Mathematical Modelling of Systems
1996, Vol. 1, No. 1, pp. 000–111

1381-2424/96/0201-00$12.00

Modelling and Verification using Linear Hybrid Automata - a Case Study*

OLAF MÜLLER AND THOMAS STAUNER†

ABSTRACT

This paper discusses the use of hybrid automata to specify and verify embedded distributed systems, that consist of both discrete and continuous components. The basis of the evaluation is an automotive control system, which controls the height of an automobile by pneumatic suspension. It has been proposed by BMW AG as a case study taken from a current industrial development. Essential parts of the system have been modelled as hybrid automata and for appropiate abstractions several safety properties have been verified. The verification has been performed using HyTech, a symbolic model checker for linear hybrid automata. The paper discusses the general appropiateness of hybrid automata to specify hybrid systems as well as advantages and drawbacks of the applied model-checking techniques.

Key words: AMS classification: automata theory (68Q68), control systems (93C99), distributed systems (68Q22), mathematical modelling (93A30), specification and verification of programs (68Q60).

1 INTRODUCTION

Hybrid Systems are systems that intermix discrete and continous components, typically controllers that interact with the physical environment. Since embedded applications are steadily increasing, especially in safety critical areas as e.g. avionics or automotive electronics, there is a need to provide specification, analysis and verification methods for hybrid systems. Recently, *hybrid automata* [1] have been proposed as a mixed discrete/continuous specification method that allows automatic verification for specific subclasses using model checking techniques.

*This work was supported (in part) with funds of the BMBF within the *KorSys* Project and funds of the Deutsche Forschungsgemeinschaft under reference number Br 887/9-1 within the priority program *Design and design methodology of embedded systems.*

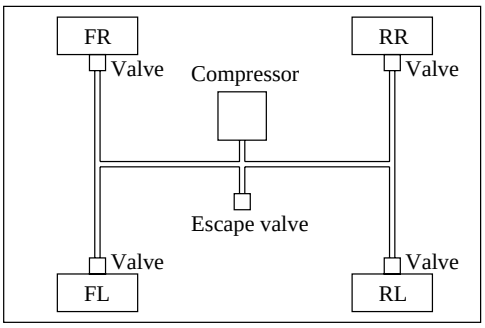
†Computer Science Department, Munich University of Technology, D-80290 Munich, Germany. {mueller,stauner}@informatik.tu-muenchen.de

The aim of this paper is to evaluate the practicability of hybrid automata. Concerning specification, the appropiateness of the modelling capabilities is discussed. Concerning verification, a representative of the existing model checkers is evaluated w.r.t. automation and scalability. We use the model checker HyTech [6], because it provides the most general input language, namely *linear hybrid automata*.

The basis of the assessment is a case study provided by our research partner BMW AG. It has been taken from an actual industrial development and consists of a system that controls the height of a chassis by pneumatic suspension. The system is specified by means of (linear) hybrid automata, and HyTech is applied to appropriate abstractions.

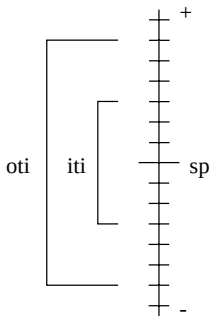
The paper is organized as follows: First, the pneumatic suspension system is described informally (Section 2). Then, hybrid automata, the model checking algorithm and HyTech are introduced (Section 3). The following sections present the evaluation results concerning specification (Section 4) and verification (Section 5). Both sections are themselves divided into two parts: the first describes the concrete experiences obtained in the case study, the second contains more general conclusions.

2 THE ELECTRONIC HEIGHT CONTROL SYSTEM



FR: front right wheel RR: rear right wheel
FL: front left wheel RL: rear left wheel

Fig. 1. The mechanic system.



sp: set point
iti: inner tolerance interval
oti: outer tolerance interval

Fig. 2. The tolerance intervals.

The main aims of the electronic height control (EHC) are to increase driving comfort, to keep the headlight load-independent and to adjust the chassis level

to off-road and on-road conditions by providing a high and a low chassis level, selectable by the driver.

This is achieved by a pneumatic suspension at each of the four wheels. The chassis level can be increased by pumping air into the suspension of the wheels and decreased by blowing air off. As there is only one compressor and one escape valve for all four wheels, to e. g. increase the level at the front right wheel, the compressor has to be turned on, the escape valve must be closed and the valve connected to the suspension of the front right wheel must be opened (Fig. 1).

Sensors measure the deviation of the actual chassis level from a defined zero level at each of the wheels. Disturbances of high frequency, caused by road holes or block pavement for example, are filtered out by low-pass filters before the values are passed on to the controller. The controller reads them with a given sampling rate and decides how valves and compressor must be operated during the next sampling interval. Escape valve or compressor respectively are turned on if the filtered chassis height is outside an outer tolerance interval for one of the wheels (Fig. 2). For example, if the height exceeds the outer tolerance interval for one wheel, the valve connected to this wheel's suspension and the escape valve are opened. The valves are closed again if the measured chassis levels at all four wheels are below the upper limit of an inner tolerance interval, in addition the filters are reset to the set point, i. e. the desired chassis level. To increase the chassis level the compressor and the valves connected to the suspensions are operated in a similar way.

