# VISUAL DESCRIPTION OF HYBRID SYSTEMS

**Radu Grosu, Thomas Stauner** [1]

*Institut für Informatik, TU München, D-80290 München, Germany, Email: {grosu,stauner}@informatik.tu-muenchen.de*

Abstract: The design of hybrid systems usually involves engineers from a number of different engineering disciplines. Hence, specification techniques are needed that can intuitively be understood by people from all these engineering communities. Furthermore, a formal foundation for these techniques in necessary to prohibit ambiguities which may be fatal in the safety critical environment of many hybrid systems. In this paper we present an example development scenario in order to demonstrate the use of *HyCharts*, a formalism that satisfies both these needs. The semantic foundation of HyCharts is outlined briefly.

Keywords: hybrid systems, specification, formal methods, requirements engineering

## 1. INTRODUCTION

Hybrid systems are dynamical systems whose behavior is characterized by both, discrete and continuous aspects. Typical examples are embedded real-time systems interacting with their physical environment.

In the past few years a number of formalisms have been proposed for the specification and verification of such systems. (Alur *et al.*, 1996) is a good starting point in this subject. However, the practical utility of almost all these formalisms is restricted either because they do not permit modular specification, like e.g. hybrid automata (Alur *et al.*, 1995), or because there is no convenient graphical representation for them, like e.g. for hybrid I/O automata (Lynch *et al.*, 1996).

In this work we show how HyACharts and HySCharts, two complementary visual description techniques for hybrid systems that achieve both these aims, can support the methodical development of hybrid systems. *HyACharts* allow the hierarchic specification of system architecture. Hybrid behavior is specified with hybrid hierarchic state transition diagrams (*HySCharts*). HySCharts may be regarded as a hybrid extension of ROOMcharts (Selic *et al.*, 1994). The components in both diagram types have clearly defined interfaces. Thus, side-effects are eliminated and modularity is obtained. HyACharts and HySCharts can be seen as hierarchic graphs. Their semantics can therefore be defined by instantiating the operators of an abstract algebra for these graphs appropriately.

This paper is organized as follows. We start with an example that gives an intuitive impression of HyCharts and shows their use in the development of hybrid systems. In particular the example sketches how HyCharts aid a methodology for the design of hybrid systems that is based on refining an abstract, hybrid model to more concrete models that are closer to implementation. An introduction to our underlying model of hybrid systems follows. After that, we outline the connection of HyCharts to abstract hierarchic graphs and indicate how this connection serves to define the semantics of HyCharts. Finally we draw some conclusions.

[1] The second author was supported with funds of the Deutsche Forschungsgemeinschaft under reference number Br 887/9-1 within the priority program *Design and design methodology of embedded systems.*

## 2. SYSTEM DEVELOPMENT WITH HYCHARTS: AN EXAMPLE

In this section we demonstrate by means of an example how HyCharts can be applied in the methodical development of hybrid systems. Basic concepts of the charts are explained informally.

As example we use an electronic height control system (EHC), taken from a former case study together with BMW. The purpose of this system is to control the chassis level of an automobile by a pneumatic suspension. The abstract model of this system which regards only one wheel was first presented in (Stauner *et al.*, 1997). Its informal requirements are as follows:

(1) Whenever the chassis level $sHeight$ is outside a certain tolerance interval, it has to be increased or decreased in order to get close to the center of the interval again. (2) Very short deviations from the tolerance interval should not be compensated. (3) After a compensation, some time should pass before the same actuator, namely a compressor that can increase the chassis level and an escape valve that can decrease it, is switched on again. (4) In any case the chassis level may not be modified by the controller when the sensor *bend* signals that the car is going through a curve.

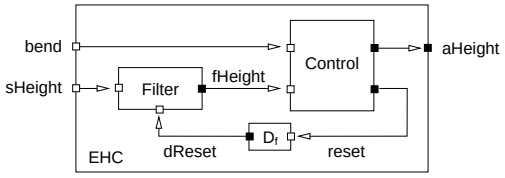


Fig. 1. HyAChart of the EHC.

