Towards Counterexample-Guided Computation of Validated Stability Certificates for Hybrid Systems *

Eike Möhlmann and Oliver Theel
Carl von Ossietzky University of Oldenburg
Department of Computer Science
D-26111 Oldenburg, Germany
{*eike.moehlmann,theel*}*@informatik.uni-oldenburg.de*

Abstract

We propose a method to obtain trustable Lyapunov-based certificates of stability for hybrid systems. A hybrid system is a system exhibiting discrete-time as well as continuous-time behaviors, e.g. embedded systems within a physical environment. Stability is a property which ensures that a system starting in any possible state will reach a desired state and remain there. Such systems are particularly useful when a certain autonomous operation is required, e.g. keeping a certain temperature or speed of a chemical reaction or steering a vehicle over a predefined track. Stable hybrid systems are extremely valuable because if an error disturbes their normal operation, they automatically "steer back" to normal operation. Stability can be certified by finding a so-called Lyapunov function. The search for this kind of functions usually involves solving systems of constraints. The state-of-the-art in Lyapunov-based stability verification is to use numerical methods to solve systems of inequalities, which if solvable indicate stability. We propose to use Satisfiability-Modulo-Theory (SMT) methods to (a) validate the results of a numerical solver and (b) use counterexamples to guide the numercial solver towards a valid solution.

Keywords

Hybrid Systems; Automatic Verification; Stability; Lyapunov Theory; Optimization; Satisfiability-Modulo-Theory

1. Introduction

In this paper we present experiments on combining Satisfiability-Modulo-Theory (SMT) methods with – the current state-of-the-art – numerical solvers, for the search of Lyapunov functions. A *Lyapunov function* can be used to certify that a given hybrid system is indeed (asymptotically) stable. It can be seen as a "generalized metric" whose value indicates for a given system state the "distance" to the desired system state; the so-called *equilibrium point*. The equilibrium point is w.l.o.g. assumed to be the origin, i.e. 0, of the state space. Lyapunov functions have a special shape that guarantees that while the system evolves the "distance" of the system state to the equilibrium point decreases. Thus, any system for which a Lyapunov function can be found, eventually reaches the equilibrium point. For some systems, it is even possible to give a bound on the rate in which the system approaches the origin. In that case, one can compute an upper bound on the time required to reach a certain region around the equilibrium point. This is especially useful if one is interested in *region stability* [1].

The search for Lyapunov functions involves generating and solving sets of *conditioned constraints*, where each constraints has the form $\forall x \in P : 0 \preceq f(\rho, x)$. Here, "$0 \preceq f(\rho, x)$" is a linear constraint on polynomials involving some free parameters ρ, where "$\preceq$" is the positive semi-definiteness operator, and P is the region in which the constraint has to be satisfied. This is called Semidefinite Programming (SDP) [2].

Conditioned constraints cannot directly be given to an SDP solver (such as CSDP [3] or SDPA [4]) because such a solver expects unconditioned constraints. So, instead, one uses the so-called *S-Procedure* which constructs unconditioned constraints. These unconditioned constraints can then be handed over to an SDP solver.

The typically used SDP solver uses some kind of interior point methods and thus numerically approximates the solution. In practice, it is a problem that such a numerical solvers sometimes suffers from numerical inaccuracies. Therefore, the constraints may be *strengthened* by adding additional "gaps" to the inequalities. This makes them more robust against these numerical issues. However, such gaps can introduce contradictions. In [5], a heuristic is proposed, which counter-acts this problem. The heuristic analyzes the conditioned constraints before the unconditioned constraints are generated, and it removes implicit equalities by substituting free parameters in the other constraints.

While such a strengthening indeed helps to obtain solutions, one still needs to validate the result. In our tool, STABHYLI [6], we already use the computation of eigenvalues and principle minors to further ratify the validity

This work has been partly supported by the German Research Foundation (DFG) as part of the Transregional Collaborative Research Center "Automatic Verification and Analysis of Complex Systems" (SFB/TR 14 AVACS).

of a solution. However, the computation of eigenvalues is usually done via numerical algorithms which again may suffer from numerical issues.

Therefore we propose the use of SMT solvers to validate candidate solutions. An SMT solver combines SAT solvers with theory solver to check the satisfiability (SAT) of first-order logic (FOL) formulae wrt. first-order theories. One instance of such a theory is nonlinear real arithmetic (NRA). A formula is satisfiable if there exists a model – a valuation for each free variable – such that using the model, the formula evaluates to $true$. In general FOL is semidecidable, i.e., there exists a procedure such that, given a model and a formula, the procedure eventually decides whether the model satisfies the formula. Due to Tarski [7], we know that NRA on the other hand is decidable with double exponential complexity.