As compressor and escape valve must not be used simultaneously, priority is given to the compressor in those cases where both would have to be used.

The actual values of the tolerance intervals are determined by the mode the car is in. The modes we regard in this case study are "car is stopped, engine off", "car is driving, engine on" and "car is driving through a bend". The last one is especially interesting, as compressor and escape valve must be switched off when the car is driving through a curve. The original study of BMW contains some more modes which mainly differ in the actual values for the inner and outer tolerance intervals.

3 LINEAR HYBRID AUTOMATA

3.1 Hybrid automata

A hybrid automaton A is a 10-tuple $(X, V, inv, init, flow, E, upd, jump, L, sync)$ with the following meaning (for a detailed introduction see [1]):

X is a finite ordered set $\{x_1, ..., x_n\}$ of variables over $\mathbb{R}$. A valuation s is a point in $\mathbb{R}^n$, the value of variable x_i in valuation s is the i-th component of s, s_i. When Φ is a predicate over the variables in X, we write $[\![\Phi]\!]$ to denote the set of valuations s for which $\Phi[X := s]$ is true.

V is a finite set of locations. A state of automaton A is a tuple (v, s) consisting of a location $v \in V$ and a valuation s. A set of states is called a region. Locations are represented as rectangles in our automata diagrams.

The automaton of Fig. 4, for example, has only one location named *idle*.

inv is a mapping from the set of locations V to predicates over the variables in X. $inv(v)$ is the invariant of location v. When control is in v, only valuations $s \in [\![inv(v)]\!]$ are allowed. In our graphical representation of hybrid automata invariants *True* are usually omitted.

For instance, location *timer* of the automaton of Fig. 5 has invariant $t \leq t\_sample$.

init is a mapping from the locations in V to predicates over X. $init(v)$ is called the initial condition of v. (v, s) is an initial state if $init(v)$ and $inv(v)$ are true for s. Initial conditions are expressed as incoming arrows marked with the condition in our automata diagrams. If the initial condition is *False*, the arrow is omitted.

Location *idle* of Fig. 4 has initial condition $f = sp$.

flow is a mapping from the locations in V to predicates over $X \cup \dot{X}$, where $\dot{X} = \{\dot{x}_1, ..., \dot{x}_n\}$ and $\dot{x}_i$ denotes the first time derivative of x_i, dx_i/dt. When control is in location v the variables evolve according to differentiable functions which satisfy the flow condition $flow(v)$.

Formally the δ time-step relation $\xrightarrow{\delta}$ is defined as $(v, s) \xrightarrow{\delta} (v, s')$ iff there is a differentiable function $\rho : [0, \delta] \to \mathbb{R}^n$ with

- $\rho(0) = s$ and $\rho(\delta) = s'$
- the invariant of v is satisfied: $\rho(t) \in [\![inv(v)]\!]$ for $t \in [0, \delta]$
- the flow condition is satisfied: $flow(v)[X, \dot{X} := \rho(t), \dot{\rho}(t)]$ is true for $t \in [0, \delta]$, where $\dot{\rho}(t) = (\dot{\rho}_1(t), ..., \dot{\rho}_n(t))$.

The filter automaton of Fig. 4 has flow condition $\dot{f} = \frac{1}{T}(h - f)$.

$E \subseteq V \times V$ is a finite multiset of edges, called transitions. A transition $(v, v') \in E$ has source location v and target location v'.[1] In our automata

[1] E is a multiset, because there may be several transitions connecting the same two locations. This is needed, for instance, to express that the same target location can be entered from the same source location under different conditions.

diagrams transitions are represented by arrows between the corresponding locations.

Our automaton of Fig. 4 has only one transition from *idle* to itself.

upd is a mapping from E to the power set of X. $upd(e)$ is called the update set of e. Variables in $upd(e)$ can change their value when transition e is taken.

The transition of the automaton of Fig. 4 has update set $\{f\}$.

jump is a mapping from E to jump conditions. The jump condition $jump(e)$ of transition e is a predicate over $X \cup upd'(e)$, where $upd'(e) = \{y_1', ..., y_k'\}$ for $upd(e) = \{y_1, ..., y_k\}$. y_i' stands for the value of variable y_i after the transition is taken, while y_i refers to the value before the transition.
The transition-step relation $\xrightarrow{e}$ is defined as $(v, s) \xrightarrow{e} (v', s')$ iff

- $e = (v, v')$
- the invariants are satisfied: $s \in [\![inv(v)]\!]$ and $s' \in [\![inv(v')]\!]$
- the variables in $upd(e)$ change according to the jump condition: $jump(e)[X, upd'(e) = s, s'[upd(e)]]$ is true, where $s'[upd(e)]$ is the restriction of s' to the variables in $upd(e)$.
- the variables not in $upd(e)$ remain constant: $s_i = s_i'$ for $x_i \in X \setminus upd(e)$.

