Multi-objective search for effective testing of Cyber-Physical Systems

Hugo Araujo[1], Gustavo Carvalho[2],
Mohammad Reza Mousavi[3], and Augusto Sampaio[4]

[1] Universidade Federal de Pernambuco - hlsa@cin.ufpe.br
[2] Universidade Federal de Pernambuco - ghpc@cin.ufpe.br
[3] University of Leicester - mm789@leicester.ac.uk
[4] Universidade Federal de Pernambuco - acas@cin.ufpe.br

Abstract. We propose a multi-objective strategy for finding effective inputs for fault detection in Cyber Physical Systems (CPSs). The main goal is to provide input signals for a system in such a way that they maximise the distance between the system's output and an ideal target, thus leading the system towards a fault; this is based on Genetic Algorithm and Simulated Annealing heuristics. Additionally, we take into consideration the discrete locations (of hybrid system models) and a notion of input diversity to increase coverage. We implement our strategy and present an empirical analysis to estimate its effectiveness.

Keywords: Cyber-Physical Systems, Search Based, Input Selection

1 Introduction

Cyber-Physical Systems (CPSs) integrate computational systems into their physical environments; components for products such as automobiles and airplanes [33] are examples of modern CPSs. In order to model the continuous and discrete dynamics often present in CPSs, hybrid models have been extensively used [6]. A typical type of CPS is a system where sensors feed input signals to a digital controller (discrete component) attached to physical actuators (continuous component) and also outputs continuous signals.

Such systems are complex since their design is typically multidisciplinary. It is not uncommon for a system component to deal with aspects of different subject areas such as computer science, physics and control engineering [22]. The importance of safety and reliability in such complex and heterogeneous systems warrant the need for further research into their verification.

Model-Based Testing techniques (MBT) can play an important role in the verification of these systems by providing precise mathematical assurances [39]. Particularly, one can design a test strategy based on a mathematical relation that decides whether the System Under Test (SUT) behaves as expected, with respect to a given specification; this is also known as a conformance relation [24].

However, finding effective inputs for detecting faults, i.e., witnesses for non-conformance, in a continuous system is not a straightforward task. Typically, one

needs a search algorithm optimised for the continuous domain in order to select values to maximise the goal, e.g., a witness to a conformance relation violation.

In the present work, we adopt the (τ,ϵ)-conformance notion [1,2]. Briefly speaking, under the same input stimuli, that are given for both specification and implementation models, the difference in the output behaviour of both systems is analysed and a distance metric is used to verify if the output behaviours of the specification and the implementation models are close enough to each other. We propose a multi-objective search for selecting inputs that violate this relation.

Our first search objective is defined as the observed distance between the output of the model and the ideal output. By maximising the distance between the outputs (of the system under test model and those of the ideal target), we can steer the system towards a fault or, more precisely, towards a challenging situation that maximises the possibility of a non-conforming verdict during conformance testing.

As for the second objective, we make use of the control elements found in hybrid system models to achieve structural coverage. Particularly, we consider the locations in a Hybrid Automata [21] model as a measure for coverage.

Finally, we propose a diversity notion as our third objective. We adopt a distance-based diversity metric that computes the Euclidean distance of previously generated inputs to generate new diverse inputs. We also make use of change point analysis in our diversity computation, so that we generate inputs that cover different areas and behave in different shapes.

The contributions of this work can be summarised as follows. We propose a multi-objective search strategy for input selection that (i) maximises the distance between the system's output and its ideal target, (ii) makes use of the control elements found in hybrid system models to achieve structural coverage, and (iii) employs a diversity metric to generate additional tests covering different areas and shapes. Furthermore, another important contribution is (iv) the empirical evaluation of these objectives in producing effective and efficient tests. We contrast our results against related approaches using examples from the literature. The formulation of the first and third objective in our context is, to the best of our knowledge, novel. Additionally, their combined usage in multi-objective search-based heuristics, and the particular way we use them for increased fault detection, is novel as well.

Section 2 considers related work. Section 3 provides the necessary background. Section 4 presents our strategy for finding inputs. Section 5 presents a case study and the results of the experiment we have performed. Finally, Section 6 gives a summary of our results and presents the next steps in our research agenda.

2 Related Work

In the literature, we can find several strategies for input selection for both discrete and continuous systems separately. For instance, discrete systems can be covered using structural notions such as node, edge and path coverage [9]. As for continuous systems, generating test data can usually be seen as an optimisation

problem, which can be solved using search techniques [40]. However, in the case of hybrid systems, which involve both discrete and continuous behaviour, only a few approaches have been proposed, and are discussed in the sequel.

One class of approaches is called property falsification, exemplified by tools such as Breach [17] and S-Taliro [10]. S-Taliro is a tool that offers an alternative solution for conformance testing of hybrid systems. Temporal verification is used to prove or falsify temporal logic properties of the system by searching for system behaviours that falsify the specification, i.e., counterexamples to Metric Temporal Logic (MTL) properties. Its conformance testing component uses the (τ, ϵ)-conformance notion [1,2], which is proposed by the same authors. As for input selection, it uses randomised testing based on stochastic optimisation techniques, such as Simulated Annealing [25], Genetic Algorithm [31], Ant Colony Optimisation [18] and Cross Entropy [38]. In our work, even though we offer fewer search-based techniques (Simulated Annealing and Genetic Algorithm), the modular structure of our solution allows for additional heuristics to be implemented. The main difference, however, is that our search is multi-objective; we consider 3 search objectives simultaneously: (i) maximising output distance from an ideal target, (ii) discrete coverage, and (iii) diversity.

A strategy that uses a notion of test diversity [29,30] for test suite generation and selection can be seen as complementary work. It uses a search algorithm, based on Hill-Climbing, that is guided to produce test outputs exhibiting a diverse set of signal features. Their approach outperforms Simulink Design Verifier (SLDV) and random test generation. Later, the same authors refined the strategy into one that considers output diversity (distance and feature based) as search criteria [28]. Unlike their strategy, we do not focus only on Simulink models nor only on outputs. We employ diversity in the input space as search criteria and consider a notion of distance from an ideal target for outputs. Moreover, we make use of change point analysis to achieve different shapes for the inputs.

As for coverage, a framework for conformance testing of hybrid systems has been proposed [15] to guide a test generation process. It focuses on state space coverage and how much of the reachable sets of the system are covered, using a notion called *star discrepancy*, which indicates how uniformly distributed are a set of testing points in the state space. As an extension [3], a coverage-based falsification strategy is presented. Instead of focusing on state coverage, a new strategy is developed based on input space coverage. The new strategy sub-divides the search space and takes random samples from these sub-regions, prioritising the ones with low robustness, where negative robustness indicates a property violation. We adopt a structural coverage as our main notion. However, the coverage of the input space is further emphasised by our diversity metric, which helps covering areas of the input space distant from the ones already covered.

3 Preliminaires

Cyber-Physical Systems [7] feature a tight integration of discrete and continuous dynamics; the semantics of CSPs can be suitably modelled as hybrid systems [16].

Here, we use hybrid automata [8] to model CPSs since it is a well-established and solid formalism, with an intuitive semantics, besides having tools supporting different analyses [10,14,19,36].

We use hybrid automata to capture the desired behaviour at a higher-level of abstraction, to make the input generation feasible. Our approach imposes no constraints on the concrete design or implementation of CPSs. Moreover, the semantics of many formalisms can be expressed in terms of hybrid automata [5]; hence, our approach can be applied to such formalisms as well.

3.1 Analysis of cyber-physical systems

We first consider a running example. Then we introduce a formal definition of hybrid automata, and finally we present a relation that captures a conformance notion of an implementation with respect to a specification.

