Dynamic, Multi-Objective Specification and Falsification of Autonomous CPS

Kevin Kai-Chun Chang, Kaifei Xu, Edward Kim, Alberto Sangiovanni-Vincentelli, and Sanjit A. Seshia

University of California, Berkeley, Berkeley CA 94709, USA
{kaichunchang,k.px,ek65,alberto,sseshia}@berkeley.edu

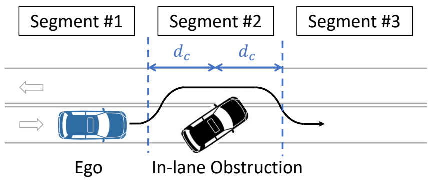
Abstract. Simulation-based falsification has proved to be an effective verification method for cyber-physical systems. Traditional approaches to falsification take as input a single or a set of temporal properties that must be satisfied by the system at all times. In this paper, we consider falsification of a more complex specification with two dimensions: multiple objectives with relative priorities and the evolution of these objectives characterized by time-varying priorities. We introduce the concept of *dynamic rulebooks* as a way to specify a prioritized multi-objective specification and its evolution over time. We develop a novel algorithm for falsifying a dynamic rulebook specification on a cyber-physical system. To evaluate our approach, we define scenarios and dynamic rulebook specifications for the domains of autonomous driving and human-robot interaction. Our experiments demonstrate that integrating dynamic rulebooks allows us to capture counterexamples more accurately and efficiently than when using static rulebooks. Moreover, our falsification framework identifies more numerous and more significant counterexamples as compared to previous approaches.

Keywords: Formal methods · Falsification · Specification · Dynamic Rulebooks · Cyber-physical systems · Autonomous driving · Human-robot interaction

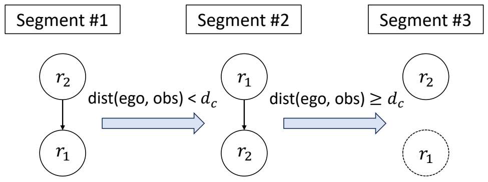
1 Introduction

Simulation-based formal analysis, such as temporal logic falsification (e.g., [1,2,6,17]), has proved to be effective in finding safety violations in cyber-physical systems (CPS) and autonomous systems enabled by artificial intelligence (AI). Numerous approaches have been proposed over the past two decades (e.g., see [18]). However, traditional approaches to falsification take as input a single or a set of temporal properties (constraints) that must be satisfied by the system at all times. As described in [14], AI-enabled autonomous systems have two important features that necessitate advances in simulation-based verification: the specification of *multiple Boolean or quantitative constraints with relative priorities*, and the *dynamic nature of constraints* and their changing priority relations over time.

As a motivating example, consider a toy autonomous vehicle scenario shown in Fig. 1a, where the ego vehicle has no other moving vehicles or people around it but faces an in-lane obstruction (such as a stalled car/road work barrier) that requires crossing the centerline of the road to avoid collision. However, this behavior violates traffic rules, presenting a scenario in which the autonomous vehicle must balance two constraints: “obeying traffic rules” and “avoiding the temporary obstruction.” Different constraints



(a) Visualization.



(b) The rulebook of the scenario.

Fig. 1: Toy scenario with lane-keeping and obstruction avoidance.

may inherently conflict, as is evident in Fig. 1a, where the ego vehicle cannot simultaneously avoid the obstruction and adhere to traffic rules. In this case, *constraints become objectives as they cannot be satisfied simultaneously*. By doing so, we can handle constraints in a more flexible way, prioritizing and trading them off. In this example, it is reasonable to prioritize avoiding collision with the obstruction over strict adherence to traffic rules when the ego vehicle is close to the obstacle, acknowledging that temporarily crossing the centerline may be deemed safe if no oncoming traffic is present, while colliding with the obstacle poses potential harm to passengers in the ego vehicle.

In addition to managing multiple objectives, the dynamics of autonomous CPS must also be taken into account. In the aforementioned scenario, the priorities among objectives can evolve over time. While the ego vehicle should prioritize avoiding collision when close to the temporary obstruction, it should prioritize obeying traffic rules when the obstruction is still far away to avoid crossing the centerline prematurely. Additionally, once the vehicle successfully navigates around the obstruction, it can deactivate the "avoiding the obstruction" objective and refocus on "obeying traffic rules" until another obstruction is detected. This example illustrates how both the objectives and their priorities can change dynamically in an autonomous CPS.

Censi et al. [4] proposed a *rulebook* structure to specify a set of objectives and their priority relations. However, the set and the relations are static, making it inadequate for addressing the dynamic nature of CPS. Viswanadha et al. [16] gave a falsification framework for multi-objective CPS specified with rulebook structures. This framework also assumed static settings where the objectives and their priorities remain fixed throughout the scenario. To the best of our knowledge, no prior work has explored the falsification problem for dynamic, multi-objective specifications. In this paper, we address the gap by proposing a *dynamic, multi-objective specification* structure and give a *novel algorithm for falsification of such structures*. Our contributions are as follows.

- Formulation of *dynamic rulebooks* capable of specifying CPS where the set of objectives and priority relations can change over time.
- Development of an efficient *algorithm for falsifying* CPS whose requirements can be modeled with dynamic rulebooks.
- Introduction of a generic falsification framework that facilitates on-the-fly construction of rulebooks.
- Experimental demonstration of our specification formalism and falsification algorithm in the domains of autonomous driving and human-robot interaction.

The remainder of this paper is organized as follows: Section 2 formulates the concept of dynamic rulebook. Section 3 defines the dynamic and multi-objective CPS falsification problem. Section 4 details our falsification framework. Section 5 develops rules and scenarios in the domains of autonomous driving and human-robot interaction. Section 6 shows experimental results. Finally, Section 7 concludes this paper.

2 Formulation of Dynamic Rulebooks

To verify a dynamic, multi-objective CPS, it is essential to assess its behavior under various conditions or inputs. To formalize this, we introduce the concept of a *scenario*, represented as a closed transition system over a set of state variables $V = \{v_1, v_2, ..., v_n\}$ with domains $\{\mathcal{D}_{v_1}, \mathcal{D}_{v_2}, ..., \mathcal{D}_{v_n}\}$:

Definition 1 (Scenario). *A scenario M is a closed transition system $M = (V, V_0, \delta_P)$, where V is the set of state variables, V_0 is the set of initial value vectors of the state variables, and $\delta_P : (\mathcal{D}_{v_1} \times \mathcal{D}_{v_2} \times ... \times \mathcal{D}_{v_n}) \times (\mathcal{D}_{v_1} \times \mathcal{D}_{v_2} \times ... \times \mathcal{D}_{v_n}) \rightarrow \mathbb{B}$ is the transition relation. Here, $V := V_r \cup V_c$, where V_r is a set of random variables whose initial values are internally sampled by the scenario*[1]*, while V_c is a set of random variables for which we need to sample their initial values. The transition relation $\delta_P(\overrightarrow{v}, \overrightarrow{v}')$ is true if and only if there is a transition from values $\overrightarrow{v}$ to $\overrightarrow{v}'$.*

