Temporal Logic Falsification of Cyber-Physical Systems: An Input-Signal-Space Optimization Approach

A. Aerts[1]   B. Tong Minh[2]   M.R. Mousavi[3]   M.A. Reniers[4]

Abstract— **Temporal logic falsification is a promising approach to model-based testing of cyber-physical systems. It starts off with a formalized system requirement specified as a Metric Temporal Logic (MTL) property. Subsequently, test input signals are generated in order to stimulate the system and produce an output signal. Finally, output signals of the system under test are compared to those prescribed by the property to falsify the property by means of a counterexample. To find such a counterexample, Markov Chain Monte-Carlo (MCMC) methods are used to construct an optimization problem to steer the test input generations to those input areas that maximize the probability of falsifying the property.**

In this paper, we identify two practical issues in the above-mentioned falsification process. Firstly, a fixed time domain of the input-signal space is assumed in this process, which restricts the frequency content of the (generated) input signals. Secondly, the existing process allows for input selection steered by the distribution of a single input variable.

We address these issues, by firstly, considering multiple time domains for input-signal space. Subsequently, an input-signal-space optimization problem is formally defined and implemented in S-TaLiRo+, an extension of S-TaLiRo (an existing implementation for solving the MTL falsification problem). Secondly, we propose a decoupled scheme that considers the distribution of each input variable independently. The applicability of the proposed solutions are experimentally evaluated on well-known benchmark problems.

I. Introduction

Model-Based Testing (MBT) is a technique that originates from software and system testing [19], [31] and has been traditionally applied to software as well as discrete event systems [9], [30]. Many attempts have been recently made to adapt MBT to the domain of cyber-physical systems (CPSs), where discrete and continuous aspects of system behavior are intertwined, cf. [7], [15], [21], [23] and references therein.

In this paper, we focus on a promising example of such attempts, namely, on temporal logic falsification [4] for CPSs. This technique evaluates the conformance of a system to a set of formal properties expressed in Metric Temporal Logic (MTL) [24]. A subset of the input-signal space is used to trigger the system and observe the output behaviour on which the formal property is checked. Hence, the goal is to find an input signal such that a slighlty deviating output signal may falsify the formal property. This technique is implemented in the S-TaLiRo (Systems Temporal Logic Robustness) toolset [8] as a Matlab toolbox.

In the above-mentioned model-based testing process, if no falsifying output trajectory or so-called falsifier is found, the CPS model is assumed to conform to the MTL property. Note that due to the nature of testing, the absence of falsifying model (output) behavior does not formally prove correctness (i.e., only non-conformance can be proved). In [5], the aforementioned techniques have been applied to a soccer wheelchair case study. In the course of this study, we observed the following two practical issues with the existing approach:

1) S-TaLiRo uses a coupled input proposal scheme for their Simulated Annealing search approach. Hence, the most constrained input will unavoidably restrict the proposal scheme in all other dimensions.
2) The existing approach uses a fixed number of control points for all input signals, for which the interval times of the control points are optimized. Suppose that no falsifiers can be found for the given CPS model, given an input-signal-space subset, it can be the case that a different subset of the input-signal space (using a different number of control points) may be able to prove non-conformance. As a result, based on a single input-signal-space subset, invalid conformance verdicts might be reached. (We refer to [26] for an analysis of the effect of sampling on the soundness and the completeness of conformance testing.)

In [4], this coupled proposal scheme is applied on models with one input-signal. However, when considering multiple inputs (higher dimensional input spaces), the effectiveness of the approach can decrease significantly, i.e., previously achieved falsification rates (based on a single input) may drop substantially. We show examples of this phenomenon in the remainder of this paper. Moreover, we devise a *decoupled proposal scheme* which, by decoupling independent input signals, improves the effectiveness of the existing approach when dealing with multiple inputs.

Furthermore, we propose an extension of the MTL falsification problem in this paper, namely an *input-signal-space optimization approach* which allows an additional optimization over the time domains of multiple input signals. As a result, when searching for falsifying model output behavior, a (non-)conformance verdict considers numerous subsets of the model input-signal space and hence, provides a more reliable conformance result.

[1]Arend Aerts is with the Precision Motion Control department, Bosch Rexroth, Eindhoven, The Netherlands `arend.aerts@boschrexroth.nl`

[3]Bryan Tong Minh is with ALTEN Mechatronics, Eindhoven, The Netherlands `bryan.tongminh@alten.nl`

[3]Mohammad Reza Mousavi is with the Department of Informatics, University of Leicester, Leicester, UK `mm789@le.ac.uk`

[4]Michel A. Reniers is with the Department of Mechanical Engineering, Eindhoven University of Technology, Eindhoven, The Netherlands `m.a.reniers@tue.nl`

Both of the above-mentioned improvements are implemented in S-TaLiRo+ (see [5], [6] for source code and application notes), an extension of S-TaLiRo which supports the input-signal space MTL falsification problem with decoupled proposal scheme. Moreover, in this work, benchmarks from [4] are used and extended to show the effectiveness of the extended and optimized approach.

The remainder of this paper is structured as follows. The (original) MTL falsification problem [4] and a black-box solution to it are introduced in Section II. In Section III, two benchmarks are introduced that are used to evaluate the proposed adaptations. In Section IV, the decoupled proposal and its experimental verification are discussed. The input-signal-space MTL falsification problem and the experimental evaluation of the associated solution are discussed in Section V. Section VI discusses an experimental evaluation of a solution where both adaptations are included. Finally, concluding remarks are presented in Section VII.

II. MTL FALSIFICATION

In this section, we recall some basic notions regarding MTL falsification that are used throughout the rest of the paper.

A. System model, input and output spaces

A (system) model Σ is assumed to be a mapping $\Delta_\Sigma : X_0 \times \mathbf{U} \to Y^R$ from a compact set of initial conditions X_0 (the set of vectors of valid input values) and a compact set of input signals $\mathbf{U} \subseteq U^R$ to output signals Y^R. In these notations U is a compact set of possible input values (the input space), Y is the set of output values (the output space), and R is a bounded time domain.

For the convergence of the test-case generation algorithm, it is assumed that the input space U is discretized and limited to a (possibly) large but finite set. Additional restrictions on the type of models that are considered in this paper are the same as in [4]. We do not list these explicitly here as they play no role in the forthcoming developments.

B. MTL properties and falsification

In order to construct (timed) specifications for CPSs, Metric Temporal Logic (MTL) [24], [27] and variants thereof such as Signal Temporal Logic (STL) [25] are used. These logics originate from Linear Temporal Logic (LTL) [28]. Although both logics are suitable to formalize system requirements in this setting, MTL is used in this paper. In the remainder of this section, we briefly and informally introduce MTL and refer to [17] for a formal and detailed treatment.

In an MTL formula, propositions refer to properties of the output space, i.e., each proposition stands for an atomic property satisfied by a subset of the output space. Propositions can be composed using classical propositional connectors such as conjunction, disjunction, and negation. Moreover temporal modalities such as $\Diamond_I \phi$ (for eventually within I) specify that formula ϕ holds somewhere within the time interval I. Other temporal modalities include $\Box_I \phi$ (for always within I) and $\phi\ \mathcal{U}_I\ \psi$ (for until within I).

The semantics of an MTL formula is the set of output signals satisfying it; as stated above, an atomic proposition stands for a set of outputs in the output space and the semantics is inductively defined for propositional and temporal operators in a straightforward manner.