Running example - DC-DC boost converter A DC-DC boost converter boosts the input voltage E to a higher output voltage. Figure 1 depicts the basic schematic of a boost converter. The system works by increasing and decreasing the inductor current. For that, the system has a switch that can be opened or closed. While the switch is closed, the current flows through the inductor generating a magnetic field. Once the switch is opened, the magnetic field is destroyed and the current must flow through the diode, transferring the accumulated energy into the capacitor. Since power must be conserved ($P = VI$), the decrease in the current means an increase in the voltage. This cycle is then repeated. Note that the control element of the boost converter transforms the otherwise continuous system into a hybrid one. For more details, see [22].

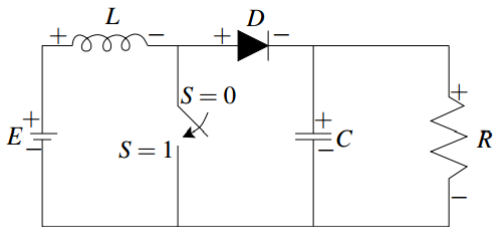


Fig. 1: DC-DC boost converter [22].

Hybrid automata Hybrid automata, defined below, can be seen as an extension of finite and timed automata. Guards, reset maps, invariants and specific dynamics for each location are added to these models, in order to allow the specification of continuous dynamics.

Definition 1 (Hybrid Automata [21]). *A hybrid automaton is defined as a tuple (Loc, V, (l_0,v_0), $\rightarrow$, I, F), where*

- *Loc is the finite set of locations;*
- *$V = V_I \uplus V_O$ is the set of continuous variables, where V_I and V_O denote the disjoint sets of input and output variables, respectively;*
- *l_0 denotes the initial location and v_0 is an initial valuation of V;*
- *$\rightarrow \subseteq Loc \times \mathcal{B}(V) \times Reset(V) \times Loc$ is the set of jumps, where:*
  - *$\mathcal{B}(V) \subseteq Val(V)$ indicates the guards under which the jump may be performed, and*
  - *$Reset(V) = \bigcup_{V' \subseteq V} Val(V')$ is the set of value assignments to the variables in V after the jump;*
- *$I : Loc \rightarrow \mathcal{B}(V)$ determines the allowed valuation of variables in each location (called the invariant of the location); and*
- *$F : Loc \rightarrow \mathcal{B}\left(V \cup \dot{V}\right)$ describes some constraints on variables and their derivatives and specifies the allowed continuous behaviour in each location.*

Locations are discrete states where each one can be viewed as a purely continuous system. Furthermore, the continuous behaviour of the entire hybrid system is captured by the valuation of a set V of continuous variables. We assume that V is partitioned into disjoint sets of input variables, denoted by V_I, and output variables, denoted by V_O. A jump represents a change in the current operating location. To perform a jump, the transition guard has to hold. Moreover, a jump is an immediate action, which does not require time to pass. During a jump event, the valuation of the continuous variables can be reset. Each location also contains a set of differential equations to describe how the continuous variables evolve in that location.

Fig. 2 shows the hybrid automaton of our running example. The four discrete states of the system are dependent on the switch (S) and diode (D) modes. The switch can be open or closed while the diode can be blocking or conducting. For instance, modes 1 and 3 represent the system state where the switch is open; in modes 2 and 4, the switch is closed. Analogously, the diode is conducting in modes 3 and 4 and blocking in modes 1 and 2. The inputs for the system are the switch S, the current I_{240} and the voltage V_{24}. The output parameters are the current I_{24} and the boosted output voltage V_{240}. Furthermore, Φ is the magnetic flux produced by the inductor, L is the inductance, q is the electric charge and C represents the capacitance.

Hybrid conformance As previously mentioned, the authors of [1,2] propose a conformance relation based on the output behaviour of a system specification and implementation models. This is formalised in a closeness relation (see Definition 2). This section is based on the theory presented in [2].

In practice, due to un-modelled physical occurrences such as noise and delays, the implementation behaviour often deviates in time and value with respect to the model [1]. The absence of margins of error can lead to undesired non-conforming verdicts due to intrinsic imprecision in measurement devices and calibration of

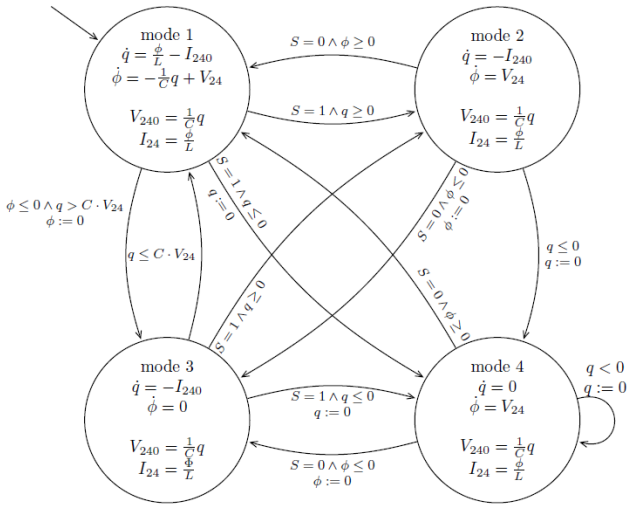


Fig. 2: Hybrid automaton of the DC-DC boost converter [4].

the implementation and testing infrastructure. Hence, in the (τ, ϵ)-conformance relation, a maximum temporal error of τ and spatial error of ϵ are allowed between the output signals (of the implementation and specification).

In the following definition, a trajectory captures the dynamical evolution of the system, representing its valuation through time. The notion of trajectory abstracts away from discrete locations. A trajectory y is a mapping $y : E \rightarrow Val(V)$, where $Val(V)$ denotes the valuation of a set of variables V, and E represents a set of the Hybrid Time Domain, which is a subset of $\mathbb{R}_+ \times \mathbb{N}$. A Hybrid Time is a tuple (t, j) corresponding to the point t in time and the number j of jumps. The set of all trajectories for a hybrid automata $\mathcal{HA}$ is denoted by Trajs($\mathcal{HA}$).

Definition 2 ((τ,ϵ)-closeness). *Consider a test duration $T \in \mathbb{R}_+$, a maximum number of jumps $J \in \mathbb{N}$, and $\tau, \epsilon > 0$; then two trajectories y_1 and y_2 are said to be (τ,ϵ)-close, denoted by $y_1 \approx_{(\tau,\epsilon)} y_2$, if 1 and 2 below hold.*

1. $\forall\, t : \mathbb{R}_+;\ i : \mathbb{N} \mid (t, i) \in dom(y_1) \wedge t \leq T \wedge i \leq J \bullet$
   $\exists\, s : \mathbb{R}_+;\ j : \mathbb{N} \mid (s, j) \in dom(y_2) \bullet$
   $\mid t - s \mid \leq \tau \wedge \mid\mid y_1(t, i) - y_2(s, j) \mid\mid \leq \epsilon.$

2. $\forall\, t : \mathbb{R}_+;\ i : \mathbb{N} \mid (t, i) \in dom(y_2) \wedge t \leq T \wedge i \leq J \bullet$
   $\exists\, s : \mathbb{R}_+;\ j : \mathbb{N} \mid (s, j) \in dom(y_1) \bullet$
   $\mid t - s \mid \leq \tau \wedge \mid\mid y_2(t, i) - y_1(s, j) \mid\mid \leq \epsilon.$

The notation $\mid e \mid$ stands for the absolute value of e, whilst $\mid\mid a - b \mid\mid$ stands for the (Euclidean) distance between a and b. A solution for a $\mathcal{HA}$ is a function $s : E \rightarrow Loc \times Val(V)$, which yields a location and a valuation given a Hybrid Time [2].