The transition relation δ_P is determined by a set of parameters $P = \{p_1, p_2, ..., p_m\}$, which are not state variables, with domains $\{\mathcal{D}_{p_1}, \mathcal{D}_{p_2}, ..., \mathcal{D}_{p_m}\}$. We can then define the *input feature set* and the *input feature space* of a scenario:

Definition 2 (Input Feature Set and Input Feature Space). *The input feature set F encompasses all features used as inputs to a scenario M, determining the initial states and transitions of the scenario. Hence, $F = V_c \cup P$. The input feature space $\mathcal{F}$ is the domain that encompasses all the variables in F.*

Scenario 1 (Lane-Keeping and Obstruction Avoidance). *Here, we leverage the scenario depicted in Fig. 1 as a running example to elaborate on our formulation. In this scenario, V_c comprises the x and y positions of the ego vehicle, while the remaining state variables, such as the ego vehicle's orientation and model, are included in V_r. The transition relation updates the state variables based on the vehicle's dynamics in the simulation. The set of parameters P includes the target speed of the ego vehicle, the initial distance from the ego vehicle to the obstruction, and the distance to the obstruction at the start of lane change, which can affect the transition dynamics. For example, given the sampled target speed of the ego vehicle, the acceleration of the ego vehicle is adjusted based on the difference between the current speed and the target speed, thus influencing the subsequent positions of the ego vehicle. Finally, the input feature set F consists of the parameters along with the positions of the ego vehicle, i.e., $F = V_c \cup P$.*

Now, given a scenario M, our aim is to verify if, under an input feature vector $\overrightarrow{f} \in \mathcal{F}$, the trajectories of the scenario (i.e., the values of each state variable $v_i \in V$ at each timestamp) meet the desired constraints or objectives, referred to as *rules*. To evaluate if a rule is violated, we define the *violation score* of a rule as follows:

[1] In probabilistic CPS modeling languages such as Scenic [8], internally-sampled variables can be implicitly sampled from prior distributions

Definition 3 (Rule and Violation Score). *Given a scenario $M = (V, V_0, \delta_P)$, the violation score (VS) s of a rule r can be expressed as a function $s : \mathcal{F} \to \mathbb{R}$.*[2] *In this paper, we design the VS such that it is negative if and only if the corresponding rule is violated. Furthermore, the lower the VS, the more severely the rule is violated.*

Scenario 1. ***(continued)*** *We define the two rules, r_1 and r_2, along with their VS. These rules are expressed using signal temporal logic (STL) formulas [11]:*

- *r_1 (avoiding the temporary obstruction):*

  STL: $G_{[T_1,T_2]}(\mathit{dist}(x_0(t), x_{obs}(t)) \geq d_{safe})$;
  VS: $\mathit{min}_{[T_1,T_2]}(\mathit{dist}(x_0(t), x_{obs}(t)) - d_{safe})$,

  where $x_0(t)$ and $x_{obs}(t)$ are the positions of the ego vehicle and the in-lane obstruction, respectively; $G_{[T_1,T_2]}$ indicates the predicate inside the parentheses should always be true within the time interval $[T_1, T_2]$.
- *r_2 (obeying the traffic rule of not crossing the centerline):*

  STL: $G_{[T_1,T_2]}(\mathit{dist}(x_0(t), L) = 0)$; ***VS:*** $-\,\mathit{max}_{[T_1,T_2]}(\mathit{dist}(x_0(t), L))$,

  where L denotes the polygon region of the lane, and $dist(x_0(t), L)$ indicates the minimum distance from the ego vehicle to the polygon. If the vehicle is within the polygon (i.e., it is on the lane), $dist(x_0(t), L) = 0$.

Indeed, VS is negative if and only if the ego vehicle violates the corresponding rule, and the lower the VS, the more severe the violation.

After defining the rules, we can introduce the *static rulebook* as proposed in [4], which specifies the rules and their priorities in a static multi-objective system.

Definition 4 (Static Rulebook). *A static rulebook B is a tuple $B = (R, \leq_R)$, where R is a set of rules and $\leq_R$ is a partial order on R indicating the relative priority of the rules. A static rulebook can be represented as a directed graph $G_B = (V_B, E_B)$, where each node in V_B represents a rule, and each edge in E_B indicates the priority relation between two rules. If there is an edge from node A to node B, then rule r_A has a higher priority than r_B.*

Finally, we propose the definition of *dynamic rulebook* for dynamic, multi-objective systems:

Definition 5 (Dynamic Rulebook). *The dynamic rulebook B_M for a scenario M is a tuple $B_M = (R_M, \leq_{R_M}, U_M)$. Similar to the static rulebook, R_M is a set of rules and $\leq_{R_M}$ is a partial order on R_M. However, the key difference lies in the dependency of R_M and $\leq_{R_M}$ on the state variables of the scenario M. As the state variables in M change, both R_M and $\leq_{R_M}$ can be updated, unlike in the static rulebook where R and $\leq_R$ remain fixed. Rules may be added or removed from R_M, and the priority*

[2] We denote the function s in this way for simplicity. Actually, the function maps the trajectory generated under an input feature vector $\overrightarrow{f} \in \mathcal{F}$ to $\mathbb{R}$.

among rules may also change. Formally, U_M is a transition function that defines these updates. Given the current R_M, $\leq_{R_M}$, and the values of the state variables in M, U_M outputs the updated R_M and $\leq_{R_M}$. A dynamic rulebook B_M can also be depicted as an evolving directed graph $G_{B_M} = (V_{B_M}, E_{B_M})$. With changes in state variables in M, nodes may be added or removed from the graph, and edges may be added, removed, or reversed.

Property 1 (Segmentation of a Dynamic Rulebook). Each time the state variables in M change, R_M and $\leq_{R_M}$ of a dynamic rulebook may be updated. Thus, from another perspective, a dynamic rulebook can be viewed as a sequence of static rulebooks, denoted $B_M = \{B_1, B_2, B_3, ...\}$, where B_i is updated to B_{i+1} with some state changes. Similarly, the graphical representation of a dynamic rulebook can also be viewed as a sequence of directed graphs. We refer to the time segment corresponding to the static rulebook B_i as the i-th segment of scenario M.

For simplicity, unless otherwise specified, the "rulebook" refers to the "dynamic rulebook" throughout the remainder of this paper. The rulebook for Scenario 1 can be defined as follows:

Scenario 1. ***(continued)*** *As shown in Fig. 1b, the rulebook can evolve over time based on the position of the ego vehicle. Initially, when the ego vehicle is still far from the obstruction, r_2 has a higher priority than r_1 to prevent premature crossing of the centerline. As the ego vehicle approaches the obstruction (i.e., when the distance is less than a threshold d_c), r_1 gains priority to ensure the avoidance of the obstruction. Once the obstruction is successfully avoided (i.e., when the distance again exceeds d_c), r_1 is removed from the rulebook (indicated by the dashed circle in Fig. 1b).*

3 Problem Formulation

The *dynamic, multi-objective CPS falsification* problem can be formulated as follows:

Given a scenario $M = (V, V_0, \delta_P)$ along with the corresponding dynamic rulebook $B_M = (R_M, \leq_{R_M}, U_M)$, which can be viewed as a sequence of static rulebooks, our goal is to sample from the input feature space $\mathcal{F}$ to identify feature vectors that violate higher-priority rules and find more number of counterexamples, for all the static rulebooks in the sequence.

