Case Study : Verification of ECML Model Using SpaceEx

Jaeyeon Jo[1,a)] Sanghyun Yoon[1,b)] Junbeom Yoo[1,c)] Hae Young Lee[2,d)] Won-Tae Kim[2,e)]

Received: xx xx, xxxx, Accepted: xx xx, xxxx

Abstract: We present a formal verification of a hybrid system from an example of a barrel-filler system. Hybrid system composes continuous elements and discrete elements. ECML (ETRI CPS Modeling Language) is a modeling formalism for hybrid systems, recently proposed by a research institute - ETRI in Korea. It extends a basic formalism DEV& DESS (Discrete EVent & Differential Equation System Specification) with various conveniences in modeling and simulation. An ECML model checking tool has not been developed yet. SpaceEx is a verification tool for hybrid automata using reachability algorithms. It integrates various approaches of hybrid system verification. The case study illustrates how verification can be performed on an ECML model using a hybrid automata translation by hands.

Keywords: Formal Verification, Hybrid System, Hybrid Automata

1. Introduction

Hybrid system models combination of continuous elements and discrete elements. The discrete part models computing elements while the continuous part models physical elements. Those elements are included in embedded systems such as automotive, medical, and avionic systems. Hybrid system has been modeled with the purpose of proving properties such as reachability, safety, and stability. A current guideline for formal verification of hybrid system is shown at [1]. HyTech[7], PHAVer[4], and SpaceEx[5] are verification tools using symbolic reachability analysis. KeyMaera[12] and HSolver[13] are verification tools using deductive proving. ECML is developed as a hybrid system modeling language for improving convenience. It extends DEV & DESS [14]. ETRI proposed it to develop[9] a cyber physical system. It needs formal verification tool to satisfy safety requirements. The reachability analysis tool for hybrid system could be used for ECML. A case study[3] is about verification of DEV & DESS using HyTech. A translation, from DEV & DESS into linear hybrid automata(LHA) input front-end of HyTech, has limitation of DEV & DESS expression which is restricted by linearity. HyTech verifies linear hybrid automata with restrictions[8]. For example, dynamics of hybrid automata are of the form $ax + b = 0$, where a and b are constants and x is variable. Using other model checking tools extends range of verifiable ECML model. SpaceEx is a tool framework[6] that uses support function[11] for non-linear hybrid automata. It also integrates PHAVer that is a model checking tool for linear hybrid automata. We expect that the wide range of ECML model can be verified using SpaceEx. In this paper, We show a case study about verification of hybrid system from an example of barrel-filler system using SpaceEx. We model barrel-filler system for ECML, then translate it into hybrid automata those behavior is same as the ECML model by hands. Then we verify the hybrid automata using SpaceEx with safety properties.

The rest of the paper is organized as follows. In the next section, we present the semantics of ECML, and hybrid automata for SpaceEx. Then in Section 3, we describe about barrel-filler model verification using SpaceEx. Our conclusion and ongoing research topic are presented in the last section.

2. Background

2.1 ECML

ETRI(Electronics and Telecommunications Research Institute) proposed ECML that extends DEV & DESS with various convenience such as hierarchies and error modeling. DEV & DESS has two flow type of values : discrete value and continuous value as opposed to ECML has three flow type of values : discrete value, event value and continuous value. Continuous value changes differential, discrete value changes drastically and preserved. Event value also changes drastically but not preserved. The definition of Basic model of ECML is belows :

[1] Konkuk University
[2] Electronics and Telecommunications Research Institute.
[a)] tm77@konkuk.ac.kr
[b)] pctkdgus@konkuk.ac.kr
[c)] jbyoo@konkuk.ac.kr
[d)] haelee@etri.re.kr
[e)] wtkim@etri.re.kr

$$BM = \langle X, Y, S, Trans^E, Trans^S, Cond^S, Rate, Out^C, Out^D, Out^E, Out^S \rangle$$

$$X = X^C \times X^D \times X^E$$

- X^C is the set of continuous input value
- X^D is the set of discrete input value
- X^E is the set of event input value

$$Y = Y^C \times Y^D \times Y^E$$

- Y^C is the set of continuous output value
- Y^D is the set of discrete output value
- Y^E is the set of event output value

