Robustness-Guided Temporal Logic Testing and Verification for Stochastic Cyber-Physical Systems

Houssam Abbas*, Bardh Hoxha*, Georgios Fainekos
Arizona State University
Tempe, Arizona, USA
Email: {hyabbas, bhoxha, fainekos}@asu.edu

Koichi Ueda
Toyota Technical Center
Gardena, CA, USA
Email: koichi.ueda@tema.toyota.com

***Abstract*—We present a framework for automatic specification-guided testing for Stochastic Cyber-Physical Systems (SCPS). The framework utilizes the theory of robustness of Metric Temporal Logic (MTL) specifications to quantify how robustly an SCPS satisfies a specification in MTL. The goal of the testing framework is to detect system operating conditions that cause the system to exhibit the worst expected specification robustness. The resulting expected robustness minimization problem is solved using Markov chain Monte Carlo algorithms. This also allows us to use finite-time guarantees, which quantify the quality of the solution after a finite number of simulations. In a Model-Based Design (MBD) process, our framework can be combined with Statistical Model Checking (SMC). Finally, we present a case study on a high fidelity engine model where the goal is to verify EPA standards on the air-to-fuel ratio.**

I. Introduction

Stochasticity is inherent in many Cyber-Physical Systems (CPS). It might arise as the result of actuator inaccuracies, sensor readings, rate of arrivals, component failure rates, etc. One important question is how such random phenomena can affect the functional correctness properties of a CPS. For instance, in a typical Model-Based Design (MBD) process a deterministic CPS model may be first developed which satisfies a set of design criteria and correctness requirements. Then, the challenge that arises is whether the more accurate stochastic variant of the CPS still satisfies the same properties and to what degree. In other cases, the development of a Stochastic CPS (SCPS) is required as part of the initial modeling process and the specifications to be verified directly on the model.

In either case, high fidelity models, e.g, internal combustion or hybrid engine models, pose substantial challenges in the functional verification process. For instance, complex engine models capture highly nonlinear physical phenomena (e.g., combustion dynamics, fuel injection dynamics, heat transfer, exhaust dynamics, etc), different modes of operation (e.g., different gears) and, on top of that, complex control algorithms (e.g., adaptive control laws for variable valve timing, powertrain control, etc). On the other hand, the functional specifications to be verified might be complex themselves, e.g., in Metric Temporal Logic (MTL) [9], [23].

In the past, to address this challenge, Statistical Model Checking (SMC) for SCPS was proposed [25], [8]. In brief, given a probability distribution on the parameters of the SCPS and a specification φ in a temporal logic, SMC computes a probability that φ holds on the SCPS along with confidence intervals. The main advantage of such methods is that in principle the underlying system can be arbitrarily complex (i.e., large number of variables, nonlinearities, complex control logic, etc).

We argue that for SCPS the probability of success of a formal specification φ might not be sufficient in all applications. For example, consider a simple, but important requirement for car manufacturers: the normalized air-to-fuel (A/F) ratio should always be within the appropriate limits (e.g., 1 ± 0.1). We need to be able to distinguish between designs that satisfy this requirement to varying degrees. All else being equal, a system design for which the worst expected behavior stays very close to 1 and the probability of failure is low should be preferred over all other correct designs. The issue of system variance around this minimum is also addressed.

Furthermore, SMC methods require the probability distribution over input space which may not be readily available. For instance, in the case of an engine with the throttle and the gear selection as inputs, either a specific driving scenario must be assessed or somehow a probability distribution over driver patterns must be provided. Albeit such model analysis are extremely useful, we argue that in many cases we do not necessarily need to analyze the system behavior under typical input scenarios, but also to discover the inputs that induce the worst system behavior - in the expected sense. For example, in the case of the A/F ratio verification, we would like to discover the throttle input and gear sequence that causes the worst expected deviation from the ideal ratio 1.

This paper presents a solution to the two aforementioned challenges. By utilizing the notion of robustness for Metric Temporal Logic (MTL) specifications [11], we can quantify how robustly a system trajectory satisfies an MTL specification. Large positive values mean that the system is robustly correct, while negative values imply falsification of the specification. Thus, the verification problem for SCPS reduces to finding a global minimizer for the expected temporal logic robustness. If the expected MTL robustness on a global minimizer is positive, then the system is correct in the expected sense. Moreover, statistics can be collected in order to assess the probability of satisfaction.

*The authors equally contributed to the work.

In order to solve the MTL verification problem for SCPS, we adopt and adapt some recent results in stochastic optimization [15] that provide finite time guarantees: how far are we from a global minimizer after a given number of samples?

