# Formal Verification of DEV&DESS Formalism Using Symbolic Model Checker HyTech

Han Choi[1], Sungdeok Cha[1], Jae Yeon Jo[2],
Junbeom Yoo[2], Hae Young Lee[3], and Won-Tae Kim[3]

[1] Korea University, Seoul, Republic of Korea
{issubi,scha}@korea.ac.kr
[2] Konkuk University, Seoul, Republic of Korea
{tm77,jbyoo}@konkuk.ac.kr
[3] Electronics and Telecommunications Research Institute,
Daejeon, Republic of Korea
{haelee,wtkim}@etri.re.kr

**Abstract.** A hybrid system is a dynamical system reacting to continuous and discrete changes simultaneously. Many researchers have proposed modeling and verification formalisms for hybrid systems, but algorithmic verification of important properties such as safety and reachability is still an on-going research area. This paper demonstrates that a basic modeling formalism for hybrid systems, DEV&DESS is an easy-to-use input front-end of a formal verification tool, HyTech. HyTech is a symbolic model checker for liner hybrid automata, and we transformed an atomic DEV&DESS model into linear hybrid automata. We are now developing translation rules from DEV&DESS models, including a coupled DEV&DESS, into linear hybrid automata, through various case studies.

## 1 Introduction

A hybrid system is a dynamical system whose behavior is a combination of continuous and discrete dynamics [5]. The discrete parts naturally model modes of operation of the system, while the continuous dynamics model physical interactions with themselves or environment, such as automotive controllers, avionic systems, defense modeling and simulation, medical equipments and controllers.

Researches on modeling, analysis and verification of hybrid systems can be classified into two classes, one starting from finite state machine (FSM) and the other originating from continuous system controls. Timed automata [3], (linear) hybrid automata [1,4], DEV&DESS [22] and CHARON [2] are modeling methods for hybrid systems, which belong to the first class. They have also been applied to various domains [11,7,20,13,14] relevantly. UPPAAL [21], HyTech [12] and KRONOS [10] are algorithmic verification tools developed for model checking. UPPAAL uses timed automata which is a finite automaton augmented with a finite number of clocks, which are real-valued variables changing continuously with a constant rate 1(one). HyTech can handle more general kinds of hybrid systems since it uses linear hybrid automata. The second class has developed

T.-h. Kim et al. (Eds.): CA/CES[3] 2011, CCIS 256, pp. 112–121, 2011.

out of researches in continuous dynamics and system control. Many researches[1] have been proposed to specify and verify hybrid systems from the viewpoint of control automation. Efficient application of these techniques, however, is rather limited to specific scopes, because of their inherent complexity. In particular, Checkmate [8], *d/dt* [6] and level set method [16] allow verification on more general dynamics than HyTech, but still have the hard problem of scalability.

DEV&DESS have been typically used as a fundamental formalism of other extended ones (e.g. CHARON and ECML [15]), since it deals with only basic behavior of hybrid systems without visual syntax. Therefore, verification of the hybrid system modeled in the DEV&DESS formalism using HyTech is a good starting-point of developing modeling and verification tools for hybrid systems, which a research institute in Korea, ETRI (Electronics and Telecommunications Research Institute) does. We first transformed the model of DEV&DESS formalism into a form of linear hybrid automata. While transforming the DEV&DESS formalism into linear hybrid automata, we found and resolved many semantic gaps between the two formalisms. We, however, still have several obstacles to overcome, and we are currently focusing on resolving them.

The outline of the paper is as follows. We briefly review the basics of the DEV&DESS formalism and the HyTech symbolic model checker in Section 2. It includes a brief introduction to linear hybrid automata. Section 3 explains hybrid systems modeled with DEV&DESS, and transformation into linear hybrid automata and formal verification using HyTech are explained in Section 4. In Section 5, we summarize out considerations on the translation from DEV&DESS into linear hybrid automata and the problems to be solved. Section 6 concludes the paper.

