A Method and a Tool for Automatic Verification of Region Stability for Hybrid Systems

Andreas Podelski Silke Wagner

January 25, 2007

MPI–I–2007–2–001 January 2007

Authors' Addresses

Andreas Podelski
Universität Freiburg
Georges-Köhler-Allee 52
D-79110 Freiburg
podelski@informatik.uni-freiburg.de

Silke Wagner
Max-Planck-Institut für Informatik
Stuhlsatzenhausweg 85
D-66123 Saarbrücken
swagner@mpi-inf.mpg.de

Abstract

We propose a model checking method and tool that integrates state abstraction techniques for the automatic proof of a stability property for hybrid systems called *region stability*. It is based on a new notion of *snapshots* which yield characteristic discretizations of trajectories. We have implemented the tool and applied it to solve a number of verification problems, including the fully automatic stability proof for the break curve behavior of a train system.

Keywords

Hybrid Systems, Region Stability, Proof Rule, Tool

1 Introduction

For a large class of correctness properties of hybrid systems, push-button verification methods such as model checking have reached a certain degree of practicality. Those properties have in common that they can be reduced to non-reachability. In contrast, for the class of *stability* properties, which is studied intensively in control theory (see e.g. [4, 5, 6, 26, 27]), practical pushdown verification methods have so far been out of reach. These properties can not be reduced to non-reachability.

Existing verification methods for stability are based on Lyapunov theory, see e.g. [4, 5, 6, 26, 27]. They all share the drawback that state abstraction techniques are intrinsically not applicable. Since such abstraction techniques (used for overapproximation of state spaces and based formally on abstract interpretation [9, 10]) have been crucial to obtain scalability in existing model checkers for hybrid systems (see e.g. [18, 19]), it seems useful to investigate new verification methods for stability.

In this paper, we will propose a model checking method and tool that integrates state abstraction techniques for the automatic proof of a particular stability property called *region stability* [28, 33]. We will describe the experiments with a prototypical implementation of our tool. The experiments yield fully automatic stability proofs for a number of typical benchmark problems including the the break curve behavior of a train system, a previously open challenge problem for the AVACS project (www.avacs.org).

Region stability means that for each trajectory there exists a point of time after which it never (again) leaves the given region. Before this time point the trajectory can run either inside or outside of the region and it can reach the region and leave it again arbitrarily often. Region stability is thus characterized by the finiteness (NOT boundedness!) of the period of time that a trajectory can spend outside of the region.

Our model checking method is based on a new notion of *snapshots*. That is, we have a new characterization of region stability in terms of the finiteness of particular (discrete) sequences of states; the states in each sequence form one of three particular kinds of time divergent *snapshots* of the trajectory. We can com-

pute effective representations of the three corresponding sets of such sequences by constraints that denote binary relations between states (binary relations represent sets of sequences in the obvious way: each pair of consecutive elements in the sequence must lie in the binary relation). The finiteness condition for a set of sequences is equivalent to the well-foundedness of the corresponding binary relation, a condition which can be tested efficiently by a tool based on constraint solving [36].

The algorithm to compute the constraint representations of the three sets of the individual kinds of snapshot sequences works in two steps. The first step is the syntactic transformation of the hybrid system into another one such that the reachability relation of the new hybrid system is the binary reachability relation between snapshot states of the original hybrid system. The second step is to compute an overapproximation of the unary reachability relation of the new hybrid system. It is this step that allows us to integrate state abstraction techniques. Our tools relies on the particular abstraction used in the model checker PHAVer [18].

In summary, the contribution of this paper is the, to our knowledge first, abstraction-based method and tool for automatically checking a form of stability on hybrid systems. The possibility to use well-established state abstraction techniques gives rise to an interesting potential of practicality, as indicated by the experiments with a prototypical implementation of our tool.

2 Preliminary Definitions

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 $\dot{\nu}$.

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
$$\tau(0) \models \mathit{init}_\ell \ .$$

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
$$(\nu, \dot{\nu}) \models \mathit{inv}_\ell \wedge \mathit{flow}_\ell \ .$$

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

Intuitively, a trajectory of a hybrid system consists of (finitely or infinitely many) smooth parts which are connected via jump discontinuities. The smooth parts follow the continuous flow in one location. Next we consider states on the same trajectory such that no jump occurs between the states.

Definition 1 (States on the same flow) *Given a hybrid system A we say that two states s and s'* lie on the same flow *of a location ℓ of A,*
$$s \rightsquigarrow_\ell s' \ ,$$
if s and s' are states on the same trajectory τ of the hybrid system A and no jump occurs between s and s' on τ; formally

1. $s = \tau(t) \ , \ s' = \tau(t') \ , \ t < t'$ *and*
2. $\exists \ell \, \forall t'' \in [t, t'] \ : \ \tau(t'') = (\ell, \_) \ .$

Definition 2 (Region stability [33]) *We call a hybrid system* stable *with respect to a 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.*
$$\forall \tau \, \exists t_0 \, \forall t \geq t_0 \ : \ \tau(t) \in \varphi$$

Region stability is reminiscent to *practical stability* [28, 20, 40, 41]; for a comparison we refer to [33].

In the remainder of this paper we restrict ourselves to regions φ that are given by

$$\varphi \quad \equiv \quad x \in [x_{min}, x_{max}] \,,$$

where x is a continuous variable in $\mathcal{V}$ and x_{min} and x_{max} are constants with $x_{min} < x_{max}$. We call such regions **interval regions**.

3 Benchmarks

The goal of this section is to illustrate the notion of region stability and to show that stability checking is practically feasible (for several challenging problems). Therefor we give eight examples of hybrid systems with different particular properties. For every example our tool can automatically check stability.

3.0.1 Example 1

Our first example (Fig. 3.1) is a hybrid system with one location and one continuous variable x. Initially the value of x is greater than 0; the flow condition is given by $\dot{x} = -1$. The region with respect to which we want to proof stability is given by $x \leq 0$.

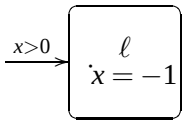

Figure 3.1: Example 1.

Intuitively it is clear that all trajectories of the system must end up in the region φ (since the value of x is strictly monotonically decreasing by -1). For every single trajectory the amount of time that the trajectory can spend outside of the region φ is finite. However, the time that a trajectory can spend outside of φ is unbounded.

3.0.2 Simple Heating System

In our second example we consider a well-known heating system for a room, see Fig. 3.2.

This system is not stable in the classical sense (wrt. an equilibrium point). We want to show stability wrt. the region $x \in [65, 82]$.

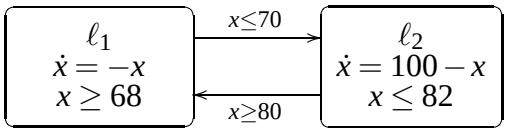

Figure 3.2: Simple heating system.

3.0.3 More Complex Heating System

Our next example is a modification of the heating system that we have seen before. The modified heating system (Fig. 3.3) consists of two continuous variables x_p and x_e; x_p stands for the temperature of the room and x_e for the temperature of an internal engine.

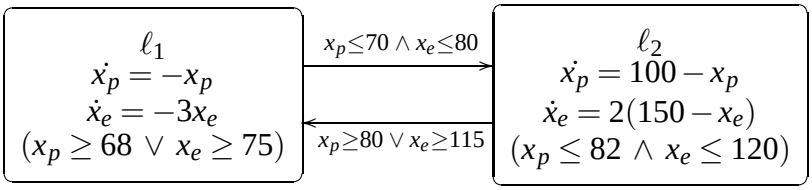

Figure 3.3: Modified heating system.

The internal engine may overheat and switch off the heater temporarily, even though the desired temperature for the room (again given by $x \in [65, 82]$) is not yet reached. This means that, starting from low, the temperature will not increase strictly monotonically but it will also decrease during some periods of time. A side effect of this behavior is that a trajectory can reach the desired region but leaves it again for some time before it stabilizes.

3.0.4 One-tank Water System

In the following example we consider a one-tank water system with a constant inflow of water (see Fig. 3.4). The volume of water in the tank is denoted by x. The tank has a pipe such that water can also flow out of the tank again. The pipe can be opened for at most 8 seconds; after that the pipe must be closed again for 10 seconds. We want to know whether the tank can be drained, no matter what the initial volume of water is; i.e. we must check whether the system is stable wrt. $x <= 0$.

As in Example 1, the time that a trajectory of this system can spend outside of the desired region is unbounded. Furthermore we prove in this example stability

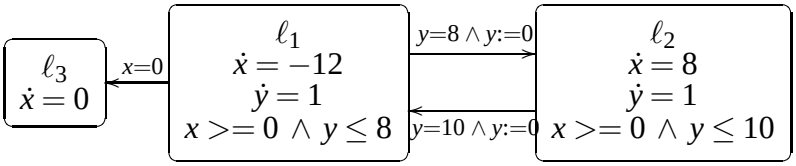

Figure 3.4: One-tank water system.

wrt. the equilibrium point $x = 0$. (Since the invariants of the system assure that $x \geq 0$, stability wrt. $x \leq 0$ implies stability wrt. the equilibrium point $x = 0$.)

3.0.5 Two-tank Water System

Now we consider a two-tank water system consisting of two tanks one upon the other. The variables x_1 and x_2 denote the volume of water in the upper tank 1 and the lower tank 2. Water flows constantly out of the system from the lower tank. The system can switch on or off the inflow of water into the upper tank, and the flow of water from the upper to the lower tank; but both tanks must not overflow. The objective is to keep the water volume of the lower tank above 6, i.e. we are interested in stability wrt. the region $x_2 > 6$. The hybrid system that models this scenario consists of two continuous variables and four locations, see Fig. 3.5.

