VeriLHyS: a Framework for LTL Specification and Verification of Hybrid Systems

Ludovico Battista[0000−0001−8197−2001], Stefano Tonetta[0000−0001−9091−7899], and Gianni Zampedri[0009−0005−5425−8490]

Fondazione Bruno Kessler, Trento 38123, Italy

Abstract. The automated verification of Linear Temporal Logic (LTL) properties over hybrid systems is an important challenge in formal methods. While numerous tools exist for checking safety and reachability, no framework currently provides a concrete language and algorithm for the full verification of LTL on hybrid models that combine discrete and continuous dynamics described in terms of differential equations. We present VeriLHyS, the first tool that enables the specification and automated verification of LTL properties for hybrid automata. The framework integrates continuous analysis techniques with symbolic model checking, leveraging Lyapunov-like, descent, and barrier certificates, as well as reachability set overapproximations, to derive sound LTL constraints on the system abstraction. These constraints are then checked using symbolic LTL model checking, ensuring correctness of the verified property on the original hybrid system. VeriLHyS supports both linear and nonlinear dynamics and is built on top of existing engines such as CORA, nuXmv, and SMT solvers. Our experiments show that it can handle complex hybrid benchmarks that go beyond the reach of existing tools.

1 Introduction

The verification of liveness properties in hybrid systems is a highly challenging problem. While safety and reachability properties have received significant attention, with mature tools and techniques available (see, e.g., [FGD+11,CÁS13,Alt15] [BD17,SÁMK17]), the automated verification of liveness properties - such as recurrence, responsiveness, and region stability - has been tackled only for specific types of discrete or continuous evolutions [AHLP00,CGMT14] or properties [CN12,PKV13,WTL14,HS20], and remains a hard challenge in general. These properties, often expressed in Linear Temporal Logic (LTL) [Pnu77], are crucial for reasoning about the long-term behavior of systems.

In this paper, we present VeriLHyS, a fully automated tool for VERIfying LTL properties of HYbrid Systems. The tool takes advantage of continuous analysis techniques and discrete-time model checking. It constructs a coarse discrete abstraction of the hybrid system and makes use of synthesized certificate functions (e.g., Lyapunov-like, descent, and barrier functions) and reachability set overapproximation (RSO) to derive a set of LTL constraints that are provably satisfied by the discrete system; it then uses symbolic model checking to verify

the desired property on the abstraction. Theoretical arguments show that this ensures that the property holds on the original hybrid system.

After a recap of the formal problem and the algorithmic solution presented in [BT25], the paper provides a detailed description of the architecture of VeriLHyS. The architecture specifies the software components of the framework together with their inputs and outputs. We then move to the input language of the tool, which extends the standard SpaceEx format [FGD+11] to allow the specification of LTL properties for hybrid systems. Then, specific parts of the verification algorithm that make VeriLHyS fully automated are detailed: (i) the synthesis of certificate functions, including Lyapunov-like, descent, and barrier certificates, obtained through LMI- and SMT-based techniques; (ii) the generation of semialgebraic regions by means of a heuristic combination of the regions appearing in the input (initial conditions and property) with sublevel sets derived from the synthesized Lyapunov-like functions; and (iii) the construction of the finite discrete abstraction used for symbolic model checking with an AllSMT procedure [SSB25] strengthened by valid constraints on the Boolean variables representing the regions statically learned by checking intersection or containment of the corresponding semialgebraic sets.

VeriLHyS is implemented by integrating several existing tools: CORA [Alt15] for reachability analysis, nuXmv [CCD+14] for symbolic model checking, numerical solvers (YALMIP [Löf04], Mosek [Mos]) for synthesis of candidate certificate functions, and SMT solvers for certificate validation and abstraction synthesis. The tool supports both linear and nonlinear (polynomial) dynamics and scales to systems with large discrete structures and complex behaviors.

We demonstrate the effectiveness of VeriLHyS on a series of benchmarks, including systems with switching logic and nonlinear flows. The tool can automatically generate and handle hundreds of regions and constraints, proving properties that are currently out of reach for other verification frameworks. The tool and benchmarks are available at https://es-static.fbk.eu/tools/verilhys.

The rest of the paper is organized as follows. Sec. 2 discusses related work and tools. Sec. 3 establishes the formal Background for our work, defining Hybrid Automata and the LTL properties, and summarizing the verification approach that VeriLHyS implements. Sec. 4 presents the Architectural Design, detailing the structure of the tool and the integration of external engines. Sec. 5 describes the Input Language used by VeriLHyS, focusing on the syntax for specifying the hybrid automaton and the LTL properties. Sec. 6 focuses on the implementation and details three key aspects: the synthesis of the certificates, the synthesis of the regions and the computation of the abstraction. Sec. 7 presents the Experimental Evaluation using a variety of complex benchmarks. Finally, Sec. 8 summarizes our contributions and discusses future work.

2 Related works

A wide range of abstraction and verification techniques have been developed for hybrid systems (see, e.g., [AGRS25]), typically under specific assumptions

on the dynamics. For linear or rectangular hybrid automata, predicate abstraction and CEGAR approaches have proved effective for verifying safety properties [ADI06], while liveness-related properties have been tackled only under restrictive conditions (see, e.g., [PW06,CN12,PKV13,WTL14,HS20,BT24]). Complete discrete abstractions are known to exist only for particular classes of continuous or discrete evolutions [AHLP00], and SMT-based model checking has been successfully applied to rectangular hybrid automata [CGMT14]. In contrast, VeriLHyS builds a simple but general abstraction that can handle arbitrary polynomial dynamics and supports the verification of full LTL properties.

Several tools address hybrid system verification, such as SpaceEx [FGD+11], Flow* [CÁS13], CORA [Alt15], HyLaa [BD17], and HyPro [SÁMK17], focusing primarily on reachability and verification of safety properties. These methods are highly effective for bounded-time verification but do not provide algorithms to prove temporal or liveness properties beyond safety. KeYmaeraX [FMQ+15] is a deductive framework that provides an interactive environment for proving temporal properties using differential dynamic logic, but the automated proof strategy for LTL verification is very limited. Some works have approached the verification of liveness properties, using Lyapunov-like functions as certificates for the validity of temporal properties [HS20], and, in the context of non-linear systems, to invalidate substrings to prove general LTL properties implied by safety properties [WTL14]. However, no implemented tools exist for these methods.

To the best of our knowledge, VeriLHyS is the first tool that provides a concrete specification language and a fully automated algorithm for verifying generic LTL formulas on hybrid systems governed by ordinary differential equations. It integrates continuous analysis (via certificate and reachability computations) with symbolic model checking into a unified and sound workflow. Moreover, the benchmark specification format adopted by VeriLHyS —an extension of the SpaceEx language to include LTL properties—ensures compatibility with existing hybrid system models and encourages the development of future tools supporting temporal logic verification, e.g., within the ARCH competition [AAB+23].

3 Background

In this section we briefly recollect the main definitions and concepts used in the paper. We refer to [BT25] for more details.

3.1 Hybrid Automata and Hybrid Traces

A *Hybrid Automaton* is given as a tuple
$H = (X, Q, q_0, E, \mathit{Init}, \mathit{Flow}, \mathit{Inv}, \mathit{Guard}, \mathit{Jump})$, where:
- X is a finite set of real-valued variables $x_1, \ldots, x_n$;
- Q is a finite collection of discrete states (also called locations);
- $q_0 \in Q$ denotes the initial location;

- $E \subseteq Q \times Q$ is the set of discrete transitions;
- $Init \subseteq \mathbb{R}^n$ specifies the set of initial continuous states;
- $Inv : Q \to 2^{\mathbb{R}^n}$ assigns to each location q an invariant $Inv(q)$, describing the set of continuous states in which the system may remain while in q;
- $Flow : Q \times \mathbb{R}^n \to \mathbb{R}^n$ is a (possibly partial) function that determines the continuous evolution in each location $q \in Q$ via a system of differential equations;
- $Guard : E \to 2^{\mathbb{R}^n}$ assigns to each edge a guard condition, specifying when the transition (q, q') is enabled depending on the current continuous state;
- $Jump : E \to 2^{\mathbb{R}^n \times \mathbb{R}^n}$ assigns to each transition a relation describing the possible resets (jumps) of the continuous variables.

A *hybrid trace* for a hybrid automaton H is an infinite sequence

$$\sigma = \langle f_0, I_0, q_0 \rangle, \langle f_1, I_1, q_1 \rangle, \dots$$

where for each $i \geq 0$:

- $q_i \in Q$ is a location,
- $I_i \subseteq \mathbb{R}_{\geq 0}$ is a (possibly unbounded) interval, with $l(I_{i+1}) = u(I_i)$,
- The intervals I_i cover $\mathbb{R}_{\geq 0}$.
- $f_i : I_i \to \mathbb{R}^n$ is analytic.

A hybrid trace is a *trajectory* (also called a *run* or a *solution*) of H if:

- the image of f_i is contained in $inv(q_i)$.
- if $l(I_i) \neq u(I_i)$, then $\frac{df_i}{dt}(t) = flow(q_i)(f_i(t))$ for all $t \in I_i$;
- if $q_i = q_{i+1}$, then $f_i \cup f_{i+1}$ is well defined and analytic on $I_i \cup I_{i+1}$. In particular, either I_i is right-closed or I_{i+1} is left-closed;
- if $q_i \neq q_{i+1}$, then I_i is right-closed, I_{i+1} is left-closed, $e = (q_i, q_{i+1}) \in E$, $f_i(u(I_i)) \in guard(e)$, and $(f_i(u(I_i)), f_{i+1}(l(I_{i+1}))) \models jump(e)$.