Transition steps are discrete, no time passes while they are taken.
In our automata diagrams we write guarded assignments $\Phi \rightarrow y_i := c$ for update set $\{y_i\}$ and jump condition $\Phi \wedge y_i' = c$. Guards *True* are omitted.

The transition of the automaton of Fig. 4 has jump condition $True \rightarrow f := sp$ (written as a guarded assignment).

L is a finite set of synchronization labels. Parallel automata which have a common synchronization label must take transitions with this label simultaneously. Transitions with different labels are interleaved.

sync is a mapping from E to $L \cup \{\tau_A\}$. $sync(e)$ is the synchronization label of transition e. The label τ_A may only be used in automaton A and marks transitions of A which are not synchronized with transitions in parallel automata. In automata diagrams each transition e is labeled with $sync(e)$ and $jump(e)$, the label τ_A is omitted.

The transition of the automaton of Fig. 4 has synchronization label set_f.

All of the above mappings are total.

Parallel composition
Parallel composition of hybrid automata can be used for specifying larger systems. A hybrid automaton is given for each part of the system, communication between the components may occur via shared variables and synchronization labels. Technically the parallel composition of hybrid automata is obtained by constructing a product automaton. It has the carthesian product of the individual automata's locations as its locations.

3.2 Verification of linear hybrid automta

Hybrid model checking [1] allows to verify properties about *linear* hybrid automata, a subclass of hybrid automata, in a fully automatic way.

A hybrid automaton A is *linear* if all its invariants and initial conditions are convex linear predicates over X, all flow conditions are convex linear predicates over $\dot{X}$ and all jump conditions are convex linear predicates over $X \cup X'$, where $X' = \{x_1', ..., x_n'\}$. A predicate over variables in a set Y is linear if it is an (in)equality between linear terms over Y. A linear term over Y is a linear combination over Y with rational coefficients. A predicate is a convex linear predicate if it is a finite conjunction of linear predicates. This definition implies that a linear differential equation cannot necessarily be converted into a linear flow condition. The notions of *linear* for differential equations and hybrid automata are different.

The model checking algorithm in [1], as e.g. implemented in HyTech, analyzes a property by checking if it is violated in any state reachable by the automaton. Therefore, it computes the set of states of the linear hybrid automaton which are reachable from a set of initial states by repeatedly applying time- and transition-steps. It can also perform backward reachability analysis in which the region is computed from which a given final region can be reached by iterating time- and transition-steps. The algorithm is a semi-decidable procedure, i.e. termination is not guaranteed.

As the algorithm is restricted to linear hybrid automata, nonlinear hybrid automata have to be approximated. A bunch of techniques to obtain linear hybrid automata from nonlinear ones or from ordinary differential equations have been developed in the literature, see [5, 7, 12]. In this paper we use linear phase portrait approximation of [7], which is conservative in the sense that every property that holds for the abstraction holds also for the original automaton.

3.3 Tool support for model checking

Several hybrid model checkers have been developed, amongst whom HyTech, KRONOS and UPPAAL are the most prominent ones. See [3] for a survey. We selected HyTech [6], as it provides the most general input language by supporting the full scope of linear hybrid automata. This seems to be an obvious choice as we want to apply as few approximations as possible. UPPAAL, for example, is restricted to *timed* automata which especially implies that all flow conditions have to be constants.

HyTech allows the analysis of models containing parameters and can thereby be used to synthesize constraints on these parameters which are necessary for the correctness of the system. Monitor automata may be used to prove complex properties of hybrid systems. Furthermore, sequences of time- and transition-steps leading from one region to another can be generated, which is very useful for finding out why given predicates are violated.

4 MODELLING THE AUTOMOTIVE SYSTEM

Next we present a model of the EHC using (linear) hybrid automata. After that we draw some conclusions on hybrid automata as a specification technique for hybrid systems. In order to be able to apply automatic verification techniques to the EHC we limit its complexity by modelling only one wheel in the following. Additionally, this allows to develop a hybrid automata model of the system which is easier to comprehend.

4.1 The model

The EHC can be decomposed into a filter, a timer, a controller, two valves and a compressor (the actors), the plant, i.e. the physical behavior of the pneumatic suspension system, and into disturbances caused by the environment (Fig. 3).

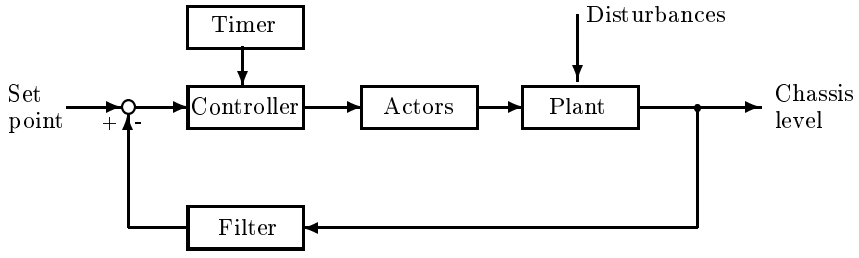


Fig. 3. Basic structure of the EHC.

In order to elaborate those characteristics of linear hybrid automata which are particularly relevant for modelling hybrid systems, we choose not to model valves and compressor explicitly. Instead, their effect on the plant will be contained in the linear hybrid automaton for the controller. Furthermore, the plant and disturbances are integrated into one automaton. A more detailed model of the EHC can be found in [10].