One advantage of our proposed framework is that even if verification with the desired probabilistic guarantees cannot be achieved, then our approach reduces to a best effort automatic test generation scheme. Due to the underlying stochastic optimization algorithm [15], the test generation process is guided by the MTL robustness metric. Thus, we refer to the latter as *Robustness-Guided Temporal Logic Testing*(RGTLT). In this case, our testing framework can handle models of arbitrary complexity.

In this paper, we also present how our framework would be utilized in an MBD process. Finally, the paper concludes with a case study on a high fidelity SimuQuest [19] engine model. The work presented in this paper is readily available for use through our Matlab toolbox S-TaLiRo [5], [1].

II. Problem Formulation

In the following, $\mathbb{R}$ is the set of real numbers, $\mathbb{R}_+$ the set of positive reals, $\mathbb{Q}$ the set of rational numbers and $\mathbb{N}$ the set of natural numbers (including zero). The extended real number line is denoted by $\overline{\mathbb{R}} = \mathbb{R} \cup \{\pm\infty\}$. Given two sets A and B, B^A is the set of all functions from A to B, i.e., for any $f \in B^A$ we have $f : A \to B$.

We fix $T = N\Delta t$ to be the maximum simulation time for the system, where $N \in \mathbb{N}$ and $\Delta t \in \mathbb{Q}$ is the sampling step. We denote by $\mathbb{T} \subset [0, N] \subseteq \mathbb{N}$ a finite simulation time domain. We view a system Σ as a mapping from a compact set of *initial operating conditions* $X_0 \subset \mathbb{R}^{n_x}$ and discrete-time *input signals* $\mathfrak{u} \in U^{\mathbb{T}}$ to discrete-time *output signals* in $Y^{\mathbb{T}}$. Here, $U \subset \mathbb{R}^{n_u}$ is a compact set of possible input values and $Y \subset \mathbb{R}^{n_y}$ is the set of output values (output space). Discrete system variables like counters and flags are modeled as integers, so X_0 and U can be "hybrid" spaces.

We impose the following assumptions / restrictions on the systems that we consider:

1) The input signals (if any) must be parameterizable using a finite number of parameters in space and time. That is, there exists a function A such that for any $\mathfrak{u} \in U^{\mathbb{T}}$, there exists a parameter vector $\lambda = [\lambda_1 \dots \lambda_m, \lambda_{m+1} \dots \lambda_{2m}]^T \in \Lambda$, where
$$\Lambda = U^m \times \{\tau \in \mathbb{R}_+^m \mid 0 = \tau_1 \leq \tau_2 \leq \dots \leq \tau_m = T\}$$
is a compact set with $m << N$, such that for all t in the support of $\mathfrak{u}$, $\mathfrak{u}(t) = A(\lambda)(t)$.
2) The output space Y is equipped with a metric $\mathbf{d}$.

For brevity, we define
$$\Theta \triangleq X_0 \times \Lambda \tag{1}$$
This will be referred to as the search space, and $\theta \in \Theta$ will be referred to as the search variable, or the decision variable.

Because we work with stochastic systems, the output signals are modeled as Y-valued stochastic processes with sample paths in $Y^{\mathbb{T}}$. Specifically, let $(\Omega, \mathcal{A}, P)$ be a probability space with probability measure P. The random events $\omega \in \Omega$ model the sources of randomness in the SCPS Σ. Then, corresponding to every decision $\theta = (x_0, \lambda)$ about initial conditions and input signal, the output of the system is a discrete-time stochastic process parametrized by θ:
$$\mathbf{Y} : (t, \omega; \theta) \in \mathbb{T} \times \Omega \times \Theta \mapsto \mathbf{Y}(t, \omega; \theta) \in Y$$

Our high level goal is to explore properties that the system Σ satisfies by observing its response (i.e. its sample paths) to particular input signals and initial conditions. We assume that the system designer can formalize the system properties in Metric Temporal Logic (MTL) [14]. Once the stochastic model is developed, the engineers need to verify that the system meets a specification φ expressed in MTL. This can be done using the notion of *trajectory robustness*, formally introduced in [10]. For convenience, a brief formal introduction is also made in the technical report [4]. For this papers' purposes, we treat robustness as a functional which takes in a sample path of $\mathbf{Y}$, and produces a robustness value:
$$\rho_\varphi : \mathbf{Y}(\cdot, \omega; \theta) \mapsto \rho_\varphi(\mathbf{Y}(\cdot, \omega; \theta)) \equiv \rho_\varphi(\omega, \theta) \in \overline{\mathbb{R}}$$
This value indicates how well the system satisfies (or falsifies) the given specification φ. A positive(resp. negative) robustness means that the system is satisfied(resp. falsified) for a particular trajectory. The specification is implicit in all that follows, and will be dropped from the notation.