$$S = S^C \times S^D \times P$$

- S^C is the set of continuous state value
- S^D is the set of discrete state value
- P is the set of phase
- $Trans^E : S \times X \to S$ is the external event transition function.
- $Out^E : S \times X^C \times X^D \to Y^E$ is the discrete event output function for external event functions.
- $Cond^S : S \times X^C \times X^D \to Bool$ is the state transition condition function.
- $Trans^S : S \times X^C \times X^D \to S$ is the state event transition function.
- $Out^S : S \times X \to Y^E$ is the discrete output function for internal state transition.
- $Rate : S \times X^C \times X^D \to S^C$ is the rate of change function
- $Out^C : S \times X^C \times X^D \to Y^C$ is the continuous value output function
- $Out^D : P \times S^D \times X^D \to Y^D$ is the discrete value output function

(1) **Intervals $\langle t_1, t_2 \rangle$ with no events:** Only the continuous states S^C change. The continuous states at the end of the interval are computed from the state at the beginning plus the integral of the rate of change function $Rate(s(t),\ x^C(t),\ x^D(t))$ $(t = \langle t_1, t_2 \rangle)$ along the interval.

(2) **An internal state event occurs first at time t in interval $\langle t_1, t_2 \rangle$:** The continuous states at the time of the transition are computed from the state at the beginning plus the integral of the rate of change function $Rate(s(t'), x^C(t'), x^D(t'))(t' \leq t_1, t])$ along the interval until time t. Likewise, the hybrid output is generated until time t. At time t, the state transition condition function $Cond^S(s(t), x^C(t), x^D(t))$ evaluates to true. That is, an internal state event occurs. Here, the internal state transition function $Trans^S(s(t), x^C(t), x^D(t))$ is executed to define a new state.

(3) **An external discrete event occurs first at time t in interval $\langle t_1, t_2 \rangle$:** The continuous states at the time of the transition are computed from the state at the beginning plus the integral of $Rate(s(t'), x^C(t'), x^D(t'))(t' \leq t_1, t])$ along the interval until t. Likewise, the continuous output is generated until time t. At time t, the external event transition function $Trans^E(s(t), x(t))$ is executed to define a new state.

2.2 Hybrid Automata

Previous study[3] shows that DEV & DESS can be translated to linear hybrid automata[2] which are input-front end of HyTech. Similarly we translate from ECML model into hybrid automata which are input front-end of SpaceEx. SpaceEx includes PHAVer to provide various verification approaches. The definition of hybrid automata is as belows.

A hybrid system $H = (Loc, Var, Lab, Edg, Act, Inv)$ consists of six components.

- Loc is a finite set of vertex called locations.
- Var is a finite set of real-valued variables.
- Lab is a finite set of *synchronization labels* that contains the stutter label $\tau \in Lab$.
- Edg is a finite set of edge called transitions.
- Act is a labeling function that assigns to each location $l \in Loc$ a set of activities.
- Inv is a labeling function that assigns to each location $l \in Loc$ an *invariant* $Inv(l) \subseteq V$.

The PHAVer verifies hybrid automata that consists of linear dynamics or hybrid automata with affine dynamics.

- $Flow(l)$ is a continuous dynamics of the form $Ax + b_0 \bowtie 0$
- $Asgn$ is of the form $x' \bowtie Ax + b_0$

The SpaceEx consider hybrid automaton that restricts the constraints below.

- $Flow(l)$ is a continuous dynamics of the form $\dot{x}(t) = Ax(t) + Bu(t) + b_0,\ u(t) \in U$
- $Asgn$ is of the form $x' = Ax + Bu + b_0,\ u \in U$

A and B are constant sets, b_0 is constant value, U is nondeterministic input set which is constrained only by the invariant. x' is an assigned value. $\bowtie \in \{<, \leq, =\}$ is an operator.

3. Verification of Barrel-filler Model

We modeled a simple example that is ECML for a barrel-filler system. Previous research[3] shows a DEV & DESS for barrel-filler system and analyze it using HyTech. We modified it differently to suit for ECML. The models' behaviors and state trajectories are also explained in detail to confirm the correctness of the ECML models.