Related Work

SMT solving has received quite some attention over the last years but, unfortunately, most available implementations are either restricted to linear arithmetic or do not support quantifiers. There is only a handful of SMT solvers which support NRA with quantifiers: Z3 [8], CVC3 [9], and CVC4 [10]. As stated above, SMT is already combining SAT solving with different theory solvers. Nevertheless, other combinations built on top of SMT solvers have been made. One combines a linear SMT solver with interval constraint propagation to solve nonlinear real arithmetic problems [11]. Another one combines convex programming with SMT solving to solve non-linear convex constraints [12]. Our approach and the mentioned methods have in common that we want to find a model for a formula. Gao et al [11] also use the result of a linear SMT solver to validate the answer of the interval constraints propagation method which is very similar to what we do, but they consider only quantifier-free formulas. In the search for a solution of the constraint systems, however, we have a single quantifier alternation that is an outer existential and an inner universal quantifier. Thus we cannot apply their method due to the lack of the quantifiers.

In contrast to safety properties, stability has not yet received that much attention wrt. automatic proving and therefor only a few tools are available. The tools known to the authors are the following:

- A tool by Podelski and Wagner which computes a sequence of snapshots and then tries to related the snapshots in decreasing sequence. If successful this certifies region stability, i.e., stability with respect to a region instead of a single equilibrium point [1].
- A tool by Oehlerking et al. implementing a powerful state space partitioning scheme to find Lyapunov functions for linear hybrid systems [13].
- RSOLVER [14] which computes Lyapunov-like functions for continuous system.
- A tool by Duggirala and Mitra that combines Lyapunov functions with searching for a well-foundedness relation for symmetric linear hybrid systems [15].

Finally various MATLAB toolboxes (YALMIP [16], SOSTOOLS [17]) that require a by-hand generation of constraint systems for the search of Lyapunov functions are available. These toolboxes do not automatic prove stability but assist in handling solvers. One such toolbox around SDP solvers is VSDP [18]. It can compute rigorous error bounds by correct handling of the rounding in floating point arithmetic and thereby validating the candidate solution. Nevertheless, it might happen that the error bounds are such that on one hand deciding whether the candidate solution is valid might be impossible. This means that the candidate needs to be rejected. On the other hand the error bounds are again computed numerically. This means it might happen that the bounds are too coarse such that former case results only from the fact of inexactness.

In conclusion, all Lyapunov-based tools use numerical/approximation methods. This means that the obtained candidate solution needs to be validated to ensure correctness.

The contributions of this paper are the following: 1) We present a method to validate stability certificates obtained by numerical approximation algorithms, 2) We sketch a very preliminary idea to guide a numerical solver away from bad candidate solutions.

The remainder of the paper is organized as follows: Section 2 gives the theoretical background and Section 3 describes how candidate solutions can be validated. In Section 4, we sketch the idea to "guide the numerical solver away from invalid solutions". Section 5 gives a simple example which leads to numerical issues in the search for Lyapunov function. These can be detected and fixed due the proposed method. A conclusion is given in Section 6.

2. Preliminaries

In this section, we define the hybrid system model, stability and the techniques required to certify stability of a hybrid system.

Definition 1: A **Hybrid Automaton** is a quintuple

$$\mathcal{H} = (\mathcal{V}, \mathcal{M}, \mathcal{T}, Flow, Inv) \text{ where}$$

- $\mathcal{V}$ is a finite set of *variables* and $\mathcal{S} = \mathbb{R}^{|\mathcal{V}|}$ is the corresponding *continuous state space*,
- $\mathcal{M}$ is a finite set of *modes*,
- $\mathcal{T}$ is a finite set of *transitions* (m_1, G, U, m_2) where
  - $m_1, m_2 \in \mathcal{M}$ are the *source and target mode* of the transition, respectively,
  - $G \subseteq \mathcal{S}$ is a *guard* which restricts the valuations of the variables for which this transition can be taken,
  - $U : \mathcal{S} \to \mathcal{S}$ is the *reset function* which might update some valuations of the variables,

- *Flow* : $\mathcal{M} \to [\mathcal{S} \to \mathcal{P}(\mathcal{S})]$ is the *flow function* which assigns a *flow* to every mode. A flow $f \subseteq \mathcal{S} \to \mathcal{P}(\mathcal{S})$ in turn assigns a closed subset of $\mathcal{S}$ to each $x \in \mathcal{S}$, which can be seen as the right hand side of a differential inclusion $\dot{x} \in f(x)$,
- *Inv* : $\mathcal{M} \to \mathcal{S}$ is the *invariant function* which assigns a closed subset of the continuous state space to each mode $m \in \mathcal{M}$, and therefore restricts valuations of the variables for which this mode can be active.

A *trajectory* of $\mathcal{H}$ is an infinite solution in form of a function $x(t)$ over time. Each solution has an associated (possibly infinite) sequence of modes visited by the trajectory. $\diamond$

Intuitively, stability is a property expressing that all trajectories of the system eventually reach an equilibrium point of the sub-state space and stay in that point forever, given the absence of errors. For technical reasons, the equilibrium point is usually assumed to be the origin of the continuous state space, i.e. 0. This is not a restriction, since a system can always be shifted such that the equilibrium is in 0 via a coordinate transformation.

In the following, we refer to $x_{\downarrow\mathcal{V}'} \in \mathbb{R}^{|\mathcal{V}'|}$ as the *sub-vector* of a vector $x \in \mathbb{R}^{\mathcal{V}}$ containing only values of variables in $\mathcal{V}' \subseteq \mathcal{V}$.