Definition 3 (Solution Pair). *Let u and y be two trajectories of types $E \to Val(V_I)$ and $E \to Val(V_O)$, respectively; (u, y) is a solution pair to a hybrid automaton $\mathcal{HA}$ if*

- $\mathrm{dom}(u) = \mathrm{dom}(y)$, *and*
- $\exists\, \phi : E \to Val(V) \mid \phi \in Trajs(\mathcal{HA}) \bullet dom(\phi) = dom(u) \wedge u = \phi \downarrow V_I \wedge y = \phi \downarrow V_O$, *where $y \downarrow V$ stands for the restriction of trajectory y to the set of variables V*

The notion of solution pair is necessary in order to abstract away from locations and distinguish between input and output trajectories. Two trajectories are considered a solution pair for a hybrid automata $\mathcal{HA}$, if there exists a trajectory for $\mathcal{HA}$ that captures the behaviour of both trajectories when it is restricted to input and output variables. We denote by $Sols(\mathcal{HA})$ the set of all Solution Pairs for $\mathcal{HA}$. Definition 4 formalises the (τ, ϵ)-conformance relation.

Definition 4 (Conformance Relation). *Consider two hybrid automata $\mathcal{HA}_1$ and $\mathcal{HA}_2$. Given a test duration $T \in \mathbb{R}_+$, a maximum number of jumps $J \in \mathbb{N}$, and $\tau, \epsilon > 0$, $\mathcal{HA}_2$* conforms to *$\mathcal{HA}_1$, denoted by $\mathcal{HA}_2 \approx_{(\tau,\epsilon)} \mathcal{HA}_1$, if and only if*

$$\forall\, u : E \to Val(V_I);\ y_1 : E \to Val(V_O) \mid (u, y_1) \in Sols(\mathcal{HA}_1) \bullet$$
$$\exists\, y_2 : E \to Val(V_O) \mid (u, y_2) \in Sols(\mathcal{HA}_2) \bullet y_1 \approx_{(\tau,\epsilon)} y_2$$

In the above definition, T and J are implicitly used in the expression $y_1 \approx_{(\tau,\epsilon)} y_2$.

4 Finding Inputs via search-based heuristics

In this section, we present our strategy for input selection: a modular and scalable process for finding inputs that are directed towards detecting non-conformance.

The main motivation behind our strategy is an efficient way to generate inputs that not only provide structural coverage and maximise diversity metrics but also maximise the possibility of finding faults. Particularly, we consider a notion called critical epsilon, which is related to the distance between two trajectories, e.g., ouptut and reference signals. Our search is performed in such a way to maximise the critical epsilon, thus, also maximising the chances to detect non-conformance. We emphasise that this particular combination is novel in this domain and our experiments (see Section 5) show that it can lead to effective test cases.

4.1 Search based inputs and critical epsilon

Given two (reference and output) signals in the specification and a fixed τ, we denote by critical epsilon the smallest ϵ that makes the two signals (τ,ϵ)-close. We formally define it as follows.

Definition 5 (Critical Epsilon). *Consider two trajectories y_1 and y_2, a test duration $T \in \mathbb{R}_+$, a maximum number of jumps $J \in \mathbb{N}$, then, the critical epsilon for y_1, y_2 and a given $\tau > 0$ is*

$$ce(\tau, y_1, y_2) = min\{\epsilon : \mathbb{R}_+ \mid y_1 \approx_{(\tau,\epsilon)} y2 \ \bullet \epsilon\}$$

Thus, by fixing the temporal margins, it is possible to build a function that computes the critical epsilon. We use this function to select the input points that generate the highest critical epsilon (see Definition 6); such inputs steer the implementation towards the area in which it is more likely to show deviating behaviour and thus a non-conforming verdict.

Definition 6 (Highest Critical Epsilon). *Consider two hybrid automata* $\mathcal{HA}_1$ *and* $\mathcal{HA}_2$*, a set of inputs* $U : E \rightarrow Val(V_I)$ *and outputs* $y_1^u, y_2^u : E \rightarrow Val(V_O) \mid (u, y_1) \in Sols(\mathcal{HA}_1) \wedge (u, y_2) \in Sols(\mathcal{HA}_2) \wedge u \in U$*, then, the highest critical epsilon for* U*,* $\mathcal{HA}_1$ *and* $\mathcal{HA}_2$ *and a given* $\tau > 0$ *is:*

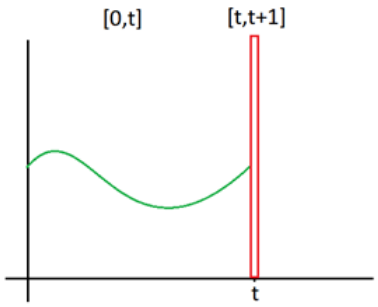
$$hce(\tau, U, \mathcal{HA}_1, \mathcal{HA}_2) = max\{u : U \bullet ce(\tau, y_1^u, y_2^u)\}$$

In summary, our strategy consists of searching for inputs that yield a greater spatial distance between the reference and the system output.

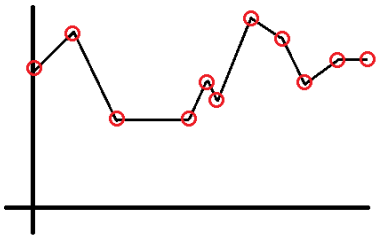
However, since continuous input spaces are infinite by definition, it is not feasible to consider every possible input. A search must be performed, which reduces test-case generation into a global optimisation problem.

For that, we have implemented two approaches: Simulated Annealing and Genetic Algorithms, which are well established probabilistic algorithms for computing global optima [20,31]. Given a function f, they attempt to heuristically approximate the global maxima or minima of f. However, their heuristic nature brings a certain degree of imprecision; this is mitigated by adjusting the parameters in such a way to find a compromise between accuracy and performance.

Figure 3a shows the core idea behind the input generation. In summary, given an input, whose time interval is $[0, t]$, we search for the input value at $(t + 1)$ that better satisfies our search metrics, e.g., the highest critical epsilon. Note that the initial input value (where $t = 0$) must be given. This process is repeated until the end of the simulation.



(a) Input generated by a search heuristic.



(b) Input generated and its change point analysis.

Fig. 3: Example of inputs.

Since the basic algorithm only searches for input values for one timestep at a time $(t+1, t+2, ...)$, it is possible that a choice that gives a lower critical epsilon at $(t+1)$ (and therefore is not selected), might result in a non-conformance verdict in the future (e.g., at $t+10$). This will not be detected by the algorithm and then the non-conformance will be missed. Thus, testing the system using multiple input trajectories should increase the odds of detecting faults, which led us to proposing notions of coverage and diversity to remedy this.

A drawback of this strategy is that it can yield unrealistic inputs. For instance, the variation rate of the resulting signal can be impractical. However, unrealistic inputs do not necessarily mean invalid inputs. The algorithm searches for inputs that fit the input domain. For instance, consider a turbine with a sensor that measures wind speed in the range of 0 to 100 m/s as input and that our algorithm finds a fault whenever the wind changes its speed from 0 to 100 m/s in 0.01 seconds. Although a fault was detected, such a high variance in wind speed might have never been recorded before and could be considered unrealistic. We consider this input unrealistic, but not strictly invalid.

These scenarios are then useful to further constrain the model. One solution we propose is for the developers to refine the model to disallow such inputs by defining preconditions (e.g., bounds of derivatives) on inputs.