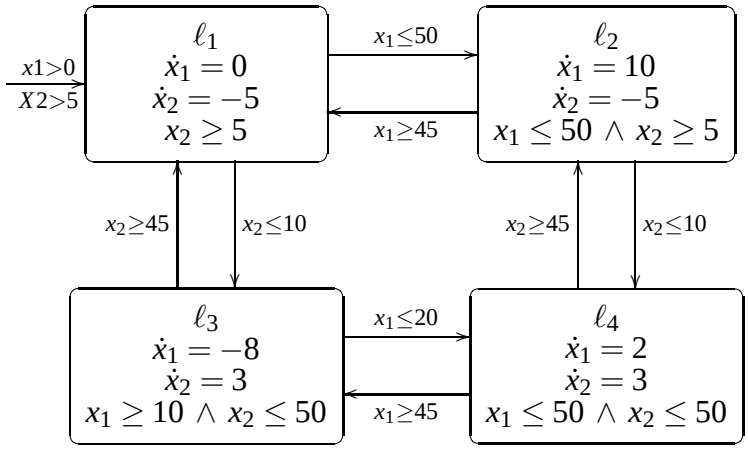

Figure 3.5: Two-tank water system

3.0.6 Distance Controller

The next example is a model of a distance controller, see Fig. 3.6. We consider two cars driving one after another. The leading car has a constant speed $v_1 > 0$.

The second car is governed by a controller that continuously senses the distance between the two cars. If the distance is greater than a given value D_{acc} the second car speeds up; if the distance is smaller than D_{dec} it slows down. The second car has a maximum speed of $v_{\max}$ and a minimum speed of 0. The goal is to prove that the distance x between the two cars is always > 0 (this means that the two cars do not crash).

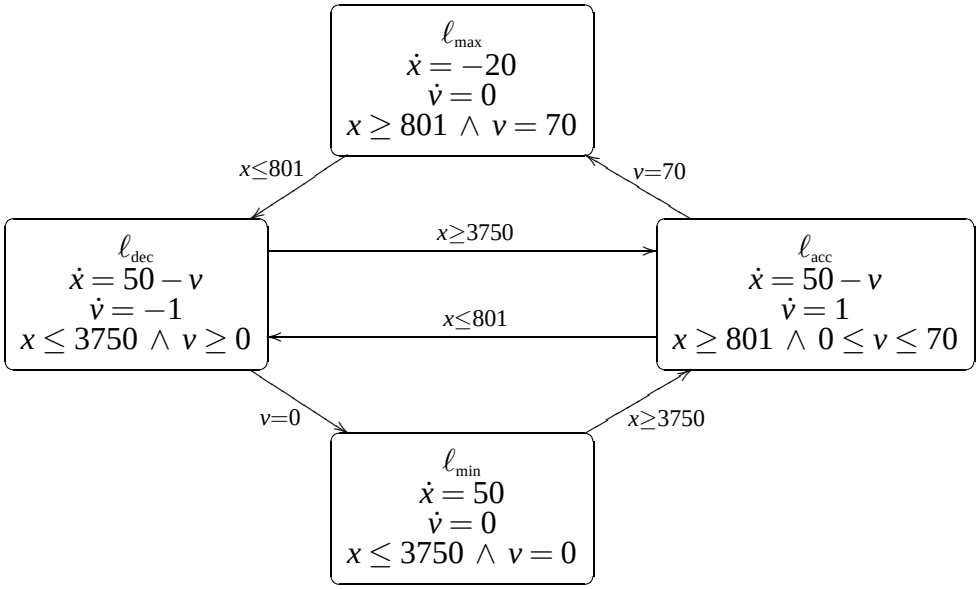

Figure 3.6: Distance controller.

3.0.7 Bouncing Ball

This example is a modification of the well-known bouncing ball, see Fig. 3.7. A ball (thought of as a point-mass) is thrown horizontally against a wall. The distance between the wall and the thrower is denoted by x_t, the distance between the wall and the ball is denoted by x_b. We assume that the ball has a constant speed and does not lose any energy with a bounce. As soon as the thrower has thrown the ball he moves towards the wall until the ball returns to him; then he throws the ball again.

Each execution of the hybrid system in Fig. 3.7 is a Zeno execution, this means a solution of the system having infinitely many discrete jumps in finite time. Nevertheless we can show that the thrower can come arbitrarily close to the wall, i.e. we can prove stability of the system wrt. the region $x_t \leq \varepsilon$ for every $\varepsilon > 0$.

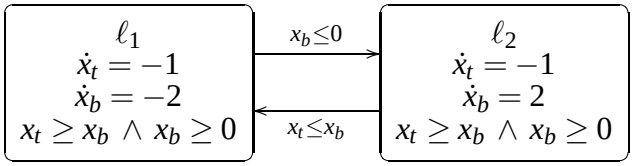

Figure 3.7: Modified bouncing ball.

3.0.8 Train Breaks

In our last example we consider the braking behavior of a train, see Fig. 3.8. Initially the train is moving with a constant speed v. Eventually it starts braking, either with one break (if the speed is ≤ 200) or with two brakes (if the speed is > 200). There is a time delay between ordering the brake application and reaching the full brake effort. The braking capacity of the train depends on the speed. If the train is decelerated to a speed between 180 and 200 the second brake is released again. We want to prove stability of the system with respect to $v \leq 0$, i.e. we want to show that the train can always stop.

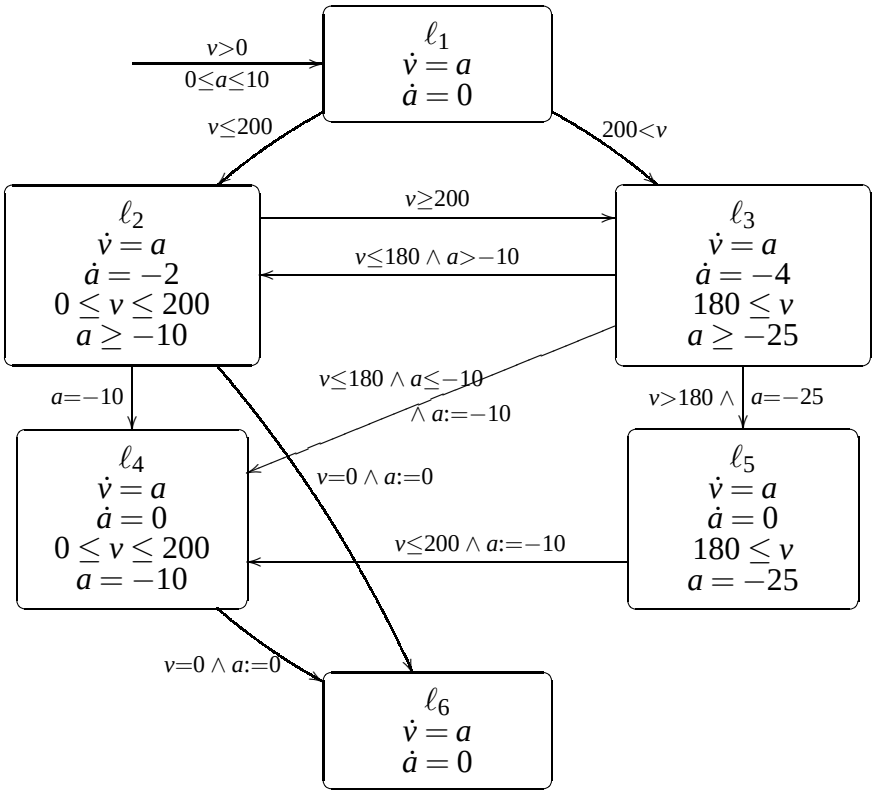

Figure 3.8: Train Brakes.

Our tool can automatically check stability for all of the examples above. The run times are listed in Fig. 3.9.

System	Run Time
Example 1	0.191s
Simple heater	0.490s
Complex heater	1.920s
One tank system	1.813s
Two tank system	16.545s
Distance controller	1.186s
Bouncing ball	4.209s
Train brake	2.589s

Figure 3.9: Run times of the eight benchmarks.

4 Stability Criterion for Linear, One-dimensional Hybrid Systems

In this section we will give a criterion for region stability in the most simple case of linear hybrid systems with only one continuous variable x (other than the default variable t that denotes the time). In Section 4.1 we consider hybrid systems with only one location (a degenerated case of a hybrid system) and in Section 4.2 we consider linear systems with arbitrarily (but finitely) many locations.

The next definition allows us to talk about "snapshots" of the computation of a hybrid system along one trajectory.

Definition 3 (Sequence of snapshots) *Given a hybrid system A and a region φ a* sequence of snapshots *is a sequence of states such that (i) all states of the sequence lie on the same trajectory τ of A, (ii) all states are not in the region φ and (iii) all pairs of consecutive states have a minimum time distance δ, where δ is an arbitrary but fixed constant greater than 0; formally*

$$s_0, s_1, s_2, \ldots$$

such that

(i) $\exists \tau \, \forall i \, \exists t_i \, : \; s_i = \tau(t_i)$,

(ii) $\forall i \, : \; s_i \notin \varphi$,

(iii) $\exists \delta > 0 \, \forall i \, : \; t_{i+1} - t_i \geq \delta$.

4.1 Linear, One-dimensional, One Location

We consider a linear hybrid system with one location. The flow condition is given by

$$\dot{x} \; = \; ax + b \, ,$$

where a and b are real constants.

The solution of $\dot{x} = ax + b$ is always a strictly monotonic function, namely

$$x(t) \;=\; c \cdot e^{at} - b/a\,,$$

where c is a constant depending on the initial value of x. Possible trajectories for such a system would e.g look like in Fig. 4.1.

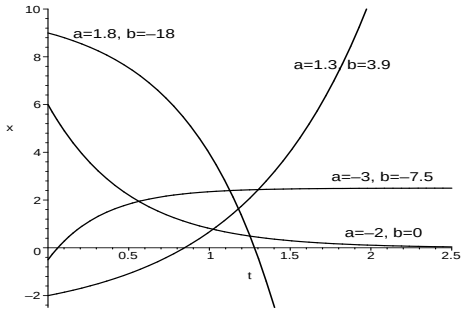

Figure 4.1: Sample trajectories of linear hybrid systems with one location and flow condition $\dot{x} = ax + b$: the trajectories are either strictly monotonic increasing to infinity or towards an asymptote, or they are strictly monotonic decreasing to minus infinity or towards an asymptote.