Definition 2: **Global Asymptotic Stability with Respect to a Subset of Variables [19].**
Let $\mathcal{H} = (\mathcal{V}, \mathcal{M}, \mathcal{T}, Flow, Inv)$ be a hybrid automaton, and let $\mathcal{V}' \subseteq \mathcal{V}$ be the set of variables that are required to converge to the equilibrium point 0. A continuous-time dynamic system $\mathcal{H}$ is called *globally stable (GS) with respect to* $\mathcal{V}'$ if for all functions $x_{\downarrow\mathcal{V}'}(t)$,

$$\forall \epsilon > 0 : \exists \delta > 0 : \forall t \geq 0 : \\ ||x(0)|| < \delta \Rightarrow ||x_{\downarrow\mathcal{V}'}(t)|| < \epsilon.$$

$\mathcal{H}$ is called *globally attractive (GA) with respect to* $\mathcal{V}'$ if for all functions $x_{\downarrow\mathcal{V}'}(t)$,

$$\lim_{t\to\infty} x_{\downarrow\mathcal{V}'}(t) = 0, \text{ i.e.,} \\ \forall \epsilon > 0 : \exists t_0 \geq 0 : \forall t > t_0 : ||x_{\downarrow\mathcal{V}'}(t)|| < \epsilon,$$

where 0 is the origin of $\mathbb{R}^{|\mathcal{V}'|}$. If a system is both, globally stable with respect to $\mathcal{V}'$ and globally attractive with respect to $\mathcal{V}'$, then it is called *globally asymptotically stable (GAS) with respect to* $\mathcal{V}'$. $\diamond$

Intuitively, GS is a boundedness condition, i.e. each trajectory starting δ-close to the origin will remain ϵ-close to the origin. GA ensures progress, i.e. for each ϵ-distance to the origin, there exists a point in time t_0 such that a trajectory always remains within this distance. By induction, it follows that every trajectory eventually approaches the origin. For a given hybrid system, this can be proven using Lyapunov Theory [20], which was originally restricted to continuous systems but has been lifted to hybrid systems.

We are handling polynomial hybrid systems. Thus, we assume that all guards, invariants, and flows are defined as expressions over polynomials. A *monomial* has the form $\prod_{v\in\mathcal{V}} v^{e_{v,j}}$ where all $e_{v,j} \in \mathbb{N}$. A weighted sum of such monomials is called a *polynomial* $g : \mathbb{R}^{|\mathcal{V}|} \to \mathbb{R}$ and has the form $g(x) = \sum_j c_j \prod_{v\in\mathcal{V}} v^{e_{j,k}}$ where all $c_j \in \mathbb{R}$. Such a polynomial is called a *parameterized polynomial* if it has the form $f(\rho, x) = \sum_j c_j \rho_j \prod_{v\in\mathcal{V}} v^{e_{v,j}}$ where ρ_j is a *free parameter* and $\rho_j \prod_{v\in\mathcal{V}} v^{e_{v,j}}$ is called a *parameterized monomial*. Indeed, the parameter in a parameterized monomial is rather optional, but to increase readability and to shorten the formulas, we assume every summand in a parameterized polynomial to have a parameter – a "dummy" parameter might be used to represent a constant 1.

Theorem 1: **Discontinuous Lyapunov Functions for a Subset of Variables [19].**
Let $\mathcal{H} = (\mathcal{V}, \mathcal{M}, \mathcal{T}, Flow, Inv)$ be a hybrid automaton and let $\mathcal{V}' \subseteq \mathcal{V}$ be the set of variables that are required to converge. If for each $m \in \mathcal{M}$, there exists a set of variables $\mathcal{V}_m$ with $\mathcal{V}' \subseteq \mathcal{V}_m \subseteq \mathcal{V}$ and a continuously differentiable function $V_m : \mathcal{S} \to \mathbb{R}$ such that

1) for each $m \in \mathcal{M}$, there exist two class K^∞ functions α and β such that
$$\forall x \in Inv(m) : \quad \alpha(||x_{\downarrow\mathcal{V}_m}||) \preceq V_m(x) \\ \wedge\, V_m(x) \preceq \beta(||x_{\downarrow\mathcal{V}_m}||),$$
2) for each $m \in \mathcal{M}$, there exists a class K^∞ function γ such that
$$\forall x \in Inv(m) : \dot{V}_m(x) \preceq -\gamma(||x_{\downarrow\mathcal{V}_m}||)$$
for each $\dot{V}_m(x) \in \left\{ \left\langle \frac{dV_m(x)}{dx} \middle| f \right\rangle \mid f \in Flow(m) \right\}$,
3) for each $(m_1, G, U, m_2) \in \mathcal{T}$,
$$\forall x \in G : V_{m_2}(U(x)) \preceq V_{m_1}(x),$$

then $\mathcal{H}$ is globally asymptotically stable with respect to $\mathcal{V}'$ and V_m is called a *Local Lyapunov Function (LLF)* of mode m.

