problem 3	$T = 4$			$T = 4.5$			$T = 5$		
Specification to be falsified	Succ. /**20**	Iter. (Succ.)	Time (Succ.)	Succ. /**20**	Iter. (Succ.)	Time (Succ.)	Succ. /**20**	Iter. (Succ.)	Time (Succ.)
$\Box_{[0,T]}\neg\mathtt{gear}_4$	0	1000	166.9	11	742.8	122.9	18	449.0	71.8
		−	−		532.3	87.5		387.7	61.9
$\Box_{[0,T]}\neg\mathtt{gear}_4 \wedge \Box_{[T,10]}\neg\mathtt{gear}_4$	17	570.1	94.0	20	250.5	40.3	20	107.5	17.6
		494.2	81.8		250.5	40.3		107.5	17.6

Problem 4. Falsification means finding input with which the gear never stays in the third consecutively for T seconds. The smaller T is, the harder the problem is. Here $\mathtt{gear}_3$ is a propositional variable. We applied $\Box$-refinement.

Problem 4	$T = 1$			$T = 2$		
Specification to be falsified	Succ. /**20**	Iter. (Succ.)	Time (Succ.)	Succ. /**20**	Iter. (Succ.)	Time (Succ.)
$\diamond\left(\Box_{[0,T]}\mathtt{gear}_3\right)$	14	556.1	132.0	20	82.8	20.6
		365.8	87.1		82.8	20.6
$\diamond\left(\Box_{[0,T]}\mathtt{gear}_3 \wedge \overline{\Box}_{[T,10]}\mathtt{gear}_3\right)$	20	105.1	36.3	20	29.7	10.2
		105.1	36.3		29.7	10.2

Problem 5. Falsification means finding input that violates the following requirement: *after the gear is shifted, it stays the same for T seconds.* (the smaller T, the harder). $\mathtt{gear}_1, \ldots, \mathtt{gear}_4$ are propositional variables. We applied $\Box$-refinement.

Problem 5 ($\varepsilon = 0.04$)	$T = 0.8$			$T = 1$			$T = 2$		
Specification to be falsified	Succ. /**20**	Iter. (Succ.)	Time (Succ.)	Succ. /**20**	Iter. (Succ.)	Time (Succ.)	Succ. /**20**	Iter. (Succ.)	Time (Succ.)
$\bigwedge_{i=1,\ldots,4} \Box\left((\neg\mathtt{gear}_i \wedge \diamond_{[0,\varepsilon]}\mathtt{gear}_i) \rightarrow (\Box_{[\varepsilon,T+\varepsilon]}\mathtt{gear}_i)\right)$	2	972.5	402.5	19	356.8	155.6	20	27.4	11.8
		724.5	297.8		322.9	140.9		27.4	11.8
$\bigwedge_{i=1,\ldots,4} \Box\left((\neg\mathtt{gear}_i \wedge \diamond_{[0,\varepsilon]}\mathtt{gear}_i) \rightarrow (\Box_{[\varepsilon,T+\varepsilon]}\mathtt{gear}_i \wedge \overline{\Box}_{[T+\varepsilon,5]}\mathtt{gear}_i)\right)$	12	561.1	349.1	20	93.1	57.8	20	42.7	26.9
		268.5	167.3		93.1	57.8		42.7	26.9

Problem 6. Falsification means finding an input signal that steers the vehicle speed v over 85 kph within T seconds, while keeping the engine speed ω below 4500 rpm. The smaller T is, the harder the problem is. We applied $\Box$-refinement.

Problem 6	$T = 10$			$T = 12$		
Specification to be falsified	Succ. /**20**	Iter. (Succ.)	Time (Succ.)	Succ. /**20**	Iter. (Succ.)	Time (Succ.)
$\Box_{[0,T]}(v \leq 85) \vee \diamond(\omega \geq 4500)$	12	714.9	141.4	17	374.5	72.2
		524.9	108.1		264.1	51.2
$\left(\Box_{[0,T]}(v \leq 85) \wedge \overline{\Box}_{[T,20]}(v \leq 85)\right) \vee \diamond(\omega \geq 4500)$	12	766.7	149.0	20	423.6	85.7
		611.2	118.9		423.6	85.7

more information to be exploited in stochastic optimization); and 2) using, in Step 3 of the above loop, the **AvSTL** evaluator based on the sliding-window algorithm in Sect. 3. See Fig. 7.

Experiments. As a model $\mathcal{M}$ we used the automatic transmission model from [18], where it is offered "as benchmarks for testing-based falsification" [18]. The same model has been used in several works [15,20,25]. The model has two input ports (throttle and brake) and six output ports (the engine speed ω, the vehicle speed v, and four mutually-exclusive Boolean ports $\mathtt{gear}_1, \ldots, \mathtt{gear}_4$ for the current gear). See [4, Appendix C] for more details. As a specification φ to falsify, we took six examples from [18], sometimes with minor modifications. They constitute Problems 1–6 in Table 2.