The filter

The filter reads the current chassis level h and outputs the filtered chassis level f. It can be modeled by the hybrid automaton of Fig. 4. The flow condition in the diagram is the usual differential equation of a filter, T is its time constant. The synchronization label set_f is used by the controller to reset the filter to the set point sp.

Thus the automaton of Fig. 4 models the same behavior of f as the differential equation $\dot{f} = \frac{1}{T}(h - f)$ with initial value $f = sp$, but extends this continuous specification by allowing a discrete reset of the filter.

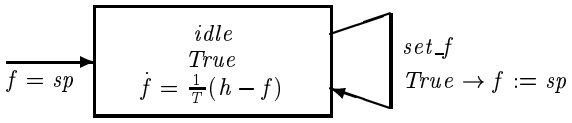


Fig. 4. Hybrid automaton for the filter.

Approximation of the filter

Unfortunately the automaton of Fig. 4 is not a linear hybrid automaton, because its flow condition is not a convex linear predicate over $\{\dot{f}, \dot{h}\}$.[2] To get a linear hybrid automaton, an approximation technique has to be applied. In our case linear phase-portrait approximation, defined in [7], can be used. This technique basically consists of the following steps:

A partitioning is chosen for the set of valuations permitted by the original location's invariant. The approximating automaton contains one new location for each partition then. The flow condition of such a new location is obtained by conservatively approximating that of the original location, e.g. by computing the extremal values of the original flow condition in the respective partition. To ensure correctness of the approximation, transitions of the original automaton must then be duplicated so that every new location can be entered and left in the same way as the original location. Furthermore, new transitions must be added which allow control to pass freely between the copies of the original location.

[2] Nevertheless this flow condition it is a linear differential equation.

[12] uses a related idea to directly obtain a linear hybrid automaton from a set of ordinary differential equations.

The partitioning we use for location *idle* is shown in Table 1. $h - f$ has unit millimeters, $\dot{f}$ has unit millimeters per second. The flow conditions were computed for a time constant $T = 2s$.

Location	Invariant	Flow condition
A	$h - f \in (-\infty, -10]$	$\dot{f} \in (-\infty, -5]$
B	$h - f \in [-10, -6]$	$\dot{f} \in [-5, -3]$
C	$h - f \in [-6, 0]$	$\dot{f} \in [-3, 0]$
D	$h - f \in [0, 6]$	$\dot{f} \in [0, 3]$
E	$h - f \in [6, 20]$	$\dot{f} \in [6, 10]$
F	$h - f \in [20, \infty)$	$\dot{f} \in [10, \infty)$

Table 1. Partitioning of location *idle*.

The approximating automaton has a *set_f* transition with jump condition $f' = sp$ from every location in $\{A, ..., F\}$ to every location in $\{A, ..., F\}$. Furthermore there are τ_{filter} transitions with jump condition *True* from every location of the approximating filter to every other location of it to allow control to pass freely between them.

Linear phase-portrait approximation is a very general technique, its approximation does not depend upon the order of the differential equation to be approximated. Unfortunately, there is no standard method for finding a partitioning for a given differential equation. In principle any partitioning may be used. In order to do automatic verification, however, it is crucial to keep the state space as small as possible. The choice of the presented partitioning for the filter was guided by this requirement. Of course we could have selected individual partitionings for f and h instead of partitioning $h - f$. However, due to the form of the original flow condition this would have resulted in a higher number of locations if we wanted to obtain the same quality of approximation.

The approximation techniques cited above are defined for individual hybrid automata. Care has to be taken, when they are applied to a single automaton of a parallel composition, as in our case: In Section 5.1 we use a test automaton that changes the value of variable h in a transition step in order to analyse the EHC's response to a step-like disturbance. Imagine the approximating filter automaton is in location D with $h = 5$ and $f = 1$. If the test automaton now wants to increase h by e.g. 4 in a transition, it cannot do so, because the invariant of location D of the approximating filter automaton does not permit a value of $h - f = 8$. However, if the test automaton is running in parallel with the original filter automaton, the transition is possible, as the invariant of the filter's *idle* location does not depend on the value of h.

A situation like this one can occur whenever an approximating automaton is

used in parallel with an automaton which modifies one of the variables occurring in the new invariants, but not in the invariant of the original, unpartitioned location, in a transition step. To overcome this problem linear phase-portrait approximation from [7] was extended to the parallel composition of hybrid automata in [10]. The idea of this extension is to augment the approximating automaton so that it synchronizes with the discrete transition which would falsify its current location's invariant. In this synchronization control changes to another location. For our example this means that it is necessary to add transitions from every location in $\{A, \ldots, F\}$ to every location in $\{A, \ldots, F\}$ which have jump condition *True* and which have the same synchronization label as the transition in the test automaton.

The timer

The timer issues signals with a certain rate and thereby tells the controller when it has to read the filter value and to make a control decision. It is described by the automaton in Fig. 5.

