Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 309 (2014) 3–12

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

A Formal Framework for Hybrid Event B

Jie Liu[a,b], Jing Liu[a,1]

[a] *Shanghai Key Laboratory of Trustworthy Computing*
Software Engineering Institute, East China Normal University
3663 Zhongshan Road(North), Shanghai, China, 200062

[b] *School of Computer Science and Technology*
University of South China
28 Changsheng(West), HengYang, China, 421001

Abstract

In this paper, we present Hybrid Event B, a formal language for modeling hybrid systems. Specifically, Hybrid Event B is an extension of Event B associating with differential dynamic logic. The main contribution of this paper is that we give the definition of a differential event in Hybrid Event B, which makes it possible using differential dynamic logic in modeling continuous dynamical systems and discrete dynamical systems. In addition, we show the proof obligations of each refinements on differential events, which supports the stepwise development of refinement. We also show the transformer semantics of the differential events and the weakest precondition refinement approach applied to hybrid systems, both of which support our stepwise development of hybrid systems.

Keywords: Event B, differential dynamic logic,continuous refinement, hybrid systems

1 Introduction

Hybrid systems are dynamical systems that combine discrete and continuous dynamics[1]. They are important for modeling embedded systems and Cyber-Physical Systems. Hybrid systems are very natural models for many application scenarios, because each part of the system can be modeled in the most natural way[2,3]. Discrete aspects of the system, such as discrete switching, computing, and control decisions can be modeled by discrete dynamics, while continuous aspects of the system, such as motion or continuous physical processes can be modeled by continuous dynamics. And hybrid systems simply combine either kind of dynamics with each other as one hybrid system in very flexible ways. The formal development of such embedded systems should then consider two different frameworks: the discrete framework of the controller and the continuous framework of the environment. As the purpose of Hybrid systems, the formal development of such closed systems

[1] Corresponding: jliu@ecnu.edu.cn

http://dx.doi.org/10.1016/j.entcs.2014.12.002

should be able to deal with these dual frameworks. In this short paper, We present some ideas and principles underlying the realization of a new project called Hybrid Event B. This project certainly absorbs the main ideas and principles already at work in Event B [4], but the kernel mathematical language of this project contains differential dynamic logic[5].

From Event B, it borrows the idea of refinement and proof. Event B [4,6] describes a formal framework for discrete system which provides a great deal of flexibility to cover a large range of system developments. A complex system model with details will be established through stepwise refinement. More concrete implementation details are provided by introducing new events and data to the system in every refinement step. Necessary proof obligations from the construction of abstract framework to refinement verified each refinement step. The transitivity of refinement ensures that the final system description is equal to the initial one. The multiple cases from diverse areas are studied and shows that this goal has been achieved. Refinement process are provided as well as precise proof obligations on all kinds of events and some tools support to discharge them [7,8,9]. In essence, Event B is a state-based specification technique like Z [10] or Alloy [11]. Therefore proof obligations clarify the reason about predicates on states.

From differential dynamic logic, it borrows flexible power in handling symbolic parameters with logical variables about hybrid systems. After continuous deduction and development, it finally forms a single specification language, which is a combination of differential dynamic logic descriptions and descriptions of system behavior and correctness statements about the system state[12]. In particular, it extends discrete dynamic logical by differential actions such that it can display continuous evolution. Due to the symbolic nature of logic, it is beneficial to use simple system actions of an isolated effect in behavior.

This paper is organized as follows. The next section introduces the necessary background on Event B and differential dynamic logic. Section 3 presents formal syntax, semantics and refinement of the Hybrid Event B. We give an application example of train control system in section 4 which shows the usage of our model language in specifying properties of hybrid systems. Section 5 introduces related work. We draw conclusions and discuss future work in Section 6.