4.2 Notions of coverage

Test coverage can be used as an indicator for measuring test quality [23], and a positive relationship between test coverage and code reliability has been empirically established [27]. However, coverage has a cost associated with it, and achieving high degrees of coverage is not always feasible or necessary [32]. A contribution of this work is the integration of coverage criteria into our strategy. In this section, we show how we have implemented structural coverage criteria that are able to impose some control on the input selection algorithm, ensuring that the generated test cases cover particular elements of the hybrid automata.

We have considered three types of structural coverage: discrete state, edge and prime path coverage [9]. Discrete state guarantees that each discrete state in the model will be visited by our strategy. Edge coverage is achieved by triggering all transitions. Finally, path coverage is a stronger notion of coverage that aims to cover a particular set of transitions in the model and encompasses both node (i.e., discrete state) and edge coverage. In this work, we emphasize discrete state coverage, due to the cost effectiveness that we obtained in our experiments.

Discrete State Coverage We adopt discrete state coverage, since a critical epsilon-based input does not guarantee that the system runs through each and every state. Given that a system can have issues in multiple states, visiting all of them can uncover non-conformance.

In this search strategy, we guide the system towards each and every discrete state (i.e., location) present in the specification. Once we move to an uncovered state, we switch the priority to finding the highest critical epsilon. This way we

guarantee at least one test per location. The main idea is to generate inputs that will guide the system towards each discrete state as quickly as possible and, then, search for problems that might arise once the system is in those specific states.

In order to guide the system towards the desired discrete state, we require information on its boundaries and information on transitions obtained through the hybrid automaton specification provided by the user.

4.3 A notion of diversity

As an additional criterion for our search we consider a diversity metric. More precisely, we adopt a distance-based diversity metric that computes the Euclidean distance of previously generated inputs to generate a new diverse input. Diversity is employed alongside the critical epsilon search.

However, a pure Euclidean distance evaluation on all points could lead to inputs that have the same shape and simple spatial shift, such as two constant signals that are distant from each other. To avoid this, our diversity metric takes in consideration the change points in the input signals. A change point analysis [35] detects sampling points in a trajectory in which there is an abrupt change. Thus, to generate more interesting and effective inputs, we only employ the diversity criteria to the change points of the previously generated ones.

In order to properly employ diversity, one needs inputs generated beforehand, from which the new inputs can be diversified. Our strategy is as follows. In the first step, a core group of inputs are generated using the highest critical epsilon metric with discrete coverage. From this group, we execute the second step, where more tests are generated using a combination of diversity metric with critical epsilon. Consider the trajectory in Figure 3b as an input generated in the first step (coverage + critical epsilon) and its change points, which are circled around. As the new input is being produced in the second step, the priority assigned to diversity and critical epsilon changes proportionally to the distance to a change point. The closer the new input gets to a change point, the more we increase the priority of the diversity metric and decrease the priority of critical epsilon, so that the new input will distance itself from regions covered by old inputs. Analogously, the further the new input gets from the change points, the more we decrease the priority of diversity and increase the priority of critical epsilon.

Lastly, employing diversity means that we can generate diverse inputs indefinitely. Hence, we have decided to let the user set a maximum number of inputs as stopping criteria; we plan to employ a more systematic approach in the future.

4.4 Mechanisation

Currently, our tool, HyConf, can read Simulink models and perform conformance testing using (τ, ϵ)-conformance notion based on user-defined parameters (τ and ϵ). Given a fixed τ, it can compute the critical epsilon for the user. For the input generation, the tool requires information about the discrete locations, which is not automatically inferred from the Simulink models. We plan to handle this

transition automatically, using, for instance, an algorithm [5] for conversion between Simulink and hybrid automata.

The strategy requires two signals in the specification: a command (or reference) signal, denoting the ideal target of the system, and an output signal, denoting the current state of the system. The choice of these signals is domain specific and requires some knowledge of the specification. A pseudo-code with additional explanation of how our algorithm works is available in the extended version of this paper [11] (Appendix A).

5 Empirical analysis

In this section, we describe the experiments performed using the proposed strategy. Section 5.1 briefly describes the case study used in the experiment; Section 5.2 details the experimental plan along with its methodology and threats to validity; and section 5.3 presents the results of the experiment.

5.1 Case study

In addition to the running example, we use a case study based on an automotive pneumatic suspension system [34]. The system's goal is to increase driving comfort by adjusting the chassis level to compensate for road disturbances. This is achieved by a pneumatic suspension that connects the valves attached to each wheel to a compressor and an escape valve.

The system aims to keep the chassis level as close as possible to a defined set point in each of the four wheels. The decision to increase or decrease the chassis level is based on the tolerance intervals defined for each wheel. The full automaton of our version of this system [11] (Appendix B) and its behaviour slightly differs from the original one[34]. The original model contains some unsupported features by the tools we use, such as synchronised parallel components and non-deterministic differential equations. We have serialised the model (by computing the overall behaviour of the constituent hybrid automata), but kept the overall behaviour intact, except for the removal of non-determinism. We have changed the non-deterministic assignments to input assignments and, thus, we added inputs that can take the same values within the original intervals and are now assigned directly to the corresponding variable derivative. The final model contains 4 locations with several differential equations each.

5.2 Experimental plan

The main goal of this study is the evaluation of strategies for input generation to test Cyber-Physical Systems (CPS). Additionally, this experiment aims to verify whether the strategy we implemented in our tool, HyConf, is more effective and efficient in terms of performance compared with the alternatives found in the literature. Another motivation behind this study is the fact that there are few empirical and controlled experiments to evaluate the efficacy of MBT

(Model-Based Testing) tools in regard to Cyber-Physical Systems. We compare our strategy against another tool called S-Taliro and also against random input generation, which can serve as a baseline measurement. Additional details of the experimental plan can be found in [11] (Appendix C). The research questions are:

- **RQ1:** Can HyConf detect more faults than the alternatives?
- **RQ2:** Can HyConf detect faults faster than the alternatives?

We analyse the effectiveness of our strategies using mutation analysis. The mutation operators used in this experiment were chosen based on a study on mutation operators for Simulink models [12]. In total, we inserted 82 and 105 faults in the boost converter and suspension system models. The mutation score is used to determine the effectiveness of the strategies (RQ1). In this experiment, the strategy that kills more mutants is deemed more effective. In order to assert efficiency (RQ2), however, we collected the number of time steps it takes for the test suites to kill mutants, and then we compute the median. The strategy with lower median is considered the faster one.

Methodology In order to answer our research questions, we define the metrics MS, which represents the Mutation Score, and Average Time of Faults Detected, ATFD, which is an extension of the APFD metric (Average Percentage of Faults Detected) [37]. The experiment is followed by a statistical analysis and a comparison of the yielded results.

The mutation score can be obtained by dividing the number of mutants killed by the total number of mutants created. In this work, we do not distinguish equivalent mutants, i.e., those mutants that conform to the specification. It is generally difficult to check whether the mutant is equivalent in a continuous domain and, thus, we assume that all mutants are not equivalent.

The ATFD metric tells us which test suite can detect mutants faster and is formally defined as follows.