Formally, given two input feature vectors $\overrightarrow{f_1}$ and $\overrightarrow{f_2}$, $\overrightarrow{f_1}$ falsifies more than $\overrightarrow{f_2}$ for segment k if the following formula holds for the static rulebook B_k:

$$\begin{aligned} &\forall i \cdot [s_i(\overrightarrow{f_2}) < s_i(\overrightarrow{f_1}) \\ &\qquad \implies \exists j \cdot ((r_i <_{R_k} r_j) \wedge (s_j(\overrightarrow{f_1}) < s_j(\overrightarrow{f_2})))], \end{aligned} \tag{1}$$

where r_i and r_j are rules with corresponding VS s_i and s_j, respectively. Recall that lower values output by s_i (or s_j) indicate more severe rule violations. Thus, the formula expresses that if $\overrightarrow{f_2}$ violates some rule i more severely than $\overrightarrow{f_1}$, then there must exist a higher-priority rule j such that $\overrightarrow{f_1}$ violates more severely than $\overrightarrow{f_2}$ on rule j. For simplicity, input feature vectors that lead to violations of some rules are called *counterexamples* in the remainder of the paper. Also, input feature vectors that falsify more severely are called "larger" counterexamples.

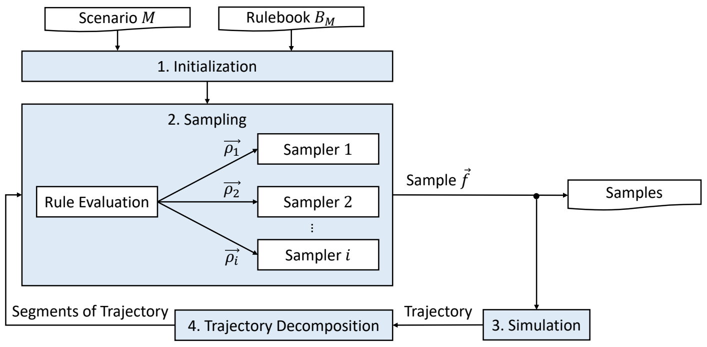


Fig. 2: Our falsification framework.

4 Our Framework

4.1 Overview of Our Framework

Fig. 2 outlines our falsification framework. As explained in Property 1, a dynamic rulebook B_M can be understood as a sequence of static rulebooks $\{B_1, B_2, B_3, ...\}$. In our approach, we create a dedicated sampler for each static rulebook in the sequence (Step 1). During the sampling process (Step 2), in each iteration, one of these samplers is chosen to generate a sample. The selection of samplers follows a sequential style: first, Sampler 1 is chosen for N iterations, where N is a user-defined number; then Sampler 2 for another N iterations, then Sampler 3 for another N iterations, and so on. The generated sample is sent to the simulator for scenario simulation (Step 3), which produces a trajectory recording the state variable values in V at each timestamp $t \in [0, T]$, where T represents the simulation time interval. The trajectory is segmented into segments, each corresponding to a static rulebook (Step 4). Following trajectory decomposition, rules of each static rulebook $B_k = (R_k, \leq_{R_k})$ are evaluated on the corresponding segment. The evaluation yields $\overrightarrow{\rho_k}$, encompassing the VS of all the rules in R_k, which is forwarded to the respective sampler as feedback. The sampler is then updated accordingly, and we proceed to sample for the next iteration (Step 2). This process iterates through Steps 2 to 4 until all samplers have been sampled N times.

4.2 Our Sampling Algorithm

In [16], the *multi-armed bandit (MAB)* sampling strategy is proposed, which is extended from [3] and the cross-entropy sampling algorithm [13]. This approach partitions each dimension of the sample space into multiple buckets and selects a bucket for sampling during each iteration. The fundamental principle of the sampler is to balance between exploitation and exploration: it seeks to identify samples that violate high-priority rules (exploitation), while also diversifying generated samples across the sample space (exploration). To achieve this balance, two matrices, $\mathcal{R}$ and $\mathcal{T}$, are employed. The exploitation matrix $\mathcal{R}$ records the source buckets of the current maximal counterexample, while the exploration matrix $\mathcal{T}$ is inversely proportional to the frequency of visits to each bucket. These matrices are then combined into a single matrix Q, from which the sampler selects the bucket with the highest corresponding value.

Algorithm 1: Initialization of Sampler k

```
Input: The feature set $F$ and the feature domain $\mathcal{F}$, the bucket size $N$, and the static
       rulebook $B_k = (R_k, \leq_{R_k})$
$d \leftarrow |F|$; // the dimension of the feature set
$\mathcal{E}_{d \times N} \leftarrow \mathcal{O}$; // initialize the error matrix as a zero matrix
$\mathcal{C}_{d \times N} \leftarrow \mathcal{J}$; // initialize the count matrix as an all-one
    matrix
$t \leftarrow 1$; // the iteration count
$i \leftarrow 0$;
$e_{max} \leftarrow 0$;
foreach rule $r$ in $R_k$ do
    $m \leftarrow$ number of rules have lower priorities than $r$;
    $\overrightarrow{w}[i] \leftarrow 2^m$; // the error weight
    $e_{max} \leftarrow e_{max} + 2^m$;
    $i \leftarrow i + 1$;
end
```

However, a limitation of the MAB approach is its reliance solely on information from the currently identified maximal counterexample. Consequently, in scenarios with complex rulebooks, it may become trapped in local optima, making it challenging to identify larger counterexamples. Moreover, the MAB sampler lacks a mechanism for distinguishing between different counterexamples. Specifically, it always increments the corresponding entry in $\mathcal{R}$ regardless of the magnitude of the counterexample.

To address these limitations, we introduce the concept of *error weight* w:

Definition 6 (Error Weight of a Rule). *For a rule r within a static rulebook $B = (R, \leq_R)$, suppose there are m number of rules which are lower in priority than r, the error weight w_r of r is defined as 2^m.*

Subsequently, whenever a counterexample is discovered, the error weights of the violated rules are aggregated to form the *error value* e, which indicates the magnitude of the counterexample.

Example 1 (Error Weight and Error Values). Consider a static rulebook $B = \{R, \leq_R\}$ consisting of four rules $r_1, r_2, r_3,$ and r_4, where $r_4 >_R r_3 =_R r_2 >_R r_1$. The error weights for rules r_4, r_3, r_2, and r_1 are 8, 2, 2, and 1, respectively. If a counterexample violates r_4, r_2, and r_1, its error value will be $8 + 2 + 1 = 11$.

Proposition 1. *Given two counterexamples $\overrightarrow{f_1}$ and $\overrightarrow{f_2}$, $\overrightarrow{f_1}$ has a higher error value than $\overrightarrow{f_2}$ if it falsifies more than $\overrightarrow{f_2}$ based on the definition in Section 3.*

Algorithms 1, 2, and 3 detail our sampling approach. In the initialization phase (Algorithm 1), an error matrix $\mathcal{E}$ and a count matrix $\mathcal{C}$ are initialized. Both matrices are of size $d \times N$, where d is the dimension of the feature set and N represents the bucket size. Additionally, following Definition 6, the error weight of each rule is computed and stored in the error weight vector $\overrightarrow{w}$ (Lines 6 to 12). The sum of error weights, representing the maximum error value of a counterexample, is stored as e_{max}.