An MTL formula is falsified on a system if there is an initial condition and an input signal, for which the the output signal does not satisfy the MTL formula.

C. MTL robustness

It is assumed that the output-signal space is equipped with a (generalized) metric defining how far apart two (sets of) output signals are (see [4]). Hence, it is possible to define the robustness of a given output signal with respect to an MTL formula in terms of the distance of the output signal from the set of output signals which violate (or satisfy) the property: a large robustness value means that the signal is far from falsifying it. Negative robustness values denote the signals violating the property and positive robustness values denote the signals satisfying it.

In the remainder of this paper, for an MTL formula φ and an output signal y, the robustness of output trajectory y w.r.t. property φ is denoted $\mathcal{D}_\varphi(y)$. For more details about the definition of this robustness value, we refer to [4]. However, this information is not needed for the developments in this paper.

D. Falsification as optimization

In [4], falsification of MTL formula φ on system $\Delta_\Sigma : X_0 \times \mathbf{U} \to Y^R$ is reduced to the following optimization problem:

$$(x_0^*, u^*) = \arg \min_{x_0 \in X_0, u \in \mathbf{U}} \mathcal{D}_\varphi(\Delta_\Sigma(x_0, u)) \qquad (1)$$

where the robustness value $\mathcal{D}_\varphi(\Delta_\Sigma(x_0, u))$ denotes the extent to which the property φ is falsified by the output associated with input signal u and initial condition x_0. In case we end up with x_0^* and u^* with $\mathcal{D}_\varphi(\Delta_\Sigma(x_0^*, u^*)) < 0$, a counterexample is found.

In Equation 1, a feasible optimization problem is formulated, i.e., the initial conditions X_0 and the input space $\mathbf{U}$ can be manipulated by (test-)case generation tools, e.g., by input-based search algorithms such as Monte-Carlo techniques. Note that Equation 1 deviates in notation from what is proposed in [4], namely the separation between x_0 and u, for purposes that become clear later. Next, the algorithmic solution of the minimization problem (e.g., test-case generation) is further discussed.

E. MTL falsification solution

In order to solve the MTL falsification problem, a counterexample to the property φ has to be found. If none is found, the property is considered (possibly marginally) satisfied on the system model. To discover such a counterexample, Monte-Carlo optimization techniques (see [22] for an overview) are used to perform a random walk over the initial conditions X_0 and input-signal space $\mathbf{U}$. This randomized search is steered by the above-discussed robustness metric

[12], [16], i.e., initial condition x_0 and input signal u which result in an output signal with a lower robustness value are favored.

In Figure 1, the MTL falsification solution is presented schematically. It contains a model Σ, an MTL robustness block representing the MTL robustness computation, and a test-case generation block containing the Monte-Carlo optimization technique. In fact, the observed closed-loop (of these three blocks) depicts a global optimization problem, i.e., to find a counterexample to the MTL falsification problem by discovering the global minimizer of the robustness degree ($\mathcal{D}_\varphi(y)$). In this case, the MTL robustness computation serves as a cost function for the Monte-Carlo optimization technique in the test-case generation. Hence, the latter begins a random walk over the input-signal space by generating and applying an input ($x_0 \in X_0, u \in \mathbf{U}$) to the model Σ and observing an output trajectory $y \in Y^R$. Subsequently, based on this trajectory and the formal property φ, a robustness value ($\mathcal{D}_\varphi(y)$) is computed which serves as 'steer' input for the next initial condition and input signal. Note that a negative robustness value indicates that a counterexample is found, while a positive value indicates that the property φ is satisfied on y (both with some margin depicted by the magnitude of $\mathcal{D}_\varphi(y)$). Finally, after identifying a falsifier, the MTL falsification solution outputs the minimal robustness value $\min(\mathcal{D}_\varphi(y))$ and the corresponding model behavior $(x_0, u, y)_{fals}$.

Mechanising this technique in a tool, such as S-TaLiRo, is a non-non-trivial aspect of MBT, in particular regarding the industrial applicability of the technique. In industry, effective and especially affordable tooling may influence or determine the choice for a particular testing approach. S-TaLiRo was developed in collaboration with industry [21] to respond to such a need and the present paper attempts to enhance its effectiveness further. The tool description in the remainder of this section is based on [4].

In order to solve an optimization problem in a black-box fashion, stochastic (search) algorithms are utilized by S-TaLiRo. Such techniques typically evade local minima, often present when solving an optimization problem (cf. the work of Abbas and Fainekos [2] where local sub-gradient descent is used to optimize the search). The toolset has a modular structure and hence, it supports a wide range of optimization engines such as Simulated Annealing (SA) [1], Cross-Entropy [29], and (extended) Ant-Colony Optimization [13]. In this paper, simulated annealing is selected as a Monte-Carlo technique to realize the test-case generation step of Figure 1, due to its proven service record [14]. In fact, to improve its convergence and test-input generation properties, SA is combined with acceptance-rejection sampling [10] and extended to the class of Markov-Chain Monte-Carlo (MCMC) techniques [18], i.e., a random walk over a Markov chain (representing the input space) is performed. To further elaborate, the SA algorithm is depicted below. The goal of SA is as described in Equation 1, i.e., to find an input (x_0, u) which produces a counterexample to the formal property φ. In Algorithm 1, the evaluation of the robustness value of a given input condition and input signal is represented by the robustness function f.

Algorithm 1: Simulated annealing algorithm (based on [4])

```
Input  : $X_0$, $\mathbf{U}$, Robustness function $f$, Proposal
         scheme PS
Output: $x_0 \in X_0, u \in \mathbf{U}$

1 Generate an initial $x_0 \in X_0,\ u \in \mathbf{U}$;
2 while $f(x_0, u) \geq 0$ do
3     $(x'_0, u') \leftarrow$ PS$(x_0, u)$;
4     $\alpha \leftarrow \mathrm{e}^{-\beta\ (\ f(x'_0, u') - f(x_0, u)\ )}$;
5     $r \leftarrow$ UniformRandomReal(0,1);
6     if $r \leq \alpha$ then
7         $(x_0, u) \leftarrow (x'_0, u')$;
8     end
9 end
```

In Algorithm 1, SA starts by generating an input signal (x_0, u) at random (Line 1). Subsequently, the robustness of the generated input signal is evaluated by $f(x_0, u)$. Based on the outcome, the algorithm either proceeds or a counterexample is found, i.e., the generated (x_0, u) produces a falsifying trajectory (Line 2). In the case of a positive valuation, based on the current input sample (x_0, u), a next input signal (x'_0, u') is generated by the proposal scheme PS (Line 3). In the implementation, in fact, the domain of the input signals is discretized. A (user-defined) number of control points is distributed equidistantly over the time domain R of the input signals. For each such control point, a value is determined and an input signal is interpolated from these. S-TaLiRo also offers other possibilities, but these are not considered in this paper. More details about the proposal scheme used by S-TaLiRo are provided in Section IV-A. Note that the generation of a next input signal, which is based on the current input sample, transforms the otherwise ordinary Monte-Carlo technique into a MCMC.

Subsequently, the robustness value of the newly proposed (input) trajectory $f(x'_0, u')$ is computed and compared to the existing (best) robustness value. Based on this comparison, a ratio α is computed using an additional parameter β (Line 4). During the iterations, the β parameter is frequently updated in order to allow the SA algorithm to escape local minima. See [4] for more details on this mechanism.

