Simulation of Hybrid Systems

Björn Schieffer

Fachbereich 14 – Informatik, Universität des Saarlandes
Postfach 15 11 50, 66041 Saarbrücken, Germany
email: schieffer@informatik.uni-saarland.de
URL: http://www-hotz.cs.uni-sb.de/schieffer

Abstract

The paper deals with the simulation problem of hybrid systems. With the help of a small hybrid system called *swing* it is demonstrated how hybrid systems may be modeled in such a way the simulation of their behavior is feasible. The paper concludes with some notes to the hybrid systems simulator *hss*.

1 Introduction

Hybrid Systems contain both, discrete and continuous changes in their state. As examples we mention a *central heating*, a *gearing* and a *chemical factory*:

- Opening and closing some valves are discrete changes in a central heating. In contrast, the flow of heated water is a continuous change.
- In a gearing, the uncoupling is a discrete change of state. The rotation of cog-wheels is a continuous change.
- Some of the discrete changes in a chemical factory are the changes of physical state. Continuous changes may be found in chemical processes.

In many cases you can find a control unit that handles the discrete changes of a hybrid system. Via some actors and sensors it is connected to its physical environment containing the continuous changes.

In literature the *safetyness* of hybrid systems is discussed intensively. Some hybrid systems looked at are the *cat and mouse* example [MP93], the *gas burner* example [ACHH93], the *reactor core* example [HH95] and the *automotive control* example [SMF97]. In all these papers verification is used to guarantee that the hybrid systems really meet the specifications. One assumption always made is the absence of any faults. Therefore, to get safetyness also in the case of damages, there is the need to determine the reasons why verified specifications may have failed. This is called the *diagnosis problem* of hybrid systems.

The diagnosis problem may be subdivided into *fault recognition*, *fault localization* and *fault determination*:

- Fault recognition determines if there is any fault in the system or if the difference between measured and expected behavior is caused by noise.
- Fault localization determines which faults may be the reason of that difference.
- Fault determination is more precise than fault localization. In addition to fault localization it computes the degree of the faults.

As an example, fault localization could declare that a filter is plugged while fault determination finds out that it is plugged at a level of 42%. Fault determination may be realized by an approximation of the current state of the hybrid system [SH97]. The approximation requires many simulations of the hybrid system. Therefore, fault determination motivates simulation as a basic task.

In the next section, we introduce a small hybrid system called *swing*. Then, we show how to model it in a structure based way. This enables us to compute the behavior of the hybrid system. We will conclude with some remarks to the hybrid systems simulator *hss*.

2 Simulation

Consider figure 1 as a running example of a hybrid system. It consists of a motor that drives a gear with the help of a track. The gear has a little tooth that may switch the power supply of the motor. The discrete state of the system is the position of the commutator

$$\psi \in \{left, right\}$$

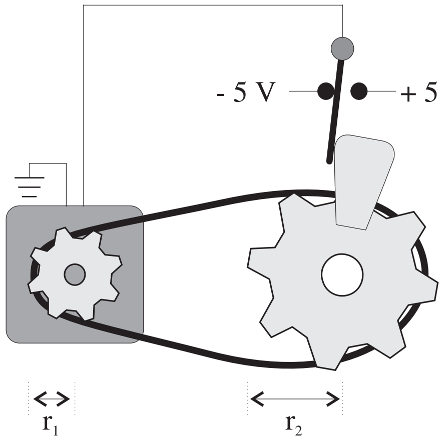

Figure 1: Hybrid system *swing*

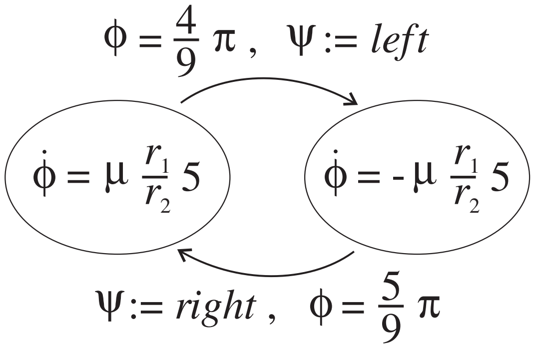

Figure 2: Hybrid automaton for the hybrid system *swing*

The continuous state is the angle of the tooth

$$\phi \in \ [0, 2\pi[$$

If $\phi = 0$ then the tooth is at 3 o'clock. If the tooth is at 12 o'clock then $\phi = \frac{\pi}{2}$. The motor turns the gear until the tooth switches the commutator. The tooth switches the commutator to the *left* position if – like in figure 1 – $\phi = \frac{4}{9}\pi$ and to the *right* position if $\phi = \frac{5}{9}\pi$. Then, the gear turns the other way round until the tooth again reaches the commutator and the direction of the motor is changed back. This is the reason why the hybrid system is called *swing*.