In Theorem 1, "$\left\langle \frac{dV(x)}{dx} \middle| f \right\rangle$" denotes the inner product between the gradient of a Lyapunov function V and a flow function f. Furthermore, each constraint has the form $0 \preceq c(x)$ where "$\preceq$" is a positive semi-definiteness operator which requires that $c(x)$ is non-negative almost everywhere and $c(0) = 0$.

Definition 3: A **Constraint** is called
- *unconditioned* iff it exhibits the form $0 \preceq f(x)$
- *conditioned* iff it exhibits the form $\forall x \in P : 0 \preceq f(x)$

where $f(x)$ is a parameterized polynomial, $P \subset \mathcal{S}$ is a subset of the continuous state space, and $\forall x \in S$ is called the condition.[1] $\diamond$

Using the S-Procedure [2], a conditioned constraint can be transformed into an unconditioned constraint. The S-Procedure restricts a constraint to some region by exploiting

1. We assume the conditions to be expressed as a conjunction of equalities and inequalities over polynomials, i.e. $\bigwedge_i g(x) \triangle 0$ where $g(x)$ is a polynomial and $\triangle$ is a relation with $\triangle \in \{=, \geq, >\}$.

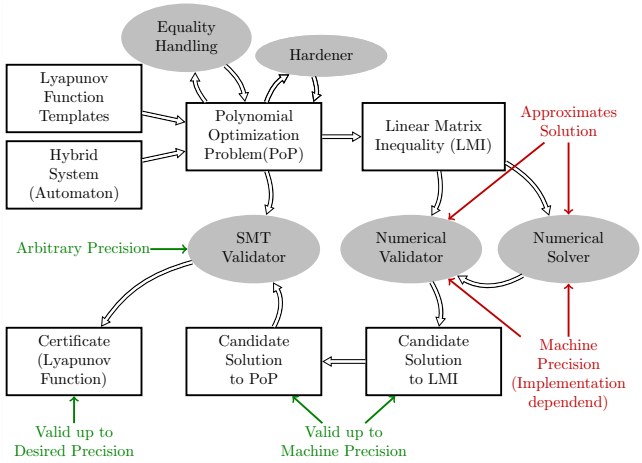


Figure 1: Overview on the Validating Solver Engine

the fact that finding a solution for

$$\left(\sum_i a_i \cdot g_i(x)\right) + \left(\sum_i b_i \cdot h_i(x)\right) \preceq g(x)$$

with $a_i \geq 0$ implies

$$\left(\bigwedge_i 0 \preceq g_i(x)\right) \bigwedge \left(\bigwedge_i 0 = h_i(x)\right) \Rightarrow 0 \preceq g(x).$$

Note that the parameters b_i of the equality conditions $0 = h_i(x)$ are not required to be non-negative.

To find a set of Lyapunov functions for a hybrid system, one also needs to supply the solver with *templates*, which are parameterized polynomials. Such a polynomial usually contains all monomials up to a particular degree that are constructible from a given set of variables. The following steps are performed using the templates: 1. Generate the constraint system described by Theorem 1, then 2. use the S-Procedure to cast the conditioned constraints into unconditioned ones, and then 3. try to solve them.[2] Upon success, we have a certificate of stability. Note that one cannot conclude non-stability from failing to solve the constraints since the described method is sound but incomplete: a higher degree template could still render the problem solvable.

3. Validating Solver Engine

In this section we describe the validating solver engine. This engine can be used to obtain validated stability certificates. Figure 1 gives an overview of the validating solver engine: 1. From the Lyapunov function templates and the hybrid system model, a linear polynomial optimization problem (PoP) is constructed. 2. An equality handler and a constraint hardener post-process the PoP. 3. The PoP is relaxed (via the S-Procedure and the Sums-of-Squares-decomposition) into a linear matrix inequality (LMI). 4. An SDP solver is used to obtain a LMI candidate solution. 5. The LMI candidate solution is validated (e.g., compute eigenvalues[3] and some principal minors). 6. If the LMI candidate solution passes this validation, then 7. it is taken as a candidate solution to the PoP. 8. This PoP candidate solution is then further validated by an SMT solver. If it is successfully validated, then it serves as a stability certificate.

The SDP solver and the numerical validator usually approximate a solution, i.e., they start with an initial valuation of the free parameters. Then they optimize the valuation in the direction which minimizes a given objective function as well as the error.[4] If (a) a certain accuracy is obtained, (b) a maximum number of steps is reached, or (c) the progress becomes too small, then the algorithm stops. Clearly, we can always increase the maximum number of steps and the desired accuracy but the closer the current valuation gets to an optimal solution the smaller the progress becomes. Therefore it is not always possible to reach the optimal solution.

The above described method can be automatized and the resulting LMIs can be solved using numerical solvers. Recall, that – unfortunately – this type of solvers do only approximate solutions and additionally suffer from numerical issues. Prior to applying the S-Procedure and the Sums-of-Squares decomposition, the constraint system has the following form:

$\exists \rho \in \mathbb{R}^m$ such that

$$\bigwedge_i \forall x \left(\bigwedge_j 0 \leq g_{(j,i)}(x)\right) \Rightarrow 0 \preceq f_i(\rho, x)$$

where $\rho = [\rho_1 \quad \dots \quad \rho_m]^T$ are the free parameters (stemming from the templates), $x \in \mathbb{R}^n$ are the system variables (stemming from the hybrid system), $g_{(j,i)}(x) : \mathbb{R}^n \mapsto \mathbb{R}$ are parameter-free polynomials describing a certain region, and $f_i(\rho, x) : \mathbb{R}^m \times \mathbb{R}^n \mapsto \mathbb{R}$ are polynomials involving free parameters. Note that the goal is to find a valuation for ρ and all $f_i(\rho, x)$ are linear in ρ.