Next, a (real) number r is sampled from a uniform distribution over [0,1] (Line 5), and subsequently compared to α (Line 6). Based on the outcome of this comparison, the new signal is either accepted (x_0, u become x'_0, u') or rejected (x_0, u remains) (Line 7). In case the new input signal has a smaller robustness value ($f(x'_0, u') - f(x_0, u) < 0$), then $\alpha > 1$ and the new signal is always accepted. In case the new input sample has a worse robustness value, then $0 < \alpha < 1$ and the new (input) signal is accepted with some non-zero probability (based on the generated r and $r \leq \alpha$

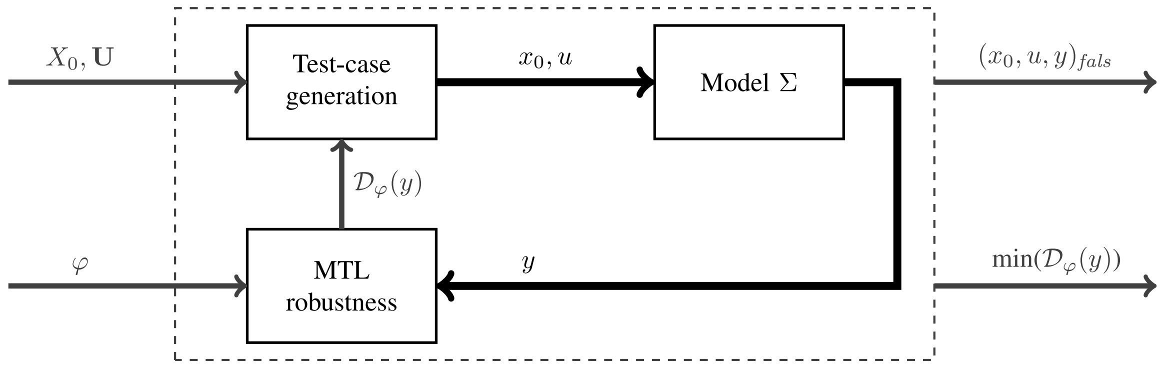

Fig. 1. MTL falsification solution (based on [4])

criteria). Intuitively, larger values of $f(x_0', u') - f(x_0, u)$ result in a smaller acceptance probability based on $r \leq \alpha$ (due to the negative exponent of e).

Since Algorithm 1 may not terminate, e.g., when the system under test indeed conforms to the formulae, the implementation has an upper bound on the number of iterations of the while loop construct.

III. Benchmarks for experimental evaluation

In order to evaluate our proposed extensions, two benchmark examples are adopted from [4], namely the automatic transmission (AT) [32] and the third order $\Delta - \Sigma$ modulator [11]. Both examples are Matlab Simulink models and the AT benchmark example is also proposed as a standard benchmark problem for hybrid system verification in [20].

A. Automatic transmission

In Figure 2, the AT benchmark example is shown. This Simulink model contains a combination of three Simulink subsystem blocks (containing multiple integrators), a number of one- or two dimensional look-up tables, and a Stateflow model which contains two sub-charts. Hence, because of the input and/or conditional dependencies of many of these components, such a relatively small model is already a challenge for formal verification methods/tools [4].

Based on a throttle input signal (U1), the AT benchmark example outputs a vehicle velocity (speed) and an engine RPM signal (RPM). Note that the brake input signal is intentionally not considered and left at zero. Moreover, in [4], for these two output signals, the following four MTL properties are formulated.

$$\varphi_1^{AT} = \neg(\ \Diamond\ \mathit{speed} \geq 120\ \wedge\ \Diamond\ \mathit{RPM} \geq 4500\ )$$
$$\varphi_2^{AT} = \neg\Diamond(\ \mathit{speed} \geq 120\ \wedge\ \Diamond_{[0,10]}\ \mathit{RPM} \geq 4500\ )$$
$$\varphi_3^{AT} = \neg\Diamond(\ \mathit{speed} \geq 120\ \wedge\ \Diamond_{[0,10]}\ \mathit{speed} \geq 125\ )$$
$$\varphi_4^{AT} = \neg\Diamond(\ \mathit{speed} \geq 120\ \wedge\ \Diamond_{[0,7.5]}\ \mathit{RPM} \geq 4500\ )$$

Formula φ_1^{AT} is the MTL formalization of the property that the car speed is always below 120 km/h or the rotational speed of the engine is always below 4500 rpm (the temporal operator $\Diamond$ stands for $\Diamond_{[0,\infty)}$, i.e., eventually now or in the future). Formula φ_2^{AT} expresses that the speed of the vehicle is always below 120 km/h or that the rotational speed of the engine is below 4500 rpm in the first 10 time units. The other formulae follow the same pattern as φ_2^{AT} and hence, are clear from that context.

For the experiments with the decoupled proposal scheme benchmark in Section IV, an additional input (U2) is added to the AT benchmark example of Figure 2. However, this input is disconnected from the rest of the model and does not influence any AT system dynamics, it solely expands the input space with a trivial second input signal. Note that the benchmark with the additional input is denoted AT*.

B. Third order $\Delta - \Sigma$ modulator

Originally, the third order $\Delta - \Sigma$ modulator is an example from the electrical domain with one input signal, three initial conditions, and three output signals. However, in this paper, the three initial conditions are added to the input space as (configurable) constants. In Figure 3, this (slightly) modified version of the third order $\Delta - \Sigma$ modulator is shown.

For the above example, the following falsification problem is formulated. Based on initial conditions in the set [-0.1,0.1], i.e., $U2, U3, U4 \in [-0.1, 0.1]$, and a configurable input signal range for $U1$, the system states $X1$, $X2$, and $X3$ should always stay within the interval [-1,1]. This is captured by the MTL formula $\Box(-1 \leq X1 \leq 1 \wedge -1 \leq X2 \leq 1 \wedge -1 \leq X3 \leq 1)$ (as before, $\Box$ stands for $\Box_{[0,\infty)}$, i.e., everywhere throughout the execution). The MTL falsification problem with this MTL formula for a model with a restricted input range for $U1$ to satisfy $U1 \in [-i, i]$ is denoted by $P_i^{\Delta-\Sigma}$.

In this paper, the following MTL falsification problems are used: $P_{0.45}^{\Delta-\Sigma}$, $P_{0.40}^{\Delta-\Sigma}$, $P_{0.35}^{\Delta-\Sigma}$, $P_{0.30}^{\Delta-\Sigma}$, $P_{0.25}^{\Delta-\Sigma}$, and $P_{0.20}^{\Delta-\Sigma}$. Note that the last three, more challenging, MTL falsification problems were not considered in [4].

