Time-Staging Enhancement of Hybrid System Falsification *

Gidon Ernst
LMU Munich, Germany
gidon.ernst@lmu.de

Ichiro Hasuo
National Institute of Informatics, Tokyo, Japan
hasuo@nii.ac.jp

Zhenya Zhang
Kyushu University, Japan
zhang.zhenya.623@m.kyushu-u.ac.jp

Sean Sedwards
University of Waterloo, Waterloo, Canada
sean.sedwards@uwaterloo.ca

Optimization-based falsification employs stochastic optimization algorithms to search for error input of hybrid systems. In this paper we introduce a simple idea to enhance falsification, namely *time staging*, that allows the *time-causal* structure of time-dependent signals to be exploited by the optimizers. Time staging consists of running a falsification solver multiple times, from one interval to another, incrementally constructing an input signal candidate. Our experiments show that time staging can dramatically increase performance in some realistic examples. We also present theoretical results that suggest the kinds of models and specifications for which time staging is likely to be effective.

1 Introduction

Hybrid Systems Quality assurance of *cyber-physical systems* (CPS) has been recognized as an important challenge, where many CPS are *hybrid systems* that combine the discrete dynamics of computers and the continuous dynamics of physical components. Unfortunately, analysis of hybrid systems poses unique challenges, such as the limited applicability of *formal verification*. In formal verification one aims to give a mathematical proof for a system's correctness. This is much harder for hybrid systems than for computer software/hardware, where the presence of continuous dynamics makes many problems more complex or even undecidable (e.g. reachability in hybrid automata).

Optimization-Based Falsification Because of these difficulties, an increasing number of researchers are turning to *optimization-based falsification* as a quality assurance measure. It is a testing method rather than that of formal verification; the problem is formalized as follows.

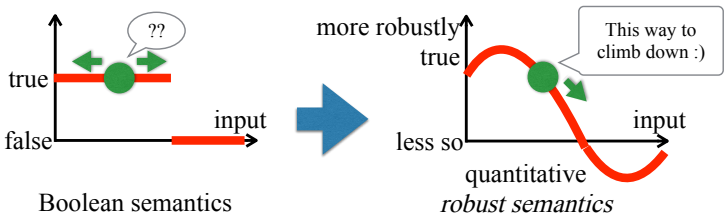
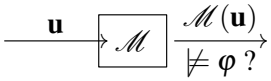


Figure 1: From Boolean to robust semantics

- **Given:** a *model* $\mathscr{M}$ (that takes an input signal $\mathbf{u}$ and yields an output signal $\mathscr{M}(\mathbf{u})$), and a *specification* φ (a temporal formula)
- **Answer:** *error input*, that is, an input signal $\mathbf{u}$ such that the corresponding output $\mathscr{M}(\mathbf{u})$ violates φ



*Presented at SNR'2018. G. Ernst and Z. Zhang were then at the National Institute of Informatics.

Remke & Tran (Eds.): Symbolic-Numeric Methods
for Reasoning about CPS and IoT (SNR'21)
EPTCS 361, 2022, pp. 25–43, doi:10.4204/EPTCS.361.5

In the optimization-based falsification approach, the above falsification problem is turned into an optimization problem. This is possible thanks to *robust semantics* of temporal formulas [14]. Instead of the Boolean satisfaction relation $\mathbf{v} \models \varphi$, robust semantics assigns a quantity $[\![\mathbf{v}, \varphi]\!] \in \mathbb{R} \cup \{\infty, -\infty\}$ that tells us, not only whether φ is true or not (by the sign), but also *how robustly* the formula is true or false. This allows one to employ hill-climbing optimization (see Fig. 1): we iteratively generate input signals, in the direction of decreasing robustness, hoping that eventually we hit negative robustness.

Optimization-based falsification is a subclass of *search-based testing*: it adaptively chooses test cases (input signals $\mathbf{u}$) based on previous observations. One can use stochastic algorithms for optimization (such as simulated annealing), which turn out to be much more scalable than many model checking algorithms that rely on exhaustive search. Note also that the system model $\mathscr{M}$ can be black box: it is enough to know the correspondence between input $\mathbf{u}$ and output $\mathscr{M}(\mathbf{u})$. An error input $\mathbf{u}$ is concrete evidence for the system's need for improvement, and thus has great appeal to practitioners.

The approach of optimization-based falsification was initiated in [14] and has been actively pursued ever since [3, 5, 7, 8, 10, 11, 20]. There are now mature tools, such as Breach [8] and S-Taliro [5], which work with industry-standard Simulink models.

Contribution We introduce a simple idea of *time staging* for enhancement of optimization-based falsification. Time staging consists of running a falsification solver repeatedly, from one input segment to another, incrementally constructing an input signal.

In general, in solving a concrete problem $\mathscr{C}$ by a metaheuristic $\mathscr{H}$ (such as stochastic optimization, evolutionary computation, etc.), a key to success is to communicate as much information as possible in the translation from $\mathscr{C}$ to $\mathscr{H}$—that is, to let $\mathscr{H}$ exploit structures unique to $\mathscr{C}$. Our idea of time staging follows this philosophy. More specifically, via time staging we communicate the *time-causal* structure of time-dependent signals—a structure that is present in some instances of the falsification problem but not in optimization problems in general—to stochastic optimization solvers.

Our implementation of time-staged falsification is based on Breach [8]. We show that this simple idea can dramatically enhance its performance in some examples. We also present some theoretical considerations on the kinds of problem instances where time staging is likely to work, and some results that aid implementation of time staging.

Structure of the Paper In §2 we informally outline optimization-based falsification and illustrate the idea of time staging. We then turn to formal developments: in §3 we review existing falsification works and in §4 we present our algorithm, augmented by some theoretical results that aid its implementation. §5 is devoted to the theoretical consideration of two specific settings in which time staging is guaranteed to work well. These settings will serve as useful "rules of thumb" for practical applications. In §6 we discuss our implementation and experimental results. We conclude in §8.

2 Schematic Overview: Falsification and Time Staging

We illustrate falsification and time staging, informally with an example.

Example Setting We take as a system model $\mathscr{M}$ a simple automotive powertrain whose input signal is 1-dimensional (the throttle u) and whose output signal is the vehicle speed v. We assume that $\mathscr{M}$ exhibits the following natural behavior: the larger u is, the quicker v grows. Let our specification be $\varphi \equiv \big(\Box(v \leq 120)\big)$, where $\equiv$ denotes the syntactic equality. To falsify φ the vehicle speed v must exceed

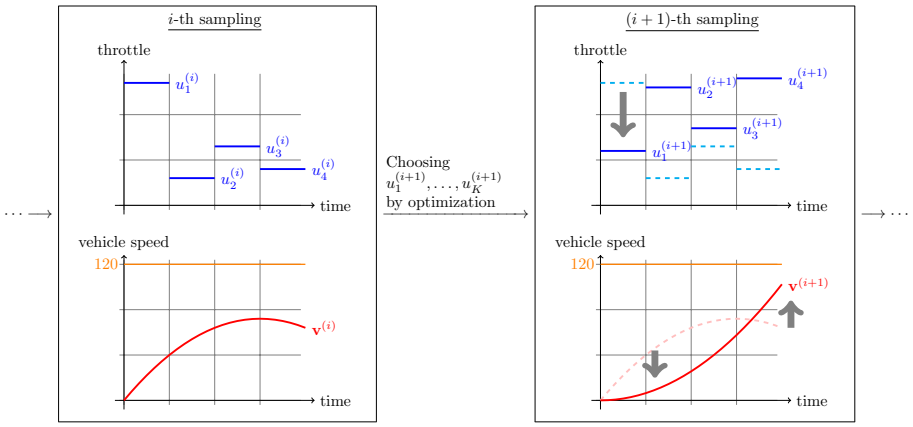


Figure 2: Conventional optimization-based falsification (without time staging).

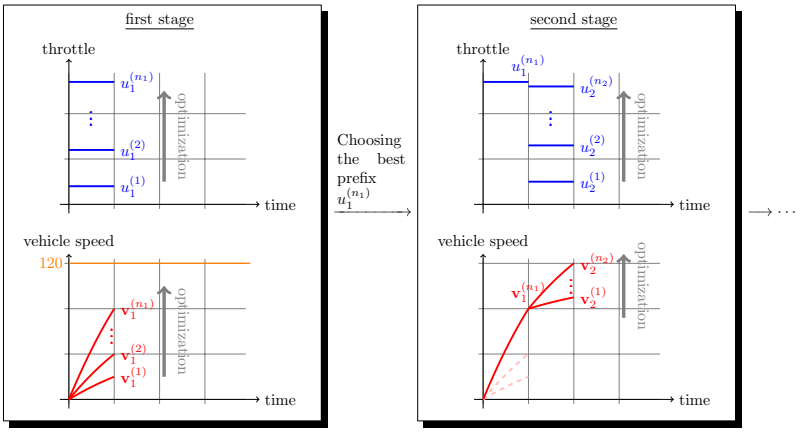


Figure 3: Falsification with time staging

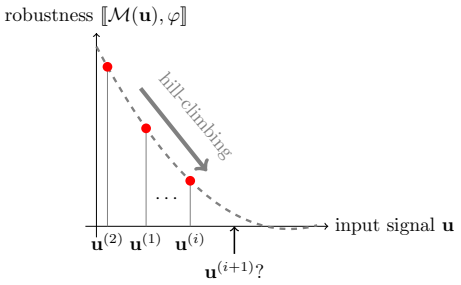


Figure 4: Hill-climbing optimization in falsification

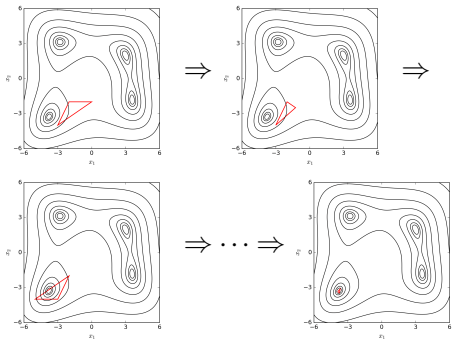


Figure 5: Nelder-Mead optimization. Here the input space is the two-dimensional square, and the (unknown) score function is depicted by contour lines. Figures are from Wikipedia

120. From the assumption about $\mathscr{M}$'s behavior, we expect u to be large in a falsifying input signal. Note that this is a simplified version of one of our experiments in §6.

Optimization-Based Falsification Fig. 2 illustrates how a conventional optimization-based falsification procedure works. In the i-th sampling one tries an input signal $\mathbf{u}^{(i)}$. Following the falsification literature we focus on piecewise constant signals. Thus a signal $\mathbf{u}^{(i)}$ is represented by a sequence $(u_1^{(i)},\dots,u_K^{(i)})$ of real numbers. See the top left of Fig. 2. The corresponding output signal $\mathbf{v}^{(i)} = \mathscr{M}(\mathbf{u}^{(i)})$ is shown below it.