Definition 7 (Average Time of Faults Detected). *Let T be a test suite containing n timesteps and let M be a set of m models with a single distinct mutant each. Let T' be an ordering of T. Let TS_i be the first time step in T' that detects the fault i. The ATFD for T' is:*

$$ATFD_{T'} = 1 - \frac{TS_1 + TS_2 + ... + TS_m}{nm} + \frac{1}{2n}$$

Similarly to APFD, ATFD can vary from 0 to 1 and a higher ATFD indicates a faster fault-detection rate. One can see the ATFD as an APFD where each time step is a distinct test case. However, if a test suite T cannot detect a fault i, then we assign to TS_i the max number of timesteps in that test suite (n). Here, timesteps can serve as time measurement, since to obtain the correct simulation time, one only needs to multiply the number of timesteps by the sampling rate.

In this experiment, each strategy creates inputs for 2 models. Due to the random nature of the search algorithms used in these tools and also to grant statistical significance, each strategy was executed 30 times for each model. The first model is the pneumatic suspension system and the second one is the boost converter (see Figure 5 [11] and Figure 2, respectively). Once the inputs were created, we performed mutation testing analysis in order to determine which strategy was more effective and accept or reject Hypothesis A (see below). Additionally, the time steps will be recorded during the execution of each strategy. These measurements were then used to assert Hypothesis B (see below).

The stopping criteria for our diversity notion in this experiment (variable *maxAdditionalInputs* in Figure 4) is the generation of 20 inputs. For S-Taliro we have matched the same number of inputs. As for the random strategy, due to its much faster input generation capabilities, instead of limiting the number of inputs generated, we let it run for the same amount of time as the slowest approach. We believe this is a fair approach to all strategies.

Hypotheses Hypotheses A and B aim to evaluate the research questions that have been explained previously. For this, null hypotheses are defined, which states that there is no difference between the strategies being analysed. This experiment aims to refute such hypotheses. Thus, alternative hypotheses are also defined, which have a complementary role to the null hypotheses, and can be accepted in case its counterpart hypotheses are rejected. We define 4 hypotheses: A0 and A1 (null and alternate, respectively), compares whether the mutation score (MS) obtained by using our strategy (HyConf) is less than or equal to the mutation score obtained by using the other strategies (OTH). Analogously, hypotheses B0 and B1 (null and alternate, respectively) consider ATFD.

$$H_{A0} : MS_{HyConf} \leq MS_{OTH} \qquad H_{A1} : MS_{HyConf} > MS_{OTH}$$
$$H_{B0} : ATFD_{HyConf} \leq ATFD_{OTH} \qquad H_{B1} \quad ATFD_{HyConf} > ATFD_{OTH}$$

Threats to validity Here we list the threats to validity that apply to this experiment. As **Internal Validity**, the mutation operators used in this experiment were chosen based on a study on mutation operators for Simulink models [12]. The number of inserted faults is decided manually, based on the complexity of each system. Furthermore, there is no limit to the amount of inputs the random approach can generate and this can be a threat to fairness amongst strategies. Thus, we have decided to let this strategy run for the same time as the slowest approach. Concerning **External Validity**, this experiment only considers 2, relatively small, examples; we cannot generalise the outcome of this experiment for a general class of CPSs. Besides, since we introduced the faults ourselves, the mutants may also not represent real world faults. As **Construct Validity**, assessment of equivalent mutants for CPS was not performed. In this case, we assume that all mutants could have been detected. Moreover, the values for τ and ϵ have a direct impact on the results: sufficiently large values would have detected all mutants and small values would have detected none. The values we have chosen are based on prior experiments and domain knowledge.

Statistical analysis The purpose of this analysis is to verify whether one should reject or accept the null and alternative hypotheses. Here, we used the RStudio, where the comparison was made between averages MS and ATFD computed using HyConf, S-Taliro and random input generation. Since our samples follow a normal distribution, the "t-test" statistical technique with a p-value < 0.05 and level of confidence of 95% is used to analyse our data.

5.3 Results

To answer the first research question, we ran the experiment 30 times for each pair (strategy $\times$ model) and analysed the mutation score (see Table 1). Despite the slight input variations due to the randomness of the employed heuristics, the number of mutants detected by HyConf were the same for each run of the experiment, and similarly for S-Taliro; this was not the case for random testing. Each group of random inputs killed a different number of mutants; in this case, we show the median score. For our tool, we show the results of using different criteria combinations: highest critical epsilon (HCE), coverage and diversity. As can be observed, HyConf consistently detected more mutants when it employs all three combined.

	Mutation Score		ATFD	
	BC	SS	BC	SS
HyConf (SA): HCE + Coverage	63/82	81/105	0.325	0.312
HyConf (SA): HCE + Diversity	55/82	68/105	0.277	0.281
HyConf (SA): HCE + Coverage + Diversity	69/82	88/105	0.364	0.362
HyConf (GA): HCE + Coverage	64/82	85/105	0.342	0.339
HyConf (GA): HCE + Diversity	59/82	70/105	0.301	0.314
HyConf (GA): HCE + Coverage + Diversity	71/82	94/105	0.373	0.371
S-Taliro (SA)	60/82	76/105	0.311	0.293
S-Taliro (GA)	62/82	80/105	0.327	0.309
Random Inputs	43/82	61/105	0.228	0.222

Table 1: Experiment results.

Analogously, to answer the second question, we computed the amount of time steps necessary to detect each mutant. We used this information to calculate the ATFD values. The median values for the 30 execution are also shown in Table 1. For visualisation purposes, we refer to a boxplot format of this data [11] (Appendix D).

We should mention that each model was simulated for a maximum of 10 time units using a sampling rate of 0.01, which gives us 1000 time steps for each created input. Since we are not interested in prioritisation of the test suites, they had a randomised order in each execution.

Furthermore, Table 2 shows the statistical test results obtained from comparing the ATFD metric. Due to the lack of variation in the collected mutation score

(i.e, the HyConf and S-Taliro were constant), a t-test could not be performed to evaluate the MS metric; however, the results are clear.

	Boost Converter	Suspension
HyConf (SA) v. S-Taliro (SA)	$6.11 * 10^{-08}$	$3.98 * 10^{-14}$
HyConf (GA) v. S-Taliro (GA)	$6.86 * 10^{-08}$	$7.74 * 10^{-14}$
HyConf (GA) v. Random	$8.73 * 10^{-10}$	$5.21 * 10^{-15}$

Table 2: Test results (p-values)

The tests outcome indicate that HyConf performed better than the alternatives, and statistically, the results are significant. Thus, based on the data shown, we can reject the null hypotheses H_{A0} and H_{B0}, and accept their alternatives which tell us that our strategy can obtain a higher mutation score than the other tools and also a higher ATFD. Another conclusion is that even though Genetic Algorithm obtained a higher mutation score and ATFD compared to Simulated Annealing, it has a higher computation cost. This trade-off is left for the engineer to decide.

6 Conclusions and future work

In this work we have proposed a strategy for generating fault-oriented inputs for Cyber-Physical Systems. The idea behind these inputs is to maximise the distance between a system's output and its ideal target, thus, leading to fault.

In order to generate inputs for continuous systems, a search based approach is often necessary; in our case, we adopt simulated annealing and genetic algorithm. We look for inputs that lead to a potential conformance violation, particularly with respect to the (τ, ϵ)-conformance notion. Also, we make use of a discrete coverage notion to find inputs that can guide the system towards new locations and we also employ a diversity metric into our input selection strategy. By doing this, we aim to increase coverage and find faults that are more difficult to detect.

Moreover, we have conducted a controlled experiment to compare the strategy we propose with related alternatives. This was performed on two distinct systems: a boost converter and a pneumatic suspension system. Overall, our approach produced evidence of superior fault detection capabilities and efficiency.