IV. Decoupled proposal scheme

In this section, we extend the existing proposal scheme (as part of the solution of the MTL falsification problem in Section II-E) to deal with multiple inputs independently. First, in Section IV-A, the proposal scheme from [4] is introduced. Then, in Section IV-B, the extended version of the proposal scheme, called the decoupled proposal scheme, is introduced. In Section IV-C, the original proposal scheme

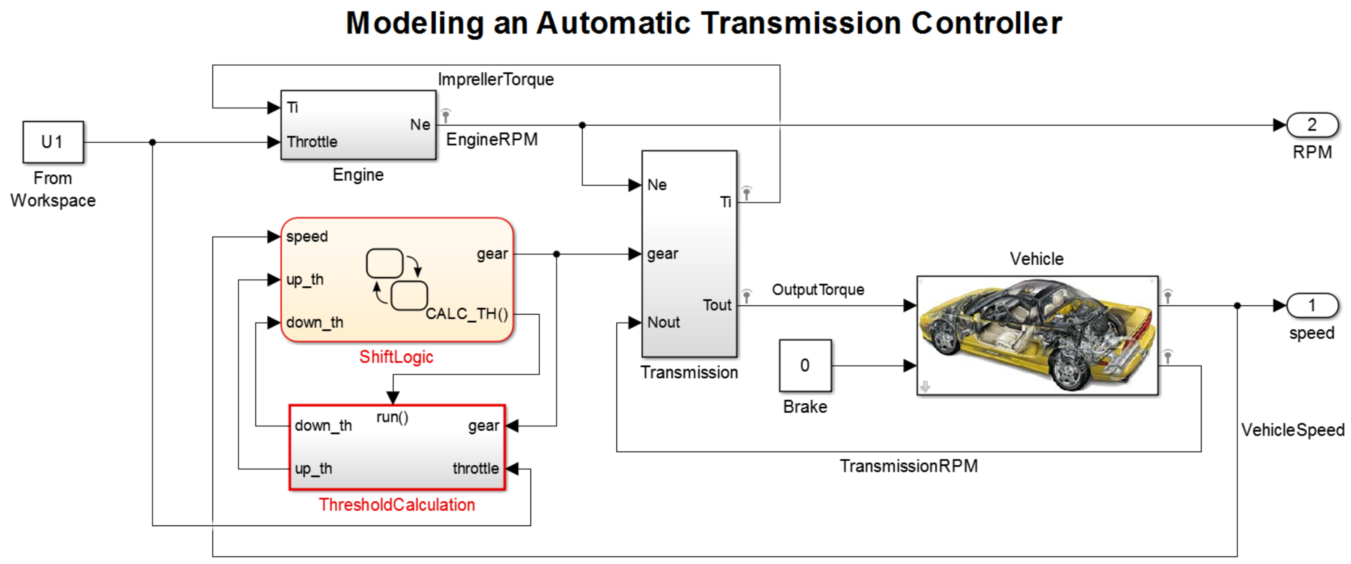

Fig. 2. Automatic transmission benchmark example

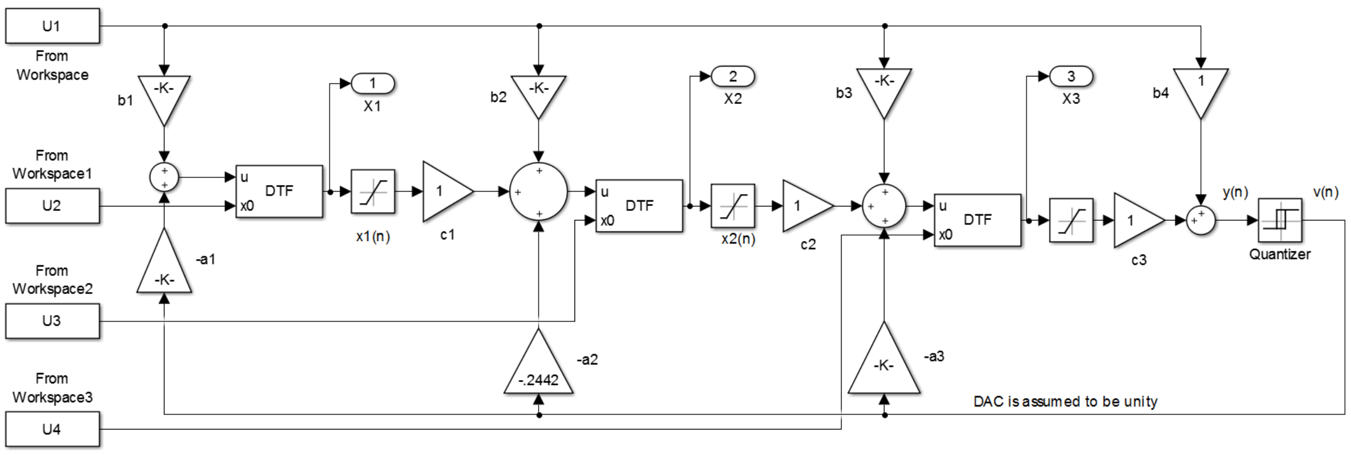

Fig. 3. Third order $\Delta - \Sigma$ modulator benchmark example

and our decoupled proposal scheme are compared on the benchmarks introduced in Section III.

A. Proposal scheme

In Section II-E, a proposal scheme is introduced which generates or samples a new initial condition and input signal (x'_0, u') based on the current values (x_0, u). In fact, because of this dependency, a Markov chain over the input-signal space is considered, i.e., every state of the Markov chain represents a (reachable) state of the input-signal space. Note that a discretized input-signal space has been assumed, and hence, the Markov chain is finite.

To construct a (x_0, u)-dependent proposal scheme, all $x'_0 \in X_0 \backslash \{x_0\}$ and $u' \in \mathbf{U} \backslash \{u\}$ are considered with some non-zero probability. Typically, such a next-state selection is realized by means of a random walk over the Markov chain. In addition, the PS should also converge to the distribution defined by the robustness function f to find the global minimizer. To have such convergence, three requirements regarding (the random walk over) the Markov chain are considered, namely detailed balance, irreducibility, and aperiodicity (see [4] for details).

Despite the above convergence requirements, it is relatively straightforward to construct a converging proposal scheme, e.g., a purely uniform sampling of the input space already indicates conformance to these requirements [4]. Typically, a PS based on a normal distribution centered at (x_0, u) provides a well-suited approach to solve the optimization problem and adheres to the convergence requirements [4]. Hence, such a technique is used in a so called hit-and-run proposal scheme, shown in Figure 4. Note that in the following explanation, for simplicity x_0 is not considered.

Firstly, some notation of Figure 4 is introduced. If a model Σ contains two input variables with input spaces U_1 and U_2, respectively, then the complete (model) input space U is given by $U_1 \times U_2$. Input signals for the different input variables all use the same time domain, which is a set of control points that lie within R. For each of these control

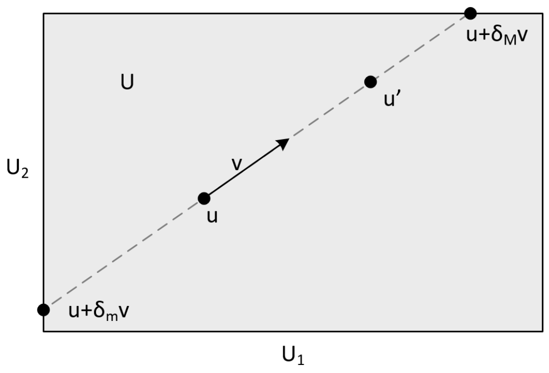

Fig. 4. Hit-and-run proposal scheme (based on [4])

points, for each of the input variables a value has to be provided.

