Formal Verification Applied to Autonomous Spacecraft Attitude Control*

Kendra Lang†
Verus Research, Albuquerque, NM, 87110

Corbin Klett‡, Kelsey Hawkins§, Eric Feron¶, Panagiotis Tsiotras‖
Georgia Institute of Technology, Atlanta, GA, 30332

Sean Phillips**
Air Force Research Laboratory, Albuquerque, NM, 87117

Formal verification tools are cited as an essential component to enable more widespread development and adoption of advanced autonomous systems. While numerous techniques and tools exist, the applicability of these tools to actual systems under development is currently uncertain. There are myriad reasons for such uncertainty, mostly stemming from assumptions necessary for such tools to work, such as: 1) The assumption that an underlying dynamics model or Simulink model is available, 2) The assumption that the dynamics are low-dimensional, 3) The assumption that the dynamics are linear or linearizable without sacrificing accuracy, and 4) The assumption that the underlying controllers and autonomy algorithms are available and easily modeled. This paper first presents a novel satellite benchmark that incorporates autonomous switching between multiple modes of operation related to attitude control. The result is a hybrid system with nonlinear rotational dynamics restricted to a manifold within each mode. Several open source verification tools are then applied to this benchmark to determine any results that can be drawn about the stability of the overall system. We provide a thorough comparison and discussion of the benefits and drawbacks of those tools we tested, none of which were capable of completely verifying stability requirements over the entire benchmark to the best of our efforts. We also discuss the significant hurdles that remain to implementing these tools on realistic autonomous systems, and the techniques we have found to be the most applicable. The contributions of this paper are: 1) a challenging benchmark on which developers can test their verification tools, and 2) a useful starting point to anyone who wants to apply formal methods to autonomous aerospace systems and to advance the conversation on what remains to be accomplished for these tools to be of practical use.

I. Nomenclature

J	=	Moment of inertia matrix
J^r	=	Reaction wheel moment of inertia martix
ω	=	Angular velocity
q	=	Quaternion attitude representation
w	=	Reaction wheel angular velocity
τ^t	=	Thruster torque
τ^r	=	Reaction wheel torque
S	=	Direction cosine matrix

*This work funded under SBIR Contract FA9453-18-P-0225. Approved for public release; distribution is unlimited. Public affairs approval #AFRL-RV-40142.

†Technical Director, Space and Autonomy, kendra.lang@verusresearch.net

‡Graduate student, School of Aerospace Engineering.

§Graduate student, School of Aerospace Engineering

¶Dutton/Ducoffe Professor of Aerospace Software Engineering, School of Aerospace Engineering

‖Davis and Andrew Lewis Chair, School of Aerospace Engineering

**Research Mechanical Engineer, Space Vehicles Directorate

II. Introduction

Autonomous systems require careful analysis prior to deployment to ensure that they meet all requirements as defined during their conception. Given their complexity, potentially high cost, and high consequences when they do not perform as desired, exhaustive testing, verification and validation are necessary. Unfortunately, simulation and live testing alone cannot cover all possible scenarios that the autonomous system will encounter. There are always edge cases and environmental effects that are not tested for but which may cause problems once the system is deployed. Formal verification is a recognized necessity for deploying such systems with confidence. Formal methods use mathematical principles to exhaustively and intelligently identify requirement violations, or to guarantee that no violations will occur. For dynamical systems, formal verification is used to check requirements, often formulated as logic expressions, such as reach and avoid properties: Does the state of the system eventually reach a desired target state, or always avoid some undesired set of states?

Numerous tools and techniques have been developed to apply formal methods to dynamical systems, many of which employ some form of reachability analysis, in which sets of states are propagated through the dynamics of the system. In all but the simplest of cases, such set propagation does not have an analytical solution and must be approximated. Different techniques require different assumptions and simplifications in order for that particular approximation to apply. Linearity is a common assumption and many tools require linear dynamics, or the ability to linearize the dynamics, such as SpaceEx [1] and CORA [2]. When the dynamics are nonlinear, the number of variables must often remain quite low. In some cases the dimensionality may be higher, such as with Flow* [3] or C2E2 [4] if the nonlinearities satisfy certain properties. Some tools, such as the Level Set Toolbox [5] allow disturbances or unknown control inputs, while others must have explicitly defined dynamics with no uncertainty, such as dReach [6]. Finally, the length of time over which reachable sets are propagated must often be short. Computing the set of states reachable from an initial set over a long time horizon (more than a few seconds) leads to approximations that become too inaccurate to be of use.

In this work, we compare a handful of open source verification tools that we believe is representative of the different reachability analysis techniques available. We do not, however, claim that this list is in any way exhaustive. We also omit simulation- and falsification-based techniques, which we will touch on in Section IV. The tools that we compare include the Level Set Toolbox, Flow*, CORA, C2E2, dReach and CoCoSim [7].

There are a few benchmarks that are widely used to compare verification tools, and authors of papers describing new verification tools often do a good job of comparing their technique to another similar technique developed elsewhere. Yet, there are few comprehensive survey papers of multiple verification techniques applied to the same benchmark. Further, benchmarks are often tailored so that the verification techniques in question can be applied to them, and then the comparison made is in terms of accuracy and computation time. The discussion of *if* such tools are applicable is then avoided. Chen et. al. [8] describe a suite of continuous and hybrid models with linear and nonlinear dynamics and compare three verification tools (SpaceEx, Flow* and dReach) on each benchmark. Most of the nonlinear benchmarks presented have only three state variables and both dReach and Flow* are able to compute reachable sets for them (SpaceEx is only applicable to linear systems). Hybrid system benchmarks are presented in [9] although the emphasis is on complexity from the hybrid switching conditions, and the dynamics within each mode are fairly simple and low-dimensional. A realistic automotive powertrain control benchmark is presented in [10] although tellingly, only a falsification-based tool S-Taliro [11] is tested on it. One important contribution is the workshop on Applied Verification of Continuous and Hybrid systems (ARCH) [12], which provides a yearly compilation of benchmarks and a friendly competition amongst verification tools broken down by category (hybrid, nonlinear, etc.) with a focus on industry applications.

