Benchmarks for Hybrid Systems Verification*

Ansgar Fehnker[1] and Franjo Ivančić[2]

[1] Carnegie Mellon University,5000 Forbes Ave, Pittsburgh, PA 15213 (USA)
ansgar@cmu.edu
[2] NEC Laboratories America Inc.,4 Independence Way, Princeton, NJ 08540 (USA)
ivancic@nec-labs.com

Abstract. There are numerous application examples for hybrid systems verification in recent literature. Most of them were introduced to illustrate a new approach to hybrid systems verification, and are therefore of a limited size. Others are case studies that serve to prove that an approach can be applied to real world problems. Verification of these typically requires a lot of domain experience to obtain a tractable, verifiable model. Verification of a case study yields a singular result that is hard to compare and time-consuming to reproduce.
This paper introduces three benchmarks for hybrid systems verification. These benchmarks are independent from a particular approach to verification, they have a limited domain, and have a simple basic structure. Nevertheless, these benchmarks can be scaled to arbitrary complexity, and offer the possibility to inject phenomena that are known to be problematic in hybrid verification. This paper presents result for a first set of instances, as an example of how these benchmark can be used to compare different tools and approaches.

1 Introduction

Recently, it has been questioned whether Hybrid System verification is scalable and applicable to real world applications. There have been a number of case studies, but these are either fairly small and serve mainly to illustrate a concept, or they are very specific and hard to compare and reproduce. This paper introduces some problems that we propose as benchmarks for evaluating and comparing tools for hybrid system design and verification.

The primary purpose of benchmarks is to compare different methods, and to provide a means to record future advances. In the area of hybrid verification this means that different verification methods are applied to the same benchmark

* This research was supported by the Defense Advanced Research Project Agency (DARPA) MoBIES project under contracts no. F3361500C1701 and F33615-02-C-0429, by the Army Research Office (ARO) under contract no. DAAD19-01-1-0485, by the National Science Foundation (NSF) under grants no. CCR-0121547 and CCR-0098072, by the Office of Naval Research (ONR) under contract no. N00014-95-1-0520. The views and conclusions in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of DARPA, ARO, ONR, NSF, the U.S. Government or any other entity.

R. Alur and G.J. Pappas (Eds.): HSCC 2004, LNCS 2993, pp. 326–341, 2004.
© Springer-Verlag Berlin Heidelberg 2004

problems. For each method this can then reveal the limits of a certain method. A useful benchmark should be scalable in different dimensions. In hybrid verification important dimensions include the number of discrete control locations, the number of continuous variables, and type of the dynamic behavior – timed, rectangular, linear, or non-linear.

Benchmarks help to determine the limits of a certain method, and these can then be compared to the limits of other methods. But equally important, knowing these limits helps to determine whether a certain method is suitable for a certain problem at all. Or vice versa, when a certain method has to be used, knowing its limitations determines how a model needs to be modified.

Especially in verification of hybrid systems it is quite common that a system can only be verified after tweaking the model and the verification parameters. Obtaining a verifiable model is often the most work in this area, and results depend often on particular smart choices, so that it becomes difficult to determine whether the particular method solved the problem or the experienced user. Having multiple instances of similar benchmarks may reduce these effects for the following reason. Tweaking model and method is inherently tedious and difficult, and a good choice of verification parameters for one instance, may not be a proper choice for another instance. When a method performed well on a range of instances, one can assume that the method mattered, rather than the experienced user finding a proper setup.

An important part in verification is, as mentioned before, the process of obtaining a model. Some methods even rely on a particular modelling framework. Benchmarks can be used for comparison of different frameworks. The benchmarks will be defined in general terms, informal but precise. Part of the benchmark problem is then to provide a formal model, and one can thus compare for example different methods for composition and synchronization.

A number of application examples and case studies have been used in the literature to evaluate tools. The generalized railway crossing example was put forward by Heitmeyer in [HJL93] as a benchmark to evaluate approaches to specify and verify real-time systems. This benchmark was for example used to illustrate the capabilities of HyTech in [HHWT95], of STeP in [BMSU97] and of the Esterel toolkit in [JPO95]. Bérad and Sierra use this benchmark in [BS00] to compare quantitatively the performance of the model checkers HyTech [HHWT95], Uppaal [LPY97] and KRONOS [Yov97]. They use a scalable model of the railway crossing example, very much like the benchmarks presented in this paper, except that this benchmark remains in the realm of timed automata.

A qualitative assessment of different algorithmic approaches in hybrid verification is given in [SSKE01]. This paper used a batch reactor system proposed in [SKPT98] to compare features, such as the interface, the logic used for specification, and the expressiveness of the modelling framework, of a number of verification tools. Since the performance of the tools was not subject of this paper, there was no need for a scalable model of the batch reactor system.