Since $\mathbf{v}^{(i)}$ does not reach the threshold 120, we move on to the $(i+1)$-th sampling and try a new input signal $\mathbf{u}^{(i+1)} = (u_1^{(i+1)},\dots,u_K^{(i+1)})$. The choice of $\mathbf{u}^{(i+1)}$ is made by an optimization algorithm. Specifically, the optimization algorithm observes the results of the previously sampled input signals $\mathbf{u}^{(1)},\dots,\mathbf{u}^{(i)}$—especially the robustness value $[\![\mathscr{M}(\mathbf{u}^{(i)}),\varphi]\!]$ that each input $\mathbf{u}^{(i)}$ achieves. In the current setting where $\varphi \equiv \big(\square(v \le 120)\big)$, the robustness value is simply the difference between 120 and the peak vehicle speed. The optimization algorithm tries to derive some general tendency, which it then uses to increase the probability that the next input signal $\mathbf{u}^{(i+1)}$ will make the robustness smaller (i.e. the peak vehicle speed higher).

Hill climbing is a prototype of such optimization algorithms. Its use in falsification is illustrated in Fig. 4, where $\mathbf{u} = (u_1,\dots,u_K)$ is depicted as one-dimensional for clarity. The actual curve for the robustness value $[\![\mathscr{M}(\mathbf{u}),\varphi]\!]$ (gray and dashed) is unknown. Still the previous observations under input $\mathbf{u}^{(1)},\dots,\mathbf{u}^{(i)}$ suggest that to the right is the climbing down direction. The next candidate $\mathbf{u}^{(i+1)}$ is picked accordingly, towards negative robustness. Another well-known optimization algorithm is the *Nelder-Mead algorithm*. See Fig. 5, where the input space is two-dimensional and the (unknown) robustness function is depicted by contour lines.

We see in the right of Fig. 2 that the new input signal $\mathbf{u}^{(i+1)} = (u_1^{(i+1)},\dots,u_K^{(i+1)})$ leads to a corresponding output signal $\mathbf{v}^{(i+1)}$ that reduces the robustness value by achieving a higher peak speed. We continue this way, $\mathbf{u}^{(i+2)},\mathbf{u}^{(i+3)},\dots$, hoping to eventually reach a falsifying input signal.

Absence of the Time-Causal Information A closer look at Fig. 2 reveals room for improvement. In Fig. 2, the new input signal $\mathbf{u}^{(i+1)}$ indeed achieves a smaller overall robustness $[\![\mathscr{M}(\mathbf{u}),\varphi]\!]$ than $\mathbf{u}^{(i)}$. However, its initial segment $u_1^{(i+1)}$ is smaller than $u_1^{(i)}$; consequently the vehicle speed $\mathbf{v}^{(i+1)}$ is smaller than $\mathbf{v}^{(i)}$ in the first few seconds. Keeping the initial segment $u_1^{(i)}$ would have achieved an even greater peak speed.

The problem here is that the *time-causal* structure inherent in the problem is not explicitly communicated to the optimization algorithm. The relevant structure is more specifically *time monotonicity*: an input prefix that achieves smaller robustness (i.e. a greater peak speed) is more likely to extend to a full falsifying input signal. Although it is possible that a stochastic optimization algorithm somehow "learns" time monotonicity, it is not guaranteed, because the structure of input spaces (the horizontal axis in Fig. 4 and the squares in Fig. 5) does not explicitly reflect time-causal structures.

While the time monotonicity is not shared by all instances of the falsification problems, we find many realistic instances that approximately satisfy the property. We discuss time monotonicity in §5, as well as in the context of our experiments in §6.

Falsification with Time Staging Our proposal of *time staging* consists of incrementally synthesizing a candidate input signal. We illustrate this in Fig. 3. In the first stage (left), we run a falsification algorithm and try to find an initial input segment that achieves low robustness (i.e. high peak speed). This first stage

comprises running n_1 samplings, as illustrated in Fig. 2. This process will gradually improve candidates for the initial input segment, in the way the arrows $\uparrow$ on the left in Fig. 3 designate. Let us assume that the last candidate $u_1^{(n_1)}$ is the (tentative) best, achieving the smallest robustness.

In the second stage (on the right in Fig. 3) we continue $u_1^{(n_1)}$ and synthesize the second input segment. This is again by running a falsification algorithm, as depicted. Note that, in each stage (a box in Fig. 3), the whole iterated process in Fig. 2 is conducted. In this way we continue to the K-th stage, always starting with the input segment that performed the best in the previous stage, thus exploiting the time-causal structure.

While time staging is not difficult to implement, there is a challenge in using it effectively. An immediate question is whether choosing the single best input segment in each stage is the optimal approach. Our current strategy favors exploitation over exploration: it might miss a falsifying signal whose robustness must decrease slowly in the earlier segments and only quickly in the latter segments. Indeed we are working on an evolutionary variant of the above time-staged algorithm, where multiple segments are passed over from one stage to another, in order to maintain diversity and conduct exploration. That said, even under the current simple strategy of picking the best one, we observe significant performance enhancement in some falsification problems. See §6.

We can summarize this trade-off in terms of the size of search spaces. Let U be the set of candidates for input segments, and K be the number of stages. Then the size of the set of whole input signals is $|U|^K$, choosing one input segment for each stage. In our staged algorithm, in contrast, the search space for each stage is U and overall our search space is $K \cdot |U|$. This reduction comes with the risk of missing some falsifying input signals. The experimental results in §6 suggest this risk is worth taking. Moreover, in §5 we present some theoretical conditions for the absence of such risk. They help users decide in practical applications when time staging will be effective.

3 Optimization-Based Falsification

From this section on we turn to the formal description and analysis of our algorithm. This section presents a review of existing works on optimization-based falsification.

System Models Let us formalize our system models.

Definition 3.1 (time-bounded signal)**.** Let $T \in \mathbb{R}_{>0}$ be a positive real. A *(time-bounded) m-dimensional signal* with a time horizon T is a function $\mathbf{w}\colon [0,T] \to \mathbb{R}^m$.

Let $\mathbf{w}\colon [0,T] \to \mathbb{R}^m$ and $\mathbf{w}'\colon [0,T'] \to \mathbb{R}^m$ be (time-bounded) signals. Their *concatenation* $\mathbf{w} \cdot \mathbf{w}'\colon [0,T+T'] \to \mathbb{R}^m$ is defined by $(\mathbf{w} \cdot \mathbf{w}')(t) := \mathbf{w}(t)$ if $t \in [0,T]$, and $\mathbf{w}'(t-T)$ if $t \in (T,T+T']$.

Let $T_1, T_2 \in (0,T]$ such that $T_1 < T_2$. The *restriction* $\mathbf{w}|_{[T_1,T_2]}\colon [0,T_2-T_1] \to \mathbb{R}^m$ of $\mathbf{w}\colon [0,T] \to \mathbb{R}^m$ to the interval $[T_1,T_2]$ is defined by $(\mathbf{w}|_{[T_1,T_2]})(t) := \mathbf{w}(T_1+t)$.

Definition 3.2 (system model $\mathscr{M}$)**.** A *system model*, with M-dimensional input, is a function $\mathscr{M}$ that takes an input signal $\mathbf{u}\colon [0,T] \to \mathbb{R}^M$ and returns $\mathscr{M}(\mathbf{u})\colon [0,T] \to \mathbb{R}^N$. Here the common time horizon $T \in \mathbb{R}_{>0}$ is arbitrary.

Furthermore, we impose the following *causality* condition on $\mathscr{M}$. For any time-bounded signals $\mathbf{u}\colon [0,T] \to \mathbb{R}^M$ and $\mathbf{u}'$, we require that $\mathscr{M}(\mathbf{u} \cdot \mathbf{u}')\big|_{[0,T]} = \mathscr{M}(\mathbf{u})$.

Note that $\mathscr{M}(\mathbf{u} \cdot \mathbf{u}') = \mathscr{M}(\mathbf{u}) \cdot \mathscr{M}(\mathbf{u}')$ does not hold in general: feeding $\mathbf{u}$ can change the internal state of $\mathscr{M}$. This motivates the following definition.

Definition 3.3 (continuation $\mathscr{M}_{\mathbf{u}}$)**.** Let $\mathscr{M}$ be a system model and $\mathbf{u}\colon [0,T] \to \mathbb{R}^M$ be a signal. The *continuation* of $\mathscr{M}$ after $\mathbf{u}$, denoted by $\mathscr{M}_{\mathbf{u}}$, is defined as follows. For an input signal $\mathbf{u}'\colon [0,T'] \to \mathbb{R}^M$: $\mathscr{M}_{\mathbf{u}}(\mathbf{u}')(t) := \mathscr{M}(\mathbf{u}\cdot\mathbf{u}')(T+t)$.

Signal Temporal Logic and Robust Semantics We review *signal temporal logic* (**STL**) [21] and its *robust semantics* [10, 14]. **Var** is the set of variables, and let $N := |\mathbf{Var}|$. Variables stand for physical quantities, control modes, etc. $\equiv$ denotes syntactic equality.

Definition 3.4 (syntax)**.** In **STL**, *atomic propositions* and *formulas* are defined as follows, respectively: $\alpha ::\equiv f(x_1,\dots,x_n) > 0$, and $\varphi ::\equiv \alpha \mid \bot \mid \neg\varphi \mid \varphi\wedge\varphi \mid \varphi\,\mathscr{U}_I\,\varphi$. Here f is an n-ary function $f:\mathbb{R}^n\to\mathbb{R}$, $x_1,\dots,x_n \in \mathbf{Var}$, and I is a closed non-singular interval in $\mathbb{R}_{\geq 0}$, i.e. $I=[a,b]$ or $[a,\infty)$ where $a,b\in\mathbb{R}$ and $a<b$.

We omit subscripts I for temporal operators if $I=[0,\infty)$. Other common connectives like $\vee,\to,\top$, $\Box_I$ (always) and $\Diamond_I$ (eventually), are introduced as abbreviations: $\Diamond_I\varphi \equiv \top\,\mathscr{U}_I\,\varphi$ and $\Box_I\varphi \equiv \neg\Diamond_I\neg\varphi$. Atomic formulas like $f(\vec{x}) \leq c$, where $c\in\mathbb{R}$ is a constant, are also accommodated by using negation and the function $f'(\vec{x}) := f(\vec{x}) - c$.

Definition 3.5 (robust semantics [9, 10])**.** For an unbounded n-dimensional signal $\mathbf{w}\colon \mathbb{R}_{\geq 0}\to\mathbb{R}^n$ and $t\in\mathbb{R}_{\geq 0}$, $\mathbf{w}^t$ denotes the *t-shift* of $\mathbf{w}$, that is, $\mathbf{w}^t(t') := \mathbf{w}(t+t')$.