We will now motivate informally the condition for region stability of one-dimensional hybrid systems with one location, and why it is sufficient. By the monotonicity of the trajectories it follows that all trajectories of a stable system can never leave the region $\varphi \equiv x \in [x_{min}, x_{max}]$ again after they have reached it *once*. (Otherwise the trajectory could never return to the region again.) Or the other way round: if a system is not stable wrt. φ then it must have a trajectory τ that (1) either never reaches the region or (2) reaches the region and leaves it again for good.

In both cases exists an infinite computation of the system (starting either from the beginning, i.e. at $\tau(0)$, or from the time point t_0 when the trajectory has just left φ, i.e. at $\tau(t_0)$) such that all states of the computation are not in the region φ.

Hence we can reduce stability wrt. an interval region φ to the existence of infinite computations outside of φ. To check whether or not such an infinite computation exists we consider sequences of snapshots of the hybrid system.

<u>Condition 1</u>: There is no infinite sequence of snapshots such that all states of the sequence lie on the same flow.

Or, in other words, if we consider an infinite sequence of snapshots $s_0, s_1, s_2, \ldots$ on the same flow then there must be a state in this sequence, say s_n, that is in the region φ.

Lemma 1 *Condition 1 is sufficient and necessary for region stability of a linear hybrid system A with one location ℓ and one continuous variable x wrt. an interval region φ.*

Proof: The if-direction is clear from the above. We refer the only if-direction to Theorem 1. □

4.2 Linear, One-dimensional, m Locations

Now we consider linear hybrid systems with several locations and one continuous variable *x*.

The next example shows that for the more general case Condition 1 is not sufficient for proving stability when there are several locations. We consider a hybrid system with two locations. In location ℓ_1 the value of *x* is increasing, in ℓ_2 the value of *x* is decreasing.

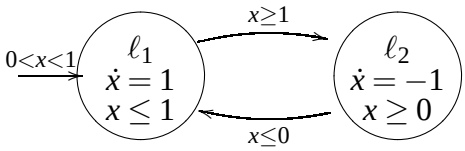

Figure 4.2: Example: Linear hybrid system with 2 locations. In location ℓ_1 the value of *x* is increasing, in ℓ_2 the value is decreasing.

We want to know whether this system is stable wrt. the interval region φ,

$$\varphi \quad \equiv \quad x \in [0.4, 1]\,;$$

one sample trajectory of this system is shown in Fig. 4.3.

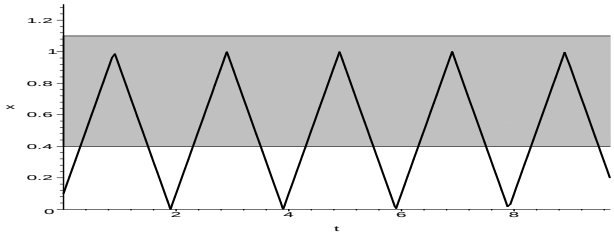

Figure 4.3: A sample trajectory of the hybrid system in Fig. 4.2. The system is not stable wrt. the grey region $\varphi \equiv x \in [0.4, 1]$ although Condition 1 holds.

If we consider an arbitrary sequence of snapshots on the same flow of this system and if we choose the time distance $\delta = 1/2$ then Condition 1 holds because

either s_0 is already in φ or (if not) s_1 is in φ or (if neither) the invariant condition of the location is violated after a time elapse of 1/2. But the system is obviously not stable wrt. φ.

Thus, for stability of a linear system with several locations we do not only need to consider the flows in each single location. We also have to take the jumps into account. We call the states just after a jump *entry-points*.

Definition 4 (Entry-points) *Given a hybrid system A a state s is called an* entry-point *if it lies on a trajectory τ of A immediately after a discrete jump; formally*

$$\begin{array}{ll} 1. & s = \tau(t) = (\ell, \nu)\,, \\ 2. & \exists\, {}'\ell \in \mathcal{L} : \lim_{{}'t \to t} \tau({}'t) \models \mathit{jump}_{{}'\ell,\ell}\,. \end{array}$$

By Entry_ℓ *we denote the set of all entry-points of a location ℓ of A.*

With help of the next condition we ensure that a hybrid system can not have infinitely many entry-points outside of the region φ.

Condition 2: There is no infinite sequence of snapshots such that all states of the sequence are entry-points.

Lemma 2 *Condition 1 and Condition 2 together are sufficient and necessary for region stability of a linear hybrid system A with m locations and one continuous variable x wrt. an interval region φ.*

Proof: if-direction: We assume that for the hybrid system A the Conditions 1 and 2 hold but A is not stable wrt. φ, i.e. the system has a trajectory τ that does not stabilize.

The first possibility is that the non-stabilizing trajectory τ will run forever outside of φ from a time point t_0 on.

$$\exists t_0\, \forall t > t_0 : \quad \tau(t) \notin \varphi$$

We consider an arbitrary sequence of entry-points on τ after t_0,

$$\tau(t_1), \tau(t_2), \tau(t_3), \ldots$$

such that all pairs of consecutive states have a minimum time distance δ. Obviously, all states of this sequence are not in φ. By Condition 2 this sequence is finite, say up to a state $\tau(t_k)$, and the computation of the system A proceeds in one location ℓ after t_k (since no more entry-point and thus no more

jump occurs). But then we know by Lemma 1 that an infinite sequence of snapshots on the flow of location ℓ must exist, a contradiction to Condition 1.

The second possibility is that the non-stabilizing trajectory τ will reach and leave the region φ infinitely often in an infinite amount of time.

$$\forall t\ \exists t', t'', t'' > t' > t: \quad \tau(t') \in \varphi \wedge \tau(t'') \notin \varphi$$

We will next show that in this case τ must have infinitely many entry-points outside of the region φ, which yields a contradiction to Condition (2).

We assume that from time point t_0 on there will be no more entry-points on τ outside of φ, i.e. $\tau(t_0)$ is the last one. We know that τ will reach the region φ after t_0 again, say at t_1, and it will leave φ again after t_1, say at t_2. The flow in each location of A is monotonic (since A is a linear system with one single continuous variable). Since no more entry-point will occur outside of φ the trajectory τ will move away from φ forever after t_2. But this is a contradiction to the assumption that τ reaches and leaves φ infinitely often in an infinite amount of time.

We refer the only if-direction of the proof to Theorem 1. □

In the sample trajectory (Fig. 4.3) of our former example, the sequence of entry-points

$$\tau(1.9) = (\ell_1, x = 0), \tau(2.9) = (\ell_1, x = 0), \tau(3.9) = (\ell_1, x = 0), \ldots$$

is for instance not finite. Hence this system cannot be stable wrt. φ.

5 Stability Criterion for Nonlinear, One-dimensional Hybrid Systems

Having found a criterion for region stability of linear hybrid systems we will next consider nonlinear systems. Again we first consider systems with one location and one continuous variable before we consider systems with several locations.

5.1 Nonlinear, One-dimensional, One Location

We consider a hybrid system where the flow condition is given by

$$\dot{x} = \cos(x)$$

and the region φ is given by

$$\varphi \quad \equiv \quad x \in [-0.5, 1.1] \,.$$

A sample trajectory of this system is given below.

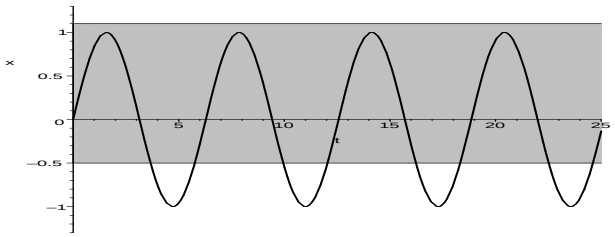

Figure 5.1: Sample trajectory of a hybrid system with one location that satisfies Condition 1. The trajectory violates stability wrt. the grey region $\varphi \equiv x \in [-0.5, 1.1]$.

Trajectories of nonlinear system can oscillate, i.e. they can change their directions during continuous flows, whereas a trajectory of a linear system can change

its direction only by taking a discrete jump (as long as the systems have only one continuous variable). We call states where the trajectory changes its direction *extremal-points*.

Definition 5 (Extremal-points) *An extremal-point s wrt. a continuous variable $x \in \mathcal{V}$ is a state on a trajectory τ at time t,*

$$s = (\ell, \nu) = \tau(t) \ ,$$

such that the x-component of the first derivative $\dot{\nu}$ changes its sign at time t.

$$\lim_{'t \to t_-} (sgn(\tau('t))_x) \quad \neq \quad \lim_{t' \to t_+} (sgn(\tau(t'))_x)$$

By Extrem_x *we denote the set of all extremal-points wrt. x.*

We shortly write

$$(\tau(t))_x$$

for the x-component of the first derivative $\dot{\nu}$ at time t. The signum function *sgn* assigns to each real number r its sign.

$$sgn(r) \quad = \quad \begin{cases} -1 & \text{if} \quad r < 0 \\ 0 & \text{if} \quad r = 0 \\ +1 & \text{if} \quad r > 0 \end{cases}$$

We must adapt Condition 1 to the more general case where extremal-points wrt. x can occur during flows.

Condition 1: There is no infinite sequence of snapshots such that
(i) all states of the sequence lie on the same flow and
(ii) no extremal-point wrt. x lies between two states of the sequence.

In the linear case the condition "no extremal-point wrt. x lies on the sequence or between two states of the sequence" is always true for all sequences of snapshots that lie on the same flow (because no extremal-points wrt. x do exist during continuous flows of one-dimensional linear hybrid systems). Hence we can use the new form of Condition 1 in Section 4 without changing the results that we have archived there.

Since the argumentation in the linear case (Section 4.1) is based on the monotonicity of the trajectories we can suspect that Condition 1 will not be sufficient for stability in the nonlinear case. Indeed, if we choose for the system above the time distance δ small enough (e.g. $\delta = 1$) then Condition 1 holds but the system is not stable wrt. φ (see Fig. 5.1).

This means that, to find a stabilization criterion for nonlinear systems, we additionally have to consider sequences of extremal-points wrt. x.

Condition 3: There is no infinite sequence of snapshots such that all states of the sequence are extremal-points wrt. x.