In the development steps of our example we want to focus on the control component of the EHC. We therefore assume the initial architecture of the system given. It is depicted as a HyAChart in Figure 1. As we do not regard the environment here, the basic components of the *system* are a filter and the controller. The filter eliminates high-frequency disturbances in $sHeight$ and thus helps to satisfy requirement (2). It may be reset to a defined zero level inside the tolerance interval. As we will see the controller uses this to fulfill requirement (3). The escape valve and the compressor are modeled within the controller. Variable $aHeight$ models their influence on the chassis level. The component labeled $D_f$ introduces a delay and ensures that the feedback between the filter and the controller is well-defined.

The behavior of a component is characterized, as intuitively shown in Figure 2 by periods where the values on the channels change smoothly and by time instances at which there are discontinuities. In our approach the smooth periods result from the analog parts of the components. The

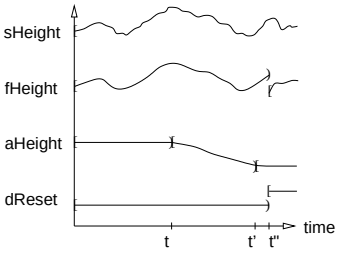


Fig. 2. A typical evolution of the EHC.

discontinuities are caused by their combinational (or discrete) parts (see Section 3).

We specify the behavior of both the combinational and the analog part of a component by a HySChart, i.e., by a hybrid, hierarchic state transition diagram, with states marked by activities and transitions marked by actions. The transitions define the discontinuities, i.e., the instantaneous actions performed by the combinational part. The activities define the smooth periods, i.e., the time consuming behavior of the analog part while the combinational part idles.

### 2.1 *Initial Design of the EHC's Controller*

In the development of the controller we start with the two most fundamental states of the EHC, *inBend* and *outBend*. Figure 3, left, shows the HySChart of the controller at this first level of decomposition.

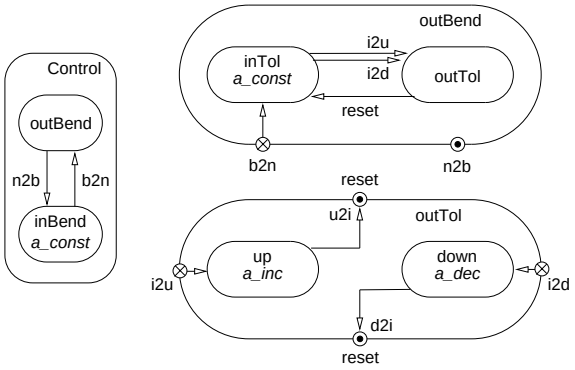


Fig. 3. The EHC's *Control* component (without sampling).

The labels of the transitions refer to *actions*, i.e. to predicates that specify when a transition is enabled and how variables change when it is taken. In these predicates we use the following conventions. A backprimed variable $x$‘ denotes the current input, an unprimed variable $x$ denotes the stored value of $x$ and a primed variable $x'$ denotes the updated value of $x$. Transitions are taken as soon as they are enabled. The actions in Figure 3, left, sense whether signal *bend* is set or not. They are both defined as $n2b \equiv b2n \equiv bend?$, where $bend?$ is a shorthand for $bend \neq bend‘ \wedge bend' = bend‘$ which is true if the stored value of bend is

different from the current value ($bend \neq bend`$) and which updates the stored value to the current value ($bend' = bend`$).

The labels written in italics within the states of the figure denote *activities*. They specify the analog part of a component, i.e. they determine how the variables evolve, when control is in the respective state. Activity *a_const* stands for $\frac{d}{dt}aHeight = 0$ and specifies that $aHeight$ remains constant when control is in $inBend$, i.e. that compressor and escape valve are off. No activity is given for $outBend$ as this state will have to be refined further and the evolution of $aHeight$ depends on the concrete substates of $outBend$. This first hierarchic level of the controller ensures that requirements (4) is satisfied.