2 Background

2.1 Event B

Event B is a specification language based on set theory and standard first-order predicate logic applied in large reactive and distributed systems. Its platform supports writing and proving of specifications. The basic concept of the specification development process is the refinement, which finally construct a precise model by steps. Machine is an important component of social Event B. It defines the dynamic behavior of the model, which consists of variables, invariants, variants, and events. Variable are divided into various types whose values may be integers, sets, relations, functions etc. Invariants refer to the system safety properties which re-

strain the value range of variables. Variants concern the correction of refinements. Events describe transitions from one state to another, which consist of guards and actions. In events, a guard is a predicate and an action is responsible for assigning a state variable with a generalized substitution. When a guard is true, an event is to be triggered even with non-deterministic selection. The execution of the actions of a special event is simultaneous. A machine will evolve into a new one which contains more details and more concrete description of the model as new variables and events are introduced in refinement. By strengthening the guards and adding actions to the new variables, abstract events become distinct. The static elements of the model are defined by contexts which are composed of carrier sets, constants, axioms, and theorems. The axioms and theorems are specified with the notation of first-order logic and set theory. The static information of a model related to the refinement constitutes a context that is visible to several machines and machines in turn can see several contexts. Some new events can permit older ones triggering by introducing variants.

2.2 Differential Dynamic Logic

The differential dynamic logic is a dynamic logic for describing hybrid systems [13,14]. As a dynamic logic it has two relevant parts. On the one hand, it has the classical first-order part. On the other, it has the program world in the modalities. The programs of differential dynamic logic are regular combinations of discrete changes of system state and continuous evolutions of the system variables. Discrete states are changed by assignments, such as $x := 10$. The assignment can model states change of a controller component. Continuous evolutions can be modeled by differential equations and invariants. The continuous evolutions of variables are used to model the continue behavior of hybrid systems. The atomic statement of differential dynamic logic are instantaneous discrete jump assignments $x := \theta$, a first-order formula H of real arithmetic, and differential equation systems $x' = \theta \wedge H$ for a continuous evolution restricted to the domain of evolution described by H. Compound statement of differential dynamic logic are generated from these atomic programs by nondeterministic choice $\cup$, sequential composition ;, and Kleene's nondeterministic repetition $*$.

3 Hybrid Event B

3.1 Syntax

In the Hybrid Event B, specifications are organized and presented as Event B,and formal logical based models are immediately processable by differential dynamic logic, which performs type checking, records dependencies and generates obligations for proving consistency. The appropriate mathematical model for embedded control systems is hybrid systems that combines the traditional state machine based models for discrete control with classical differential equations based models for continuously evolving physical activities.

A Hybrid Event B model is composed of two components: machine and contexts. Machines contain the system variables, invariants, variants, and events, which define the dynamic behavior of the model. Contexts contain carrier sets, constants, axioms, and theorems, which define the static elements of the model.

CONTEXT
    Sets
    Contants
    Axioms
MACHINE
    variables;
    invariants;
    variants;
    events.

Central to a Hybrid Event B description is the definition of the events, called differential event, each consisting of a guard G over the variables, and a body, usually written as an assignment S on the variables. Such events are constructed as:

event:= WHEN
        G(V)
    THEN
        S

The atomic body S is instantaneous discrete jump assignments $x := \theta$ and differential equation $x' = \theta \wedge H$ for a continuous evolution restricted to the domain H of evolution described by a first-order formula. The body describes the changes of variables upon event execution. Flexibility of differential dynamic logic will be used to represent the progression of system values along states over time during a hybrid evolution. Further differential dynamic logic clauses provide a list of state variables, an invariant constraining and relating the state variables, and operations on the state variables, the latter having optional parameters and results.

3.2 Semantics

Discrete transitions are described as instantaneous assignments of values to state variables. They implicit discrete state changes. In hybrid automata, the discrete transitions attached to edges which describe the evolution modes of hybrid systems from one node to the other. They can be expressed resets x:=y or adjustments of variables like x := x + 1. When simultaneous changes of multiple variables are handled, discrete transitions can be combined to sets of discrete transitions with simultaneous effect following corresponding techniques in the discrete case. For example, the discrete transitions set $x := x - 8$, $y := 2x$ expresses that x is decreased by 8 and, simultaneously, variable y is set to $2x$, which is evaluated before a receives its new value. If x initial value is 10, then x=2,y=20.

Continuous variation in system dynamics is described using differential event $x' = \theta \wedge H$ as evolution constraints. The differential equation $x' = \theta$ is used for

modeling and inference about continuous behavior. H is a predicate invariant which constraint the area of all the values. For example the continuous movement of a train along the track, train position z evolves with velocity v. $z' = v, v' = a \wedge v \leq 100$ expresses that the evolution only applies as long as the speed is $v \leq 100$. This is an evolution along the differential equation $z' = v, v' = a$ that is restricted to remain within the region $v \leq 100$, i.e., to accelerate before $v \leq 100$. Such an evolution can stop at any time within $v \leq 100$, it could even continue with transient grazing along the border $v = 100$, but it is never allowed to enter $v > 100$.