3.1 ECML model of Barrel-filler

We design a barrel-filler system for ECML. The system fills a barrel with a specific inflow rate and puts the barrel out whenever the barrel is filled up to a specific water level.

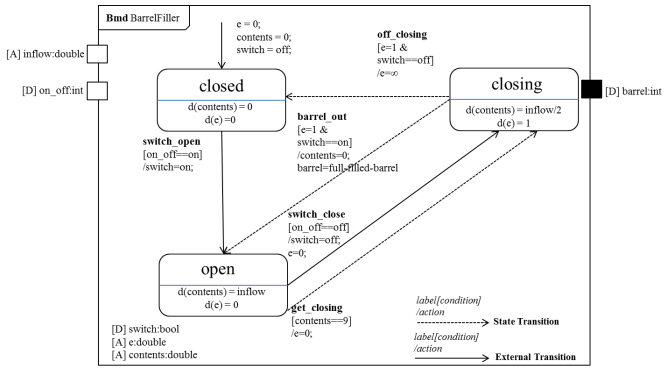

Fig. 1 A barrel-filler model for ECML

The barrel filler system modeled in this paper originated from [14]. **Fig. 1** is a graphical representation of the ECML model.

The model has two input ports and an output port. *on_off* is a discrete input that determines the start or stop of filling barrels, while *inflow* is a continuous input that is the rate of input water flow. *barrel* is a discrete output. The system puts *full-filled-barrel* out to the discrete output port *barrel* when the barrel is filled to a specific level.

This model has three phases such as *open*, *closed*, and *closing*. We added the closing state and timing constraint to the original barrel filler system in [14]. The modified barrel filler system fills the barrel with the inflow rate of $\dot{1}$ to $\dot{2}$. And if the contents is equated to 9, it transits to the *closing* state. It resets *e* for system idles in closing state during 1 time unit. In the closing state, the inflow rate is changed to $\dot{1}$, and it transit to the closed state or open state after 1 time unit. We combined with transitions those are from *closing* to *closed* and *closed* to *open* into *closing* to *open* from model of [3]. The model of [3] must be transits to *closed* after *closing*, but its behavior is same as our model. It also outputs a *full-filled-barrel* to the output port and resets *contents*.

Fig. 2 shows trajectory of a sample experiment of the modified barrel-filler system. Initially the valve is closed and the barrel is empty. At time 1, the filling is initiated by the discrete input signal *on*. At time 2, the input signal *off* stops the filling and the state of the valve changes from the *open* to *closing*. The inflow rate in the *closing* state is 1, half the rate of the *open* state, and the system resides there until time 3, since the *e* requires 1 time unit for closing valve. At time 5, the filling starts again by the *on* signal, and the *contents* increases at the rate of 2 until it reaches 9 at time 8. At time 8, the system goes into the *closing* state due to an internal event, and the *contents* increases at the rate of 1 for 1 time unit(until time 9). When the closing time expires, an internal event occurs. The *full-filled-barrel* is released. Contents are also reset to 0 and the filling restarts again immediately.

3.2 Verification of barrel-filler system

We translated from barrel-filler model for ECML into hybrid automata by hands. Some of translation rule from