During the updating phase (Algorithm 2), a feedback vector $\overrightarrow{\rho_k}$ and a source bucket vector $\overrightarrow{b}$ are input. $\overrightarrow{\rho_k}$ consists of the output violation scores of all the rules in the

Algorithm 2: Updating of Sampler k

```
Input: The feedback vector $\overrightarrow{\rho_k}$ and the source bucket vector $\overrightarrow{b}$
1  $e \leftarrow 0$;
2  for $i \leftarrow 0$ to $|R_k| - 1$ do
3      if $\overrightarrow{\rho_k}[i] < 0$ then
4          $e \leftarrow e + \overrightarrow{w}[i]$
5      end
6  end
7  for $i \leftarrow 0$ to $d$ do
8      $j \leftarrow \overrightarrow{b}[i]$;
9      $\mathcal{E}[i][j] \leftarrow \mathcal{E}[i][j] + e$;
10     $\mathcal{C}[i][j] \leftarrow \mathcal{C}[i][j] + e_{max}$;
11     $t \leftarrow t + 1$;
12 end
```

rulebook. $\overrightarrow{b}$ contains the source buckets of the current sample. For example, if the i-th feature is sampled from the j-th bucket, then $\overrightarrow{b}[i] = j$. In Lines 1 to 6, the sampler computes the corresponding error value based on $\overrightarrow{\rho_k}$. In Lines 7 to 12, for each i-th dimension of the feature space, the source bucket $\overrightarrow{b}[i]$ is extracted. The error value is then added to the corresponding entry of $\mathcal{E}$, and e_{max} is added to the corresponding entry of $\mathcal{C}$. This ensures that if sampling from the j-th bucket for the i-th feature leads to a greater number of and larger counterexamples, $\mathcal{E}[i][j]$ will become larger. Similarly, the magnitude of entries in $\mathcal{C}$ reflects the sampling frequency from the corresponding bucket.

After sampler updating, in the sampling phase (Algorithm 3), we normalize the error matrix by performing element-wise division of $\mathcal{E}$ by $\mathcal{C}$ and set the result as the exploitation matrix $\mathcal{R}$. Since $\mathcal{R}$ is proportional to $\mathcal{E}$, it encourages the sampler to sample more from the buckets that have generated more and larger counterexamples. For the exploration aspect, we adopt the definition of from [16], setting the exploration matrix $\mathcal{T}$ as $\sqrt{\frac{ln(t)}{\mathcal{C}}}$, where t is the current sampling iteration. Since the exploration matrix is inversely correlated to the sampling frequency $\mathcal{C}$, it encourages sampling from less frequently visited buckets. We then introduce a balance coefficient δ to control the balance between the two terms, combining them into matrix $\mathcal{Q}$. For each dimension of the feature space, the bucket corresponding to the highest entry value in $\mathcal{Q}$ is selected for sampling.

Example 2 demonstrates efficiency of our sampling approach over the MAB strategy:

Example 2 (Comparison of Our Sampling Approach against the MAB approach). Consider a static rulebook $B = \{R, \leq_R\}$ consisting of five rules r_1, r_2, r_3, r_4, and r_5, where $r_5 >_R r_4 >_R r_3 >_R r_2 >_R r_1$. Suppose at some point during the falsification process, a counterexample is found that violates r_1, r_2, r_3, and r_4, making it the current maximal counterexample. Later, a larger counterexample is found that violates only r_5.

In the previous MAB approach, upon finding the new maximal counterexample, all information pertaining to previous counterexamples is discarded. Consequently, the sampler must "rediscover" the buckets associated with violations of r_1, r_2, r_3, and r_4.

Algorithm 3: Sampling from Sampler k

```
Input: The balance coefficient $\delta$
Output: A feature vector (i.e., sample) $\overrightarrow{f} \in \mathcal{F}$ and a source bucket vector $\overrightarrow{b}$
1 $\mathcal{R} \leftarrow \mathcal{E}/\mathcal{C}$; // element-wise division
2 $\mathcal{T} \leftarrow \sqrt{\frac{ln(t)}{\mathcal{C}}}$; // element-wise division
3 $\mathcal{Q} \leftarrow \mathcal{R} + \sqrt{\delta} \cdot \mathcal{T}$;
4 for $i \leftarrow 0$ to $d$ do
5     $j^* \leftarrow \text{argmax}_j\, \mathcal{Q}[i][j]$;
6     $\overrightarrow{f}[i] \leftarrow$ uniformly randomly sample from the range of bucket $j^*$;
7     $\overrightarrow{b}[i] \leftarrow j^*$;
8 end
```

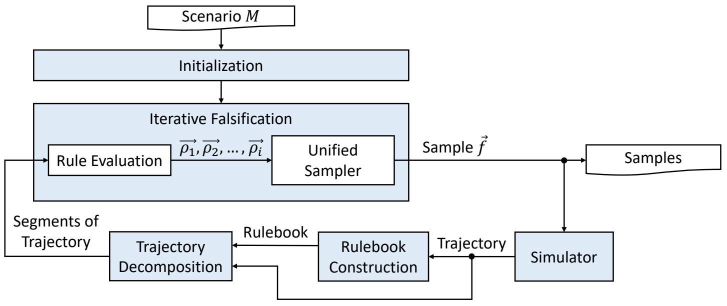


Fig. 3: Our generic falsification framework.

On the contrary, our sampling strategy leverages information from both types of counterexamples, facilitating the efficient discovery of even larger counterexamples (e.g. those violating all rules r_1, r_2, r_3, r_4, and r_5). Furthermore, employing our error weight scheme, the error values of the two counterexamples are 15 and 16, respectively. These values reflect the "importance" of the counterexamples and serves as weights during sampler updating.

4.3 Extension to a Generic Framework

In our problem formulation, we assume the dynamic rulebook B_M is given. Specifically, the corresponding sequence of static rulebooks is predetermined before the falsification process starts. However, in some scenarios, updates to the rulebook may vary from one simulation iteration to another, depending on the trajectories of state variables. Under such circumstances, the falsification framework proposed in Sections 4.1 and 4.2 becomes infeasible. An example of this type of scenario is provided in Scenario 4.

To accommodate this variability, we propose a more versatile framework employing a "unified" sampler, as shown in Fig. 3. Unlike dedicated samplers for individual static rulebooks, a single unified sampler generates samples throughout the entire falsification process. Moreover, the rulebook is not predefined; rather, it is constructed iteratively based on simulator-generated trajectories.

The initialization of the unified sampler mirrors Algorithm 1, with the key distinction that error weights are computed before each update. Sampling from the sampler also resembles Algorithm 3. For sampler updating, we have to consider the feedback vectors from all segments, $\overrightarrow{\rho_1}, \overrightarrow{\rho_2}, ..., \overrightarrow{\rho_i}$. To facilitate this, we introduce the concept of *normalized error value*:

Definition 7 (Normalized Error Value). *As defined in Section 4.2, the error value e of a counterexample is the sum of error weights of violated rules. The normalized error value e_{norm} is computed as the ratio of error value e to the maximum possible error value e_{max} (i.e., the error value when all rules are violated), expressed as $e_{norm} = \frac{e}{e_{max}}$.*

Taking the rulebook defined in Example 1 as an example, the maximum error value (e_{max}) is the sum of the error weights of all rules, which is 13. If a counterexample violates r_4, r_2, and r_1, its normalized error value (e_{norm}) will be $\frac{11}{13} \approx 0.846$.