2.1 Modeling

Among others [Bra95, Mal97] one can find *phase transition diagrams* [Ben96], *hybrid statecharts* [MP93] or *hybrid automata* [ACHH93] to model a hybrid system. The hybrid system *swing* could be modeled by the hybrid automaton of figure 2.

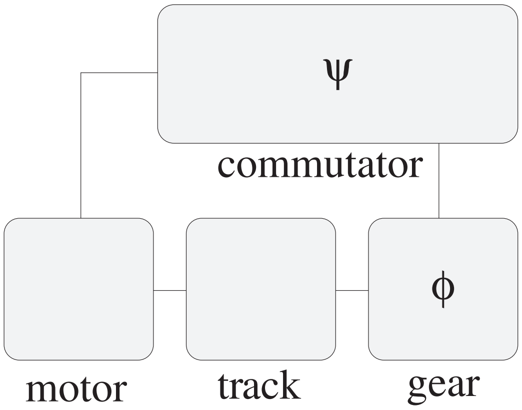

Figure 3: The way a structure based model looks like

All these models do not take the structure of the hybrid system into consideration. This may hamper the design of more complex hybrid systems. A structure based model should look like figure 3 where the components and links correspond to the structure of the system. The discrete state ψ and the continuous state ϕ are local to the component they belong to.

Let us try to complete the figure in such a way it results in a model that enables us to compute the behavior of the hybrid system. A link between two components corresponds to a variable both components share. Each component has local relations between the variables of its links. As an example, the component *motor* and the component *track* share the variable circulation speed v_1 while the components *track* and *gear* share the variable circulation speed v_2. As the circulation speed is the same at every point of a track the component *track* includes the relation

$$v_1 = v_2$$

According to figure 4 the variable angle of the tooth ϕ is shared by the components *gear* and *commutator* and the variable Voltage U is shared by the components *commutator* and *motor*. With the help of some constant factor μ the component *motor* adds a relation between the Voltage U at the input of the motor and its circulation speed v_1:

