Model Checking of Hybrid Systems: From Reachability towards Stability

Andreas Podelski and Silke Wagner

Max-Planck-Institut für Informatik, Saarbrücken, Germany [⋆]

Abstract. We call a hybrid system *stable* if every trajectory inevitably ends up in a given region. Our notion of stability deviates from classical definitions in control theory. In this paper, we present a model checking algorithm for stability in the new sense. The idea of the algorithm is to reduce the stability proof for the whole system to a set of (smaller) proofs for several one-mode systems.

1 Introduction

Consider a heating system for a plant that consists of a heater and an internal engine. The internal engine may overheat and switch off the heater temporarily, even though the desired temperature is not yet reached. This means that, starting from low, the temperature will not increase strictly monotonically but it will also decrease during some (relatively short) periods of time. We do not know when exactly such periods start and how long they will be. A sample trajectory of the system is shown in Fig. 1.

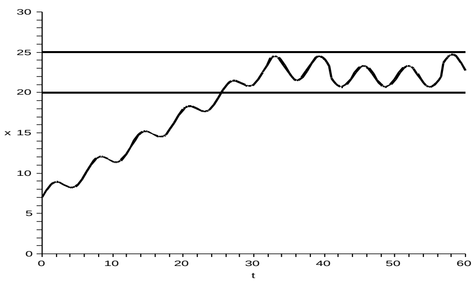


Fig. 1. Sample trajectory of the heating system.

When does such a system behaves correctly? Informally, we expect that the heating system will bring the temperature of the plant to a range between 20 and 25 degrees and then keep it there, whatever the initial temperatures of the plant and the heater are and whatever the exact time points are when the controller switches the heater from “on” to “off” and back. In this paper we introduce a new notion of stability that allows one to formalize the corresponding notion of correctness. We will then give an algorithm

[⋆] This work was partly supported by the German Research Council (DFG) as part of the Transregional Collaborative Research Center “Automatic Verification and Analysis of Complex Systems” (SFB/TR 14 AVACS). See `www.avacs.org` for more information.

to verify that a hybrid system is stable in our sense. The algorithm is parameterized by the constraint solver that it calls as a subroutine in each of its different steps. Using a constraint solver for linear constraints, we obtain a specific algorithm for linear hybrid systems.

The contribution of this paper is threefold. First, our notion of stability fills the need to specify correctness properties of hybrid systems that cannot be formalized by "classical" notions like asymptotic or exponential stability [2, 17, 23]. An example of such a system is the above mentioned heating system. Another example may be an aircraft that oscillates around an optimal course within a certain allowance. Any realistic model of such a system does not satisfy a stability property in the classical sense yet the performance of the aircraft is acceptable.

Second, we use existing reachability tools to reduce the stability proof for the whole system to a set of proofs for one-mode systems [12, 11, 29]. We also show (and this is the third part of our contribution) how one can carry over techniques that are used in program analysis for termination proofs to stability proofs for one-mode hybrid systems [25, 26, 9, 4].

2 Related Work

In this section, we discuss the relation between our and classical notions of stability in control theory, and the relation between our algorithm and verification methods in control theory and model checking.

Stability is a central themes in control theory. There are many different variations of this property, such as asymptotical stability or exponential stability [2, 3, 17, 19, 21]. These classical notions of stability refer to a single equilibrium point. As we have pointed out in the introduction, stability with respect to one point does not seem to be always adequate. In the example of the heating system, where the temperature is specified by upper and lower bounds, such an equilibrium point does not even exist.

The example of the heating system shows that it is not always possible to express stability with respect to a region in terms of e.g. asymptotical stability. The other way round, asymptotical stability with respect to a point x_0 is expressible as stability with respect to *every* region $(x_0 - \varepsilon, x_0 + \varepsilon)$, for $\varepsilon > 0$. However, it does not seem clear how to compare these notions of stability. In particular we don't see how one could use existing techniques for proving classical stability (e.g. [2, 3, 18–20]) to prove that a hybrid system is stable with respect to a given region.

Verification methods for non-reachability properties (or properties that can be reduced to non-reachability) for hybrid systems have been intensively studied by both computer scientists and control theorists [32, 10, 34, 8, 27] and have lead to popular verification systems such as PHAVer [12], HSolver [29], d/dt [11] and CheckMate [7]. Stability properties (in the classical as well as in our sense) are fundamentally different from (non-)reachability. The methods used in reachability analysis are inherently not applicable to stability. This means it is not possible to check stability with existing tools for reachability.

The open problem that this paper attacks is the question whether model checking for our new definition of stability is possible. Our results together with preliminary experiments in a prototypical implementation implicate that this is possible in principle.

3 Preliminaries: Hybrid Systems and Trajectories

In this section, we rephrase the classical definitions of the syntax and semantics of hybrid systems [1, 13, 14].

A **hybrid system** is a tuple (fixed from now on)

$$A = (\mathcal{L}, \mathcal{V}, (jump_{\ell,\ell'})_{\ell,\ell' \in \mathcal{L}}, (flow_\ell)_{\ell \in \mathcal{L}}, (inv_\ell)_{\ell \in \mathcal{L}}, (init_\ell)_{\ell \in \mathcal{L}})$$

consisting of the following components:

1. a finite set $\mathcal{L}$ of locations.
2. a finite set $\mathcal{V}$ of real-valued variables, including a variable t that denotes the time.
3. a family $(jump_{\ell,\ell'})_{\ell,\ell' \in \mathcal{L}}$ of formulas over $\mathcal{V}$ representing the possible jumps from location ℓ to location ℓ'.
4. a family $(flow_\ell)_{\ell \in \mathcal{L}}$ of formulas over $\mathcal{V}$ and $\dot{\mathcal{V}}$ specifying the continuous variable update in location ℓ. We use $\dot{\mathcal{V}} = \{\dot{x}_1, \dot{x}_2, \ldots\}$ for the set of dotted variables. A variable $\dot{x}$ represents the first derivative of x with respect to time, i.e. $\dot{x} = dx/dt$. Especially the derivative of time t with respect to itself is always equal to 1, $\dot{t} = 1$.
5. a family $(inv_\ell)_{\ell \in \mathcal{L}}$ of formulas over $\mathcal{V}$ representing the invariant condition in location ℓ.
6. a family $(init_\ell)_{\ell \in \mathcal{L}}$ of formulas over $\mathcal{V}$ representing the initial states of the system.