## 2 Background

### 2.1 DEV&DESS Formalism

The *discrete event and differential equation specified system (DEV&DESS)* [18,17] formalism is a system theoretic formalism [22] for modular, hierarchical, combined discrete/continuous system modeling. The DEV&DESS formalism is defined as follows[22,18]:

$$DEV\&DESS = \langle\ X^{discr},\ X^{cont},\ Y^{discr},\ Y^{cont},\ S,\ \delta_{int},\ C_{int},\ \delta_{int},\ \lambda^{discr},\ ta,\ f,\ \lambda^{cont}\ \rangle$$

where,

- $X^{discr}$ and $Y^{discr}$ are the set of discrete input and output from DEVS
- $X^{cont}$ and $Y^{cont}$ are the structured set of continuous value inputs and outputs
- $S = S^{discr} \times S^{cont}$ is the cartesian product of continuous and discrete states
- $\delta_{ext} : Q \times X^{cont} \times X^{disc} \rightarrow S$ is the external state transition function, where $Q = \{(s^{discr}, s^{cont}, e) | s^{discr} \in S^{discr}, s^{cont} \in S^{cont}, e \in R_0^+\}$ is the set of total states with the elapsed time $e$

[1] See the website `http://wiki.grasp.upenn.edu/hst/index.php?n=Main.HomePage` managed by U. Penn.

- $\delta_{int} : Q \times X^{cont} \rightarrow S$ is the internal state transition function
- $C_{int} : Q \times X^{cont} \rightarrow Bool$ is the state event condition function
- $\lambda_{discr} : Q \times X^{cont} \rightarrow Y^{discr}$ is the discrete event output function
- $\lambda_{cont} : Q \times X^{cont} \rightarrow Y^{cont}$ is the continuous output function
- $f : Q \times X^{cont} \rightarrow S^{cout}$ is the rate of change function
- $ta : S^{discr} \rightarrow R_0^+ \cup \infty$ is the time advance function

The semantics of the DEV&DESS are informally described as follows:

1. **Intervals $\langle t_1, t_2 \rangle$ with no events:** Only $S^{cont}$ and $e$ change. $\lambda^{cont}$ is operating throughout the interval.
2. **A state event occurs first at time $t$ in interval $\langle t_1, t_2 \rangle$:** $\delta_{int}$ is executed to define a new state. $e$ is set to 0, and $\lambda^{discr}$ is called to generate an output event at time $t$. $\lambda^{cont}$ is calculated from time $t_1$ to $t$.
3. **A discrete event at the external input port occurs first at time $t$ in interval $\langle t_1, t_2 \rangle$:** $\delta_{ext}$ is executed to define a new state at time $t$. $e$ is set to 0, and $\lambda^{cont}$ is calculated from time $t_1$ to $t$.

Fundamental of DEV&DESS is to understand how the discrete part is affected by the continuous part (See [22] for detials). Events occur whenever a condition specified on the continuous elements becomes true. In the interval between events, the continuous input, state and output values change continuously.

### 2.2 HyTech and Linear Hybrid Automata

HyTech [12] is a symbolic model checker for linear hybrid automata [4]. As other model checkers [9], it checks a formal model of a system automatically for correctness with respect to requirements. It also provides parametric analysis. If a system model has parameters whose values are not decided yet, then HyTech can compute necessary and sufficient conditions on the parameter's value.

Hybrid automata[1] has been a background formalism of various modeling and verification tools for hybrid systems. However, since the hybrid automata model is too expressive to be proved automatically by model checkers with reasonable computational cost, HyTech uses a restricted class of hybrid automata, which called linear hybrid automata. A hybrid automaton is a linear hybrid automaton if it satisfies following two requirements:

1. **Linearity:** For every control mode $v \in V$, $flow(v)$, $inv(v)$ and $init(v)$ are finite conjunctions of linear inequalities.
2. **Flow independence:** For every control mode $v \in V$, $flow(v)$ is a predicate over the variables in $\dot{X}$ only and does not contain any variables from $X$.

With linear hybrid automata, the HyTech symbolic model checker verifies safety properties and analyzes correct conditions for parameters. Many embedded systems which belong to hybrid systems, on the other hand, may be not the linear hybrid automata. They are not apt to meet the constraints above. In such cases, we carefully have to approximate or abstract the system model into linear hybrid automata. However, from a practical point of view, we still face a number of important hybrid systems which are classified into linear hybrid automata.

## 3 Hybrid System Modeling

To demonstrate our approach, we modeled an atomic DEV&DESS model for a simple barrel filler system originated in [22]. Since there is no visual representation in the DEV&DESS formalism, we used graphical notations proposed by [19] for better understanding. In graphical notations, states mean the phases which are mutually exclusive state space[22], while solid and dotted transitions symbolize external and internal events, respectively. ⟨Fig.1⟩ denotes the DEV&DESS model for a barrel filler system. The formal definition of DEV&DESS model is skipped for the limit of space.