Stauner et al. presented an automotive level control system in [SMF97] as a hybrid automaton with linear dynamics. They provided an abstraction of the system and verified the system with HyTech. Alternative approaches to hybrid

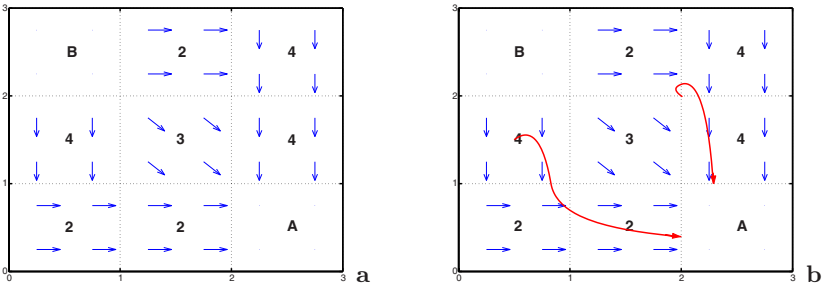

Fig. 1. a. The map determines the desired velocity of the moving object, depending on the position of the object. **b.** Two trajectories of objects moving in the plane. Both objects eventually reach cell **A**.

verification were applied by [Feh98,BM99,EB99] to the same problem. These papers use the automotive level control system to prove a concept, and consider therefore only a single instance. Comparison of the results is hindered by the fact that all papers use slight modifications, simplifications or additional assumptions to obtain the result.

There are numerous other application examples, but most of them share that they are either not scalable, fairly particular to a certain framework, or that results are hard to compare and reproduce. The next section presents three benchmark problems for hybrid systems that tackle these problems. The first considers a moving object, the second leaking valves in a network of pressurized pipelines, and the last a house with a limited number of heaters. The benchmarks are scalable, and to offer the possibility to inject phenomena that are known be problematic for verification. Section 3 discusses the different problem dimensions and characteristics of the benchmarks. Section 4 presents first results for one of the benchmarks. Finally, Section 5 concludes the paper by giving a few guidelines on how to use these benchmarks.

2 Benchmarks

2.1 Navigation Benchmark

The first benchmark deals with an object[1] that moves in the $\mathbb{R}^2$ plane. The desired velocity $\mathbf{v}_d$ is determined by the position of the object in an $n \times m$ grid, and the desired velocities may take values $(\sin(i * \pi/4), \cos(i * \pi/4))$, for $i = 0, \ldots, 7$. We assume that the length and the width of a cell is 1, and that the lower left corner of the grid is the origin. An example of a 3×3 grid is depicted in Figure 1.a, where the label i in each cell refers to the desired velocity. In addition, the grid contains cells labelled **A** that have to be reached and cells labelled **B** that ought to be avoided.

[1] This object can be thought of as a vehicle, though the dynamics are not exactly vehicle dynamics

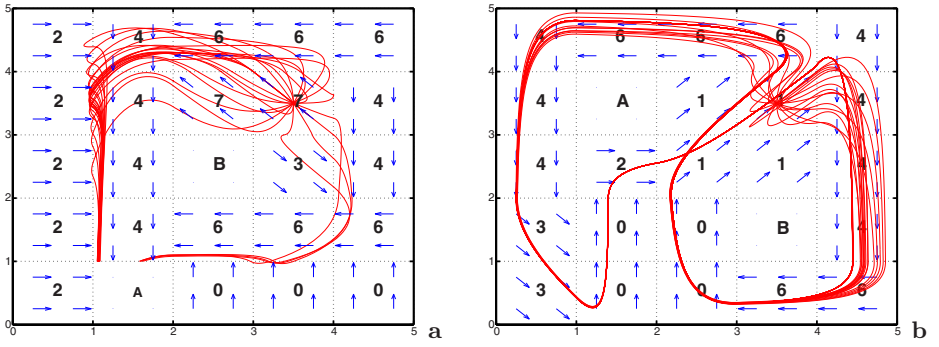

Fig. 2. A number of different trajectories for the two instances of the navigation benchmark.

Given $\mathbf{v}_d$ the behavior of the actual velocity $\mathbf{v}$ is determined by the differential equation $\dot{\mathbf{v}} = A(\mathbf{v} - \mathbf{v}_d)$, where $A \in \mathbb{R}^{2\times 2}$ is assumed to have eigenvalues with strictly negative real part. This guarantees that the velocity will converge to the desired velocity. Figure 1.b shows two trajectories, with $A = \begin{pmatrix} -1.2 & 0.1 \\ 0.1 & -1.2 \end{pmatrix}$. Both satisfy the property that **A** should be reached, and **B** avoided.

If the trajectory leaves the grid, the desired velocity $\mathbf{v}_d$ is the velocity of the closest cell. Hence, the outer cells are assumed to be unbounded in the direction of the border of the map. For example, the desired velocity in Figure 1 is $(\sin(4 * \pi/4), \cos(4 * \pi/4))$ for all $\mathbf{x} = (x_1, x_2)^T$ with $x_1 >= 2$ and $x_2 >= 1$

An instance of this benchmark is characterized by the initial condition on $\mathbf{x}$ and $\mathbf{v}$, by matrix A in the differential equation for $\mathbf{v}$ and by the map of the grid, which can be represented as $n \times m$ matrix with elements from $\{0, \ldots, 7\} \cup \{\mathbf{A}, \mathbf{B}\}$. For the example in Figure 1 this matrix is

$$\begin{matrix} \mathbf{B}\ 2\ 4 \\ 4\ 3\ 4 \\ 2\ 2\ \mathbf{A} \end{matrix} \tag{1}$$

We will refer to this matrix as the map of an instance.