A **state** s is a pair (ℓ, ν) consisting of a location ℓ of $\mathcal{L}$ and a valuation ν of all variables over the set $\mathcal{V}$. We write $\Sigma_{\mathcal{V}}$ for the set of all variables valuations ν and $\Sigma = \mathcal{L} \times \Sigma_{\mathcal{V}}$ for the set of all states. A set of states is also called a **region**. A valuation over the set $\dot{\mathcal{V}}$ of dotted variables is denoted by $\dot{\nu}$.

Note that a linear flow formula $flow_\ell$ can also be specified over $\mathcal{V}$ and $\mathcal{V}'$ (instead of $\mathcal{V}$ and $\dot{\mathcal{V}}$). A formula $flow_\ell(x_1, \ldots, t, x_1', \ldots, t')$ represents the flow of duration $t' - t$ in location ℓ, where the values of the continuous variables change from $x_1, \ldots, t$ to $x_1', \ldots, t'$.

A **trajectory** τ of a hybrid system A is a function mapping time points t in $\mathbb{R}^+$ to states in Σ such that the following conditions hold:

Let ν be the real-valued component of τ at time point t.

1. If $\tau(0)$ has location ℓ, then $\tau(0)$ must satisfy the initial condition of that location, formally
$$\tau(0) \models init_\ell \,.$$
2. If ν is differentiable at t, and both $\tau(t)$ and the left-limit of τ at t,
$$\lim_{t' \to t_-} \tau(t') \,,$$

have an equal location ℓ, then the pair $(\nu, \dot{\nu})$ of variable valuation and valuation of the first derivatives satisfies the invariant and the flow condition of location ℓ, formally

$$(\nu, \dot{\nu}) \models inv_\ell \wedge flow_\ell \ .$$

3. If the left-limit of τ at t has location ℓ and $\tau(t)$ has a different location ℓ', then the real-valued component of the left-limit of τ at t must satisfy the jump condition from location ℓ to location ℓ',formally The values of the continuous variables remain unchanged during a jump.

$$\lim_{t' \to t_-} \tau(t') \models \ jump_{\ell,\ell'} \ .$$

The values of the continuous variables remain unchanged during a jump.

Example:

We take a simplified model of a temperature controller with an internal engine which we depict in Fig.2.

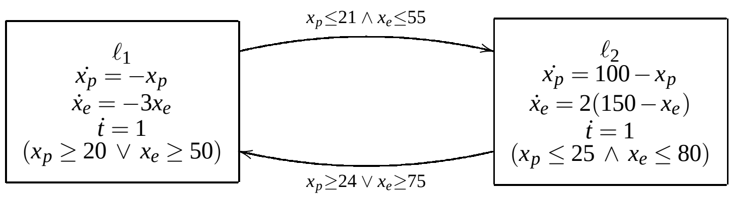

$x_p \leq 21 \wedge x_e \leq 55$
ℓ_1
$\dot{x_p} = -x_p$
$\dot{x_e} = -3x_e$
$\dot{t} = 1$
$(x_p \geq 20 \vee x_e \geq 50)$
ℓ_2
$\dot{x_p} = 100 - x_p$
$\dot{x_e} = 2(150 - x_e)$
$\dot{t} = 1$
$(x_p \leq 25 \wedge x_e \leq 80)$
$x_p \geq 24 \vee x_e \geq 75$

Fig. 2. Temperature controller.

The temperature of a plant is controlled through a thermostat, which continuously senses the temperature and turns a heater on and off. The system has three *variables* x_p, x_e and t,

$$\mathcal{V} = \{x_p, x_e, t\} \ ,$$

where x_p models the temperature of the plant, x_e models the temperature of the internal engine and t models the total elapse of time. The two states "on" and "off" of the heater correspond to the two *locations* ℓ_1 and ℓ_2 of the overall system,

$$\mathcal{L} = \{\ell_1, \ell_2\} \ .$$

The temperature fall resp. rise is governed by differential equations. Namely, in location ℓ_1, where the heater is off, the temperature falls according to the *flow condition* $flow_{\ell_1}$.

$$flow_{\ell_1}(x_p, x_e, t, \dot{x}_p, \dot{x}_e, \dot{t}) \ \equiv \ (\dot{x}_p = -x_p \ \wedge \ \dot{x}_e = -3x_e \ \wedge \ \dot{t} = 1)$$

In location ℓ_2, where the heater is on, the temperature rises as specified by $flow_{\ell_2}$.

$$flow_{\ell_2}(x_p, x_e, t, \dot{x}_p, \dot{x}_e, \dot{t}) \ \equiv \ (\dot{x}_p = 100 - x_p \ \wedge \ \dot{x}_e = 2(150 - x_e) \ \wedge \ \dot{t} = 1)$$

The heater itself has an engine that may overheat. The heater is turned off not only when the plant gets too hot but also when the engine is overheated. We assume that a ventilator aids cooling down the engine; that is it cools down faster than it heats up. The engine is overheated if its temperature exceeds 80 degrees; if it is cooled down to 55 degrees, the heater can again be turned on.

The controller can switch the heater from “off” to “on” and back (which corresponds to switches between the modes for the overall system) according to the *jump conditions* on the edges between the two modes.

$$jump_{\ell_1,\ell_2}(x_p,x_e,t) \equiv (x_p \leq 21 \wedge x_e \leq 55)$$
$$jump_{\ell_2,\ell_1}(x_p,x_e,t) \equiv (x_p \geq 24 \vee x_e \geq 75)$$

The controller *must* switch the heater from “off” to “on” and thus trigger a switch of the locations ℓ_1 and ℓ_2 before the *invariant condition* of the location ℓ_1 is violated (i.e. before the temperature of the plant is below 20 and the temperature of the heater is below 50).

$$inv_{\ell_1} \;\equiv\; (x_p \geq 20 \vee x_e \geq 50)$$

Similarly, the controller must switch from “on” to “off” before the temperature of the plant is above 25 or the temperature of the heater is above 80.

$$inv_{\ell_2} \;\equiv\; (x_p \leq 25 \wedge x_e \leq 80)$$

4 Stability

In this section, we introduce our notion of stability and investigate its expressiveness.