If we think of a random variable ρ over $(\Omega, \mathcal{A}, P)$ induced by the functional ρ, we see that its distribution is parametrized by the decision θ:
$$R_\rho(z; \theta) = P(\{\omega \in \Omega \mid \rho(\omega, \theta) \leq z\}) \tag{2}$$

Formally, in this work, we solve the following problem.

Problem 1 ($\mathbb{E}$RGMC Problem): Take an SCPS Σ, an MTL specification φ, a test duration $T > 0$. Define the expected robustness of the SCPS w.r.t. φ as
$$U : \Theta \to \mathbb{R}$$
$$U(\theta) = \mathbb{E}_P[\rho(\omega, \theta)] = \int_\Omega \rho(\omega, \theta) dP(\omega) \tag{3}$$
Compute the minimum expected robustness of the system with respect to the MTL specification:
$$U_* = \inf\{U(\theta) | \theta \in \Theta\}$$
An overview of our proposed solution to Problem 1 appears in Fig. 1. The sampler produces a point x_0 from the set of initial conditions and a vector of parameters λ that characterize the control input signal $\mathfrak{u}$. These are passed to the system simulator which returns a set of J sample paths (a.k.a. output trajectories) $y_1, \dots, y_J$, where J is computed after the considerations in Section IV. The set of trajectories is then analyzed by the MTL robustness analyzer which returns a vector of robustness values for each trace representing the best estimate for the parameter found so far. In turn, the robustness scores computed are used by the stochastic sampler to decide on a next input to analyze. The process terminates after a maximum number of tests.

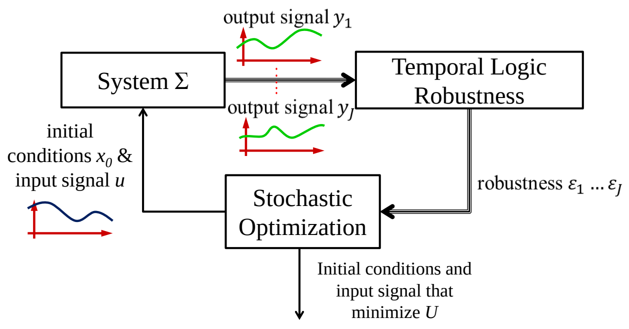
output signal y_1
System Σ
Temporal Logic Robustness
output signal y_J
initial conditions x_0 & input signal u
robustness $\varepsilon_1 \dots \varepsilon_J$
Stochastic Optimization
Initial conditions and input signal that minimize U

Fig. 1. Overview of the solution to finding the θ on the search space that corresponds to the minimum expected robustness value for a MTL specification.

III. Stochastic Optimization Algorithm

In this section we present the optimization algorithm we use to solve Problem 1. Algorithm 1, which we call $\mathbb{E}$RGMC, is a Simulated Annealing (SA) algorithm first proposed in [16], applicable to expected value objectives. For reasons on which we will elaborate in the next section, we use Algorithm 1 to maximize the following modified objective function: $V(\theta) \triangleq \mathbb{E}_P[1/(1+e^{\rho(\theta)})]$. By maximizing V, we minimize U. For completeness, a proof is included in the technical report[4]. Thus, $\mathbb{E}$RGMC is a maximization algorithm.

Algorithm 1 $\mathbb{E}$RGMC - Expected Robustness Guided Monte Carlo Algorithm

```
Require: Search space $\Theta$, number of tests $K$, number of independent extractions $J$ , scaling factor $b > 0$, parameter $\delta > 0$
    Draw a random sample $\theta_0 \in \Theta$.
    for $i = 0$ to $K - 1$ do
        Draw a random candidate $\tilde{\theta}_{i+1}$ according to the symmetric proposal kernel $Q$.
        Draw $J$ independent extractions $\{\rho(\omega_{i+1}^{(j)}, \tilde{\theta}_{i+1}),\ j = 1 \dots, J\}$ by simulating the system $J$ times and computing the robustness of the $J$ sample paths.
        Scale all $J$ robustness values: $\bar{\rho} = 1/(1 + e^{\rho/b})$
        Calculate the acceptance probability
            $$a = \frac{\prod_{j=1}^{J}[\bar{\rho}(\omega_{i+1}^{(j)}, \tilde{\theta}_{i+1}) + \delta]}{\prod_{j=1}^{J}[\bar{\rho}(\omega_{i}^{(j)}, \theta_{i}) + \delta]}$$
        $r$ = uniform random number between 0 and 1
        if $(r < a)$ then
            $\theta_{i+1} = \tilde{\theta}_{i+1}$
        else
            $\theta_{i+1} = \tilde{\theta}_{i}$
        end if
    end for
    return $\theta_K$
```