For the following explanation, only one control point is considered. Note that in most practical cases, multiple control points are present and hence, the search space becomes higher dimensional than the discussed 2D example. Now, based on Figure 4, the following three phases/steps of the proposal scheme are elaborated. Note that in this paper, only convex input domains are considered, i.e., there are no dependencies between U_1 and U_2 constraining the overall input space U.

1) Firstly, from the value of the input variables $u = (u_1, u_2)$, based on a normal probability distribution, a random unit vector v is generated in the search space U. In practice, such a unit vector is generated by $v = \frac{h}{|h|^2}$, with h being a sampled vector in U (from a normal distribution).
2) Next, based on the generated unit vector v, the minimal and maximal displacement along this vector are identified, i.e., the smallest value δ_m and largest value δ_M are determined which keep $u + \delta v$ with $\delta \in [\delta_m, \delta_M]$ in the input space U.
3) Finally, based on a value s sampled from a uniform probability distribution over [-1,1] with zero mean, the sampled value s is multiplied with δ_m or δ_M (depending on the sign of s) to generate δ, resulting in the next input value u' (along the vector v): $u' = u+\delta v$ where $\delta = -s\delta_m$ if $s < 0$ and $\delta = s\delta_M$, otherwise.

Hence, the above proposal scheme is used to generate new input samples/signals. Note that this concept also applies for x_0, which only adds additional dimensions to the PS search space.

B. Decoupled proposal scheme

In Section IV-A, a hit-and-run proposal scheme is introduced which performs a (steered) random walk or search over a convex input space. Moreover, as explained in that subsection, a typical (convex) model input space is often higher dimensional. In this paper, based on benchmark examples (discussed later on) and experiences with the previously mentioned soccer wheelchair [5], it is observed that the performance/convergence of this proposal scheme is inadequate, i.e., higher dimensional input spaces heavily constrain any movement or search in the input space. To elaborate, when expanding Figure 4 with numerous dimensions, there is a (increasingly) high probability for a single dimension to result in a (very) small δ_m or δ_M. As a consequence also the displacements in all other dimensions are restricted by this small value. This results in a (strong) performance decrease, i.e., convergence of the hit-and-run proposal scheme tends to go to infinity (but remains bounded).

To improve the proposal scheme of S-TaLiRo, a decoupled version is proposed, i.e., every input signal is bounded individually (see [6] for the Matlab implementation). In Figure 5, this concept is visualized. Note that the notation is adopted from Section IV-A (with the assumption that for the figure, only one control point is considered). To briefly elaborate, by generating a vector v_i and applying only the corresponding displacement bounds (δ_{im} and δ_{iM}), heavy constrained input signals do not influence other (minor-constrained) input signals. For example, if u (with v_1 and v_2) is situated in the lower right corner, then v_1 is heavily constrained in its forward direction but that does not influence the forward movement of v_2.

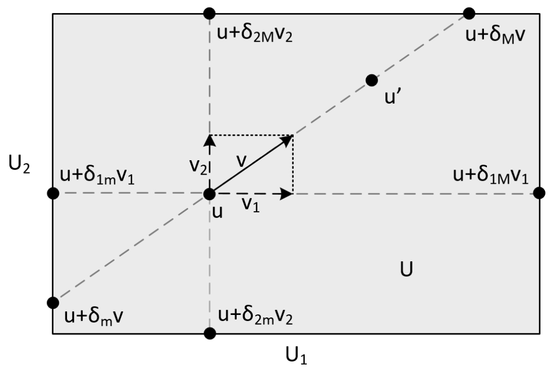

Fig. 5. Decoupled Proposal Scheme (DPS): illustrated for two dimensions

The three steps of the decoupled proposal scheme are exactly the same as those for the proposal scheme discussed before, with the only difference that these steps are performed for each input separately. The resulting u_i' are combined into the input signal u' as expected.

In Figure 5, two input variables are considered with only one control point. However, in case of multiple control points in U_i, v_1 and v_2 become higher dimensional, then U_1 and U_2 represent the set of all possible input values as a function of time (control points). Note that in this case, in Figure 5, the displacement bounds apply on the entire input signal, i.e., based on the considered control points of a signal u_i, the minimum and maximum of the set of all displacement bounds are considered.

In Section IV-A, three convergence criteria for the proposal scheme are briefly mentioned. Without going into detail, based on its similar structure to the original PS, the decoupled proposal scheme conforms to these requirements as well, i.e., detailed balance, irreducibility, and aperiodicity of the Markov chain (representing all states of the input space) are achieved.

C. Experimental results for the decoupled proposal scheme

To evaluate the performance differences between the original proposal scheme and the decoupled version, the two benchmarks introduced in Section III are considered.

Note that the AT* benchmark example has an additional input with the same range but different domain, i.e., 210 control points (for the additional input) versus 7 control points (throttle input). Hence, the resulting input space is 217-dimensional. Moreover, the input space of the third order modulator is 13-dimensional, i.e., 13 control points (10 for the input signal and 3 initial conditions).

In Tables I and II, both benchmark examples are run for 100 times in which every run iterates 1000 times through the MTL falsification solution (given a random generated initial falsification solution). As observed, for both benchmark examples, the decoupled proposal scheme shows a large performance increase.

In particular, the experiments with the AT* benchmark show a considerable performance increase when considering more difficult falsification problems (e.g., φ_3^{AT}) up to the point where no falsifiers can be found (φ_4^{AT}). Moreover, the decoupled proposal scheme almost achieves a maximal falsification rate for both φ_1^{AT} and φ_2^{AT}. The (original) proposal scheme achieves a much lower falsification rate for these simpler properties which is mainly due to the number of (algorithm) iterations which is insufficient to compensate for the long convergence times. In addition, no meaningful conclusions about timing can be made (other than that DPS is faster) due to large differences in falsification rates, i.e., falsification rates until 50-70% are heavily influenced (negatively) by non-falsifying runs with maximal run times, e.g., see φ_1^{AT*}.

TABLE I

DECOUPLED PROPOSAL SCHEME BENCHMARK FOR AT*

ψ	# Fals.		Robustness		Time (sec)	
	PS	DPS	PS	DPS	PS	DPS
φ_1^{AT*}	51%	99%	0.18 38.18 8452	15.442 15.442 0	0.046 37.2 210.6	0.041 3.47 33.98
φ_2^{AT*}	31%	95%	0.015 48.4 11683	0.95 8.22 46.48	0.047 45.1 157.6	0.404 8.02 122.04
φ_3^{AT*}	1%	54%	0.03 34.46 6400	0.0051 1 11.2	43.93 50.14 0.57	5.07 31.7 300.4
φ_4^{AT*}	0%	0%	0.17 24.75 3112	0.16 0.52 3	48.6 49.8 0.126	47.96 48.59 0.29

PS: proposal scheme, DPS: decoupled proposal scheme, # runs: 100, maximum # iterations: 1000, Matlab seed: 123. Legend: **# Fals.:** percentage of falsified runs, **Robustness:** minimum, average, and variance of the non-falsifying runs, **Time:** minimum, average, and variance of execution time per run

For the third order modulator benchmark example, a maximal performance increase of the DPS is detected for problems $P_{0.45}^{\Delta-\Sigma}$ until $P_{0.35}^{\Delta-\Sigma}$ in the form of a 100% falsification rate. Hence, the original benchmark in [4] is extended with three increasingly difficult falsification problems ($P_{0.30}^{\Delta-\Sigma}$ until $P_{0.20}^{\Delta-\Sigma}$) to detect the performance boundaries of the decoupled proposal scheme. As a result, the DPS showed the ability to detect falsifiers upto a (modulator) input bound of [-0.20,0.20]. Moreover, the (original) proposal scheme clearly is incapable to deal with such hard falsification problems (within 1000 iterations).