3.2 LTL

We use the standard syntax of LTL and the semantics over hybrid traces as defined in [CRT09]. We adopt the variant HLTL defined in [BT25] where the propositions of LTL can be either a discrete location $q \in Q$ or a region predicate $R \subseteq \mathbb{R}^n$ over the continuous variables. For simplicity, we use LTL throughout this paper to refer to this hybrid variant.

Let $Expr(X)$ denote the set of Boolean combinations of polynomial constraints over X (called *regions*). The syntax of LTL formulas is given by:

$$\varphi ::= q \mid R \mid \varphi \vee \varphi \mid \neg\varphi \mid \varphi U \varphi$$

where $q \in Q$ and $R \in Expr(X)$. We use standard abbreviations: $\varphi \wedge \psi := \neg(\neg\varphi \vee \neg\psi)$, $\varphi \to \psi := \neg\varphi \vee \psi$, $F\varphi := \top U \varphi$, $G\varphi := \neg F \neg\varphi$. In the rest of the paper, we often refer to *local regions* $Z = (R_Z, Q_Z)$: they can be thought as formulas of the form $R_Z \wedge (\bigvee_{q \in Q_Z} q)$: they represent regions that can be reached only in certain locations.

Given a hybrid trace $\sigma = \langle f_0, I_0, q_0 \rangle, \langle f_1, I_1, q_1 \rangle, \dots$ and an LTL formula φ, the satisfaction relation $\sigma, i \models \varphi$ (at position i) is defined inductively as:

- $\sigma, i \models q$ iff $q_i = q$,
- $\sigma, i \models R$ iff for all $t \in I_i$, $f_i(t) \models R$,
- $\sigma, i \models \varphi_1 \vee \varphi_2$ iff $\sigma, i \models \varphi_1$ or $\sigma, i \models \varphi_2$,
- $\sigma, i \models \neg\varphi$ iff $\sigma, i \not\models \varphi$,
- $\sigma, i \models \varphi_1 U \varphi_2$ iff there exists $k \geq i$ such that $\sigma, k \models \varphi_2$ and for all j with $i \leq j < k$, $\sigma, j \models \varphi_1$.

We write $\sigma \models \varphi$ if $\sigma, 0 \models \varphi$.

Given a hybrid automaton H and an LTL formula φ over Q and X, we write $H \models \varphi$ if for all hybrid traces σ that satisfy initial conditions (*i.e.*, the starting location is q_0 and $f_0(0)$ is in *init*) that are *ground* for φ (*i.e.*, the truth of each region predicate is constant over each interval I_i), it holds that $\sigma \models \varphi$.

3.3 Certificate functions and RSO

We use mainly two methods to get information on the behaviour of the system: (i) *certificate functions* and (ii) *Reachability Set Over-approximation (RSO)*. We briefly recall these two concepts and refer to [BT25, Section 2.4] for more details.

Certificate functions Certificate functions are real-valued functions defined over a subset of $X \times Q$. They are used to prove properties about the system's behavior, such as safety, stability, and progress. In VeriLHyS, we use three types of certificate functions: Lyapunov-like functions, Descent functions, and Barrier functions.

- *Lyapunov-like functions* help establish stability, even when the system does not converge to a single equilibrium. They consist in non-negative functions that tend to zero along trajectories within designated locations.
- *Descent functions* are similar in spirit but strictly decrease (with temporal derivative $< -\varepsilon$) along trajectories within designated locations. They are useful for proving progress properties.
- *Barrier functions* are used to certify safety. They ensure that trajectories starting in a safe region (where the barrier function is non-positive) cannot cross into unsafe regions (where the function becomes positive), provided the system remains within a specified subset of locations.

Each certificate function F is defined as a pair (F_f, F_Q), where $F_Q \subseteq Q$ is the set of locations where the function's conditions hold, and F_f is a scalar function from $X \times F_Q$ to $\mathbb{R}$. We use these functions to derive LTL constraints that are guaranteed to hold on the original hybrid system.

Reachability Set Over-approximation (RSO) Reachable Set Overapproximations (*RSO*) are a powerful tool for analyzing the behavior of hybrid systems by estimating where the system can evolve over time. Given a hybrid system H, an initial region R, and a starting location q, an RSO with granularity τ up to time T is a finite collection of triples (Y_i, J_i, p_i), where: Y_i is a region in the state space, $J_i = [l_i\tau, u_i\tau]$ is a time interval, and $p_i \in Q$ is a system location, such that for every run $\sigma = \langle f_j, I_j, q_j \rangle$ with $f_0(0) \in R$ and $q_0 = q$, for every $0 \leq t \leq T$ and every j such that $t \in \operatorname{Dom} f_j$, the point $f_j(t)$ is inside $\bigcup_{i \mid t \in J_i, p_i = q_j} Y_i$.

Object	**Property**	**Type**
Lyapunov-like Function	$G(F(G(\neg Z) \vee (\neg q)))$	Liveness
Descent Function	$G(F(G(\neg Z) \vee (\neg q)))$	Liveness
Barrier Function	$G((\bigvee Z_1) \rightarrow (G(\neg((\bigvee Z_2))W(\neg q)))$	Safety
Reachable Set Overapproximation	$R_1 \wedge (Z_1 U(Z_2 \wedge Z_2 U(Z_3 \wedge Z_3 U \ldots)))$	Liveness

Table 1. Summary of the form of LTL properties generated from each analysis technique. Z_i represents regions and q represents a location.

Informally, the RSO captures all possible positions the system may occupy at any time t within $[0, T]$, given its initial condition, by covering them with a union of regions indexed by time and location.

3.4 Algorithm

The algorithm implemented in VeriLHyS is described in [BT25], we briefly summarize it here. It takes as input: a hybrid system H as defined in Definition 3.1, an LTL formula φ, and (optionally) a set of user-defined local regions $\mathcal{Z}_{add}$, and performs the following steps:

1. it extracts the initial set of local regions $\mathcal{Z}_{in}$ from φ and the initial condition (see Section A.1);
2. it synthesizes and validates Lyapunov-like $\mathcal{L}$ and descent functions $\mathcal{D}$;
3. it synthesizes additional local regions $\mathcal{Z}_{Lyap}$ to be used in the abstraction. Let $\mathcal{Z} = \mathcal{Z}_{in} \cup \mathcal{Z}_{Lyap} \cup \mathcal{Z}_{add}$;
4. it computes the barrier functions $\mathcal{B}$. Let $\mathcal{F}$ be $\mathcal{L} \cup \mathcal{D} \cup \mathcal{B}$;
5. it computes an LTL formula γ_f over $\mathcal{Z}$ and Q for each $f \in \mathcal{F}$;
6. it performs RSO for each local region $(R_Z, Q_Z) \in \mathcal{Z}$ and each $q \in Q_Z$ and computes the LTL formula $\gamma_{Z,q}$ over $\mathcal{Z}$ and Q (the granularity τ and time T are fixed parameters of the algorithm).
7. it computes the discrete abstraction D_H, which has a variable v_Z for each local region $(R_Z, Q_Z) \in \mathcal{Z}$ and a variable v_q for each location $q \in Q$;
8. it transforms φ and each generated γ_i into LTL properties $\widehat{\varphi}$ and $\widehat{\gamma_i}$ substituting Z with v_Z and q with v_q;
9. if $D_H \models (\bigwedge_{f \in \mathcal{F}} \widehat{\gamma_f} \wedge \bigwedge_{Z \in \mathcal{Z}, q \in Q_Z} \widehat{\gamma_{Z,q}}) \rightarrow \widehat{\varphi}$, then it returns $True$;
10. else it returns $Unknown$.

The correctness of the algorithm is given by [BT25, Theorems 1 - ... - 5].

LTL constraints In Table 1 we summarize the form of the LTL constraints that are obtained the steps 5 and 6 above. Lyapunov-like and Descent functions capture progress properties: they ensure that regions that assume values bounded from zero (or from $-\infty$ for descent functions) are eventually left (if the systems stays in the valid locations of the function). Barrier functions capture safety properties: they ensure that if the system cannot pass from a region that assumes only non-negative values to a region that assumes only positive values (if the

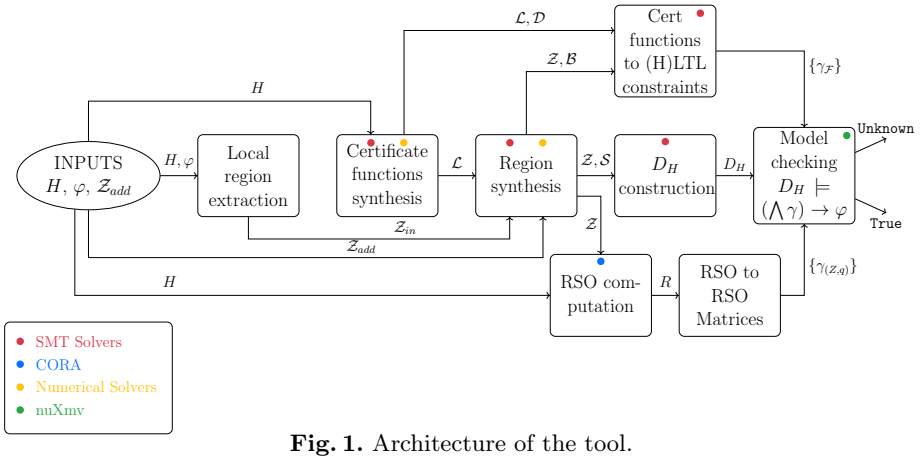

Fig. 1. Architecture of the tool.

systems stays in the valid locations of the function). Finally, RSO constraints the evolution of the system: if the RSO tells us that we go from R_1 to R_2 without passing to R_3, this is implied by the LTL constraint given in the Table.