Algorithm 1 iterates a Markov transition kernel with stationary (or 'target') distribution [16]

$$\pi(d\theta; J, \delta) \propto (V(\theta) + \delta)^J \mu(d\theta) \tag{4}$$

where μ is the Lebesgue measure. π is concentrated around the global maximizers of V, so with a correspondingly high probability, sampling from π yields points with large objective value. Therefore, as the distribution of the chain P_{θ_k} converges to π, the generated samples $\theta_k, \theta_{k+1}, \dots$ have V-values close to the global maximum with high probability. The next section quantifies this statement by providing convergence rate bounds.

Given that every iteration of Algorithm 1 generates J extractions of the random robustness ρ, this data can be used to generate a point estimate of the robustness variance $var(\theta)$ at the decision θ. To obtain a good quality estimate without doing further costly simulations, bootstrapping can be used to re-sample the obtained J values $\rho_1 \dots \rho_J$. The resulting variance estimate is valuable feedback to the designer, since a positive but small average robustness with a large variance indicates a probability of failing the specification.

IV. Finite-time guarantees for $\mathbb{E}$RGMC

The sequence of samples generated by Simulated Annealing (SA) is known to converge in probability to the global minimum of the objective function. However, this does not tell us how fast convergence happens, nor how far we are from the optimum after a given number of iterations. This sort of finite-time guarantee is important in practice, since it helps to determine the cut-off point at which to terminate the algorithm's run. In this section, we apply the results in [15] to provide finite-time guarantees for SA solving $\mathbb{E}$RGMC in general SCPSs. The results of [15] provide finite-time guarantees for Algorithm 1 under rather simple and unrestrictive conditions. Recall that $\Theta = X_0 \times \Lambda$ is the search space.

Assumption 1: Θ has a finite Lebesgue measure. The objective function is well-defined point-wise, measurable and bounded between 0 and 1.

In general, our U is not bounded between 0 and 1, and its bounds (if any) are a priori unknown. So we used a logistic map with scaling factor $b > 0$ to enforce this requirement: $\rho \in \mathbb{R} \mapsto \bar{\rho} \triangleq \frac{1}{1+e^{\rho/b}} \in [0, 1]$. The objective function passed to Algorithm 1 is then effectively $V(\theta) = \mathbb{E}[\bar{\rho}(\theta)]$, which can be seen to satisfy Assumption 1. Note that maximizing V minimizes U. The first guarantee uses a set which contains 'most' near-maximizers:

Definition 1: Let $\alpha \in (0, 1]$ and $\epsilon \in [0, 1]$. Then the approximate domain optimizer with imprecision ϵ and residual domain α is given by

$$\Theta(\epsilon, \alpha) = \{\theta \in \Theta \mid \mu(\{\theta' \ s.t.\ V(\theta') > V(\theta) + \epsilon\}) \leq \alpha\mu(\Theta)\}$$

Loosely speaking, there aren't many points θ' with objective value much better than the value of the points in $\Theta(\epsilon, \alpha)$. ϵ controls how much is 'much better', and α controls how many is 'not many'. Therefore, if we can generate samples from $\Theta(\epsilon, \alpha)$ with small ϵ and α, this increases the probability of finding points such that $V(\theta) > V^* - \epsilon$. Recall that π is the target distribution of Algorithm 1 (see (4)).

Proposition 1: [15, Prop. 3] Let $\alpha \in (0, 1]$, $\epsilon \in [0, 1]$, and $\delta > 0$. Assume J satisfies

$$J \geq \frac{1 + \epsilon + \delta}{\epsilon} \left[\log \frac{\sigma}{1 - \sigma} + \log \frac{1}{\alpha} + 2 \log \frac{1 + \delta}{\delta} \right]$$

Let P_{θ_k} be the distribution of the MCMC chain at step k. Then $P_{\theta_k}(\Theta(\epsilon,\alpha);J,\delta) \geq \sigma - \|P_{\theta_k} - \pi(\cdot;J,\delta)\|_{TV}$ where $\|\cdot\|_{TV}$ is the Total Variation of the distribution. Because P_{θ_k} converges to the stationary distribution π in total variation, convergence rates automatically carry over from TV convergence to SA convergence.

In general, U need not be Lipschitz continuous. If we add that assumption, a stronger results holds as asserted in Prop. 2.