Definition 1 (Transition semantics of differential event) *The two tuples $(\upsilon, \omega)$is a transition relation on states. It specifies which state ω is reachable from a state υ by differential event of the hybrid systems and is defined as follows:*

1. $(\upsilon, \omega) \in DE(x_1 := \theta_1, ..., x_n := \theta_n)$ iff $\upsilon([x_1/\theta_1] \ldots [x_n/\theta_n])$ equals state ω. Additionally, the value of the other variables $y \notin x_1, ..., x_n$ keeps the original value, i.e., $DE(\upsilon, y) = DE(\omega, y)$

2. $(\upsilon, \omega) \in DE(x_1' := \theta_1, \ldots, x_n' := \theta_n \wedge H)$ iff for time $t \geq 0$, there is a function $f : [0, t] \to$ States with $f(0) = \upsilon, f(t) = \omega$. The differential equations is respected: for each x_i, $DE(f(\tau), x_i)$ is continuous in τ on [0, t] and has a erivative of value $DE(f(\tau), \theta_i)$ at each time $\tau \in (0, r)$. Additionally, The value of other variables $y \notin x_1, ..., x_n$ keeps the original value, i.e., $(f(\tau), y) = DE(\upsilon, y)$ for all $\tau \in [0, t]$. Further, the invariant holds, i.e.,$DE(f(\tau) \wedge H) = true$ for each $\tau \in [0, t]$.

3.3 Refinement of Model

It is clearly impossible to build a single model representing once and for all the future system because of the size of the state and the number of its transitions. We are rather going to construct an ordered sequence of models, where each model is supposed to be a refinement of the one preceding it in the sequence. This means that there may be some changes in a refined model usually. The refinement relation defined within Hybrid Event B possesses some fundamental properties which are of great practical importance in performing incremental development and proof. In hybrid systems, we pay attention to from a continue systems to a continue ones. The refinement relation is transitive, which means that we can incrementally verify the final implementation by verifying each individual refinement step. The refinement relation is monotonic with respect to all the constructs of the dynamic logic language, which means that the subcomponents of an operation refinement can be refined independently. The refinement of an dynamic logic language incurs a number of proof obligations. We consider the most general case of simultaneous data and algorithmic refinement.

Refinement of Hybrid Event B, as formalized in the refinement calculus, is based on the following definition[16]. An differential event DE_1 is refined by an differential event DE_2, denoted by $DE_1 \sqsubseteq DE_2$, if the differential event DE_2 can reach all the same post-states Q at least from the same pre-states as the differential event DE_1, i.e.$\forall Q : wp(DE_1; Q) \Rightarrow wp(DE_2; Q)$. But It is quite difficult to use the complex weakest precondition semantics for the differential system. Therefore, we would like

to have an alternative, but equally powerful, characterization for the refinement, which is more useful.

For differential event $x' = \theta \wedge H$ satisfying definition 1, a continuous transitions satisfy a continuous evolution in accordance with the differential equation $x' = \theta$. Then, its value is continuous on [0, t] and have a continuous derivative on the open interval (0, t). States is isomorphic to real space spanned by the variables of the differential equations, because derivatives are not defined on the closed interval [0, t] if t = 0. For the purpose of a differential event, states are fully determined by an assignment of a real value to each occurring variable, which are finitely many. A continuous transition in accordance with $x' = \theta$ is possible from υ to ω. Therefore, there is a continuous function f of time $t \geq 0$ from state υ to ω such that f gives a solution of the differential equation $x' = \theta$. H is invariant region that constraint f always resides within H during the whole evolution. Further, only variables subject to a differential equation change during such a continuous transition.