Example 1 shows a very simple version of such a constraint system.

Example 1:

$$\exists \rho \in \mathbb{R} : \forall x \in \mathbb{R} : true \Rightarrow 0 \preceq \rho \cdot x^2$$

For this example a numerical solver might return the candidate solution $\rho_{\mathsf{NUM}} = -1 \cdot 10^{-k}$, where k is sufficiently large. On one hand, this is in most cases sufficient and

2. Further steps are required, like translating the polynomial constraints into linear matrix inequalities (LMIs) which, in turn, can be solved using solvers for positive semi-definite problems. This type of translation can be done by using the Sums-of-Squares decomposition [21].

3. Note that computing eigenvalues has issues on its own since it is wellknown to be numerical ill-conditioned [22]

4. Roughly speaking the error determines the quality of the solution: the closer to 0, the better (in the sense of feasibility) the solution is.

allows to conclude feasibility of the above problem. On the other hand, reusing this candidate solution might render any successive calculation invalid. However, STABHYLI composes Lyapunov functions as conic combinations of other Lyapunov functions. This means that in order to guarantee validity of the composed Lyapunov function, we need to assure validity of the combined Lyapunov functions.

Validating Candidate Solutions

We tried to directly solve constraint systems obtained from applying the Lyapunov Theorem, using the SMT solvers Z3 [8], CVC3 [9], and CVC4 [10]. Unfortunately, none of the solvers is able to solve relevant instances – the solvers always returned `unknown`. However, we were able to use SMT solvers to successfully (in-)validate candidate solutions. This use of SMT solvers for validating solutions is beneficial since it can be done with exact arithmetic, i.e., using rationals with full precision. This allows us to conclude that a candidate solution is actually valid and further computation that reuse the candidate solution are also trustworthy.

Remark: If we want solve the constraint systems directly using an SMT solver then we can only use those solvers that support the logic `NRA` (non-linear real arithmetic) since the constraints involve polynomials. In contrast if we want to validate candidate solutions then we can use solvers that support `QF_NRA` (quantifier-free non-linear real arithmetic). Even though both logics are decidable in theory, the first one is computationally intractable in general.

To validate a candidate solution, we substitute the free parameters ρ in the constraint system by the candidate solution ρ_{NUM}. For Example 1, we obtain:

$$\forall x \in \mathbb{R} : \mathit{true} \Rightarrow 0 \preceq -1 \cdot 10^{-k} \cdot x^2. \tag{1}$$

To validate the candidate solution we simply iterate through all constraints and ask whether the negation is satisfiable. For Equation 1, we obtain:

$$\exists x \in \mathbb{R} : \mathit{true} \wedge -1 \cdot 10^{-k} \cdot x^2 < 0. \tag{2}$$

Then, an SMT solver reports one of the following:

case `unknown` We can not assure validity of the candidate solution – this might happen due to insufficient memory or computation time.
case `unsat` We know that the candidate solution is valid.
case `sat` We have to reject the invalid candidate solution.

For Equation 2, an SMT solver might return `sat` with the model $x_{\mathsf{SMT}} = 1$ and, indeed, x_{SMT} serves as a counterexample showing that ρ_{NUM} is an invalid solution.

Remark: In our experiments none of the solvers ever returned `unknown` while validating candidate solutions.