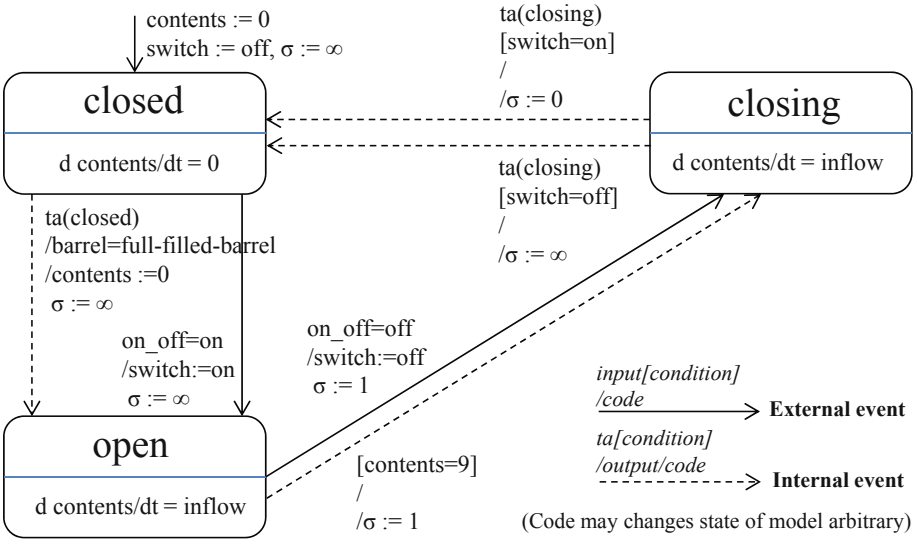


**Fig. 1.** An atomic DEV&DESS model for the Barrel Filler

The system fills a barrel with a certain rate of inflow while the valve of pipe is opened. When the barrel is filled up, it stops filling, then puts the full-filled barrel out. This process involves changing the barrel with a new empty one. The valve of the system can be changed by signal from the input port '*on_off*', the external event, or by the state event occurred when '*contents*' becomes a cutoff value while the valve is opened; the cutoff value is set as 9 for prevention of overflow because of the assumption that the capacity of a barrel is 10 liter. The continuous variable '*contents*' represents the level of water, increasing its value continuously with derivative. Its derivative is the same as the continuous input '*inflow*' while the valve is not closed. We also assumed the inflow of water is decreased as half, from 0.5 to 0.25, when the valve is closing since the valve is not fully opened. The variable '*switch*', which contains the last signal value of input '*on_off*', is used for differentiation the two events that change the state of the valve, since '*contents*' is the only available information. The '$\sigma$' which is renewed at every phase changes, represents result of time advanced function.