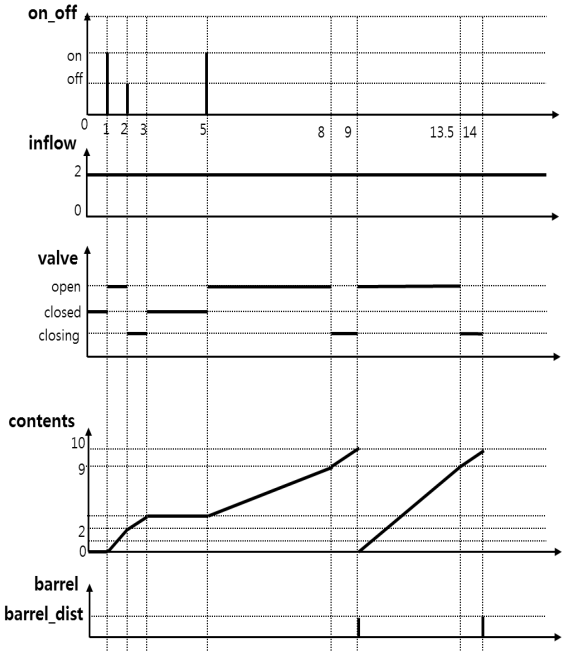

Fig. 2 A behavior of barrel-filler model

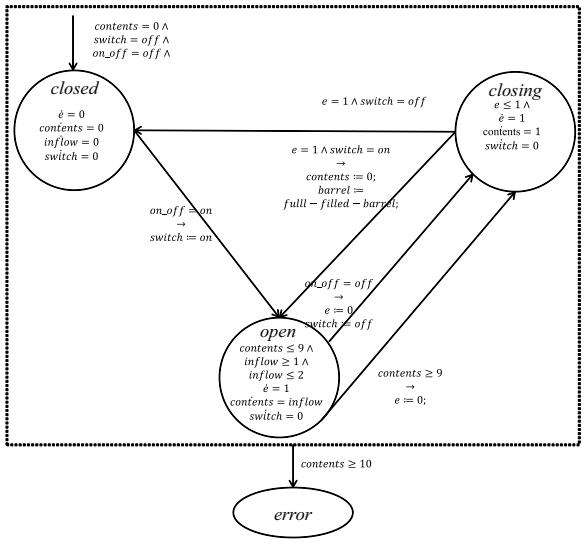

Fig. 3 A barrel-filler model for hybrid automata

ECML to linear hybrid automata proposed by [10]. **Fig. 3** shows barrel-filler model for linear hybrid automata. The barrel-filler automata consists of locations such as *closed*, *open*, and *closing* and transitions and variables such as *contents*, *e*, *switch*, and *inflow*. When the control mode is *closed*, value of variables doesn't changed during time advances except *switch*. A variable *switch* has nondeterministic value as flow of *switch* has not defined in this models. When *switch* changes on, the system changes control mode from *closed* to *open*. When the control mode is *open*, *contents* starts increased by *flow*, formally described $dcontents/dt = inflow$. The value of $inflow$ is non-deterministic and its range is $[1, 2]$. Contents is filling until *contents* is approached to 9. As soon as switch changes off or contents approaches to 9, control mode changes to closing. *contents* increase by half of inflow.

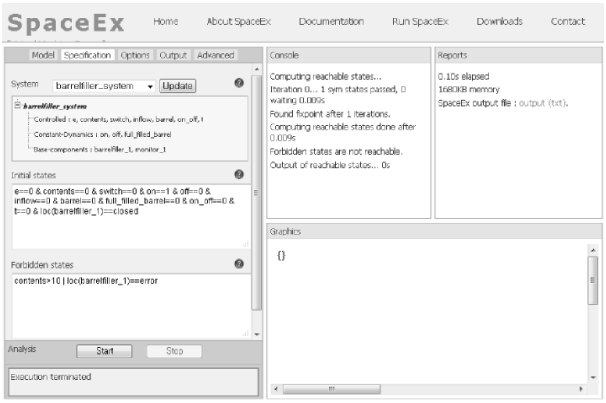

Fig. 4 Formal verification using SpaceEx

Barrel-filler system must restricts a safety property that is "*contents* must not exceeds 10 and system cannot be error state.". **Fig. 4** depicts the SpaceEx Web interfaces that shows performing safety verification of hybrid automata. We set the forbidden state as $contents > 10|loc(barrelfiller_1) = error$. *contents* is variable of barrel-filler system and must not exceeds 10. $barrelfiller_1$ is instance of linear hybrid automata for barrel-filler system. $loc(barrelfiller_1) = error$ means barrel-filler system control modes must not in error states. We set the scenario by PHAVer that verifies linear hybrid automata. The verification result is "Forbidden states are not reachable.". It means that barrel-filler system satisfies the safety requirements.

