Hybrid Model Checking with the FUJABA Real-Time Tool Suite[*]

Stefan Henkler, Martin Hirsch, Claudia Priesterjahn
Software Engineering Group
University of Paderborn
Warburger Str. 100
D-33098 Paderborn, Germany
[shenkler|mahirsch|cpr]@uni-paderborn.de

ABSTRACT

Advanced mechatronic systems use their software to exploit local and global networking capabilities to enhance their functionality and to adapt their local behavior. Such systems include complex hard real-time coordination at the network level. This coordination is further reflected locally by complex reconfiguration in form of mode management and control algorithms. As such hybrid systems often contain safety-critical requirements, a proper approach for the safety analysis is mandatory. In former papers we have presented a compositional verification approach for the real-time and safety analysis. We present in this paper the integration of the hybrid verification tool PHAVer in the FUJABA Real-Time Tool Suite which enables also to consider hybrid requirements.

1. INTRODUCTION

For mechatronic systems [2], which have to be developed in a joint effort by teams of mechanical engineers, electrical engineers, and software engineers, the advances in networking and processing power provide many opportunities. The development of such systems will therefore at first require means to develop software for the complex hard real-time coordination of its subsystems at the network level. Secondly, software for the complex reconfiguration of the local behavior in form of mode management and control algorithms is required, which has to proper coordinate the local reconfiguration with the coordination at the network level.

The interplay between the discrete software models and the continuous controllers is the cause for hybrid requirements. As such systems often contain safety-critical requirements, a proper approach for the safety and hybrid analysis is mandatory.

In [3] and [5] we have presented the FUJABA Real-Time Tool Suite which enables the model-based development of mechatronic Systems as well as the formal verification of the real-time coordination and the correct embedding of the employed controllers. But, the current Tool Suite lacks in the support for analyzing hybrid behavior.

We present in this paper the integration of the hybrid verification tool PHAVer [6] into the FUJABA Real-Time Tool Suite based on the mappings from Real-Time Statecharts to hierachical Timed Automata [7, 8]. In detail, we have to map Hybrid Reconfiguration Charts (HRC), our modeling approach for hybrid systems, to Hybrid Input/Output Automata (HIOA) [6], the input model of PHAVer.

In the remainder of this paper, we at first present in Section 2 the conceptual integration of the PHAVer tool. Then, we outline in Section 3 the tool support and evaluation. Finally, we conclude and present future work.

2. HYBRID MODEL CHECKING OF UML MODELS

In this section the concept of the integration of the hybrid model checker PHAVer and the necessary mapping rules are presented. Figure 1 shows the actions that have to be performed for the model checking of HRCs with PHAVer. The main step is the transformation from the HRC model into a HIOA. The HIOA and a specification to be checked build the input for PHAVer. The model checker states if the model fulfills the specification or not. In order that the verification

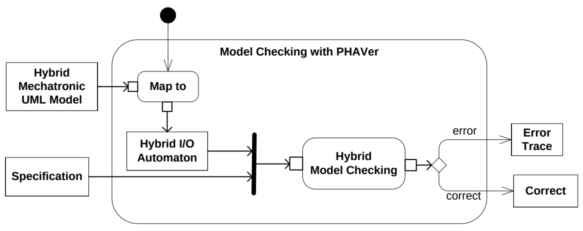


Figure 1: Hybrid Model Checking with PHAVer

will perform correctly the mapping must keep the semantics of the original model. Otherwise we might find false positives or false negatives during verification. The following paragraphs introduce the mapping rules established in this work.

Start State.

A start state in a HRC can embed configurations of the component. The continuous values at the component's ports can thus be deployed in this state. However, in contrast to HIOA reconfiguration charts can not specify their inital values or range of values. The start state of the HRC is mapped to the start state of the HIOA consisting of a location and variable allocations. Each continuous variable is assigned the value 0 (Figure 2).

[*]This work was developed in the course of the Collaborative Research Center 614 – Self-optimizing Concepts and Structures in Mechanical Engineering – University of Paderborn, and was published on its behalf and funded by the Deutsche Forschungsgemeinschaft.

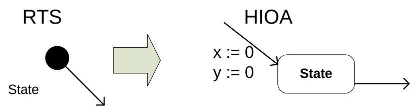


Figure 2: Mapping a start state

Clocks and Clock Resets.