The idea is as follows: We assume that, for all trajectories τ, all sequences of extremal-points wrt. x are finite outside of φ (i.e. there are only finitely many extremal-points wrt. x that are not in φ). This can be either because only finitely many extremal-points wrt. x do exist; or there are infinitely many extremal-points wrt. x and from a certain time point on all of them are in the region φ. In the second case we are done with the stability proof because with two extremal-points wrt. x all states of the trajectory in between are also in φ (because φ is an interval region). In the first case we must additionally consider all computations after the last extremal-point wrt. x. But if no more extremal-point wrt. x occurs the trajectory is monotonic and we can reduce stability to Condition 1.

Lemma 3 *Condition 1 and Condition 3 together are sufficient and necessary for region stability of a nonlinear hybrid system A with one location ℓ and one continuous variable x wrt. an interval region φ.*

Proof: The if-direction is clear from the above. We refer the only if-direction to Theorem 1. □

5.2 Nonlinear, One-dimensional, m Locations

Comparing the two examples from Section 4.2 and 5.1 (see Fig. 4.3 and Fig. 5.1) there is a significant difference: the reason *why* the trajectories are not monotone, namely because of the discrete jumps in the linear case and because of the oscillations in the nonlinear case. One could ask why we distinguish between entry- and extremal-points wrt. x although we could also have formulated Lemma 2 using Condition 3 instead of Condition 2.

The reason becomes clear if we look at an example of a nonlinear hybrid systems with more than one location, see Fig 5.2.

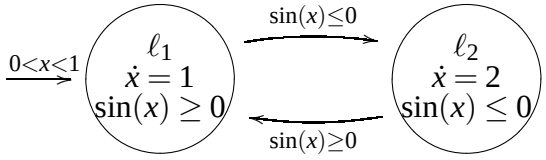

Figure 5.2: Example of a nonlinear system with 2 locations. All trajectories are strictly monotonic increasing towards infinity. Hence the system is not stable wrt. $\varphi \equiv x \in [0,50]$.

We want to know whether this hybrid system is stable wrt. the region

$$\varphi \quad \equiv \quad x \in [0,50] \,.$$

The system above does not have any extremal-points wrt. x since the derivative of x is always greater than 0; this means that all sequences of extremal-points wrt. x are finite and hence Condition 3 holds. Condition 1 also holds because of the invariant conditions of the locations. But the system is obviously not stable wrt. the region φ since all trajectories are strictly monotonically increasing towards infinity.

This example shows that we can not replace Condition 2 by Condition 3, but we need to consider both sequences of entry-points *and* sequences of extremal-points.

Lemma 4 *Condition 1, Condition 2 and Condition 3 together are sufficient and necessary for region stability of a nonlinear hybrid system A with m locations and one continuous variable x wrt. an interval region* φ.

Proof: (Sketch of the if-direction) We assume that the Conditions 1-3 hold but the system has a non-stabilizing trajectory τ.

By Condition 2 all sequences of snapshots that consist of entry-points are finite outside of φ. Because τ is non-stabilizing there must either be infinitely many oscillations in an infinite amount of time, i.e. infinitely many extremal-points wrt. x, that bring the trajectory out of the region again, a contradiction to Condition 3.

Or, if all sequences of snapshots that consist of extremal-points wrt. x are also finite, the computation of the system must end up in a flow outside of φ without extremal-points in between. But this yields a contradiction to Condition 1. □

6 The general case: A Stability Criterion for Linear or Nonlinear, n-dimensional Hybrid Systems with m Locations

In a final step we give the complete criterion for stability of a general (linear or nonlinear) hybrid system wrt. an interval region φ in terms of three conditions. We will prove that the three conditions together are not only sufficient but also necessary for region stability.

Theorem 1 *Condition 1, Condition 2 and Condition 3 together are sufficient and necessary for region stability of a hybrid system A wrt. an interval region* φ*.*

Proof: It follows from Lemma 4 that the Conditions 1-3 are sufficient for stability since the proof of Lemma 4 does not make use of the fact that x is the only continuous variable.

It remains to show that the Conditions 1-3 are necessary for stability. Therefor we prove that "stability with respect to φ" implies that *all possible sequences of snapshots are finite*. From this implication it follows immediately that the Conditions 1-3 must hold for stable systems.

By definition, stability wrt. a region φ means

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

We consider a sequence of snapshots

$$\tau(t_1), \tau(t_2), \tau(t_3), \ldots \, .$$

For a proof by contradiction we assume that this sequence is infinite. Since the system A is stable there must exist a time point t_0 such that all states on

A : hybrid system
φ : interval region

Condition 1 There is no infinite sequence of snapshots such that
(i) all states of the sequence lie on the same flow and
(ii) no extremal-point wrt. x lies between two states of the sequence.

Condition 2 There is no infinite sequence of snapshots such that
all states of the sequence are entry-points.

Condition 3 There is no infinite sequence of snapshots such that
all states of the sequence are extremal-points wrt. x.

Region stability of A with respect to φ

Figure 6.1: Proof rule for region stability of hybrid systems wrt. interval regions. A sequence of snapshots is a sequence of states such that (i) all states of the sequence lie on the same trajectory τ of A, (ii) all states are not in the region φ and (iii) all pairs of consecutive states have a minimum time distance δ, where δ is an arbitrary but fixed constant greater than 0. The notion is formally defined in Definition 3.

the trajectory τ are in φ after t_0. We look at the state $\tau(t_k)$ of the sequence above where k is given by

$$k = 1 + \lceil t_0/\delta \rceil .$$

Since δ is a positive constant the index k is positive and finite. It follows for the time point t_k:

$$\begin{aligned} t_k &\geq k \cdot \delta \\ &\geq t_0 + \delta . \end{aligned}$$

But this implies that $\tau(t_k)$ is in the region φ, a contradiction. □

In Fig. 6.1 we give a proof rule for region stability of hybrid systems. By Theorem 1 the three conditions of the proof rule together are both sufficient and necessary for region stability.

7 Stability with Respect to Multi-dimensional Regions

We can extend all our previous results to n-dimensional regions, i.e. to regions φ that can be expressed as cartesian products of intervals.

$$\varphi \quad \equiv \quad (x_1,\ldots,x_n) \ \in \ [x_1^{min},x_1^{max}] \times \ldots \times [x_n^{min},x_n^{max}] \ ,$$

We call such regions **box regions**. The idea is to prove that the hybrid system is stable wrt. each single one-dimensional component φ_i of the cartesian product,

$$\varphi_i \quad \equiv \quad x_i \in [x_i^{min},x_i^{max}] \ , \qquad i = 1 \ldots n \ .$$

The following lemma shows that this yields stability wrt. the n-dimensional region φ.

Lemma 5 *A hybrid system A is stable wrt. an n-dimensional box region φ if and only if A is stable wrt. each interval region φ_i.*

Proof: **if**-direction: Assume A is stable wrt. to each one-dimensional region φ_i, formally

$$\begin{aligned} \forall \tau \, \exists t_0^1 \, \forall t \geq t_0^1 \quad &: \quad \tau(t) \in \varphi_1 \ , \\ &\ldots \\ \forall \tau \, \exists t_0^n \, \forall t \geq t_0^n \quad &: \quad \tau(t) \in \varphi_n \end{aligned}$$

We take the maximum over all time points t_0^i and call it t_0.

$$t_0 \ = \ \max_{i=1\ldots n} t_0^i$$

This means that from time point t_0 on the trajectory τ is in all one-dimensional regions φ_i,

$$x_1 \in [x_1^{min},x_1^{max}] \ \wedge \ \ldots \ \wedge \ x_n \in [x_n^{min},x_n^{max}] \ ,$$

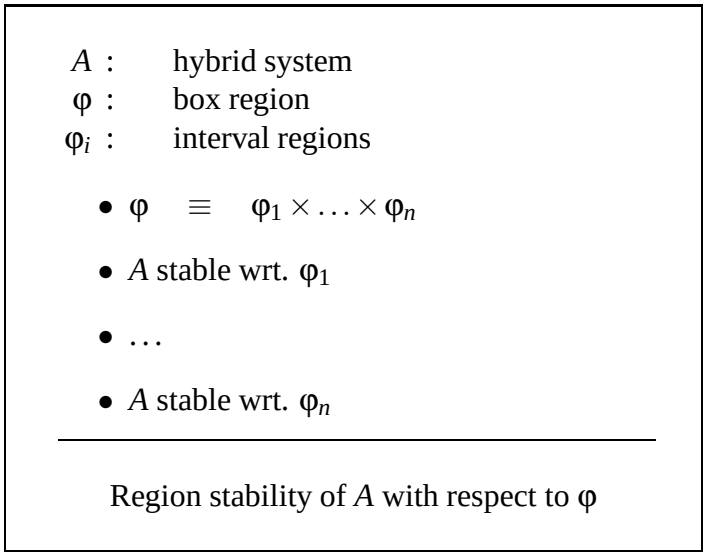

Figure 7.1: Proof rule for region stability of hybrid systems wrt. box regions.

and hence in the n-dimensional box region φ which implies that A is stable wrt. φ.

only if-direction: Assume A is stable wrt. n-dimensional box region φ, i.e.

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

Especially this means that from t_0 on the trajectory τ is in all one-dimensional interval regions φ_i.

$$\forall \tau \, \exists t_0 \, \forall t \geq t_0 \; : \; \tau(t) \in \varphi_1 \wedge \ldots \; \tau(t) \in \varphi_n \; .$$

Thus A is stable wrt. each interval region φ_i. □

The proof rule for stability of hybrid systems wrt. box regions is given below. The conditions of the proof rule together are sufficient and necessary for region stability.

8 Implementation

The goal of this section is to show how one can compute an effective representation of the set of sequences of snapshots for each kind of snapshot. In order to check finiteness of sequences of snapshots we give an implicit representation of the set of these sequences by constraints that denote binary relations. (A binary relation represents the set of all sequences such that each pair of consecutive elements lies in the relation).