We have found that a more realistic benchmark causes many reachability tools to break down. The question of if and when such tools are applicable is therefore highly relevant. To that end, the contribution of this paper is twofold. First, we present a novel spacecraft benchmark with several unique features to challenge existing (and new) verification tools. These features include: Eleven state variables, four modes of operation with state-triggered switching between modes, and an attitude state represented by a quaternion which is restricted to lie on the unit norm manifold. Second, we provide a comparison of six distinct verification tools as applied to our spacecraft benchmark. We are specifically interested in the stability of the system, defined as the ability to reach and stay within one of the operating modes of the spacecraft. None of the tools can verify this property, but we do discuss the sub-problems that can be tackled with each tool and provide a comparison across those. In doing so, we identify the gaps in current techniques and provide recommendations for how to apply formal methods to autonomous systems and where improvements and research efforts are needed.

The paper is organized as follows. In Section III, we describe the spacecraft benchmark in detail. In Section IV, we provide an overview of each of the verification tools that we compare. In Section V, we present the results of our

analysis using each verification tool and discuss those results. Conclusions are given in Section VI.

III. Spacecraft Benchmark

The benchmark we present here is designed to include the most fundamental aspects of a satellite system that must be considered when verifying controller performance. These elements include: Nonlinear dynamics, rotational motion restricted to a manifold, and state-based mode switching whereby each mode of operation is governed by different objectives and controllers. The benchmark comprises a satellite with thrusters, four reaction wheels in a standard pyramid configuration, and four modes of operation that it autonomously switches between. The dynamics within each mode are captured by the basic equations of motion for a rigid body. The attitude itself is parameterized by a quaternion. The torques acting on the satellite are from the reaction wheels, the thrusters, or a combination thereof. No additional external forces are considered, nor are any detailed actuation constraints other than saturation of the reaction wheels.

We use a quaternion to define the attitude of the spacecraft, namely, $q = [q_0, q_1, q_2, q_3]^\top = [q_0, \eta^\top]^\top$ where the scalar is q_0 and the vector component is η. The basic rotational dynamics expressed in the body reference frame governing the angular velocity of the spacecraft, ω, and the kinematics for the quaternion are given by

$$J\dot{\omega} = \tau^t - \omega \times J\omega, \tag{1}$$

$$\dot{q} = \frac{1}{2}\begin{bmatrix} -q_1 & -q_2 & -q_3 \\ q_0 & -q_3 & q_2 \\ q_3 & q_0 & -q_1 \\ -q_2 & q_1 & q_0 \end{bmatrix}\omega, \tag{2}$$

where τ^t represents thruster torques about the center of mass of the spacecraft in the body frame and J is the moment of inertia matrix for the spacecraft.

For an unactuated body, the total angular momentum is simply $L = J\omega$. However, rotating the reaction wheels induces a moment on the spacecraft, which enables control of the spacecraft's angular velocity by applying torques to the wheels. The vector of reaction wheel velocities is given by w, controlled by a torque vector τ^r, with the wheel dynamics expressed as

$$J^r\dot{w} = \tau^r, \tag{3}$$

where J^r is the moment of inertia for the wheels. The contribution of each of the wheels to the total angular momentum of the system is projected onto the body frame by use of the direction cosine matrix S:

$$S = \begin{bmatrix} \cos(\beta) & 0 & -\cos(\beta) & 0 \\ 0 & \cos(\beta) & 0 & -\cos(\beta) \\ \sin(\beta) & \sin(\beta) & \sin(\beta) & \sin(\beta) \end{bmatrix}, \tag{4}$$

where $\beta = \pi/6$ is the angle between each wheel's axis and the x-y plane. The full angular momentum of the actuated spacecraft is $L = J\omega + SJ^r w$. The dynamics in (1) are simplified using this relationship as

$$J\dot{\omega} = \tau^t - S\tau^r - \omega \times (J\omega + SJ^r w)\,. \tag{5}$$

The full state of the spacecraft is given by (q, ω, w), i.e., the quaternion attitude parameterization, angular velocity, and reaction wheel speeds, resulting in an 11-dimensional system whose state evolves according to (5), (3), and (2).

In this paper, we consider the case where the satellite can switch autonomously between modes. Specifically, the satellite switches between four distinct modes of operation: Detumble, Dump, Slew, and Stabilize. Detumble is used to reduce the total angular momentum of the spacecraft and is often used when a satellite is first launched from a deployment vehicle, such as an ESPA ring, or after a loss of control when the spacecraft begins to spin. Dump is used when the reaction wheels have saturated, meaning they can no longer generate torques to control the satellite. In Dump mode the thrusters are activated to reduce the velocity of the reaction wheels. In reality, other actuators such as magnetic torque rods may be used continuously for wheel desaturation, but for simplicity we assume that only thrusters are available and are therefore triggered only when necessary. Slew mode drives the satellite to a commanded pointing configuration. Stabilize mode can be thought of as an Inertial Hold mode, where once the pointing command is reached, Stabilize mode maintains that commanded attitude. The transitions between modes are depicted in Figure 1.

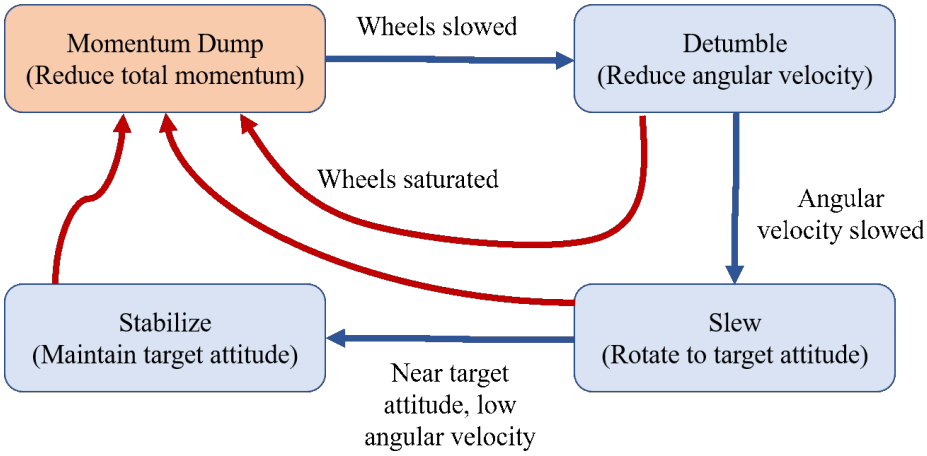


Fig. 1 Modes of the hybrid satellite benchmark.