Given rulebook B and feedback vectors $\overrightarrow{\rho_1}, \overrightarrow{\rho_2}, ..., \overrightarrow{\rho_i}$, the sampler computes error weights and normalized error values. Subsequently, segment-wise normalized error values are averaged to derive the *average error value* e_{avg}. Lastly, e_{avg} is added to the corresponding entry of $\mathcal{E}$, while the corresponding entry of $\mathcal{C}$ is incremented by 1, reflecting e_{avg}'s range between 0 and 1.

Although the generic falsification framework supports on-the-fly rulebook construction, it may not outperform the scenario-segmentation-based framework when the rulebook is available in advance. While the framework with a unified sampler can implicitly account for dependencies between segments, it also records less information compared to the framework with dedicated samplers. In Section 6.1, we provide a thorough comparison of the two frameworks.

5 Scenarios

To validate the effectiveness of our falsification framework, we construct multiple scenarios spanning the domains of autonomous driving and human-robot interaction. These scenarios are designed to simulate real-world situations while examining the variability of rulebooks. Before diving into specific scenarios, we first define four kinds of fundamental rules, along with their VS, that are essential in both domains [10, 15]:

Rule A (Distance). *Two objects i and j, with positions $x_i(t)$ and $x_j(t)$, must maintain a safe distance from each other within a specified time interval, denoted as $r_{A,i,j}$.*

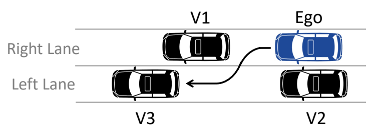
STL: $G_{[T_1,T_2]}(dist(x_i(t), x_j(t)) \geq d_{safe})$; ***VS:*** $min_{[T_1,T_2]}(dist(x_i(t), x_j(t)) - d_{safe})$

Rule B (Staying in Region). *An object i, with position $x_i(t)$, must stay within a specific region during a specified time interval, denoted as $r_{B,i,reg}$, where "reg" denotes the region.*

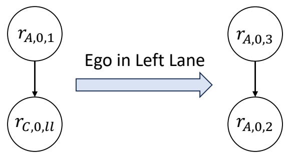
STL: $G_{[T_1,T_2]}(dist(x_i(t), reg) == 0)$; ***VS:*** $-max_{[T_1,T_2]}(dist(x_i(t), reg))$

Rule C (Reaching Goal). *An object i, with position $x_i(t)$, must reach its destination within a specified time interval, denoted as $r_{C,i,goal}$, where "goal" denotes the destination region.*

STL: $F_{[T_1,T_2]}(dist(x_i(t), goal) == 0)$; ***VS:*** $-min_{[T_1,T_2]}(dist(x_i(t), goal))$



(a) The visualization.



(b) The rulebook.

Fig. 4: The lane change scenario (Scenario 2).

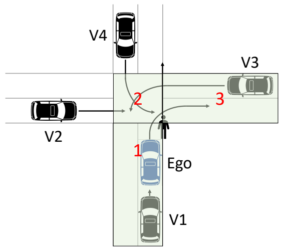


Fig. 5: The lane keeping and intersection crossing scenario (Scenario 3).

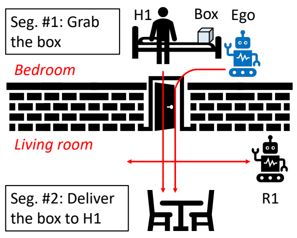


Fig. 6: The object fetching and delivery scenario (Scenario 5).

Rule D (Lane Keeping). *An object i, with position $x_i(t)$, must not deviate too far from the center of a specified lane within a specified time interval, denoted as $r_{D,i,cent}$, where "cent" denotes the center of the lane.*

STL: *$G_{[T_1,T_2]}(dist(x_i(t), cent) \leq d_{keep})$*; ***VS:*** *$d_{keep} - max_{[T_1,T_2]}(dist(x_i(t), cent))$*

It's important to note that the above rules are abstract and can be refined into multiple instances in practical scenarios. For example, each pair of pedestrians, vehicles, or objects on the road may require a distinct instance of Rule A. Additionally, the safe distance d_{safe} and the lane-keeping distance d_{keep} may vary across different instances.

In addition to the obstruction avoidance scenario (Scenario 1), we have developed three additional scenarios within the autonomous driving domain. We use 0 to denote the ego vehicle, positive integers $(1, 2, ...)$ to represent other vehicles, and negative integers $(-1, -2, ...)$ to indicate pedestrians. For example, $r_{A,0,1}$ specifies the requirement to maintain a safe distance between the ego vehicle and vehicle 1.

Scenario 2 (Lane Change). *Fig. 4a illustrates a lane change scenario, and Fig. 4b illustrates the corresponding rulebook. In this scenario, the ego vehicle intends to transition from the right lane to the left lane. Prior to initiating the lane change, the vehicle must maintain a safe distance from vehicle 1 ($r_{A,0,1}$) and attempt to overtake vehicle 2 to finish lane change ($r_{C,0,ll}$, where ll denotes the left lane). Upon successfully completing the lane change, the rules become maintaining a safe distance from vehicles 3 ($r_{A,0,3}$) and 2 ($r_{A,0,2}$).*

A noteworthy aspect of the rulebook is the disjoint nature of nodes in the two segments. This feature enables the evaluation of our framework's performance under significant rule changes.

Scenario 3 (Lane Keeping and Intersection Crossing). *Fig. 5 illustrates a lane keeping and intersection crossing scenario, segmented into three parts. In the first segment,* *the ego vehicle performs lane keeping on an incoming lane of an intersection. The goal is to reach the intersection (denoted as "$inter$"). Moving to the second segment, upon reaching the intersection, the ego vehicle aims to make a right turn amidst the presence of other vehicles and a pedestrian crossing and reach the outgoing lane (denoted as "ol"). Finally, in the third segment, the ego vehicle resumes lane keeping on the outgoing lane. Throughout this process, the ego vehicle must adhere to rules ensuring a safe distance from pedestrians and other vehicles, as well as remaining within the drivable area (denoted as "da"). Formally, the rulebook for this scenario is defined as follows:*

Segment 1: $r_{A,0,1} > r_{B,0,da} > r_{C,0,inter} > r_{D,0,cent}$

Segment 2: $r_{A,0,-1} > r_{A,0,1} = r_{A,0,2} = r_{A,0,3} = r_{A,0,4} > r_{B,0,da} > r_{C,0,ol}$

Segment 3: $r_{A,0,2} = r_{A,0,3} = r_{A,0,4} > r_{B,0,da} > r_{D,0,cent}$

Scenario 4 (Intersection Management). *In this scenario, we shift our perspective from the ego vehicle to a centralized intersection manager. This manager employs a first-come-first-serve principle to determine the passing order of vehicles at the intersection. Upon a vehicle's arrival, it informs the manager, who then adds it to the waiting queue. As the intersection becomes vacant, the manager allows the vehicle at the forefront of the queue to proceed while others must halt. The manager ensures the correct passing order while maintaining safe distances between vehicles.*

To model the rulebook for this scenario, rules are dynamically added or removed as vehicles arrive or depart. When a new vehicle joins the queue, distance rules between it and existing vehicles are appended to the rulebook, alongside a rule enforcing the sequential passing order. For example, assume the current waiting queue is $q = \{V_1, V_2, ..., V_i\}$ and a vehicle V_{i+1} arrives, rules $r_{A,1,i+1}, r_{A,2,i+1}, ..., r_{A,i,i+1}$ and a rule enforcing "V_i must exit the intersection before V_{i+1}" are added. Conversely, when a vehicle exits, the corresponding rules are removed. The distance rules hold higher priority than the passing order rule.