We distinguish three different kinds of snapshots; for each kind of snapshot we compute the constraint that implicitly represents the corresponding set of sequences. For each of the three cases the algorithm to compute the constraints is based on the syntactic transformation of the original hybrid system into another one such that the reachability relation of the transformed hybrid system is the binary reachability relation between snapshots of the original system; an overapproximation of the unary reachability relation of the transformed hybrid system is computed by using dedicated abstraction techniques that have been developed for safety proofs of hybrid systems (see e.g. [18]).

Fig. 8.1 shows the control flow graph of the overall algorithm. The input is a hybrid system and a region φ. In the first step the algorithm computes a transformed hybrid system such that the reachability relation of the transformed hybrid system is the binary reachability relation between snapshots of the original system. In the second step the algorithm uses PHAVer, a tool for (unary) reachability analysis, to compute the reachability relation of the transformed hybrid system. The last step is to check whether all relations in the output of PHAVer are well-founded. (*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 rela-

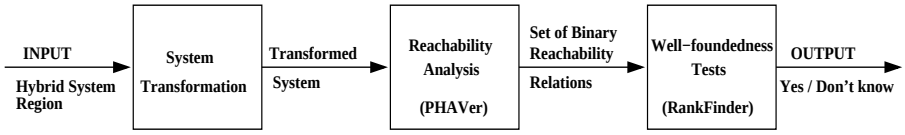

Figure 8.1: Control flow graph of the overall algorithm.

tion.) Our algorithm uses RankFinder ([36]) for the well-foundedness tests. The output of the algorithm is a Yes / Don't know answer.

8.1 System transformation

To find an effective representation of the set of sequences of snapshots for each kind of snapshot, we first consider the representation that is directly given by the original hybrid system. Approaches to verification methods for stability that are based on this representation use Lyapunov theory. But this representation suffers from the non-applicability of suitable abstraction techniques.

Therefore we use an implicit representation of the set of sequences of snapshots by constraints that denote binary relations (such that all sequences are ordered by the relations). That means a binary relation represents the set of all sequences of snapshots $s_0, s_1, s_2, \ldots$ such that all pairs of snapshots (s_i, s_{i+1}) lie in the relation.

We will now show by means of examples how one can compute the constraints in each of the three cases. Afterwards we will give the formal account of the system transformation.

8.1.1 Sequences of snapshots on the same flow of a trajectory.

First we describe how one can compute the representation for all sequences of snapshots on the same flow of a trajectory without extremal-points in between.

We consider the hybrid system with one location and one continuous variable in Fig. 8.2. We want to know whether the system is stable wrt. the region $\varphi \equiv x \in [-1,1]$.

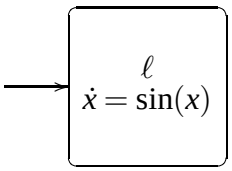

Figure 8.2: Hybrid system with one location and one continuous variable

Fig. 8.3 shows the system transformation for the computation of sequences of snapshots on monotonically *increasing* parts of a trajectory. (The transformed system for the *de*creasing parts is quite similar; it differs from this system only in the guards of the locations ℓ^0 and ℓ^1 that are given by $\sin(x) < 0$ and $\sin(x') < 0$ respectively.)

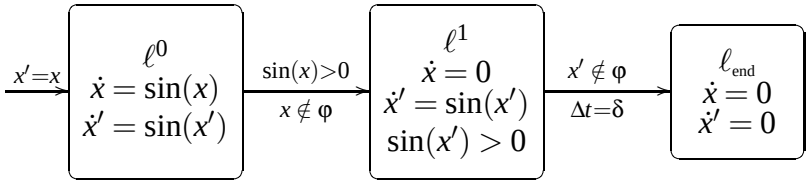

Figure 8.3: Transformed system for the computation of sequences of snapshots on monotonic parts of a trajectory.

The transformed system has two continuous variables, namely the original variable x and its copy x'. Initially the values of x and x' are the same. In the location ℓ^0 the continuous flows of the two variables are identical, each of which corresponding to the flow of the original system. This means that one can view a state (s,s') of the transformed system in ℓ^0 as a pair of identical states of the original system. The guard of the location ℓ^0 means that the values of x (and thus also the values of x') are monotonically increasing since the first derivative $\dot{x} = \sin(x)$ must be > 0.

The transformed system can jump from the location ℓ^0 to the location ℓ^1 only if the value of x is not in the region φ (i.e. either if $x > 1$ or if $x < -1$). Until the discrete jump the values of x and x' are identical. After the jump only x' continues evolving as before whereas x is fixed from now on (since in the location ℓ^1 the flow of x is constantly 0). The values of x' are still increasing according to the guard of ℓ^1. This means that in ℓ^1 one can view a state (s,s') of the transformed system as a pair of states on a monotonically increasing part of a trajectory of the original system, where the first state s is not in the region φ.

To make sure that the value of x' is also not in φ the transformed system can jump to the third location ℓ_{end} only if $x' > 1$ or if $x' < -1$. Additionally the jump condition ensures that the transformed system must take time δ in the location ℓ^1. In ℓ_{end} the values of both variables x and x' are fixed. A state (s,s') of the transformed system in ℓ_{end} can be viewed as a pair of states on a monotonically increasing part of a trajectory of the original system where both states are not in φ and have a time distance δ. (The constant $\delta > 0$ is the discretization width of the trajectory.)

Altogether this means that the reachability relation of the transformed hybrid system in ℓ_{end} is the binary reachability relation between snapshots of the original system that lie on the same flow of a trajectory without extremal-points in between.

8.1.2 Sequences of extremal points

Next we describe the system transformation for the computation of sequences of extremal-points. We consider again the system in Fig. 8.2 and the transformed system in Fig. 8.4.

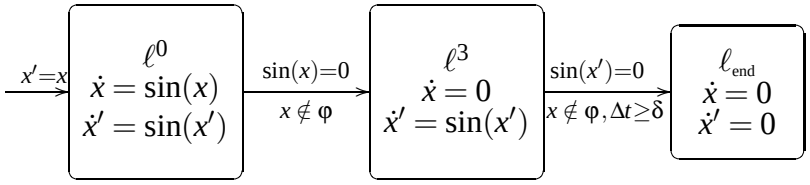

Figure 8.4: Transformed system for the computation of sequences of extremal-points.

The basic idea for the system transformation is the same as before: we make a copy x' of the continuous variable x, we eventually fix the value of x and let x' continue evolving, and then we also fix the value of x'. But there are two differences to the transformed system in Fig. 8.3.

First, the guards of the locations of the transformed system in Fig. 8.4 are given by the guard of the original system (that is true in this example). Second, we must add the condition $\sin(x) = 0$ to both jump conditions (from ℓ^0 to ℓ^3 and from ℓ^3 to ℓ_{end}) to ensure that a state (s,s') of the transformed system in ℓ_{end} corresponds to a pair of extremal-points (that are states where the first derivative of x is 0) of the original system. Again both states lie outside of the region φ.

8.1.3 Sequences of entry-points

Now we consider the simple heating system in Fig. 3.2 to describe the system transformation for the computation of sequences of entry-points. The region φ is given by $x \in [65,82]$. For simplicity we only consider entry-points of the location ℓ_1. (An entry-point of ℓ_1 is a state just after a discrete jump from ℓ_2 to ℓ_1, i.e. the jump condition $x \geq 80$ must hold.)

In this case it does not suffice to make a copy of the continuous variable x only but we must duplicate the whole hybrid system, see Fig. 8.5.

The intuition is that a state (s,s') of the transformed system in ℓ_{end} corresponds to a pair of entry-points of the location ℓ_1 of the original system. To ensure that s is an entry-point of ℓ_1 (of the original system) we fix the value of x just after a discrete jump (of the transformed system) from ℓ_2^0 to ℓ_1^4. The jump condition $x > 82$ ensures both that the jump condition from ℓ_2^0 to ℓ_1^0 (i.e. $x \geq 80$) holds and that x is not in the region φ (i.e. $(x < 65 \vee x > 82)$). For the same reason the

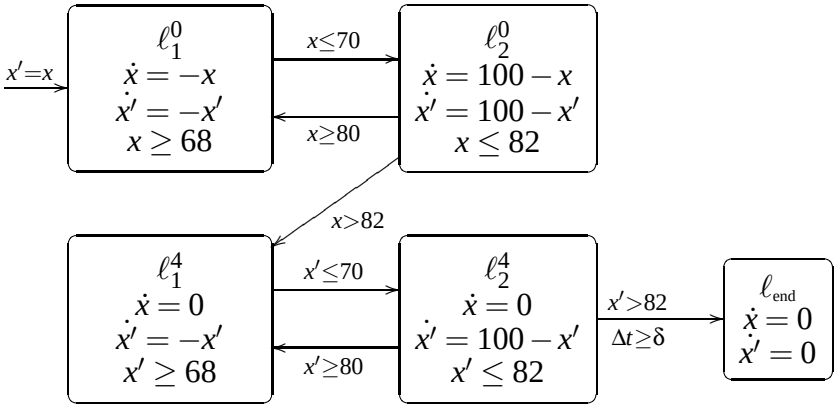

Figure 8.5: Transformed system for the computation of sequences of entry-points.

jump condition from ℓ_2^4 to ℓ_{end} ensures that s' is an entry-point of the location ℓ_1 of the original system that does not lie in φ.