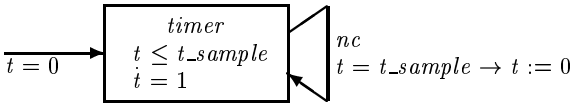


Fig. 5. Hybrid automaton for the timer.

The transition with synchronization label nc is taken every t_sample time units. It is used to force the controller, which also uses label nc, to take an appropriate transition every t_sample time units.

The controller

The controller reads the filtered chassis level, determines the control mode and operates valves and compressor whenever the timer takes its nc transition. It is defined by the hybrid automaton in Fig. 6.

In this automaton we model only two modes, *driving* and *bend*. Mode *stopped*, as well as the other modes, is similar to *driving* and could be integrated easily.

$[sp + otl_d, sp + otu_d]$ is the outer tolerance interval, $[sp + itl_d, sp + itu_d]$ is the inner tolerance interval in mode *driving*. Variable u is needed to specify that locations tr_1 and tr_2 are left immediately after entry. These location are always entered with $u = 0$. As the invariant demands that u remains zero and as the flow condition demands that it increases with time, no time may elapse in them. Therefore they are called transient locations.

$\dot{c}$ is used to model the actors' influence on the chassis level. The exact effect escape valve and compressor have on the chassis level depends on many factors

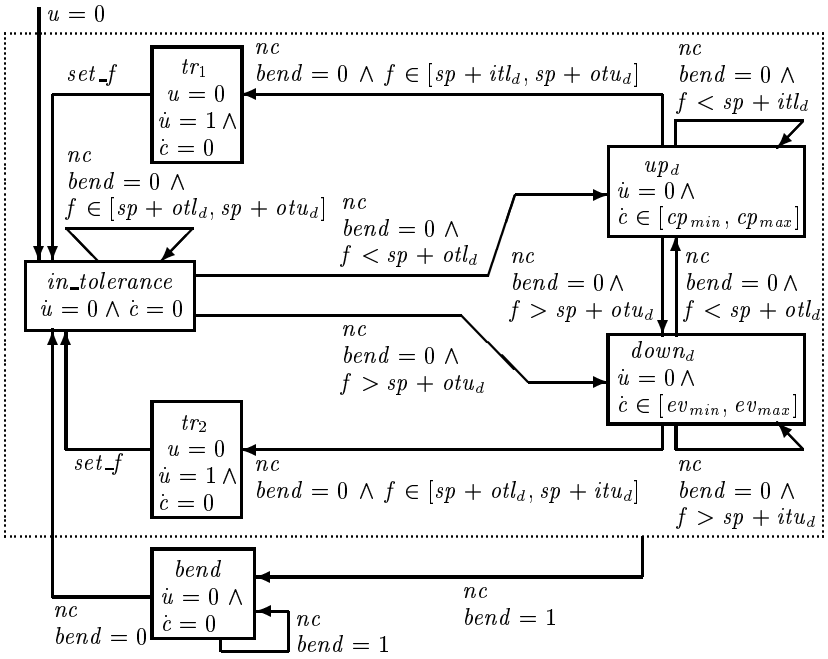


Fig. 6. Hybrid automaton for the controller.

which we do not want to model explicitly. Therefore we only state that usage of the compressor increases the chassis level while usage of the escape valve decreases it with a rate in a certain interval.

The central locations of the controller automaton are *in_tolerance*, up_d, $down_d$ and *bend*. The dotted box is used to indicate that every location in it has a *nc* transition with jump condition $bend = 1$ to location *bend*. This models that suspending control in curves has priority over keeping the chassis level within the tolerance intervals.

In location up_d the compressor is on and the valve connected to the suspension of the wheel is open, the chassis level is therefore increased by $\dot{c} \in [cp_{min}, cp_{max}]$. In location $down_d$ the escape valve and the valve to the wheel's suspension are open, the chassis level is decreased by $\dot{c} \in [ev_{min}, ev_{max}]$. In *in_tolerance* and in *bend* the compressor is off and both valves are closed, the controller does not influence the chassis level, $\dot{c} = 0$.

Starting from the initial state, the controller has to wait in *in_tolerance* until the timer takes its *nc* transition. Then it reads the filtered chassis height

f and goes to location up_d or $down_d$ if f is outside the outer tolerance interval, otherwise it reenters *in_tolerance*. Location up_d is left when the timer makes a *nc* transition if f is no longer smaller than the lower limit of the inner tolerance interval $sp + itl_d$, otherwise it is reentered. If f is greater than the outer upper tolerance limit $sp + otu_d$ now, $down_d$ has to be entered. Otherwise the transition to tr_1 is taken. As it is transient, tr_1 is left immediately after entry, the filter is reset by the *set_f* transition and *in_tolerance* is entered. Location $down_d$ works in a similar way.

Except of the *set_f* transitions, all these transitions are only possible if $bend = 0$. If the controller notices that the car is in a curve ($bend = 1$), location *bend* is entered. In this location the value of variable *bend* is read whenever the timer takes its *nc* transition. When $bend = 0$ again, *in_tolerance* is entered.

The plant and disturbances

Within the automaton in Fig. 7 we model the physical behavior of the pneumatic suspension system, the disturbances we want to regard and the occurrence of curves.