Let $\mathbf{w}\colon \mathbb{R}_{\geq 0}\to\mathbb{R}^N$ be a signal (recall $N=|\mathbf{Var}|$), and φ be an **STL** formula. We define the *robustness* $[\![\mathbf{w},\varphi]\!] \in \mathbb{R}\cup\{\infty,-\infty\}$ as follows, by induction. Here $\sqcap$ and $\sqcup$ denote infimums and supremums of real numbers, respectively.

$$
\begin{array}{ll}
[\![\mathbf{w}, f(x_1,\cdots,x_n)>0]\!] := f\big(\mathbf{w}(0)(x_1),\cdots,\mathbf{w}(0)(x_n)\big) & [\![\mathbf{w},\bot]\!] := -\infty \\
[\![\mathbf{w},\neg\varphi]\!] := -[\![\mathbf{w},\varphi]\!] & [\![\mathbf{w},\varphi_1\wedge\varphi_2]\!] := [\![\mathbf{w},\varphi_1]\!]\sqcap[\![\mathbf{w},\varphi_2]\!] \\
[\![\mathbf{w},\varphi_1\,\mathscr{U}_I\,\varphi_2]\!] := \bigsqcup_{t\in I}\big([\![\mathbf{w}^t,\varphi_2]\!]\sqcap\bigsqcap_{t'\in[0,t)}[\![\mathbf{w}^{t'},\varphi_1]\!]\big) &
\end{array}
$$

Here are some intuitions and consequences. The robustness $[\![\mathbf{w}, f(\vec{x})>c]\!]$ stands for the vertical margin $f(\vec{x})-c$ for the signal $\mathbf{w}$ at time 0. A negative robustness value indicates how far the formula is from being true. The robustness for the eventually modality is computed by $[\![\mathbf{w},\Diamond_{[a,b]}(x>0)]\!] = \bigsqcup_{t\in[a,b]}\mathbf{w}(t)(x)$.

The original semantics of **STL** is Boolean, given by a binary relation $\models$ between signals and formulas. The robust semantics refines the Boolean one, in the sense that: $[\![\mathbf{w},\varphi]\!]>0$ implies $\mathbf{w}\models\varphi$, and $[\![\mathbf{w},\varphi]\!]<0$ implies $\mathbf{w}\not\models\varphi$. Optimization-based falsification via robust semantics [14] hinges on this refinement. See [10].

Although the definitions so far are for unbounded signals only, we note that the robust semantics $[\![\mathbf{w},\varphi]\!]$, as well as the Boolean satisfaction $\mathbf{w}\models\varphi$, allows straightforward adaptation to time-bounded signals (Def. 3.1). See Appendix A.

Falsification Solvers In the next definition, a prototype of a score function ρ is given by the robustness ρ_φ of a given **STL** specification φ. The generality of allowing other ρ is needed later in §4.

$$\rho_\varphi(\mathbf{v}) := [\![\mathbf{v},\varphi]\!] \tag{1}$$

Definition 3.6 (falsification solver)**.** A *falsification solver* is a stochastic algorithm Falsify that takes, as input: 1) a system model $\mathscr{M}$ (Def. 3.2) with M-dimensional input; 2) a score function ρ that takes an output signal $\mathbf{v}$ of $\mathscr{M}$ and returns a score $\rho(\mathbf{v})\in\mathbb{R}\cup\{-\infty,\infty\}$; and 3) a time horizon $T\in\mathbb{R}_{>0}$. The algorithm Falsify returns an M-dimensional signal $\mathbf{u}\colon [0,T]\to\mathbb{R}^M$.

Algorithm 1 Internal Structure of a Falsification Solver $\mathsf{Falsify}(\mathscr{M},\rho,T)$

Require: a system model $\mathscr{M}$, a score function ρ, and $T \in \mathbb{R}_{>0}$

1: $\mathsf{U} \leftarrow ()$ ▷ the list U collects all the candidates $\mathbf{u}\colon [0,T] \to \mathbb{R}^M$
2: **while** $\neg\mathsf{InitialSamplingDone}(T,\mathsf{U})$ **do**
3: $\quad \mathbf{u} \leftarrow \mathsf{InitialSampling}(T)$ ▷ $\mathbf{u}\colon [0,T] \to \mathbb{R}^M$ is sampled following some recipe
4: $\quad \mathsf{U} \leftarrow \mathsf{cons}(\mathsf{U},\mathbf{u})$
5: **while** $\neg\mathsf{OptimizationSamplingDone}(\mathscr{M},\rho,T,\mathsf{U})$ **do**
6: $\quad \mathbf{u} \leftarrow \mathsf{OptimizationSampling}(\mathscr{M},\rho,T,\mathsf{U})$
7: $\quad\quad$ ▷ $\mathbf{u}$ is sampled, so that $\rho(\mathscr{M}(\mathbf{u}))$ becomes small, based on previous samples in U
8: $\quad \mathsf{U} \leftarrow \mathsf{cons}(\mathsf{U},\mathbf{u})$
9: $\mathbf{u} \leftarrow \arg\min_{\mathbf{u}\in\mathsf{U}} \rho(\mathscr{M}(\mathbf{u}))$
10: **return u** ▷ a trial is successful if $\rho(\mathscr{M}(\mathbf{u})) < 0$

Each invocation $\mathsf{Falsify}(\mathscr{M},\rho,T)$ of the solver is called a *falsification trial*. It is *successful* if the returned signal $\mathbf{u}$ satisfies $\rho(\mathscr{M}(\mathbf{u})) < 0$. Note that the returned signal $\mathbf{u}$ can differ in every trial, since Falsify is a stochastic algorithm.

We further assume the internal structure of the solver Falsify follows the scheme in Algorithm 1. It consists of two phases. The first *initial sampling* phase collects some candidates for $\mathbf{u}\colon [0,T] \to \mathbb{R}^M$ regardless of the system model $\mathscr{M}$ or the score function ρ. In the second *optimization sampling* phase, a stochastic optimization algorithm is employed to sample a candidate $\mathbf{u}$ that is likely to make the score $\rho(\mathscr{M}(\mathbf{u}))$ small.

Implementation of Falsification Solvers Both Breach [8] and S-Taliro [5] take industry-standard Simulink models as system models. For input signal candidates the tools focus on piecewise constant signals; they are represented by sequences $(u_1,\dots,u_K)$ of real numbers, much like in §2. Here K is the number of *control points*; in our staged algorithm we use the same K for the number of stages.

The tools offer multiple stochastic optimization algorithms for the optimization sampling phase, including *CMA-ES* [6], *global Nelder-Mead* and *simulated annealing*. The initial sampling phase is mostly by random sampling. Additionally, in Breach with global Nelder-Mead, so-called *corner samples* are added to the list U. The number of corner samples grows exponentially as K grows, i.e. as we have more control points.

4 Time Staging in Optimization-Based Falsification

Definition 4.1 (time-staged deployment of falsification solver)**.** Let $\mathscr{M}$ be a system model, φ be an **STL** formula, and $T \in \mathbb{R}_{>0}$ be a time horizon. Let $K \in \mathbb{N}$ be a parameter; it is the number of time stages. The *time-staged deployment* of a falsification solver Falsify is the procedure in Algorithm 2. On the line 3, the model $\mathscr{M}_{\mathbf{u}}$ is the continuation of $\mathscr{M}$ after $\mathbf{u}$ (Def. 3.3); the score function $\partial_{\mathbf{v}}\rho_\varphi$ is defined by

$$(\partial_{\mathbf{v}}\rho_\varphi)(\mathbf{v}') \;:=\; \rho_\varphi(\mathbf{v}\cdot\mathbf{v}') \;\overset{(1)}{=}\; [\![\mathbf{v}\cdot\mathbf{v}',\varphi]\!] \;. \tag{2}$$

The whole procedure is stochastic (since Falsify is); an invocation is called a *time-staged falsification trial*. It is *successful* if the returned signal $\mathbf{u}$ satisfies $[\![\mathscr{M}(\mathbf{u}),\varphi]\!] < 0$.

A falsification trial (i.e. an invocation of Algorithm 1) is an iterative process: the more we sample, the more likely we obtain a falsifying input signal. Since we run multiple falsification trials in Algorithm 2

Algorithm 2 Time-Staged Deployment of a Falsification Solver

Require: a falsification solver Falsify, a system model $\mathscr{M}$, an **STL** formula φ, $T \in \mathbb{R}_{>0}$ and $K \in \mathbb{N}$

1: $\mathbf{u} \leftarrow ()$ ▷ the input prefix obtained so far. We start with the empty signal $()$
2: **for** $j \in \{1, \dots, K\}$ **do**
3: $\quad \mathbf{u}' \leftarrow \mathsf{Falsify}(\mathscr{M}_{\mathbf{u}}, \partial_{\mathscr{M}(\mathbf{u})}\rho_{\varphi}, \frac{T}{K})$ ▷ synthesizing the j-th input segment
4: $\quad \mathbf{u} \leftarrow \mathbf{u} \cdot \mathbf{u}'$ ▷ concatenate $\mathbf{u}'$, after which the length of $\mathbf{u}$ is $\frac{jT}{K}$
5: **return u** ▷ a time-staged falsification trial is successful if $[\![\mathscr{M}(\mathbf{u}), \varphi]\!] < 0$

(one trial for each of the K stages), an important question is how we distribute available time to different stages.

A simple strategy is to fix the number of samples in each phase of Algorithm 1. Then the predicates $\mathsf{InitialSamplingDone}(T, \mathsf{U})$ and $\mathsf{OptimizationSamplingDone}(\mathscr{M}, \rho, T, \mathsf{U})$ are given by $|\mathsf{U}| > N^{\mathrm{init}}_{\mathrm{max}}$ and $|\mathsf{U}| > N^{\mathrm{opt}}_{\mathrm{max}}$, where $N^{\mathrm{init}}_{\mathrm{max}}, N^{\mathrm{opt}}_{\mathrm{max}}$ are constants.

An *adaptive* strategy, that we also implemented for the optimization sampling phase, is to continue sampling until we stop seeing progress. Here we fix a parameter $N^{\mathrm{stuck}}_{\mathrm{max}}$, and we stop after $N^{\mathrm{stuck}}_{\mathrm{max}}$ consecutive samplings without reducing robustness. A similar strategy of adaptively choosing the number of samples can be introduced for random sampling in the initial sampling phase (the lines 2–4 of Algorithm 1).

4.1 Towards Efficient Implementation

A key to speedup of Algorithm 2 is in the line 3; more specifically, how we handle the previous input prefix $\mathbf{u}$. Here we discuss two directions, one on the model $\mathscr{M}_{\mathbf{u}}$ and the other on the score function $\partial_{\mathbf{v}}\rho_{\varphi}$. (We note that the suggested enhancements are not currently used in our implementation, because of performance reasons. See below.)

Continuation of Models Optimization-based falsification has a very wide application domain. Since it only requires a *black-box* model $\mathscr{M}$, the concrete form of $\mathscr{M}$ can vary from a program to a Simulink model and even a system with hardware components (*HILS*). These models can be very big, and usually the bottleneck in falsification lies in *simulation*, that is, to compute $\mathscr{M}(\mathbf{u})$ given an input signal $\mathbf{u}$.