Our goal is to examine the effect of our modification to S-TaLiRo. For the model $\mathcal{M}$ (that is fixed) and each of the six specifications φ, experiments are done with:

– $\mathcal{M}$ and the original **STL** formula φ, as a control experiment; and
– $\mathcal{M}$ and the **AvSTL** formula φ' that is obtained from φ as a refinement. The latter specifically involves $\diamondsuit$-*refinement* and $\square$-*refinement* described in Sect. 2.5.

Faster, or more frequent, falsification in the latter setting witnesses effectiveness of our **AvSTL** approach. We note that falsifying φ' indeed means falsifying φ, because of the soundness of the refinement (Propositions 2.9 and 2.10).

A single falsification *trial* consists of at most 1000 *iterations* of the S-TaLiRo loop. For each specification φ (i.e. for each problem in Table 2) we made 20–100 falsification trials, sometimes with different parameter values T. We made multiple trials because of the stochastic nature of S-TaLiRo.

Experiment Results and Discussion. The experiment results are in Table 2. We used Matlab R2014b and S-TaLiRo ver.1.6 beta on ThinkPad T530 with Intel Core i7-3520M 2.90GHz CPU with 3.7 GB memory. The OS is Ubuntu14.04 LTS (64-bit).

Notable performance improvement is observed in Problems 3–5, especially in their harder instances. For example, our **AvSTL** enrichment made 17 out of 20 trials succeed in Problem 3 ($T = 4$), while no trials succeeded with the original **STL** specification. A similar extreme performance gap is observed also in Problem 5 ($T = 0.8$).

Such performance improvement in Problems 3–5 is not surprising. The specifications for these problems are concerned solely with the propositional variables $\mathtt{gear}_i$ (cf. Remark 2.7); and the space robustness semantics for **STL** assigns to these specifications only 0 or 1 (but no values in-between) as their truth values. We can imagine such "discrete" robustness values give few clues to stochastic optimization algorithms.

Both of $\diamondsuit$- and $\square$-refinement in Sect. 2.5 turn out to be helpful. The latter's effectiveness is observed in Problems 3–5; the former improves a success rate from 32/100 to 81/100 in Problem 1 ($T = 40$).

Overall, the experiment results seem to support our claim that the complexity of (computing robustness values in) **AvSTL** is tractable. There is no big difference in the time each iteration takes, between the **STL** case and the **AvSTL** case.

5   Conclusions and Future Work

We introduced **AvSTL**, an extension of **STL** with *averaged* temporal operators. It adequately captures both space and time robustness; and we presented an algorithm for computing robustness that is linear-time with respect to the "size" of an input signal. Its use in falsification of CPS is demonstrated by our prototype that modifies S-TaLiRo.

As future work, we wish to compare our averaged temporal operators with other quantitative temporal operators, among which are the *discounting* ones [5,6]. The latter are closely related to *mean-payoff* conditions [10,14] as well as to *energy constraints* [8,9], all of which are studied principally in the context of automata theory.

Application of **AvSTL** to problems other than falsification is another important direction. Among them is *parameter synthesis*, another task that S-TaLiRo is capable of. We are now looking at application to *sequence classification* (see e.g. [21]), too, whose significant role in model-based development of CPS is widely acknowledged.

References

1. TaLiRo-tools. https://sites.google.com/a/asu.edu/s-taliro/s-taliro. Accessed 26 January 2015
2. Abbas, H., Hoxha, B., Fainekos, G.E., Deshmukh, J.V., Kapinski, J., Ueda, K.: Conformance testing as falsification for cyber-physical systems. In: CoRR, abs/1401.5200 (2014)
3. Abbas, H., Hoxha, B., Fainekos, G.E., Deshmukh, J.V., Kapinski, J., Ueda, K.: Wip abstract: conformance testing as falsification for cyber-physical systems. In: ACM/IEEE International Conference on Cyber-Physical Systems, ICCPS, Berlin, Germany, April 14–17, 2014, pp. 211. IEEE Computer Society (2014)
4. Akazaki, T., Hasuo, I.: Time robustness in MTL and expressivity in hybrid system falsification (extended version) (2015). http://arxiv.org/abs/1505.06307
5. Almagor, S., Boker, U., Kupferman, O.: Discounting in LTL. In: Ábrahám, E., Havelund, K. (eds.) TACAS 2014 (ETAPS). LNCS, vol. 8413, pp. 424–439. Springer, Heidelberg (2014)
6. Alur, R., Feder, T., Henzinger, T.A.: The benefits of relaxing punctuality. J. ACM **43**(1), 116–146 (1996)

7. Annpureddy, Y., Liu, C., Fainekos, G., Sankaranarayanan, S.: S-TaLiRo: a tool for temporal logic falsification for hybrid systems. In: Abdulla, P.A., Leino, K.R.M. (eds.) TACAS 2011. LNCS, vol. 6605, pp. 254–257. Springer, Heidelberg (2011)