Within each mode, a simple proportional-derivative controller was developed that stabilizes the dynamics to converge to the desired target state within that mode. Lyapunov functions were derived proving stability for each mode individually. Transitions between modes are triggered automatically based on conditions on the current state of the spacecraft (note these are not pre-determined or time-triggered). The state feedback employed both for within-mode controllers and mode-switching are assumed to operate in continuous time and with perfect, complete knowledge of the spacecraft's state. A description of each mode, its controller, and the switching behavior, is given below.

A. Detumble

Detumble mode is designed to bring the angular velocity of the spacecraft below a certain threshold, thus eliminating any excess spin. The controller uses the reaction wheels to generate counter torques to reduce the angular velocity and does not use the thrusters. The controller commands the following torques.

$$\tau^r = S^\dagger K_d^{\text{det}} \omega + K_h M_h w, \tag{6}$$

$$\tau^t = 0, \tag{7}$$

where $S^\dagger$ denotes the pseudoinverse of S. The control gain is K_d^{det} and $K_h M_h w$ is an inner control loop run in all modes that slowly tries to balance the momentum across the wheels. K_h is a gain constant and M_h is a momentum distribution matrix,

$$M_h = \begin{bmatrix} -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix}. \tag{8}$$

Detumble mode switches to Slew once the following condition is met:

$$\bigwedge_{i=x,y,z} |\omega_i| \leq \omega_{\text{det}}. \tag{9}$$

The parameter ω_{det} is a value near zero, which we set to 0.01 rad/sec. Alternately, Detumble switches to Dump mode when the following guard condition is satisfied:

$$\bigvee_{i=1,2,3,4} |w_i| > w_{\text{sat}}. \tag{10}$$

The parameter w_{sat} is the saturation value of the reaction wheels, which we set to 600 rad/sec.

B. Dump

The Dump mode is designed to reduce the angular velocity of the reaction wheels below a given threshold. Dump mode is entered automatically when any one of the wheels reaches the saturation limit of 600 rad/sec. Thrusters are used in Dump mode to generate the necessary torque required to reduce the angular velocity of the reaction wheels, and the controller commands the following torques.

$$\tau^r = -K_d^{\text{dump}}\omega + K_h M_h w \tag{11}$$

$$\tau^t = -SK_d^{\text{dump}} w \tag{12}$$

The transition from Dump mode to Detumble occurs when the following condition is satisfied:

$$\bigwedge_{i=1,2,3,4} |w_i| \leq w_{\text{dump}}. \tag{13}$$

The parameter w_{dump} is a value close to zero, which we set to 30 rad/sec.

C. Slew

The Slew mode is designed to point the spacecraft towards a single point (it could alternately track a reference trajectory). Once the spacecraft is close enough to its target, and the angular velocity is sufficiently low, Slew mode then transitions to Stabilize. Alternately, if the reaction wheels saturate during the slewing maneuver, the spacecraft will instead transition back to Dump. The controller in Slew mode commands the following torques.

$$\tau^r = -S^\dagger \left(2K_p^{\text{slew}} q_0 \eta + K_d^{\text{slew}} \omega\right) - S^\dagger \omega \times (J\omega + SJ^r w) + K_h M_h w, \tag{14}$$

$$\tau^t = 0. \tag{15}$$

Note that the Slew controller linearizes the angular velocity dynamics by compensating for the Coriolis term $\omega \times (J\omega + SJ^r w)$. Slew mode transitions to Stabilize when the following condition is met:

$$\left(\bigwedge_{i=0,1,2,3} |q_i - q_i^d| < q_{\text{tgt}}\right) \bigvee \left(\bigwedge_{i=0,1,2,3} |q_i + q_i^d| < q_{\text{tgt}}\right) \bigwedge \left(\bigwedge_{i=x,y,z} |\omega_i| < \omega_{\text{tgt}}\right), \tag{16}$$

with $q_{tgt} = 0.02$, $\omega_{\text{tgt}} = 0.03$, and q^d is the desired quaternion. Alternately, Slew will transition to Dump when condition (10) is satisfied.

D. Stabilize

Stabilize mode is similar to Slew, but increases the control gains to quickly keep the spacecraft pointed at a single attitude. The controller and dynamics in Stabilize mode are thus the same as in Slew mode, but the gains are higher. Once in Stabilize, the spacecraft either remains in Stabilize or transitions back to Dump if the wheels saturate.

E. Requirements

There are numerous potential requirements that may be placed on this benchmark. Indeed, generating a requirements list is itself a complicated and time-consuming task. For the purposes of this paper, the primary requirement to consider is stable behavior. Within-mode stability has already been proven via Lyapunov functions for each mode individually. However, multi-mode hybrid systems can produce instabilities by introducing oscillations between modes, or never converging to the desired state value because of the interplay between modes and its effect on the state's overall dynamics. It is this type of "global" stability across modes that is of primary interest, i.e. will the system converge to Stabilize and remain there? It should be noted that this benchmark has not been intentionally designed to exhibit any specific behavior, other than within-mode stability.

IV. Verification Tools

The benchmark developed in Section 2.1 was used to begin a study of existing open-source (and primarily academic) verification tools that are currently available. The objective was to determine what, if any, existing tools and/or the underlying theoretical concepts they utilize, are applicable to spacecraft hybrid attitude verification. The following tools were picked for comparison: the Level Set Toolbox, Flow*, dReach, CoCoSim, C2E2, and CORA. Each tool uses a quite distinct approach to verification. We briefly describe each tool below, including the underlying techniques it employs, the user interface provided, and to what elements of the spacecraft benchmark it may apply.

A. Level Set Toolbox

The Level Set Toolbox (LST) uses Hamilton-Jacobi methods to construct forward and backward reachable sets. It represents the reachable set as the zero sub-level set of a value function $V(x,t)$ and then propagates that value function over time as the solution to a Hamilton-Jacobi-Bellman partial differential equation. This method allows the LST to generate reachable sets for nonlinear dynamics with both control and disturbance inputs that act to either grow or shrink the reachable set. This generality is also a limitation, in that the LST computes the value function by gridding the state space and computation time scales exponentially with the dimension of the system. The technique employed by the LST is in general therefore limited to systems with at most four dimensions and its accuracy is dependent on the fineness of the grid used. Recent developments extend Hamilton-Jacobi reachability to higher-dimensional nonlinear systems [13] but only when the system is decomposable into lower-dimensional subsystems. Our spacecraft benchmark does not fall into this class of nonlinear system. Because the LST cannot handle an 11-dimensional system (which is the size of the satellite state), and the satellite benchmark cannot be exactly decomposed, the constituent parts must be approximately decomposed and verified independently. The LST also can be used to analyze hybrid dynamics, but requires a significant amount of knowledge and effort by the user – the hybrid dynamics are not handled automatically. In general, the LST does not have a user interface and requires sufficient knowledge by the user. It is an open source MATLAB toolbox, and thus provides flexibility in terms of modifying outputs and plots.