In the line 3 of Algorithm 2, therefore, using the definition $\mathscr{M}(\mathbf{u} \cdot \mathbf{u}')(T + t)$ in Def. 3.3 is in principle not a good strategy: it requires simulation of $\mathscr{M}$ for the whole prefix $\mathbf{u} \cdot \mathbf{u}'$, which can be avoided if we can directly simulate the continuation $\mathscr{M}_{\mathbf{u}}$. In Simulink this is possible by saving the snapshot of the model after a simulation, via the `SaveFinalState` model configuration parameter. In our implementation we do not do so, though, because the overhead of saving and loading snapshots is currently greater than the cost of simulating. This balance can become different, if we figure out a less expensive way to use snapshots, or if we study more complex models.

Derivative of Formulas The situation is similar with the score function $\partial_{\mathscr{M}(\mathbf{u})}\rho_{\varphi}$ in the line 3 of Algorithm 2. Using the presentation $\rho_{\varphi}(\mathscr{M}(\mathbf{u}) \cdot \mathbf{v}')$ in (2) requires scanning the same prefix $\mathscr{M}(\mathbf{u})$ repeatedly. Desired here is a *syntactic* presentation of $\partial_{\mathscr{M}(\mathbf{u})}\rho_{\varphi}$, that will be given as an **STL** formula $\partial_{\mathscr{M}(\mathbf{u})}\varphi$ such that $\partial_{\mathscr{M}(\mathbf{u})}\rho_{\varphi} = \rho_{(\partial_{\mathscr{M}(\mathbf{u})}\varphi)}$. This would allow one to utilize available algorithms for computing robustness values $[\![\mathbf{v}, \partial_{\mathscr{M}(\mathbf{u})}\varphi]\!]$.

Definition 4.2 (derivative of flat **STL** formulas)**.** Let $T \in \mathbb{R}_{>0}$, and $\mathbf{v}\colon [0, T] \to \mathbb{R}^N$ be a signal. Given an **STL** formula φ that is *flat* in the sense that it does not have nested temporal operators, the *derivative*

$\partial_{\mathbf{v}}\varphi$ by $\mathbf{v}$ is defined inductively as follows.

$$\begin{array}{ll} \partial_{\mathbf{v}}\big(f(\vec{x})>0\big) :\equiv \mathsf{c}_{[\![\mathbf{v},f(\vec{x})>0]\!]} & \partial_{\mathbf{v}}\bot :\equiv \bot \\ \partial_{\mathbf{v}}(\neg\varphi) :\equiv \neg\partial_{\mathbf{v}}\varphi & \partial_{\mathbf{v}}(\varphi_1\wedge\varphi_2) :\equiv (\partial_{\mathbf{v}}\varphi_1)\wedge(\partial_{\mathbf{v}}\varphi_2) \\ \partial_{\mathbf{v}}(\varphi_1\,\mathscr{U}_I\,\varphi_2) :\equiv \mathsf{c}_{[\![\mathbf{v},\varphi_1\mathscr{U}_I\varphi_2]\!]}\vee\big((\mathsf{c}_{[\![\mathbf{v},\Box\varphi_1]\!]}\wedge\varphi_1)\,\mathscr{U}_{I-T}\,\varphi_2\big) & \end{array}$$

Here the interval $I-T$ is obtained by shifting endpoints, such as $[a,b]-T=[a-T,b-T]$. For each $r\in\mathbb{R}$, the notation c_r abbreviates the atomic formula $r>0$, where r is thought of as a constant function. We use the fact that $[\![\mathbf{w},\mathsf{c}_r]\!]=r$.

Until formulas $\varphi_1\,\mathscr{U}_I\,\varphi_2$ are split into the evaluation on the signal prefix $\mathbf{v}$ (first disjunct), and a continuation (second disjunct). The constant $\mathsf{c}_{[\![\mathbf{v},\Box\varphi_1]\!]}$ injects the robustness of φ_1 seen so far into the residual formula (recall that both of $\Box$ and $\wedge$ take the infimum). It follows that $\partial_{\mathbf{v}}\big(\Box_I\,\varphi\big)\equiv\mathsf{c}_{[\![\mathbf{v},\Box\varphi]\!]}\wedge\Box_{I-T}\,\varphi$ and $\partial_{\mathbf{v}}\big(\Diamond\,\varphi\big)\equiv\mathsf{c}_{[\![\mathbf{v},\Diamond\varphi]\!]}\vee\Diamond_{I-T}\,\varphi$.

Proposition 4.3. *Let $T\in\mathbb{R}_{>0}$, $\mathbf{v}\colon[0,T]\to\mathbb{R}^N$ be a signal, and φ be a flat **STL** formula. We have, for any $\mathbf{v}'\colon[0,T']\to\mathbb{R}^N$, $[\![\mathbf{v}',\partial_{\mathbf{v}}\varphi]\!]=[\![\mathbf{v}\cdot\mathbf{v}',\varphi]\!]$.*

A proof is in Appendix B. Use of derivatives for timed specifications is also found e.g. in [22]. The settings are different, though: Boolean semantics in [22] while our semantics is quantitative. Our restriction to flat formulas comes mainly from this difference, and lifting the flatness restriction seems hard.

5 Sufficient Conditions for Time Staging

We present some theoretical analyses of the performance of time staging that indicate to which class of systems the time-staged approach can apply. We give some *sufficient* conditions under which the approach is guaranteed to work. However, it should be noted that it is *not* necessary that a concrete system satisfies these conditions strictly as these are rather restrictive. Nevertheless, we believe that users with expert domain knowledge can judge whether their models satisfy these conditions approximately. This way our results provide those users with "rules of thumb."

As we discussed in the last paragraph of §2, the potential performance advantage by time staging comes from the reduction of search spaces from $|U|^K$ to $K\cdot|U|$. Here U is the set of potential input segments for each stage, and K is the number of stages. This advantage comes at the risk of missing out some error input signals. The following basic condition (3), that we call *incremental falsification*, ensures that there is no such risk. More precisely, we can decompose the "best" input signal $\mathbf{u}$ into a first stage $\mathbf{u}_1$ and its remainder $\mathbf{u}_2$ such that the entire falsification problem (left hand side) is solved by greedy optimization of the initial segment (inner $\arg\min_{\mathbf{u}_1}$), and subsequent optimization of the continuation (outer $\min_{\mathbf{u}_2}$). For all choices of T_1,T_2 with ranges $\mathbf{u}\colon[0,T_1+T_2]\to\mathbb{R}^m$ and $\mathbf{u}_i\colon[0,T_i]\to\mathbb{R}^m$:

$$\min_{\mathbf{u}}[\![\mathscr{M}(\mathbf{u}),\varphi]\!]\;=\;\min_{\mathbf{u}_2}\Big[\!\Big[\mathscr{M}\Big(\big(\arg\min_{\mathbf{u}_1}[\![\mathscr{M}(\mathbf{u}_1),\varphi]\!]\big)\cdot\mathbf{u}_2\Big),\varphi\Big]\!\Big] \tag{3}$$

Algorithm 2 repeatedly unfolds (3) by picking constant $T_1=T/K$ where T is the time horizon and K is the number of stages. The rest of this section is devoted to the search for concrete sufficient conditions for (3).

Monotone Systems and Ceiling Specifications We formalize the time monotonicity property in §2. That it implies incremental falsification (3) can be easily proved.

Definition 5.1 (time-monotone falsification problem)**.** A system model $\mathscr{M}$ and an **STL** formula φ are said to *constitute a time-monotone falsification problem* if, for any input signals $\mathbf{u}_1, \mathbf{u}_1' \colon [0,T_1] \to \mathbb{R}^m$ and $\mathbf{u}_2 \colon [0,T_2] \to \mathbb{R}^m$, $[\![\mathscr{M}(\mathbf{u}_1), \varphi]\!] \leq [\![\mathscr{M}(\mathbf{u}_1'), \varphi]\!]$ implies $[\![\mathscr{M}(\mathbf{u}_1 \cdot \mathbf{u}_2), \varphi]\!] \leq [\![\mathscr{M}(\mathbf{u}_1' \cdot \mathbf{u}_2), \varphi]\!]$.

We investigate yet more concrete conditions that ensures time monotonicity. The following condition on system models is assumed in the example of §2.

Definition 5.2 (monotone system, ceiling specification)**.** Let x be a variable (for output). A system model $\mathscr{M}$ is said to be *monotone* in x if, for each $\mathbf{u}_1, \mathbf{u}_1' \colon [0,T_1] \to \mathbb{R}^M$ and $\mathbf{u}_2 \colon [0,T_2] \to \mathbb{R}^M$, $\mathscr{M}(\mathbf{u}_1)(T_1)(x) \leq \mathscr{M}(\mathbf{u}_1')(T_1)(x)$ implies $\mathscr{M}(\mathbf{u}_1 \cdot \mathbf{u}_2)(T_1+T_2)(x) \leq \mathscr{M}(\mathbf{u}_1' \cdot \mathbf{u}_2)(T_1+T_2)(x)$.

An **STL** formula of the form $\Box(x < c)$, where x is a variable and $c \in \mathbb{R}$ is a constant, is called a *ceiling formula*.

One can speculate that a monotone system and a ceiling specification $\Box(x < c)$, like those in §2, constitute a time-monotone falsification problem. The speculation is not true, unfortunately; a counterexample is easily constructed using a model $\mathscr{M}$ whose output signal is not increasing. We can instead show the following weaker property.

Definition 5.3 (truncated time monotonicity)**.** A system model $\mathscr{M}$ and an **STL** formula φ *constitute a truncated time-monotone falsification problem* if, for any input $\mathbf{u}_1, \mathbf{u}_1' \colon [0,T_1] \to \mathbb{R}^m$ and $\mathbf{u}_2 \colon [0,T_2] \to \mathbb{R}^m$, $[\![\mathscr{M}(\mathbf{u}_1), \varphi]\!] \leq [\![\mathscr{M}(\mathbf{u}_1'), \varphi]\!]$ implies existence of $T \in (0,T_1]$ such that $[\![\mathscr{M}((\mathbf{u}_1|_{[0,T]}) \cdot \mathbf{u}_2), \varphi]\!] \leq [\![\mathscr{M}((\mathbf{u}_1'|_{[0,T]}) \cdot \mathbf{u}_2), \varphi]\!]$.

Proposition 5.4. *Let $\mathscr{M}$ be a model monotone in x, and $\varphi \equiv \big(\Box(x < c)\big)$. Then $\mathscr{M}$ and φ constitute a truncated time-monotone falsification problem.*

The proof, in Appendix C.1, constructs a concrete choice of T in Def. 5.3. Specifically it is the instant $T \in [0,T_1]$ in which the robustness $[\![\mathscr{M}(\mathbf{u}_1|_{[0,T]}), \varphi]\!]$ is minimum. In the scenario of §2 this is the instant that the vehicle speed is in its peak. Note that truncated time monotonicity does not guarantee incremental falsification as per (3), but it implies that the current rigid time staging at $0, \frac{T}{K}, \frac{2T}{K}, \dots, \frac{(K-1)T}{K}$ is not optimal. These theoretical considerations suggest potential improvement of the staged procedure in Def. 4.1 with adaptive choice of stages, which is left for future work.