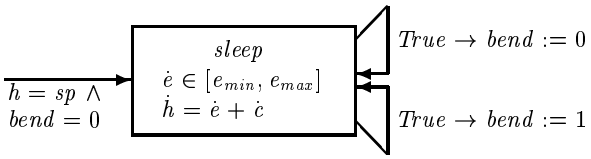


Fig. 7. Hybrid automaton for the plant and disturbances.

By modifying variable *bend* the transitions in Fig. 7 model that the car may enter and leave a curve anytime.

The measured chassis level, which is the relevant physical quantity of the suspension system, is influenced by disturbances from the outside world, e.g. the road, and by escape valve and compressor. As we cannot make any predictions on when a disturbance will occur and how strong it will be, we say that disturbances of limited strength may occur anytime. There is no sense in allowing disturbances of unlimited strength, since the real system, i.e. the stability of the car, is limited.

We specify that the rate of change of the chassis level at one wheel $\dot{h}$ is the sum of the changes due to disturbances, denoted by $\dot{e}$, and the changes due to compressor and escape valve, $\dot{c}$ (Fig. 7). In order to be able to prove upper and lower bounds for the chassis level in the presence of disturbances it is necessary to demand that disturbances cannot be stronger than compressor and escape valve respectively, i.e. $e_{min} \geq ev_{max}$ and $e_{max} \leq cp_{min}$. If the disturbances

were stronger, the controller could not ensure bounds for the chassis level even if it operated correctly. In our system we set $e_{min} = ev_{max}$ and $e_{max} = cp_{min}$.

The influence of the environment is of course seriously limited this way. In reality disturbances stronger than compressor and escape valve are certainly possible. In a more realistic model of the environment a stochastic characterization of disturbances would be used. Stochastic models, however, go beyond the expressiveness of hybrid automata.

4.2 Discussion

Advantages of hybrid automata
In this case study hybrid automata turned out to be an advantageous formalism for modelling hybrid systems. Their high expressiveness allows to merge the specification of discrete and continuous aspects freely. It is not necessary to introduce an artificial division between aspects of both worlds where there is none in the real system. Nevertheless such a division can be made if it seems to be appropriate.

In several more traditional approaches to modelling hybrid systems discrete and continuous specification techniques like statecharts and block diagrams are used in parallel. For example, the basis for this case study was a specification with Statemate statecharts and Matrix_X block diagrams. Thus the system had to be split into a discrete and a continuous part. However, a filter with discrete reset operation is a good example for a system which cannot naturally be split into a discrete and a continuous part. Nevertheless, it can easily be modelled using (nonlinear) hybrid automata (see Section 4.1). Furthermore these traditional approaches usually do not have a common semantic foundation for their discrete and continuous specification techniques. Therefore formal analysis, like model checking, is impossible.

Deficits encountered
Despite of their great advantages hybrid automata do have some deficits in detail.

Firstly, they are not compositional. If we have a system of two parallel hybrid automata, A and B, and automaton A' has the same set of reachable states as A this does not imply that the system consisting of A' in parallel to B behaves in the same way as A in parallel to B. In particular it does not suffice to approximate only one component of a parallel composition of hybrid automata, which we outlined in the section on the filter automaton. Furthermore, modular verification is impossible. Compositionality for hybrid systems is difficult to achieve. Work in this direction can e.g. be found in [2].

Secondly, hierarchic locations as they are known from statecharts [4] would be useful to specify larger systems. In Fig. 6 we used the dotted box to increase clarity of the diagram by reducing the number of transitions in it. This box can be interpreted as a hierarchic location, but it is only a syntactic means here. The case study in [8] contains a further example which stresses the utility of a hierarchy concept for hybrid automata. As long as hierarchic locations are pure syntax they could easily be integrated into the basic model of hybrid automata.

Another inconvenience in the specification of hybrid systems with hybrid automata arises, because a transition may only have one synchronization label. In the controller automaton of Fig. 6 we had to insert the transient locations tr_1 and tr_2 in order to model the synchronization of timer, controller and filter when the controller locations up_d or $down_d$ are left in order to enter *in_tolerance*. If one transition could have more than one label, we could have modeled the same behavior by simply labeling the transitions from up_d and $down_d$ to *in_tolerance* with *nc* and *set_f*, like it is possible in statecharts. However, in a model with this feature we would naturally want to distinguish between input and output labels. Together with synchronous communication this leads to a micro-/macrostep semantics like in statecharts with all its associated problems (see e.g. [13]).

For the novice user the concept of invariants in hybrid automata can lead to subtle errors. If for example an invariant demands that no more than t time units may elapse in a location, but the location cannot be left within this time, the system stops execution. For a reactive system this is not desirable, hence we consider it to be an error in the specification. As these kinds of errors are difficult to detect, the invariants in hybrid automata must be used carefully in order to avoid such an implicit form of termination.

Some of these weaknesses of hybrid automata as a specification language may be attributed to the fact that they are a compromise between a practical specification technique and a formalism which enables automatic verification.