TABLE II

DECOUPLED PROPOSAL SCHEME BENCHMARK FOR THIRD ORDER MODULATOR

ψ	# Fals.		Robustness		Time (sec)	
	PS	DPS	PS	DPS	PS	DPS
$P_{0.45}^{\Delta-\Sigma}$	67%	100%	0.0017 0.033 0.0004	–	0.059 11.82 48.66	0.042 0.843 0.9
$P_{0.40}^{\Delta-\Sigma}$	47%	100%	0.0029 0.062 0.0008	–	0.064 16.97 58.52	0.043 1.31 1.81
$P_{0.35}^{\Delta-\Sigma}$	11%	100%	0.0031 0.079 0.0018	–	5.54 19.1 9.16	0.065 3.2 8.41
$P_{0.30}^{\Delta-\Sigma}$	2%	86%	0.0049 0.12 0.0029	0.0037 0.024 0.0003	21.49 22.95 0.112	0.155 9.23 45.8
$P_{0.25}^{\Delta-\Sigma}$	0%	13%	0.0318 0.16 0.0018	0.0071 0.065 0.0009	20.18 20.36 0.028	0.134 19.44 16.6
$P_{0.20}^{\Delta-\Sigma}$	0%	1%	0.1003 0.19 0.0015	0.0036 0.1023 0.001	20.12 20.54 0.116	20.11 20.56 0.13

PS: proposal scheme, DPS: decoupled proposal scheme, # runs: 100, maximum # iterations: 1000, Matlab seed: 123. Legend: **# Fals.:** percentage of falsified runs, **Robustness:** minimum, average, and variance of the non-falsifying runs, **Time:** minimum, average, and variance of execution time per run

V. INPUT-SIGNAL-SPACE OPTIMIZATION APPROACH FOR MTL FALSIFICATION

In Section II, the MTL falsification problem is denoted as an optimization problem to find a falsifying input pair (x_0^*, u^*) which triggers model output behaviour y^* that falsifies the MTL property φ. Moreover, to find such a falsifying pair, the set of initial conditions X_0 and input-signal space $\mathbf{U}$ are used as search spaces. To choose a suitable $\mathbf{U}$ for the above described optimization problem (besides guesswork), knowledge of the system or model Σ is required. Since the model-based testing step is performed in a black-box fashion, no or minimal model knowledge is used or even available. As a result, especially since the temporal aspect or domain of the input signals has severe consequences for its frequency content, this choice becomes non-trivial, i.e., specific model resonances or sensitivities to certain input stimuli (with higher chance to falsify) may not be triggered. In this section, the MTL falsification problem is adapted to include a form of input-signal-space optimization.

A. Input-signal space MTL falsification problem

To avoid using a specific (model) input-signal space $\mathbf{U}$ and to move towards a (pure) black-box model-based testing technique, the input-signal-space MTL falsification problem is proposed, i.e., besides x_0 and u, the input-signal space

$\mathbf{U} \subseteq U^R$ is included in the optimization problem as a search variable. As a result, based on computed robustness values, several time domains for the input-signal spaces can be considered during the solution process. Hence, based on Equation 1, the input-signal-space MTL falsification problem is denoted in Equation 2.

$$(x_0^*, u^*, \mathbf{U}^*) = \arg \min_{\mathbf{U} \subseteq U^R} \min_{x_0 \in X_0, u \in \mathbf{U}} \mathcal{D}_\varphi(\Delta_\Sigma(x_0, u)) \tag{2}$$

In the above equation, an extensive but feasible optimization problem is formulated, i.e., by expanding Equation 1 with a variable input-signal space $\mathbf{U}$, the search space is extended with another optimization dimension. As a result, the MTL falsification solution becomes computationally heavier, i.e., the required number of simulations to achieve convergence grows exponentially fast (compared to the original solution). However, the algorithm does gain the ability to look for an input-signal space $\mathbf{U}$ which triggers the least robust output behaviour. Now, based on the (original) MTL falsification solution of Figure 1, the input-signal-space solution is presented in Figure 6.

As observed in Figure 6, the original MTL falsification solution is included (unmodified) as part of a higher-level optimization loop. In fact, this (higher-level) loop comprises only one other block, namely the $\mathbf{U}$-space generation block. Connected to this block is the output of the (original) MTL robustness solution (min($\mathcal{D}_\varphi(y)$), see Figure 1). Now, based on an initially generated input space $(X_0, \mathbf{U})$, the original MTL falsification solution provides a minimal robustness value min($\mathcal{D}_\varphi(y)$) to the $\mathbf{U}$-space generation block. This value, together with the provided inputs X_0 and U^R, determines if it is favorable to stay close to the current (possibly less robust) input-signal space $\mathbf{U}$ or to move away from this (possibly very robust) input region. Inspired by Algorithm 1, this decision process in the $\mathbf{U}$-space generation block is further elaborated in Algorithm 2. Note that in this case, $g(\mathbf{U})$ represents the outcome of the (original) MTL falsification problem with $(X_0, \mathbf{U})$ as input and min($\mathcal{D}_\varphi(y)$) as output.

Algorithm 2: $\mathbf{U}$-space generation algorithm

```
Input : Input space $U^R$, Minimal robustness function
        $g$, Proposal scheme PS
Output: $\mathbf{U} \subseteq U^R$

1 Generate an initial $\mathbf{U} \subseteq U^R$;
2 while $g(\mathbf{U}) \geq 0$ do
3     $\mathbf{U}' \leftarrow$ PS($\mathbf{U}$);
4     $\alpha \leftarrow e^{-\lambda\ (\ g(\mathbf{U}') - g(\mathbf{U})\ )}$;
5     $r \leftarrow$ UniformRandomReal(0,1);
6     if $r \leq \alpha$ then
7         $\mathbf{U} \leftarrow \mathbf{U}'$;
8     end
9 end
```

As observed in Algorithm 2, similar to Algorithm 1, simulated annealing is used as optimization engine to search for the least robust input-signal space $\mathbf{U}$. Initially, an input-signal space $\mathbf{U}$ is generated/sampled (Line 1) and its (minimal) robustness value is evaluated by $g(\mathbf{U})$ (Line 2). Based on the outcome, the algorithm either proceeds or a counterexample is found, i.e., the generated input-signal space $\mathbf{U}$ contains an input signal u which triggers model output behaviour y that falsifies the MTL property φ. In case of a positive valuation, identical to Algorithm 1, an optimization process starts to find the minimal (and if possible negative) robustness value min(min($\mathcal{D}_\varphi(y)$)). For more details about this process, see Section II-E. Note that the parameter λ of Algorithm 2 is the equivalent of parameter β in Algorithm 1. Henceforth, the two optimization loops are referred to as the frequency optimization loop (extended, focusing on the time domain of the input signals) and the amplitude optimization loop (original, focusing on the input space of the input signals).