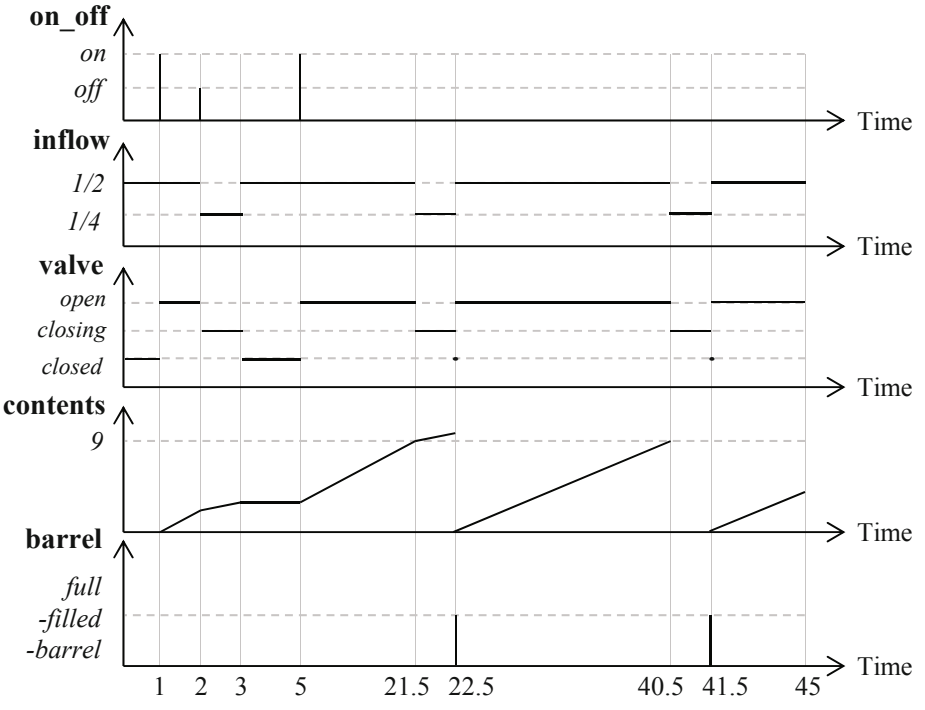


**Fig. 2.** The trajectories for the Barrel Filler

In order to guarantee the correct behavior of the barrel filler model, we made a simple scenario for the barrel filler system as ⟨Fig.2⟩. In the whole scenario, the model starts filling a barrel twice by the 'on' signal, and stops once by the 'off' signal. Even though the value of '*contents*' stops increasing occasionally by the '*on_off*' signal, as time goes on, the state event is occurred whenever the '*contents*' reaches at 9. The state event causes the model puts a 'full-filled-barrel' out when the valve is closed, and the series of processes above are repeated immediately after a barrel is produced.

For translation the barrel filler model into linear hybrid automata, it is needed to confirm whether the model satisfies the conditions of linear hybrid automata or not, since it is one of the necessary condition for translation. First, the rate of change function '$f$' is influenced only by the input '*inflow*'. Although '*inflow*' has different value according to the state of the valve, it has constant value while system stays in the same state. Therefore continuous state variable '*contents*' always increases linearly under same phase, meaning that the model has '*flow independence condition*' of linear hybrid automata. Second, it is able to find out all possible phases and its changes, since all events and states of the model are decidable. Thus every condition of concerned variables in phases can be expressed by conjunction of linear inequalities, and we can assure that the model above satisfies the '*linearity*' of linear hybrid automata.

## 4 Verification Using HyTech

This section demonstrates that properties of barrel filler model can be verified with HyTech through translation into linear hybrid automata model. We introduce the translated linear hybrid automata first.

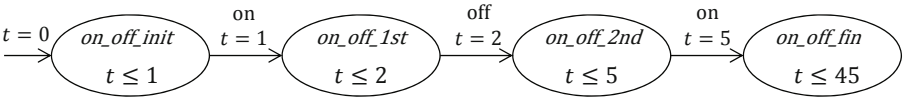


**Fig. 3.** Automata for discrete input '*on_off*'

The main issue faced during translation was that hybrid automata[1] doesn't have any notations to express I/O behavior, different from the DEV&DESS formalism. Thus we used an additional automaton, as described in ⟨Fig.3⟩, which shows the same behavior with the input '*on_off*'. The variable '$t$' is a clock for global time, and the automaton coordinates with the barrel filler automata through synchronization labels 'on', 'off'.

The barrel filler model with explained scenario can be translated into a linear hybrid automaton as depicted in ⟨Fig.4⟩. The variable '$e$' represents the elapsed time in DEV&DESS and it is compared at every control mode with the variable '$\sigma$' for checking occurrence of time event. ⟨Fig.5⟩ shows an excerption of the state trace from 0 to 22.5 time unit by the HyTech execution. The value '$-1000$'