Stateless Systems and Reachability Specifications Here is another sufficient condition.

Definition 5.5 (stateless system, reachability formula)**.** A system model $\mathscr{M}$ is said to be *stateless* if, for any input signals $\mathbf{u}_1, \mathbf{u}_1' \colon [0,T_1] \to \mathbb{R}^m$ and $\mathbf{u}_2 \colon [0,T_2] \to \mathbb{R}^m$, we have $\mathscr{M}(\mathbf{u}_1 \cdot \mathbf{u}_2)|_{(T_1,T_2]} = \mathscr{M}(\mathbf{u}_1' \cdot \mathbf{u}_2)|_{(T_1,T_2]}$.

An **STL** formula $\Diamond\psi$, where ψ is modality-free, is called a *reachability formula*.

Note that being stateless is a sufficient but not necessary condition for $\mathscr{M}(\mathbf{u}_1 \cdot \mathbf{u}_2) = \mathscr{M}(\mathbf{u}_1) \cdot \mathscr{M}(\mathbf{u}_2)$. Statelessness requires insensitivity to previous input prefixes, but a stateless system can still be sensitive to time.

Proposition 5.6. *Let $\mathscr{M}$ be a stateless system and φ be a reachability specification $\varphi \equiv (\Diamond\psi)$. Then $\mathscr{M}$ and φ satisfy the incremental falsification property (3).*

A proof is easy. A typical situation in which we would appeal to Prop. 5.6 is when: the specification is $\Diamond(x < c_1 \vee x > c_2)$ (which can be hard to falsify if $c_1 < c_2$ are close); and the system is already in its stable state (so that its behavior does not depend much on what happened during the transient phase). Our experiments in §6 demonstrate that time staging can drastically improve performance in such settings.

Table 1: Experimental results. Each column shows how many falsification trials succeeded (out of 20), and the average runtime. S1: $\square_{[0,30]}\ (v < 120)$. S2: $\square_{[0,30]}\ (g = 3 \rightarrow v \geq 30)$. S3: $\diamondsuit_{[10,30]}\ (v \leq v_{\min} \vee v \geq v_{\max})$, where: $v_{\min} = 50$, $v_{\max} = 60$ (easy); $v_{\min} = 53$, $v_{\max} = 57$ (hard). S4: $\square_{[0,10]}(v < v) \vee \diamondsuit_{[0,30]}(\omega > \omega_{\max})$, where: $v_{\min} = 80, \omega_{\max} = 4500$ (easy); $v_{\min} = 50, \omega_{\max} = 2700$ (mid); $v_{\min} = 50, \omega_{\max} = 2520$ (hard). The specification S for the Abstract Fuel Control model is $\neg(\diamondsuit_{[t_1,t_2]}\square_{[0,t']}(AF - AF_{\text{ref}} > \delta * 14.7))$, where: $t_1 = 0$, $t_2 = 6$, $t' = 3$, $\delta = 0.07$ (init); $t_1 = 6$, $t_2 = 26$, $t' = 4$, $\delta = 0.01$ (stable). Starred numbers 0* or 20* indicate that GNM is deterministic so all trials yield the same result.

model	Automatic Transmission														Abst. Fuel Ctrl.			
spec.	S1		S2		S3 easy		S3 hard		S4 easy		S4 mid		S4 hard		S init		S stable	
algorithm	time	#/20	time	#/20	time	#/20	time	#/20	time	#/20	time	#/20	time	#/20	time	#/20	time	#/20
CMA-ES	27s	**20**	**5s**	**20**	39s	14	57s	0	32s	16	37s	9	59s	0	49s	0	82s	1
+TS	52s	15	15s	16	**9s**	19	23s	**11**	15s	14	**14s**	**14**	24s	3	30s	0	42s	1
+A-TS	41s	18	15s	17	**9s**	16	21s	**10**	26s	14	22s	14	**20s**	**5**	26s	0	41s	0
SA	50s	5	43s	7	37s	9	55s	0	35s	6	36s	9	47s	**5**	51s	0	76s	2
+TS	37s	**20**	33s	**16**	**11s**	**19**	33s	**8**	21s	14	25s	13	51s	0	47s	**1**	54s	**7**
+A-TS	34s	**20**	18s	**17**	**9s**	**18**	26s	**4**	**16s**	**18**	21s	11	30s	2	34s	0	42s	**5**
GNM	**6s**	**20***	61s	0*	56s	0*	55s	0*	43s	0*	46s	0*	53s	0*	50s	0*	86s	0*
+TS	42s	20*	15s	**20***	**13s**	**20***	25s	**20***	**11s**	**20***	45s	0*	52s	0*	30s	**20***	**20s**	**20***
+A-TS	20s	20*	16s	**20***	**10s**	**20***	26s	**20***	13s	20*	45s	0*	43s	0*	37s	0*	**19s**	**20***

Corner Samples for Global Nelder-Mead The reduction of search spaces from $|U|^K$ to $K \cdot |U|$ has its analogue in the number of corner samples in Breach with global Nelder-Mead (lines 2–4 of Algorithm 1, see the last paragraph of §3). Originally the number of corner samples is $2^{K \cdot M}$, where K is the number of control points and M is the number of input values. By introducing K time stages, the total number of corner samples is reduced to $K \cdot 2^M$.

6 Experiments

We compare the success rate and time consumption of the proposed method. The benchmarks here use automotive Simulink models that are commonly used in the falsification literature. Specifications are chosen taking the deliberations of §5 into account, namely with ceiling specifications (Def. 5.2, including the example of §2), a reachability specification (Def. 5.5) and a combination thereof.

The base line is Algorithm 1 implemented by Breach [8]. The methods proposed in §4 are implemented on top of Breach: the time-staged Algorithm 2 (TS), and the adaptive strategy (A-TS, the one described after Def. 4.1). All three algorithms (plain, TS, A-TS) are combined with different optimization solvers: CMA-ES, simulated annealing (SA), global Nelder-Mead (GNM), obtaining a total of nine configurations.

The results in Table 1 indicate that both success rate and runtime performance are significantly improved by time staging, often finding counterexamples when non-staged Breach fails (e.g. columns *S3 hard* and *S init*). Furthermore, we see that while the adaptive algorithm (A-TS) does not necessarily lead to a higher success rate in comparison to the time-staged one (TS), it gives yet another runtime performance improvement. However, as discussed in detail in §6, there is no overall best algorithm, and time staging affects the optimization algorithms differently depending on the problem.

Benchmarks *Automatic Transmission* is a Simulink model that was proposed as a benchmark for falsification in [15]. It has input values *throttle* $\in [0, 100]$ and *brake* $\in [0, 325]$, and outputs the car's speed v,

the engine rotation ω, and the selected gear g.

With this model we consider five specifications **S1–5**. The first two are ceiling ones. Specification **S1** $\square_{[0,30]}\ (v < 120)$ (cf. the example in §2) states the speed be always below a threshold. This property is easily falsified with *throttle* $= 100$. Specification **S2** $\square_{[0,30]}\ (g = 3 \rightarrow v \geq 30)$ states that it is not possible to drive slowly in a high gear. A falsifying trajectory first has to speed up to reach this gear and subsequently roll out until speed falls below the threshold. This latter part of the trajectory can again be seen as a ceiling specification. Note that this property is interesting because the robustnes does not provide any guidance unless gear 3 has been entered by the system.

Specification **S3** is a reachability problem, $\diamond_{[10,30]}\ (v \leq v_{\min} \vee v \geq v_{\max})$, that encodes the search for a trajectory that keeps the speed between a lower and upper bound. The falsification problem consists of two sub-challenges: 1) hitting this speed interval precisely after an initial acceleration up to 10s simulated time; and 2) maintaining a correct speed till the time horizon. This suggests that a natural decomposition of the problem can indeed be achieved by separating these two aspects in time.

Specification **S4** $\square_{[0,10]}(v_{\min} < v) \vee \diamond_{[0,30]}(\omega > \omega_{\max})$ expresses that speed $v_{\min}$ can only be reached with an engine rotation exceeding a threshold $\omega_{\max}$. This specification is mentioned in [15] and evaluated in e.g., [3,4], too. To falsify, a trajectory must be found that reaches speed v early with an engine rotation lower than $\omega_{\max}$. The difficulty increases with higher $v_{\min}$ and lower $\omega_{\max}$, respectively. The formula represents the mixture of ceiling and reachability specifications.

The second system model is *Abstract Fuel Control* from [17]. It takes two input values, *pedal angle* (from $[0, 61.1]$) and *engine speed* (from $[0, 1100]$); it outputs *air-fuel ratio AF*, which influences fuel efficiency and performance of the car. The value of AF is expected to be close to a reference value AF_{ref}. According to [17], this setting corresponds to the so-called *normal mode*, where $AF_{\text{ref}} = 14.7$ is constant.

We used the specification $\neg(\diamond_{[t_1,t_2]}\square_{[0,t']}(AF - AF_{\text{ref}} > \delta * 14.7))$: the air-fuel ratio does not deviate from an acceptable range for more than t' seconds. We evaluated this specification with two parameter sets: the initial period with a larger error margin, and the stable period with a smaller margin. See Table 1 for parameter values.

Experimental Setup and Results For the experiments with the Automatic Transmission model, the input signals were piecewise constant with 5 control points. The time horizon was $T = 30$. The parameters outlined in §4 were as follows: the maximum number of samplings for each plain (non-staged) falsification trial was 150 (initial and optimization samplings combined). In the time-staged (TS) trials, we make the number of stages coincide with that of control points. Analogously, the sampling budget per stage was set to 30 for $K = 5$ stages, resulting in overall 150 samplings. The adaptive algorithm (A-TS) ran with the threshold $N^{\text{stall}}_{\max} = 30/2 = 15$ per each of five stages. The experiments with the Abstract Fuel Control model were run up to the time horizon $T = t_2 + t'$ where t_2 and t' are as in Table 1. We used three and five stages, respectively, for the initial and stable specifications. These again coincide with the number of control points. The TS algorithms conducted 30 samplings in each stage.

The experiments ran Breach version 1.2.9 and MATLAB R2017b on an Amazon EC2 c4.8xlarge instance with a 36 core Intel(R) Xeon(R) CPU (2.90GHz) and 58G of main memory. However, we did not use the opportunity to parallelize, and the time reported is in the same order of magnitude as that of a modern desktop workstation.