As future work, we plan to further improve our input generation strategy. For instance, we are studying additional types of coverage that we can consider and which other types of parameters we can use to tune our multi-objective search. With our diversity metric, we can generate inputs indefinitely. Thus, we let the user define a maximum number but we plan to use a more systematic approach in the future. Additionally, some of the steps in this strategy can be mechanised further and fully integrated into our tool. Finally, we also plan to integrate our strategy with the NAT2TEST [13] framework, which would allow us to define a test strategy for CPS based on natural-language specifications.

References

1. Abbas, H., Hoxha, B., Fainekos, G.E., Deshmukh, J.V., Kapinski, J., Ueda, K.: Conformance testing as falsification for cyber-physical systems. In: Proceedings of the ACM/IEEE 5th International Conference on Cyber-Physical Systems (ICCPS 2014). p. 211. IEEE (2014)
2. Abbas, H.Y.: Test-based falsification and conformance testing for cyber-physical systems. Ph.D. thesis, Arizona State University (2015)
3. Adimoolam, A., Dang, T., Donzé, A., Kapinski, J., Jin, X.: Classification and coverage-based falsification for embedded control systems. In: International Conference on Computer Aided Verification. pp. 483–503. Springer (2017)
4. Aerts, A.: Model-based design and testing of mechatronic systems: an industrial case study. Master's thesis, Eindhoven University of Technology, Eindhoven, Netherlands (2016)
5. Agrawal, A., Simon, G., Karsai, G.: Semantic translation of simulink/stateflow models to hybrid automata using graph transformations. Electronic Notes in Theoretical Computer Science **109**, 43–56 (2004)
6. Alur, R.: Principles of cyber-physical systems. MIT Press (2015)
7. Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T.A., Ho, P.H., Nicollin, X., Olivero, A., Sifakis, J., Yovine, S.: The algorithmic analysis of hybrid systems. Theoretical computer science **138**(1), 3–34 (1995)
8. Alur, R., Courcoubetis, C., Henzinger, T.A., Ho, P.H.: Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In: Hybrid systems, pp. 209–229. Springer (1993)
9. Ammann, P., Offutt, J.: Introduction to software testing. Cambridge University Press (2016)
10. Annpureddy, Y., Liu, C., Fainekos, G., Sankaranarayanan, S.: S-taliro: A tool for temporal logic falsification for hybrid systems. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 254–257. Springer (2011)
11. Araujo, H., Carvalho, G., Mousavi, M., Sampaio, A.: Multi-objective search for effective testing of Cyber-Physical Systems: Extended Report. `https://www.cs.le.ac.uk/people/mm789/pub/sefm-2019-full.pdf` (2019)
12. Binh, N.T., et al.: Mutation operators for simulink models. In: Knowledge and Systems Engineering (KSE), 2012 Fourth International Conference on. pp. 54–59. IEEE (2012)
13. Carvalho, G., Barros, F., Carvalho, A., Cavalcanti, A., Mota, A., Sampaio, A.: Nat2test tool: From natural language requirements to test cases based on csp. In: Software Engineering and Formal Methods, pp. 283–290. Springer (2015)
14. Chen, X., Ábrahám, E., Sankaranarayanan, S.: Flow*: An analyzer for non-linear hybrid systems. In: International Conference on Computer Aided Verification. pp. 258–263. Springer (2013)
15. Dang, T., Nahhal, T.: Coverage-guided test generation for continuous and hybrid systems. Formal Methods in System Design **34**(2), 183–213 (2009)
16. De Schutter, B., Heemels, W., Lunze, J., Prieur, C., et al.: Survey of modeling, analysis, and control of hybrid systems. Handbook of Hybrid Systems Control–Theory, Tools, Applications pp. 31–55 (2009)
17. Donzé, A.: Breach, a toolbox for verification and parameter synthesis of hybrid systems. In: International Conference on Computer Aided Verification. pp. 167–170. Springer (2010)

18. Dorigo, M., Birattari, M.: Ant colony optimization. Springer (2010)
19. Frehse, G., Le Guernic, C., Donzé, A., Cotton, S., Ray, R., Lebeltel, O., Ripado, R., Girard, A., Dang, T., Maler, O.: Spaceex: Scalable verification of hybrid systems. In: International Conference on Computer Aided Verification. pp. 379–395. Springer (2011)
20. Gelfand, S.B., Mitter, S.K.: Analysis of simulated annealing for optimization. In: Decision and Control, 1985 24th IEEE Conference on. vol. 24, pp. 779–786. IEEE (1985)
21. Goebel, R., Sanfelice, R.G., Teel, A.R.: Hybrid dynamical systems. IEEE Control Systems **29**(2), 28–93 (2009)
22. Heemels, W., De Schutter, B.: Modeling and control of hybrid dynamical systems. TU Eindhoven, Lecture notes course 4K160 (2013)
23. Horgan, J.R., London, S., Lyu, M.R.: Achieving software quality with testing coverage measures. Computer **27**(9), 60–69 (1994)
24. Khakpour, N., Mousavi, M.R.: Notions of Conformance Testing for Cyber-Physical Systems: Overview and Roadmap (Invited Paper). In: Aceto, L., de Frutos Escrig, D. (eds.) 26th International Conference on Concurrency Theory (CONCUR 2015). Leibniz International Proceedings in Informatics (LIPIcs), vol. 42, pp. 18–40. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2015)
25. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. science **220**(4598), 671–680 (1983)
26. Konak, A., Coit, D.W., Smith, A.E.: Multi-objective optimization using genetic algorithms: A tutorial. Reliability Engineering & System Safety **91**(9), 992–1007 (2006)
27. Malaiya, Y.K., Li, M.N., Bieman, J.M., Karcich, R., Skibbe, B., et al.: The relationship between test coverage and reliability. In: Proceedings of 1994 IEEE International Symposium on Software Reliability Engineering. pp. 186–195. IEEE (1994)
28. Matinnejad, R., Nejati, S., Briand, L., Bruckmann, T.: Test generation and test prioritization for simulink models with dynamic behavior. IEEE Transactions on Software Engineering (2018)
29. Matinnejad, R., Nejati, S., Briand, L.C., Bruckmann, T.: Effective test suites for mixed discrete-continuous stateflow controllers. In: Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering. pp. 84–95. ACM (2015)
30. Matinnejad, R., Nejati, S., Briand, L.C., Bruckmann, T.: Automated test suite generation for time-continuous simulink models. In: proceedings of the 38th International Conference on Software Engineering. pp. 595–606. ACM (2016)
31. Mitchell, M.: An introduction to genetic algorithms. MIT press (1998)
32. Mockus, A., Nagappan, N., Dinh-Trong, T.T.: Test coverage and post-verification defects: A multiple case study. In: Empirical Software Engineering and Measurement, 2009. ESEM 2009. 3rd International Symposium on. pp. 291–301. IEEE (2009)
33. Mosterman, P.J., Zander, J.: Cyber-physical systems challenges: a needs analysis for collaborating embedded software systems. Software & Systems Modeling **15**(1), 5–16 (2016)
34. Müller, O., Stauner, T.: Modelling and verification using linear hybrid automata–a case study. Mathematical and Computer Modelling of Dynamical Systems **6**(1), 71–89 (2000)
35. Picard, D.: Testing and estimating change-points in time series. Advances in applied probability **17**(4), 841–867 (1985)