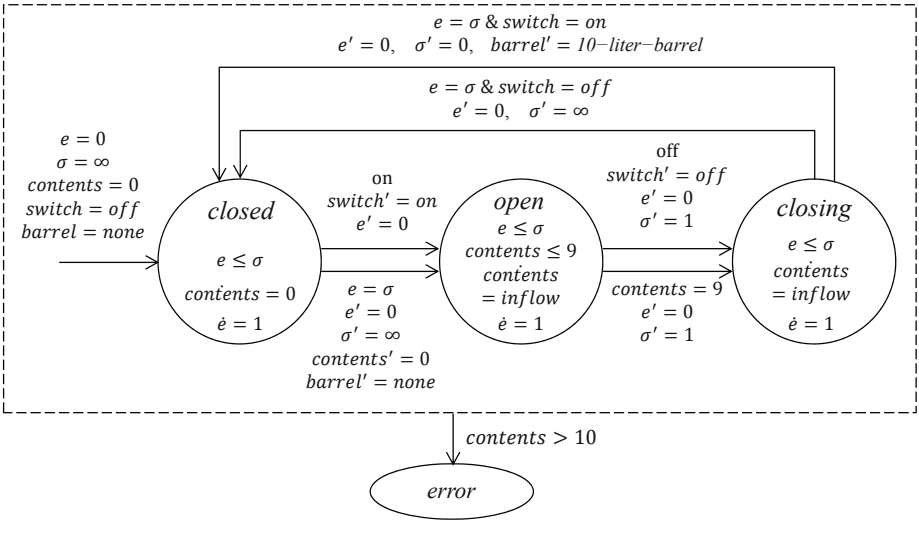


**Fig. 4.** A linear hybrid automata model for the barrel filler model

```
Time: 0.000
Location: closed.on_off_init
sigma=1000 & switch=0 & barrel=0
& contents=0 & e=0 & t=0
---------------
 VIA 1.000 time units
---------------
Time: 1.000
Location: closed.on_off_init
 sigma=1000 & switch=0 & barrel=0
& contents=0 & e=1 & t=1
--------------------------------
  VIA: on
--------------------------------
Time: 1.000
Location: open.on_off_1st
 sigma=1000 & switch=1 & barrel=0
& contents=0 & e=0 & t=1
---------------
 VIA 1.000 time units
---------------
Time: 2.000
Location: open.on_off_1st
 sigma=1000 & switch=1 & barrel=0
& 2contents=1 & e=1 & t=2
--------------------------------
  VIA: off
--------------------------------
Time: 2.000
Location: closing.on_off_2nd
 sigma=1 & switch=0 & barrel=0 &
2contents=1 & e=0 & t=2
---------------
 VIA 1.000 time units
---------------
Time: 3.000
Location: closing.on_off_2nd
sigma=1 & switch=0 & barrel=0 &
4contents=3 & e=1 & t=3
--------------------------------
  VIA:
--------------------------------
Time: 3.000
Location: closed.on_off_2nd
 sigma=1000 & switch=0 & barrel=0
& 4contents=3 & e=0 & t=3
---------------
 VIA 2.000 time units
---------------
Time: 5.000
Location: closed.on_off_2nd
 sigma=1000 & switch=0 & barrel=0
& 4contents=3 & e=2 & t=5
--------------------------------
  VIA: on
--------------------------------
Time: 5.000
Location: open.on_off_fin
 sigma=1000 & switch=1 & barrel=0
& 4contents=3 & e=0 & t=5
---------------
 VIA 16.500 time units
---------------
Time: 21.500
Location: open.on_off_fin
 sigma=1000 & switch=1 & barrel=0
& contents=9 & 2e=33
 & 2t= 43
--------------------------------
  VIA:
--------------------------------
Time: 21.500
Location: closing.on_off_fin
 sigma=1 & switch=1 & barrel=0 &
contents=9 & e=0 & 2t=43
---------------
 VIA 1.000 time units
---------------
Time: 22.500
Location: closing.on_off_fin
 sigma=1 & switch=1 & barrel=0 &
4contents=37 & e=1 & 2t=45
--------------------------------
  VIA:
--------------------------------
Time: 22.500
Location: closed.on_off_fin
 sigma=0 & switch=1 & barrel=10 &
4contents=37 & e=0 & 2t=45
--------------------------------
  VIA:
--------------------------------
Time: 22.500
Location: open.on_off_fin
 sigma=1000 & switch=1 & barrel=0
& contents=0 & e=0 & 2t=45
```

**Fig. 5.** Reachable states for the barrel filler model

is used for representing infinity since HyTech doesn't support such expression. Analysis on the result confirmed that the translated linear hybrid automaton shows the same behavior with the DEV&DESS model illustrated in ⟨Fig.2⟩.

The HyTech symbolic model checker performs a couple of valuable analysis on the linear hybrid automata models, i.e., model checking of safety properties and parametric analysis. The barrel filler system model has a safety requirement such as '*Content of barrel should not over 10 liter*'. For safety verification, we added the monitor automata, the unsafe state '*error*' in ⟨Fig.4⟩, which reports when the '*unsafe*' state is entered. The model can be confirmed the satisfaction of the safety requirements by finding the path to the unsafe state. HyTech can perform such analysis by computing all reachable states and checking whether the unsafe state can be reached or not. By execution of HyTech, we were successfully able to get the result that the model satisfies the safety requirement.