The map and the matrix A determines mainly the size and complexity of an instance. Proper choices are used to stress a certain aspect in hybrid verification. Figure 2.a shows a few trajectories for an instance with a 5×5 map. For this instance we chose $A = \begin{pmatrix} -0.8 & -0.2 \\ -0.1 & -0.8 \end{pmatrix}$. This instance satisfies the requirements, but a few trajectories are getting close to cell **B**; this instance thus puts an emphasis on numerical accuracy of a method. Figure 2.b shows trajectories for another instance with the same initial conditions, but a different map and with $A = \begin{pmatrix} -1.2 & 0.1 \\ 0.2 & -1.2 \end{pmatrix}$. None of the trajectories in Figure 2.b reaches **A**, but all of them avoid **B**. This instance is an example of a hybrid systems with behaviors of infinite length.

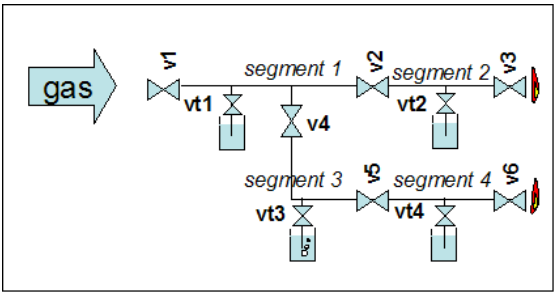

Fig. 3. A simple network with four segments

2.2 Leak Test Procedure

The next benchmark is inspired by earlier work presented in [TPP97], but it differs from it in that we define the dynamics as ordinary differential equations. The benchmark deals with the detection of leaks in a pressurized network. The network is a tree, with a source of gas (methane) at the root, and burners at the leaves of the tree. Figure 3 depicts an example of such a network. Each segment is connected via a tap valve to a trickle device. This device is nothing more than a cup of water with an incomming pipe. If the tap valve is open, bubbles indicate that the pressure in the segment is above a certain threshold.

Leaking valves in the network can lead to flammable clouds during shutdown periods. A leak test procedure checks for leaks across the network, to reduce the risk of explosions. The leak test procedure first pressurizes the network. It then closes all valves, and tests segment by segment, starting with the segments close to the burners.

For each segment the test is performed as follows: First, wait for a certain amount of time, then open the tap valve. Absence of bubbling indicates that a downstream valve is leaking and has to be replaced. Note that segments, such as segment 1 in the Figure 3, can have more than one downstream valve. When the bubbling does start, wait for it to stop. If it does not stop within a given time, a leak in the upstream valve is assumed. When bubbling does stop, the test for this segment is completed.

The leak test procedure starts with the leaf-segments of the network. The other segments are tested as soon as the tests for all downstream segments are completed. This means for the network in Figure 3, that the procedure starts with segments 4 and 2, as soon as the network is pressurized. When the test for segment 4 is completed successfully, the test for segment 3 will begin. As soon as the tests for segment 2 and 3 are completed, the test for segment 1 will begin. If a leak is detected at any time during the procedure the complete procedure aborts.

The pressure in a segment depends on the pressure in adjacent segments, it depends on what valves are open or closed, and it depends on whether the tap valve is open or closed. For each segment i we model the pressure by a state

variable x_i. We refer for simplicity to pressure at the source of the gas as x_0 and to the pressure in the environment as x_{n+1}. For the source of gas and the environment we assume that the pressure is either constant, or can take any value in a given interval.

For a segment i_0 with adjacent segments $i_1, \ldots, i_k$ we have

$$\dot{x}_{i_0} = \sum_{j=1,\ldots,k} c_j f(x_{i_0}, x_{i_k}) + dg(x_{i_0}) \tag{2}$$

The constants c_j depend on whether the valve between segment i_0 and segment i_j is open, and the constant d depends on whether the tap valve of segment i_0 is open. The functions f and g are defined as follows.

$$f(x,y) = \begin{cases} -\sqrt{x-y} \text{ if } x \geq y \\ \sqrt{y-x} \text{ otherwise} \end{cases} \tag{3}$$

$$g(x) = \begin{cases} -\sqrt{x - z_{off}} \text{ if } x \geq z_{off} \\ 0 \qquad\qquad\quad \text{otherwise} \end{cases} \tag{4}$$

with z_{off} a constant that determines at what pressure bubbling starts.

Segment 3 in Figure 3 for example is connected to valves **v4** and **v5** and tap valve **vt3**. This leads to eight possible combinations of open and closed valves. Assumed that the rate is for an open valve 1, for a closed valve 0.01, and for the tap valve 0.1. Supposed that that the tap valve **vt3** and **v5** are open and valve **v4** is closed, we then obtain for (2):

$$x_3 = f(x_3, x_4) + 0.01\, f(x_3, x_1) + 0.1\, g(x_3) \tag{5}$$

Supposed that the valves **vt3** and **v5** are closed and valve **v4** is open we obtain:

$$x_3 = 0.01\, f(x_3, x_4) + f(x_3, x_1) \tag{6}$$

The verification problem is to show that leaking valves are detected properly. We require the following for each segment:

1. If a segment is tested and if **none** of its upstream valves leaks, then the bubbling should start.
2. If a segment is tested, and if an upstream valve leaks, then the bubbling should not start. We assume that the model is deadlock free, and that time can pass.
3. If the root segment is tested, the test should detect correctly whether or not the downstream valve leaks.