Definition 1 (Stability). *We call a hybrid system* stable *with respect to a given region* φ *if for every trajectory* τ *there exists a point of time* t_0 *such that from then on, the trajectory is always in the region* φ.

$$\forall \tau \, \exists t_0 \, \forall t \geq t_0 \; : \; \tau(t) \in \varphi$$

In the example of the heating system, the correctness property we are interested in is this: whatever the initial temperature of the plant is and whatever the initial temperature of the heater is and whatever the exact time points are when the controller switches the heater from “off” to “on” and back, the temperature of the plant will finally be between 20 and 25 degrees (and it may oscillate between these bounds). We can now formalize this correctness property as the stability wrt. the region $\varphi \;\equiv\; x_p \in [20,25]$.

We can express stability in temporal logic, in LTL or in CTL*. In CTL*, for example, one would say that *a*ll trajectories *f*inally *g*lobally are in the region φ.

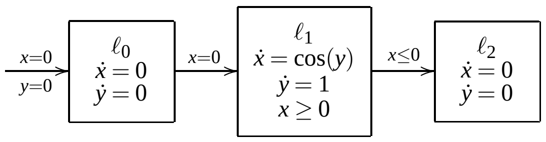
$$A(FG)\,\varphi$$

One might think of verifying stability by applying a CTL* model checker to a finite state abstraction of the given hybrid system. However, there exist no abstraction techniques that would preserve the stability property (except for trivial cases).

The following CTL formula is stronger than, but not implied by stability.

$$AF\ AG\ \varphi$$

The hybrid system below is stable with respect to the region $x = 0$. However, it does not satisfy $AF\ AG\ (x = 0)$; if the system stays in location ℓ_0 forever, it always has the option to switch to location ℓ_1 where it would go outside the region $x = 0$.



One might think of verifying stability wrt. φ by using fixpoint iteration in order to compute the set of states satisfying the formula $\neg EG\ \neg EF\ \neg\varphi$, which is equivalent to $AF\ AG\ \varphi$. The problem here would be to find practical approximation techniques for greatest fixpoint iteration which is needed for the computation of $\neg EG$.

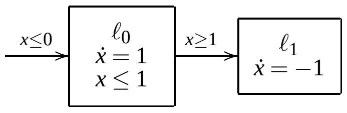
We will now introduce yet another property that is stronger than stability (and stronger than $AF\ AG\ \varphi$). Our algorithm to prove stability is based on this property.

Definition 2 (Strong Attractor). *We call a region* φ *a* strong attractor *of a hybrid system A if every trajectory* τ *of A will (1) finally reach the attractor* φ *and (2) once in* φ *it will never leave the region again.*

$$\exists t_0 \in \mathbb{R}^+ \begin{cases} \forall t < t_0 :\ \tau(t) \notin \varphi \\ \forall t \geq t_0 :\ \tau(t) \in \varphi \end{cases}$$

Our terminology refers to the notion of attractor in the theory of dynamical systems, where the *basin of attraction* is a specified region (and not necessarily the whole state space, as with strong attractors) and where trajectories are required to converge towards the given region φ (and need not finally reach φ).

The region $\varphi \equiv x \leq 0$ is not a strong attractor for the hybrid system below, which, however, satisfies the temporal property $AF\ AG\ \varphi$.



A hybrid system can be stable wrt. a region without having that region as a strong attractor. For example a slightly damped pendulum that oscillates around the origin with initial amplitude $x = 100$ is certainly stable wrt. $x < 1$, but the region $x < 1$ is not a strong attractor of the system. In fact, this system does not have any strong attractor at all.

5 Algorithm

In this section we describe in detail our algorithm.

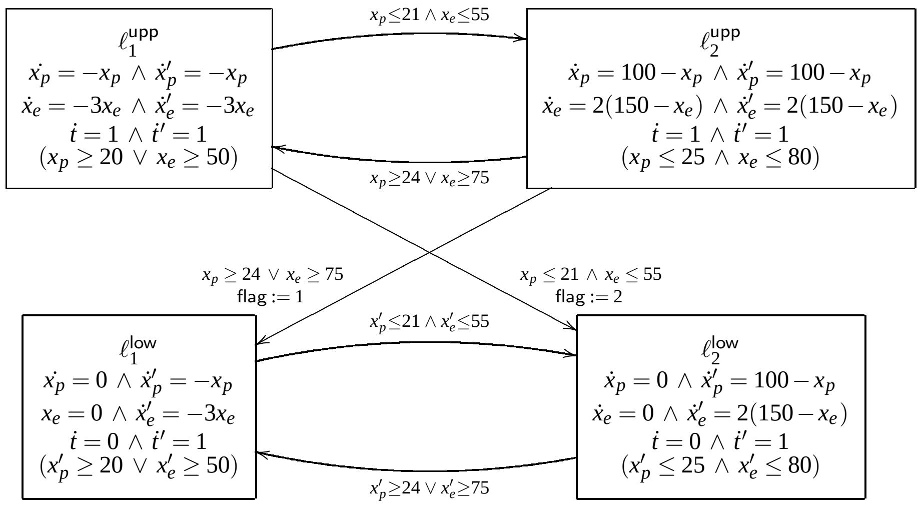
The input of the algorithm is a hybrid system A and a region φ. The output is a "yes/don't know" answer. If the the answer is "yes", the system A is stable wrt. φ. If the algorithm answers "don't know", the system may be stable or unstable.

Again, our algorithm doesn't check directly whether the system A is stable with respect to the region φ, but it checks whether φ is a strong attractor of A with the whole state space as its basin of attraction, which implies stability.

The algorithm proceeds in four steps.