36. Platzer, A., Quesel, J.D.: Keymaera: A hybrid theorem prover for hybrid systems (system description). In: International Joint Conference on Automated Reasoning. pp. 171–178. Springer (2008)
37. Rothermel, G., Untch, R.H., Chu, C., Harrold, M.J.: Test case prioritization: An empirical study. In: Software Maintenance, 1999.(ICSM'99) Proceedings. IEEE International Conference on. pp. 179–188. IEEE (1999)
38. Rubinstein, R.Y., Kroese, D.P.: The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning. Springer Science & Business Media (2013)
39. Tretmans, J.: Formal methods and testing. chap. Model Based Testing with Labelled Transition Systems, pp. 1–38. Springer-Verlag, Berlin, Heidelberg (2008)
40. Windisch, A., Al Moubayed, N.: Signal generation for search-based testing of continuous systems. In: Software Testing, Verification and Validation Workshops, 2009. ICSTW'09. International Conference on. pp. 121–130. IEEE (2009)

A Strategy pseudo-code

Figure 4 shows the pseudo-code for our strategy. Given the set of locations in a hybrid automaton and the initial value for the input, the algorithm uses a search-based heuristic, e.g., Simulated Annealing, to find the next value for the input signal that results in either a change of locations or the highest critical epsilon. The output of this algorithm is a discretised input.

```
00 function main(){
01   foreach location in HA{
02     testcase = createInput(location, HA, initialInput);
03     testSuite.add(testcase);
04   }
05   for (i = 0; i <= maxAdditionalInputs; i++) {
06     testcase = createDiverseInput(testSuite, initialInput);
07     testSuite.add(testcase);
08   }
09   return testSuite;
10 }
11
12
13 function createInput(location, HA, input){
14   for (i = 0; i <= simulationEndTime; i++) {
15     if (system.currentLocation ≠ location) {
16       inputIteration = search(guideToLocation(location, HA), criticalEpsilon(), input);
17     }else{
18       inputIteration = search(criticalEpsilon(), input);
19     }
20     input.append(inputIteration);
21   }
22   return input;
23 }
24
25
26 function createDiverseInput(testSuite, initialInput){
27   changePoints = changePointAnalysis(testSuite);
28   for (i = 0; i <= simulationEndTime; i++) {
29     foreach changePoint in changePoints{
30       d = euclideanDistance(input, changePoint);
31       changePriority(criticalWeight, d);
32       changePriority(diversityWeight, 1 / d);
33     }
34     inputIteration = search(diversity(), criticalEpsilon(), diversityWeight, criticalWeight);
35     }
36     input.append(inputIteration);
37   }
38   return input;
39 }
```

Fig. 4: Pseudo-code used in HyConf.

The algorithm works as follows. It first creates a core group of inputs generated based on critical epsilon and discrete coverage (lines 01 to 04). This group of inputs is typically small (the same number of locations in the HA) but very effective. Consider the running example depicted in Figure 1, since there are 4 states, our strategy generates a core group of 4 input trajectories. Notice that, for the initial state, the algorithm only needs to prioritise critical epsilon. For the remaining states, the algorithm takes in consideration the possible values of the

variables in order to enter the state and the path it can take. For instance, in order to cover the state *"mode 2"* from the initial state (*"mode 1"*), the switch must be connected ($S = 0$) and the algorithm searches for inputs where the electric charge is greater than zero ($q \geq 0$). Once these two criteria have been met, the algorithm detects it has entered the state *"mode 2"* and only focus on critical epsilon (lines 15 to 19 in Figure 4).

After that, it uses a diversity metric coupled with critical epsilon to find inputs that are distant from the the ones already generated (lines 05 to 08). As mentioned in Section 4.3, the diversity only considers the change points of past inputs (line 27). The weight of the critical epsilon metric is proportional to the distance to the change points while the weight of the diversity metric is inversely proportional (lines 29 and 33). Thus, as the new input point being created approaches the position of a change point, the search increases the priority of the diversity metric and lowers the priority of the critical epsilon. It is worth mentioning that the initial point in every input is always a change point, thus the initial point for the new inputs being created is always distant from the existent ones.

Being a multi-objective search [26] means that the objectives are meant to be fulfilled concurrently. Whenever the search heuristic uses *guideToLocation*() or *euclideanDistance*() as a metric, it also takes in consideration *criticalEpsilon*().

B Case Study - Suspension System

The automaton we present here (see Figure 5) is a slightly modified version of the original one presented in [34], since the original model contains some unsupported features by the tools we use, such as synchronised parallel components and non-deterministic differential equations. We have serialised the model (by computing the overall behaviour of the constituent hybrid automata), but kept the overall behaviour intact, except for the removal of non-determinism. We have changed the non-deterministic assignments to input assignments, and thus, we added inputs that can take the same values within the original intervals and are now assigned directly to the corresponding variable derivative. The model we present here contains 4 locations with several differential equations each.

The system aims to keep the chassis level as close as possible to a defined set point in each of the four wheels. The decision to increase or decrease the chassis level is based on the tolerance intervals defined for each wheel. We consider $[sp + otl, sp + otu]$ and $[sp + itl, sp + itu]$ as the outer and inner tolerance intervals, respectively. Here, *sp* represents the set point, which is the target value of the chassis level, and *itu, itl, otu* and *otl* represent the inner and outer tolerance thresholds along with their respective upper and lower values.

The system receives 3 inputs and it outputs the current chassis level c. The inputs are *dist*, *cp* and *ev*. The first (*dist*) corresponds to the disturbance level coming from the environment, which indicates road perturbations such as small depressions or elevations. The last two (*cp* and *ev*) dictate the change in the chassis level performed by the compressor and the escape valve, respectively.

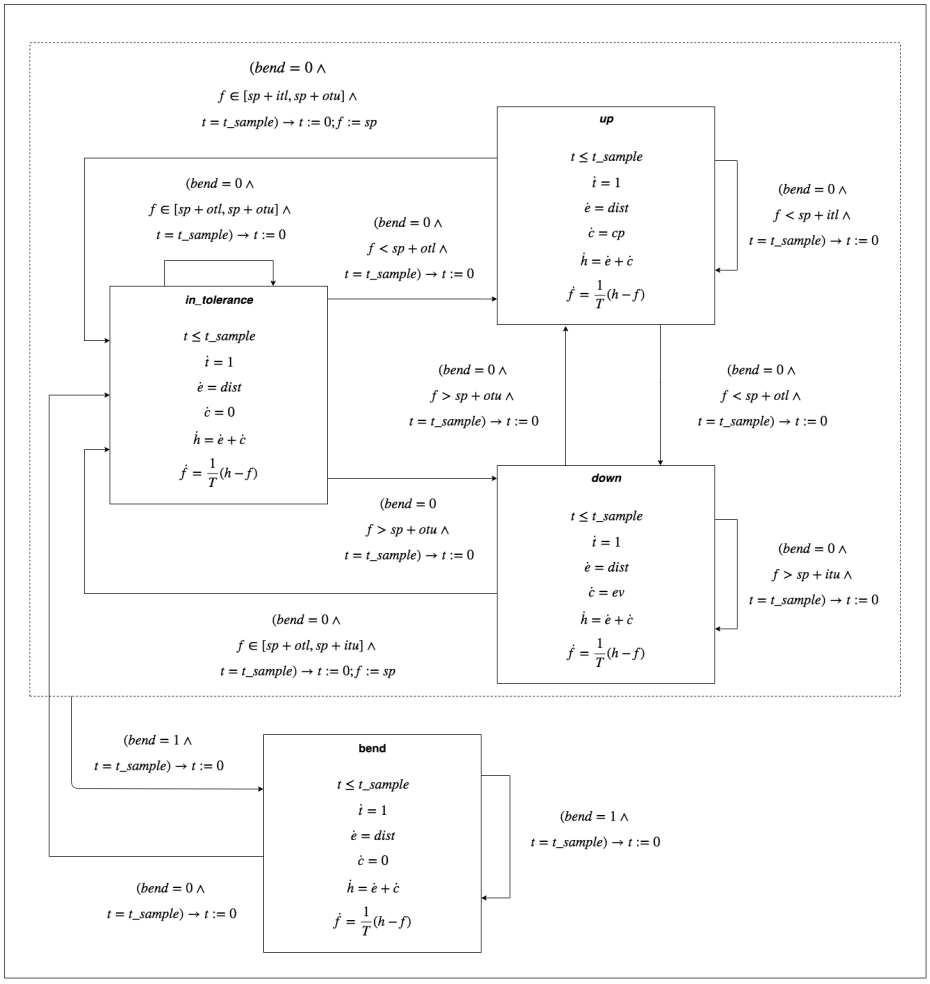


Fig. 5: Hybrid automaton of the suspension system.