8.1.4 Formal Description of the System Transformation

Given a hybrid system

$$A = \big(\mathcal{L}, \mathcal{V}, (\mathsf{flow}_\ell), (\mathsf{jump}_{\ell,\ell'}), (\mathsf{inv}_\ell), (\mathsf{init}_\ell)\big)$$

for which we want to prove stability wrt. the interval region

$$\varphi \quad \equiv \quad x \in [x_{min}, x_{max}] \, ;$$

the transformed system A^T is given by:

1. Locations: Each location ℓ_i of the original system corresponds to five locations $\ell_i^0, \ldots, \ell_i^4$ in the transformed system. We refer to the set of all locations from ℓ_1^k to ℓ_m^k as $\mathcal{L}^k$,

   $$\mathcal{L}^k = \{\ell_1^k, \ldots, \ell_m^k\} \, , \, k \in \{0,1,2,3,4\} \, .$$

   In addition, the transformed system has two locations ℓ_{init} and ℓ_{end}. Altogether, the set $\mathcal{L}^T$ of locations of the transformed system consists of the following components:

   $$\mathcal{L}^T \;=\; \{\ell_{\mathsf{init}}\} \cup \mathcal{L}^0 \cup \mathcal{L}^1 \cup \mathcal{L}^2 \cup \mathcal{L}^3 \cup \mathcal{L}^4 \cup \{\ell_{\mathsf{end}}\}$$

2. Variables: The set $\mathcal{V}^T$ of variables of the transformed system contains all variables of $\mathcal{V}$, their primed versions, and an additional variable called flag.

   $$\mathcal{V}^T \;=\; \mathcal{V} \cup \{\mathsf{flag}\} \cup \mathcal{V}'$$

3. Flow constraints: First, in the locations ℓ_{init} and ℓ_{end} the flow of all continuous variables is 0.

$$\begin{aligned} \mathsf{flow}^T_{\ell_{\mathsf{init}}}(x_1,\ldots,t',\dot{x}_1,\ldots,\dot{t}') &\equiv \bigwedge_{x\in\mathcal{V}^T} \dot{x}=0 \\ \mathsf{flow}^T_{\ell_{\mathsf{end}}}(x_1,\ldots,t',\dot{x}_1,\ldots,\dot{t}') &\equiv \bigwedge_{x\in\mathcal{V}^T} \dot{x}=0 \end{aligned}$$

In each location ℓ^0_i of $\mathcal{L}^0$, the flow of the variables $x_1,\ldots,t$ in the transformed system is the same as the flow of $x_1,\ldots,t$ in the original system; each variable $x'_1,\ldots,t'$ behaves exactly like its unprimed version, that is the flow of $x'_1,\ldots,t'$ is equal to the flow of the original system after replacing the variables $x_1,\ldots,t$ by their primed versions $x'_1,\ldots,t'$.

$$\begin{aligned} \mathsf{flow}^T_{\ell^0_i}(x_1,\ldots,t',\dot{x}_1,\ldots,\dot{t}') \equiv\ & \mathsf{flow}_{\ell_i}(x_1,\ldots,\dot{t}) \wedge \dot{\mathsf{flag}}=0 \\ & \wedge \mathsf{flow}_{\ell_i}(x'_1,\ldots,\dot{t}') \end{aligned}$$

In each location of $\mathcal{L}^1\cup\ldots\cup\mathcal{L}^4$ the values of the variables $x_1,\ldots,t$ are fixed, i.e. the flow of them is constantly 0. The variables $x'_1,\ldots,t'$ keep on evolving as before.

$$\begin{aligned} \mathsf{flow}^T_{\ell^k_i}(x_1,\ldots,t',\dot{x}_1,\ldots,\dot{t}') \equiv\ & \Big(\bigwedge_{x\in\mathcal{V}\cup\{\mathsf{flag}\}} \dot{x}=0\Big) \wedge \mathsf{flow}_{\ell_i}(x'_1,\ldots,\dot{t}')\,, \\ & k\in\{1,\ldots,4\} \end{aligned}$$

4. Jump constraints: A jump is possible from the location ℓ_{init} to any location ℓ^0_i of $\mathcal{L}^0$ if the initial condition of the location ℓ_i of the original system A is fulfilled for the variables $(x_1,\ldots,t)$.

$$\mathsf{jump}^T_{\ell_{\mathsf{init}},\ell^0_i}(x_1,\ldots,t') \equiv \mathsf{init}_{\ell_i}(x_1,\ldots,t)$$

The second kind of jumps are jumps between between two locations of $\mathcal{L}^0$ and between two locations of $\mathcal{L}^4$, respectively. The condition for a jump between the location ℓ^0_i and the location ℓ^0_j for the variables $(x_1,\ldots,t,x'_1,\ldots,t')$ corresponds to the jump condition between the locations ℓ_i and ℓ_j of the original system A for the variables $(x_1,\ldots,t)$. Similarly a jump condition between the locations ℓ^4_i and ℓ^4_j 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.

$$\begin{aligned} \mathsf{jump}^T_{\ell^0_i,\ell^0_j}(x_1,\ldots,t') &\equiv \mathsf{jump}_{\ell_i,\ell_j}(x_1,\ldots,t) \\ \mathsf{jump}^T_{\ell^4_i,\ell^4_j}(x_1,\ldots,t') &\equiv \mathsf{jump}_{\ell_i,\ell_j}(x'_1,\ldots,t') \end{aligned}$$

To compute sequences on monotonic flows we allow jumps from $\ell_i^0 \in \mathcal{L}^0$ to $\ell_i^1 \in \mathcal{L}^1$ (and to $\ell_i^2 \in \mathcal{L}^2$, respectively) if the first derivative of x is greater than 0 (and less than 0, respectively) and if x does not lie in φ.

$$\begin{aligned} \mathsf{jump}^T_{\ell_i^0,\ell_i^1}(x_1,\ldots,t') &\equiv \dot{x} > 0 \wedge \neg\varphi \\ \mathsf{jump}^T_{\ell_i^0,\ell_i^2}(x_1,\ldots,t') &\equiv \dot{x} < 0 \wedge \neg\varphi \end{aligned}$$

Similarly the transformed system can take a jump from a location $\ell_i^0 \in \mathcal{L}^0$ to a location $\ell_i^3 \in \mathcal{L}^3$ if the first derivative of x is 0 and if x does not lie in φ.

$$\mathsf{jump}^T_{\ell_i^0,\ell_i^3}(x_1,\ldots,t') \equiv \dot{x} = 0 \wedge \neg\varphi$$

A jump from ℓ_i^1 to ℓ_{end} (and from ℓ_i^2 to ℓ_{end}, respectively) is possible if the first derivative of x' is greater than 0 (and less than 0, respectively), if x' does not lie in φ, and if if the system has spend time δ in the location ℓ_i^1 (and in ℓ_i^2, respectively).

$$\begin{aligned} \mathsf{jump}^T_{\ell_i^1,\ell_{\mathsf{end}}}(x_1,\ldots,t') &\equiv \dot{x}' > 0 \wedge \neg\varphi \wedge t' - t = \delta \\ \mathsf{jump}^T_{\ell_i^2,\ell_{\mathsf{end}}}(x_1,\ldots,t') &\equiv \dot{x}' < 0 \wedge \neg\varphi \wedge t' - t = \delta \end{aligned}$$

Similarly the transformed system can jump from from ℓ_i^3 to ℓ_{end} if the system has spend time δ in the location ℓ_i^3, if the first derivative of x' is 0, and if x' is not in φ.

$$\mathsf{jump}^T_{\ell_i^3,\ell_{\mathsf{end}}}(x_1,\ldots,t') \equiv \dot{x}' = 0 \wedge \neg\varphi \wedge t' - t \geq \delta$$

For the computation of pairs of entry-points of the original system we need a jump from a location $\ell_i^0 \in \mathcal{L}^0$ to a location $\ell_j^4 \in \mathcal{L}^4$ whenever the jump condition between the locations ℓ_i and ℓ_j of the original system A is possible but only if x is not in the region φ. During the jump the value of flag is set to the index j of the target location.

$$\mathsf{jump}^T_{\ell_i^0,\ell_j^4}(x_1,\ldots,t') \equiv \mathsf{jump}_{\ell_i,\ell_j}(x_1,\ldots,t) \wedge \neg\varphi \wedge \mathsf{flag} := j$$

The transformed system can jump from $\ell_i^4 \in \mathcal{L}^4$ to ℓ_{end} if x is not in φ, the value of flag is j, and the jump condition from ℓ_i to ℓ_j of the original system A holds for the primed variables. Additionally we must guarantee the discretization width δ (for $\delta > 0$ constant).

$$\begin{aligned} \mathsf{jump}^T_{\ell_i^4,\ell_{\mathsf{end}}}(x_1,\ldots,t') \equiv \bigvee_{\ell_j \in \mathcal{L}} \Big(& \mathsf{jump}_{\ell_i,\ell_j}(x'_1,\ldots,t') \wedge \neg\varphi \\ & \wedge \mathsf{flag} = j \wedge t' - t \geq \delta \Big) \end{aligned}$$

5. Invariant conditions: For the locations ℓ_{init} and ℓ_{end} the invariant condition is true.

$$\begin{aligned} \mathsf{inv}^T_{\ell_{\mathsf{init}}}(x_1,\ldots,t') &\equiv \text{true} \\ \mathsf{inv}^T_{\ell_{\mathsf{end}}}(x_1,\ldots,t') &\equiv \text{true} \end{aligned}$$

For a location ℓ_i^0 in $\mathcal{L}^0$ the invariant condition over $x_1,\ldots,t,x_1',\ldots,t'$ is the same as the invariant condition of the original system A for ℓ_i over $x_1,\ldots,t$.

$$\mathsf{inv}^T_{\ell_i^0}(x_1,\ldots,t') \equiv \mathsf{inv}_{\ell_i}(x_1,\ldots,t)$$

For a location $\ell_i^1 \in \mathcal{L}^1$ (and $\ell_i^2 \in \mathcal{L}^2$, respectively) the invariant condition over $x_1,\ldots,t,x_1',\ldots,t'$ corresponds to the invariant condition of the original system A for ℓ_i over $x_1',\ldots,t'$ in addition to the condition $\dot{x} > 0$ (and $\dot{x} < 0$, respectively).

$$\begin{aligned} \mathsf{inv}^T_{\ell^1}(x_1,\ldots,t') &\equiv \mathsf{inv}_{\ell}(x_1',\ldots,t') \wedge \dot{x} > 0 \\ \mathsf{inv}^T_{\ell^2}(x_1,\ldots,t') &\equiv \mathsf{inv}_{\ell}(x_1',\ldots,t') \wedge \dot{x} < 0 \end{aligned}$$

For a location $\ell_i^3 \in \mathcal{L}^3$ or $\ell_i^4 \in \mathcal{L}^4$ the invariant condition over $x_1,\ldots,t,x_1',\ldots,t'$ is the same as the invariant condition of the original system A for ℓ_i over $x_1',\ldots,t'$

$$\begin{aligned} \mathsf{inv}^T_{\ell_i^3}(x_1,\ldots,t') &\equiv \mathsf{inv}_{\ell_i}(x_1',\ldots,t') \\ \mathsf{inv}^T_{\ell_i^4}(x_1,\ldots,t') &\equiv \mathsf{inv}_{\ell_i}(x_1',\ldots,t') \end{aligned}$$

6. Initial conditions: Initially, each variable x_i has the same value as x_i', the value of t is equal to the value of t', and the value of flag is set to 0; the system starts at time point $t = 0$ in the location ℓ_{init}.

$$\begin{aligned} \mathsf{init}^T_{\ell_{\mathsf{init}}} &\equiv \bigwedge_{x \in \mathcal{V}} x = x' \wedge t = 0 \wedge \mathsf{flag} = 0 \\ \mathsf{init}^T_{\ell} &\equiv \text{false} \quad \forall\, \ell \neq \ell_{\mathsf{init}} \end{aligned}$$

8.2 Reachability Analysis and Well-foundedness Tests

Now we are given a transformed hybrid system such that the reachability relation of the transformed hybrid system is the binary reachability relation between snapshots of the original system. We use PHAVer [18] for the reachability analysis of

the transformed system. The output of PHAVer is a set of constraints, given by a disjunction of conjunctions of linear inequalities. The constraints denote binary relations of the original hybrid system.