Step 1: Transformation $\mathbf{A} \mapsto \mathbf{A}^{\mathcal{T}}$

The first step of the algorithm is to transform the given hybrid system into a new one. Program transformation has been used recently in program analysis for termination proofs for finite state systems and infinite programs [4, 9]. For the example of the heating system, Fig.3 shows the relevant part of the transformed system.



ℓ_1^{upp}
$\dot{x}_p = -x_p \wedge \dot{x}'_p = -x_p$
$\dot{x}_e = -3x_e \wedge \dot{x}'_e = -3x_e$
$\dot{t} = 1 \wedge \dot{t}' = 1$
$(x_p \geq 20 \vee x_e \geq 50)$
$x_p \leq 21 \wedge x_e \leq 55$
ℓ_2^{upp}
$\dot{x}_p = 100 - x_p \wedge \dot{x}'_p = 100 - x_p$
$\dot{x}_e = 2(150 - x_e) \wedge \dot{x}'_e = 2(150 - x_e)$
$\dot{t} = 1 \wedge \dot{t}' = 1$
$(x_p \leq 25 \wedge x_e \leq 80)$
$x_p \geq 24 \vee x_e \geq 75$
$x_p \geq 24 \vee x_e \geq 75$
flag := 1
$x_p \leq 21 \wedge x_e \leq 55$
flag := 2
$x'_p \leq 21 \wedge x'_e \leq 55$
ℓ_1^{low}
$\dot{x}_p = 0 \wedge \dot{x}'_p = -x_p$
$x_e = 0 \wedge \dot{x}'_e = -3x_e$
$\dot{t} = 0 \wedge \dot{t}' = 1$
$(x'_p \geq 20 \vee x'_e \geq 50)$
ℓ_2^{low}
$\dot{x}_p = 0 \wedge \dot{x}'_p = 100 - x_p$
$\dot{x}_e = 0 \wedge \dot{x}'_e = 2(150 - x_e)$
$\dot{t} = 0 \wedge \dot{t}' = 1$
$(x'_p \leq 25 \wedge x'_e \leq 80)$
$x'_p \geq 24 \vee x'_e \geq 75$

Fig. 3. Transformed system.

We will explain next the characteristics of the transformation. Each state of the new system corresponds to a pair (s, s') of states s and s' of the original system. Whenever the state s' is reachable from the state s in the original system, where s is a state just after a discrete jump, then the state corresponding to the pair (s, s') is reachable in the new system. We refer to this property as *binary reachability*, i.e. a pair of states (s, s') is called binary reachable in a hybrid system A, if there exists a trajectory τ of A such that

1. s is a state on τ at time point t: $s = \tau(t)$;
2. s' is a state on τ at time point t': $s' = \tau(t')$;
3. $t < t'$.

Take the states $s = (\ell_2;\, x_p = 17.1,\, x_e = 50,\, t = 0.157)$ and $s' = (\ell_2;\, x_p = 21.4,\, x_e = 60,\, t = 0.209)$. The state s can reach the state s' (both the temperature of the plant and the engine increase in a time period of 5) in a trajectory of the original system; the trajectory starts in the initial state $s_0 = (\ell_1;\, x_p = 20,\, x_e = 80,\, t = 0)$. The state

$$(\ell_2^{\mathsf{low}};\, x_p = 17.1,\, x_e = 50,\, t = 0.157,\, x'_p = 21.4,\, x'_e = 60,\, t' = 0.209)$$

of the new system corresponds to the pair (s,s'). We will now see that this state is reachable (in the transformed system). The state

$$(\ell_1^{\mathsf{upp}};\, x_p = 20,\, x_e = 80,\, t = 0,\, x'_p = 20,\, x'_e = 80,\, t' = 0)$$

is an initial state of the transformed system; it corresponds to the pair (s_0,s_0). Looking at Fig.3 we see that it can reach the state

$$(\ell_1^{\mathsf{upp}};\, x_p = 17.1,\, x_e = 50,\, t = 0.157,\, x'_p = 17.1,\, x'_e = 50,\, t' = 0.157)$$

(namely, when the transformed system stays in the location ℓ_1^{upp} from time point 0 to time point 10). That state can jump (by taking a transition into the lower half of the system) to the state

$$(\ell_2^{\mathsf{low}};\, x_p = 17.1,\, x_e = 50,\, t = 0.157,\, x'_p = 17.1,\, x'_e = 50,\, t' = 0.157)\,.$$

¿From now on (after a transition into the lower half of the system), only the primed variables keep changing. Looking at Fig.3 we see that this state can reach (by staying in the same location) the state that corresponds to the pair (s,s').

We will formalize next the program transformation. Given a hybrid system A we assume that the set $\mathcal{L}$ of locations of A contains m elements ℓ_1 to ℓ_m, and the set $\mathcal{V}$ consists of $n+1$ real-valued variables, namely x_1 to x_n and t. The transformed system $A^{\mathcal{T}}$,

$$A^{\mathcal{T}} = (\mathcal{L}^{\mathcal{T}}, \mathcal{V}^{\mathcal{T}}, (jump^{\mathcal{T}}_{\ell,\ell'})_{\ell,\ell' \in \mathcal{L}_{\mathcal{T}}}, (flow^{\mathcal{T}}_{\ell})_{\ell \in \mathcal{L}_{\mathcal{T}}}, (inv^{\mathcal{T}}_{\ell})_{\ell \in \mathcal{L}_{\mathcal{T}}}, (init^{\mathcal{T}}_{\ell})_{\ell \in \mathcal{L}_{\mathcal{T}}})\,,$$

consists of the following components.