8. Bouyer, P., Fahrenberg, U., Larsen, K.G., Markey, N., Srba, J.: Infinite runs in weighted timed automata with energy constraints. In: Cassez, F., Jard, C. (eds.) FORMATS 2008. LNCS, vol. 5215, pp. 33–47. Springer, Heidelberg (2008)

9. Brenguier, R., Cassez, F., Raskin, J.-F.: Energy and mean-payoff timed games. In: Fränzle, M., Lygeros, J. (eds.) Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control, pp. 283–292. ACM, New York (2014)

10. Chatterjee, K., Henzinger, T.A., Jurdzinski, M.: Mean-payoff parity games. In: 20th IEEE Symposium on Logic in Computer Science, LICS, 26–29 June 2005, Chicago, IL, USA, pp. 178–187. IEEE Computer Society (2005)

11. Donzé, A.: Breach, a toolbox for verification and parameter synthesis of hybrid systems. In: Touili, T., Cook, B., Jackson, P. (eds.) CAV 2010. LNCS, vol. 6174, pp. 167–170. Springer, Heidelberg (2010)

12. Donzé, A., Ferrère, T., Maler, O.: Efficient robust monitoring for STL. In: Sharygina, N., Veith, H. (eds.) CAV 2013. LNCS, vol. 8044, pp. 264–279. Springer, Heidelberg (2013)

13. Donzé, A., Maler, O.: Robust satisfaction of temporal logic over real-valued signals. In: Chatterjee, K., Henzinger, T.A. (eds.) FORMATS 2010. LNCS, vol. 6246, pp. 92–106. Springer, Heidelberg (2010)

14. Ehrenfeucht, A., Mycielski, J.: Positional strategies for mean payoff games. Int. J. Game Theor. **8**(2), 109–113 (1979)

15. Fainekos, G.E., Sankaranarayanan, S., Ueda, K., Yazarel, H.: Verification of automotive control applications using S-TaLiRo. Am. Control Conf. (ACC) **2012**, 3567–3572 (2012)

16. Fainekos, G.E., Pappas, G.J.: Robustness of temporal logic specifications for continuous-time signals. Theor. Comput. Sci. **410**(42), 4262–4291 (2009)

17. Fränzle, M., Lygeros, J. (eds.) 17th International Conference on Hybrid Systems: Computation and Control (part of CPS Week), HSCC 2014, Berlin, Germany, April 15–17, 2014. ACM (2014)

18. Hoxha, B., Abbas, H., Fainekos, G.: Benchmarks for temporal logic requirements for automotive systems. In: Proceedings of Applied Verification for Continuous and Hybrid Systems (2014)

19. Jin, X., Deshmukh, J.V., Kapinski, J., Ueda, K., Butts, K.: Powertrain control verification benchmark. In: Fränzle, M., Lygeros, J. (eds.) Proceedings of the 17th international conference on Hybrid systems: computation and control, pp. 253–262. ACM, New York (2014)

20. Jin, X., Donzé, A., Deshmukh, J.V., Seshia, S.A.: Mining requirements from closed-loop control models. In: Belta, C., Ivancic, F. (eds.) Proceedings of the 16th International Conference on Hybrid Systems: Computation and Control, HSCC 2013, pp. 43–52. ACM, New York (2013)

21. Kong, Z., Jones, A., Ayala, A.M., Gol, E.A., Belta, C.: Temporal logic inference for classification and prediction from data. In: Fränzle, M., Lygeros, J. (eds.) Proceedings of the 17th international conference on Hybrid systems: computation and control, pp. 273–282. ACM, New York (2014)

22. Lemire, D.: Streaming maximum-minimum filter using no more than three comparisons per element. Nord. J. Comput. **13**(4), 328–339 (2006)

23. Maler, O., Nickovic, D.: Monitoring temporal properties of continuous signals. In: Lakhnech, Y., Yovine, S. (eds.) FORMATS 2004 and FTRTFT 2004. LNCS, vol. 3253, pp. 152–166. Springer, Heidelberg (2004)

24. Sankaranarayanan, S., Fainekos, G.: Falsification of temporal properties of hybrid systems using the cross-entropy method. In: Proceedings of the 15th ACM International Conference on Hybrid Systems: Computation and Control, HSCC 2012, pp. 125–134. ACM, New York, NY, USA (2012)
25. Yang, H., Hoxha, B., Fainekos, G.: Querying parametric temporal logic properties on embedded systems. In: Nielsen, B., Weise, C. (eds.) ICTSS 2012. LNCS, vol. 7641, pp. 136–151. Springer, Heidelberg (2012)
26. Zutshi, A., Deshmukh, J.V., Sankaranarayanan, S., Kapinski, J.: Multiple shooting, cegar-based falsification for hybrid systems. In: Proceedings of the 14th International Conference on Embedded Software, EMSOFT 2014, pp. 5:1–5:10, ACM, New York, NY, USA (2014)