Suppose for example that valve **v5** leaks. The requirement for segment 3 is satisfied, if the test detects an upstream leak, or if the test for this segment is not performed. The latter will be the case if the test of segment 4 will detect a leak, either in valve **v5** or **v6**. If a leak is detected, the procedure is terminated, and segment 3 will not be tested.

An instance of the benchmark is defined by the topology of the network, the waiting times of the procedure per segment, the constants for open and

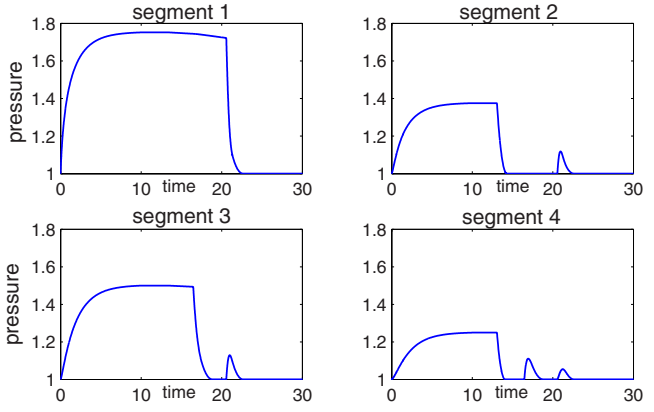

Fig. 4. The leak test procedure starts with segments 2 and 4. In this simulation it correctly finds that no leaks are present.

closed valves and the thresholds for bubbling. Each valve in the network has a unique upstream segment, and each segment has a unique upstream valve. The network with n segments and m valves is defined by an n-tuple, that defines for each segment the upstream valve, and an m-tuple, that defines for each valve the upstream segment. The root valve has as upstream node 0. The network in Figure 3 has four segments and six valves. The connections from segments to upstream valves are encoded as `seg2val` $= (1, 2, 4, 5)$ and the connection of valves to upstream segments as `val2seg` $= (0, 1, 2, 1, 3, 4)$.

Given the network, we define the initial pressure either by an n-tuple, whose elements are scalars or intervals. The pressure in the environment and at the gas source is given as either a constant or constraint to an interval. In the latter case the pressure may change arbitrarily within this interval. The simulation result in Figure 4 uses for the pressure at the source and in the environment $x_0 = 2$, and $x_5 = 1$, respectively, and as the pressure of the segments $i = 1, \ldots, 4$ at time zero $x_i(0) = 1$.

For each valve we have to define the flow rate c_i in (2). For each instance we define two m-tuples. The first defines the rate for open valves, the second defines it for closed valves. Elements of these tuples can be intervals as well. In this case the rate takes a constant value in that interval, i.e. it is an uncertain parameter of the problem. The simulation shown in Figure 4 uses for the open valves $c_{open} = (1, 1, 1, 1, 1, 1)$, for the closed valves $c_{closed} = (0.01, 0.01, 0.01, 0.01, 0.01, 0.01)$. In addition, we have to indicate which valves are leaking. In the instance we assume that none is leaking. The constant b in (2) for the tap valve, and the threshold z when bubbling starts is the same for all segments. The simulation in Figure 4 uses $d = 0.1$ and $z_{off} = 1.1$. There is no flow and thus no bubbling when the tap valve is closed.

Finally, one has to define the durations that determine the leak test procedure. This is the initialization time, which in the example is $t_{init} = 10$, and for

each segment waiting time before it opens its tap valve, and finally the time it waits at most for the bubbling to stop. These times may vary from segment to segment, and they are defined as n-tuples. The simulation in Figure 4 uses $t_{wait} = (3, 3, 3, 3)$ for the waiting times and $t_{test} = (3, 3, 3, 3)$ as maximal duration of the bubbling.

Figure 4 shows simulation result for an instance of the leak test problem. In the initial phase all valves are open. They are closed after 10 minutes, and testing of segment 2 and 4 begins. After 3 minutes the tap valves of these segments are opened. Bubbling starts in both segments since pressure is above 1.1 bar. Thus none of the downstream valves is found to be leaking.

After about 2 minutes segment 4 stops bubbling, and the procedure starts with segment 3. Bubbling stops in segment 2 shortly after that. Neither of the upstream valves of segment 2 and 4 is found to be leaking. The test for segment 3 opens the tap valve, it detects bubbling, and proceeds with the test. When bubbling in segment 2 and 3 stops, segment 1 is tested. This test also completes successfully. Note that the pressure rises in segment 2, 3 and 4, when the downstream valve of segment 1 is opened.