When the car is not in a curve we need to distinguish whether the chassis level is in the tolerance interval or not. Therefore, the next hierarchic layer of the controller refines state $outBend$ and consists of the two states $inTol$ and $outTol$ (Figure 3, top right). For state $inTol$ we already know that compressor and escape valve need not be operated. Hence $aHeight$ remains constant which again is expressed by activity *a_const* in state $inTol$.

State $outTol$ is further refined into *up* and *down*, as shown in Figure 3, bottom right. The activities in these two states denote that $aHeight$ increases or decreases, respectively. Activity *a_dec* stands for $\frac{d}{dt}aHeight \in [ev_-, ev_+]$ which means that variable $aHeight$ evolves smoothly with its derivative satisfying the given inequation when control is in *down*. Activity *a_inc* stands for $\frac{d}{dt}aHeight \in [cp_-, cp_+]$. The constants $ev_-$ and $ev_+$ are negative, $cp_-$ and $cp_+$ are positive. The actions $i2u$ and $i2d$ test whether the filtered chassis level is below or above the tolerance interval. The actions $u2i$ and $d2i$ test whether it is inside the tolerance interval, and not too close to its lower boundary or upper boundary, respectively, again. Hence, the developed version of the controller fulfills requirement (1) now. In detail these actions stand for the following predicates:

$$i2u \equiv fHeight` \leq lb,$$
$$i2d \equiv fHeight` \geq ub,$$
$$u2i \equiv lb + c \leq fHeight` \leq ub,$$
$$d2i \equiv lb \leq fHeight` \leq ub - c$$

Where $lb$, $ub$ and $c$ are appropriate constants.

Action *reset* sends a reset signal to the filter which causes the filter value $fHeight$ to be set to a predefined zero level inside the tolerance interval. This way the controller ensures that some time passes before $fHeight$ leaves the tolerance interval again and compressor or escape valve are operated again (requirement (3)). The action stands for $reset' = \neg reset$ meaning that the new value of *reset* results from toggling the old value.

The circles at the boundary of Figure 3, top right and bottom right are *interface points*. The HySChart can receive control from or give control to the next higher level of hierarchy there. The labels at the interface points indicate from which transition control is received or to which transition it is given, respectively. Thus, the interface points provide modularity for HySCharts. In the example the interface point labeled $n2b$ is not connected to any substate of $outBend$. This specifies that it can be taken from any substate of it if action $n2b$ is true. Thus, it is a *preemptive* transition. The preemption semantics used in HySCharts is weak preemption, i.e. preemptive transitions may be overwritten at lower levels of hierarchy. For this reason weak preemption is well-suited for refinement. (Note that transition $n2b$ is not overwritten in the presented example.)

## 2.2 *Towards Implementation*

Now that we have an initial design that satisfies the informal requirements, we want to take a step towards implementation. We will now refine the initial design into one in which the filtered chassis level $fHeight$ is not read continuously, but with a certain sampling rate. Choosing a sampling rate is done after a deep understanding of the system. It should not alter the properties we expect the system to satisfy.

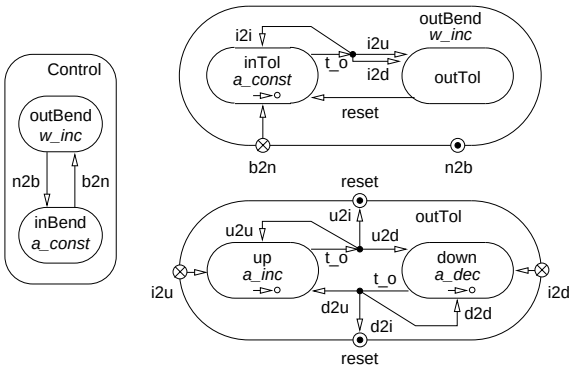


Fig. 4. The EHC's *Control* component (with sampling).

As the controller's reaction to curves is not affected by sampling $fHeight$, we do not expect that the first hierarchic layer of the controller needs to be modified strongly. In fact all changes can be kept local to the substates of $outBend$.