B. Flow*

Flow* is a verification tool for hybrid systems with nonlinear continuous dynamics that checks invariance (or safety) properties by generating forward reachable sets. It uses a finite set of Taylor models to overapproximate the forward reachable set over a finite time horizon. The reachable sets are computed at discrete time steps either fixed by the user or varied to reduce error (smaller step sizes reduce the approximation error). Taylor models approximate a nonlinear function through a Taylor polynomial approximation up to a certain order, plus an interval term to capture the approximation error. Flow* can handle high-dimensional systems when the dynamics are linear or not overly nonlinear. It can also accept time-invariant disturbances and some time-varying ones, although the amount of variation must be small. It outputs a plot of the forward reachable sets initialized from a user-provided set of states. Flow* also will return a yes/no answer to the invariance question of whether the system will avoid a user-specified unsafe set. Flow* takes as input a user-generated model description in the form of a text file with a Flow*-specific syntax. It only runs on Linux operating systems.

C. CORA

CORA (COntinuous Reachability Analyzer) is a MATLAB toolbox that generates forward reachable sets for hybrid systems with linear dynamics. Linear dynamical systems have unique properties that make it easier to propagate certain types of sets through their dynamics, such as polytopes and zonotopes. CORA has a number of features for manipulating different types of geometric shapes, but primarily uses zonotopes to represent an initial set of states, and then propagates that zonotope through a system's linear dynamics. The zonotope representation gets progressively more complicated as it is propagated and so simplification techniques are applied. CORA has a built-in feature to handle nonlinear systems by linearizing the dynamics at every time step, using the center of the next reachable set as the local linearization point. It takes as input the dynamical system's differential equation, expressed as a MATLAB function. Because it is a MATLAB toolbox, it is platform-independent, and also is flexible in the type of visualization it produces, although has examples and built-in functions for plotting the zonotopes that represent the reachable set. It is designed to guarantee an over-approximation of the true forward reachable set. It also allows for disturbance inputs.

D. C2E2

C2E2 (Compare Execute Check Engine) is a verification tool for hybrid systems with nonlinear continuous dynamics that produces forward reachable sets and checks safety/invariance properties. It produces reachable sets in a distinctly different way from either Flow* or the LST, by first simulating a set of trajectories, and then producing boundaries around each of those simulated trajectories within which any other trajectory initialized in a region around that trajectory is guaranteed to remain within. It does so by generating "annotated" functions, which are Lyapunov-like functions that relate the distance between two trajectories at any given time as a function of the distance between their initial conditions and time. The concept is equivalent to incremental stability, although does not explicitly require the dynamics to be contracting or incrementally stable – although this does greatly improve the accuracy of the reachable sets. C2E2 is a standalone tool with a user interface that takes as input an XML file that contains the states, dynamics, and switching conditions of the system. It only functions on Linux operating systems. The user also inputs an unsafe set that the system must avoid, and C2E2 then generates the forward reachable sets, which are visualized through plots, and outputs a yes/no answer of whether the system indeed avoids that unsafe set. C2E2 does not allow disturbance inputs.

E. dReach

dReach is a verification tool for hybrid systems with nonlinear continuous dynamics that checks invariance (or safety) properties, rather than producing reachable sets. It will provide a certification that a safety property is always satisfied for a given set of initial conditions, or a counterexample if it does not always hold. It does so by encoding the continuous and discrete dynamics as a first order logic over real numbers and then providing verification using an SMT solver called dReal. dReach computes satisfiability with a delta-complete decision procedure. Delta is a precision value that relaxes an accuracy parameter in the SMT solver in order to make the problem more tractable. Using a relaxation parameter means that the unsafe behavior isn't certain but that it is probable; the result uncovers a potential robustness issue. However, if the system does not reach an unsafe region under the delta relaxation, then it is indeed safe. If dReach shows that the system does reach the unsafe state, then it is actually a delta-bounded over-approximation of the system that reaches the unsafe state. dReach requires the nonlinear hybrid system to be written in a text file using a syntax specific to dReach and works on Mac and Linux operating systems only. This file is then processed, and a certification of safety is returned (in the form of a yes/no output). In the case of a "no" answer, plots are produced that visualize a counterexample to the invariance property.

F. CoCoSim

CoCoSim is currently under development at NASA. It operates directly over a Simulink/Stateflow model, maps that model to a Lustre file, and then calls an external Satisfiability Modulo Theory (SMT) solver that accepts Lustre files as input, such as Kind2 [14]. SMT solvers are not restricted to linear behaviors but are unpredictable in their ability to handle nonlinearities. Further, they can only check invariance-type properties that must always hold. CoCoSim is therefore useful for checking the correctness of mode logic, such as ensuring only one mode is only ever active at a time, and also ensuring states never reach certain values. The outputs of CoCoSim are the Lustre file derived from the Simulink model and the results of the SMT solver, which, like dReach, are a yes/no certification that a property is satisfied, and a counterexample and plot in the case of a "no" answer. CoCoSim is a MATLAB plug-in, and will work on either Windows, Mac, or Linux operating systems, but the SMT solver it can call is restricted depending on which operating system is used.

G. Comparison

Each tool is capable of handling hybrid dynamics at some level, with varying amounts of effort required on the part of the user to enable the full hybrid analysis. We summarize the different features of each tool in Table 1. A rigorous comparison of each tool's performance on the benchmark presented in Section III is given below.

V. Results: Application to Benchmark

The conclusion from the assessment of each verification tool is that none is sufficiently capable on its own to study the spacecraft benchmark, and hence hybrid nonlinear attitude control in general. Each tool has strengths and weaknesses, and some are better fit to the complexities of the non-linear spacecraft benchmark than others, but none were able to verify any properties of the full hybrid benchmark on their own. The benchmark dynamics and mode switching conditions could not simply be written in the model format used by the tool, and then the "verify" command

Table 1 Summary of verification tools.