5 ANALYZING THE AUTOMOTIVE SYSTEM

In the following we report on the verification of several safety-critical properties of the EHC using HyTech. First, four properties and their model-checking results are described (Subsection 5.1). Then, the model-checking experiences are discussed more generally (Subsection 5.2).

The verification results were obtained for a slightly optimized and modified specification of the EHC. The basic version used for verification differs from that in section 4.1 by including the control modes *stopped* and *driving*, but excluding mode *bend*. The exact model can be found in [11].

5.1 Verification results

Bounds for the chassis level
A first essential requirement is to show that the system keeps the chassis level within certain bounds and to identify these bounds. HyTech computes that the outer tolerance limits $-40mm$ and $20mm$ are never exceeded by more than $7mm$ for a sampling rate of once every second, $t\_sample = 1s$. The computation takes 62 minutes[3] of CPU time. Because of complexity problems it was not possible to choose a more realistic sampling rate.

Escape valve and compressor
It is essential that escape valve and compressor are never used at the same time. To examine this property it is necessary to include escape valve and compressor into the model. They are realized by trivial hybrid automata that toggle between the states on/off and closed/open, respectively. HyTech proves the desired property for the resulting model in 58 minutes.

Suspending control in curves
Next we prove that the system does not try to change the chassis level when the car is driving through a curve. To treat this safety-critical property we reinclude curves in our model of the EHC. Furthermore, the system is augmented by a monitor automaton[4] which measures the time between a curve being entered and control being suspended or the curve being left again, whatever occurs first. HyTech analyzes the resulting model in 73 seconds and verifies that control is suspended at most *t_sample* time units after a curve started, if its duration is longer than *t_sample* time units.

Step response of the EHC
In the following we analyze a property related to stability: We examine whether the controller reaches *in_tolerance* again some time after a disturbance or whether it can continue to use compressor and escape valve infinitely by e.g. operating them alternately. To examine this the system response to a step-like disturbance is regarded. We are aware of the fact that control systems engineering provides more powerful methods concerning stability. The intention is to apply *full automation* to a limited example.

We replace the hybrid automaton for the plant and disturbances by a model in which we assume that there are no continuous disturbances. Instead the new automaton specifies that h is modified discretely in a transition with jump

[3] All performance figures were obtained on a Sun Sparcstation 20 with 128 MB RAM and 350 MB swap space.

[4] A monitor automaton is a further automaton which is added via a parallel composition in order to observe some critical properties.

condition $h' = h + j$ at some point in time. For complexity reasons we cannot regard arbitrary values of j. As an example we therefore restrict j to lie within a certain interval for which the controller is expected to use the escape valve. As outlined in section 4.1, the approximating filter automaton is also modified to work correctly with this automaton.

By adding a monitor automaton to this model we use HyTech's parametric analysis facility to compute that the controller leaves location *in_tolerance* at most $4.3s$ after the disturbance and reenters it after at most $22.3s$. The chassis level then lies in $[-1mm, 6mm]$ and *in_tolerance* is not left again. The analysis takes 370 seconds.

5.2 Discussion of hybrid model checking

A great advantage of model-checking is its full *automation*. However, because of the efficiency limits of the tool we had to restrict the model substantially in order to analyze vital properties of the electronic height control system. This diminished the automation potential, as finding the right abstractions is a manual and error-prone process. There are several reasons for the insufficient scalability.

First, running out of memory is a serious limiting factor in practice, whereas the theoretical result that the algorithm does not necessarily terminate only plays an inessential role. This space restriction seems not to be caused by a specific tool implementation, but to be inherently related to the algorithm's complexity and the representation of the data. For model-checking of discrete systems a special encoding of the state space by BDDs produced an immense increase in efficiency. Therefore, data encoding techniques should be an important field of further research. The complexity problem should also be overcome by a compositional automata theory, which would allow to reduce the complexity of the whole model to the size of the single components to be verified. Remember the last section for a discussion of such matters. Furthermore, nonlinear hybrid automata and the necessarily applied approximation techniques increase complexity substantially. In our case, the filter approximation caused the main complexity of the entire model.

Second, the present version of HyTech (1.04) is not yet efficiently implemented. Standard arithmetic libraries, which are included in HyTech often cause arithmetic overflows. These overflows are a severe restriction in practice. The libraries are very sensitive to the specific values of the constants in the model. As the user does not get any feedback on which constants have caused the overflow, trial and error must be used to identify them. In our models the use of fractions turned out to be disadvantageous as far as overflows are concerned. Therefore we only use integer constants in the current models. The problem of overflows arises because HyTech uses integer arithmetics without

any roundoff mechanism. IF standard libraries with floating point arithmetics would be used, the overflows would vanish. However, in this case the correctness of the result can no longer be guaranteed because of the roundoff errors.

6 CONCLUSION

We successfully verified several safety-critical properties for an abstraction of an automotive control system using the hybrid model checker HyTech. Compared to other real-life case studies performed with HyTech [9, 8], the electronic height control system seems to be distinctly more complex than those because of the approximation necessary for the filter and the complex environment. Nevertheless, the system is quite simple and some of the properties seem to be obvious by inspection. One should recognize, however, that the properties verified are interesting to be automatically checked for more complex systems.