$$v_1 \quad = \quad \mu\, 2\, \pi\, r_1\, U$$

The component *gear* includes the relation

$$\dot{\phi} \quad = \quad \frac{v_2}{2\pi r_2}$$

Finally the component *commutator* tells us

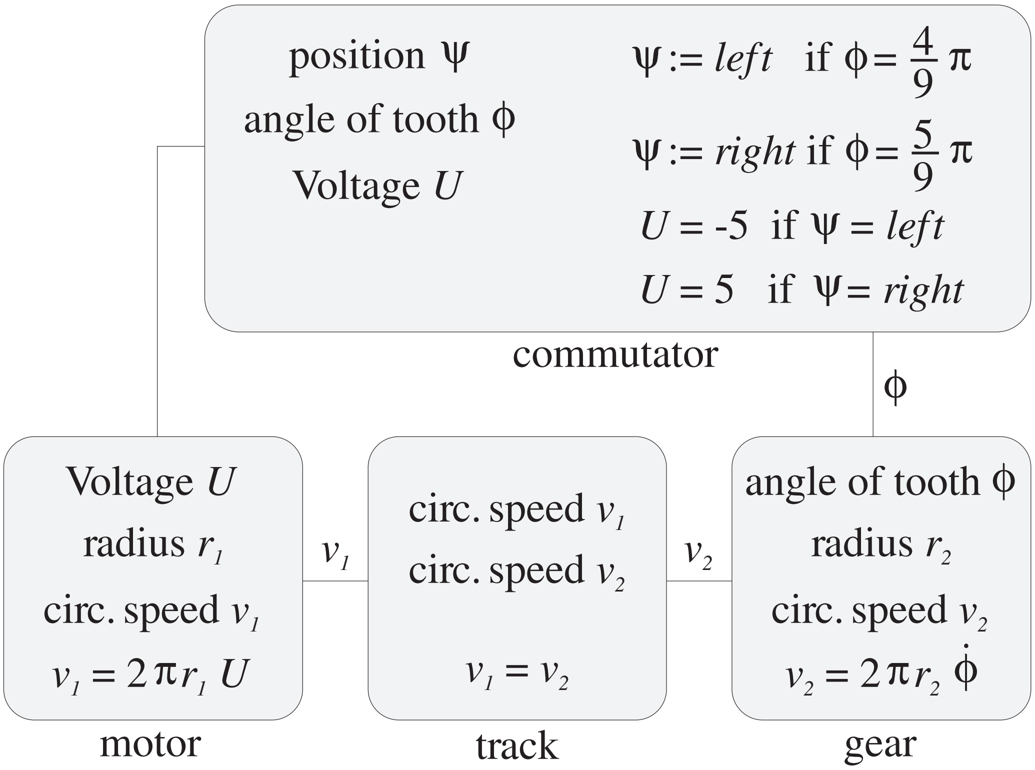

Figure 4: A structure based model

$$
\begin{array}{lcl}
\psi & := & left \qquad\qquad \text{if} \quad \phi = \frac{4}{9}\pi \\
\psi & := & right \qquad\qquad \text{if} \quad \phi = \frac{5}{9}\pi \\
U & = & -5 \qquad\qquad \text{if} \quad \psi = left \\
U & = & 5 \qquad\qquad\; \text{if} \quad \psi = right
\end{array}
$$

2.2 Computing the Behavior

Let V be the variables of the hybrid system *swing* and T a discrete time interval.

$$
\begin{array}{rcl}
V & := & \{\phi, \psi, v_1, v_2, \dot{\phi}, U\} \\
T & := & \{t_0, t_0+\delta_t, t_0+2\delta_t, t_0+3\delta_t, \ldots, t_1\} \\
\Omega & := & \mathbb{R} \cup \{left, right\} \\
sim & : & T \times V \rightarrow \Omega
\end{array}
$$

The function *sim* is called *simulation* if all of the discussed relations are met at any time $t \in T$.

The computation of the behavior of a model with shared variables and local relations is possible if there is an approximation algorithm that calculates the values of the variables so that all relations are fulfilled. One can do without approximation if

- all of the relations are functions. This is the case in logic simulation [Sch94]. Then, the components are boolean gates, the links are signals and the relations correspond to the functional behavior of the gates.
- all relations are linear equations or linear differential equations.

The hybrid system *swing* is an example of the second case. To demonstrate how to compute *sim* for it, let us distinguish between the *variables with inertia* V_i and the *variables without inertia* V_0. The term *inertia* shall emphasize the fact that the value of these variables at time $t+\delta_t$ depends on its value at time t. Therefore, V_i expresses the state of the hybrid system while V_0 are variables without memory.

$$
\begin{array}{rcl}
V_i & = & \{\phi, \psi\} \\
V_0 & = & \{v_1, v_2, \dot{\phi}, U\}
\end{array}
$$

The angle of the tooth ϕ is obviously in V_i. The position of the commutator ψ is in V_i because the value of the discrete variable ψ is only changed if $\phi \in \{\frac{4}{9}\pi, \frac{5}{9}\pi\}$. It depends on the current value of ψ in all other cases. In the real world the variables v_1, v_2 and $\dot{\phi}$ would belong to V_i too, but to keep the example simple we assume the motor, gear and track to have no mass.[1] $U \in V_0$ because the Voltage U at time $t + \delta_t$ does not depend on the value of U at time t.

We assume that $sim(t_0, v_i)$ is known for each $v_i \in V_i$. Then, we want to compute $sim(t, v)$ for all $t \in T$ and all $v \in V$. We use the principle of induction. We will show how to compute $sim(t + \delta_t, v)$ for any $t \in T$ and any $v \in V$ if $sim(t, v_i)$ is known for each $v_i \in V_i$.

$sim(t, v_0)$ may be estimated for each $v_0 \in V_0$ by a linear equation system. It depends on the discrete state ψ. If $\psi = left$ the linear equations are

v_1	v_2	$\dot{\phi}$	U	
1	0	0	$-\mu\, 2\, \pi\, r_1$	0
1	-1	0	0	0
0	1	$-2\, \pi\, r_2$	0	0
0	0	0	1	-5

If $\psi = right$ the linear equations only slightly differ. But in general, the equations might look totally different.

v_1	v_2	$\dot{\phi}$	U	
1	0	0	$-\mu\, 2\, \pi\, r_1$	0
1	-1	0	0	0
0	1	$-2\, \pi\, r_2$	0	0
0	0	0	1	5

Now, we know $sim(t, v)$ for each $v \in V$. Therefore, we are able to approximate the value of ϕ at time $t + \delta_t$ with numeric integration of the Euler method or the Runge-Kutta method. This allows us to determine the value of ψ at time $t + \delta_t$ too and the induction step is complete.