To model sampling we introduce a new variable $w$. In the substates of $outBend$ is must evolve in pace with physical time, i.e. it has a constant derivative of 1, which is specified by activity $w\_inc \equiv \frac{d}{dt}w = 1$. In $inBend$ the evolution of $w$ is irrelevant, nothing is specified there. Figure 4, left, shows the new top level of the HySChart

for the controller. Note that the only change was made in state *outBend* and is local to this state.

The controller with sampling is only allowed to read the filtered chassis level at the end of a sampling interval. For the HySChart this means that all the transitions inside *outBend* may only be taken when $w$ reaches a certain threshold, i.e. the end of the sampling interval. Therefore all transitions emerging from *inTol*, *up* and *down* have action *t_o*, which stands for predicate $w = t_{sample}$, as common enabling condition. These *t_o* transitions then lead to *choice points*, which are well-known from Statecharts (Harel, 1987) and ROOMcharts. There, they split into several alternative branches. Some of these branches are newly introduced, because in the sampling model scenarios in which $fHeight$ traverses the hole tolerance interval within one sampling period are possible. Hence, transitions $u2d$ and $d2u$ leading directly from *up* to *down* and vice versa are necessary. As every state must be left after a sampling period in order to read the chassis level, we must also add transitions that allow to reenter the old state (transitions $i2i$, $u2u$ and $d2d$ in the HySCharts of Figure 4, right). The corresponding actions are defined as follows:

$$\begin{aligned} u2d &\equiv fHeight` \geq ub, \\ d2u &\equiv fHeight` \leq lb, \\ i2i &\equiv lb \leq fHeight` \leq ub, \\ u2u &\equiv fHeight` \leq lb + c, \\ d2d &\equiv fHeight` \geq ub - c \end{aligned}$$

After a transition was taken $w$ must be reset to zero in order to model that the next sampling interval starts. In the HySChart of Figure 4 this is specified by marking the states *inTol*, *up* and *down* with the entry action symbol $\rightarrow\circ$. The entry action is executed whenever the associated state is entered. In our example all three states have the same entry action which resets $w$, namely $entry \equiv w' = 0$

This example demonstrates two major benefits of HyCharts. First, is shows that hierarchy is not only useful for discrete states, but also for continuous activities. In the model of the controller with sampling *w_inc* is an example for a hierarchic activity. In a formalism like hybrid statecharts (Kesten and Pnueli, 1992) this would not be allowed and we would have to add activity *w_inc* to the states *inTol*, *up* and *down* explicitly. Clearly, this has negative consequences for further changes or refinements of these states.

Secondly, the development step performed above shows that HyCharts allow to make refinement steps from a very abstract model towards implementation within the same specification formalism. We think this feature of hybrid description techniques in general is highly useful for analog/digital codesign.

## 3. ABSTRACT VIEW OF HYBRID SYSTEMS

This section explains our understanding of hybrid systems and presents the underlying system model.

### 3.1 *Hybrid systems*

We model a *hybrid system* by a network of autonomous *components* that communicate in a *time synchronous* way via directed channels (Figure 5, left). Time synchrony is achieved by letting time flow uniformly for all components.

Every channel reflects a communication history of the system. Formally, such a history is a function $\mathbb{R}_+ \rightarrow M$, where $\mathbb{R}_+$ is the set of non-negative real numbers, our time scale, and $M$ is the set of messages transmitted on the channel (Müller and Scholz, 1997). In our view, the interface behavior of hybrid system components is characterized by intervals $[t, t')$ in which no discrete actions occur, i.e. the values on the channels only change "infinitely smoothly", and by time instances $t, t', \ldots$ at which discrete actions take place and (possibly) cause discontinuities in a component's output (Figure 2). A realistic component cannot perform actions at arbitrary frequency as it is not infinitely fast. Hence, we demand that a *hybrid communication history* or *hybrid stream* only has finitely many discontinuities during any finite interval $[t, t') \subset \mathbb{R}_+$, i.e. that it is *piecewise* infinitely smooth.