Lemma 1 *Differential event have unique solutions[12][Lemma 1]. "For each differential event , each state υ and time $t \geq 0$, there is at most one function $f : [0, t] \to$ States satisfying transition semantics of differential event.*

Proof. Let be a differential event $x'_1 := \theta_1, \ldots, x'_n := \theta_n$ with invariant region H. Using simple computations in the field of rational fractions, we can assume the right-hand sides θ_i of the differential equations to be of the form p_i/q_i for polynomials p_i, q_i. The set of points in real space where $q_i = 0$ holds is closed. As a finite union of closed sets, the set where $q_1 = 0 \vee \ldots \vee q_n = 0$ holds is closed. Hence, the valuations of the i are continuously differentiable on the complement of the latter set, which is open. Thus, as a consequence of Picard.Lindelofs theorem, the solutions are unique on each connected component of this open domain. Consequently, solutions are unique when restricted to H, which, by assumption, entails $q_1 \neq 0 \wedge \ldots \wedge q_n \neq 0$."

Lemma 2 *Consider two differential event $DE_1 : X'_1 = \theta \wedge H_1$ and $DE_2 : X'_2 = \phi \wedge H_2$. DE_1 is refined by DE_2, $DE_1 \sqsubseteq DE_2$. The proof obligation of invariant region H is $H_1 \Leftrightarrow H_2$.*

Proof. We assume $H_1 \Rightarrow H_2$ does not hold. Then, there exists invariant region F such that $F \Rightarrow H_1$ holds, but $F \Rightarrow H_2$ does not hold. For differential event DE_1 and DE_2, there exits two solutions f and g. The f and g are continuous in duration $t \geq 0$. Since all the evolutions are continuous, there exists a post-states Q and $F(f(\xi)) \wedge \neg Q = wp(DE_1, Q)$. Because $F \Rightarrow H_2$ does not hold, and so $F(g(\varepsilon)) \wedge \neg Q = wp(DE_2, Q)$ does not hold. And hence, $wp(DE_1; Q) \Rightarrow wp(DE_2; Q)$ does not hold. Therefore, $DE_1 \sqsubseteq DE_2$ cannot hold. The case $H_2 \Rightarrow H_1$ is proven similarly.

Lemma 3 *Let DE_1 and DE_2 are differential event as in Lemma 2, f and g be their solutions, τ and ς be the termination time of f and g. DE_1 is refined by DE_2, $DE_1 \sqsubseteq DE_2$. The proof obligation of the solutions that f and g are isomorphism.*

Proof. First f and g termination states are the same. Because $DE_1 \sqsubseteq DE_2$, we have the same postcondition Q and $Q[f(\tau)/X]) = Q[g(\varsigma)/X]$. This holds only if $f(\tau) = g(\varsigma)$.

Next the f and g reach the same states. Because $DE_1 \sqsubseteq DE_2$ holds, we know

that f and g is unique solutions of DE_1 and DE_2 by Lemma 1. DE_2 must reach all the same post-states at least from the same pre-states as DE_1 . Therefore, the same states are reached.

Last f and g reach the same states by the same order. By definition 1, the evolutions of f and g are continuous and determined. Now, consider a post-state Q that is reachable by DE_1. Due to uniqueness and finiteness of evolutions all the states $wp(DE_1; Q)$ are connected by one and only one continuous finite evolution. Similarly, all the states $wp(DE_2; Q)$ are connected by a continuous finite. But then, f and g must reach the same states by the same order, because otherwise f either contains undetermined or is discontinuous.

4 Case study: train control system

The example comes from[16] to illustrate use of weak simulation proof obligations. In this example a train travels a distance of 2560 m. It starts by accelerating to traveling velocity, which it then keeps for the most of the journey. At the end, the train decelerates to a full stop. Fig. 1 depicts how the velocity changes during the journey. We develop now a Hybrid Event B modeling the traveling train. We start with a discrete model describing only the state changes, and add the details of continuous time dynamics in a stepwise manner.