2.3 Room Heating Benchmark

This last benchmark deals with a house with a number of rooms that are heated by a limited number of heaters. The temperature in each room depends on the temperature of the adjacent rooms, on the outside temperature, and on whether a heater is in the room. The number of heaters is assumed to be smaller than the number of rooms, and each room may have at most one heater. The heater is controlled by a typical thermostat, i.e. it is switched on if the temperature is below a certain threshold, and off if it is beyond another (higher) threshold.

When the temperature in a room falls below a certain level, it may get a heater from one of the adjacent rooms, provided that the temperature in this room is significantly higher. In this way the heaters are shared by the different rooms to maintain some minimum temperature in all rooms.

Let x_i be the temperature in room i, u the outside temperature, and h_i a boolean variable that is 1 when there is a heater in the room and switched on, and 0 otherwise. The temperature of a room depends linearly on the difference of the temperature with the other rooms, the difference with the outside temperature, and on whether the heater is present and switched on or off. The system dynamics are given by

$$\dot{x}_i = c_i h_i + b_i(u - x_i) + \sum_{i \neq j} a_{i,j}(x_j - x_i) \tag{7}$$

with constants $a_{i,j}, b_i, c_i$. We assume that the heat exchange is symmetric, i.e. $a_{i,j} = a_{j,i}$, and that all heaters are identical. We say that rooms i and j are adjacent if $a_{i,j} > 0$.

Each heater has a thermostat that switches the heater on if the temperature in a room is below a certain threshold, and off when the temperature reaches a

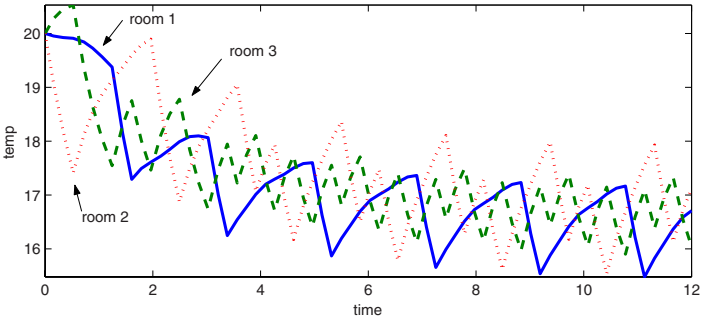

Fig. 5. Simulink simulation of the temperature of three rooms, that are heated by two heaters.

higher temperature. For each room we define thresholds on_i and off_i; the heater in room i is on if $x_i \leq on_i$ and off if $x_i \geq \mathit{off}_i$.

A heater is moved from room j to an adjacent room i if the following holds

- room i has no heater
- room j has a heater
- temperature $x_i \leq get_i$
- the difference $x_j - x_i \geq dif_i$

The constants get_i and dif_i may differ for each room. When two or more heaters can be moved, the choice is made non-deterministically.

An instance is defined by the number of rooms n, the number of heaters m, the coefficients $a_{i,j}, b_i, c_i$, and the thresholds $on_i, \mathit{off}_i, get_i, dif_i$. In addition, one needs to know how the heaters are initially distributed over the different rooms.

Figure 5 shows simulation results for an instance of the room heating benchmark. Let x be the vector of temperatures, h be a boolean vector that denotes whether a heater is on or off. The continuous behavior is then governed by

$$\dot{x} = \begin{pmatrix} -0.9 & 0.5 & 0 \\ 0.5 & -1.3 & 0.5 \\ 0 & 0.5 & -0.9 \end{pmatrix} x + \begin{pmatrix} 0.4 \\ 0.3 \\ 0.4 \end{pmatrix} u + diag(6,7,8)\, h \tag{8}$$

The outside temperature is constantly $u = 4$. We assume initially $x(0) = (20,20,20)^T$ and $h(0) = (1,1,0)^T$. The thresholds for the heaters are $\mathit{off} = (21,21,21)^T$ and $on = (20,20,20)^T$. The control strategy is determined by $get = (18,18,18)^T$ and $dif = (1,1,1)^T$.

Figure 5 shows the simulation results for this instance. Even though the initial condition is a single point, the simulation does not cover the complete behavior. The control strategy includes non-deterministic choices, and the simulation shows just one possible path. At time 1 the temperature in room 2 is below 18. The differences $x_1 - x_2$ and $x_3 - x_2$ are both greater than 1, and the controller has to make non-deterministic choice from which room to move the heater to room 2. In this simulation room 1 is chosen.

For the room heating problem we are considering the following requirements:

- The temperature in all rooms is always above a given threshold.
- All rooms get eventually a heater.
- In all rooms there will be eventually no heater.

The last two requirements ensure that each room has at least once a heater, and shares the heater at least once with other rooms.

3 Benchmark Characteristics

The benchmarks that were presented in the pervious section can be used to examine different aspects of hybrid system verification. All benchmarks were chosen such that they are scalable in different problem dimensions.

The most constraining problem dimensions in hybrid verification are the number of continuous state variables and the number of discrete locations and transitions.

Each instance of the navigation benchmark has 4 continuous state variables, and linear dynamics. The navigation benchmark can be modelled with just 10 discrete states, 8 for each possible commanded direction and one for the cells labelled **A** and **B**, respectively. The number of transitions typically increases with a larger map, independent of how the cells are mapped to locations in the model. However, the number of transitions is bounded by four outgoing transitions per cell - namely to the neighboring cells of the grid map.