Tool	Verification Type	Output	Theoretical Technique	Dynamics	Model Format	Interface
Level Set Toolbox	Reachability analysis	Forward or backward reachable sets	Hamilton-Jacobi PDEs	Nonlinear with disturbances	MATLAB function	Integrates with MATLAB on any OS
Flow*	Reachability analysis	Forward reachable sets and yes/no safety answer	Taylor models	Nonlinear hybrid	Flow* model	Command line on Linux
CORA	Reachability analysis	Forward reachable sets	Zonotopes propagated through linear dynamics	Linear hybrid (nonlinear with linearization)	MATLAB function	Integrates with MATLAB on any OS
C2E2	Reachability analysis	Forward reachable sets and yes/no safety answer	Simulated trajectories with proven bounds	Nonlinear hybrid	XML model	Graphical user interface on Linux
dReach	Invariance analysis	Yes/no safety answer, counterexample when not safe	Nonlinear SMT solver	Nonlinear hybrid	dReach model	Command line interface on Mac/Linux
CoCoSim	Invariance analysis	Yes/no certification, counterexample when not safe	SMT solvers and assume-guarantee reasoning	Linear hybrid (sometimes nonlinear)	Simulink model	Integrates with MATLAB on any OS

issued with successful results.

Instead, a sequence of much simpler requirements were tested on individual modes of the benchmark using each tool as applicable.

A. Requirements for Testing

Requirement 1: For simplified Slew mode reaction wheel dynamics (17), and treating the reaction wheel torque as a bounded time-varying disturbance $\tau^r \in [-1, 1]$, can the system reach Dump (i.e. can the reaction wheels saturate) within 20 seconds starting from $w_i \in [-10, 10]$ for $i = 1, 2, 3, 4$?

$$J^r \dot{w} = \tau^r + K_h M_h w \tag{17}$$

Requirement 2: For the full Slew mode dynamics and initial set of states (18) starting in Slew, can the system avoid the unsafe set $\mathcal{U}_1 = \{(q, \omega, w) : w_2 > 600\}$ for 20 seconds?

$$\begin{aligned} q &= [0.1826, 0.3651, 0.5477, 0.7303]^T \\ \omega &= [0.04, 0.04, 0.04]^T \\ w_i &\in [-10, 10]\,,\ i = 1, 2, 3, 4 \end{aligned} \tag{18}$$

Tool	Full Benchmark	Requirement 1	Requirement 2	Requirement 3	Requirement 4	Requirement 5
Level Set Toolbox	Unable	Safe, 660 sec	Unable	Unable		
Flow*	Unable	Safe, 2.5 sec	Safe, 7620 sec	Safe, 109 sec	Safe, 269 sec	Unknown, 1127 sec
CORA	Unable	N/A	Safe, 21 sec	Safe, 19 sec	Safe, 97 sec	Unable
C2E2	Unable	N/A	Safe, 1920 sec	Safe, 1 sec	Safe, 60 sec	Safe, 120 sec
dReach	Unable	N/A	Safe, 1 sec	N/A		
CoCoSim	Some properties verified, < 1 sec	N/A				

Table 2 Summary of verification tool performance results. The result obtained for each requirement ("Safe" when satisfied) is reported along with time to compute answer. When a tool was not tested on the requirement, it is labeled as "N/A." When a tool was applied but could not return an answer, it is labeled as "Unable."

Requirement 3: For the full Dump mode dynamics and initial set of states (19) starting in Dump, can the system avoid the unsafe set $\mathcal{U}_2 = \{(\omega, w) : \omega_x > 0.6\}$ for 20 seconds?

$$\begin{aligned} \omega_i &\in [0.02, 0.04]\ ,\ i = x, y, z \\ w &= [600, 400, 200, -400]^T \end{aligned} \tag{19}$$

Requirement 4: For the full Dump mode dynamics and initial set of states (19) starting in Dump, can the system avoid the unsafe set $\mathcal{U}_2$ for 60 seconds?

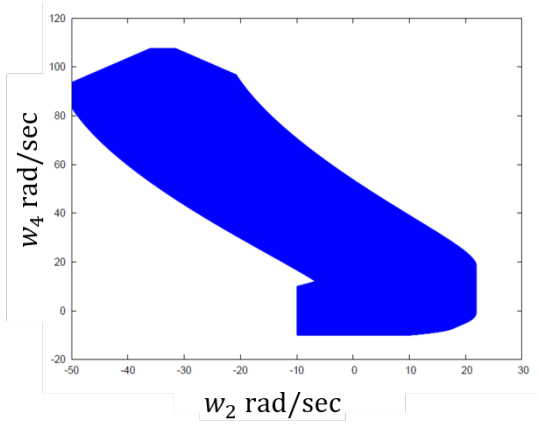
Requirement 5: For the full Dump mode dynamics and initial set of states (19) starting in Dump, can the system avoid the unsafe set $\mathcal{U}_2$ for 100 seconds?

Requirements 1 and 2 seek to determine if Dump mode is reachable from Slew mode. While a transition from Slew to Dump does not exclude the ability of the system to reach Stabilize mode, it does indicate the possibility of getting stuck in a Slew → Dump → Detumble → Slew loop under certain initial conditions. However, the requirements are restricted to whether a single reaction wheel exceeds the upper bound on its angular velocity. To determine whether Dump mode is reachable actually requires checking if any one of eight separate requirements is verified (checking reachability of the upper and lower bounds for all four reaction wheels). Defining the unsafe set as a disjunction of multiple inequalities is challenging if not impossible for the tools under comparison, and so only a single inequality was used to define the unsafe set. Requirement 1 is a further simplification of Requirement 2 in order to test the LST, and is checked using the LST and Flow*. Requirements 3 through 5 are similar but now we are interested in whether the spacecraft's angular velocity gets too high while in Dump. This requirement is independent of the spacecraft's attitude. The verification problem is thus posed over a 7-dimensional system as opposed to an 11-dimensional system for Requirements 2 through 4. Requirements 4 and 5 are included to determine each tool's ability to verify requirements over longer time horizons.