Assumption 2: Θ is compact and is contained in a ball of radius R. U is Lipschitz continuous with constant L.

Proposition 2: Let assumptions 1 and 2 hold. Let $\alpha \in (0,1]$, $\epsilon \in (0,1]$, $\sigma \in (0,1)$, and $\delta > 0$. Define the set of value optimizers $\Theta^*(\epsilon) = \{\theta \in \Theta \mid \forall \theta' \in \Theta, V(\theta') \leq V(\theta) + \epsilon\}$ If J satisfies $J \geq \frac{1+\epsilon+\delta}{\epsilon}\left[\log\frac{\sigma}{1-\sigma} + n\log\frac{LR}{\epsilon} + 2\log\frac{1+\delta}{\delta}\right]$, then $P_{\theta_k}(\Theta^*(\epsilon);J,\delta) \geq \sigma - \|P_{\theta_k} - \pi(\cdot;J,\delta)\|_{TV}$.

V. Design Process using $\mathbb{E}$RGMC

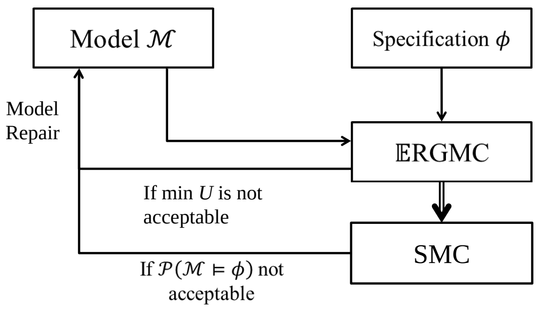
Model $\mathcal{M}$
Specification ϕ
Model Repair
$\mathbb{E}$RGMC
If min U is not acceptable
SMC
If $\mathcal{P}(\mathcal{M} \models \phi)$ not acceptable

Fig. 2. Overview of the model based design process using $\mathbb{E}$RGMC.

Our framework for Model Based design verification is illustrated in Fig 2. Once a model is developed, we devise MTL formulae $\varphi_1 ... \varphi_n$ for the properties that the system should satisfy. For every φ, we run $\mathbb{E}$RGMC and find the regions of the search space with the minimum expected robustness value. Call this region Θ_φ. If the minimum expected robustness is too low (or worse, negative, indicating that the specification has been violated), we can go back and make modifications and repair the model. On the other hand, if we are satisfied with the minimum expected robustness, we run statistical model checking (SMC) methods, e.g. [25], to estimate the probability that the model will satisfy the specification when operating in Θ_φ. This is important because even if the minimum expected robustness is positive, the probability of satisfying φ can still be low, due to the variance in the system behavior around the minimum. If the probability calculated by SMC is too low, we go back to model development and repair the model. If the probability level meets the requirements, we accept the model. Therefore, $\mathbb{E}$RGMC and SMC complement each other, each providing important information to the model developers. This way we calculate the probability of success in the "worst" region in the input space, in the average sense.

VI. Experimental Results

We present a case study of a high fidelity engine model. Two examples from the automotive industry are presented in the technical report [4]. We provide a description of the system and illustrate the advantages of using the proposed framework.

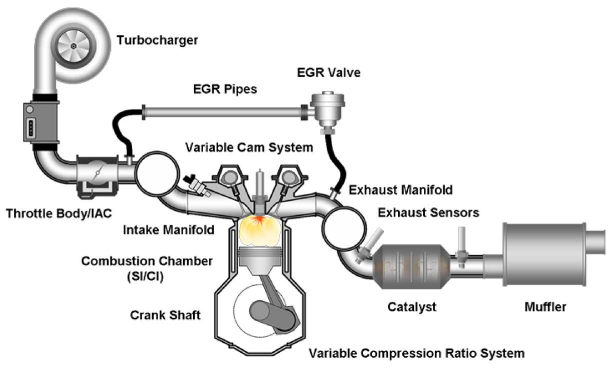

Fig. 3. Case Study 1. SimuQuest Enginuity model components. Used with permission, ©SimuQuest[19].

The following features were added to S-TaLiRo to enable and facilitate the simulations. First, support for parallel simulations was added. Second, the search space for input parameters was expanded: previously, the input signal was parametrized using an interpolation function with m signal levels evenly distributed over the simulation interval $[0,T]$. That is, $\mathfrak{u}$ was obtained by interpolating the sequence $((u_1,0),(u_2,\frac{T}{m-2}),(u_3,\frac{2T}{m-2}),\ldots,(u_m,T)) \in (U\times[0,T])^m$. Now the control times are free, so $\mathfrak{u}$ is obtained by interpolating $((u_1,0),(u_2,\tau(2)),(u_3,\tau(3)),\ldots,(u_m,T))$, and the vector τ is a free search variable subject to the constraint $\tau(1) = 0 < \tau(2) < \ldots < \tau(m) = T$. The features were easily incorporated in the modular architecture of S-TaLiRo.