Each clock t_k of the original timed automaton is mapped as a continuous variable t_k such that for each location of the HIOA $\dot{t}_k = 1$ is satisfied.

Further in HIOA discrete assignments can be assigned to transitions. Therefore a clock reset can simply be modeled by assigning 0 to the clock to be reset (Figure 3).

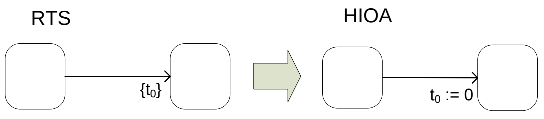


Figure 3: Mapping clock resets

Time Invariants and Time Guards.

Because of the direct mapping of clocks to continuous variables both time invariants and time guards can directly be taken over to the HIOA (Figure 4).

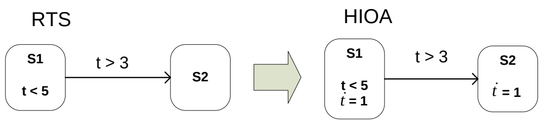


Figure 4: Mapping guards

Urgent and Non-Urgent Transitions.

As HIOA allow only urgent transitions, all non-urgent transitions of the hybrid reconfiguration chart are mapped to urgent-transitions. Thus the original semantics are kept since "non-urgent" says that a transitions does not need to fire immediately after its activation. In case such a transition would actually fire immediately no unexpected behavior would result [12].

Synchronous Communication.

The mapping for synchronous communication is depicted in Figures 5 and 6. Figure 5 shows the synchronization modeled by a HRC. Parallel states of hybrid reconfiguration charts synchronize via synchronization channels. Each synchronization contains exactly one sender and one receiver. At the starting point the HRCs as depicted in Figure 5 start with states $S1$ and $S3$. When the event e is activated the statecharts switch to $S2$ and 4 synchronously.

Figure 6 shows the same synchronization modeled by two HIOA. Generally transitions of two parallel HIOAs will fire simultaneously, if they are labeled with the same synchronization marks and are activated at the current moment. $S1$ is the source of the transitions sending messages. That is why all incident transitions have to contain the variable *e_send* representing the sender. For all incoming transitions *e_send* is set to 1. All outgoing transitions must reset this value to 0 - the initial value. The receiving transition

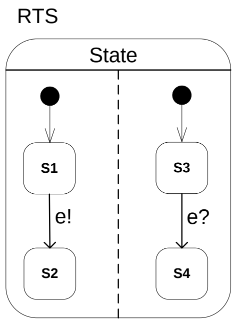


Figure 5: Synchronous communication modeled by hybrid reconfiguration chart

$(S3, S4)$ will only be fired, if *e_send* is set to 1 and another transition also marked with the label e is activated. Thus sender and receiver can only fire synchronously and the original semantics of synchronous communication in HRCs are kept.

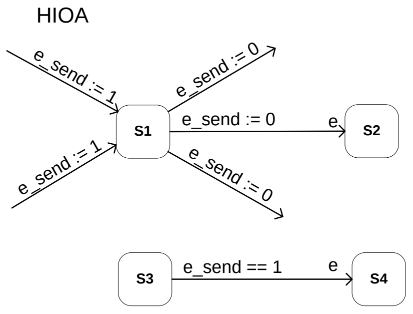


Figure 6: Synchronous communication modeled by HIOA

Embedding of Component Instances.

In HRCs the behavior of controllers is described by differential equations over the incoming and outgoing signals. The same applies to the component instances embedded in states of hybrid reconfiguration charts. As the continuous dynamics of hybrid systems in PHAVer is also described by differential equations they can be directly applied to HIOA.

Application Order.

The existing XML exchange format for the transformation of Mechatronic UML models to the UPPAAL input format [8] has been extended by continuous parts to enable the verification of continuous parts of mechatronic systems. Also the transformation of Real-Time Statecharts into hierarchical Timed Automata has been extended by continuous parts. In that way the following subset of the needed mappings can already be performed: stop states, entry()-, do() and exit()-methods and the history operator.

Afterwards the mapping rules presented in this paper are applied. To guarantee the correctness of the resulting models the mapping steps have to be applied in a defined order. For instance the mapping rule for clocks demands that all locations of the resulting model satisfy a certain property. Therefore it must not happen that one of following mapping steps adds a location that does not satisfy this property.

The following list shows the appropriate application order of the transformation steps.