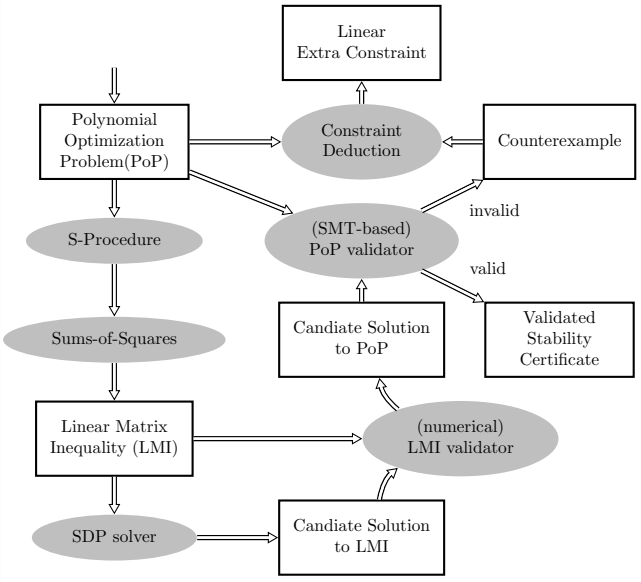


Figure 2: Overview of the Guiding Solver Engine

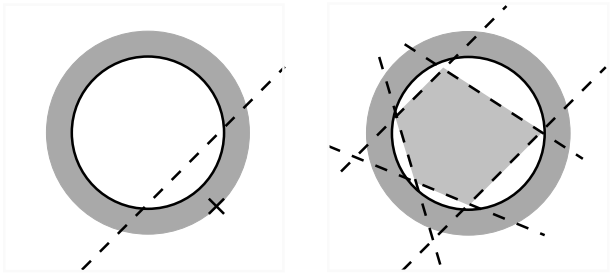
4. Guiding Solver Engine

In the previous section, we have shown how to use SMT solvers to validate solutions. Due to the nature of the numerical solvers (i.e., that they approximate solutions) we can expect many solutions returned by a numerical solver to be invalidated by the SMT solver. It even gets worse because of the way, the constraint systems are presented to the numerical solver. By default, the goal is to minimize $-\alpha$, β, and $-\gamma$ (see Theorem 1). In most cases this leads to a situation where the optimal feasible solution lies on the border of the solution space. That means that the optimization direction increases the attractiveness of invalid candidate solutions attractive.

To overcome this issue, we would like to gain knowledge from a counterexample. We propose to compute the error or residual at the point x_{SMT} and to extend the constraint system with an additional constraint.

Figure 2 gives an overview of the guiding (and validating) solver engine. It extends the proposed engine from Figure 1 by the constraint deduction with possibly exploiting a generated counterexample from the SMT-based validator. The idea is as follows: If the SMT solver returns `sat` and a model x_{SMT}, we construct an additional constraint that rules out the candidate solution ρ_{NUM}. The engine might also check for consistency and then be restarted using the extended constraint system. We call "adding of additional constraints" *guiding*, since it might not force the numerical solver to return valid solutions. Instead it may push the solver away from bad solutions.

Figure 3 sketches the idea. The inner circle represents the solution space while the gray outer ring represents the invalid candidate solutions that might be returned by the



(a) Solution space with a first counterexample

(b) Final solution space

Figure 3: Sketch of the solution space of a constraint system.

solver due to numerical issues and approximation inaccuracies. On the left side an "X" marks a possible candidate solution which is then successfully identified as invalid. The dashed line represents a constraint that is then added to rule out the invalid candidate solution – and hopefully other invalid candidate solutions as well. The right side figure represents a final result in which the original solution space is narrowed down to the most inner gray polygon due to four more constraints, that were added. Any solution that might be returned by the numerical solver that is within the gray polygon is valid. Even more, if again due to numerical issues, the solution lies "a bit outside" of the inner gray polygon but within the white circle, then this is not a problem since such a candidate would still satisfy the original constraint system.

Remark: Note, that the goal is to find solutions to the original constraint system. This fact allows us to exclude the added constraint from validation as they do not need to be satisfied since their sole purpose is to guide the numerical solver.

In the following, we explain how the additional constraints are constructed.

Guiding

By evaluating the all violated constraints $0 \leq f_i(\rho, x)$ using the counterexample x_{SMT} and the candidate solution ρ_{NUM}, we obtain

$$\mathsf{res}_i = f_i(\rho_{\mathsf{NUM}}, x_{\mathsf{SMT}}),$$

where $\mathsf{res}_i < 0$. Now, we can extend the constraint system from Example 1 by a constraint

$$|\mathsf{res}_i| \leq f_i(\rho, x_{\mathsf{SMT}}),$$

which is linear in ρ and does not contain quantifiers. In our running example, this leads to the constraint $|\mathsf{res}| \leq \rho \cdot x_{\mathsf{SMT}}^2$. The extended constraint system is as follows

Example 2:

$$\begin{aligned} \exists \rho \text{ such that } \quad & \forall x : true \Rightarrow 0 \preceq \rho \cdot x^2 \\ \wedge\ & |\mathsf{res}| \leq \rho \cdot x_{\mathsf{SMT}}^2 \end{aligned}$$

This extended constraint system can be handed back to the numerical solver, asking for a new ρ'_{NUM}. This will be again validated using the SMT solver. If we obtain a new counterexample x'_{SMT}, then we extend the problem again until either

- a valid solution is found,
- a maximum number of iterations has been reached, or
- the set of constraints with which we extended the original problem, contains a contradiction.

The last alternative might happen if the real solution space is empty or has a very small interior. Note that the additional constraints are all pure linear, unconditioned, and quantifier free. Thus linear programming (or again an SMT solver) might be used to further check contradiction-freeness, i.e., satisfiability of the additional constraints.

Such contradictions can be introduced because the constraints that we add are more restrictive than the original constraints. The rationale behind was the expectation that the numerical solver will achieve the same residual on the added constraint as on the original constraint. In this way the numerical solver will not return a solution to the extended constraint system. But – fortunately – that candidate satisfies the original constraint system. Thus we would have obtained a validatable solution.