Most of the experimental results presented were generated using the A2C2 cluster system at ASU where each run was simulated in parallel.

A. Case Study: Engine Model

We will work with a high fidelity engine model from the SimuQuest Enginuity [19] Matlab/Simulink tool package. The goal of this case study is to illustrate the design process using our proposed method $\mathbb{E}$RGMC.

The Enginuity tool package includes a library of modules for engine component blocks. It also includes pre assembled models for standard engine configurations. In this work, we will use the Port Fuel Injected (PFI) spark ignition, 4 cylinder inline engine configuration. It models the effects of combustion from first physics principles on a cylinder-by-cylinder basis, while also including regression models for particularly complex physical phenomena. Simulink reports that this is a 56 state model.

We have used industry estimates for the sensor noise in several parts of the model. We have included the noise in the engine model thereby making the model stochastic. An interesting specification for an engine is the settling time for the A/F ratio, which is the quotient between the air mass and fuel mass flow. A model output is the normalized A/F ratio λ which should ideally be 1, indicating that the ratio of the air and fuel flow is the same as the stoichiometric ratio. Under

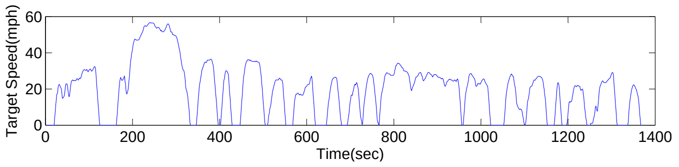

Fig. 4. Case Study 1. US EPA Urban Dynamometer Driving Schedule

engine operating conditions, this output fluctuates $\pm\%10$. We define the specification using MTL as follows:

$$\phi = \Box_{[1,102]}((\lambda \text{ out of bounds}) \rightarrow \Diamond_{[0,1]}\Box_{[0,1]}\neg(\lambda \text{ out of bounds}))$$

The specification states that always from 1 to 102 seconds, if the A/F ratio output should go out of bounds, then within 1 second it will settle inside the bounds and stay there for a second. The input to the model u is the throttle level which is between 0 and 100. We set the transmission to automatic and the engine ignites in the first second of the simulation. The total simulation time is 105 seconds. In this case study, the robustness metric which we will use to measure the satisfiability of the MTL formula will be the state robustness metric as defined in [10]. We are interested to know whether the model would satisfy the settling time specification under the United States Environmental Protection Agency's Urban Dynamometer Driving Schedule (UDDS), which is commonly called "LA4" or "the city test", which represents city driving conditions. Due to the computational limitations, we limit the simulation time to 105 seconds, and therefore we only test a part of the 1352 seconds of the "LA4" driving pattern. The driving pattern is presented in Figure 4.

The input signal u was modeled using Piecewise Cubic Hermite Interpolating Polynomials with 14 control points for the throttle angles which were distributed in the simulation time in such a way that the target speed is within ± 3 of the LA4 driving pattern. We run our algorithm which returns the minimum expected robustness value of 0.23.

Since we are testing only a section of the driving pattern we cannot claim that the system meets the specification for the whole driving pattern. Therefore, we will conduct an unconstrained search of the search space, with throttle levels between 0 and 100. We use 7 control points for the throttle angles and include the timing distribution of the control points as part of the search space. Our search space now includes 12 search variables.

The following is an illustration of the $\mathbb{E}$RGMC design process that we followed:

1. We run the $\mathbb{E}$RGMC algorithm which returns a minimum expected value -9.48. This was not an acceptable value and, therefore, we had to go back and modify the model.

2. After analyzing the output traces we noticed that in a significant number of trials the engine is stalling. Upon further investigation, we find where the issue lies. When fuel gets injected to a particular cylinder, a portion of the fuel is added to a puddle on the walls of the cylinder and the remainder is ingested during the next intake stroke of that cylinder.

The deposition of fuel on the port walls is commonly referred to as wall-wetting. Under transient conditions, the faster the throttle moves, due to the wall-wetting dynamics, the engine experiences a sustained lean excursion which can cause the engine to stall. In practice, the wall-wetting dynamics are compensated using a transient fueling control strategy.

The model we are using does not include a transient fueling strategy. Therefore as part of the design process, we go back to the model and introduce a rate limiter which will limit the increase and decrease of the acceleration.