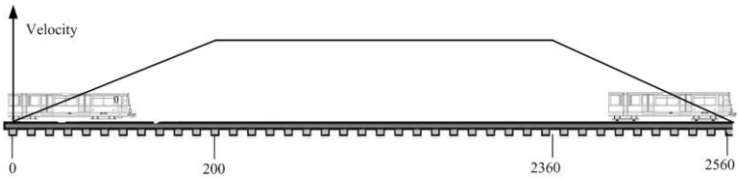

Fig. 1. travelling train

Initial specification: The hybrid system is the initial specification describing only the discrete state changes, the jumps from one event to another, during the journey.

accel when $0 \leq x < 200$ then x:=200	const when $200 \leq x < 2360$ then x:=2360	decel when $2360 \leq x < 2560$ then x:=2560

Refinement specification: we introduce the notion of velocity by adding acceleration and deceleration to the specification. Such a hybrid action system, where v denotes the velocity, is

accel	const	decel
when	when	when
$0 \leq x < 200$	$200 \leq x < 2360$	$2360 \leq x < 2560$
then	then	then
$x' := V \wedge V' = 1$	$x' := V$	$x' := V \wedge V' = -1$

We can get the refinement relation used for proving the correctness must describe what we mean by acceleration and deceleration:

1. accel refinement relation: $v = \sqrt{2x} \quad \wedge 0 \leq x < 200$
2. const refinement relation: $v = 20 \wedge 200 \leq x < 2360$
3. decel refinement relation: $v = \sqrt{5120 - 2x} \quad \wedge \ 2360 \leq x < 2560$

The proof obligations (Lemma 1) and (Lemma 2) hold trivially, because there are no new intermediate actions in train. for each differential event and each duration , there is at most refinement relation satisfying. Differential equations have unique solutions, so Lemma3 hold also.

5 Related work

Hybrid systems do not accept equivalent finite-state abstractions owing to general limits of numerical approximation[17,18]. Zhou et al. presented a few possible methods in extraction and properties research on piecewise continuous states [19]. One useful way for reasoning about hybrid systems with a mixture of continuous and discrete states is to apply multiple calculus rules and a non-constructive oracle by extending duration calculus with mathematical expressions. Although continuous statements [20] and a hybrid variant [21] are introduced into CSP in the form of differential equations to describe hybrid systems with extended duration calculus, The verification method is not presented. Davoren and Nerode [22,23] have done a lot of researches in semantics of modal u-calculus and topological aspects of hybrid systems. Hilbert-style calculi was presented to prove formulas which are valid for hybrid systems simultaneously. However, system behavior especially some unknown effects should be axiomatized declaratively in propositional modal logics with terms of abstractions. So Hilbert-style calculi cannot obtain enough information about a particular system. Ronkko et al.[24] proposed a weakest precondition semantics in higher-order logic by built-in derivatives. But this method can only provide a notational variant of classical mathematics because it is hard to verify the higher-order logic. To obtain a sound development approach for hybrid systems, we present Hybrid Event B that combine both the Event B as stepwise refinement and the

differential dynamic logic as a significantly more expressive specification language. Finally, we think Hybrid Event B as a more uniform model language for hybrid systems that is amenable to stepwise development for hybrid systems.

6 Conclusion and future work

In hybrid systems design and modeling, the problem is how to from a continue systems to a continue ones, together with specifying logistical properties or constrains. For modeling continue dynamic systems and refinement relations, we have proposed a differential dynamical logical based. The logic is used to express the logical connections between all kinds of expressions in hybrid systems. Thus, discrete system and continue system can specification together. We have constructed Hybrid Event B modeling language for hybrid Systems, which is supposed to be a refinement of the one preceding it in the sequence. Then, we give formal semantics of differential event of Hybrid Event B. Finally, the proof obligations of refinement is showed. In the future, we will work on the verification and tool support of Hybrid Event B.

Acknowledgment

We would like to thank the reviewers of this paper for their valuable comments. This work is partially supported by the projects funded by the 973 Program 2009CB320702, NSFC 91318301, 61170084 and 61073022. The Shanghai Key Lab is supported by Shanghai Knowledge Service Platform ZF 1213, 863 Project 2011AA010101 and Science and technology project of Hunan Province 2011GK3192.

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.Theor. Comput. Sci. 138(1), 3-34 (1995)

[2] Davoren, J.M., Nerode, A.: Logics for hybrid systems. IEEE 88(7), 985-1010 (July 2000)

[3] Henzinger, T.A.: The theory of hybrid automata. In: LICS. pp. 278-292. IEEE Computer Society, Los Alamitos (1996)

[4] J-R. Abrial. Modeling in Event-B: System and Software Engineering. Cambridge University Press, 2010.