4 Design

4.1 High-level architecture

VeriLHyS implements the algorithm described in Section 3.4 and is structured according to the main phases of the algorithm. Figure 1 shows the high-level functional architecture of the tool, detailing the software components in term of input and output. These correspond to the main phases of the algorithm:

1. **Local region extraction.** Reads the hybrid automaton H and the property φ and extracts local regions of interest $\mathcal{Z}_{\text{in}}$.
2. **Certificate functions synthesis.** Synthesizes and validates Lyapunov-like and descent functions using (a) SOS/LMI-based numerical synthesis followed by SMT validation, or (b) direct SMT-based synthesis.
3. **Regions synthesis.** From Lyapunov-like functions it generates sublevel-set regions and (optionally) their images through admissible jumps. It employs numerical optimization followed by SMT validation to compute maxima and minima of functions; the image is computed by finding the inverse of the reset map when possible and considering resets. For each generated sublevel set it also produces the associated barrier function.
4. D_H **construction.** Constructs the finite transition system (FTS) D_H with Boolean variables for locations and (local) regions, encoding geometric properties of the system (as adjacency of regions, possible transitions allowed by jumps, and non-empty combinations) in invariant and transition constraints. Such constraints are computed using AllSMT queries to SMT solvers.

5. **Cert functions to LTL constraints.** Converts certificate functions into LTL constraints $\gamma_{\mathcal{F}}$.
6. **RSO computation.** Computes Reachable Set Overapproximations (RSO) for pairs (Z, q) up to horizon T with granularity τ using CORA.
7. **RSO to RSO Matrices.** Converts RSOs into matrices that captures the reachability information on the abstracted system (see [BT25] for details). These are readily converted into LTL constraints $\gamma_{Z,q}$.
8. **Model Checking $D_H \models (\bigwedge \gamma) \rightarrow \varphi$.** Checks whether $D_H \models \left(\bigwedge \gamma_f \wedge \gamma_{Z,q}\right) \rightarrow \varphi$ via symbolic LTL model checking using nuXmv.

Implementation bindings As shown in Figure 1, we use the following tools: SOSTools for numerical synthesis of certificate functions; CORA (MATLAB) for RSO; z3 and cvc5 for AllSMT checks and certificate synthesis and validation; pyVMT and pySMT for model construction and manipulation of formulas; nuXmv for LTL model checking.

5 Input details

VeriLHyS takes in input a hybrid automaton and an LTL property. These are specified in an extension of the modeling language employed by SpaceEx [FGD+11].

Formally, the input format is defined by a grammar expressed in Compact RelaxNG format, which builds upon the original schema of the SpaceEx language. The grammar introduces the following additions:

InitialLocation An optional element that specifies the initial location of the system. The value must correspond to a location defined within the model's location list. For example,

```
<initial_location>1</initial_location>
```

InitialRegion An optional element used to define the initial region of the system's state space. For example,

```
<initial_region>x &gt; 0 &amp; x &lt; 51/5</initial_region>
```

Invar, Reach and LTLProp These elements allow the specification of invariants (Gp), reachability (Fp), and general LTL formulas, respectively, which are used during system analysis. For example,

```
<invariant>x &gt;= 0</invariant>
<reach>(x == 1)</reach>
<ltl-property> (F (G ( x &lt; 1)) ) &amp;&amp; y </ltl-property>
```

`Invar`, `Reach` and `LTLProp` elements must contain syntactically valid LTL expressions, which conform to the following grammar:

```
<ltl_expr> :: basic_expr
            | X <ltl_expr>
            | G <ltl_expr>
            | F <ltl_expr>
            | <ltl_expr> U <ltl_expr>
            | <ltl_expr> R <ltl_expr>
```

`ltl_expr` expressions can be composed using standard logical operators and can also incorporate `basic_expr` expressions, which support classical arithmetic operations and functions over reals.

In contrast to the assumptions made by the native SpaceEx language, VeriLHyS enforces syntactic and semantic validation of expressions embedded within the standard tags `flow`, `guard` and `reset-relation`.

Additionally, VeriLHyS supports the specification of auxiliary regions via a dedicated parameter. This parameter must reference an external file that conforms to the following syntax:

```
<?xml version="1.0" encoding="UTF-8"?>
    <additional-regions version="0.1">
        <region>
            <polynomial-constraints>
            (x1_sp &gt;= 19/20) &amp; (x2_sp &gt;= 1/20)
            </polynomial-constraints>
            <valid-locations>
                <loc>all</loc>
            </valid-locations>
        </region>
        <region>
        ...
        </region>
    </additional-regions>
```

6 Implementation details

Key aspects of VeriLHyS proofs are the synthesis of the certificate functions, the synthesis of regions, and the construction of the discrete abstraction. In this section, we detail some heuristics used to make these phases more effective.

6.1 Certificate synthesis

VeriLHyS implements the algorithm described in [BT25] to synthesize the certificate functions. Two implementation synthesis strategies are used to find Lyapunov-like and Descent functions: (i) numerical synthesis using Sum-of-Squares (SOS), followed by SMT-based validation, and (ii) direct SMT-based synthesis. The resulting functions are validated to ensure they satisfy the required conditions on flows and jumps. These certificates will later be used to derive LTL constraints.

Based on several tests, we noticed that strategy (i) was more effective for Lyapunov-like functions, while strategy (ii) was more effective for Descent functions. In particular, the non-existence of a quadratic Lyapuov-like function took a long time to be proved by SMT solvers, while numerical solvers were not effective in finding Descent functions. For this reason, we implemented strategy (i) for Lyapunov-like functions and strategy (ii) for Descent functions.

6.2 Region synthesis

The local regions used for abstraction come from three sources: (i) the initial set of regions $\mathcal{Z}_{in}$ extracted from the property φ and the initial condition; (ii) user-defined regions $\mathcal{Z}_{add}$; (iii) regions $\mathcal{Z}_{Lyap}$ generated as sublevel sets of the synthesized Lyapunov-like functions. The union of these three sets forms the complete set of local regions $\mathcal{Z}$ used for abstraction.

The regions in $\mathcal{Z}_{in}$ consist in the local region $(init, \{q_i\})$ and $(\psi, \{q_i\}_{i\in I})$ given by the subtrees of the DAG representation of φ of the form $(\psi(X) \wedge \bigvee_{i\in I} q_i)$, where ψ is a semialgebraic formula over the continuous variables X and q_i are locations; more details on this can be found in Appendix A.

The regions in $\mathcal{Z}_{add}$ are given to the tool as input by the user. As already mentioned, we have defined a simple grammar for specifying them.

The regions in $\mathcal{Z}_{Lyap}$ are synthesised using Lyapunov-like functions. From each validated Lyapunov-like function (L_f, L_Q), the tool generates a set of sublevel-set regions of the form $(L_f \leq a, L_Q)$, where a ranges over a set of critical values computed from the regions in $\mathcal{Z}_{in} \cup \mathcal{Z}_{add}$.

Critical values computation The constants that define the sublevel sets of Lyapunov-like functions are critical to ensure that the sublevel sets are meaningful and interact appropriately with the information abstracted in the discrete system. We compute these constants a for each Lyapunov-like function based on the minimum and maximum values that the Lyapunov functions attain over existing regions.

In particular, for each Lyapunov function (L_f, L_Q), we:

1. identify the local regions $(R_Z, Q_Z) \in \mathcal{Z}_{in} \cup \mathcal{Z}_{add}$ such that $C \cap Q_Z \neq \emptyset$;
2. for each such local region (R_{Z_i}, Q_{Z_i}), compute the minimum and maximum values of L_f over the polynomial constraints defining R_i:

$$k^{(i)}_{\min} = \min_{x\in R_{Z_i}} L_f(x), \quad k^{(i)}_{\max} = \max_{x\in R_{Z_i}} L_f(x).$$

   This is done using numerical optimization techniques, followed by SMT-based validation to ensure correctness. When the validation fails, we proceed by increasing (resp. decreasing) slightly the candidate maximum (resp. minimum) value and re-validate; if this fails after a fixed number of attempts, we discard the value;
3. collect all critical values $\{a_i\}$: the value zero and all minima and maxima from step 2;
4. sort the critical values in ascending order, compute intermediate values between each consecutive pair a_i, a_{i+1}, and add them to the set $\{a_i\}$; we sort again the set of constants $\{a_i\}$ including these intermediate values; we also remove duplicates.
5. for each constant a_i, define a new local region as: $(\{x \in \mathbb{R}^n \mid L_f(x) \leq a_i\}, L_Q)$.

This procedure aims to ensure that the sublevel sets are constructed using meaningful thresholds derived from the behavior of the Lyapunov function over known regions. By including intermediate values, we increase the granularity of the abstraction, aiming to capture more behaviours. Finally, for every generated sublevel set, the associated barrier function $(L_f - a, L_Q)$ is added to the set of certificates.

Images through jumps To keep some information on the behaviour of the system after jumps, we generate additional local regions by considering the images through admissible jumps of the synthesized sublevel sets.

To compute these images, for each $(q_1, q_2) \in E$ such that $q_1 \in L_Q$, we compute the intersection of the sublevel set with the guard $L_f \leq a \cap guard((q_1, q_2))$, and we compute its image through the reset map $jump((q_1, q_2))$. In general, the computation of the image of a semialgebraic set through a polynomial map is can be tackled using quantifier elimination techniques. However, these techniques are often impractical. In our tool, we compute these images only in the case the reset map is invertible, is a projection, or is a combination of these two. In these cases the explicit computation of the image is straightforward.