Given that the number of state variables is fixed, but the number of transitions (and probably also of the locations) increases with the size of the map, this benchmark is suitable to determine the influence of the complexity of the switching logic on the performance of a method.

All instances of the leak test benchmark have one continuous state variable for each segment to model the pressure. The different branches are tested concurrently, and for each branch an additional continuous state variable is necessary to time the steps of the procedure. Testing the different branches concurrently may also introduce interleavings and branching, but since the procedure contains no loops, each discrete control location can be visited at most once. This yields a rather simple discrete control structure for the leak test benchmark. This benchmark is therefore aimed to investigate the influence of an increasing complexity in the continuous part on the performance of a verification method.

Given an instance of the room heating benchmark with h heaters and r rooms where $1 \leq h \leq r$, there are $\binom{r}{h}$ many ways to distribute the h heaters across the r rooms. In addition, in each of these configurations, each heater may either be turned on or turned off, which brings the total to $\binom{r}{h} \cdot 2^h$ many configurations. The dynamics can be different in each of these cases, and thus introduce an additional control location. The number of transitions from each location is, in worst case, $h + h(r - h)$. This number is derived from the fact that each heater can switch from on to off (or vice versa), and that each heater can move to its adjacent empty rooms, in worst case $r - h$ many. This benchmark is to be

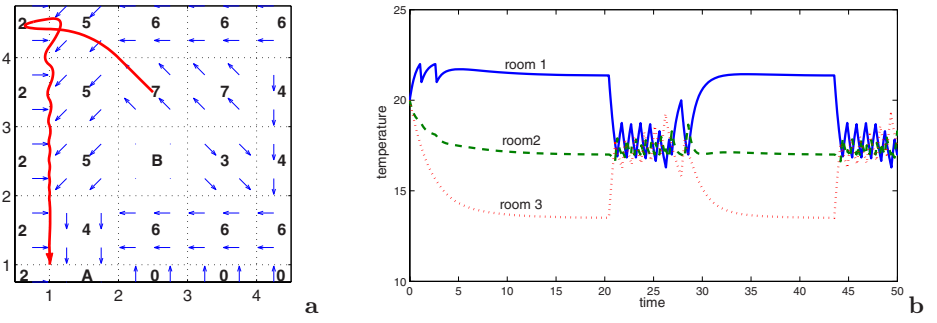

Fig. 6. a. Instance of the navigation benchmark that can exhibit chattering close to $x = 1$. **b.** Simulation result for an instance of the heating benchmark. The system changes periodically between an almost stable state and and a period of fast switching.

expected to grow fastest in complexity, when rooms and heaters are added. Most current hybrid systems verification tools require to flatten the model a priori. We expect that verification of large benchmark instances will only become feasible when the modularity in the model is exploited.

Whether an approach to verification can be used for a certain problem often depends on the kinds of dynamics. Some approaches assume e.g. that the dynamics are linear. An approach to deal with nonlinear system is then to use abstractions with simplified dynamics. The navigation benchmark and the heating benchmark both have linear dynamics. The leak test benchmark has, as mentioned before, non-linear dynamics, due to the square root in (3) and (4). But since the functions f and g are continuous and monotonic, we expect that finding suitable abstractions should be possible, if it is necessary at all.

Besides the number of continuous variables, the number of discrete locations transitions and the kind of dynamics, an instance may exhibit other characteristics that may make it hard to analyze. A first example are hybrid systems that chatter, i.e. they take many discrete transitions in a short amount of time. A particular problem can be caused by Zeno behavior, i.e. that an infinite number of transitions are taken in a finite amount of time. Figure 6.a shows an instance of the navigation benchmark that chatters around $x = 1$. Even though it is not Zeno, it may cause problems during analysis.

The opposite effect can be observed when an instance includes (stable) equilibrium points. In this case the system can stay in the equilibrium forever, without taking any further discrete transition. Figure 6.b shows simulation results for an instance with one heater for three rooms. When the heater is in room 1 and switched on, then there is a stable equilibrium just below the 17 degrees threshold for room 2. When the temperature drops below 17 degrees, it may obtain the heater from room 1. In this case this threshold is eventually reached and we observe long periods with no switching, interspersed with periods of fast switching.

Table 1. Characteristics of the different benchmarks.

	navigation benchmark	leak test benchmark	room heating benchmark
continuous variables	4	1 per segment (plus 1 per branch)	1 for each room
dynamics	linear	non-linear	linear
discrete locations	at least 10	at least 3 per segment	$\binom{r}{h}2^h$ for h heaters and r rooms
chattering / Zeno behavior	some instances	none	none
convergence to equibrium	none	none	some instances
non-deterministic switching	none	none	some instances
sampled transitions	none	some instances	none

The instances of benchmarks can exemplify different kinds of switching. The heating example has non-deterministic switching, as already noted in the previous section. Any model of the heating instances should capture this non-determinism. Some approaches to verification benefit if transitions can only happen at sampling times [SK01]. The analysis can treat these transitions differently, such that it is not necessary to include the continuous state variables to time the steps of the procedure. The leaking procedure for example waits for a certain time before it tests for bubbling. Autonomous switching in contrast can happen at any point in time.