3. After making the changes, we rerun the $\mathbb{E}$RGMC algorithm and we get a minimum expected value of 0.048.

4. We run Bayesian statistical model checking and find that the system, with 0.99 confidence, satisfies the specification with 0.9956 probability. The algorithm returned a positive robustness for 227 consecutive tests.

Finally, we check whether the engine fails the specification ϕ under varying environmental conditions. The two parameters included in the search space are the atmospheric pressure and temperature which will remain constant throughout the simulation. The atmospheric pressure ranges between 46.6 kPa to 101 kPa corresponding to the pressure at altitudes at sea level up to $20,000$ ft. The temperature ranges between $-40°$ and $40°$ Celsius. The simulation results return a minimum expected value of -2.0781, indicating that the model does not satisfy the specification. The environmental parameters at the minimum expected robustness value are: Atmospheric pressure $= 55.56$ kPa and Temperature $= 34.12°$C. The atmospheric pressure corresponds to an elevation of about $15,000$ ft.

VII. Related work

For deterministic systems, a thorough review of the different testing methodologies can be found in [3], [22]. Out of the extensive literature, the most related works are [18], [3], [12], [17]. The works [18], [3] investigate the problem of falsification by property guided search using Markov Chain Monte Carlo techniques. Monte-Carlo testing techniques were developed in [12] in order to perform random walks over the state space of a finite system. Another temporal logic robustness optimization framework is presented in [17].

The verification problem for stochastic or probabilistic hybrid systems has received much attention over the years. The focus of the early work was mainly on probabilistic reachability verification problems [7]. Temporal logic model checking and verification problems for stochastic hybrid systems have also been researched for a long time [20], [2]. Many of such works are focused on building system abstractions that can be checked using existing verification tools such as PRISM [13]. Recently, the focus has shifted to statistical model checking methods for Cyber-Physical Systems [25], [8] using as foundational principle the seminal work by Younes and Simmons [24].

In our work, instead of computing the probability that a system satisfies the property with a certain confidence, we focus on the problem of computing the worst "expected" system behavior as a quantitative value. Moreover, the worst system behavior can be returned to the user for *system debugging*. Note that debugging as a design process is not possible when using probabilistic verification techniques [13] or even statistical model checking [25], [8].

The work that is the closest to ours appeared recently in [6]. In brief, the authors in [6] also formulate an optimization problem over system parameters with the goal of optimizing the average temporal logic robustness. However, their approach utilizes an optimization algorithm [21] that first learns an approximation of the robustness landscape. Due to the learning step their framework is limited to up to 10 search variables. We remark that the search spaces of the examples presented in [6] were up to 2 dimensions. Since the parameters of the examples were not published a direct comparison was not possible.

Beyond the formal guarantees and the new theoretical results that we provide in our work, we have also demonstrated that our framework can handle industrial strength systems with 14 dimensional search spaces. Our framework has been included in our publicly available tool S-TaLiRo [1].

VIII. Conclusions

Testing and verification of Stochastic Cyber Physical Systems is a challenging problem. In this work, we have presented the verification problem for SCPS as a global optimization problem of the expected temporal logic robustness. We have utilized recent results in stochastic optimization and proved that for a specific class of cyber physical systems, we can approximate the global minimizer in finite time. For the systems outside this class, our method is reduced to a best effort automatic test generation scheme. We have extended our tool S-TaLiRo to work with the class of SCPS and search over a wider class of input signals. We have presented a framework that will utilize both guided search over the state space as well as statistical model checking methods to assist engineers in analyzing SCPSs. The use of our framework is presented through experiments and a case study of a high-fidelity engine model.

IX. Acknowledgments

The work presented here benefited from the input of Raymond Turin, Founder and CTO at SimuQuest, who provided assistance in working with the SimuQuest Enginuity model. Finally, the authors would like to thank James Kapinski and Jyotirmoy Deshmukh for the helpful discussions.

This work was partially funded under NSF awards CNS 1116136, CNS 1319560, IIP-0856090 and the NSF I/UCRC Center for Embedded Systems.

References

[1] TaLiRo Tools. https://sites.google.com/a/asu.edu/s-taliro/.

[2] A. Abate, J.-P. Katoen, and A. Mereacre. Quantitative automata model checking of autonomous stochastic hybrid systems. In *Proceedings of the 14th international conference on Hybrid systems: computation and control*, pages 83–92. ACM, 2011.

[3] H. Abbas, G. E. Fainekos, S. Sankaranarayanan, F. Ivancic, and A. Gupta. Probabilistic temporal logic falsification of cyber-physical systems. *ACM Transactions on Embedded Computing Systems*, 12(s2), May 2013.