We defined the statement for parametric analysis as '*When the valve should start closing to avoid overflowing water of barrel?*'. It will find out constraints for the cutoff value which will result in water overflowing. First, HyTech searches the reachable states. Then the predicates for the necessary and sufficient conditions for visiting unsafe state are calculated by existential quantification using the reachable states and the unsafe state. The result denotes that the barrel always overflows, that is, unsafe value for parameter cutoff. As executed with such series of processes, the predicate `'4cutoff > 39'` for the unsafe cutoff value was output, and from the negation of this result, the safe cutoff value can be derived, `'4cutoff <= 39'`. It also implies if the cutoff value is 9.75, then the model puts a ten liter filled barrel exactly.

## 5 Further Considerations on the Translation from DEV&DESS into Linear Hybrid Automata

Components structuring DEV&DESS and linear hybrid automata are different from each other, and we need to map one to the other appropriately while keeping their behavioral equivalence. First of all, we used the concept of 'phase[22]' in the DEV&DESS formalism to compose control modes which are the fundamental elements of linear hybrid automata. The biggest challenge for the translation is how to express I/O behavior of the DEV&DESS formalism in linear hybrid automata. Our solution in the paper is to use parallel automata which shows the same behavior with the original input trajectories of the DEV&DESS model. The basic ideas for translation are as follows:

- The variable 'e' is added to symbolize the elpased time in DEV&DESS. Its value is refreshed at every every control mode changes.
- The variable '$\sigma$' is used for time advanced function '*ta*'.
- A labeling function '*init*' is derived from initial condition.
- A labeling function '*flow*' is derived from change of rate function '$f$' with continuous input value.
- A labeling function '*jump*' is derived from state and external events. The conditions for time event is also included at every control mode.
- A labeling function '*inv*' is derived from state events and time events.

Based on the guidelines above, we could translate the DEV&DESS model into an equivalent linear hybrid automata model successfully. We could also perform formal verification of HyTech on the translated models too. We, however, found some problems which should be resolved for more seamless translations.

- In the case that the external and internal events conflict in DEV&DESS model, assignment the order of priority between jump conditions is difficult, since hybrid automata does not provide any such semantics. Therefore, It may happen that exploring unintended trajectories of transformed automata.
- If the continuous input trajectories are the form of stairs, figuring out the whole input trajectories is required before translation even if it satisfies '*flow independece condition*'. It is due to that the phase may map to the several control modes, not one by one, since HyTech doesn't allow using variables to express rate condition.

## 6 Conclusion

This paper translated an atomic DEV&DESS model for a barrel filler system into linear hybrid automata, and then performed model checking of safety requirement and parametric analysis using the HyTech symbolic model checker. Translation from the DEV&DESS formalism into linear hybrid automata has its won significance, since verification tools dealing with DEV&DESS have not been proposed while DEV&DESS has been used as a basic formalism for various

modeling tools, such as CHARON and ECML. Therefore, if the translation is well defined and a mechanical translator is also supported, then verifying the model can be done by using existing tools such as HyTech. We are now developing translations for coupled DEV&DESS formalism to linear hybrid automata and transition rules for the broad applications.

**Acknowledgement.** This work was supported by the IT R&D Program of MKE/KEIT [10035708, "The Development of CPS(Cyber-Physical Systems) Core Technologies for High Confidential Autonomic Control Software"]. This research was partially supported by the National IT Industry Promotion Agency (NIPA) under the program of Software Engineering Technologies Development.