1. **Variables**: The set $\mathcal{V}^{\mathcal{T}}$ of variables contains all variables of $\mathcal{V}$, and their primed versions.

$$\begin{aligned}\mathcal{V}^{\mathcal{T}} &= \{x_1,\dots,x_n,t,x'_1,\dots,x'_n,t'\}\\ &= \mathcal{V} \cup \mathcal{V}'\end{aligned}$$

2. **Locations**: Each location of the original system is duplicated, i.e. a location ℓ of the original system corresponds to two locations ℓ^{upp} and ℓ^{low} in the transformed system. We refer to the set of all locations from ℓ_1^{upp} to ℓ_m^{upp} as $\mathcal{L}^{\mathsf{upp}}$,

$$\mathcal{L}^{\mathsf{upp}} = \{\ell_1^{\mathsf{upp}},\dots,\ell_m^{\mathsf{upp}}\}$$

and to the set of locations from ℓ_1^{low} to ℓ_m^{low} as $\mathcal{L}^{\mathsf{low}}$.

$$\mathcal{L}^{\mathsf{low}} = \{\ell_1^{\mathsf{low}}, \ldots, \ell_m^{\mathsf{low}}\}$$

In addition, the transformed system has a location ℓ^{init}. Altogether, the set $\mathcal{L}^{\mathcal{T}}$ of locations of the transformed system consists of the following components:

$$\mathcal{L}^{\mathcal{T}} = \{\ell^{\mathsf{init}}\} \cup \mathcal{L}^{\mathsf{upp}} \cup \mathcal{L}^{\mathsf{low}} .$$

3. **Initial conditions**: Initially, each variable x_i has the same value as x_i' and the value of t is equal to the value of t'; the system starts in ℓ^{init}.

$$init_{\ell}^{\mathcal{T}}(x_1,\ldots,t') \equiv \begin{cases} \{(x_1,\ldots,t') \in \Sigma_{\mathcal{V}^{\mathcal{T}}} \;:\; x_1 = x_1' \wedge \ldots \wedge t = t'\} & , \quad \text{if } \ell = \ell^{\mathsf{init}} \\ \text{false} & , \quad \text{otherwise} \end{cases}$$

4. **Jump conditions**: There are two types of switches in the transformed system. The first type occurs between two locations of $\mathcal{L}^{\mathsf{upp}}$ or between two locations of $\mathcal{L}^{\mathsf{low}}$, respectively. A jump condition between location ℓ_i^{upp} and location ℓ_j^{upp} for the variables $(x_1,\ldots,t,x_1',\ldots,t')$ conforms to the jump condition between the locations ℓ_i and ℓ_j of the original system A for the variables $(x_1,\ldots,t)$. Analogously, a jump condition between the locations ℓ_i^{low} and ℓ_j^{low} of the transformed system corresponds to the jump condition from location ℓ_i to ℓ_j of the original system after replacing the variables $x_1,\ldots,t$ by their primed versions.

$$(jump_{\ell,\ell'}^{\mathcal{T}})(x_1,\ldots,t,x_1'\ldots,t') \equiv \begin{cases} (jump_{\ell,\ell'})(x_1,\ldots,t) & , \quad \text{if } \ell,\ell' \in \mathcal{L}^{\mathsf{upp}} \\ (jump_{\ell,\ell'})(x_1',\ldots,t') & , \quad \text{if } \ell,\ell' \in \mathcal{L}^{\mathsf{low}} \end{cases}$$

The second type of switches are nondeterministic jumps either between the location ℓ^{init} and a location of $\mathcal{L}^{\mathsf{upp}}$, or between a location of $\mathcal{L}^{\mathsf{upp}}$ and a location of $\mathcal{L}^{\mathsf{low}}$. A jump is always possible from the location ℓ^{init} to any location of $\mathcal{L}^{\mathsf{upp}}$, if the invariant condition of the target location is fulfilled.

$$(jump_{\ell^{\mathsf{init}},\ell'}^{\mathcal{T}})(x_1,\ldots,t,x_1'\ldots,t') \equiv (inv_{\ell'}^{\mathcal{T}})(x_1,\ldots,t,x_1'\ldots,t') \quad , \quad \text{if } \ell' \in \mathcal{L}^{\mathsf{upp}}$$

A jump from a location ℓ_i^{upp} to a location ℓ_j^{low} is possible whenever the jump condition from ℓ_i to ℓ_j is fulfilled in the original system. We use the variable $\mathsf{flag} \notin \mathcal{V}^{\mathcal{T}}$ as a discrete variable that ranges over the set $\{1,\ldots,m\}$ of indices of the locations of the system. During the jump, the index j of the target location is memorized in the variable flag.

$$(jump_{\ell_i^{\mathsf{upp}},\ell_j^{\mathsf{low}}}^{\mathcal{T}})(x_1,\ldots,t,x_1'\ldots,t') \equiv (jump_{\ell_i,\ell_j})(x_1,\ldots,t) \wedge \mathsf{flag} := j$$

If there is no jump outgoing from ℓ_i possible in the original system, the jump condition from ℓ_i^{upp} to ℓ_i^{low} in the transformed system is true.

$$\forall j \neq i : (jump_{\ell_i,\ell_j})(x_1,\ldots,t) \equiv \mathsf{false} \Rightarrow (jump_{\ell_i^{\mathsf{upp}},\ell_i^{\mathsf{low}}}^{\mathcal{T}})(x_1,\ldots,t,x_1'\ldots,t') \equiv \mathsf{true}$$

All other jump conditions are false.

5. **Flow conditions**: First, in the locations ℓ^{init} no flow of the continuous variables proceeds.

$$(flow_\ell^{\mathcal{T}})(x_1,\dots,t',\dot{x}_1,\dots,\dot{t}') \equiv \bigwedge_{x\in\mathcal{V}^{\mathcal{T}}} \dot{x}=0\ , \quad \text{if } \ell=\ell^{\mathsf{init}}$$

In each location ℓ_i^{upp} of $\mathcal{L}^{\mathsf{upp}}$, the flow of the variables $x_1,\dots,t$ in the transformed system is the same as the flow of $x_1,\dots,t$ in the original system; each variable $x'_1,\dots,t'$ behaves exactly like its unprimed version, that is the flow of $x'_1,\dots,t'$ is equal to the flow of the original system after replacing the variables $x_1,\dots,t$ by their primed versions $x'_1,\dots,t'$.

$$(flow_\ell^{\mathcal{T}})(x_1,\dots,t',\dot{x}_1,\dots,\dot{t}') \equiv flow_\ell(x_1,\dots,\dot{t}) \wedge flow_\ell(x'_1,\dots,\dot{t}')\ , \quad \text{if } \ell\in L^{\mathsf{upp}}$$