6.3 SMT-based computation of the discrete abstraction

The computation of the abstract FTS D_H, as described in [BT25, Definition 14], is achieved with a series of quantifier elimination queries computed with an AllSMT procedure (see, e.g., [SSB25]). The FTS is built using pyVMT [pyV22] and pySMT [GM15] libraries. The AllSMT is performed using z3 [dMB08] and cvc5 [BBB+22] solvers. An important optimization that we implemented is inspired by standard static learning techniques in SMT (see, e.g., [ABC+02]) and precomputes a set of valid constraints to simplify the AllSMT queries.

Static learning While synthesizing the regions given by sublevels of Lyapunov-like functions, we save a set of propositions that come naturally from the definitions of the synthesized regions. Given a Lyapunov-like function (L_f, L_Q), let $Z_{a_1}, \ldots, Z_{a_k}$ be the local regions generated as sublevel sets of L_f; suppose that $a_1 < a_2 < \ldots < a_k$. We consider two types of propositions that can be learned statically:

- *Type-1 static learnt propositions.* Clearly, the sublevel sets are nested: therefore, for every $i < j$, the following implication holds: $v_{Z_{a_i}} \rightarrow v_{Z_{a_j}}$.
- *Type-2 static learnt propositions.* During the generation of sublevel sets, we have associated to each Lyapunov-like function (L_f, L_Q) and each local region Z in $\mathcal{Z}_{in} \cup \mathcal{Z}_{add}$ the maximum M_{Z,L_f} and the minimum m_{Z,L_f} of L_f on Z. Whenever $M_{Z,L_f} < a_i$, the following implication holds: $v_Z \rightarrow v_{Z_{a_i}}$, and whenever $m_{Z,L_f} > a_i$, the following implication holds: $v_{Z_{a_i}} \rightarrow (\neg v_Z)$.

We call $\mathcal{S}$ the conjunction of all such implications for all Lyapunov-like functions. The actual AllSMT queries are then performed on the formula $\lambda \wedge \mathcal{S}$, which is equisatisfiable to λ but is simpler to solve, as it contains additional information that the solver can exploit.

Benchmark	**dim**	$\|Q\|$	**flow**	$\|\mathcal{Z}_{in}\|$	$\|\mathcal{Z}_{add}\|$	LTL properties tested
Linear Overtake	4	3	Lin	2	0	$G(q_1 \rightarrow ((\neg(B))Uq_3))$, $G(F(q_1)), G(\neg B)$
Multiple choice example	2	2+2n	Lin	2	0	$F(G(Z_{target}))$
Navigation Benchmark	4	25	Lin	1	14	$G(F(Z_0))$
Non-linear circular	2	1	Nonlin	2	11	$G(F(R_{0,\alpha})), G(\neg B)$
Multilayer control lin	3	n	Lin	2	0	$G(q_i \rightarrow (\neg q_J W q_K)) \rightarrow (G(\neg B))$, $G(G(q_1) \rightarrow F(G(Z_0)))$
Multilayer control nonlin	3	n	Nonlin	2	0	$G(q_i \rightarrow (\neg q_J W q_K)) \rightarrow (G(\neg B))$, $G(G(q_1) \rightarrow F(G(Z_0)))$
Bball	2	1	Lin	2	0	$G(F(v \leq 1))$
Circle	2	2	Lin	2	0	$G(Z_0 \rightarrow F(Z_1))$
Drivetrain	10	4	Lin	1	0	$G(q_1 \wedge (F\neg(q_2)) \rightarrow F(\neg(q_3)))$
Rendezvous	4	1 to 3	Nonlin	1	0	$F(G(q_{final}))$

Table 2. Characteristics of the benchmarks. When $|\mathcal{Z}_{add}| = 0$, we did not need any additional handcrafted region to prove the properties. Some information on the LTL properties can be found in the description of the benchmarks, here they are presented to show the diversity of the properties verified.

7 Experimental evaluation

7.1 Benchmarks

We evaluate the effectiveness and robustness of our tool in proving LTL properties of hybrid systems across several benchmarks:

- Linear Overtake [BT25]: A 4-dimensional linear hybrid system modeling overtaking behavior of a vehicle. Here, q_1 represents the initial phase of the overtake and q_3 the final phase; B is the region occupied by the overtaken vehicle.
- Multiple Choice [BT25]: A scalable 2D benchmark with discrete jumps. The system passes through several location before reaching a final location with converging dynamics. Z_{target} is a small region around the target point.
- Navigation Benchmark [FI04]: A 4D system (space and velocity variables) with linear dynamics and 25 locations. Here we prove that the system always eventually returns to a specific region Z_0.
- Non-linear Circular [BT25]: A 2D system with non-linear dynamics converging to the unit circle. We prove that the system always returns to the starting region $R_{0,\alpha}$ (that changes based on α) avoiding an obstacle B.
- Multilayer Control (Linear and Nonlinear) [BT25]: A scalable 3D system with multiple locations converging (with linear or nonlinear dynamics) to different points in the space. We prove that if the switching takes place only among contiguous locations, then the system always avoids an obstacle B,

and that if the system stays in location q_1, it eventually converges to a region Z_0.
- Bball [Alt15]: 2D linear systems representing a bouncing ball. We prove that the velocity v of the ball is always eventually less than a threshold.
- Circle [Alt15]: A 2D linear system with two locations representing circular motion. We prove a reactive property.
- Drivetrain [Alt15]: A 10D linear system modeling a vehicle's drivetrain. Available in 2 different versions. We prove a property on the possible evolution of locations.
- Rendezvous [Alt15,AAB+23]: A 4D nonlinear system modeling a spacecraft rendezvous maneuver. Available in 6 different versions. We prove that the system eventually reaches the final location.

The properties of these benchmarks are presented in Table 2. We test four configurations of our algorithm: with/without images through jumps (as explained in Section 6.2) and with/without stabilizing constraints [CS12]. In particular, CF1 is without both, CF2 is without stabilizing constraints but with images, CF3 is without images but with stabilizing constraints, and CF4 is with both. For systems that admit several configurations, we report the entry *Param* that identifies the specific version of the model.

Metrics. We collect execution times (in seconds) for key components:
- T_{TOT}: total time.
- T_{Lyap}: Lyapunov-like function synthesis.
- T_{ASJump}, T_{ASReg}, T_{AdReg}: AllSMT-based abstractions.
- T_{RSO}, T_{RSOMat}: RSO and reachability matrix computation.
- T_{p_i}: time for solving the model checking problem for property i.

We also report:
- N_{LTL}: number of synthesized LTL constraints.
- $|\mathcal{Z}|$: number of regions.
- md_{RSO}, ms_{RSO}: average temporal depth and size of LTL properties coming from RSO.
- ms_{bar}: average size of LTL properties coming from barrier functions.

We do not show depths and sizes of other LTL constraints since they are fixed (see Table 1). The parameters T and τ are used by CORA in the computation of RSO. We set $T = 10$ and $\tau = 0.1$ for all benchmarks, and we changed τ to 0.01 when we were not able to prove the properties. We run benchmarks with 1h timeout. Navigation benchmark was particularly challenging, and we set a timeout of 10h for it.

Experiments were run on a 12-core 3.5 GHz machine with 32 GB RAM. Full logs and additional details are available in the Appendix, Section B.

7.2 Results

We here recollect some key observations from our experiments, and we refer the reader to the Appendix, Section B for the tables of all the results.

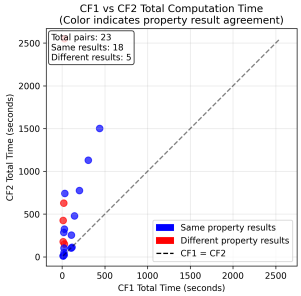

Fig. 2. Comparison of properties proved under CF1 and CF2 configurations across all benchmarks.

Effectiveness Our approach is effective in proving a variety of LTL properties on all the benchmarks considered, as shown in Table 2. All the shown properties were proved in at least one configuration (we refer again to the Appendix, Section B for more details).

In Figure 2 we compare the total time when using CF1 and CF2 (resp. without and with considering images through jumps in the abstraction), underlining the cases where we have obtained different results in terms of properties proved. In particular, it is expected that CF2 is more effective than CF1, at cost of a higher execution time. This is confirmed by the results, and paves the way to understanding when the use of images is more beneficial.

Scalability We evaluate the scalability of our approach on the Multiple Choice benchmark and the Multilayer Control benchmarks, which are both parametric in the number of locations. Results are shown in Figure 4 and Figure 5, respectively. They confirm a sustainable scalabitility of our approach, which is further confirmed by the results on the other benchmarks, with particular reference to the Navigation benchmark (see Table 3), where we manage 222 LTL constraints and 37 regions.

We also present in Figure 3 a comparison of total execution time against three metrics across all benchmarks: number of regions, LTL property size, and LTL property temporal depth. To improve readability, we excluded Navigation benchmark that was run with a different timeout. We notice that the conncection with the complexity at level of LTL contraints complexity appears stronger than the one at level of number of regions, suggesting a possible improvement of the tool via better ways to handle such constraints.

T_{TOT}	Config	T_{Lyap}	T_{ASJump}	T_{ASReg}	T_{AdReg}	T_{RSO}	T_{RSOMat}	$T\varphi_1$	Res φ_1	md_{RSO}	ms_{RSO}	ms_{bar}	N_{LTL}	$\lvert\mathcal{Z}\rvert$
24459.45	CF1	5.92	9.65	1.22	23.12	22.24	263.15	23915.48	True	35.33	2256.00	148.64	222	37

Table 3. Computation times and statistics for model *navigation* with $\tau = 0.01$ and $T = 10$.