We check for each single relation that it is well-founded. For the well-foundedness tests we use RankFinder [36]. If every relation is well-founded then all sequences of snapshots (for each kind of snapshot) are finite outside of the region φ, which means that the original hybrid system is stable wrt. the region φ.

8.3 Experimental Comparisons

In an old version of our tool (see [33]) we check that a given region φ is a "strong attractor" of a given hybrid system which implies that the hybrid system is stable wrt. φ. However, strong attraction is a sufficient but not a necessary condition for stability. We compare the feasibility and the run times of the old version of our tool with the new version that is described in this paper. The new version is based on a new characterization of region stability (described in Sections 4-6) that is both sufficient and necessary.

System	Old version [33]	New version
Example 1	0.234s	0.219s
Simple heater	0.202s	0.575s
Complex heater	stability not known	2.206s
One tank system	0.428s	1.915s
Two tank system	stability not known	20.602s
Distance controller	0.384s	0.692s
Bouncing ball	2.821s	4.843s
Train brake	stability not known	4.669s

Figure 8.6: Run time comparison between an old and the new version of our tool.

The new version of our tool is compatible with the current version of PHAVer (v. 0.37) (that we use for the reachability analysis) whereas we have to use an older version of PHAVer (v. 0.2.7) with the old version of the tool [33]. The following table gives information about the run times when we use the new version of our tool with the different versions of PHAVer.

System	PHAVer 0.2.7	PHAVer 0.37
Example 1	0.219s	0.191s
Simple heater	0.575s	0.490s
Complex heater	2.206s	1.920s
One tank system	1.915s	1.813s
Two tank system	20.602s	16.545s
Distance controller	0.692s	1.186s
Bouncing ball	4.843s	4.209s
Train brake	4.669s	2.589s

Figure 8.7: Run time comparison between PHAVer versions 0.2.7 and 0.37.

9 Related Work

Proof rules for many well-known notions of stability (e.g. asymptotical or exponential stability) are based on Lyapunov functions. There is a large body of work on Lyapunov theory, especially in the community of control theory [3, 4, 26, 27, 28]. However the synthesis of Lyapunov functions is viewed as a hard problem for which manual intervention is necessary. Until now there has been no high expectation in automation.

Tools that synthesize Lyapunov functions are successful only in application of hybrid systems with a small number of locations [5, 6]. Such methods suffer from the fact that no appropriate abstraction technique is known for the synthesis of Lyapunov functions. The reason why this is the case is essentially that we know very good overapproximation techniques but no underapproximation techniques.

We can classify existing tools for verifying region stability into two classes: (1) tools that abstract hybrid systems into finite state models and apply model checkers to this abstraction [16]; and (2) tools that use unary reachability to check that after a given time point t all states outside of the stable region are unreachable [19]. The first class suffers from the fact that abstraction to finite state models is intrinsically inappropriate for systems with finite but unboundedly long trajectories outside of the region (in a finite state system where all executions have finite length, the executions must have bounded length); the second class also only works if all computations have bounded length outside of the region. They fail to prove stability e.g. for the first example in Section 3 (a system with one location where the flow condition is given by $\dot{x} = -1$ and the region φ is given by $x \leq 0$).

In [33] the authors give a method for automatically verifying region stability by checking that the region is a "strong attractor" of the hybrid system. This method is sound but not complete, i.e. it works well for hybrid systems whose trajectories never leave the stable region after they have reached it *once*. But for more general cases (e.g. Example 3 or Example 5 in Section 3) the method fails to prove stability.

10 Conclusion

We have given a model checking method and tool that integrates state abstraction techniques for the automatic proof of a stability property for hybrid systems, to our knowledge for the first time. It is based on a new notion of snapshots which yield characteristic discretizations of trajectories. We have described the experiments with a prototypical implementation of our tool. The experiments yield fully automatic stability proofs for a number of typical benchmark problems including the the break curve behavior of a train system, a previously open challenge problem for the AVACS project (www.avacs.org).

In the present version, the implementation of our tool applies a linear constraint solver [36] to check the well-foundedness of the binary relation between snapshots; this is appropriate since the PHAVer tool computes an overapproximation of the binary relation in the form of linear constraints. For future work, we plan to investigate the usefulness of well-foundedness checks by non-linear constraint solvers [11] in a different version of our tool.

Bibliography

[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 HSCC'93.

[2] R. Alur, T. Dang, and F. Ivancic. Reachability Analysis of Hybrid Systems via Predicate Abstraction. In ACM transactions on embedded computing systems, 2004.

[3] M.S. Branicky. Stability of Hybrid Systems: State of the Art. In CDC'97.

[4] M.S. Branicky. Multiple Lyapunov Functions and Other Analysis Tools for Switched and Hybrid Systems. In Trans. on Automatic Control, 1998.

[5] H. Burchardt, J. Oehlerking, and O. Theel. The Role of State-space Partitioning in Automated Verification of Affine Hybrid System Stability. In CCCT'05.

[6] H. Burchardt, J. Oehlerking, and O. Theel. Towards Push-of-a-Button Stability Verification for Discrete-Time Hybrid Systems. In PRDC'05.

[7] A.R. Bradley, H. Sipma, and Z. Manna, Termination of Polynomial Programs. In VMCAI'05.

[8] P. Cousot. Proving Program Invariance and Termination by Parametric Abstraction, Lagrangian Relaxation and Semidefinite Programming. In VMCAI'05.

[9] P. Cousot, and R. Cousot. Abstract Interpretation: a Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. In POPL'77.

[10] P. Cousot, and R. Cousot. Abstract Interpretation Frameworks. In Journal of Logic and Computation, 1992.

[11] P. Cousot. Proving Program Invariance and Termination by Parametric Abstraction, Lagrangian Relaxation and Semidefinite Programming. In VMCAI'05.

[12] B. Cook, A. Podelski, and A. Rybalchenko. Abstraction Refinement for Termination. In SAS'05.

[13] B. Cook, A. Podelski, and A. Rybalchenko. Termination Proofs for Systems Code. In PLDI'06.

[14] M. Colon, and H. Sipma. Synthesis of Linear Ranking Functions. In TACAS'01.

[15] M. Colon, and H. Sipma. Practical Methods for Proving Program Termination. In CAV'02.

[16] W. Damm, G. Pinto, and S. Ratschan. Guaranteed Termination in the Verification of LTL Properties of Non-linear Robust Discrete Time Hybrid Systems. In ATVA'05.

[17] F. Delmotte, M. Egerstedt, and E. Verriest. Hybrid Function Approximation: A Variational Approach. In CDC'05.

[18] G. Frehse. PHAVer , http://www.cs.ru.nl/˜goranf.

[19] G. Frehse, B.H. Krogh, R.A. Rutenbar. Verifying Analog Oscillator Circuits Using Forward/Backward Abstraction Refinement. In DATE'06.

[20] Z.-S. Feng, Y.-Q. Liu, F.-W. Guo. Criteria for Practical Stability in the pth Mean of Nonlinear Stochastic Systems. In Applied Mathematics and Computation, 1992.

[21] Z.-S. Feng. Lyapunov Stability and Practical Stability of Nonlinear Delaystochastic Systems: a Unified Approach. In IEEE Conference on Decision and Control, 1993.

[22] A. Gotsman, B. Cook, A. Podelski, A. Rybalchenko, and M. Vardi. Proving that Software Eventually Does Something Good. In POPL'07.

[23] T.A. Henzinger. The Theory of Hybrid Automata. In Logic in Computer Science, 1996.

[24] T.A. Henzinger, P.-H. Ho, and H. Wong-Toi. Algorithmic Analysis of Nonlinear Hybrid Systems. In Automatic Control, 1998.

[25] M. Kloetzer, and C. Belta. Reachability Analysis of Multi-affine Systems. In HSCC'06.

[26] D. Liberzon. Switching in Systems and Control. Birkhäuser, 2003.

[27] D. Liberzon, J.P. Hespanha, and A.S. Morse. Stability of Switched Systems: a Lie-algebraic Condition. In Systems and Control Letters, 1999.

[28] V. Lakshmikantham, S. Leela, and A.A. Martynyuk. Practical Stability of Nonliear Systems. World Scientific Pub Co Inc, 1990.

[29] S. Pettersson. Analysis and Design of Hybrid Systems. Ph.D. Thesis, Chalmers University of Technology, Göteborg, Sweden, 1999.

[30] A. Podelski, and A. Rybalchenko. Transition Invariants. In LICS'04.

[31] A. Podelski, and A. Rybalchenko. A Complete Method for the Synthesis of Linear Ranking Functions. In VMCAI'04.

[32] A. Podelski, and A. Rybalchenko. Transition Predicate Abstraction and Fair Termination. In POPL'05.