In each location of $\mathcal{L}^{\mathsf{low}}$ the values of the variables $x_1,\dots,t$ are fixed, i.e. the flow of them is constant. The variables $x'_1,\dots,t'$ keep on evolving as before.

$$(flow_\ell^{\mathcal{T}})(x_1,\dots,t',\dot{x}_1,\dots,\dot{t}') \equiv \bigwedge_{x\in\mathcal{V}} \dot{x}=0 \wedge flow_\ell(x'_1,\dots,\dot{t}')\ , \quad \text{if } \ell\in\mathcal{L}^{\mathsf{low}}$$

6. **Invariant conditions**: For the location ℓ^{init}, the invariant condition is true.

$$(inv_\ell^{\mathcal{T}})(x_1,\dots,t,x'_1,\dots,t') \equiv \mathsf{true}\ , \quad \text{if } \ell=\ell^{\mathsf{init}}$$

For a location ℓ_i^{upp} in $\mathcal{L}^{\mathsf{upp}}$ (or ℓ_i^{low} in $\mathcal{L}^{\mathsf{low}}$, respectively), the invariant condition over $x_1,\dots,t,x'_1,\dots,t'$ is the same as the invariant condition of the original system A for ℓ_i over $x_1,\dots,t$ (or $x'_1,\dots,t'$, respectively).

$$(inv_\ell^{\mathcal{T}})(x_1,\dots,t,x'_1,\dots,t') \equiv \begin{cases} inv_\ell(x_1,\dots,t) & , \quad \text{if } \ell\in\mathcal{L}^{\mathsf{upp}} \\ inv_\ell(x'_1,\dots,t') & , \quad \text{if } \ell\in\mathcal{L}^{\mathsf{low}} \end{cases}$$

Step 2: Reachability analysis

In the second step, our algorithm applies a procedure on the transformed system that computes an overapproximation of the set of all reachable states of the transformed system. The procedure is implemented by existing reachability tools as PHAVer [12], d/dt [11] or HSolver [29]. As result we obtain a set of constraints, given by a disjunction of conjunctions of linear inequalities in case of PHAVer . Each constraint is marked by the location of the transformed system it is related to. In the example of the heating system, one constraint in the output of PHAVer is e.g.

$$\begin{aligned} \ell_2^{\mathsf{low}}: \quad & \mathsf{flag}=2,\ x_p\geq 20,\ -x_p\geq -21,\ x_e\geq 0,\ -x_e\geq -55, \\ & -x'_p>-20,\ -x'_e\geq -80,\ -x'_e+300(t'-t)\geq 245,\ t'-t\geq 1, \\ & x'_e-140(t'-t)\geq -90,\ x'_p-75(t'-t)\geq -75 \end{aligned}$$

In our notation, we identify a constraint with the relation that it denotes. We view a unary relation over the variables $\mathcal{V}^{\mathcal{T}}$ of the transformed system as a binary relation over the values of the variables $\mathcal{V}$ and their primed versions $\mathcal{V}'$ of the original system. Each relation refers to pairs of valuations (ν,ν'), where the pair of states $((\ell,\nu),(\ell',\nu'))$ is binary reachable in the original system for some ℓ and ℓ'.

In the remainder of this paper we only talk about binary relations over pairs of valuations of the original system (and not about unary relations over valuations of the transformed system) when we refer to relations in the output of the reachability tool.

Step 3: Computation of a Lyapunov-like function

For the third step of the algorithm we consider the finite subset C of disjuncts in the output of the reachability tool where the value of the variable flag is equal to the index of the location the relation is related to. These relations refer to pairs of valuations (ν,ν') such that the pair of states $((\ell_i,\nu),(\ell_i,\nu'))$ is binary reachable in the original system for the location ℓ_i whose index i is equal to the value of the variable flag.

We prove for each single relation c of C that its conjunction with the negation of the region φ and with $t'-t \geq \delta$

$$c \wedge \neg\varphi \wedge (t'-t \geq \delta)$$

is well-founded, where $\delta > 0$ is arbitrarily small. *Well-founded* means that there is no infinite sequence of states $s_1, s_2, s_3, \ldots$ such that each pair of consecutive states (s_i, s_{i+1}) satisfies the relation.

To show that a relation is well-founded, the algorithm applies a procedure that automatically constructs a *Lyapunov-like* function for the relation. By a Lyapunov-like function we mean a function r over the real-valued variables $\mathcal{V}$, such that (1) $r(x_1,\ldots,x_n,t) \geq 0$ for all $(x_1,\ldots,x_n,t)$, and (2) $r(x_1,\ldots,x_n,t) < r(x_1',\ldots,x_n',t')$ for all $(x_1,\ldots,x_n,t,x_1',\ldots,x_n',t')$ that fulfill the considered constraint. For relations that are given by conjunctions of linear inequalities, the algorithm computes a Lyapunov-like function using RankFinder [31], a tool for synthesizing linear ranking functions [24, 5, 6].

In section 6, we will prove that this condition suffices to show, that every trajectory of the original system A inevitably reaches the region φ.

For the sample formula above, we obtain the result "Ranking: $r = [1,0,0]$" which means that the Lyapunov-like function

$$r(x_p,x_e,t) = x_p$$

is a witness for *inevitability* of the evolution towards φ.

Step 4: Invariance

In a final step our algorithm checks the entailment between constraints in the form below, where c is a constraint given by the output of the reachability tool in Step 2 of the algorithm (the renaming of all variables in φ to their primed versions yields φ').

$$\varphi \wedge c \quad \models \quad \varphi'$$

This check proves that the region φ is an invariant of the system, i.e. each evolution of a state in the region leads to another state that is in the region again. For linear constraints c the algorithm uses the linear constraint solver clp(Q,R) [16] for this entailment check.

6 Correctness

In this section we investigate the correctness of the algorithm. The algorithm is sound (its definite answers are correct) and not complete (it may return don't know answers).

Soundness The hybrid system A is stable wrt. φ if (1) every trajectory of A must reach the region φ after a amount of finite time, and (2) from then on it will never leave the region again.