Note, that for numerical reasons, a condition like $\phi = \frac{4}{9}\pi$ will never evaluate to *true*. Therefore, the equality has to be changed into an approximative equality, but this is not topic of this paper. If you are interested see [HVS95, HC98].

3 The Hybrid Systems Simulator *hss*

We have implemented an hybrid systems simulator called *hss* which is part of the hybrid systems diagnosis tool *hsd*. If you like to simulate a hybrid system you first have to define the *domain*. Therefore, we have developed a language to describe the physical basis of the components and links that are used. For the hybrid system *swing*, the domain should describe what motors, tracks, gears and commutators are and the different links used between them, like *speed* or *angle*. For example, the description of a gear looks like that:

```
component     gear
{
 const Real   r      "radius";
 variable_0   d_phi  "derivative of phi",
 variable_0   v      "circulation speed",
 variable_i   phi    "angle of tooth"
  derivative  d_phi;
 link speed   l1     : (v);
 link angle   l2     : (phi);
 relation     r1     2*Pi*r*d_phi - v = 0;
}
```

In a second step, you may now use the graphical editor of *hss* to design a hybrid system of that domain. The components of your design are instances of the elements you created in the first step. It is of great use for the designer that *hss* works hierarchically. Therefore, the motor again might be a hybrid system consisting of an armature, some coils and so on.

After you have set the initial values of all variables of V_i you may now use the simulator of *hss* to compute the behavior of your design. Then, you may look at a visualization of the results of that simulation.

4 Conclusions

The computation of the variables in V_0 should not be restricted to the case of linear equation solving. The linear case enables us to model and to simulate many hybrid systems, but to be more general, approximation algorithms to solve non linear equation systems should be included.

The author wants to thank the German Research Foundation for granting this work within the Graduiertenkolleg *Efficiency and Complexity of Algorithms and Computer Systems*.

References

[ACHH93] R. Alur, C. Courcoubetis, Th. Henzinger, and Pei-Hsin Ho. Hybrid Automata: An Algorithmic Approach to the Specification and Verification of Hybrid Systems.

[1] On the other side, if the motor, the gear and the track would have no mass the motor would stop within the change of the commutator. But we don't worry about that.

In *LNCS*, volume 736, pages 209–229. Springer, 1993.

[Ben96] A. Benveniste. Compositional and Uniform Modelling of Hybrid Systems. In *LNCS*, volume 1066, pages 41–51. Springer, 1996.

[Bra95] M. Branicky. Universal computation and other cababilities of hybrid and continuous dynamical systems. *Theoretical Computer Science*, 138:67–100, 1995.

[HC98] G. Hotz and Th. Chadzelek. Analytic Machines. *to appear in: Theoretical Computer Science, special issue: Computability and Complexity in Analysis*, 1998.

[HH95] Th. Henzinger and Pei-Hsin Ho. The Cornell HYbrid TECHnology Tool. In *LNCS*, volume 999, pages 265–293. Springer, 1995.

[HVS95] G. Hotz, G. Vierke, and B. Schieffer. Analytic machines. Technical Report TR95-025, Electronic Colloquium on Computational Complexity (http://www.eccc.uni-trier.de/eccc), 1995.

[Mal97] O. Maler. *Hybrid and Real-Time Systems*, volume 1201 of *LNCS*. Springer, 1997.

[MP93] Z. Manna and A. Pnueli. Verifying Hybrid Systems. In *LNCS*, volume 736, pages 4–35. Springer, 1993.

[Sch94] B. Schieffer. Fast compiler-driven hierarchical logic simulation. In *Proceedings of the European Simulation Multiconference 94*, pages 991–998, 1994.

[SH97] B. Schieffer and G. Hotz. Diagnosis of Tank Ballast Systems. In *Advances in Intelligent Data Analysis, Reasoning about Data*, pages 597–608. Springer LNCS 1280, 1997.

[SMF97] Th. Stauner, O. Müller, and M. Fuchs. Using HyTech to Verifying an Automotive Control System. In *LNCS*, volume 1201, pages 139–153, 1997.