[33] A. Podelski, and S. Wagner. Model Checking of Hybrid Systems: From Reachability towards Stability. In HSCC'06.

[34] S. Ratschan, and Z. She. Safety Verification of Hybrid Systems by Constraint Propagation Based Abstraction Refinement. In HSCC'05.

[35] S. Ratschan and Z. She Constraints for Continuous Reachability in the Verification of Hybrid Systems. In Proc. International Conference on Artificial Intelligence and Symbolic Computation, 2006.

[36] A. Rybalchenko. RankFinder, http://www.mpi-inf.mpg.de/~rybal/rankfinder.

[37] S. Sankaranarayanan, H. Sipma, and Z. Manna. Constraint-Based Linear-Relations Analysis. In SAS'04.

[38] A. Tiwari. Termination of Linear Programs. In CAV'04.

[39] H. Ye, A.N. Michel, and L. Hou. Stability Analysis of discontinuous Dynamical Systems with Applications. In Proc. International Federation of Automatic Control, 1996.

[40] G. Zhai, and A.N. Michel. On Practical Stability of Switched Systems. In CDC'02.

[41] G. Zhai, and A.N. Michel. Generalized Practical Stability Analysis of Discontinuous Dynamical Systems. In CDC'03.

Below you find a list of the most recent technical reports of the Max-Planck-Institut für Informatik. They are available by anonymous ftp from `ftp.mpi-sb.mpg.de` under the directory `pub/papers/reports`. Most of the reports are also accessible via WWW using the URL `http://www.mpi-sb.mpg.de`. If you have any questions concerning ftp or WWW access, please contact `reports@mpi-sb.mpg.de`. Paper copies (which are not necessarily free of charge) can be ordered either by regular mail or by e-mail at the address below.

Max-Planck-Institut für Informatik
Library
attn. Anja Becker
Stuhlsatzenhausweg 85
66123 Saarbrücken
GERMANY
e-mail: `library@mpi-sb.mpg.de`

MPI-I-2006-5-006	G. Kasnec, F.M. Suchanek, G. Weikum	Yago - A Core of Semantic Knowledge
MPI-I-2006-5-005	R. Angelova, S. Siersdorfer	A Neighborhood-Based Approach for Clustering of Linked Document Collections
MPI-I-2006-5-004	F. Suchanek, G. Ifrim, G. Weikum	Combining Linguistic and Statistical Analysis to Extract Relations from Web Documents
MPI-I-2006-5-003	V. Scholz, M. Magnor	Garment Texture Editing in Monocular Video Sequences based on Color-Coded Printing Patterns
MPI-I-2006-5-002	H. Bast, D. Majumdar, R. Schenkel, M. Theobald, G. Weikum	IO-Top-k: Index-access Optimized Top-k Query Processing
MPI-I-2006-5-001	M. Bender, S. Michel, G. Weikum, P. Triantafi lou	Overlap-Aware Global df Estimation in Distributed Information Retrieval Systems
MPI-I-2006-4-010	A. Belyaev, T. Langer, H. Seidel	Mean Value Coordinates for Arbitrary Spherical Polygons and Polyhedra in R^3
MPI-I-2006-4-009	J. Gall, J. Potthoff, B. Rosenhahn, C. Schnoerr, H. Seidel	Interacting and Annealing Particle Filters: Mathematics and a Recipe for Applications
MPI-I-2006-4-008	I. Albrecht, M. Kipp, M. Neff, H. Seidel	Gesture Modeling and Animation by Imitation
MPI-I-2006-4-007	O. Schall, A. Belyaev, H. Seidel	Feature-preserving Non-local Denoising of Static and Time-varying Range Data
MPI-I-2006-4-006	C. Theobalt, N. Ahmed, H. Lensch, M. Magnor, H. Seidel	Enhanced Dynamic Reflectometry for Relightable Free-Viewpoint Video
MPI-I-2006-4-005	A. Belyaev, H. Seidel, S. Yoshizawa	Skeleton-driven Laplacian Mesh Deformations
MPI-I-2006-4-004	V. Havran, R. Herzog, H. Seidel	On Fast Construction of Spatial Hierarchies for Ray Tracing
MPI-I-2006-4-003	E. de Aguiar, R. Zayer, C. Theobalt, M. Magnor, H. Seidel	A Framework for Natural Animation of Digitized Models
MPI-I-2006-4-002	G. Ziegler, A. Tevs, C. Theobalt, H. Seidel	GPU Point List Generation through Histogram Pyramids
MPI-I-2006-4-001	A. Efremov, R. Mantiuk, K. Myszkowski, H. Seidel	Design and Evaluation of Backward Compatible High Dynamic Range Video Compression
MPI-I-2006-2-001	T. Wies, V. Kuncak, K. Zee, A. Podelski, M. Rinard	On Verifying Complex Properties using Symbolic Shape Analysis
MPI-I-2006-1-007	H. Bast, I. Weber, C.W. Mortensen	Output-Sensitive Autocompletion Search
MPI-I-2006-1-006	M. Kerber	Division-Free Computation of Subresultants Using Bezout Matrices
MPI-I-2006-1-005	A. Eigenwillig, L. Kettner, N. Wolpert	Snap Rounding of Bzier Curves
MPI-I-2006-1-004	S. Funke, S. Laue, R. Naujoks, L. Zvi	Power Assignment Problems in Wireless Communication
MPI-I-2005-5-002	S. Siersdorfer, G. Weikum	Automated Retraining Methods for Document Classification and their Parameter Tuning
MPI-I-2005-4-006	C. Fuchs, M. Goesele, T. Chen, H. Seidel	An Emperical Model for Heterogeneous Translucent Objects
MPI-I-2005-4-005	G. Krawczyk, M. Goesele, H. Seidel	Photometric Calibration of High Dynamic Range Cameras
MPI-I-2005-4-004	C. Theobalt, N. Ahmed, E. De Aguiar, G. Ziegler, H. Lensch, M.A. Magnor, H. Seidel	Joint Motion and Reflectance Capture for Creating Relightable 3D Videos

MPI-I-2005-4-003	T. Langer, A.G. Belyaev, H. Seidel	Analysis and Design of Discrete Normals and Curvatures
MPI-I-2005-4-002	O. Schall, A. Belyaev, H. Seidel	Sparse Meshing of Uncertain and Noisy Surface Scattered Data
MPI-I-2005-4-001	M. Fuchs, V. Blanz, H. Lensch, H. Seidel	Reflectance from Images: A Model-Based Approach for Human Faces
MPI-I-2005-2-003	H.d. Nivelle	Using Resolution as a Decision Procedure
MPI-I-2005-2-002	P. Maier, W. Charatonik, L. Georgieva	Bounded Model Checking of Pointer Programs
MPI-I-2005-2-001	J. Hoffmann, C. Gomes, B. Selman	Bottleneck Behavior in CNF Formulas
MPI-I-2005-1-008	C. Gotsman, K. Kaligosi, K. Mehlhorn, D. Michail, E. Pyrga	Cycle Bases of Graphs and Sampled Manifolds
MPI-I-2005-1-007	I. Katriel, M. Kutz	A Faster Algorithm for Computing a Longest Common Increasing Subsequence
MPI-I-2005-1-003	S. Baswana, K. Telikepalli	Improved Algorithms for All-Pairs Approximate Shortest Paths in Weighted Graphs
MPI-I-2005-1-002	I. Katriel, M. Kutz, M. Skutella	Reachability Substitutes for Planar Digraphs
MPI-I-2005-1-001	D. Michail	Rank-Maximal through Maximum Weight Matchings
MPI-I-2004-NWG3-001	M. Magnor	Axisymmetric Reconstruction and 3D Visualization of Bipolar Planetary Nebulae
MPI-I-2004-NWG1-001	B. Blanchet	Automatic Proof of Strong Secrecy for Security Protocols
MPI-I-2004-5-001	S. Siersdorfer, S. Sizov, G. Weikum	Goal-oriented Methods and Meta Methods for Document Classification and their Parameter Tuning
MPI-I-2004-4-006	K. Dmitriev, V. Havran, H. Seidel	Faster Ray Tracing with SIMD Shaft Culling
MPI-I-2004-4-005	I.P. Ivrissimtzis, W.-. Jeong, S. Lee, Y.a. Lee, H.-. Seidel	Neural Meshes: Surface Reconstruction with a Learning Algorithm
MPI-I-2004-4-004	R. Zayer, C. Rssl, H. Seidel	r-Adaptive Parameterization of Surfaces
MPI-I-2004-4-001	J. Haber, C. Schmitt, M. Koster, H. Seidel	Modeling Hair using a Wisp Hair Model
MPI-I-2004-2-007	S. Wagner	Summaries for While Programs with Recursion
MPI-I-2004-2-002	P. Maier	Intuitionistic LTL and a New Characterization of Safety and Liveness
MPI-I-2004-2-001	H. de Nivelle, Y. Kazakov	Resolution Decision Procedures for the Guarded Fragment with Transitive Guards
MPI-I-2004-1-006	L.S. Chandran, N. Sivadasan	On the Hadwiger's Conjecture for Graph Products
MPI-I-2004-1-005	S. Schmitt, L. Fousse	A comparison of polynomial evaluation schemes
MPI-I-2004-1-004	N. Sivadasan, P. Sanders, M. Skutella	Online Scheduling with Bounded Migration
MPI-I-2004-1-003	I. Katriel	On Algorithms for Online Topological Ordering and Sorting
MPI-I-2004-1-002	P. Sanders, S. Pettie	A Simpler Linear Time 2/3 - epsilon Approximation for Maximum Weight Matching
MPI-I-2004-1-001	N. Beldiceanu, I. Katriel, S. Thiel	Filtering algorithms for the Same and UsedBy constraints
MPI-I-2003-NWG2-002	F. Eisenbrand	Fast integer programming in fixed dimension
MPI-I-2003-NWG2-001	L.S. Chandran, C.R. Subramanian	Girth and Treewidth
MPI-I-2003-4-009	N. Zakaria	FaceSketch: An Interface for Sketching and Coloring Cartoon Faces