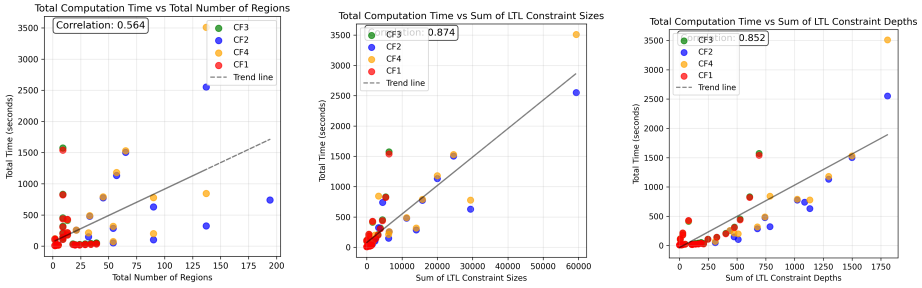

Fig. 3. Total execution time as a function of (a) number of regions, (b) LTL property size, and (c) LTL property temporal depth, across all benchmarks.

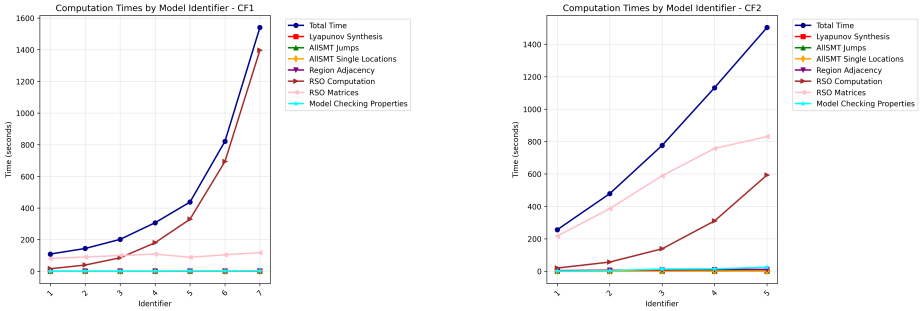

Fig. 4. Execution times for the Multiple Choice benchmark with increasing locations, under CF1 and CF2 configurations.

7.3 Limitations

We only reported the cases of successful verification to showcase the applicability and scalability of the tool and give some insights on the results. As the general problem of LTL verification for hybrid systems is undecidable, the VeriLHyS framework is not guaranteed to terminate successfully for all inputs and in many cases it may instead return 'Unknown' or time out. These unresolved cases generally occur when the continuous analysis phase fails to yield sufficiently tight sound overapproximations. For instance, the certificate synthesis may fail to find suitable Lyapunov-like functions to define significant regions for the abstraction, or the RSO computation may produce very coarse bounds due to high-dimensional state space or prolonged time horizons, leading to ambiguous LTL constraints. This could in principle take advantage of the automatic tuning of CORA, but we currently do not exploit this feature. All benchmark results, including the cases that currently return 'Unknown' or time out, are publicly available on the project website as open challenges.

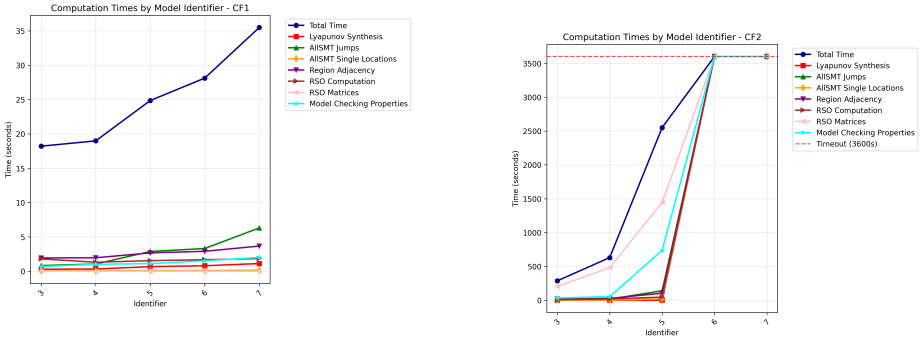

Fig. 5. Execution times for the Multilayer Control linear benchmark with increasing polytope vertices and sides.

8 Conclusions and Future Works

In this paper, we presented VeriLHyS, a novel framework that addresses the automated verification of Linear Temporal Logic (LTL) properties over hybrid systems with the continuous dynamics expressed in terms of differential equations. While existing tools primarily focus on restricted properties like safety and reachability, VeriLHyS is the first tool to provide a concrete language and algorithm for specifying and verifying full LTL specifications on this types of hybrid automata.

Our approach integrates techniques from both continuous and discrete verification domains. By leveraging Lyapunov-like, descent, and barrier certificates, alongside Reachability Set Overapproximations computed by engines like CORA, VeriLHyS is able to derive sound, guaranteed LTL constraints that hold on the original hybrid system. These constraints are then encoded as assumption on a finite system abstraction using the nuXmv symbolic model checker to prove more general LTL properties. The experimental results demonstrate that VeriLHyS is practical and scalable, capable of handling complex hybrid benchmarks with both linear and non-linear dynamics that are beyond the range of current verification tools.

Future work will focus on enhancing the precision and scope of the framework. Specifically, we plan to investigate methods to improve the quality of the synthesized regions, which should help minimize the number of 'Unknown' verification results. Furthermore, we intend to explore extensions of the framework to richer properties with metric operators.

References

AAB+23. Alessandro Abate, Matthias Althoff, Lei Bu, Gidon Ernst, Goran Frehse, Luca Geretti, Taylor T. Johnson, Claudio Menghi, Stefan Mitsch, Stefan Schupp, and Sadegh Soudjani. The ARCH-COMP Friendly Verification Competition for Continuous and Hybrid Systems. In *TOOLympics@ETAPS*, volume 14550 of *Lecture Notes in Computer Science*, pages 1–37. Springer, 2023.

ABC+02. Gilles Audemard, Piergiorgio Bertoli, Alessandro Cimatti, Artur Kornilowicz, and Roberto Sebastiani. Integrating Boolean and Mathematical Solving: Foundations, Basic Algorithms, and Requirements. In *AISC*, volume 2385 of *Lecture Notes in Computer Science*, pages 231–245. Springer, 2002.

ADI06. Rajeev Alur, Thao Dang, and Franjo Ivančić. Counterexample-guided predicate abstraction of hybrid systems. *Theoretical Computer Science*, 354(2):250–271, 2006. Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2003).

AGRS25. Alessandro Abate, Mirco Giacobbe, Diptarko Roy, and Yannik Schnitzer. *Model Checking and Strategy Synthesis with Abstractions and Certificates*, pages 360–391. Springer Nature Switzerland, Cham, 2025.

AHLP00. R. Alur, T.A. Henzinger, G. Lafferriere, and G.J. Pappas. Discrete abstractions of hybrid systems. *Proceedings of the IEEE*, 88(7):971–984, 2000.

Alt15. Matthias Althoff. An introduction to CORA 2015. In *Proc. of the 1st and 2nd Workshop on Applied Verification for Continuous and Hybrid Systems*, pages 120–151. EasyChair, December 2015.

BBB+22. Haniel Barbosa, Clark W. Barrett, Martin Brain, Gereon Kremer, Hanna Lachnitt, Makai Mann, Abdalrhman Mohamed, Mudathir Mohamed, Aina Niemetz, Andres Nötzli, Alex Ozdemir, Mathias Preiner, Andrew Reynolds, Ying Sheng, Cesare Tinelli, and Yoni Zohar. cvc5: A versatile and industrial-strength SMT solver. In *TACAS (1)*, volume 13243 of *Lecture Notes in Computer Science*, pages 415–442. Springer, 2022.

BD17. Stanley Bak and Parasara Sridhar Duggirala. HyLAA: A Tool for Computing Simulation-Equivalent Reachability for Linear Systems. In *HSCC*, pages 173–178. ACM, 2017.

BT24. Ludovico Battista and Stefano Tonetta. Formal verification of stability for parametric affine switched systems. *IFAC-PapersOnLine*, 58(11):37–42, 2024. 8th IFAC Conference on Analysis and Design of Hybrid Systems ADHS 2024.

BT25. Ludovico Battista and Stefano Tonetta. Deriving liveness properties of hybrid systems from reachable sets and lyapunov-like certificates. Accepted for publication in Proceedings of ATVA, available at https://es-static.fbk.eu/people/lbattista/pages/hybrid_ltl.html, 2025.

CÁS13. Xin Chen, Erika Ábrahám, and Sriram Sankaranarayanan. Flow*: An Analyzer for Non-linear Hybrid Systems. In *CAV*, volume 8044 of *Lecture Notes in Computer Science*, pages 258–263. Springer, 2013.

CCD+14. Roberto Cavada, Alessandro Cimatti, Michele Dorigatti, Alberto Griggio, Alessandro Mariotti, Andrea Micheli, Sergio Mover, Marco Roveri, and Stefano Tonetta. The nuxmv symbolic model checker. In Armin Biere and Roderick Bloem, editors, *CAV*, volume 8559 of *Lecture Notes in Computer Science*, pages 334–342. Springer, 2014.

CGMT14. Alessandro Cimatti, Alberto Griggio, Sergio Mover, and Stefano Tonetta. Verifying LTL Properties of Hybrid Systems with K-Liveness. In *CAV*, volume 8559 of *Lecture Notes in Computer Science*, pages 424–440. Springer, 2014.

CN12. Rebekah Carter and Eva M. Navarro-López. Dynamically-driven timed automaton abstractions for proving liveness of continuous systems. In Marcin Jurdzinski and Dejan Nickovic, editors, *Formal Modeling and Analysis of Timed Systems - 10th International Conference, FORMATS 2012, London, UK, September 18-20, 2012. Proceedings*, volume 7595 of *Lecture Notes in Computer Science*, pages 59–74. Springer, 2012.