4. Conclusion

We presented a translating approach for formal verification of ECML models using hybrid automata as an example of barrel-filler system, which is input front-end of SpaceEx. ECML, a modeling language for cyber physical system, extension of DEV & DESS, models coupling between discrete and continuous elements. Hybrid automata also models combination of continuous and discrete dynamics. SpaceEx is reachability analysis tool for hybrid automata. Barrel-filler system is a suggested example for verification of ECML models. We model an ECML model for barrel-filler system and we translated from it into linear hybrid automata. We showed that ECML model can be translated into linear hybrid automata and verified using PHAVer in SpaceEx. An ECML model with non-linear dynamics has not been verified yet. We will try to verify ECML model as non-linear hybrid automata.

Acknowledgments This work was supported by the IT R&D Program of MKE/KEIT[12ND-1310, "The Development of CPS(Cyber-Physical Systems) Core Technologies for High Confidential Automatic Control Software"]. This research was supported by the MKE(The Ministry of Knowledge Economy), Korea, under the ITRC(Information Technology Research Center) support program supervised by the NIPA(National IT Industry Promotion Agency"(NIPA-2012-(H0301-12-3004))).

References

[1] Alur, R.: *Formal verification of hybrid systems*, Embedded Software (EMSOFT), 2011 Proceedings of the International Conference on, pp. 273–278 (2011).

[2] Alur, R. and Courcoubetis, C. and Halbwachs, N. and Henzinger, T.A. and Ho, P.H. and Nicollin, X. and Olivero, A. and Sifakis, J. and Yovine, S.: *The algorithmic analysis of hybrid systems* Theoretical computer science, Elsevier, Vol 138, No. 1, pp 3–34 (1995).

[3] Choi, H. and Cha, S. and Jo, J.Y. and Yoo, J. and Lee, H.Y. and Kim, W.T.: *Formal Verification of DEV&DESS Formalism Using Symbolic Model Checker HyTech*, Control and Automation, and Energy System Engineering, Springer, pp. 112–121 (2011).

[4] Frehse, G.: *PHAVer: Algorithmic verification of hybrid systems past HyTech*, Hybrid Systems: Computation and Control, Springer, pp. 258–273 (2005).

[5] Frehse, G. and Le Guernic, C. and Donzé, A. and Cotton, S. and Ray, R. and Lebeltel, O. and Ripado, R. and Girard, A. and Dang, T. and Maler, O.: *SpaceEx: Scalable Verification of Hybrid Systems*, Computer Aided Verification, Springer, pp. 379–395 (2011).

[6] Frehse, G. and Ray, R.: *Design principles for an extendable verification tool for hybrid systems*, Analysis and Design of Hybrid Systems (ADHS) (2009).

[7] Henzinger, T.A. and Ho, P.H. and Wong-Toi, H.: *HyTech: A model checker for hybrid systems*, International Journal on Software Tools for Technology Transfer (STTT), Springer, Vol. 1, No. 9, pp. 110–122 (1997).

[8] Henzinger, T.A. and Preussig, J. and Wong-Toi, H.: *Some lessons from the hytech experience*, Decision and Control, 2001. Proceedings of the 40th IEEE Conference on, IEEE, Vol. 3, pp. 2887–2892 (2001)

[9] Jeon, J. and Chun, I.G. and Kim, W.T.: *Metamodel-Based CPS Modeling Tool*, Embedded and Multimedia Computing Technology and Service, Springer, pp 285–291 (2012)

[10] Jo, J. and Yoo, J. and Choi, H. and Cha, S. and Lee, H.Y. and Kim, W.T.: *Translation from ECML to Linear Hybrid Automata*, Embedded and Multimedia Computing Technology and Service, Springer, pp. 293–300 (2012)

[11] Le Guernic, C. and Girard, A.: *Reachability analysis of hybrid systems using support functions*, Computer Aided Verification, Springer, pp. 540–554 (2009)

[12] Platzer, A. and Quesel, J.D.: *KeYmaera: A hybrid theorem prover for hybrid systems (system description)*, Automated Reasoning, Springer, pp. 171–178 (2008).

[13] Ratschan, S. and She, Z.: *HSolver: Verification of hybrid systems based on the constraint solver RSolver* (online), available from ⟨http://hsolver.sourceforge.net/⟩ (accessed 2012-07-17).

[14] Zeigler, B.P. and Praehofer, H. and Kim, T.G.: *Theory of modeling and simulation: Integrating discrete event and continuous complex dynamic systems* Academic Pr (2000).