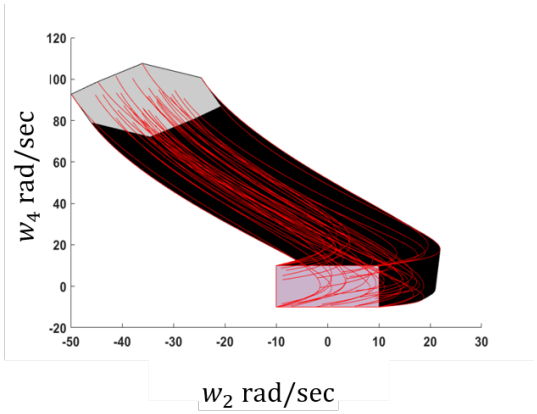
B. Tool Comparison

The results are summarized in Table 2. If a tool was not tested on a particular requirement it is labeled as "N/A," whereas if the requirement was tested but the results were unobtainable because the tool could not complete the analysis for any reason, it is labeled as "Unable." Otherwise, the result of the verification and the computation time to reach that result are reported. If the outcome is "Safe," the tool found that no trajectory starting from within the initial set of states reached the unsafe set. Computation times are not rigorously computed. Flow* and C2E2 were run on a virtual Linux machine while the others were run in Windows. Computation times are therefore for guidance only to give an idea of the differences in magnitude across tools.

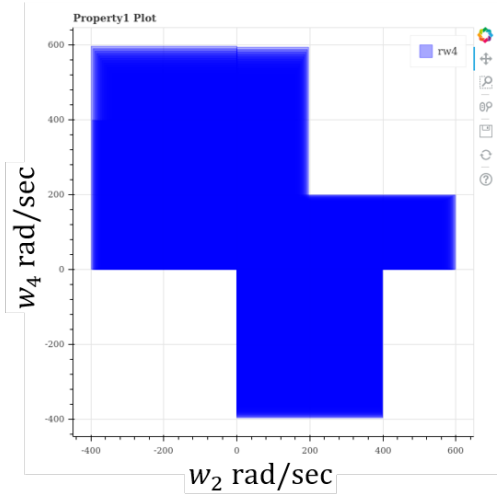
The LST could only verify Requirement 1. For comparison purposes, Flow* was also tested on Requirement 1. Due to the linear nature of the simplified dynamics, Flow* was able to compute the approximate reachable set much faster



(a) Flow*



(b) CORA



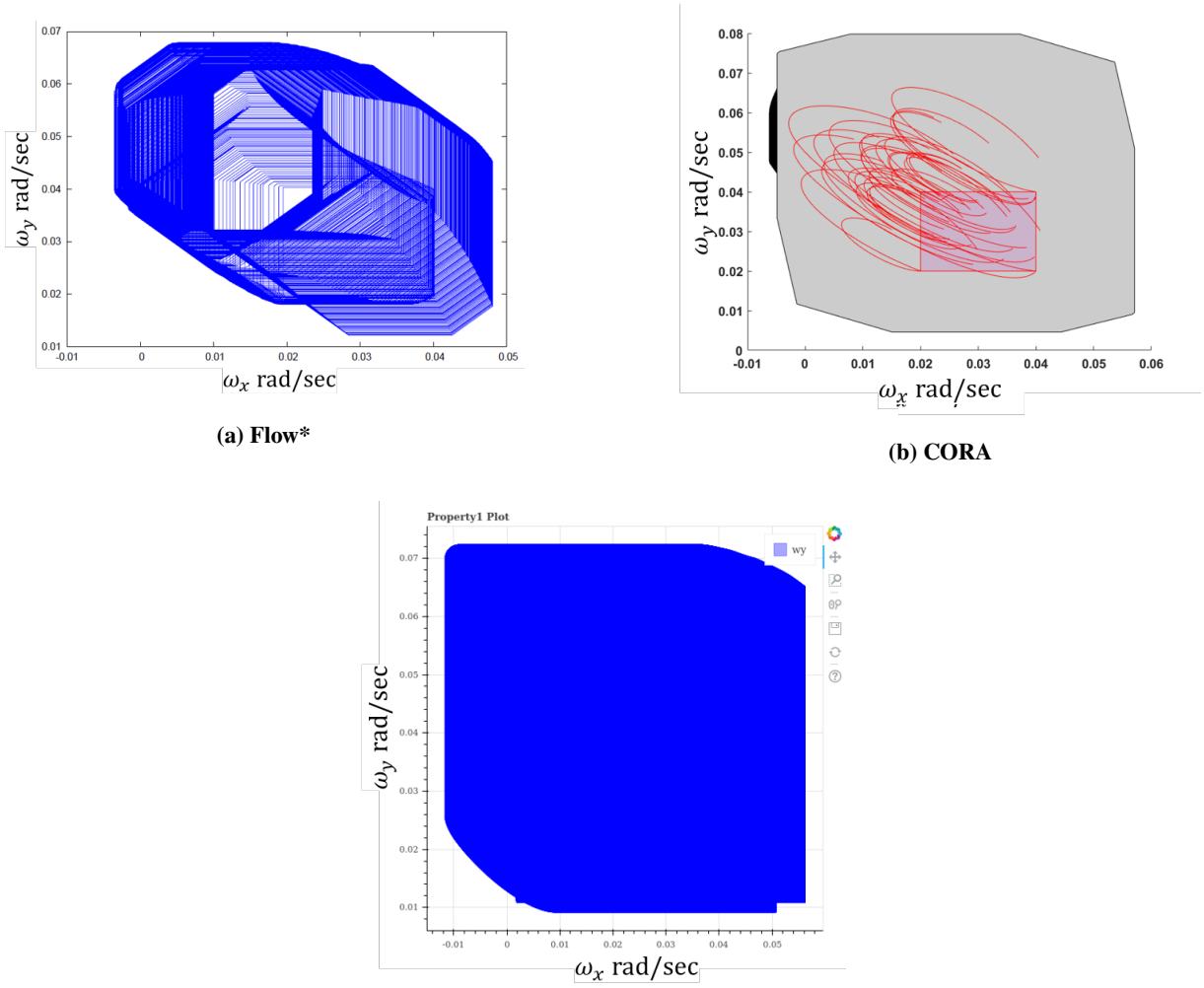
(c) C2E2

Fig. 2 Reachable tubes produced by Flow*, CORA, and C2E2 in Slew mode to check Requirement 2, projected onto (w_2, w_4).

than the LST. Every tool except for CoCoSim and the LST was tested on Requirement 2 for a broad comparison. The tools that compute reachable tubes (Flow*, C2E2, and CORA) were further tested on Requirements 3, 4, and 5 in order to compare their ability to handle longer time horizons.