Components can now be formalized as total, time guarded relations over input and output communication histories. A relation is called *time guarded* if its output up to time $t$ only depends on its inputs up to $t$. We use relations instead of functions in order to allow nondeterminism. This nondeterminism can be used to express under-specification which quite naturally occurs in the early phases of system development that HyCharts are supposed to aid.

In principle a component can be specified by any description technique whose semantics results in such a relation. With a slight modification this e.g. allows to use block diagrams, which are widely used in Control Theory, to specify purely continuous components. For mixed discrete/continuous components, HySCharts are particularly appropriate.

### 3.2 *Hybrid machines*

A component specified by a HySChart is modeled by a *hybrid machine*, as shown in Figure 5, right [2].

[2] This diagram can itself be seen as a HyAChart.

This machine consists of three basic parts: a *combinational* (or discrete) part ($Com$), an *analog* (or continuous) part ($Ana$) and a *feedback* loop. The feedback models the *state* (or memory) of the

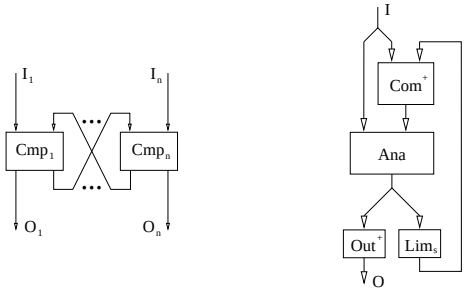


Fig. 5. The hybrid-machine computation model.

machine. Together with the limes from the left, $Lim_s$, it allows the component to remember at each moment of time $t$ the input received and the output produced "just before" $t$.

The combinational part is concerned with the control of the analog part and has *no memory*. It instantaneously and nondeterministically maps the current input and the fed back state to the next state.[3] The next state is used by the analog part to select an *activity* among a set of activities (or execution modes) and it is the starting state for this activity. The combinational part can only select a new next state (different from the fed back state) at distinct points in time. During the intervals between these time instances it passes the fed back state without modification, i.e. it *idles*, and the selection of the corresponding activity is stable for that interval.

The analog part describes the input/output behavior of the component whenever the combinational part idles. Hence, it adds to the component the temporal dimension. It may select a new activity whenever there is a discrete change in the input it receives from the environment or the combinational part.

### 3.3 *Example*

In the exemplary evolution of the EHC of Figure 2, the combinational part of the controller always idles except at times $t$ and $t'$, the filter's combinational part is only active at time $t''$. At time $t$ the controller realizes that the filtered chassis level, $fHeight$, is too high and opens the escape valve. In the model this means that, driven by the combinational part, the controller's analog part selects a new activity that decreases $aHeight$ and that starts with the value determined by the combinational part, i.e. the value right before the transition in this case. Due to the environment this decrease in turn causes $sHeight$ and finally $fHeight$ to decrease. At time $t'$ the controller's combinational part senses that $fHeight$ is in the desired tolerance interval again, selects a new next state with escape valve closed and sends the reset signal to the filter. At time $t''$ the filter receives this signal, its combinational part reacts by resetting $fHeight$ and its analog part selects an activity starting with this new value.

[3] Again nondeterminism models under-specification.

## 4. THE SEMANTIC FOUNDATION OF HYCHARTS

From an abstract point of view, both HyACharts and HySCharts consist of a set of nodes connected by a set of (typed) arcs, i.e. they are *hierarchic graphs*.[4] We interpret each node in such a graph as a relation between its inputs and outputs, as denoted by the arcs. Hierarchic graphs are constructed by putting nodes next to another and interconnecting them with the operators in Figure 6. Furthermore the connectors in Figure 7 may be applied to e.g. join, split or commute arcs.

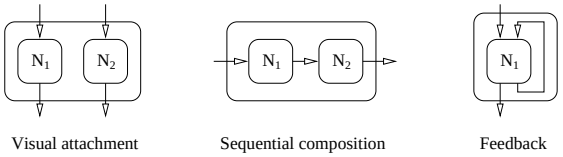


Fig. 6. The composition operators.