CRT09. Alessandro Cimatti, Marco Roveri, and Stefano Tonetta. Requirements validation for hybrid systems. In Ahmed Bouajjani and Oded Maler, editors, *Computer Aided Verification*, pages 188–203, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

CS12. Koen Claessen and Niklas Sörensson. A liveness checking algorithm that counts. In *FMCAD*, pages 52–59. IEEE, 2012.

dMB08. Leonardo Mendonça de Moura and Nikolaj S. Bjørner. Z3: an efficient SMT solver. In *TACAS*, volume 4963 of *Lecture Notes in Computer Science*, pages 337–340. Springer, 2008.

FGD+11. Goran Frehse, Colas Le Guernic, Alexandre Donzé, Scott Cotton, Rajarshi Ray, Olivier Lebeltel, Rodolfo Ripado, Antoine Girard, Thao Dang, and Oded Maler. SpaceEx: Scalable Verification of Hybrid Systems. In *CAV*, volume 6806 of *Lecture Notes in Computer Science*, pages 379–395. Springer, 2011.

FI04. Ansgar Fehnker and Franjo Ivančić. Benchmarks for hybrid systems verification. In Rajeev Alur and George J. Pappas, editors, *Hybrid Systems: Computation and Control*, pages 326–341, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

FMQ+15. Nathan Fulton, Stefan Mitsch, Jan-David Quesel, Marcus Völp, and André Platzer. KeYmaera X: An axiomatic tactical theorem prover for hybrid systems. In Amy P. Felty and Aart Middeldorp, editors, *CADE*, volume 9195 of *LNCS*, pages 527–538. Springer, 2015.

GM15. Marco Gario and Andrea Micheli. Pysmt: a solver-agnostic library for fast prototyping of smt-based algorithms. In *SMT Workshop 2015*, 2015.

HS20. Hyejin Han and Ricardo G. Sanfelice. Linear temporal logic for hybrid dynamical systems: Characterizations and sufficient conditions. *Nonlinear Analysis: Hybrid Systems*, 36:100865, 2020.

Löf04. J. Löfberg. Yalmip : A toolbox for modeling and optimization in matlab. In *In Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004.

Mos. Mosek. `https://www.mosek.com/`.

PKV13. Erion Plaku, Lydia E. Kavraki, and Moshe Y. Vardi. Falsification of LTL safety properties in hybrid systems. *Int. J. Softw. Tools Technol. Transf.*, 15(4):305–320, 2013.

Pnu77. Amir Pnueli. The Temporal Logic of Programs. In *FOCS*, pages 46–57. IEEE Computer Society, 1977.

PW06. Andreas Podelski and Silke Wagner. Model Checking of Hybrid Systems: From Reachability Towards Stability. In *HSCC*, volume 3927 of *Lecture Notes in Computer Science*, pages 507–521. Springer, 2006.

pyV22. PyVmt: a python library to interact with transition systems. `https://github.com/pyvmt/pyvmt`, 2022. [Github repository].

SÁMK17. Stefan Schupp, Erika Ábrahám, Ibtissem Ben Makhlouf, and Stefan Kowalewski. HyPro: A C++ Library of State Set Representations for Hybrid Systems Reachability Analysis. In *NFM*, volume 10227 of *Lecture Notes in Computer Science*, pages 288–294, 2017.

SSB25. Giuseppe Spallitta, Roberto Sebastiani, and Armin Biere. Disjoint projected enumeration for SAT and SMT without blocking clauses. *Artificial Intelligence*, 345:104346, 2025.

WTL14. Tichakorn Wongpiromsarn, Ufuk Topcu, and Andrew G. Lamperski. Automata theory meets barrier certificates: Temporal logic verification of nonlinear systems. *IEEE Transactions on Automatic Control*, 61:3344–3355, 2014.

A Additional details on implementation

A.1 Parsing and Initialization

The tool starts by parsing the hybrid automaton H and the LTL property φ. It also accepts an optional set of user-defined local regions $\mathcal{Z}_{\text{add}}$. From these inputs, it extracts the initial set of local regions $\mathcal{Z}_{\text{in}}$, which includes the initial region and the local regions appearing in φ. These regions will serve as the basis for synthesis of additional regions and abstraction of the hybrid system.

Extraction of regions. We associate a boolean variable to the local region $init \wedge q_0$.

Then, we process φ as a DAG and prepare (a) a rewritten formula $\hat{\varphi}$ where certain subgraphs are abstracted by fresh Boolean variables, and (b) a dictionary $\mathcal{R}$ mapping each variable to a structured *Local Region*.

A subformula is a *Local Region* iff it has the shape $\psi(X) \wedge \left(\bigvee_{i \in I} q_i\right)$, where ψ is *semialgebraic* over X (no temporal operators, no symbols outside X) and each $q_i \in Q$. The location disjunction is tested after flattening nested `or`. We also treat any *pure* semialgebraic $\psi(X)$ as a *Local Region* with valid locations `all`. We use memoization to ensure that identical subgraphs are mapped to the same Boolean variable, see Algorithm 1.

B Additional details on implementation

We here recollect the tables with results from our experimental evaluation. For the multilayer control model, we study also another property (namely $G(\neg B)$) which is false on the hybrid system (and our tool correctly returns *Unknown*).

Algorithm 1: Traverse & Extract(φ, Q, X). $S(f)$ holds iff a DAG walk of f finds only arithmetic/Boolean connectives, symbols from X, and no temporal nodes. $D(f)$ holds iff, after flattening $\vee$, every disjunct is a symbol from Q.

$M \leftarrow \emptyset$ `// memoization (DAG-aware)`
$\mathcal{R} \leftarrow [\,]$ `// dictionary of` $(b, \text{LocalRegion})$
Function `Visit(`f`):`
  if $f \in M$ **then return** $M[f]$
  `// LocalRegion match in memoization`
  if $f \equiv \psi \wedge \lambda$ *with* $S(\psi)$ *and* $D(\lambda)$ **then**
    $b \leftarrow$ fresh Boolean
    $\mathcal{R} \leftarrow \mathcal{R} \cup [(b, \text{LocalRegion}(\psi, \text{valid} = \text{atoms}(\lambda)))]$
    $M[f] \leftarrow b$; **return** b
  `// Global semialgebraic constraint`
  if $S(f)$ **then**
    $b \leftarrow$ fresh Boolean
    $\mathcal{R} \leftarrow \mathcal{R} \cup [(b, \text{LocalRegion}(f, \text{valid} = \texttt{all}))]$
    $M[f] \leftarrow b$; **return** b
  `// Default: preserve structure; traverse children`
  Rebuild f' from children $\{\, \texttt{Visit}(g) \mid g \in \text{children}(f) \,\}$
  $M[f] \leftarrow f'$; **return** f'
$\widehat{\varphi} \leftarrow \texttt{Visit}(\varphi)$; **return** $(\widehat{\varphi}, \mathcal{R})$.

T_{TOT}	Config	T_{Lyap}	T_{ASJump}	T_{ASReg}	T_{AdReg}	T_{RSO}	T_{RSOMat}	$T\varphi_1$	Res φ_1	md_{RSO}	ms_{RSO}	ms_{bar}	N_{LTL}	$\lvert\mathcal{Z}\rvert$
23.53	CF1	0.07	0.01	0.01	0.14	4.28	10.45	0.27	True	16.00	141.33	0.00	3	4
23.31	CF2	0.07	0.01	0.01	0.14	4.21	10.30	0.27	True	16.00	141.33	0.00	3	4
23.94	CF3	0.07	0.01	0.01	0.14	4.23	10.33	0.61	True	16.00	141.33	0.00	3	4
23.86	CF4	0.07	0.01	0.01	0.14	4.20	10.33	0.58	True	16.00	141.33	0.00	3	4

Table 4. Computation times and statistics for model *circle* with $\tau = 0.1$ and $T = 10$.

T_{TOT}	Config	T_{Lyap}	T_{ASJump}	T_{ASReg}	T_{AdReg}	T_{RSO}	T_{RSOMat}	$T\varphi_1$	Res φ_1	$T\varphi_2$	Res φ_2	$T\varphi_3$	Res φ_3	md_{RSO}	ms_{RSO}	ms_{bar}	N_{LTL}	$\lvert\mathcal{Z}\rvert$
26.51	CF1	0.38	1.24	0.16	4.32	1.72	0.00	1.10	True	0.60	False	0.56	False	0.00	0.00	47.50	74	18
150.72	CF2	0.37	3.97	0.91	13.80	4.58	75.85	5.14	True	1.25	True	1.70	True	11.93	333.64	55.93	121	32
36.66	CF3	0.39	1.14	0.17	4.15	1.89	0.00	3.01	True	3.17	False	3.10	False	0.00	0.00	47.50	74	18
212.06	CF4	0.35	4.03	0.94	13.03	4.02	71.31	47.02	True	8.86	True	9.79	True	11.93	333.64	55.93	121	32

Table 5. Computation times and statistics for model *linear overtake* with $\tau = 0.1$ and $T = 10$.