Up to our knowledge, this is the first article with a general discussion of hybrid automata and hybrid model checking so far. Based on this discussion we draw the following conclusions:

Hybrid automata represent a remarkable compromise between the specification and verification of hybrid systems. The specification formalism allows to integrate discrete and continous components in one framework in a natural and semantically correct way. Nevertheless, it supports despcription techniques — based on differential equations and state descriptions —, which are familiar to both control engineers and software engineers. The automatic verification method is restricted to *linear* hybrid automata, but conservative approximation techniques allow to apply it to nonlinear hybrid automata as well.

However, every compromise demands a tribute from both sides. On the specification side, the restriction to linear hybrid automata is unnatural im many cases. Furthermore, for larger systems better facilities to structure the system, in particular modularity and hierarchy, are necessary. Last but not least, modelling the physical plant often even goes beyond the expressiveness of nonlinear hybrid automata, stochastic description elements would be useful.

On the verification side, however, linear hybrid automata already raise significant scalability problems. Probably, this cannot be essentially improved by optimizing specific tool implementations, which are nevertheless necessary. Even worse, approximations due to nonlinear hybrid automata cause a state explosion, which is hardly manageable in industrial-scale systems. Due to the complexity results in [1] a semi-decidable model-checking algorithm for nonlinear hybrid automata is unlikely. Nevertheless the present situation could be improved by developing further approximation techniques and by identifying

further classes of hybrid automata for which efficient model checking algorithms exist.

A first important step in future work should be to turn hybrid automata into a compositional description technique. This would significantly compensate some of the drawbacks mentioned for both specification and verification.

ACKNOWLEDGEMENTS

We thank BMW AG for their permission to report on the electronic height control system and their excellent introduction into this case study.

REFERENCES

1. Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T.A., Ho, P.-H., Nicollin, X., Olivero, A., Sifakis, J., and Yovine, S.: The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138:3–34, 1995.
2. Alur, R., Henzinger, T.A.: Modularity for timed and hybrid systems. In *CONCUR 97: Concurrency Theory*, Lecture Notes in Computer Science 1243, pages 74–88. Springer-Verlag, 1997.
3. Alur, R., Henzinger, T.A., Sontag, E.D.: *Hybrid Systems III*. Lecture Notes in Computer Science 1066. Springer Verlag, 1996.
4. Harel, D.: Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, 8(3):231–274, March 1987.
5. Henzinger, T.A., Ho, P.-H.: Algorithmic analysis of nonlinear hybrid systems. In P. Wolper, editor, *CAV 95: Computer-aided Verification*, Lecture Notes in Computer Science 939, pages 225–238. Springer-Verlag, 1995.
6. Henzinger, T.A., Ho, P.-H., Wong-Toi, H.: A user guide to HYTECH. In E. Brinksma, W.R. Cleaveland, K.G. Larsen, T. Margaria, and B. Steffen, editors, *TACAS 95: Tools and Algorithms for the Construction and Analysis of Systems*, Lecture Notes in Computer Science 1019, pages 41–71. Springer-Verlag, 1995.
7. Henzinger, T.A., Wong-Toi, H.: Linear phase-portrait approximations for nonlinear hybrid systems. In R. Alur, T.A. Henzinger, and E.D. Sontag, editors, *Hybrid Systems III*, Lecture Notes in Computer Science 1066, pages 377–388. Springer-Verlag, 1996.
8. Henzinger, T.A., Wong-Toi, H.: Using HYTECH to synthesize control parameters for a steam boiler. In J.-R. Abrial, E. Börger, and H. Langmaack, editors, *Formal Methods for Industrial Applications: Specifying and Programming the Steam Boiler Control*, Lecture Notes in Computer Science 1165. Springer-Verlag, 1996.
9. Ho, P.-H., Wong-Toi, H.: Automated analysis of an audio control protocol. In P. Wolper, editor, *CAV 95: Computer-aided Verification*, Lecture Notes in Computer Science 939, pages 381–394. Springer-Verlag, 1995.
10. Stauner, T.: *Specification and Verification of an Electronic Height Control System using Hybrid Automata*. Master thesis, Technische Universität München, 1997.

11. Stauner, T., Müller, O., Fuchs, M.: Using HYTECH to verify an automotive control system. In *Proc. Int. Workshop on Hybrid and Real-Time Systems (HART'97)*, volume 1201 of *Lecture Notes in Computer Science*, pages 139–153. Springer-Verlag, 1997.
12. Stursberg, O., Kowalewski, S., Hoffmann, I., Preußig, J.: Comparing timed and hybrid automata as approximations of continuous systems. In *Hybrid Systems IV*, volume 1273 of *Lecture Notes in Computer Science*. Springer-Verlag, 1997.
13. von der Beeck, M.: A Comparison of Statecharts Variants. In H. Langmaack, W.-P. de Roever, and J. Vytopil, editors, *Proc. Formal Techniques in Real–Time and Fault–Tolerant Systems (FTRTFT'94)*, volume 863 of *Lecture Notes in Computer Science*, pages 128 – 148. Springer, 1994.