1. First apply the rules presented in [8].
2. Map the synchronous communication.
3. Map the non-urgent transitions, since following mappings can not add new locations or transitions to the model.
4. Map the clocks.
5. Map time guards, time invariants and clock resets.

3. TOOL SUPPORT AND EVALUATION

In this section we explain the implementation of the approach. First, we describe the implementation as a plugin for FUJABA. Thereafter we give an evaluation example and point out the adaptability of the verification to real world examples.

3.1 The Plugin

In Figure 7 the architecture of the required FUJABA plugins is depicted. The three plugins HybridComponenent, UMLRT2, and RealtimeStatechart allows us to model the behavior as well as the structure of hybrid systems. The plugins on the bottom left side realize the integration of model checker. The UMLRTModelchecking provides an interface for model checker which allows to export the model and add constraints to the model [4]. The actual model checking engine is provide by the specific model checker, in our case the PHAVer-Plugin.

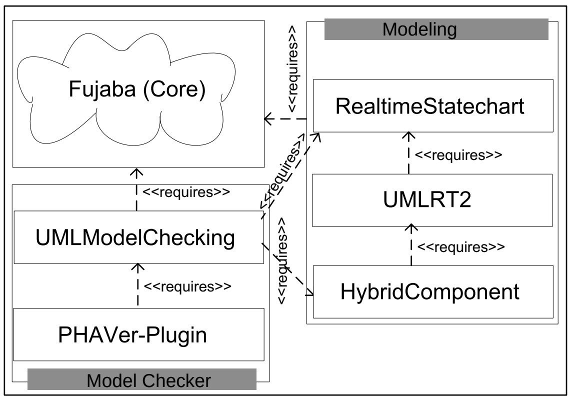
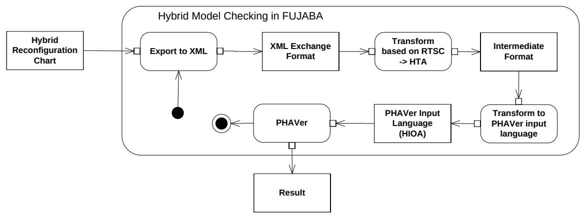


Figure 7: Architecture of the plugin

Figure 8 shows the transformation steps taken by the implemented FUJABA plugin. First the hybrid reconfiguration chart is exported to the XML exchange format provided by the UMLModelChecking Plugin. This action is followed by the transformation into an intermediate format and the transformation into the final PHAVer input format the hybrid input output automata.

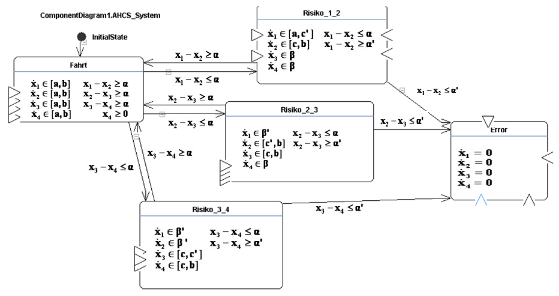
3.2 Evaluation

As an evaluation example we take the AHCS case study from [10]. We consider the model of a central controller for a automated highway and analyze the controller itself for safety properties, particularly for the specification that no two vehicles on the automated highway collide with each other. The controller enforces speed limits on vehicles on the automated highway to achieve this purpose. In Figure 9 the hybrid reconfiguration chart to which realizes the behavior of the AHCS for four vehicles is depicted. x_k is the distance of one vehicle k to the beginning of the highway and $\dot{x}_k$ the velocity of the vehicle. At the beginning (state "'Fahrt"') the velocity for each vehicle is in the interval $\dot{x} \in [a, b]$. When two vehicles i, j come within a distance α $(x_i - x_j < \alpha)$ of each other, we call this a "'possible"' collision event and the controller switches to the "'Risiko_i_j"' state. The controller asks the approaching vehicle to slow down by reducing the upper bound to $\dot{x}_i \in [a, c']$ and asks the leading vehicles to speed up by increasing the lower bound to $\dot{x}_j \in [c, b], c > c'$; it also requires that all other cars not involved in the possible collision slow down to a constant velocity β for vehicles behind the critical region and $\beta', \beta' > \beta$ for vehicles in front of the critical region. When the distance between the two vehicles involved in the possible collision exceeds α, the controller switches to the "'Fahrt"' state . Otherwise the "'Error"' state will be reached, if $x_j - x_i < \alpha', alpha' < \alpha$