The results are shown in Table 1. They are grouped by the underlying stochastic optimization algorithm: CMA-ES, simulated annealing (SA) and global Nelder-Mead (GNM). In each group, we compare plain (unstaged) Breach to the time-staged (TS) and the adaptive time-staged (A-TS) ones. We compare average runtimes (lower is better) and the success rate (higher is better), aggregated over 20 falsifica-

tion trials for each configuration. Those good results which deserve attention are highlighted in bold. Note that the implementation of the global Nelder-Mead algorithm in Breach uses a deterministic source of quasi-randomness (Halton sequences), which implies that whether GNM finds a counterexample is consistent across all trials (marked with an asterisk $*$).

Discussion Focusing on the Automatic Transmission model first, we see that CMA-ES works well for S1, although GNM performs even better (6s, supposedly because it uses corner samples, see §3). Time staging introduces overhead to CMA-ES and GNM, because each stage is optimized individually. In contrast, simulated annealing (SA) benefits from time staging for the two ceiling specifications S1–2. We presume that since SA emphasizes exploration, it benefits from exploitation added by time staging (cf. §2).

The second specification S2 is slightly more complex: before gear 3 is reached, there is no guidance from the robustness semantics, because $[\![\_, g = 3]\!] = -\infty$ masks any quantitative information on v. Hence, falsifying this property needs some luck during the collection of initial samples in Algorithm 1. CMA-ES apparently exploits this, see top of column S2 of Table 1. Considering the other algorithms, SA and GNM, both benefit from time staging: exploitation of time causality prevents these good trajectory prefixes from being discarded accidentally once the required gear is reached (cf. Fig 2).

The results for S3 are evaluated with two different choices of parameters. The harder instance was falsified by the time-staged algorithms only, which can likely be attributed to the flattening of the search space from size $|U|^K$ to $K \cdot |U|$ (§5). S4 is evidently harder than the previous ones. Time staging improves performance in a general tendency but not in all cases (SA for S4 hard).

The results for the Abstract Fuel Control model (the last two columns in Table 1) show that the time-staged algorithms boost the ability to falsify some rare events. The specification for the initial stage where *AF* is still unstable (S init) can be considered a rare event since all the three non-staged algorithms failed to falsify it. Time-staged SA and time-staged GNM managed to find error inputs. In the last column (stable period) is remarkable, too, where success rate and run time of SA and GNM significantly improved.

Overall, while the performance of the non-staged algorithms suffers from tightening the bounds, the time-staged versions are able to find falsifying trajectories with good success rates while at the same time exhibiting significantly shorter runtimes.

7 Related Work

Falsification is a special case of search-based testing, so considerable research efforts have been made towards *coverage* [3, 7, 20]. The benefits of coverage in falsification guarantees are twofold. Firstly, they indicate confidence for correctness in case no counterexamples are found. Paired with sound robustness estimates for simulations, one can cover a an infinite parameter space with finitely many simulations. C2CE [13] is a recent tool that computes approximations of reachable states using such an approach. Secondly, coverage can be utilized for a better balance between *exploration* and *exploitation*: stochastic optimization algorithms can be called in an interleaved manner, in which coverage guides further exploration. The approach based on Rapidly-Exploring Random Trees [11] puts an emphasis on exploration by achieving high coverage of the state space. In their algorithm, robustness-guided hill-climbing optimization plays a supplementary role. Compared to these works, our current results go in an orthogonal direction, by utilizing time causality to enhance exploitation. The so-called multiple-shooting approach to falsification [24] can be seen of a generalization of RRTs. It consists of: an upper layer that searches

for an abstract error trace given by a succession of cells; and a lower layer where an abstract error trace is concretized to an actual error trace by picking points from cells. The approach can discover falsifying traces by backwards search from a goal region, but needs to concatenate partial traces with potential gaps, which can fail. Furthermore it is unclear how to extend it to general STL specifications. A survey of simulation based approaches has been done by Kapinski et al. [18].

Monotonicity has been exploited in different ways for falsification. Robust Neighborhood Descent [1, 2] (RED) searches for trajectories incrementally, restarting the search from points of low robustness. Descent computation of RED assumes explicit derivatives of the dynamics to guarantee convergence to (local) minima. It is the same principle underlying Prop. 5.4) and our experiments indicate that this principle is useful for black-box optimization, too. In [2], RED is paired with simulated annealing to combine local and global search and to account for more exploration. Doing so for our present work remains to be done in the future. In [16], Hoxha et al. mine parameters θ under which specifications $\phi[\theta]$ are satisfied or falsified by the system. They show that the robust semantics of formulas is monotone in θ and use that fact to tighten such parameters. This is orthogonal to this work as it does not use monotonicity of the system itself. Kim et al. [19] use an idea similar to [16] to partition specifications into upper bounds and lower ceilings. However, instead of robustness-guided optimization they use exhaustive exploration of the input space in a way that in turn requires that the system dynamics is monotone in the choice of each input at each time point. This is different from our Def 5.1 of time-monotonicity that aims at incrementally composing good partial choices.

The recent work [12] introduces a compositional falsification framework, focusing on those systems which include machine-learning (ML) components that perform tasks such as image recognition. While the current work aims at the orthogonal direction of finding rare counterexamples, we are interested in its combination with the results in [12], given the increasingly important roles of ML algorithms in CPS.

8 Conclusions and Future Work

We have introduced and evaluated the idea of time staging to enhance falsification for hybrid systems. The proposed method emphasizes exploitation over exploration as part of stochastic optimization. As there is no single algorithm that fits every problem (as a consequence of having no free lunch [23]), having a variety of methods at disposal permits the user of a system to choose the one suitable for the problem at hand. We have shown that the proposed approach is a good fit for problems that suitable exhibit time-causal structures, where it significantly outperforms non-staged algorithms.

Two obvious directions for future work have been pointed out already. Instead of just picking the best trajectory for each stage, it might be beneficial to retain a few, potentially diverse ones in the spirit of evolutionary algorithms (§2). For example, it would be interesting to explore the space between this work on one hand and coverage-driven rapidly-exploring random trees.

Another idea is to discover time stages adaptively (§5, the discussion after Prop. 5.4). For the experiments presented here, we chose to set uniformly fixed stages, which runs the risk of either being too coarse grained (missing some falsifying input), or being too fine grained (wasting analysis time).

Finally, another future direction is to explore variations of robust semantics to mitigate discrete propositions like $g = 3$ (§6), for example using averaging modalities [4]. Other t-norms than min/max for the semantics of conjunction/disjunction could preserve more information from different subformulas.

Acknowledgement. This work is supported by ERATO HASUO Metamathematics for Systems Design Project (No. JPMJER1603), Japan Science and Technology Agency.

References

[1] Houssam Abbas, Andrew K. Winn, Georgios E. Fainekos & A. Agung Julius (2014): *Functional gradient descent method for Metric Temporal Logic specifications*. In: *American Control Conference, ACC 2014, Portland, OR, USA, June 4-6, 2014*, IEEE, pp. 2312–2317, doi:10.1109/ACC.2014.6859453.

[2] Houssam Y Abbas (2015): *Test-based falsification and conformance testing for cyber-physical systems*. Ph.D. thesis, Arizona State University.

[3] Arvind S. Adimoolam, Thao Dang, Alexandre Donzé, James Kapinski & Xiaoqing Jin (2017): *Classification and Coverage-Based Falsification for Embedded Control Systems*. In Rupak Majumdar & Viktor Kuncak, editors: *Computer Aided Verification - 29th Int. Conf., CAV 2017*, *LNCS* 10426, Springer, pp. 483–503, doi:10.1007/978-3-319-63387-9_24.

[4] Takumi Akazaki & Ichiro Hasuo (2015): *Time Robustness in MTL and Expressivity in Hybrid System Falsification*. In Daniel Kroening & Corina S. Pasareanu, editors: *Computer Aided Verification - 27th Int. Conf., CAV 2015*, *LNCS* 9207, Springer, pp. 356–374, doi:10.1007/978-3-319-21668-3_21.

[5] Yashwanth Annpureddy, Che Liu, Georgios E. Fainekos & Sriram Sankaranarayanan (2011): *S-TaLiRo: A Tool for Temporal Logic Falsification for Hybrid Systems*. In Parosh Aziz Abdulla & K. Rustan M. Leino, editors: *Tools and Algorithms for the Construction and Analysis of Systems - 17th Int. Conf., TACAS 2011*, *LNCS* 6605, Springer, pp. 254–257, doi:10.1007/978-3-642-19835-9_21.

[6] Anne Auger & Nikolaus Hansen (2005): *A restart CMA evolution strategy with increasing population size*. In: *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2005*, IEEE, pp. 1769–1776, doi:10.1109/CEC.2005.1554902.

[7] Jyotirmoy V. Deshmukh, Xiaoqing Jin, James Kapinski & Oded Maler (2015): *Stochastic Local Search for Falsification of Hybrid Systems*. In Bernd Finkbeiner, Geguang Pu & Lijun Zhang, editors: *Automated Technology for Verification and Analysis - 13th Int. Symp., ATVA 2015*, *LNCS* 9364, Springer, pp. 500–517, doi:10.1007/978-3-319-24953-7_35.

[8] Alexandre Donzé (2010): *Breach, A Toolbox for Verification and Parameter Synthesis of Hybrid Systems*. In Tayssir Touili, Byron Cook & Paul B. Jackson, editors: *Computer Aided Verification, 22nd Int. Conf., CAV 2010*, *LNCS* 6174, Springer, pp. 167–170, doi:10.1007/978-3-642-14295-6_17.

[9] Alexandre Donzé, Thomas Ferrère & Oded Maler (2013): *Efficient Robust Monitoring for STL*. In Natasha Sharygina & Helmut Veith, editors: *Computer Aided Verification - 25th Int. Conf., CAV 2013*, *LNCS* 8044, Springer, pp. 264–279, doi:10.1007/978-3-642-39799-8_19.

[10] Alexandre Donzé & Oded Maler (2010): *Robust Satisfaction of Temporal Logic over Real-Valued Signals*. In Krishnendu Chatterjee & Thomas A. Henzinger, editors: *Formal Modeling and Analysis of Timed Systems - 8th Int. Conf., FORMATS 2010*, *LNCS* 6246, Springer, pp. 92–106, doi:10.1007/978-3-642-15297-9_9.

[11] Tommaso Dreossi, Thao Dang, Alexandre Donzé, James Kapinski, Xiaoqing Jin & Jyotirmoy V. Deshmukh (2015): *Efficient Guiding Strategies for Testing of Temporal Properties of Hybrid Systems*. In Klaus Havelund, Gerard J. Holzmann & Rajeev Joshi, editors: *NASA Formal Methods - 7th Int. Symp., NFM 2015*, *LNCS* 9058, Springer, pp. 127–142, doi:10.1007/978-3-319-17524-9_10.

[12] Tommaso Dreossi, Alexandre Donzé & Sanjit A. Seshia (2017): *Compositional Falsification of Cyber-Physical Systems with Machine Learning Components*. In Clark Barrett, Misty Davies & Temesghen Kahsai, editors: *NASA Formal Methods - 9th Int. Symp., NFM 2017*, *LNCS* 10227, pp. 357–372, doi:10.1007/978-3-319-57288-8_26.