Assume that the set C contains all relations c in the output of the reachability analysis for the transformed system (computed in Step 2 of the algorithm) where the value of the variable flag is equal to the index of the location the relation is related to. Again, these relations refer to pairs of valuations (ν,ν') such that the pair of states $((\ell,\nu),(\ell,\nu'))$ is binary reachable in the original system for the same location ℓ. We must show that property (1) holds for A if the conjunction of each relation c in the set C with the negation of the region φ and with $t'-t \geq \delta$ (for any arbitrary small $\delta > 0$)

$$c \wedge \neg\varphi \wedge (t'-t \geq \delta)$$

is well founded.

The well-known combinatorial argument used to show that this is indeed sufficient (and that the algorithm is correct) is standard in the theory of Büchi automata and has been used so far only for linear temporal properties of discrete systems [28, 25, 26].

Theorem 1. *Assume a hybrid system A and a solution of the reachability analysis for the transformed system $A^{\mathcal{T}}$ such that the set C consists of all relations c of the solution where the value of the variable* flag *is equal to the index of the location the relation is related to.*

The hybrid system A reaches a region φ in every trajectory after a finite amount of time if C is a finite set of relations where the conjunction of each relation c of C with the negation of the region φ and with $t'-t \geq \delta$

$$c \wedge \neg\varphi \wedge (t'-t \geq \delta)$$

is well-founded for any arbitrary small $\delta > 0$.

Proof. For a proof by contradiction we assume that each relation $c \wedge \neg\varphi \wedge (t'-t \geq \delta)$, for c in c, is well-founded but A does not reach the region φ in every trajectory. Let τ be a trajectory of the system A that does not reach φ.

We consider a *discretization* of the trajectory τ by a time interval $\delta > 0$, that is the infinite sequence

$$\tau(0), \tau(\delta), \tau(2\delta), \ldots$$

The sequence is infinite, but we have only finitely many locations. Hence, at least one location, say ℓ, appears infinitely often in the sequence. This means that we can build an infinite subsequence $\tau_0, \tau_1, \tau_2, \ldots$ of the sequence $\tau(0), \tau(\delta), \tau(2\delta), \ldots$, such that all states on the subsequence have the same location ℓ.

We now use the assumption that c is a finite union of relations, say

$$C = c_1 \cup \ldots \cup c_k$$

such that for each relation c_j of c its conjunction with the negation of the region φ and with $t' - t \geq \delta$

$$c_j \wedge \neg\varphi \wedge (t' - t \geq \delta)$$

is well-founded.

We define a function g with finite range that maps an ordered pair of indices of the sequence $\tau_0, \tau_1, \tau_2, \ldots$ to the index j of the relation c_j that contains the corresponding pair of states.

$$g(k,l) \overset{\text{def.}}{=} j \quad \text{if} \quad (\tau_k, \tau_l) \in c_j$$

Furthermore the function g induces an equivalence relation $\sim$ on pairs of indices of the sequence $\tau_0, \tau_1, \tau_2, \ldots$.

$$(k_1, l_1) \sim (k_2, l_2) \quad \overset{\text{def.}}{\Leftrightarrow} \quad g(k_1, l_1) = g(k_2, l_2)$$

The index of $\sim$ is finite since the range of g is finite. By Ramsey's Theorem [28], there exists an infinite set of indices K such that all pairs from K belong to the same equivalence class. Thus, there exists m and n in K, with $m < n$, such that for every k and l in K, with $k < l$, we have $(k,l) \sim (m,n)$. Let $k_1, k_2, \ldots$ be the ascending sequence of elements of K. Hence, for the infinite sequence $\tau_{k_1}, \tau_{k_2}, \ldots$ we have

$$(\tau_{k_i}, \tau_{k_{i+1}}) \in c_j \quad \text{for all } i \geq 1$$

By our assumption that τ does not reach φ, each state τ_{k_i} is not in the region φ, which yields that

$$(\tau_{k_i}, \tau_{k_{i+1}}) \in c_j \wedge \neg\varphi \quad \text{for all } i \geq 1$$

Because we have chosen a discretization of τ by δ, this is a contradiction to the well-foundedness of $c_j \wedge \neg\varphi \wedge (t' - t \geq \delta)$. □

Incompleteness The algorithm may fail to prove the stability of a correct system (and return a don't know answer) for one of the following three reasons.

First, the output of the existing reachability tools (that we use in Step 2 of our algorithm) is only an overapproximation of the set of all reachable states (and not the set itself) due to the fact that reachability in general is undecidable.

The second point is the incompleteness of general well-foundedness tests (used in Step 3 of the algorithm). Complete tests exist only in some restricted cases (e.g. in the form of termination checkers for small classes of programs [24, 33]).

The third source of incompleteness is that the algorithm checks whether a region φ is a strong attractor of the system, which is only a sufficient but not necessary condition for stability wrt. φ; see Section 4.

7 Conclusion and Future Work

Previous notions of stability refer to a single equilibrium point. We have introduced a new notion of stability that refers to a region instead. For some cases of hybrid systems, this gives the appropriate formalization of their correctness. We have situated our notion in the landscape of related properties in control theory and model checking.

Verification methods for non-reachability properties (or properties that can be reduced to non-reachability) for hybrid systems have been intensively studied by both computer scientists and control theorists [32, 10, 34, 8, 27] and have lead to popular verification systems such as PHAVer [12], HSolver [29], d/dt [11] and CheckMate [7].

There are many methods for the verification of hybrid systems for non-reachability properties or properties that can be reduced to non-reachability; stability does not belong to them. We have given an algorithm to verify stability properties (in the new sense) for general hybrid systems. The algorithm is parameterized by the constraint solver that it calls as a subroutine in each of its different steps. Using a constraint solver for linear constraints, we obtain a specific algorithm for linear hybrid systems.

The crucial step of the algorithm is the computation of binary reachability (a precise enough approximation of the binary reachability relation). Thanks to a source-to-source transformation, this step can be implemented using an off-the-shelf tool for (unary) reachability. Future work consists of evaluating existing (or new) reachability tools in our context, where we use them not for safety but for stability.