Hybrid Model Checking in FUJABA
Hybrid Reconfiguration Chart
Export to XML
XML Exchange Format
Transform based on RTSC -> HTA
Intermediate Format
Transform to PHAVer input language
PHAVer Input Language (HIOA)
PHAVer
Result

Figure 8: Hybrid Model Checking in FUJABA



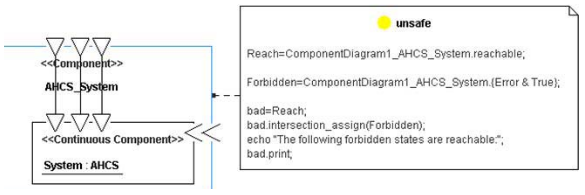
ComponentDiagram1.AHCS_System
InitialState
Fahrt
$\dot{x}_1 \in [a,b]$ $x_1 - x_2 \geq \alpha$
$\dot{x}_2 \in [a,b]$ $x_2 - x_3 \geq \alpha$
$\dot{x}_3 \in [a,b]$ $x_3 - x_4 \geq \alpha$
$\dot{x}_4 \in [a,b]$ $x_4 \geq 0$
Risiko_1_2
$\dot{x}_1 \in [a,c']$ $x_1 - x_2 \leq \alpha$
$\dot{x}_2 \in [c,b]$ $x_1 - x_2 \geq \alpha'$
$\dot{x}_3 \in \beta$
$\dot{x}_4 \in \beta$
Risiko_2_3
$\dot{x}_1 \in \beta'$ $x_2 - x_3 \leq \alpha$
$\dot{x}_2 \in [c',b]$ $x_2 - x_3 \geq \alpha'$
$\dot{x}_3 \in [c,b]$
$\dot{x}_4 \in \beta$
Risiko_3_4
$\dot{x}_1 \in \beta'$ $x_3 - x_4 \leq \alpha$
$\dot{x}_2 \in \beta'$ $x_3 - x_4 \geq \alpha'$
$\dot{x}_3 \in [c,c']$
$\dot{x}_4 \in [c,b]$
Error
$\dot{x}_1 = 0$
$\dot{x}_2 = 0$
$\dot{x}_3 = 0$
$\dot{x}_4 = 0$
$x_1 - x_2 \geq \alpha$
$x_1 - x_2 \leq \alpha$
$x_2 - x_3 \geq \alpha$
$x_2 - x_3 \leq \alpha$
$x_3 - x_4 \geq \alpha$
$x_3 - x_4 \leq \alpha$
$x_1 - x_2 \leq \alpha'$
$x_2 - x_3 \leq \alpha'$
$x_3 - x_4 \leq \alpha'$

Figure 9: Evaluation example

To ensure that no collision between the vehicles happen we have to check that the "'Error"' state will be never reached as long as the systems runs. To check this constraint we have to compute the followings steps in PHAVer:

- compute all reachable state: Reach = System.reachable
- define the forbidden states: Forbidden = System.{Error? & True}
- compute the set $Reach \cap Forbidden$: bad.intersection_assign(Forbidden)

In Figure 10 a screenshot of the constraint is depicted. For the evaluation of the example, we chose the following parameterization of the variables. WE set the number of vehicles



```
Reach=ComponentDiagram1_AHCS_System.reachable;

Forbidden=ComponentDiagram1_AHCS_System.{Error & True};

bad=Reach;
bad.intersection_assign(Forbidden);
echo "The following forbidden states are reachable:";
bad.print;
```

Figure 10: Specification of the constraint

to 4, 6, 8, 12, 14, and 16. The other variables are set to:

$$a = 0, b = 100, c = 70, c' = 50, \alpha = 100, \alpha' = 10, \beta = 40, \beta' = 80$$

.

The results of the evaluation are presented in Table 3.2 as well as a visualization of the results is given in 11. In detail the time and memory consumption are analyzed. One result is that the runtime of the model checker exponentially increase with the number of vehicles. The same holds for the memory consumption. It is to be noted that although PHAVer is more efficient than other model checkers [6] the runtime exponentially increase with the number of variables used in the HIOAs [11]. Hence it can take a long time to get a result if any result is computed.