Param	T_{TOT}	Config	T_{Lyap}	T_{ASJump}	T_{ASReg}	T_{AdReg}	T_{RSO}	T_{RSOMat}	$T\varphi_1$	Res φ_1	md_{RSO}	ms_{RSO}	ms_{bar}	N_{LTL}	$\|\mathcal{Z}\|$
1	108.70	CF1	0.20	0.09	0.03	0.98	15.73	80.40	0.47	True	4.21	46.37	84.00	77	9
1	256.12	CF2	0.21	0.24	0.13	3.95	19.53	216.75	1.31	True	6.39	161.97	106.14	113	21
1	112.31	CF3	0.20	0.09	0.03	0.96	15.74	80.74	2.42	True	4.21	46.37	84.00	77	9
1	258.26	CF4	0.20	0.24	0.13	3.91	19.03	211.80	5.66	True	6.39	161.97	106.14	113	21
2	143.74	CF1	0.27	0.16	0.04	1.09	39.02	89.76	0.88	True	3.40	45.24	113.14	107	9
2	478.39	CF2	0.26	0.50	0.21	6.38	56.59	385.62	3.70	True	5.84	180.88	174.00	203	33
2	143.57	CF3	0.26	0.16	0.04	1.09	37.57	87.08	3.06	True	3.40	45.24	113.14	107	9
2	490.25	CF4	0.26	0.49	0.20	6.30	56.90	387.89	10.10	True	5.84	180.88	174.00	203	33
3	201.44	CF1	0.38	0.20	0.04	1.17	85.17	99.76	1.09	True	2.94	44.52	142.29	137	9
3	776.11	CF2	0.39	0.75	0.27	8.42	138.25	588.79	13.49	True	5.25	183.07	246.57	293	45
3	209.38	CF3	0.37	0.20	0.04	1.17	85.38	100.20	4.75	True	2.94	44.52	142.29	137	9
3	791.53	CF4	0.37	0.76	0.28	8.33	139.15	590.58	16.82	True	5.25	183.07	246.57	293	45
4	306.59	CF1	0.53	0.26	0.05	1.28	179.98	108.49	1.63	True	2.54	44.86	171.43	167	9
4	1131.90	CF2	0.52	1.02	0.34	10.57	310.22	757.12	14.75	True	4.81	181.58	319.14	383	57
4	317.74	CF3	0.51	0.26	0.05	1.30	180.14	109.90	6.77	True	2.54	44.86	171.43	167	9
4	1179.63	CF4	0.53	1.02	0.34	10.38	311.12	760.66	31.96	True	4.81	181.58	319.14	383	57
5	437.26	CF1	0.72	0.33	0.07	1.34	328.91	88.11	2.10	True	2.32	47.98	192.86	187	9
5	1503.05	CF2	0.72	1.24	0.39	11.00	593.36	830.46	24.37	True	4.82	198.28	373.43	443	65
5	454.89	CF3	0.72	0.33	0.07	1.37	334.69	89.61	7.44	True	2.32	47.98	192.86	187	9
5	1531.28	CF4	0.71	1.24	0.38	11.26	598.35	834.98	27.64	True	4.82	198.28	373.43	443	65
6	821.07	CF1	0.96	0.41	0.07	1.54	693.26	103.96	2.14	True	2.26	53.13	222.00	217	9
6	833.14	CF3	0.98	0.39	0.07	1.50	694.65	104.14	7.67	True	2.26	53.13	222.00	217	9
6	3581.93	CF4	1.20	1.94	0.59	21.69	1831.63	1583.65	50.02	True	4.63	206.79	446.00	533	77
7	1540.38	CF1	1.24	0.44	0.08	1.59	1396.75	117.07	2.65	True	2.19	57.57	251.14	247	9
7	1573.82	CF3	1.26	0.44	0.08	1.62	1419.07	118.34	9.18	True	2.19	57.57	251.14	247	9

Table 6. Computation times and statistics for model *multiple choice* with $\tau = 0.1$ and $T = 10$.

Param	T_{TOT}	Config	T_{Lyap}	T_{ASJump}	T_{ASReg}	T_{AdReg}	T_{RSO}	T_{RSOMat}	$T\varphi_1$	Res φ_1	$T\varphi_2$	Res φ_2	$T\varphi_3$	Res φ_3	md_{RSO}	ms_{RSO}	ms_{bar}	N_{LTL}	$\|\mathcal{Z}\|$
3	18.20	CF1	0.28	0.83	0.06	1.93	1.79	0.00	0.24	False	0.22	True	0.23	False	0.00	0.00	36.41	41	20
3	286.87	CF2	0.28	4.55	1.16	15.30	8.69	202.67	17.08	False	1.12	True	14.95	False	10.76	364.29	55.24	143	54
3	22.06	CF3	0.28	0.79	0.06	1.85	1.70	0.00	1.30	False	1.26	True	1.26	False	0.00	0.00	36.41	41	20
3	318.65	CF4	0.28	4.55	1.13	15.31	8.69	203.72	24.54	False	9.23	True	21.48	False	10.76	364.29	55.24	143	54
4	18.97	CF1	0.32	1.05	0.05	1.96	1.31	0.00	0.34	False	0.28	True	0.38	False	0.00	0.00	29.27	45	24
4	629.88	CF2	0.34	17.44	2.53	27.03	18.08	481.95	10.21	True	4.80	True	38.34	False	9.44	410.61	54.00	243	90
4	25.57	CF3	0.33	1.02	0.05	1.90	1.34	0.00	1.74	False	1.90	True	2.06	False	0.00	0.00	29.27	45	24
4	778.66	CF4	0.34	17.39	2.49	27.44	18.78	501.86	51.55	True	40.85	True	54.76	False	9.44	410.61	54.00	243	90
5	24.84	CF1	0.67	2.87	0.10	2.67	1.54	0.00	0.38	False	0.37	True	0.36	False	0.00	0.00	28.89	55	29
5	2551.01	CF2	0.66	139.27	19.45	104.43	46.58	1443.61	67.26	True	26.25	True	644.26	False	9.46	518.33	59.78	379	137
5	31.85	CF3	0.68	2.81	0.10	2.64	1.46	0.00	3.05	False	1.95	True	1.60	False	0.00	0.00	28.89	55	29
5	3509.08	CF4	0.78	133.79	19.22	106.37	50.30	1545.54	351.72	True	129.52	True	828.32	False	9.46	518.33	59.78	379	137
6	28.11	CF1	0.80	3.31	0.11	2.90	1.68	0.00	0.52	False	0.52	True	0.48	False	0.00	0.00	28.62	65	34
6	38.91	CF3	0.78	3.39	0.11	2.89	1.59	0.00	2.84	False	2.47	True	4.70	False	0.00	0.00	28.62	65	34
7	35.47	CF1	1.12	6.30	0.14	3.66	1.84	0.00	0.69	False	0.67	True	0.64	False	0.00	0.00	28.43	75	39
7	49.69	CF3	1.19	6.22	0.14	3.60	1.74	0.00	3.49	False	3.19	True	3.56	False	0.00	0.00	28.43	75	39

Table 7. Computation times and statistics for model *multilayer control linear* with $\tau = 0.1$ and $T = 10$.

Param	T_{TOT}	Config	T_{Lyap}	T_{ASJump}	T_{ASReg}	T_{AdReg}	T_{RSO}	T_{RSOMat}	$T\varphi_1$	Res φ_1	$T\varphi_2$	Res φ_2	$T\varphi_3$	Res φ_3	md_{RSO}	ms_{RSO}	ms_{bar}	N_{LTL}	$\|\mathcal{Z}\|$
3	20.59	CF1	0.45	0.88	0.07	2.01	3.45	0.00	0.24	False	0.19	True	0.23	False	0.00	0.00	36.41	41	20
3	50.47	CF2	0.45	4.50	1.15	14.34	5.58	0.00	1.31	False	0.73	True	1.21	False	0.00	0.00	55.24	109	54
3	23.51	CF3	0.48	0.85	0.07	2.06	2.12	0.00	1.31	False	1.23	True	1.31	False	0.00	0.00	36.41	41	20
3	72.33	CF4	0.45	4.50	1.16	14.39	5.67	0.00	6.05	False	5.11	True	8.13	False	0.00	0.00	55.24	109	54
4	20.04	CF1	0.48	1.09	0.06	1.95	1.42	0.00	0.32	False	0.25	True	0.31	False	0.00	0.00	29.27	45	24
4	103.61	CF2	0.47	17.83	2.84	27.45	11.46	0.00	6.17	False	2.71	True	6.83	False	0.00	0.00	54.00	177	90
4	25.59	CF3	0.49	1.09	0.05	1.95	1.33	0.00	1.77	False	1.89	True	2.01	False	0.00	0.00	29.27	45	24
4	200.76	CF4	0.48	17.95	2.86	28.00	9.47	0.00	26.59	False	25.02	True	40.88	False	0.00	0.00	54.00	177	90
5	26.63	CF1	1.10	3.09	0.10	2.83	1.55	0.00	0.39	False	0.34	True	0.43	False	0.00	0.00	28.89	55	29
5	323.78	CF2	1.10	108.36	17.06	65.95	17.56	0.00	26.20	False	11.34	True	29.13	False	0.00	0.00	59.78	271	137
5	36.12	CF3	1.10	3.01	0.10	2.78	1.47	0.00	2.06	False	3.07	True	3.02	False	0.00	0.00	28.89	55	29
5	844.69	CF4	1.11	108.04	17.12	66.08	15.65	0.00	161.53	False	142.06	True	155.40	False	0.00	0.00	59.78	271	137
6	30.89	CF1	1.93	3.49	0.10	3.01	1.70	0.00	0.54	False	0.46	True	0.48	False	0.00	0.00	28.62	65	34
6	741.42	CF2	1.25	279.19	32.94	105.56	27.57	0.00	84.05	False	36.60	True	104.73	False	0.00	0.00	63.94	373	194
6	42.66	CF3	1.25	3.59	0.11	3.16	1.58	0.00	4.51	False	2.65	True	2.43	False	0.00	0.00	28.62	65	34
6	2611.80	CF4	2.99	356.67	44.31	149.62	36.25	0.00	585.23	False	345.21	True	519.90	False	0.00	0.00	63.94	373	194
7	39.08	CF1	3.01	6.33	0.14	3.73	1.86	0.00	0.81	False	0.59	True	0.69	False	0.00	0.00	28.43	75	39
7	52.24	CF3	2.74	6.26	0.14	3.73	1.73	0.00	3.34	False	6.10	True	3.22	False	0.00	0.00	28.43	75	39

Table 8. Computation times and statistics for model *multilayer control nonlinear* with $\tau = 0.1$ and $T = 10$.