[4] H. Abbas, B. Hoxha, G. Fainekos, and K. Ueda. Robustness-guided temporal logic testing and verification for stochastic cyber-physical systems. Online, Oct 2013. http://www.public.asu.edu/~gfaineko/pub/HSCC14tech.pdf.

[5] Y. S. R. Annapureddy, C. Liu, G. E. Fainekos, and S. Sankaranarayanan. S-taliro: A tool for temporal logic falsification for hybrid systems. In *Tools and algorithms for the construction and analysis of systems*, volume 6605 of *LNCS*, pages 254–257. Springer, 2011.

[6] E. Bartocci, L. Bortolussi, L. Nenzi, and G. Sanguinetti. On the robustness of temporal properties for stochastic models. In *Proceedings of the 2nd International Workshop on Hybrid Systems and Biology*, 2013.

[7] C. G. Cassandras and J. Lygeros. *Stochastic Hybrid Systems*. CRC Press, 2006.

[8] A. David, D. Du, K. G. Larsen, A. Legay, M. Mikucionis, D. B. Poulsen, and S. Sedwards. Statistical model checking for stochastic hybrid systems. In *Proceedings First International Workshop on Hybrid Systems and Biology*, number 92 in EPTCS, pages 122–136, 2012.

[9] G. Fainekos, S. Sankaranarayanan, K. Ueda, and H. Yazarel. Verification of automotive control applications using s-taliro. In *Proceedings of the American Control Conference*, 2012.

[10] G. E. Fainekos. *Robustness of Temporal Logic Specifications*. PhD thesis, Department of Computer and Information Science, University of Pennsylvania, 2008.

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

[12] R. Grosu and S. Smolka. Monte carlo model checking. In *Tools and Algorithms for the construction and analysis of systems*, volume 3440 of *LNCS*, pages 271–286. Springer, 2005.

[13] A. Hinton, M. Kwiatkowska, G. Norman, and D. Parker. PRISM: A tool for automatic verification of probabilistic systems. In *Tools and Algorithms for the Construction and Analysis of Systems*, volume 3920 of *LNCS*, pages 441–444. Springer, 2006.

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

[15] A. Lecchini, J. Lygeros, and J. M. Maciejowski. Stochastic optimization on continuous domains with finite-time guarantees by markov chain monte carlo methods. *IEEE Transactions on Automatic Control*, 55:2858–2863, Dec. 2010.

[16] P. Muller. Simulation based optimal design. In B. J.O., J. M. Bernardo, A. Dawid, and S. A. F. M., editors, *Proc. 6th Valencia Int. Meeting on Bayesian Statistics*, pages 459–474. Oxford, 1999.

[17] A. Rizk, G. Batt, F. Fages, and S. Soliman. Continuous valuations of temporal logic specifications with applications to parameter optimization and robustness measures. *Theor. Comput. Sci.*, 412(26):2827–2839, 2011.

[18] S. Sankaranarayanan, R. M. Chang, G. Jiang, and F. Ivancic. State space exploration using feedback constraint generation and monte-carlo sampling. In *Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, pages 321–330. ACM, 2007.

[19] Simuquest. Enginuity. http://www.simuquest.com/products/enginuity. Accessed: 2013-10-14.

[20] J. Sproston. Decidable model checking of probabilistic hybrid automata. In M. Joseph, editor, *International School and Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems*, volume 1926 of *LNCS*, pages 31–45. Springer, 2000.

[21] N. Srinivas, A. Krause, S. M. Kakade, and M. W. Seeger. Information-theoretic regret bounds for gaussian process optimization in the bandit setting. *IEEE Transactions on Information Theory*, 58(5):3250–3265, 2012.

[22] S. Tripakis and T. Dang. *Model-Based Design for Embedded Systems*, chapter Modeling, Verification and Testing using Timed and Hybrid Automata, pages 383–436. CRC Press, 2009.

[23] H. Yang, B. Hoxha, and G. Fainekos. Querying parametric temporal logic properties on embedded systems. In *Testing Software and Systems*, pages 136–151. Springer, 2012.

[24] H. L. S. Younes and R. G. Simmons. Probabilistic verification of discrete event systems using acceptance sampling. In *14th International Conference Computer Aided Verification*, volume 2404 of *LNCS*, pages 223–235. Springer, 2002.

[25] P. Zuliani, A. Platzer, and E. M. Clarke. Bayesian statistical model checking with application to simulink/stateflow verification. In *13th ACM International Conference on Hybrid Systems: Computation and Control*, pages 243–252, 2010.