In preliminary experiments, we have run the different steps on a number of examples (using PHAVer [12] and RankFinder [31]), including the example of the heating system. The experiments indicate a promising potential of our method.

Out of the three sources of incompleteness of our algorithm, two are inherent due to recursion-theoretic properties. The question is whether the third source of incompleteness can be circumvented by an alternative to our present definition of strong attractors and ways to compute them.

Acknowledgment

We thank Oded Maler and Thao Dang for numerous insightful discussions on reachability and stability analysis for hybrid system, Stefan Ratschan for comments and suggestions on this paper, and Li Hong and Goran Frehse for their help with the first practical experiments. We thank Oliver Theel, Martin Fränzle and their students for discussions during the AVACS meetings.

References

1. R. Alur, C. Courcoubetis, T.A. Henzinger, and P.-H. Ho. Hybrid Automata. An Algorithmic Approach to the Specification and Verification of Hybrid Systems. In Hybrid Systems: Computation and Control, 1993.
2. M.S. Branicky. Stability of hybrid systems: State of the art. In Conference on Decision and Control, 1997.
3. M.S. Branicky. Multiple Lyapunov functions and other analysis tools for switched and hybrid systems. In Trans. on Automatic Control, 1998.
4. A. Biere, C. Artho, and V. Schuppan. Liveness checking as safety checking. In Formal Methods for Industrial Critical Systems (FMICS), 2002.
5. A. Bradley, Z. Manna, and H.B. Sipma. Linear Ranking with Reachability. In Computer Aided Verification (CAV), 2005.
6. A. Bradley, Z. Manna, and H.B. Sipma. The Polyranking Principle. In International Colloquium on Automata, Languages and Programming (ICALP), 2005.

7. A. Chutinan, A. Fehnker, Z. Han, J. Kapinski, R. Kumar, B.H. Krogh, and O. Stursberg. CheckMate, http://www.ece.cmu.edu/ webk/checkmate.
8. E.M. Clarke, A Fehnker, Z. Han, B. Krogh, O. Stursberg, and M. Theobald. Verification of Hybrid Systems Based on Counterexample-Guided Abstraction Refinement. In Tools and Algorithms for the Construction and Analysis of Systems (TACAS), 2003.
9. B. Cook, A. Podelski, A. Rybalchenko. Termination Proofs for Systems Code. Submitted to Conference on Programming Language Design and Implementation (PLDI), 2006.
10. M. Colon, S. Sankaranarayanan, and H. Sipma. Linear invariant generation using non-linear constraint solving. In Computer Aided Verification (CAV), 2003.
11. T. Dang and O. Maler. d/dt, http://www-verimag.imag.fr/ tdang/Tool-ddt/ddt.html.
12. G. Frehse. PHAVer , http://www.cs.ru.nl/ goranf.
13. T.A. Henzinger. The Theory of Hybrid Automata. In Logic in Computer Science (LICS), 1996.
14. T.A. Henzinger, P.-H. Ho, and H. Wong-Toi. Algorithmic analysis of nonlinear hybrid systems. In Automatic Control, 1998.
15. T. Henzinger, P.-H. Ho, and H. Wong-Toi. HyTech, http://www-cad.eecs.berkeley.edu/ tah/HyTech.
16. C. Holzbaur. clp(Q,R), http://www.ai.univie.ac.at/clpqr.
17. D. Liberzon. Switching in Systems and Control. Birkhäuser, 2003.
18. D. Liberzon, and A.A. Agrachev. Lie-algebraic stability criteria for switched systems. In Control and Optimization, 2001.
19. D. Liberzon, J.P. Hespanha, and A.S. Morse. Stability of switched systems: a Lie-algebraic condition. In Systems and Control Letters, 1999.
20. D. Liberzon, and M. Margaliot. Lie-algebraic stability conditions for nonlinear switched systems and differential inclusions, Systems and Control Letters, to appear.
21. V. Lakshmikantham, S. Leela, and A.A. Martynyuk. Practical Stability of Nonliear Systems. World Scientific Pub Co Inc, 1990.
22. A. Papachristodoulou, and S. Prajna. On the Construction of Lyapunov Functions using the Sum of Squares Decomposition. In Conference on Decision and Control (CDC), 2002.
23. S. Pettersson. Analysis and Design of Hybrid Systems. Ph.D. Thesis, Chalmers University of Technology, Göteborg, Sweden, 1999.
24. A. Podelski, and A. Rybalchenko. A complete Method for the Synthesis of Linear Ranking Functions. In Verification, Model Checking and Abstract Interpretation (VMCAI), 2004.
25. A. Podelski, and A. Rybalchenko. Transition invariants. In Logic in Computer Science (LICS), 2004.
26. A. Podelski, and A. Rybalchenko. Transition Predicate Abstraction and Fair Termination. In Principles of Programming Language (POPL), 2005.
27. S. Prajna, and A. Jadbabaie. Safety Verification of Hybrid Systems Using Barrier Certificates. In Hybrid Systems: Computation and Control, 2004.
28. F.P. Ramsey. On a problem of formal logic. In Proc. of the London Mathematical Society 30, 1930.
29. S. Ratschan, and Z. She. HSolver, http://www.mpi-sb.mpg.de/ ratschan/hsolver.
30. S. Ratschan, and Z. She. Safety Verification of Hybrid Systems by Constraint Propagation Based Abstraction Refinement. In Hybrid Systems: Computation and Control, 2005.
31. A. Rybalchenko. RankFinder, http://www.mpi-inf.mpg.de/ rybal/rankfinder.
32. S. Sankaranarayanan, H. Sipma, and Z. Manna. Constructing Invariants for Hybrid Systems. In Hybrid Systems: Computation and Control, 2004.
33. A. Tiwari. Termination of linear programs. In Computer Aided Verification (CAV), 2004.
34. A. Tiwari, H. Ruess, H. Saidi and N. Shankar. Automatic Generation of Invariants. In Tools and Algorithms for the Construction and Analysis of Systems (TACAS), 2001.