On Requirement 2, dReach reached the Safe conclusion the fastest, followed by CORA. Flow* and C2E2 were significantly slower. dReach does not compute reachable sets and so we cannot comment on its accuracy other than to say it reached the correct conclusion. Flow* and CORA produced nearly identical reachable tubes, as seen in Figure 2 (a) and (b), respectively, despite Flow* taking over two hours to compute the reachable tube. C2E2 produced a very over-approximate reachable tube compared to both Flow* and CORA, as seen in Figure 2(c). The reachable tubes have a similar shape across Flow*, C2E2, and CORA for Requirements 3, 4, and 5, although are more over-approximate for C2E2 and CORA. Figure 3 compares the reachable tubes produced by each tool for Requirement 4.

CoCoSim was able to operate on the entire benchmark, modeled in Simulink, but because of its underlying SMT solver (the Kind2 model checker was used), it could not verify the same requirements as the other tools. We first analyzed the mode logic, to check that (1) only one mode is enabled at a time, and (2) that each mode switch obeys the transitions we defined (e.g., Detumble is always activated after Dump). We also checked the torques generated in each mode to determine if they always remain bounded, and then used this property to verify plant-level properties, such as the reaction wheels do not saturate under the assumption that the torques are bounded (assume-guarantee reasoning).



(a) Flow*

(b) CORA

(c) C2E2

Fig. 3 Reachable tubes produced by Flow*, CORA, and C2E2 in Dump mode to check Requirement 4, projected onto (ω_x, ω_y).

CoCoSim could verify all mode logic properties except that Slew always transitions to Stabilize under the correct guard conditions. The property of bounded torques was proven valid in the Dump and Detumble modes but was unknown in Slew and Stabilize. Plant-level properties were much harder to prove and CoCoSim could not provide an answer to many. In the case of checking whether the reaction wheels saturate under a bounded torque assumption, the Kind2 model checker returned that this property was valid, although we could construct a counterexample in which the reaction wheels did saturate. A sample of the requirements checked, the assumptions made for each requirement, and the results produced by the Kind2 model checker through CoCoSim are given in Table 3.

None of the above tools, however, could verify properties of interest over the entire system. Specifically, we could not determine whether the system is guaranteed to reach Stabilize within a finite amount of time and then remain in stabilize, or if the there are initial conditions that trigger a repeating loop between Dump, Detumble, and Slew. CoCoSim was the only tool capable of operating on the entire benchmark, but is unable to verify all desired properties. The fastest tools are dReach, CORA, and CoCoSim. dReach is a new tool, and while promising in terms of speed, still has bugs to be worked out and we found it to be somewhat unreliable. CORA is fast but breaks down when the initial set is too large or the time horizon too long. It also tends to seriously overapproximate the reachable set because of the linearization. The most reliable (not too overapproximate) reachable sets are produced (for problems that are solvable) by the LST and Flow*. The LST is the most versatile in terms of dynamics but is limited to low-dimensional systems. Flow* is slow

Requirement	Assumptions	Kind2 Result	Timeout
Only one mode active at a time	$\|q\| = 1$, $\|\omega_i\| < 1$, $\|w_i\| < 600$	Valid	0.271 sec
In Detumble, $\|\tau^r\| < 15$	$\|\omega_i\| < 0.26$ and $\|w_i\| < 30$	Valid	1.159 sec
In Slew, $\|\tau^r\| < 15$	$\|q\| = 1$, $\|\omega_i\| < 0.26$, $\|w_i\| < 600$	Unknown	N/A
$\|w_i\| < 600$	$\|\tau^r\| < 15$, $\|\tau^t\| < 15$	Valid	0.515 sec

Table 3 CoCoSim verification results, using the model checker Kind2.

for nonlinear systems and depending on the dynamics and initial set often could not compute the reachable set to the desired accuracy. C2E2 could quickly verify requirements for the 7-dimensional system needed for Requirements 3, 4, and 5, although the reachable tube is overapproximate, but slowed down significantly for the 11-dimensional model needed for Requirement 2. The limitations of each tool suggest that some type of abstraction is required to simplify the dynamics in order for reachability methods to be useful. This would allow the replacement of complicated nonlinear terms with bounded disturbances, which, when treated in the worst-case, still will guarantee overapproximated reachable sets. However, only CORA and the LST allow for these disturbances when generating forward reachable sets.

C. Falsification as a Successful Alternative

Ultimately, we used a falsification approach to successfully analyze the system. Falsification techniques use search and optimization methods to find counterexample trajectories that violate the requirement of interest. The advantage to such an approach is that it can operate over more complex systems. The disadvantage is that if no counterexample is found, there is typically no guarantee that one does not exist, since nonlinear optimization is used and we cannot guarantee that a global optima has been found. The falsification approach we took optimized the reaction wheel velocities subject to initializing in Dump with fixed initial attitude and variable initial angular velocity and reaction wheel velocities. We wrote our own falsification algorithm and also used S-Taliro [11] to find a trajectory in which the reaction wheels saturate. Our tailored falsification algorithm found a violation right away. S-Taliro was able to find a violation as well, although took several restarts of the solver before a violation was found.

In both cases, it was made apparent that there are initial conditions that lead to exactly the type of cycling between Dump, Detumble, and Slew that we hoped to avoid. This behavior is shown in Figure 4. The identification of these violations ultimately led us to realize that our benchmark has a flaw in the Dump mode. Total angular momentum is not reduced in Dump, and so there is the potential to continually saturate the reaction wheels if the initial total momentum is sufficiently high.

Therefore, while all the verification tools we presented above have the advantage of formally proving that a violation will occur if it exists, they ultimately were not useful for proving the property of interest because they could not handle our hybrid, nonlinear spacecraft benchmark. Falsification, on the other hand, does not have the same guarantees as reachability methods but is much more amenable to complex systems. In the end, the falsification approach enabled us to find an error in our benchmark design.