[13] Parasara Sridhar Duggirala, Sayan Mitra, Mahesh Viswanathan & Matthew Potok (2015): *C2E2: A Verification Tool for Stateflow Models*. In Christel Baier & Cesare Tinelli, editors: *Tools and Algorithms for the Construction and Analysis of Systems - 21st Int. Conf., TACAS 2015*, *LNCS* 9035, Springer, pp. 68–82, doi:10.1007/978-3-662-46681-0_5.

[14] Georgios E. Fainekos & George J. Pappas (2009): *Robustness of temporal logic specifications for continuous-time signals*. *Theor. Comput. Sci.* 410(42), pp. 4262–4291, doi:10.1016/j.tcs.2009.06.021.

[15] Bardh Hoxha, Houssam Abbas & Georgios E. Fainekos (2014): *Benchmarks for Temporal Logic Requirements for Automotive Systems*. In Goran Frehse & Matthias Althoff, editors: *1st and 2nd Int. Workshops on Applied veRification for Continuous and Hybrid Systems, ARCH@CPSWeek 2014 and 2105*, *EPiC Series in Computing* 34, EasyChair, pp. 25–30, doi:10.29007/xwrs.

[16] Bardh Hoxha, Adel Dokhanchi & Georgios E. Fainekos (2018): *Mining parametric temporal logic properties in model-based design for cyber-physical systems*. *STTT* 20(1), pp. 79–93, doi:10.1007/s10009-017-0447-4.

[17] Xiaoqing Jin, Jyotirmoy V. Deshmukh, James Kapinski, Koichi Ueda & Kenneth R. Butts (2014): *Powertrain control verification benchmark*. In Martin Fränzle & John Lygeros, editors: *17th International Conference on Hybrid Systems: Computation and Control (part of CPS Week), HSCC'14, Berlin, Germany, April 15-17, 2014*, ACM, pp. 253–262, doi:10.1145/2562059.2562140.

[18] James Kapinski, Jyotirmoy V Deshmukh, Xiaoqing Jin, Hisahiro Ito & Ken Butts (2016): *Simulation-based approaches for verification of embedded control systems: an overview of traditional and advanced modeling, testing, and verification techniques*. *IEEE Control Systems* 36(6), pp. 45–64, doi:10.1109/MCS.2016.2602089.

[19] Eric S. Kim, Murat Arcak & Sanjit A. Seshia (2016): *Directed Specifications and Assumption Mining for Monotone Dynamical Systems*. In Alessandro Abate & Georgios E. Fainekos, editors: *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control, HSCC 2016, Vienna, Austria, April 12-14, 2016*, ACM, pp. 21–30, doi:10.1145/2883817.2883833.

[20] Jan Kurátko & Stefan Ratschan (2014): *Combined Global and Local Search for the Falsification of Hybrid Systems*. In Axel Legay & Marius Bozga, editors: *Formal Modeling and Analysis of Timed Systems - 12th Int. Conf., FORMATS 2014*, *LNCS* 8711, Springer, pp. 146–160, doi:10.1007/978-3-319-10512-3_11.

[21] Oded Maler & Dejan Nickovic (2004): *Monitoring Temporal Properties of Continuous Signals*. In Yassine Lakhnech & Sergio Yovine, editors: *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems, Joint Int. Confs. on Formal Modelling and Analysis of Timed Systems, FORMATS 2004 and Formal Techniques in Real-Time and Fault-Tolerant Systems, FTRTFT 2004*, *LNCS* 3253, Springer, pp. 152–166, doi:10.1007/978-3-540-30206-3_12.

[22] Dogan Ulus, Thomas Ferrère, Eugene Asarin & Oded Maler (2016): *Online Timed Pattern Matching Using Derivatives*. In Marsha Chechik & Jean-François Raskin, editors: *Tools and Algorithms for the Construction and Analysis of Systems - 22nd Int. Conf., TACAS 2016*, *LNCS* 9636, Springer, pp. 736–751, doi:10.1007/978-3-662-49674-9_47.

[23] David Wolpert & William G. Macready (1997): *No free lunch theorems for optimization*. *IEEE Trans. Evolutionary Computation* 1(1), pp. 67–82, doi:10.1109/4235.585893.

[24] Aditya Zutshi, Jyotirmoy V. Deshmukh, Sriram Sankaranarayanan & James Kapinski (2014): *Multiple shooting, CEGAR-based falsification for hybrid systems*. In Tulika Mitra & Jan Reineke, editors: *2014 International Conference on Embedded Software, EMSOFT 2014, New Delhi, India, October 12-17, 2014*, ACM, pp. 5:1–5:10, doi:10.1145/2656045.2656061.

A STL Semantics for Time-Bounded Signals

Definition A.1 (robust semantics for time-bounded signals)**.** Let $T \in \mathbb{R}_{>0}$, $\mathbf{w}\colon [0,T] \to \mathbb{R}^N$ be a time-bounded signal, and φ be an **STL** formula. We define the *robustness* $[\![\mathbf{w},\varphi]\!]^T \in \mathbb{R}\cup\{\infty,-\infty\}$ of $\mathbf{w}$ with respect to φ as follows, by induction. Here the superscript T is an annotation that designates the time horizon.

$$\begin{array}{ll} [\![\mathbf{w}, f(x_1,\dots,x_n)>0]\!]^T := f\big(\mathbf{w}(0)(x_1),\dots,\mathbf{w}(0)(x_n)\big) & [\![\mathbf{w},\bot]\!]^T := -\infty \\ [\![\mathbf{w},\neg\varphi]\!]^T := -[\![\mathbf{w},\varphi]\!]^T & [\![\mathbf{w},\varphi_1\wedge\varphi_2]\!]^T := [\![\mathbf{w},\varphi_1]\!]^T \sqcap [\![\mathbf{w},\varphi_2]\!]^T \\ [\![\mathbf{w},\varphi_1\,\mathscr{U}_I\,\varphi_2]\!]^T := \bigsqcup_{t\in I\cap[0,T]} \big([\![\mathbf{w}^t,\varphi_2]\!]^{T-t} \sqcap \prod_{t'\in[0,t)} [\![\mathbf{w}^{t'},\varphi_1]\!]^{T-t'}\big) & \end{array}$$

The Boolean semantics $\models$, found e.g. in [10], allows a similar adaptation to time-bounded signals, too.

B Brzozowski Derivative of Flat STL Formulas

In the time-staged falsification procedure we often encounter the following situation: an **STL** formula φ and a signal $\mathbf{v}\colon [0,T] \to \mathbb{R}^N$ are fixed; and we have to compute robustness $[\![\mathbf{v}\cdot\mathbf{v}',\varphi]\!]$ for a number of different signals $\mathbf{v}'\colon [0,T'] \to \mathbb{R}^N$. To aid such computation, a natural idea is to use a syntactic construct $\partial_{\mathbf{v}}\varphi$ of *(Brzozowski) derivative*. It should be compatible with robust semantics in the sense that $[\![\mathbf{v}\cdot\mathbf{v}',\varphi]\!] = [\![\mathbf{v}',\partial_{\mathbf{v}}\varphi]\!]$, reducing the computation of the LHS to that of the RHS.

Similar use of derivatives is found e.g. in [22]. The settings are different, though: Boolean semantics is used in [22] while we use quantitative robust semantics. In fact, the definition of derivatives in this section focuses on *flat* formulas (i.e. free from nested modalities). This restriction is mandated by the quantitative semantics, as our proof later suggests. Anyway, the definitions and results in this section are new to the best of our knowledge.

We need the following extension of **STL** syntax.

Definition B.1 (extended **STL**)**.** We extend the syntax of **STL** (Def. 3.4) by atomic propositions c_r for each $r \in \mathbb{R}$. The robust semantics in Def. 3.5 (and that in Def. A.1 in Appendix A) is extended accordingly: $[\![\mathbf{w},\mathsf{c}_r]\!] := r$.

Intuitively c_r is an atomic proposition that constantly returns the robustness value r.

Definition B.2 (derivative)**.** Let $T \in \mathbb{R}_{>0}$, and $\mathbf{v}\colon [0,T] \to \mathbb{R}^N$ be a time-bounded signal. For each extended **STL** formula φ, we define its *derivative* $\partial_{\mathbf{v}}\varphi$ by $\mathbf{v}$ by the following induction.

$$\begin{array}{ll} \partial_{\mathbf{v}}\big(f(\vec{x})>0\big) :\equiv \mathsf{c}_{[\![\mathbf{v},f(\vec{x})>0]\!]} & \partial_{\mathbf{v}}\mathsf{c}_r :\equiv \mathsf{c}_r \qquad \partial_{\mathbf{v}}\bot :\equiv \bot \\ \partial_{\mathbf{v}}(\neg\varphi) :\equiv \neg\partial_{\mathbf{v}}\varphi & \partial_{\mathbf{v}}(\varphi_1\wedge\varphi_2) :\equiv (\partial_{\mathbf{v}}\varphi_1)\wedge(\partial_{\mathbf{v}}\varphi_2) \\ \partial_{\mathbf{v}}(\varphi_1\,\mathscr{U}_I\,\varphi_2) :\equiv \mathsf{c}_{[\![\mathbf{v},\varphi_1\mathscr{U}_I\varphi_2]\!]} \vee \big((\mathsf{c}_{[\![\mathbf{v},\Box\varphi_1]\!]}\wedge\varphi_1)\,\mathscr{U}_{I-T}\,\varphi_2\big) & \end{array}$$

Here the interval $I-T$ is obtained from I by shifting both of its endpoints earlier by T, such as $[a,b]-T = [a-T,b-T]$.

Definition B.3 (flat **STL** formula)**.** An **STL** formula φ is *flat* if it does not have nested temporal modal operators. This means: if $\varphi_1\,\mathscr{U}_I\,\varphi_2$ is a subformula of φ, then neither φ_1 nor φ_2 contains $\mathscr{U}$.