Given the varying order of vehicle arrivals and departures across simulations, the rulebook must be constructed on-the-fly during falsification.

For objects in the human-robot interaction (HRI) domain, we use 0 to denote the ego robot, positive integers to represent obstacles or other robots, and negative integers to indicate humans. For example, $r_{A,0,-1}$ specifies the requirement to maintain a safe distance between the ego robot and human 1.

Scenario 5 (Object Fetching and Delivery). *Figure 6 illustrates this scenario. In the first segment, the ego robot aims to grab a box of important documents from a bed while avoiding a human organizing clothes nearby. In the second segment, the human moves from the bedroom to the living room for work, and the ego robot must deliver the documents to the human while avoiding another robot vacuuming the area. Formally, the rulebook can be defined as follows:*

Segment 1: $r_{A,0,-1} > r_{C,0,box}$; *Segment 2:* $r_{A,0,1} > r_{C,0,-1}$

A noteworthy aspect of this rulebook is that in Segment 1, the robot must maintain a safe distance from the human ($r_{A,0,-1}$), while in Segment 2, the robot aims to approach the human ($r_{C,0,-1}$). This reflects how interactions between objects in an autonomous CPS evolve over time.

6 Experiments

We conducted our experiments on a Ubuntu 20.04 Linux workstation equipped with a 3.7 GHz CPU and 64 GB of RAM. The scenarios are described using the Scenic programming language [7, 8]. Scenic is a domain-specific probabilistic programming language tailored for formally describing scenarios in CPS, encompassing domains such as autonomous driving and robotics. The falsification framework is implemented in Python 3. In addition, our framework is seamlessly integrated with VerifAI [6], a formal verification environment offering a range of built-in sampling algorithms.

For the experiments in the autonomous driving domain, we utilized the Scenic built-in Newtonian simulator and the CARLA simulator [5]. For experiments in the robotics domain, we used the Meta Habitat 3.0 simulator [12] that offers modeling and simulation of both robots and humanoids. For each segment within every scenario in the autonomous driving domain, we selected three different random seeds and perform 300 iterations per seed, resulting in a total of 900 samples per segment. Similarly, for Scenario 5 in the HRI domain, we used three different random seeds with 200 iterations per seed, resulting in a total of 600 samples per segment. For each input feature, we divide its range into 5 buckets. The balance coefficient δ is set to 2 for the autonomous driving experiments and 1 for the HRI experiments.

To assess the capabilities of our falsification approach, we employ several metrics, including the maximum normalized error value among all samples (Max e_{norm}), the average normalized error value of all samples (Avg. e_{norm}), the percentage of the maximum counterexample out of all samples (Pct. Max CE), and the percentage of counterexamples out of all samples (Pct. CE). Note that due to its definition, e_{norm} may not be suitable for comparisons across different rulebooks.

6.1 Results

In this section, our aim is to address three research questions.

RQ1: Would using a dynamic rulebook result in finding more number of and larger counterexamples than a monolithic static rulebook? Table 1 compares the falsification results obtained using (1) dynamic rulebook with our sampling algorithm depicted in Fig. 2, (2) static rulebook with our sampling algorithm depicted in Fig. 2, and (3) static rulebook with the MAB algorithm, which has been shown to perform best among previous sampling approaches for static multi-objective systems in [16]. For experiments with static rulebooks, we combined the static rulebooks from each segment into a monolithic static rulebook, manually adjusting the priorities to encompass the requirements across all segments of a scenario. For Scenario 1, the monolithic static rulebook includes r_1 and r_2 with the same priority level. For Scenario 2, as the rules in the two static rulebooks are disjoint, they can be merged into a monolithic static rulebook directly. This results in a rulebook with 4 rules ($r_{A,0,1}, r_{C,0,ll}, r_{A,0,2}, r_{A,0,3}$) and 2 edges ($r_{A,0,1}$ pointing to $r_{C,0,ll}$ and $r_{A,0,3}$ pointing to $r_{A,0,2}$). For Scenario 3, the monolithic static rulebook includes all the rules, prioritizing $r_{A,0,-1}$ highest, followed by all r_A-type rules, then $r_{B,0,da}$, all r_C-type rules, and finally $r_{D,0,cent}$. Notably, for Scenario 5, however, the conflicting rules $r_{A,0,-1}$ (maintaining a safe distance from the human) in Segment 1 and $r_{C,0,-1}$ (approaching the human) in Segment 2 make it

Table 1: Comparison of dynamic and static rulebooks.

Rulebook	Scen./Seg.	Max e_{norm}	Avg. e_{norm}	Pct. Max CE	Pct. CE	Scen./Seg.	Max e_{norm}	Avg. e_{norm}	Pct. Max CE	Pct. CE
Dynamic		1.000	**0.313**	**31.3%**	**31.3%**		0.733	**0.547**	**7.7%**	**99.0%**
Static	1/1	1.000	0.057	5.7%	5.7%	3/1	0.733	0.530	0.3%	98.3%
Static+MAB		1.000	0.057	5.7%	5.7%		0.733	0.521	2.0%	96.3%
Dynamic		1.000	**0.725**	**58.8%**	100.0%		0.976	**0.819**	**20.7%**	**98.0%**
Static	1/2	1.000	0.673	50.9%	100.0%	3/2	0.976	0.719	15.0%	96.3%
Static+MAB		1.000	0.623	43.4%	100.0%		0.976	0.734	11.3%	95.3%
Dynamic		1.000	**0.571**	**57.1%**	**57.1%**		0.867	**0.302**	**4.0%**	**62.0%**
Static	1/3	1.000	0.357	35.7%	35.7%	3/3	0.867	0.215	1.7%	**62.0%**
Static+MAB		1.000	0.341	34.1%	34.1%		0.867	0.229	1.3%	58.0%
Dynamic		1.000	**0.351**	15.4%	**73.9%**		1.000	0.743	53.2%	86.3%
Static	2/1	1.000	0.301	**16.1%**	57.8%	5/1	N.A.	N.A.	N.A.	N.A.
Static+MAB		1.000	0.262	9.6%	59.1%		N.A.	N.A.	N.A.	N.A.
Dynamic		1.000	**0.471**	**45.0%**	**51.4%**		1.000	0.836	75.3%	100.0%
Static	2/2	1.000	0.284	21.3%	42.6%	5/2	N.A.	N.A.	N.A.	N.A.
Static+MAB		1.000	0.279	21.6%	40.6%		N.A.	N.A.	N.A.	N.A.

Table 2: Comparison of sampling algorithms.