Accelerated Guiding

We can accelerate this method further by alternatively extending the constraint system as follows:

Example 3:

$$\begin{aligned} \exists \rho \text{ such that } & \\ \forall x : true \Rightarrow 0 \preceq \rho \cdot x^2 & \\ \wedge\ \rho \cdot x_{\mathsf{SMT}}^2 \geq |\mathsf{res}| \cdot \mathsf{acc}, & \end{aligned}$$

where $0 < \mathsf{acc}$ is an acceleration factor. Using $1 < \mathsf{acc}$ can lead to a validatable solution more quickly, but the drawback is that at the same time this narrows down the solution space faster. In the worst case, it might happen that the solution space becomes empty due to contradictions before a validatable solution is found. On the other hand, if $0 < \mathsf{acc} < 1$ is chosen, then the solution space is not narrowed down that fast. The drawback is that it might happen that there is nearly no increase in the quality of candidate solution returned by the numerical solver.

5. Example

Consider a sub-model of the hybrid system given in Figure 4 consisting of the two modes **Mode1** and **Mode2**, only.

$$f_i(x) = -c_i \cdot x$$

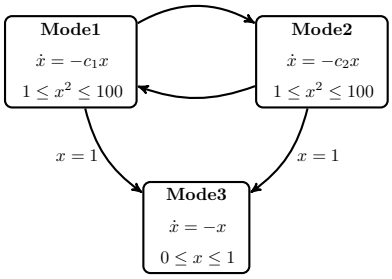


Figure 4: Bad Scaled Hybrid System

where $\mathsf{inv} = 1 \leq x^2 \wedge x^2 \leq 100$ is the invariant of both modes and arbitrary switching is allowed. Choosing the coefficients c_i to be badly scaled, e.g. $c_1 = 10^{-10}$ and $c_2 = 10^{10}$ together with using $V_3 \cdot x^2 + V_2 \cdot x + V_1$ as the template for both modes leads to the constraint system:

$$\begin{aligned}
&\exists \alpha, \beta, \gamma : \\
&\qquad \mathsf{inv} \Rightarrow \alpha \cdot x^2 \preceq V_3 \cdot x^2 + V_2 \cdot x + V_1 \\
&\wedge\ \mathsf{inv} \Rightarrow V_3 \cdot x^2 + V_2 \cdot x + V_1 \preceq \beta \cdot x^2 \\
&\wedge\ \mathsf{inv} \Rightarrow \gamma \cdot x^2 \preceq 2c_1 V_3 \cdot x^2 + c_1 V_2 \cdot x \\
&\wedge\ \mathsf{inv} \Rightarrow \gamma \cdot x^2 \preceq 2c_2 V_3 \cdot x^2 + c_2 V_2 \cdot x \\
&\wedge\ 0 < \alpha \\
&\wedge\ 0 < \beta \\
&\wedge\ 0 < \gamma.
\end{aligned}$$

One might easily see that $V_1 = V_2 = 0$, $V_3 = \alpha = \beta = 1$, $\gamma = 2c_1$ is a valid solution. Nevertheless, both SDPA and CSDP report "Lack of Progress" and thus the quality of the solution is unknown. Indeed, both solution have the problem that γ is slightly too large and CSDP additionally chooses V_3 and β slightly too small.

However, the SMT-base validator finds counterexamples. And after deducing two more constraints in case of SDPA and four more constraints in case of CSDP, both solvers – even though they still report "Lack of Progress" – return a valid solution.

Remark: The numerical validation is not helpful in this example since the eigenvalues are very close to 0. For the initial constraint system, i.e., without any additional constraints, computing the eigenvalues correctly indicates that the candidate solutions are not valid. In case of the final constraint system – the one obtained by guiding – computing eigenvalues falsely indicates that the solutions are invalid. Here using the SMT-based validator allows us to recover the solutions and lets us conclude stability of the hybrid system.

6. Conclusion

We have presented an approach to validate candidate solution obtained by numerically solvers. We have applied this approach in the process of solving constraint systems that arise in search for Lyapunov functions. The validation works by using SMT solvers to check satisfiability of the negated constraints. Such a validation is needed if the Lyapunov functions are reused, e.g. as barrier certificates proving unreachability of certain bad states, as a basis for composition, or as an estimator on the rate of convergence.

While this validation is already very helpful, one would also like to be able to gain knowledge from invalid candidate solutions. In Section 4, we have also shown a way to guide the numerical solver away from invalid candidate solutions. That way, it becomes more likely that "true solutions" are found. Future work includes to further analyze this approach and to give it a more theoretical basis. The relation to δ-satisfiability [23] is of special interest.

References

[1] A. Podelski and S. Wagner, "Region Stability Proofs for Hybrid Systems," in *Formal Modelling and Analysis of Timed Systems (FORMATS'07)*, ser. Lecture Nodes in Computer Science, vol. 4763. Springer, 2007, pp. 320–335.

[2] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, Mar. 2004.

[3] B. Borchers, "CSDP, a C Library for Semidefinite Programming." *Optimization Methods and Software*, vol. 10, pp. 613–623, 1999.

[4] K. Fujisawa, K. Nakata, M. Yamashita, and M. Fukuda, "SDPA Project : Solving Large-Scale Semidefinite Programs," *Journal of the Operations Research Society of Japan*, vol. 50, no. 4, pp. 278–298, Dec. 2007.