B. Experimental results for the input-signal-space optimization approach

To evaluate the input-signal-space problem, it cannot be directly compared to the (original) MTL falsification problem, i.e., to fix the temporal domain of the latter is a rather ill-constructed comparison. Hence, the original MTL falsification problem is performed using several randomly selected (temporal) domains in the form of different numbers of control points (CPs). As a result, based on an identical CP interval, the input-signal-space problem contains a frequency and amplitude optimization loop (SA) while the original MTL falsification problem contains an amplitude optimization loop with a uniform coverage of the CP range (UR).

Note that due to testing durations, the maximum simulation number for every benchmark is limited to one million model simulations and correspondingly subdivided, i.e., each optimization loop is granted 1000 model simulations which are further subdivided over runs versus iterations.

Note that in [6], the Matlab implementation of the input-signal-space solution (in the form of S-TaLiRo+) is available with which the benchmark is performed.

In Tables III and IV, the input-signal-space problem is shown with 25 runs of 40 iterations of the frequency optimization loop, and 5 runs with 200 iterations of the amplitude optimization loop. For the frequency loop, more runs are preferred (25) compared to the amplitude loop (5) where more iterations are preferred (200 versus 40) since the same frequency/input-signal space can be considered multiple times. For the AT benchmark example, the CP number ranges between 7 and 210 points while the modulator benchmark example varies between 2 and 65 points. In addition, this benchmark is based on the original proposal scheme.

As observed in Tables III and IV, for both the benchmarks, better falsification rates are achieved with the input-signal-space MTL optimization solution.

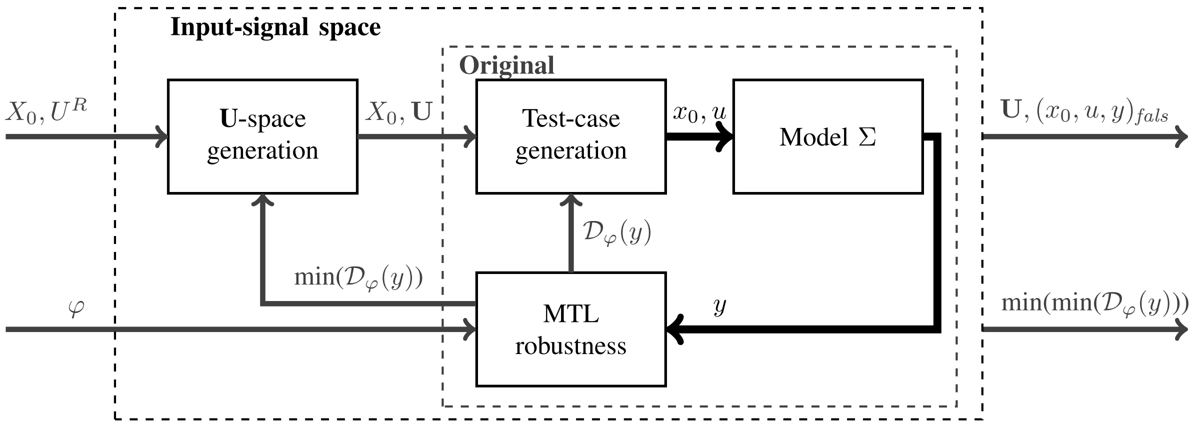

Fig. 6. Input-signal-space falsification problem solution

TABLE III

INPUT-SIGNAL-SPACE OPTIMIZATION BENCHMARK FOR AT

ψ	# Fals.		Robustness		Time (sec)	
	UR	SA	UR	SA	UR	SA
φ_1^{AT}	100%	100%	–	–	2.78 511 213250	2.18 325.57 66252
φ_2^{AT}	88%	100%	13.31 26.93 172.46	–	5.62 886.2 442310	8.96 430.73 68795
φ_3^{AT}	28%	100%	27.14 293.9 46631	–	97.89 1625 235340	243.26 706.21 99547
φ_4^{AT}	0%	0%	0.36 385 50152	0.17 0.18 0.00016	1719 1731 73	1718 1733 2693

UR: S-TaLiRo with uniform frequency coverage, SA: input-signal-space optimization solution (S-TaLiRo+), # frequency runs: 25, maximum # frequency iterations: 40, # amplitude runs: 5, maximum # amplitude iterations: 200, Matlab seed: 123. Legend: **# Fals.:** percentage of falsified frequency runs, **Robustness:** minimum, average, and variance of the non-falsifying runs, **Time:** minimum, average, variance of execution time per frequency run

TABLE IV

INPUT-SIGNAL-SPACE OPTIMIZATION BENCHMARK FOR THIRD ORDER MODULATOR

ψ	# Fals.		Robustness		Time (sec)	
	UR	SA	UR	SA	UR	SA
$P_{0.35}^{\Delta-\Sigma}$	100%	100%	–	–	10.65 172.74 30265	4.0827 169.6 30390
$P_{0.30}^{\Delta-\Sigma}$	56%	80%	0.0472 0.078 0.00	0.0122 0.0227 0.00	46.85 674.6 111090	0.99 520.7 81002
$P_{0.25}^{\Delta-\Sigma}$	24%	40%	0.0782 0.132 0.0011	0.0039 0.076 0.0019	54.83 698.83 45503	220.27 653.24 45819
$P_{0.20}^{\Delta-\Sigma}$	0%	16%	0.1363 0.18 0.0005	0.0261 0.114 0.0022	940.4 946.64 8.6	286.3 861.7 47295

UR: S-TaLiRo with uniform frequency coverage, SA: input-signal-space optimization solution (S-TaLiRo+), # frequency runs: 25, maximum # frequency iterations: 40, # amplitude runs: 5, maximum # amplitude iterations: 200, Matlab seed: 123. Legend: **# Fals.:** percentage of falsified frequency runs, **Robustness:** minimum, average, and variance of the non-falsifying runs, **Time:** minimum, average, variance of execution time per frequency run

Regarding the third order modulator benchmark example, the difference between the number of falsifiers found is rather small. Analysis showed that counterexamples where found over the whole temporal/CP range and hence, since the UR combination covers the temporal domain uniformly, a higher falsification rate is achieved compared to the AT benchmark example (which only contains a small falsifying CP range). As a result, it is demonstrated that when a particular region of interest needs to be located, in this case less robust input-space regions, the input-signal-space MTL optimization solution performs exceptionally well. In case such a specific region does not exist and a very wide spectrum is 'interesting', the use of a steered method becomes less efficient. However, when considering the robustness values in Table IV, in all cases, the input-signal-space MTL optimization solution reaches smaller (robustness) values indicating weaker system behaviour (closer to the falsifying bounds). Note that for designers of CPSs or mechatronic systems, such behaviour can still be of great importance.

VI. EXPERIMENTAL EVALUATION OF COMBINED DECOUPLED PROPOSAL SCHEME AND INPUT-SIGNAL-SPACE OPTIMIZATION

In Table V, some experimental results of the third order modulator benchmark example with the input-signal-space optimization approach combined with the decoupled proposal scheme are demonstrated. As observed, compared to the results of Tables II and IV, more complex MTL falsification problems become (easily) feasible, i.e., a 100% falsification rate is achieved until $P_{0.15}^{\Delta-\Sigma}$ (compared to a 1% and 16% falsification rate for $P_{0.20}^{\Delta-\Sigma}$ for the separate improvements). Hence, for the benchmarks studied in this paper, combining the input-signal-space optimization approach and decoupled proposal scheme provides another MTL falsification performance increase.