The control flow is described as follows. The system starts in the **in_tolerance** state, which represents the state where the chassis is within the tolerance interval and all valves, as well as the compressor, are closed. Whenever the controller needs to increase the chassis level, the system switches to the **up** state. Analogously, the system switches to the **down** state when the chassis level needs to be decreased.

We notice how, in Figure 5, the origin of the transition to the **bend** state is a superstate composed by the **in_tolerance**, **up** and **down** states. We do not have this notion of superstates in our hybrid automata representation, but we have chosen to depict it this way to help with visualisation. This means that, at any point in time, the system might enter the **bend** state, in which case the system should maintain the compression levels.

There is a common pattern in most transitions: first, the system checks if it needs to enter or exit the **bend** state. Then, there is a chassis level check to verify whether it warrants a state change. Finally, the timer needs to be equal to the system period for the transition to occur. Two exceptions to this pattern occur in transitions coming from **up** and **down**, and going to **in_tolerance**. In these cases, there is an additional operation, i.e., the filtered chassis level is reset to the set point.

Each location also has an invariant followed by 5 differential equations. The invariant checks if the timer is still less than the period, while the equations represent the required changes in the system variables as well as the output c. In the **up** and **down** states, the chassis level changes are dictated by the compressor and escape valve inputs, respectively; it remains constant in the **in_tolerance** and **bend** states. Furthermore, we note that the system uses the filtered chassis level f instead of the current chassis level c on its guards. The computation of f uses a combination of the current chassis level, environmental disturbances and a filter constant T.

C Extra details of the experimental plan

C.1 Mutant Operators

The mutation operators used in this experiment were chosen based on a study on mutation operators for Simulink models [12] and are shown in Table 3, along with the number of inserted faults for the boost converter (BC) and suspension system (SS) models.

Operator	**BC**	**SS**
Constant Change	12	14
Variable Change	13	18
Variable Negation	6	9
Constant Replacement	8	12
Statement Change	9	13
Delay Change	8	13
Relational Operator Replacement	10	10
Arithmetic Operator Replacement	10	10
Total for each model	82	105

Table 3: Mutation operators and number of faults.

In the experiment, a higher priority was given to variable change and constant change due to the complexity in detecting this type of faults.

C.2 Variable selection

Here, we define the dependent and independent variables used in this experiment. Variables of both types exert some influence in the experiment, however,

dependent variables are the ones that the experiment focuses on whilst independent variables are those that must be fixed beforehand in order to control the experiment. The experiment was carried out using a computer with Intel Core i5 processor, 8GB of RAM and Windows 10 as operating system and the Matlab 2018b framework.

The dependent variables will be measured during the experiment, as a means to evaluate the hypotheses. These are:

- Time recorded while inputs are being created.
- Number of inputs created.
- Number of mutants killed.

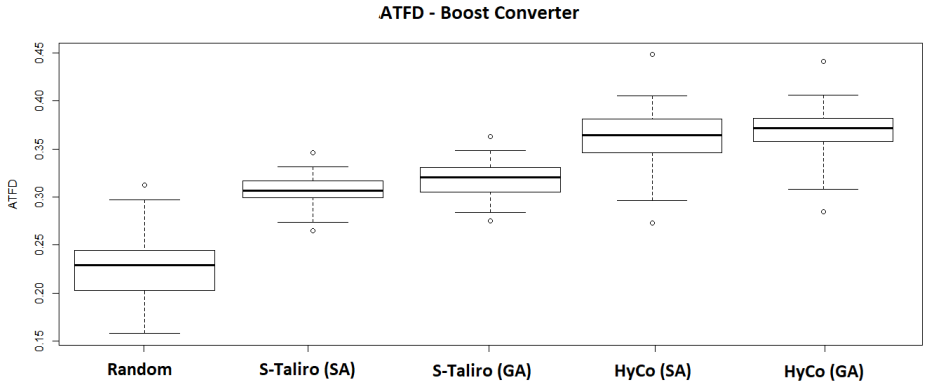
As for independent variables, their control must be done in such a way as to avoid bias. The pre-fixed variables are:

- The models to which the strategies must create inputs.
- The introduced defects (mutants).
- Versions of all the utilised software.
- Configuration of the computer in which the experiment will be performed.
- Human experience with software testing and the technologies used.

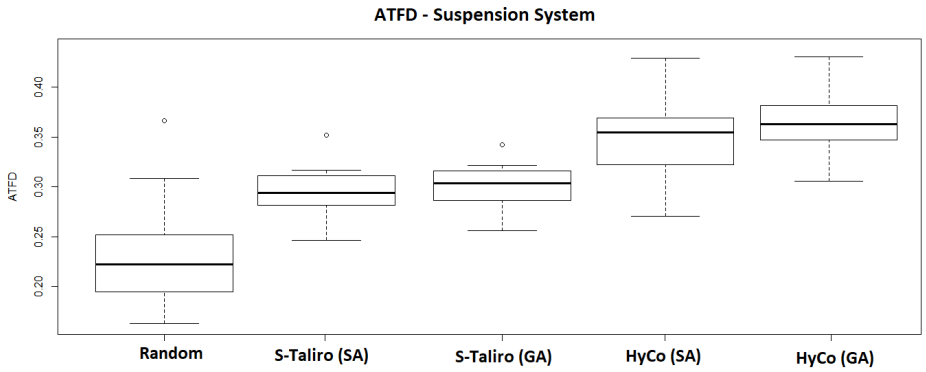
D ATFD - Boxplot results

The results below are depicted in boxplot format. In each figure, we compare the strategies used (x-axis) against the ATFD obtained (y-axis). Each boxplot shows the median (line across the boxes), top and bottom quartile, upper and lower fence and also outliers (small circles). Figure 6a indicates that the median ATFD values for random, S-Taliro (SA), S-Taliro (GA), HyConf (SA) and HyConf (GA) strategies obtained from the boost converter example are 0.228, 0.311, 0.327, 0.364 and 0.373 respectively. Analogously, in the case of the suspension system (see Figure 6b), the values are 0.222, 0.293, 0.309, 0.362 and 0.371 respectively.

In each figure, we compare the strategies used (x-axis) against the ATFD obtained (y-axis). Each boxplot shows the median (line across the boxes), top and bottom quartile, upper and lower fence and also outliers (small circles). Figure 6a indicates that the median ATFD values for random, S-Taliro (SA), S-Taliro (GA), HyConf (SA) and HyConf (GA) strategies obtained from the boost converter example are 0.228, 0.311, 0.327, 0.364 and 0.373 respectively. Analogously, in the case of the suspension system (see Figure 6b), the values are 0.222, 0.293, 0.309, 0.362 and 0.371 respectively.



(a) ATFD values for the boost converter example.



(b) ATFD values for the suspension system example.

Fig. 6: ATFD values