Proposition B.4. *Let $T \in \mathbb{R}_{>0}$, $\mathbf{v}\colon [0,T] \to \mathbb{R}^N$ be a signal, and φ be a flat **STL** formula. We have, for each $T' \in \mathbb{R}_{>0}$ and $\mathbf{v}'\colon [0,T'] \to \mathbb{R}^N$, $[\![\mathbf{v}',\partial_{\mathbf{v}}\varphi]\!] = [\![\mathbf{v}\cdot\mathbf{v}',\varphi]\!]$.*

Proof. By induction on the construction of φ. Most equalities below follow from the definition of ∂ and that of $[\![\_]\!]$.

$$\begin{aligned}
[\![\mathbf{v}', \partial_{\mathbf{v}}\big(f(\vec{x}) > 0\big)]\!] &= [\![\mathbf{v}', \mathsf{c}_{[\![\mathbf{v}, f(\vec{x})>0]\!]}]\!] = [\![\mathbf{v}, f(\vec{x}) > 0]\!] = [\![\mathbf{v}\cdot\mathbf{v}', f(\vec{x}) > 0]\!] \\
[\![\mathbf{v}', \partial_{\mathbf{v}}\mathsf{c}_r]\!] &= [\![\mathbf{v}', \mathsf{c}_r]\!] = r = [\![\mathbf{v}\cdot\mathbf{v}', \mathsf{c}_r]\!] \\
[\![\mathbf{v}', \partial_{\mathbf{v}}\bot]\!] &= [\![\mathbf{v}', \bot]\!] = -\infty = [\![\mathbf{v}\cdot\mathbf{v}', \bot]\!] \\
[\![\mathbf{v}', \partial_{\mathbf{v}}(\neg\varphi)]\!] &= [\![\mathbf{v}', \neg\partial_{\mathbf{v}}\varphi]\!] = -[\![\mathbf{v}', \partial_{\mathbf{v}}\varphi]\!] \overset{\text{I.H.}}{=} -[\![\mathbf{v}\cdot\mathbf{v}', \varphi]\!] = [\![\mathbf{v}\cdot\mathbf{v}', \neg\varphi]\!] \\
[\![\mathbf{v}', \partial_{\mathbf{v}}(\varphi_1 \wedge \varphi_2)]\!] &= [\![\mathbf{v}', (\partial_{\mathbf{v}}\varphi_1) \wedge (\partial_{\mathbf{v}}\varphi_2)]\!] = [\![\mathbf{v}', \partial_{\mathbf{v}}\varphi_1]\!] \sqcap [\![\mathbf{v}', \partial_{\mathbf{v}}\varphi_2]\!] \\
&\overset{\text{I.H.}}{=} [\![\mathbf{v}\cdot\mathbf{v}', \varphi_1]\!] \sqcap [\![\mathbf{v}\cdot\mathbf{v}', \varphi_2]\!] = [\![\mathbf{v}\cdot\mathbf{v}', \varphi_1 \wedge \varphi_2]\!]
\end{aligned}$$

Here is a nontrivial case.

$$\begin{aligned}
&[\![\mathbf{v}', \partial_{\mathbf{v}}(\varphi_1 \mathscr{U}_I \varphi_2)]\!] \\
&= [\![\mathbf{v}', \mathsf{c}_{[\![\mathbf{v}, \varphi_1 \mathscr{U}_I \varphi_2]\!]}]\!] \sqcup [\![\mathbf{v}', \big((\mathsf{c}_{[\![\mathbf{v}, \Box\varphi_1]\!]} \wedge \varphi_1) \mathscr{U}_{I-T} \varphi_2\big)]\!] \\
&= [\![\mathbf{v}, \varphi_1 \mathscr{U}_I \varphi_2]\!] \sqcup \bigsqcup_{t \in (I-T)\cap[0,T']} \big([\![\mathbf{v}'^t, \varphi_2]\!] \sqcap [\![\mathbf{v}, \Box\varphi_1]\!] \sqcap \bigsqcap_{t' \in [0,t)} [\![\mathbf{v}'^{t'}, \varphi_1]\!]\big) \\
&= \bigsqcup_{t \in I\cap[0,T]} \big([\![\mathbf{v}^t, \varphi_2]\!] \sqcap \bigsqcap_{t' \in [0,t)} [\![\mathbf{v}^{t'}, \varphi_1]\!]\big) \\
&\quad \sqcup \bigsqcup_{t \in (I-T)\cap[0,T']} \big([\![\mathbf{v}'^t, \varphi_2]\!] \sqcap \big(\bigsqcap_{t' \in [0,T]} [\![\mathbf{v}^{t'}, \varphi_1]\!]\big) \sqcap \bigsqcap_{t' \in [T,T+t)} [\![(\mathbf{v}\cdot\mathbf{v}')^{t'}, \varphi_1]\!]\big) \\
&\overset{(*)}{=} \bigsqcup_{t \in I\cap[0,T]} \big([\![(\mathbf{v}\cdot\mathbf{v}')^t, \varphi_2]\!] \sqcap \bigsqcap_{t' \in [0,t)} [\![(\mathbf{v}\cdot\mathbf{v}')^{t'}, \varphi_1]\!]\big) \\
&\quad \sqcup \bigsqcup_{t'' \in I\cap[T,T+T']} \big([\![(\mathbf{v}\cdot\mathbf{v}')^{t''}, \varphi_2]\!] \sqcap \big(\bigsqcap_{t' \in [0,T]} [\![(\mathbf{v}\cdot\mathbf{v}')^{t'}, \varphi_1]\!]\big) \sqcap \bigsqcap_{t' \in [T,t'')} [\![(\mathbf{v}\cdot\mathbf{v}')^{t'}, \varphi_1]\!]\big) \\
&= \bigsqcup_{t \in I\cap[0,T]} \big([\![(\mathbf{v}\cdot\mathbf{v}')^t, \varphi_2]\!] \sqcap \bigsqcap_{t' \in [0,t)} [\![(\mathbf{v}\cdot\mathbf{v}')^{t'}, \varphi_1]\!]\big) \\
&\quad \sqcup \bigsqcup_{t'' \in I\cap[T,T+T']} \big([\![(\mathbf{v}\cdot\mathbf{v}')^{t''}, \varphi_2]\!] \sqcap \bigsqcap_{t' \in [0,t'')} [\![(\mathbf{v}\cdot\mathbf{v}')^{t'}, \varphi_1]\!]\big) \\
&= \bigsqcup_{t \in I\cap[0,T+T'']} \big([\![(\mathbf{v}\cdot\mathbf{v}')^t, \varphi_2]\!] \sqcap \bigsqcap_{t' \in [0,t)} [\![(\mathbf{v}\cdot\mathbf{v}')^{t'}, \varphi_1]\!]\big) \\
&= [\![\mathbf{v}\cdot\mathbf{v}', \varphi_1 \mathscr{U}_I \varphi_2]\!]
\end{aligned}$$

In $(*)$ we used the following facts. Firstly, for a formula ψ without temporal operators, we have $[\![\mathbf{v}, \psi]\!] = [\![\mathbf{v}\cdot\mathbf{v}', \psi]\!]$. Secondly, if $\mathbf{v}$'s domain is $[0,T]$ and $t \in [0,T]$, then $\mathbf{v}^t \cdot \mathbf{v}' = (\mathbf{v}\cdot\mathbf{v}')^t$.

□

Note that the flatness assumption on φ is crucially used in the proof step. Modifying Def. 4.2 in order to accommodate nested modalities seems hard, after analyzing the proof step $(*)$.

C Omitted Proofs

C.1 Proof of Prop. 5.4

Proof. By the definitions we have, for each input signal $\mathbf{u}\colon [0,T] \to \mathbb{R}^M$,

$$[\![\mathscr{M}(\mathbf{u}), \square(x < c)]\!] = \bigsqcap_{t\in[0,T]} c - \mathscr{M}(\mathbf{u})(t)(x) \ .$$

Therefore the assumption $[\![\mathscr{M}(\mathbf{u}_1), \varphi]\!] \leq [\![\mathscr{M}(\mathbf{u}_1'), \varphi]\!]$ expands to

$$\bigsqcap_{t\in[0,T_1]} c - \mathscr{M}(\mathbf{u}_1)(t)(x) \leq \bigsqcap_{t\in[0,T_1]} c - \mathscr{M}(\mathbf{u}_1')(t)(x) \ . \tag{4}$$

The first infimum in the above is taken over a compact domain $[0,T_1]$; therefore there exists $T \in [0,T_1]$ that achieves the infimum. Let T be such a real number. The following is obvious.

$$\begin{aligned} &\bigsqcap_{t\in[0,T_1]} c - \mathscr{M}(\mathbf{u}_1)(t)(x) = \bigsqcap_{t\in[0,T]} c - \mathscr{M}(\mathbf{u}_1)(t)(x) = c - \mathscr{M}(\mathbf{u}_1)(T)(x) \\ &\leq \bigsqcap_{t\in[0,T_1]} c - \mathscr{M}(\mathbf{u}_1')(t)(x) \leq \bigsqcap_{t\in[0,T]} c - \mathscr{M}(\mathbf{u}_1')(t)(x) \ . \end{aligned} \tag{5}$$

Another immediate consequence, derived using the causality of $\mathscr{M}$ (Def. 3.2), is

$$c - \mathscr{M}(\mathbf{u}_1|_{[0,T]})(T)(x) \ \leq \ c - \mathscr{M}(\mathbf{u}_1'|_{[0,T]})(T)(x) \ . \tag{6}$$

Our goal is to show $[\![\mathscr{M}(\mathbf{u}_1|_{[0,T]} \cdot \mathbf{u}_2), \varphi]\!] \leq [\![\mathscr{M}(\mathbf{u}_1'|_{[0,T]} \cdot \mathbf{u}_2), \varphi]\!]$.

$$\begin{aligned} &[\![\mathscr{M}(\mathbf{u}_1'|_{[0,T]} \cdot \mathbf{u}_2), \square(x < c)]\!] \\ &= \bigsqcap_{t\in[0,T+T_2]} c - \mathscr{M}(\mathbf{u}_1'|_{[0,T]} \cdot \mathbf{u}_2)(t)(x) \\ &= \bigsqcap_{t\in[0,T]} c - \mathscr{M}(\mathbf{u}_1')(t)(x) \sqcap \bigsqcap_{t\in(T,T+T_2]} c - \mathscr{M}(\mathbf{u}_1'|_{[0,T]} \cdot \mathbf{u}_2|_{[0,t-T]})(t)(x) && (*) \\ &\geq \bigsqcap_{t\in[0,T]} c - \mathscr{M}(\mathbf{u}_1)(t)(x) \sqcap \bigsqcap_{t\in(T,T+T_2]} c - \mathscr{M}(\mathbf{u}_1'|_{[0,T]} \cdot \mathbf{u}_2|_{[0,t-T]})(t)(x) \quad \text{by (4)} \\ &\geq \bigsqcap_{t\in[0,T]} c - \mathscr{M}(\mathbf{u}_1)(t)(x) \sqcap \bigsqcap_{t\in(T,T+T_2]} c - \mathscr{M}(\mathbf{u}_1|_{[0,T]} \cdot \mathbf{u}_2|_{[0,t-T]})(t)(x) && (\dagger) \\ &= \cdots \\ &= [\![\mathscr{M}(\mathbf{u}_1|_{[0,T]} \cdot \mathbf{u}_2), \square(x < c)]\!]. \end{aligned}$$

In the above we heavily used the causality of $\mathscr{M}$ (Def. 3.2). For example, in the step $(*)$ above, causality is used in deriving $\mathscr{M}(\mathbf{u}_1'|_{[0,T]} \cdot \mathbf{u}_2)(t) = \mathscr{M}(\mathbf{u}_1')(t)$. In the step $(\dagger)$ we applied the monotonicity of $\mathscr{M}$ to the signals $\mathbf{u}_1|_{[0,T]}$, $\mathbf{u}_1'|_{[0,T]}$ and $\mathbf{u}_2|_{[0,t-T]}$. Note that (6) allows to do so.

□