T_{TOT}	Config	T_{Lyap}	T_{ASJump}	T_{ASReg}	T_{AdReg}	T_{RSO}	T_{RSOMat}	$T\varphi_1$	Res φ_1	md_{RSO}	ms_{RSO}	ms_{bar}	N_{LTL}	$\lvert\mathcal{Z}\rvert$
13.48	CF1	0.00	0.01	0.00	0.08	2.00	2.49	0.05	True	7.00	63.00	0.00	5	3
12.91	CF2	0.00	0.01	0.01	0.08	1.98	2.56	0.06	True	7.00	63.00	0.00	5	3
13.37	CF3	0.00	0.01	0.01	0.09	2.01	2.52	0.08	True	7.00	63.00	0.00	5	3
13.72	CF4	0.00	0.01	0.01	0.08	2.13	2.60	0.09	True	7.00	63.00	0.00	5	3

Table 9. Computation times and statistics for model *bball* with $\tau = 0.1$ and $T = 10$.

Param	T_{TOT}	Config	T_{Lyap}	T_{ASJump}	T_{ASReg}	T_{AdReg}	T_{RSO}	T_{RSOMat}	$T\varphi_1$	Res φ_1	$T\varphi_2$	Res φ_2	md_{RSO}	ms_{RSO}	ms_{bar}	N_{LTL}	$\lvert\mathcal{Z}\rvert$
0.05	428.42	CF1	0.11	0.00	0.03	1.71	134.28	277.59	5.39	True	0.15	True	13.00	275.00	0.00	6	13
0.05	11.11	CF1	0.11	0.00	0.00	0.02	3.29	0.00	0.04	False	0.03	False	0.00	0.00	0.00	0	2
0.05	11.14	CF2	0.12	0.00	0.00	0.02	3.37	0.00	0.02	False	0.02	False	0.00	0.00	0.00	0	2
0.05	427.16	CF2	0.12	0.00	0.03	1.74	134.65	276.06	5.43	True	0.15	True	13.00	275.00	0.00	6	13
0.05	11.15	CF3	0.11	0.00	0.00	0.02	3.31	0.00	0.03	False	0.03	False	0.00	0.00	0.00	0	2
0.05	413.31	CF3	0.11	0.00	0.03	1.72	126.29	267.73	6.02	True	0.64	True	13.00	275.00	0.00	6	13
0.05	11.15	CF4	0.11	0.00	0.00	0.02	3.30	0.00	0.03	False	0.03	False	0.00	0.00	0.00	0	2
0.05	423.54	CF4	0.11	0.00	0.03	1.72	133.22	271.16	6.02	True	0.64	True	13.00	275.00	0.00	6	13
0.08	216.01	CF1	0.14	0.00	0.03	1.70	104.25	95.76	4.95	True	0.11	True	14.00	303.00	0.00	2	13
0.08	10.83	CF1	0.12	0.00	0.00	0.02	3.53	0.00	0.02	False	0.03	False	0.00	0.00	0.00	0	2
0.08	213.54	CF2	0.12	0.00	0.03	1.72	102.08	96.17	4.91	True	0.11	True	14.00	303.00	0.00	2	13
0.08	10.64	CF2	0.11	0.00	0.00	0.02	2.98	0.00	0.02	False	0.02	False	0.00	0.00	0.00	0	2
0.08	210.74	CF3	0.12	0.00	0.03	1.71	99.78	95.41	4.55	True	0.32	True	14.00	303.00	0.00	2	13
0.08	10.68	CF3	0.12	0.00	0.00	0.02	2.97	0.00	0.03	False	0.03	False	0.00	0.00	0.00	0	2
0.08	213.64	CF4	0.11	0.00	0.03	1.74	99.72	98.75	4.55	True	0.31	True	14.00	303.00	0.00	2	13
0.08	10.88	CF4	0.11	0.00	0.00	0.02	3.01	0.00	0.03	False	0.03	False	0.00	0.00	0.00	0	2
0.11	182.34	CF1	0.11	0.00	0.03	1.71	72.75	94.97	5.03	True	0.09	True	14.00	295.00	0.00	2	13
0.11	11.15	CF1	0.11	0.00	0.00	0.02	3.34	0.00	0.02	False	0.02	False	0.00	0.00	0.00	0	2
0.11	10.07	CF2	0.11	0.00	0.00	0.02	2.79	0.00	0.02	False	0.02	False	0.00	0.00	0.00	0	2
0.11	178.84	CF2	0.11	0.00	0.03	1.71	73.04	91.48	4.99	True	0.10	True	14.00	295.00	0.00	2	13
0.11	188.87	CF3	0.11	0.00	0.03	1.74	76.01	97.12	5.31	True	0.28	True	14.00	295.00	0.00	2	13
0.11	10.85	CF3	0.11	0.00	0.00	0.02	2.82	0.00	0.04	False	0.03	False	0.00	0.00	0.00	0	2
0.11	10.65	CF4	0.12	0.00	0.00	0.02	2.81	0.00	0.03	False	0.03	False	0.00	0.00	0.00	0	2
0.11	177.85	CF4	0.12	0.00	0.03	1.72	72.07	90.50	5.30	True	0.28	True	14.00	295.00	0.00	2	13

Table 10. Computation times and statistics for model *nonlinear circle* with $\tau = 0.01$ and $T = 10$.

Param	T_{TOT}	Config	T_{Lyap}	T_{ASJump}	T_{ASReg}	T_{AdReg}	T_{RSO}	T_{RSOMat}	$T\varphi_1$	Res φ_1	$T\varphi_2$	Res φ_2	md_{RSO}	ms_{RSO}	ms_{bar}	N_{LTL}	$\lvert\mathcal{Z}\rvert$
100	105.06	CF1	0.06	0.02	0.01	0.00	95.05	0.00	0.03	False	0.03	True	0.00	0.00	0.00	2	2
100	104.06	CF2	0.06	0.01	0.01	0.00	93.17	0.00	0.03	False	0.03	True	0.00	0.00	0.00	2	2
100	100.28	CF3	0.06	0.02	0.01	0.00	89.68	0.00	0.05	False	0.04	True	0.00	0.00	0.00	2	2
100	101.01	CF4	0.06	0.02	0.01	0.00	90.33	0.00	0.05	False	0.04	True	0.00	0.00	0.00	2	2
5	115.12	CF1	0.05	0.01	0.01	0.00	105.49	0.00	0.03	False	0.02	True	0.00	0.00	0.00	2	2
5	113.78	CF2	0.06	0.02	0.01	0.00	104.08	0.00	0.03	False	0.02	True	0.00	0.00	0.00	2	2
5	117.17	CF3	0.06	0.01	0.01	0.00	107.70	0.00	0.04	False	0.04	True	0.00	0.00	0.00	2	2
5	117.23	CF4	0.06	0.02	0.01	0.00	107.07	0.00	0.05	False	0.04	True	0.00	0.00	0.00	2	2

Table 11. Computation times and statistics for model *drivetrain* with $\tau = 0.1$ and $T = 10$.

Param	T_{TOT}	Config	T_{Lyap}	T_{ASJump}	T_{ASReg}	T_{AdReg}	T_{RSO}	T_{RSOMat}	$T\varphi_1$	Res φ_1	$T\varphi_2$	Res φ_2	md_{RSO}	ms_{RSO}	ms_{bar}	N_{LTL}	$\lvert\mathcal{Z}\rvert$
NoPass6	17.62	CF1	0.71	0.02	0.05	5.85	0.00	0.00	0.05	True	0.05	False	0.00	0.00	24.67	11	5
NoPass6	18.20	CF2	0.72	0.02	0.05	5.80	0.00	0.00	0.05	True	0.05	False	0.00	0.00	24.67	11	5
NoPass6	18.35	CF3	0.70	0.02	0.05	5.81	0.00	0.00	0.10	True	0.12	False	0.00	0.00	24.67	11	5
NoPass6	17.98	CF4	0.68	0.02	0.05	5.61	0.00	0.00	0.10	True	0.12	False	0.00	0.00	24.67	11	5
NonLin4	11.22	CF1	0.82	0.00	0.00	0.02	2.41	0.00	0.02	True	0.03	False	0.00	0.00	0.00	0	2
NonLin4	11.21	CF2	0.83	0.00	0.00	0.02	2.37	0.00	0.02	True	0.02	False	0.00	0.00	0.00	0	2
NonLin4	11.37	CF3	0.82	0.00	0.00	0.02	2.36	0.00	0.02	True	0.03	False	0.00	0.00	0.00	0	2
NonLin4	11.20	CF4	0.83	0.00	0.00	0.02	2.39	0.00	0.02	True	0.03	False	0.00	0.00	0.00	0	2
NonLinPas4	11.24	CF1	1.56	0.01	0.01	0.02	0.80	0.00	0.04	True	0.05	False	0.00	0.00	0.00	6	2
NonLinPas4	11.53	CF2	1.47	0.01	0.01	0.02	0.79	0.00	0.04	True	0.05	False	0.00	0.00	0.00	6	2
NonLinPas4	12.00	CF3	1.56	0.01	0.01	0.02	0.83	0.00	0.05	True	0.08	False	0.00	0.00	0.00	6	2
NonLinPas4	11.71	CF4	1.57	0.01	0.01	0.02	0.82	0.00	0.06	True	0.08	False	0.00	0.00	0.00	6	2

Table 12. Computation times and statistics for model *rendezvous* with $\tau = 0.1$ and $T = 10$.