4 Benchmark Results

In the following, a few experimental results will be presented that have been obtained for instances of the navigation benchmark. These result serve as example of how benchmarks can be used to compare tools, and how they can be used to examine the impact of characteristics of an instance on the result

The set of instances in Table 2 tests the effects of varying initial conditions. The variation is in the set of initial conditions for the velocity of the object. The instances are both run using the `d/dt` tool [Dan99] as well as the predicate abstraction based verifier [ADI02] of the CHARON toolkit [AGH+00].

The model of these benchmark instances has been differs slightly from the description given in Section 2.1. It is assumed that the width of each cell is $1+2\epsilon$ for $0 \leq \epsilon < 0.5$, and that the lower left corner of the grid is located at $(-\epsilon, -\epsilon)^T$. The square grid cells are arranged such that neighboring cells overlap for 2ϵ. First consider the case that $\epsilon = 0$. Then each cell can be described by its lower left corner at some $(j, k)^T \in \mathbb{R}^2$ and its upper right corner at $(j+1, k+1)^T$ for $j, k \in \mathbb{N}$. However, if each cell has width $1 + 2\epsilon$ and overlaps as described above, then it can be described by having its lower left corner at $(j - \epsilon, k - \epsilon)^T \in \mathbb{R}^2$

Table 2. Varying initial conditions on the velocity of the moving object

Instance	Initial Conditions
NAV01	$\mathbf{x}_0 \in [-0.3, 0.3], \mathbf{v}_0 \in [-0.3, 0]$
NAV02	$\mathbf{x}_0 \in [-0.3, 0.3], \mathbf{v}_0 \in [-0.3, 0.3]$
NAV03	$\mathbf{x}_0 \in [-0.4, 0.4], \mathbf{v}_0 \in [-0.4, 0.4]$

and its upper right corner at $(j+1+\epsilon, k+1+\epsilon)^T$. In fact, in certain regions of the grid three and even four cells may overlap. This introduces non-determinism and improves numerical stability of various analyses, since determining the exact time of the switch of the moving object between the cells may be numerically hard to compute. At the same time this model is an abstraction of the instance, in the sense that all behaviors of the instance will be contained in the model. A script has been developed that produces a CHARON model from a short textual description of an instance of this benchmark.

The instance uses the map and the matrix given in Figure 1, but uses different initial conditions. The initial position of the object is the grid cell just above the attracting cell labelled **A**, with different sets of initial velocities. The set of initial conditions are described in Table 2. We choose $\epsilon = 0.1$ for the overlap. The first instance NAV01 described in Table 2 is an easily verifiable instance of the benchmark model, since the object has an initial starting velocity pointing towards the attracting cell. However, the second and third instance are of somewhat higher complexity since the object may start off with an initial velocity that is pointing away from the attracting cell directly towards the bad cell.

The experiments were performed both with the CHARON based verifier and the `d/dt` tool. The latter supplies some library functions to the predicate abstraction based verification tool of CHARON. As expected, both tools were able to verify instance NAV01 without any significant user-guidance in a few seconds. In fact, it turned out that for this instance, the `d/dt` tool outperformed the predicate abstraction based method of the CHARON tool with respect to the computation time. This is mainly due to the fact that certain initialization steps of the CHARON based tool are not needed by `d/dt`.

When trying to verify instance NAV02, both tools were able to complete the verification task and prove safety (avoidance of the bad cell). However, during the verification of this instance several verification parameters needed to be adjusted in both tools to complete the task. The verification of instance NAV03, however, was proven in the CHARON based verifier with the same set of parameters, while the `d/dt` tool was not able to complete the verification task. Either, the time-step was too large, and safety could not be proven, or the verification task was not completed due to a memory overflow.

In this section we showed how to use instances of a benchmark to compare different approaches to hybrid verification. It became also clear that a proper setup of the verification algorithm is important, too. More results on this benchmark, and on verification results for some instances of the room heating benchmark can be found in [Iva03]. The considered set of instances of the navigation benchmark

Table 3. Textual description of the first instance of the navigation benchmark. See Figure 1 and Table 2

```
MAP=[B 2 4; 2 3 4; 2 2 A]
MatA = [ -1.2 0.1;0.1 -1.2]
x0 in [2,3]x[1,2]
v0 in [-0.3,0.3]x[-0.3,0]
```

tests the CHARON based verifier with respect to its adaptiveness to verification tasks where the number of locations grows substantially. It also provides more background to the approaches used in this section.

5 Conclusions

This paper presents three benchmarks for hybrid verification. These benchmarks are scalable in the number of continuous variables and the number of discrete locations. This helps to asses how different approaches deal with increasing complexity. Furthermore, instances can be chosen to exhibit certain characteristics that maybe problematic for certain approaches.

These benchmarks are aimed at all methods for hybrid verification. This includes methods for computer aided verification that require user-interaction. A successful application of a verification approach should be able to prove or disprove the properties for a number of benchmark instances.