[5] E. Möhlmann and O. E. Theel, "Towards Automatic Detection of Implicit Equality Constraints in Stability Verification of Hybrid Systems," in *Proceedings of the 1th Congreso Nacional de Ingeniera Informática / Aplicaciones Informáticas y de Sistemas de Información, CoNaIISI 2013, November 21-22, 2013, Córdoba, Argentina*, M. M. Marciszack, R. M. Muñoz, and M. A. Groppo, Eds. Red de Carreras de Ingeniería Informática / Sistemas de Informatión (RIISIC), 2013.

[6] ——, "Stabhyli: A Tool for Automatic Stability Verification of Non-Linear Hybrid Systems," in *Proceedings of the 16th international conference on Hybrid systems: computation and control (HSCC'13)*, C. Belta and F. Ivancic, Eds. ACM, 2013, pp. 107–112.

[7] A. Tarski, "A Decision Method for Elementary Algebra and Geometry," *Bulletin of the American Mathematical Society*, vol. 59, 1951.

[8] L. De Moura and N. Bjørner, "Z3: An Efficient SMT Solver," in *Proceedings of the Theory and Practice of Software, 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, ser. TACAS'08/ETAPS'08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 337–340.

[9] C. Barrett and C. Tinelli, "CVC3," in *Proceedings of the 19th International Conference on Computer Aided Verification (CAV'07)*, ser. Lecture Notes in Computer Science, W. Damm and H. Hermanns, Eds., vol. 4590. Springer-Verlag, Jul. 2007, pp. 298–302, berlin, Germany.

[10] C. Barrett, C. L. Conway, M. Deters, L. Hadarean, D. Jovanović, T. King, A. Reynolds, and C. Tinelli, "CVC4," in *Proceedings of the 23rd International Conference on Computer Aided Verification (CAV'11)*. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 171–177.

[11] S. Gao, M. Ganai, F. Ivancic, A. Gupta, S. Sankaranarayanan, and E. Clarke, "Integrating ICP and LRA solvers for deciding nonlinear real arithmetic problems," in *Formal Methods in Computer-Aided Design (FMCAD'10), 2010*, Oct 2010, pp. 81–89.

[12] P. Nuzzo, A. A. A. Puggelli, S. A. Seshia, and A. L. Sangiovanni-Vincentelli, "CalCS: SMT Solving for Nonlinear Convex Constraints," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2010-100, Jun 2010. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-100.html

[13] J. Oehlerking, H. Burchardt, and O. E. Theel, "Fully Automated Stability Verification for Piecewise Affine Systems," in *Hybrid Systems: Computation and Control*, ser. Lecture Nodes in Computer Science, vol. 4416. Springer, 2007, pp. 741–745.

[14] S. Ratschan and Z. She, "Providing a Basin of Attraction to a Target Region of Polynomial Systems by Computation of Lyapunov-Like Functions," *SIAM J. Control and Optimization*, vol. 48, no. 7, pp. 4377–4394, 2010.

[15] P. S. Duggirala and S. Mitra, "Lyapunov abstractions for inevitability of hybrid systems," in *Proceedings of the 15th international conference on Hybrid systems: computation and control (HSCC'12)*. ACM, 2012, pp. 115–124.

[16] J. Löfberg, "Yalmip : A toolbox for modeling and optimization in MATLAB," in *Proceedings of the 13th Conference on Computer-Aided Control System Design (CACSD'04)*, Taipei, Taiwan, 2004.

[17] A. Papachristodoulou, J. Anderson, G. Valmorbida, S. Prajna, P. Seiler, and P. A. Parrilo, *SOSTOOLS: Sum of squares optimization toolbox for MATLAB*, `http://arxiv.org/abs/1310.4716`, 2013.

[18] C. Jansson, D. Chaykin, and C. Keil, "Rigorous error bounds for the optimal value in semidefinite programming," *SIAM J. Numerical Analysis*, vol. 46, no. 1, pp. 180–200, 2007.

[19] J. Oehlerking, "Decomposition of Stability Proofs for Hybrid Systems," Ph.D. dissertation, Carl von Ossietzky University of Oldenburg, Department of Computer Science, Oldenburg, Germany, 2011.

[20] M. Lyapunov, "Problème général de la stabilité du movement," in *Ann. Fac. Sci. Toulouse, 9.* Université Paul Sabatier, 1907, pp. 203–474.

[21] S. Prajna and A. Papachristodoulou, "Analysis of switched and hybrid systems - beyond piecewise quadratic methods," in *American Control Conference, 2003. Proceedings of the 2003*, vol. 4, June 2003, pp. 2779–2784 vol.4.

[22] J. Wilkinson, *Rounding Errors in Algebraic Processes*. Dover Publications, Incorporated, 1994.

[23] S. Gao, J. Avigad, and E. M. Clarke, "δ-Complete Decision Procedures for Satisfiability over the Reals," in *Proceedings of the 6th International Joint Conference on Automated Reasoning (IJCAR'12)*, ser. Lecture Notes in Computer Science, B. Gramlich, D. Miller, and U. Sattler, Eds., vol. 7364. Springer, 2012, pp. 286–300.