Algorithm	Scen./Seg.	Max e_{norm}	Avg. e_{norm}	Pct. Max CE	Pct. CE	Scen./Seg.	Max e_{norm}	Avg. e_{norm}	Pct. Max CE	Pct. CE
Ours		1.000	**0.313**	**31.3%**	**31.3%**		0.733	**0.547**	**7.7%**	**99.0%**
MAB	1/1	1.000	0.169	16.9%	16.9%	3/1	0.733	0.533	1.0%	98.7%
Halton		1.000	0.100	10.0%	10.0%		0.733	0.524	1.0%	97.7%
Ours		1.000	**0.725**	**58.8%**	100.0%		0.976	**0.819**	**20.7%**	**98.0%**
MAB	1/2	1.000	0.668	50.1%	100.0%	3/2	0.976	0.706	17.0%	95.3%
Halton		1.000	0.516	27.3%	100.0%		0.976	0.760	15.3%	93.0%
Ours		1.000	0.571	57.1%	57.1%		0.867	**0.302**	**4.0%**	**62.0%**
MAB	1/3	1.000	**0.753**	**75.3%**	**75.3%**	3/3	0.867	0.214	3.3%	61.7%
Halton		1.000	0.299	29.9%	29.9%		0.867	0.251	2.0%	57.0%
Ours		1.000	**0.351**	**15.4%**	**73.9%**		1.000	0.743	53.2%	86.3%
MAB	2/1	1.000	0.283	10.6%	63.4%	5/1	1.000	**0.776**	**53.3%**	**92.5%**
Halton		1.000	0.289	9.6%	67.1%		1.000	0.748	49.0%	**92.5%**
Ours		1.000	**0.471**	**45.0%**	**51.4%**		1.000	**0.836**	**75.3%**	100.0%
MAB	2/2	1.000	0.437	43.2%	44.8%	5/2	1.000	0.824	73.7%	100.0%
Halton		1.000	0.297	27.9%	33.3%		1.000	0.822	73.0%	100.0%

impossible to merge into a monolithic rulebook, highlighting the limitation of static rulebooks in addressing dynamic CPS requirements.

As shown in the table, across almost all segments, dynamic rulebooks yield higher Avg. e_{norm}, Pct. Max CE, and Pct. CE compared to static rulebooks. This improvement occurs because dynamic rulebooks decompose scenario requirements into a sequence of smaller static rulebooks, allowing the framework to precisely identify counterexamples relative to the rulebook of each segment, unlike static rulebooks which cannot differentiate between segments. Similarly, dynamic rulebooks yield higher Avg. e_{norm}, Pct. Max CE, and Pct. CE than the static rulebook with the MAB setting in all segments. Moreover, comparing the results of the static rulebook with the MAB setting to the dynamic rulebook with the MAB setting (row 2 in Table 2), the latter yields better results in the majority of segments, demonstrating that under different sampling algorithms, dynamic rulebooks still capture more and larger counterexamples than static rulebooks.

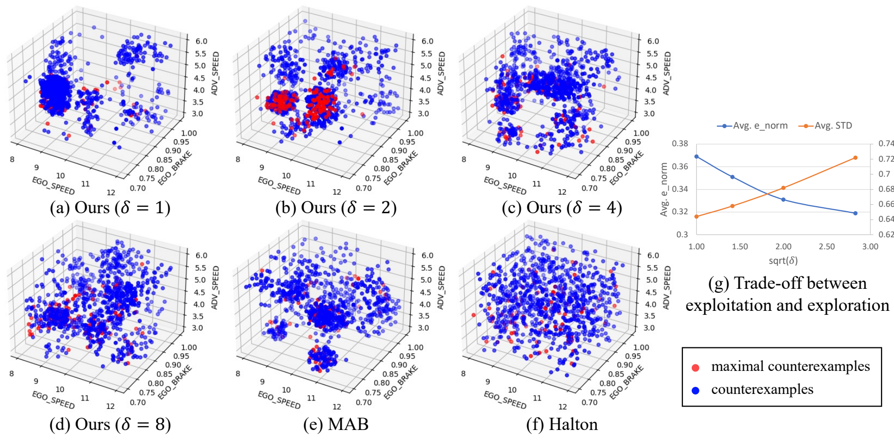
RQ2: Given a system specified with a dynamic rulebook, does our sampling algorithm identify a greater number of and larger counterexamples compared to previous sampling approaches? Table 2 compares the falsification results obtained using various sampling approaches. We compare our algorithm against multi-armed bandit (MAB) sampler, which is noted as the best previous sampling approach for multi-objective systems [16], and an efficient passive sampler, the Halton sampler [9].

Table 3: Analysis of the generic framework.

Scen./Seg.	Framework	Max e_{norm}	Avg. e_{norm}	Pct. Max CE	Pct. CE
4	Segmentation	N.A.	N.A.	N.A.	N.A.
	Unified	0.959	0.636	0.7%	100.0%
2/1	Segmentation	1.000	0.351	15.4%	73.9%
	Unified	1.000	0.303	13.1%	64.1%
2/2	Segmentation	1.000	0.471	45.0%	51.4%
	Unified	1.000	0.351	34.4%	36.6%

Table 4: Average violation scores of Scenario 2.

Algorithm / Rules	$r_{A,0,1}$	$r_{C,0,ll}$	$r_{A,0,2}$	$r_{A,0,3}$
Ours	0.337	0.361	-2.439	-0.906
MAB	0.410	0.524	-2.683	-1.384
Halton	0.466	0.455	-1.364	-0.398



(a) Ours ($\delta = 1$) (b) Ours ($\delta = 2$) (c) Ours ($\delta = 4$) (g) Trade-off between exploitation and exploration (d) Ours ($\delta = 8$) (e) MAB (f) Halton

Fig. 7: The trade-off between exploitation and exploration in Scenario 2.

Notably, the MAB sampler has been adapted for dynamic settings. The table shows that our framework yields a higher Avg. e_{norm}, Pct. Max CE, and Pct. CE than the MAB sampler in eight out of the ten segments, validating the effectiveness of our improvement to the MAB sampler. Similarly, our framework also outperforms the Halton sampler in almost all segments.

RQ3: Can our framework be adapted to diverse domains of applications? As illustrated in Tables 1 and 2, our framework demonstrates comparable or superior performance in scenarios from both the autonomous driving and human-robot interaction domains.

6.2 Discussion

In this section, we provide a thorough analysis of the generality, flexibility, and efficiency of our framework.

Analysis of Our Unified-Sampler-based Approach. In Section 4.3, we propose a unified-sampler-based approach that supports on-the-fly rulebook construction. We validate its effectiveness with the intersection management scenario (Scenario 4). Since this scenario requires on-the-fly rulebook construction, the scenario-segmentation-based framework depicted in Sections 4.1 and 4.2 become unavailable. As shown in Table 3, our unified sampler successfully identifies a large counterexample with a normalized error value of 0.959 (the maximum possible normalized error value is 1). Moreover, it identifies more than 100 types of counterexamples, demonstrating the effectiveness of our approach in scenarios with non-fixed rulebooks. However, for scenarios where the rulebooks are available in advance, the scenario-segmentation-based framework may still have an advantage. For instance, in Scenario 2, our scenario-segmentation-based framework achieves higher error values and a greater percentage of counterexamples than the unified-sampler-based approach. This is because the former creates a dedicated sampler for each time segment, allowing it to capture counterexamples more accurately.