[5] A. Platzer, Differential dynamic logic for hybrid systems. J. Autom. Reas., vol. 41, no. 2, pp. 143C189, 2008.

[6] C. Metayer, J.-R. Abrial, and L. Voisin. Event-B language, 2005. RODIN Project Deliverable 3.2, http://rodin.cs.ncl.ac.uk/deliverables/D7.pdf, accessed 25/5/10. bf

[7] M. J. Butler and S. Hallerstede. The Rodin formal modeling tool. In BCS-FACS Christmas 2007 Meeting Formal Methods In Industry, 2007.

[8] J-R. Abrial, M. J. Butler, S. Hallerstede, and L. Voisin. A road map for the Rodin toolset. In E.Borger, M. J. Butler, J. P. Bowen, and P. Boca, editors, ABZ, volume 5238 of Lecture Notes in Computer Science, page 347.Springer, 2008.

[9] J-R. Abrial, M. J. Butler, S. Hallerstede, T. S. Hoang, F. Mehta, and L. Voisin. Rodin: an open toolset for modelling and reasoning in Event-B. STTT, 12(6):447-466, 2010.

[10] J. C. P. Woodcock and J. Davies. Using Z: Specification, Re nement, and Proof. Prentice Hall, 1996.

[11] D. Jackson. Alloy: a lightweight object modelling notation. ACM Trans. Softw. Eng. Methodol., 11(2):256-290, 2002

[12] A. Platzer, Differential-algebraic dynamic logic for differential-algebraic programs, J. Log. Comput., vol. 20, no. 1, pp. 309C352, 2010.

[13] A. Platzer, Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics. Heidelberg: Springer, 2010.

[14] A. Platzer,The complete proof theory of hybrid systems, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-CS-11-144, Nov 2011.

[15] R. J. R. Back. Refinement calculus, part II: Parallel and reactive programs. In J.W. de Bakker, W.-P. de Roever, and G. Rozenberg, editors, Stepwise Refinement of Distributed Systems: Models, Formalisms, Correctness. Proceedings. 1989, volume 430 of Lecture Notes in Computer Science. Springer-Verlag, 1990.

[16] M.Ronkko, K.Sere, Refinement and continuous behavior, in: F.W. Vaandrager, J.H. van Schuppen(Eds.), Proceedings of Hybrid Systems: Computation and Control, Lecture Notes in Computer Science,Vol. 1569, Springer, Berlin, 1999, pp. 223C237.

[17] Platzer, A., Clarke, E.M.: The image computation problem in hybrid systems model checking. In: Bemporad, A., Bicchi, A., Buttazzo, G. (eds.) HSCC, LNCS, vol. 4416, pp. 473C486. Springer, Berlin (2007)

[18] Henzinger, T.A.: The theory of hybrid automata. In: LICS, pp. 278C292. IEEE Computer Society, Los Alamitos (1996)

[19] Zhou, C., Ravn,A.P., Hansen, M.R.:An extended duration calculus for hybrid real-time systems.In: Grossman, R.L., Nerode, A., Ravn, A.P., Rischel, H. (eds.) Hybrid Systems, LNCS, vol. 736,pp. 36C59. Springer, Berlin (1992)

[20] Jifeng, H.: From CSP to hybrid systems. In: Roscoe, A.W. (ed.) A Classical Mind: Essays in Honour of C. A. R. Hoare, pp. 171C189. Prentice Hall, Hertfordshire (1994)

[21] Chaochen, Z., Ji,W., Ravn,A.P.:Aformal description of hybrid systems. In: Alur, R., Henzinger, T.A., Sontag, E.D. (eds.) Hybrid Systems, LNCS, vol. 1066, pp. 511C530. Springer, Berlin (1995)

[22] Davoren, J.M., Nerode, A.: Logics for hybrid systems. Proc. IEEE 88(7), 985C1010 (2000).

[23] Davoren, J.M.: On hybrid systems and the modal -calculus. In: Antsaklis, P.J., Kohn, W., Lemmon, M.D., Nerode, A., Sastry, S. (eds.) Hybrid Systems, LNCS, vol. 1567, pp. 38C69. Springer, Berlin (1997).

[24] Ronkkon,M., Ravn, A.P., Sere, K.: Hybrid action systems. Theor. Comput. Sci. 290(1), 937-73(2003)