We will provide for each benchmark 30 instances. A valid model of an instance should include all behaviors of an instance. This means that a model of a room heating instance has to maintain the non-deterministic choice, rather than resolving the non-determinism. On the other hand, it does include abstractions (or over-approximations) that preserve the behavior of the instance. In Subsection 4 we present a model for the navigation benchmark that extends each cell by ϵ in each direction, and thus contains all behaviors described in the benchmark.

The instances and the description of the benchmarks will be maintained on a web-page (`http://www.ece.cmu.edu/~ansgar/benchmark/`). Each instance of this benchmark is given by a brief textual description as depicted in Table 3. On this web-page we will also put a Simulink models of a number of instances. These models, however, should not be used as baseline for verification, but just as auxiliary to gain some insight into a benchmark. The Simulink models are just particular implementations of benchmark instances, and in some cases -due to limitations in Simulink's modelling framework - just approximation of a proper implementation.

Acknowledgement. The authors thank Rajeev Alur and Bruce Krogh for their input and feedback on defining the scope and purpose of the benchmarks presented in this paper.

References

[ADI02] R. Alur, T. Dang, and F. Ivančić, *Reachability analysis of hybrid systems via predicate abstraction*, 5^{th} Int. Workshop on Hybrid Systems: Computation and Control, LNCS 2289, Springer, 2002, pp. 35–48.

[AGH+00] R. Alur, R. Grosu, Y. Hur, V. Kumar, and I. Lee, *Modular specification of hybrid systems in charon*, 3^{rd} Int. Workshop on Hybrid Systems: Computation and Control, LNCS 1790, Springer, 2000.

[BM99] A. Bemporad and M. Morari, *Verification of hybrid systems via mathematical programming*, 2^{nd} Int. Workshop on Hybrid Systems: Computation and Control, LNCS 1569, Springer, 1999.

[BMSU97] N. Bjorner, Z. Manna, H. Sipma, and T. Uribe, *Deductive verification of real-time systems using STeP*, 4^{th} Int. AMAST Workshop on Real-time Systems, LNCS 1231, Springer, 1997.

[BS00] B. Bérad and L. Sierra, *Comparing verification with HyTech, Kronos and Uppaal on the railroad crossing example*, Tech. Report LSV-00-2, CNRS & ENS de Chachan, France, 2000.

[Dan99] T. Dang, *Vérification et synthèse des systémes hybrides*, Ph.D. thesis, Verimag, Grenoble, 1999.

[EB99] N. Elia and B. Brandin, *Verification of an automotive active leveler*, Proc. of the 1999 American Control Conference, 1999.

[Feh98] A. Fehnker, *Automotive control revisited – Linear inequalities as approximation of reachable sets*, Int. Workshop on Hybrid Systems: Computation and Control, LNCS 1386, Springer, 1998.

[HHWT95] T.A. Henzinger, P.H. Ho, and H. Wong-Toi, *HyTech: The next generation*, IEEE Real-Time Systems Symposium, 1995.

[HJL93] C.L. Heitmeyer, R.D. Jeffords, and B.G. Labaw, *A benchmark for comparing different approaches for specifying and verifying real-time systems*, 10th IEEE Workshop on Real-Time Operating Systems and Software, IEEE Computer Society Press, 1993.

[Iva03] F. Ivančić, *Modeling and analysis of hybrid systems*, Ph.D. thesis, School of Engineering and Applied Science, University of Pennsylvania, 2003.

[JPO95] L.J. Jagadeesan, C. Puchol, and J.E. Von Olnhausen, *Safety property verification of Esterel programs and applications to telecommunications software*, 7^{th} Int. Conference On Computer Aided Verification, LNCS 939, Springer Verlag, 1995.

[LPY97] K.G. Larsen, P. Pettersson, and W. Yi, UPPAAL *in a Nutshell*, Int. Journal on Software Tools for Technology Transfer **1** (1997), no. 1–2, 134–152.

[SK01] B.I. Silva and B.H. Krogh, *Modeling and verification of hybrid system with clocked and unclocked events*, 40^{th} Conference on Decision and Control, 2001.

[SKPT98] O. Stursberg, S. Kowalewski, J. Preussig, and H. Treseler, *Block-diagram based modelling and analysis of hybrid processes under discrete control*, J. Europeen des Syst. Automatises **32** (1998), no. 9-10, 1097–1118.

[SMF97] T. Stauner, O. Müller, and M. Fuchs, *Using HyTech to verify an automotive control system*, Int. Workshop on Hybrid and Real-Time Systems, LNCS 1201, Springer, 1997.

[SSKE01] B.I. Silva, O. Stursberg, B. Krogh, and S. Engell, *An assessment of the current status of algorithmic approaches to the verification of hybrid systems*, 40^{th} IEEE Conf. on Decision and Control, 2001, pp. 2867–2874.

[TPP97] A. Turk, S. Probst, and G. Powers, *Verification of a chemical process leak test procedure*, 9^{th} Int. Conference On Computer Aided Verification, LNCS 1254, Springer, 1997.

[Yov97] S. Yovine., *Kronos: A verification tool for real-time systems*, Int. Journal of Software Tools for Technology Transfer **1** (1997), no. 1/2,.