## References

1. Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T.A., Ho, P.H., Nicollin, X., Olivero, A., Sifakis, J., Yovine, S.: The algorithmic analysis of hybrid systems. Theoretical Computer Science 138(1), 3–34 (1995)
2. Alur, R., Dang, T., Esposito, J., Hur, Y., Ivančić, F., Vijay Kumar, I.L., Mishra, P., Pappas, G.J., Sokolsky, O.: Hierarchical modeling and analysis of embedded systems. Proceedings of the IEEE 91(1), 11–28 (2003)
3. Alur, R., Dill, D.L.: A theory of timed automata. Theoretical Computer Science 126 (1994)
4. Alur, R., Henzinger, T.A., Ho, P.H.: Automatic symbolic verification of embedded systems. IEEE Transactions on Software Engineering 22(3), 181–201 (1996)
5. Antsaklis, P.J., Stiver, J.A., Lemmon, M.D.: Interface and Controller Design for Hybrid Control Systems. In: Antsaklis, P.J., Kohn, W., Nerode, A., Sastry, S.S. (eds.) HS 1994. LNCS, vol. 999, pp. 462–492. Springer, Heidelberg (1995)
6. Asarin, E., Dang, T., Maler, O.: The **d/dt** Tool for Verification of Hybrid Systems. In: Brinksma, E., Larsen, K.G. (eds.) CAV 2002. LNCS, vol. 2404, pp. 365–770. Springer, Heidelberg (2002)
7. Balluchi, A., Benvenuti, L., Benedetto, M., Pinello, C., Sangiovanni-Vincentelli, A.: Automotive engine control and hybrid systems: challenges and opportunities. Proceedings of the IEEE 88(7), 888–912 (2000)
8. Chutinan, A., Krogh, B.H.: Verification of Polyhedral-Invariant Hybrid Automata Using Polygonal Flow Pipe Approximations. In: Vaandrager, F.W., van Schuppen, J.H. (eds.) HSCC 1999. LNCS, vol. 1569, pp. 76–90. Springer, Heidelberg (1999)
9. Clarke, E.M., Grumberg, O., Peled, D.A.: Model Checking. MIT Press (1999)
10. Daws, C., Olivero, A., Trypakis, S., Yovine, S.: The Tool Kronos. In: Alur, R., Sontag, E.D., Henzinger, T.A. (eds.) HS 1995. LNCS, vol. 1066, pp. 208–219. Springer, Heidelberg (1996)
11. Esposito, J.M., Kim, M.: Using formal modeling with an automated analysis tool to design and parametrically analyze a multirobot coordination protocol: A case study. IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans 37(3), 285–297 (2007)
12. Henzinger, T.A., Ho, P.H., Wong-Toi, H.: Hytech: a model checker for hybrid systems. Software Tools for Technology Transfer 1(1-2), 110–122 (1997)
13. Henzinger, T.A., Wong-Toi, H.: Using Hytech to Synthesize Control Parameters for a Steam Boiler. In: Abrial, J.-R., Börger, E., Langmaack, H. (eds.) Dagstuhl Seminar 1995. LNCS, vol. 1165, pp. 265–282. Springer, Heidelberg (1996)

14. Kim, T.G., Sung, C.H., Hong, S.Y., Hong, J.H., Choi, C.B., Kim, J.H., Seo, K.M., Bae, J.W.: Devsim++ toolset for defense modeling and simulation and interoperation. The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology 8(3), 129–142 (2011)
15. Lee, D.A., Lee, J.H., Yoo, J., Kim, D.H.: Systematic verification of operational flight program through reverse engineering. In: International Conference on Advanced Software Engineering & Its Applications (submitted, 2011)
16. Mitchell, I., Tomlin, C.J.: Level Set Methods for Computation in Hybrid Systems. In: Lynch, N.A., Krogh, B.H. (eds.) HSCC 2000. LNCS, vol. 1790, pp. 310–323. Springer, Heidelberg (2000)
17. Praehofer, H.: Systems Theoretic Foundations for Combined Discrete Continuous System Simulation. Ph.D. thesis, Department of Systems Theory, University of Linz, Autria (1991)
18. Praehofer, H., Auernig, F., Reisinger, G.: An environment for devs-based multiformalism simulation in common lisp/CLOS. Discrete Event Dynamic Systems: Theory and Application 3, 119–149 (1993)
19. Praehofer, H., Pree, D.: Visual modeling of devs-based multiformalism systems based on higraphs. In: Simulation Conference Proceedings, pp. 595–603 (December 1993)
20. Tomlin, C., Pappas, G., Sastry, S.: Conflict resolution for air traffic management: a study in multiagent hybrid systems. IEEE Transactions on Automatic Control 43(4), 509–521 (1998)
21. UPPAAL (2010), `http://www.uppaal.com/`
22. Zeigler, B.P., Praehofer, H., Kim, T.G.: Theory of Modeling and Simulation. Academic Press (2000)