VI. Conclusion

We have presented a spacecraft benchmark comprising four modes of operation with autonomous switching between modes. Within each mode, the 11-dimensional state of the system evolves according to nonlinear differential equations. Further, the attitude of the spacecraft is described using a quaternion which is restricted to the unit ball manifold in four dimensions. The purpose of the benchmark is to provide a test case for formal verification techniques on a more realistic autonomous system than is often used for tool comparison and demonstration purposes. To that end, we tested six tools that we believe are representative of the types of techniques available for formal verification. The Level Set Toolbox, Flow*, CORA, and C2E2 each compute reachable sets which can then be queried for intersection with an unsafe set in order to check safety requirements. Each computes the reachable sets, however, using a different approximation technique. dReach and CoCoSim rely on different underlying SMT solvers to prove requirement satisfaction or return a counterexample. Ultimately, none of these tools was able to determine stability of the entire benchmark, posed as checking whether the system reaches Stabilize mode within a certain time horizon without cycling between the other modes repeatedly. Alternately, falsification-based verification, which does not provide the same guarantees as the above methods, was used and returned counterexamples which demonstrated that the system can indeed incur

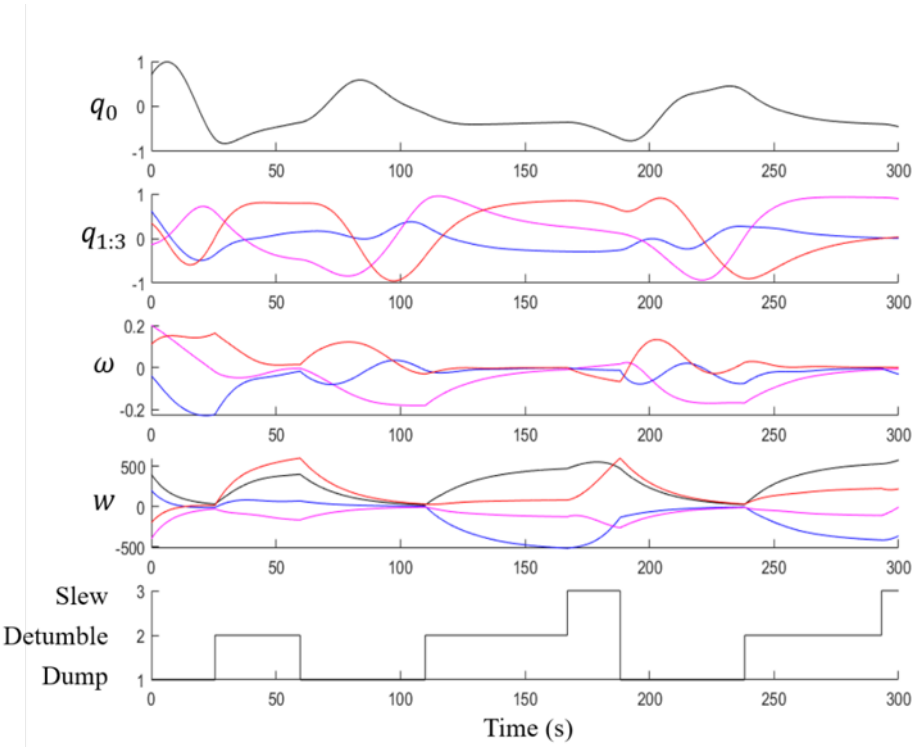


Fig. 4 For certain initial conditions found using falsification techniques, the spacecraft cycles between Dump, Detumble, and Slew without converging to Stablize.

instabilities that lead to repeated cycling between Dump, Detumble, and Slew without reaching Stabilize. These results strongly suggest that many existing formal verification tools, despite continued research and progress, are not capable of analyzing "real-world" systems, at least not without significant model simplifications and decomposition. Statistical and falsification methods, on the other hand, provide useful results despite lacking some of the rigor and formality of the other methods.

Acknowledgements

The authors would like to thank Hamza Bourbouh, Guillaume Brat, and Pierre-Loic Garoche for their support and help with NASA's CoCoSim tool. Hamza Bourbouh conducted the CoCoSim analysis on the spacecraft benchmark and was invaluable for producing the analysis presented here.

References

[1] Goran, F., Le Guernic, C., Donzé, A., Cotton, S., Ray, R., Lebeltel, O., Ripado, R., Girard, A., Dang, T., and Maler, O., "SpaceEx: Scalable verification of hybrid systems," *Computer Aided Verification*, 2011.

[2] Althoff, M., "An introduction to CORA 2015," *Workshop on Applied Verification for Continuous and Hybrid Systems*, 2015.

[3] Chen, X., Abraham, E., and Sankaranarayanan, S., "FLOW*: An analyzer for non-linear hybrid systems," *Computer Aided Verification*, 2013, pp. 258 – 263.

[4] Fan, C., Bolun, Q., Mitra, S., Viswanathan, M., and Duggirala, P. S., "Automatic reachability analysis for nonlinear hybrid models with C2E2," *Computer Aided Verification*, 2016, pp. 531 – 538.

[5] Mitchell, I., and Templeton, J., "A toolbox of Hamilton-Jacobi solvers for analysis of nondeterministic continuous and hybrid systems," *Hybrid Systems: Computation and Control*, 2005, pp. 480–494.

[6] Kong, S., Gao, S., Chen, W., and Clarke, E., "dReach: δ-reachability analysis for hybrid systems," *Tools and Algorithms for the Construction and Analysis of Systems*, 2015, pp. 200 – 205.

[7] NASA, "CoCoSim: Automated Analysis and Simulation framework for Simulink/Stateflow," , 2017. URL `https://coco-team.github.io/cocosim/`.

[8] Chem, X., Schupp, S., Makhlouf, I. B., Ábrahám, E., Frehse, G., and Kowalewski, S., "A benchmark suite for hybrid systems," *NASA Formal Methods Symposium*, 2015.

[9] Fehnker, A., and Ivančić, "Benchmarks for hybrid system verification," *Hybrid Systems: Computation and Control*, 2004, pp. 326 – 341.

[10] Jin, X., Deshmukh, J. V., Kapinski, J., Ueda, K., and Butts, K., "powertrain control verification benchmark," *Hybrid Systems: Computation and Control*, 2014, pp. 253 – 262.

[11] Annpureddy, Y., Liu, C., Fainekos, G., and Sankaranarayanan, S., "S-TaLiRo: A tool for temporal logic falsification for hybrid systems," *Tools and Algorithms for the Construction and Analysis of Systems*, 2011, pp. 254 – 257.

[12] Frehse, G., and Althoff, M. (eds.), *ARCH19: 6th International Workshop on Applied Verification of Continuous and Hybrid Systems*, 2019.

[13] Chen, M., Herbert, S. L., Vashishtha, M. S., Bansal, S., and Tomlin, C. J., "Decomposition of reachable sets and tubes for a class of nonlinear systems," *IEEE Transactions on Automatic Control*, Vol. 63, No. 11, 2018, pp. 3675 – 3688.

[14] Champion, A., Mebsout, A., Sticksel, C., and Tinelli, C., "The Kind2 model checker," *Computer Aided Verification*, 2016, pp. 510–517.