VII. CONCLUDING REMARKS

In this work, we proposed two improvements upon the MTL falsification method [4], namely an input-signal-space

TABLE V

INPUT-SIGNAL-SPACE OPTIMIZATION AND DECOUPLED PROPOSAL SCHEME BENCHMARK FOR THIRD ORDER MODULATOR

ψ	# Fals.	Robustness	Time (sec)
	SA+ DPS	SA+ DPS	SA+ DPS
$P^{\Delta-\Sigma}_{0.20}$	100%	−	1.5531 91.6 14807
$P^{\Delta-\Sigma}_{0.15}$	100%	−	0.7938 178.53 20570
$P^{\Delta-\Sigma}_{0.10}$	0%	0.0613 0.141 0.0015	960.35 972.5 111.5

# frequency runs: 25, maximum # frequency iterations: 40, # amplitude runs: 5, maximum # amplitude iterations: 200, Matlab seed: 123. Legend: **# Fals.:** percentage of falsified frequency runs, **Robustness:** minimum, average, and variance of the non-falsifying runs, **Time:** minimum, average, and variance of execution time per frequency run

optimization approach and a decoupled proposal scheme for the simulated annealing optimization engine. These two improvements were implemented as an extension of S-Taliro and based on experimental benchmarks, proved to work exceptionally well, i.e., based on a broad range of control points for the input space, the weakest (less robust) control point region was consistently located.

In this work, simulated annealing is selected as optimization approach for the (input-signal-space) MTL falsification solution. However, as mentioned in Section II-E, several other options are available. Hence, a future research direction is a (performance) comparison between optimization algorithms for the type of optimization problem put forward in this work. We would like to investigate our approach for arbitrary polytopes as input-spaces and whether for such input-spaces, the probability of sampling points is non-zero along the lines of [3].

Acknowledgements. We are grateful to Houssam Abbas and Georgios Fainekos for their comments on a draft of this paper.

REFERENCES

[1] E. Aarts and J. Korst. *Simulated annealing and Boltzmann machines: A stochastic approach to combinatorial optimization and neural computing*. Wiley, 1988.

[2] H. Abbas and G. E. Fainekos. Computing descent direction of MTL robustness for non-linear systems. In *Proc. ACC'13*, pages 4405-4410, IEEE, 2013.

[3] H. Abbas and G. E. Fainekos. Convergence proofs for Simulated Annealing falsification of safety properties. In *Proc. Allerton'12* pages 1594-1601, IEEE, 2012.

[4] H. Abbas, G. E. Fainekos, S. Sankaranarayanan, F. Ivančić, and A. Gupta. Probabilistic temporal logic falsification of cyber-physical systems. *ACM TECS*, 12(2s):95, 2013.

[5] A. Aerts. Model-based design and testing of mechatronic systems: an industrial case study. Master's thesis, CST report 2016.052, TU Eindhoven, 2016.

[6] A. Aerts. S-TaLiRo+. Online SVN repository, https://svn.riouxsvn.com/staliroplus, Revision 2, 2016.

[7] A. Aerts, M. R. Mousavi, and M. A. Reniers. Model-based testing of cyber-physical systems. In *Cyber-Physical Systems: Foundations, Principles and Applications*, chapter 19. Elsevier, 2017.

[8] Y. Annpureddy, C. Liu, G. E. Fainekos, and S. Sankaranarayanan. S-TaLiRo: A tool for temporal logic falsification for hybrid systems. In *Proc. TACAS'11*. Springer, 2011.

[9] C. G. Cassandras and S. Lafortune. *Introduction to discrete event systems*. Springer, 2008.

[10] S. Chib and E. Greenberg. Understanding the Metropolis-Hastings algorithm. *The American Statistician*, 49(4):327–335, 1995.

[11] T. Dang, A. Donzé, and O. Maler. Verification of analog and mixed-signal circuits using hybrid system techniques. In *Proc. FMCAD'04*, volume 3312, pages 21–36. Springer, 2004.

[12] A. Donzé and O. Maler. Robust satisfaction of temporal logic over real-valued signals. In *Proc. FORMATS'10*, volume 6246 of *LNCS*, pages 92–106. Springer, 2010.

[13] M. Dorigo and T. Stützle. Ant colony optimization: overview and recent advances. *Tech. Rep., Universite Libre de Bruxelles*, 2009.

[14] K. A. Dowsland and J. M. Thompson. Simulated annealing. In *Handbook of Natural Computing*, pages 1623–1655. Springer, 2012.

[15] T. Dreossi, T. Dang, A. Donzé, J. Kapinski, X. Jin, and J. V. Deshmukh. Efficient guiding strategies for testing of temporal properties of hybrid systems. In *Proc. NFM'15*, pages 127–142. Springer, 2015.

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

[17] G. E. Fainekos, S. Sankaranarayanan, K. Ueda, and H. Yazarel. Verification of automotive control applications using S-TaLiRo. In *Proc. ACC'12*, pages 3567–3572. IEEE, 2012.

[18] W. R. Gilks. *Markov chain Monte Carlo*. Wiley, 2005.

[19] H.-G. Gross. *Component-based software testing with UML*. Springer, 2005.

[20] B. Hoxha, H. Abbas, and G. Fainekos. Benchmarks for temporal logic requirements for automotive systems. In *Proc. ARCH@CPSWeek 2015 and 2015*, pages 25–30, 2015.

[21] B. Hoxha, H. Bach, H. Abbas, A. Dokhanchi, Y. Kobayashi, and G. E. Fainekos. Towards formal specification visualization for testing and monitoring of cyber-physical systems. In *Proc. DIFTS'14*, 2014.

[22] M. H. Kalos and P. A. Whitlock. *Monte Carlo methods*. Wiley, 2008.

[23] N. Khakpour and M. R. Mousavi. Notions of conformance testing for cyber-physical systems: Overview and roadmap. In *Proc. CONCUR'15*, volume 42 of *LIPIcs Proc.*, pages 18–40. Schloss Dagstuhl, 2015.

[24] R. Koymans. Specifying real-time properties with metric temporal logic. *Real-time systems*, 2(4):255–299, 1990.

[25] O. Maler and D. Nickovic. Monitoring temporal properties of continuous signals. In *Proc. FORMATS'04*, vol. 3253 of *LNCS*, pages 152–166. Springer, 2004.

[26] M. Mohaqeqi and M. Mousavi. Sound Test-Suites for Cyber-Physical Systems. In *Proc. TASE'16*. IEEE CS, 2016.

[27] J. Ouaknine and J. Worrell. Some recent results in metric temporal logic. In *Proc. FORMATS'08*, vol. 5215 of *LNCS*, pages 1–13, 2008.

[28] A. Pnueli. The temporal logic of programs. In *Proc. FOCS'77*, pages 46–57. IEEE, 1977.

[29] R. Y. Rubinstein and D. P. Kroese. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning*. Springer, 2013.

[30] A. C. Uselton and S. A. Smolka. A compositional semantics for Statecharts using labeled transition systems. *Proc. CONCUR'94*, pages 2–17. Springer, 1994.

[31] M. Utting and B. Legeard. *Practical model-based testing: a tools approach*. Morgan Kaufmann, 2010.

[32] Q. Zhao, B. H. Krogh, and P. Hubbard. Generating test inputs for embedded control systems. *IEEE Control Systems*, 23(4):49–57, 2003.