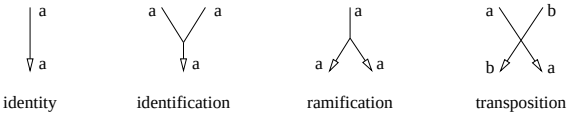


Fig. 7. The connectors.

In principle a HyAChart's or HySChart's semantics is obtained from the hierarchic graph by using one of two consistent interpretations of the abstract operators and connectors in Figures 6 and 7.

The *multiplicative interpretation* is used for HyACharts. Its effect is that all the nodes in a HyAChart are active in parallel. They receive data on all their input channels and send data along all their output channels. Thus, nodes closely correspond to system components in a data-flow network. Hence, the HyAChart models the data-flow in a system.

The *additive interpretation* is used for HySCharts. Its effect is that only one node can be active or *have control* at each time. A node receives

[4] For HySCharts some simple transformations are necessary to reduce entry/exit actions and preemption to such graphs.

control on one of its *entry points*, i.e. its incoming arcs, and passes control on on one of its *exit points*, i.e. its outgoing arcs. Thus, nodes closely correspond to the control states and arcs to the transitions in an automaton. The whole graph models the control-flow in the automaton.

A variant of this interpretation is used to obtain the analog part of a hybrid machine from the HySChart. Here, the activities of states at consecutive levels of hierarchy in the HySChart are sequentially composed. Each such sequence of sequentially composed activities corresponds to a solvable initial value problem. The interpretation of the visual attachment operator determines that only one sequence is active at a time. Furthermore, it switches between the sequences depending on the input the analog part receives form the combinational part and the environment.

A detailed explanation together with a definition of the mapping from HyChart syntax to their semantics is not possible within the limited space of this paper. We therefore refer the interested reader to (Grosu and Stauner, 1998).

## 5. CONCLUSION

By means of an example we introduced HyCharts and indicated how their main features like hierarchy of states and continuous activities, preemption and entry/exit actions can be utilized in the development of hybrid systems. In our development scenario we sketched elements of a methodology for the development of hybrid systems. We outlined the underlying system and machine model for HyCharts and revealed their connection to abstract hierarchic graphs, which serve as the semantic foundation for the two description techniques.

For use in practice, future work will have to focus on tool support and implementation techniques for HyCharts.

## ACKNOWLEDGEMENTS

We thank Robert Sandner for carefully reading a draft version of this paper.

## 6. REFERENCES

Alur, R., C. Courcoubetis, N. Halbwachs, T.A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis and S. Yovine (1995). The algorithmic analysis of hybrid systems. *Theoretical Computer Science* **138**, 3–34.

Alur, R., Henzinger, T.A. and Sontag, E.D., Eds.) (1996). *Hybrid Systems III: Verification and Control*. LNCS 1066. Springer-Verlag.

Grosu, R. and T. Stauner (1998). Modular and visual specification of hybrid systems – an introduction to HyCharts. Technical Report TUM-I9801. Technische Universität München.

Harel, D. (1987). Statecharts: A visual formalism for complex systems. *Science of Computer Programming*.

Kesten, Y. and A. Pnueli (1992). Timed and hybrid statecharts and their textual representation. In: *Formal Techniques in Real-Time and Fault-Tolerant Systems 2nd International Symposium*. LNCS 571. Springer-Verlag.

Lynch, N.A., R. Segala, F.W. Vaandrager and H.B. Weinberg (1996). Hybrid I/O automata. In: *Hybrid Systems III*. LNCS 1066. Springer-Verlag.

Müller, O. and P. Scholz (1997). Functional specification of real-time and hybrid systems. In: *Proc. Hybrid and Real-Time Systems (HART)*. LNCS 1201. Springer-Verlag.

Selic, Bran, Garth Gullekson and Paul T. Ward (1994). *Real-Time Object-Oriented Modeling*. John Wiley and Sons Ltd. Chichester.

Stauner, T., O. Müller and M. Fuchs (1997). Using HyTech to verify an automotive control system. In: *Proc. Hybrid and Real-Time Systems (HART)*. LNCS 1201. Springer-Verlag.