Number of vehicles	Runtime [s]	Memory [MB]
4	$0,16$	3 MB
8	$2,6$	11 MB
10	$11,34$	16 MB
12	$156,7$	37 MB
14	$5120,3$	112 MB
16	$14092,34$	354 MB

Table 1: Evaluation results

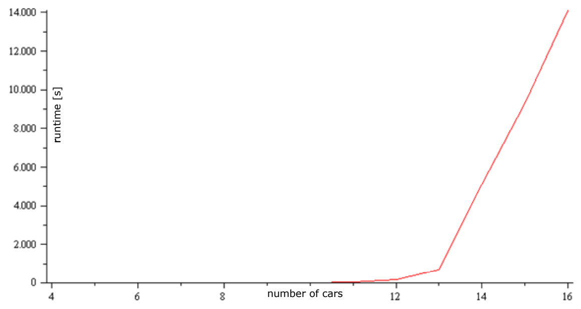


Figure 11: Evaluation results

4. CONCLUSION AND FUTURE WORK

In this paper, we have presented the integration of the hybrid verification tool PHAVer in the FUJABA Real-Time Tool Suite. We have shown the mapping of our hybrid modeling approach, Hybrid Reconfiguration Charts, to the input model of PHAVer (Hybrid Input/Output Automata). In the evaluation, we have shown that, in principle, the model checking of hybrid systems does not scale. Therefore, appropriate abstractions are required as shown in [9]. In the future, we want to consider a better integration of the PHAVer tool in the compositional verification approach by using the assume/guarantee approach of PHAVer [1] which enables a better scalability.

5. REFERENCES

[1] S. Berezin, S. Campos, and E. M. Clarke. Compositional reasoning in model checking. *Lecture Notes in Computer Science*, 1536:81–102, 1998.

[2] D. Bradley, D. Seward, D. Dawson, and S. Burge. *Mechatronics*. Stanley Thornes, 2000.

[3] S. Burmester, H. Giese, S. Henkler, M. Hirsch, M. Tichy, A. Gambuzza, E. Müch, and H. Vöcking. Tool support for developing advanced mechatronic systems: Integrating the fujaba real-time tool suite with camel-view. In *Proc. of the 29th International Conference on Software Engineering (ICSE), Minneapolis, Minnesota, USA*, pages 801–804. IEEE Computer Society Press, May 2007.

[4] S. Burmester, H. Giese, M. Hirsch, and D. Schilling. Incremental design and formal verification with UML/RT in the FUJABA real-time tool suite. In *Proceedings of the International Workshop on Specification and vaildation of UML models for Real Time and embedded Systems, SVERTS2004, Satellite Event of the 7th International Conference on the Unified Modeling Language, UML2004*, October 2004. to appear.

[5] S. Burmester, H. Giese, M. Hirsch, D. Schilling, and M. Tichy. The fujaba real-time tool suite: Model-driven development of safety-critical, real-time systems. In *Proc. of the 27th International Conference on Software Engineering (ICSE), St. Louis, Missouri, USA*, pages 670–671. ACM Press, May 2005.

[6] G. Frehse. Phaver: Algorithmic verification of hybrid systems past hytech. pages 258–273. Springer, 2005.

[7] H. Giese and S. Burmester. Real-Time Statechart Semantics. Technical Report tr-ri-03-239, University of Paderborn, Paderborn, Germany, June 2003.

[8] M. Hirsch. Effizientes Model Checking von UML-RT Modellen und Realtime Statecharts mit UPPAAL. Master's thesis, University of Paderborn, June 2004.

[9] M. Hirsch, S. Henkler, and H. Giese. Modeling Collaborations with Dynamic Structural Adaptation in Mechatronic UML. In *Proc. of the ICSE 2008 Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS'08),Leipzig, Germany*, pages 1–8. ACM Press, May 2008. to appear.

[10] S. K. Jha, B. H. Krogh, J. E. Weimer, and E. M. Clarke. Reachability for linear hybrid automata using iterative relaxation abstraction. In A. Bemporad, A. Bicchi, and G. C. Buttazzo, editors, *HSCC*, volume 4416 of *Lecture Notes in Computer Science*, pages 287–300. Springer, 2007.

[11] X. Li, S. J. Aanand, and L. Bu. Towards an efficient path-oriented tool for bounded reachability analysis of linear hybrid systems using linear programming. *Electron. Notes Theor. Comput. Sci.*, 174(3):57–70, 2007.

[12] A. Steinke. Integration Hybrider Rekonfigurations-charts mit Matlab/Simulink-Modellen. Master's thesis, University of Paderborn, 2007.