Balance Between Exploitation and Exploration. There is a trade-off between exploitation and exploration during sampling. Our approach provides a flexible method to balance these two aspects. As shown in Algorithm 3, the balance coefficient δ is an input to the sampler, influencing the exploration term $\mathcal{T}$. A larger δ shifts the algorithm towards greater exploration. Figures 7 (a) to (d) illustrate the sampled points for Segment 1 of Scenario 2 with varying δ values. As δ increases, the samples become more uniformly distributed across the sampling space. Furthermore, Fig. 7 (g) plots the average normalized error (blue curve), representing exploitation, alongside the average standard deviations of the input features (orange curve), representing exploration. The trade-off between these two metrics is evident. Users can thus adjust the δ value according to their requirements, demonstrating the flexibility of our framework.

Additionally, Fig. 7 (d) to (f) visually show that the diversity of our algorithm lies in between that of MAB and Halton while attaining highest average e_{norm}. This shows that our algorithm can better identify counterexamples while maintaining a higher diversity of samples than MAB.

Quantitative Falsification Results. In the computation of e_{norm}, we only consider whether each rule is violated or not. To further assess the degree of violation, we compute the average violation score (VS) for all samples. Recall from Definition 3 that a lower VS indicates a more severe violation. Using Scenario 2 as an example, Table 4 shows that our approach achieves a similar degree of violation as the MAB algorithm while consistently attaining a lower VS compared to the Halton sampler.

Runtime Analysis. For the largest scenario, Scenario 3, we compute the average runtime to generate a sample. The average sampling times for our algorithm, the MAB sampler, and the Halton sampler are 8.84 seconds, 11.17 seconds, and 14.15 seconds, respectively. These results show the efficiency of our sampling algorithm.

7 Conclusion

Traditional approaches to multi-objective falsification of autonomous CPS fall short in capturing evolving priorities and objectives. To overcome this limitation, we introduced dynamic rulebooks to capture how objectives and their priorities change over time. In addition, we presented a falsification framework tailored for dynamic, multi-objective systems. Using experiments across various scenarios in the autonomous driving and human-robot interaction domains, we validated the effectiveness of our framework. Future work includes extending the application of dynamic rulebooks and our falsification algorithm to additional domains and real-world testing.

Acknowledgments. This work was supported in part by DARPA contract FA8750-23-C-0080 (ANSR), by Toyota and Nissan under the iCyPhy Center, by NSF grant 2303564, by C3DTI, and by Berkeley Deep Drive.

References

1. Abbas, H., Fainekos, G., Sankaranarayanan, S., Ivančić, F., Gupta, A.: Probabilistic temporal logic falsification of cyber-physical systems. ACM Trans. Embed. Comput. Syst. **12**(2s) (May 2013). https://doi.org/10.1145/2465787.2465797, https://doi.org/10.1145/2465787.2465797
2. Annpureddy, Y., Liu, C., Fainekos, G., Sankaranarayanan, S.: S-taliro: A tool for temporal logic falsification for hybrid systems. In: Abdulla, P.A., Leino, K.R.M. (eds.) Tools and Algorithms for the Construction and Analysis of Systems. pp. 254–257. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
3. Carpentier, A., Lazaric, A., Ghavamzadeh, M., Munos, R., Auer, P.: Upper-confidence-bound algorithms for active learning in multi-armed bandits. In: International Conference on Algorithmic Learning Theory. pp. 189–203. Springer (2011)
4. Censi, A., Slutsky, K., Wongpiromsarn, T., Yershov, D., Pendleton, S., Fu, J., Frazzoli, E.: Liability, ethics, and culture-aware behavior specification using rulebooks. In: 2019 International Conference on Robotics and Automation (ICRA). pp. 8536–8542. IEEE (2019)
5. Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: CARLA: An open urban driving simulator. In: Proceedings of the 1st Annual Conference on Robot Learning. pp. 1–16 (2017)
6. Dreossi, T., Fremont, D.J., Ghosh, S., Kim, E., Ravanbakhsh, H., Vazquez-Chanlatte, M., Seshia, S.A.: VerifAI: A toolkit for the formal design and analysis of artificial intelligence-based systems. In: International Conference on Computer Aided Verification. pp. 432–442. Springer (2019)
7. Fremont, D.J., Dreossi, T., Ghosh, S., Yue, X., Sangiovanni-Vincentelli, A.L., Seshia, S.A.: Scenic: A language for scenario specification and scene generation. In: Proceedings of the 40th annual ACM SIGPLAN conference on Programming Language Design and Implementation (PLDI) (June 2019)
8. Fremont, D.J., Kim, E., Dreossi, T., Ghosh, S., Yue, X., Sangiovanni-Vincentelli, A.L., Seshia, S.A.: Scenic: A language for scenario specification and data generation. Machine Learning **112**(10), 3805–3849 (2023)
9. Halton, J.H.: Algorithm 247: Radical-inverse quasi-random point sequence. Communications of the ACM **7**(12), 701–702 (1964)
10. Helou, B., Dusi, A., Collin, A., Mehdipour, N., Chen, Z., Lizarazo, C., Belta, C., Wongpiromsarn, T., Tebbens, R.D., Beijbom, O.: The reasonable crowd: Towards evidence-based and interpretable models of driving behavior. In: 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 6708–6715. IEEE (2021)
11. Maler, O., Nickovic, D.: Monitoring temporal properties of continuous signals. In: International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems. pp. 152–166. Springer (2004)
12. Puig, X., Undersander, E., Szot, A., Cote, M.D., Partsey, R., Yang, J., Desai, R., Clegg, A.W., Hlavac, M., Min, T., Gervet, T., Vondrus, V., Berges, V.P., Turner, J., Maksymets, O., Kira, Z., Kalakrishnan, M., Malik, J., Chaplot, D.S., Jain, U., Batra, D., Rai, A., Mottaghi, R.: Habitat 3.0: A co-habitat for humans, avatars and robots (2023)
13. Rubinstein, R.Y., Kroese, D.P.: The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation, and machine learning, vol. 133. Springer (2004)
14. Seshia, S.A., Sadigh, D., Sastry, S.S.: Toward verified artificial intelligence. Communications of the ACM **65**(7), 46–55 (2022)
15. Viswanadha, K., Indaheng, F., Wong, J., Kim, E., Kalvan, E., Pant, Y., Fremont, D.J., Seshia, S.A.: Addressing the IEEE AV test challenge with Scenic and VerifAI. In: 2021 IEEE International Conference on Artificial Intelligence Testing (AITest). pp. 136–142. IEEE (2021)

16. Viswanadha, K., Kim, E., Indaheng, F., Fremont, D.J., Seshia, S.A.: Parallel and multi-objective falsification with Scenic and VerifAI. In: Runtime Verification: 21st International Conference. pp. 265–276. Springer (2021)
17. Waga, M.: Falsification of cyber-physical systems with robustness-guided black-box checking. In: Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control. pp. 1–13 (2020)
18. Zhou, X., Gou, X., Huang, T., Yang, S.: Review on testing of cyber physical systems: Methods and testbeds. IEEE Access **6**, 52179–52194 (2018). https://doi.org/10.1109/ACCESS.2018.2869834