A Correct by Construction Approach for the Modeling and the Verification of Cyber-Physical Systems in Event-B

Meryem Afendi^(✉)

Université Paris-Est Créteil, LACL, 94010 Creteil, France
meryem.afendi@u-pec.fr

Abstract. Cyber-Physical Systems (CPSs) connect the real world to software systems through a network of sensors and actuators: physical and discrete components interact in complex ways by involving different spatial and temporal scales. One of the most common architectures for CPSs is a discrete software controller which interacts with its physical environment in a closed-loop schema where input from sensors is processed and output is generated and communicated to actuators. We are concerned with the construction and verification of the correctness of such discrete controller using a correct by construction approach, which requires correct integration of discrete and continuous models.

1 Introduction

In CPSs [1], the measurement of continuous behaviors is performed by sensors. Ideally sensors have a continuous access to these measurements, which can be captured by an abstract model of CPSs, called *Event-Triggered* system by Kopetz in [2]. However, implementing such models is difficult in practice. Therefore, an approach to develop CPSs is to introduce a more realistic model called *Time-Triggered* system where sensors take periodic measurements [2]. Contrary to *Event-Triggered* models, properties on *Time-Triggered* models are difficult to verify since their controllers are discrete. Platzer et al. [3, 4] use this approach to model hybrid systems, which represent the most common mathematical models for CPSs. They have proved that a *Time-Triggered* model is a refinement of an *Event-Triggered* model, by using hybrid programs (HPs) [3] and an extension of the differential dynamic logic (d $\mathcal{L}$), called the differential refinement logic (dR $\mathcal{L}$). In addition to d $\mathcal{L}$ formulas, dR $\mathcal{L}$ introduces formulas of the form $\alpha \leq \beta$, α refines β , with α and β denoting HPs. However dR $\mathcal{L}$ is not supported by any prover and dR $\mathcal{L}$ formulas can only be manually proved, which heavily restricts its use, especially in an industrial context. This is why we propose the development of an approach that takes advantage of dR $\mathcal{L}$ and applies the reasoning of HPs in Event-B [5], a formal method based on set theory, first-order logic and predicate logic.

2 State of the Art

The development of techniques and tools to effectively design hybrid systems has drawn the attention of many researchers [6–10]. Traditional approaches are based on simulation tools like Matlab/Simulink [11]. Since these tools are time-consuming and produce results tainted with uncertainty, these approaches can be very expensive and difficult to apply. To overcome these limitations, several formal approaches, which can be grouped into two categories, have been proposed: *model-checking-based* approaches [6, 7] and *proof-based* approaches [3, 8–10], have been proposed. Proof-based approaches use deductive verification to prove the properties of hybrid systems. One of the strong points of these approaches is that they support the description of any kind of hybrid systems. However, they require significant effort and a high expertise in modelling and proof phases. A. Platzer introduced in [3] a first-order dynamic logic in the domain of real ($\mathbb{R}$), called $d\mathcal{L}$, to specify hybrid systems and verify their correctness using its associated proof calculus. The major advantage of this logic is its ability to handle differential equations, even those with non-polynomial solutions. Moreover, $d\mathcal{L}$ and its associated proof calculus are supported by two automatic formal verification tools, KeYmaera and its successor KeYmaera X [12]. In [10], Dupont et al. introduced a proof-based approach to model and prove hybrid systems in Event-B. They use the Theory plug-in of Event-B to define theories that handle continuous aspects of hybrid systems such as differential equations. The behavior of CPSs is specified by three Event-B models: *System* model that is used to describe the continuous evolution of the physical part, *State-System* model that refines the previous model by adding the evolution of the discrete part (the controller) and *Controlled-System* model that refines *State-System* model by adding the interaction between the physical part and the discrete part.

3 Proposed Approach

Our work is supported in part by the DISCONT project [13] funded by the French National Research Agency (ANR). The objective of this project is to elaborate a correct-by-construction method, based on Event-B, to specify hybrid systems models. Two approaches are considered. The first one, developed by Dupont et al. [10], is based on a translation of hybrid automata in Event-B. In our approach we propose to model the high-level structure of hybrid programs in Event-B, and more precisely the generic templates defined for modelling *Event* and *Time-Triggered* systems in $d\mathcal{R}$ to take advantages of its proof obligations. To handle continuous aspects of hybrid systems in Event-B, we use the abstract model *System* of Dupont et al. as a starting model, as well as their theories developed to handle differential equations and continuous functions. To validate our approach, we chose the *Stop Sign* case study [14] which deals with a stop sign controller whose objective is to ensure the stopping of a car before a stop signal *SP*. In this case study, the differential equation that represents the evolution of the physical part is linear and can be easily solved. To handle more difficult

differential equations we plan to integrate an external mathematical tool as a back-end tool in Event-B associated tools.

4 Current Results and Future Work

So far, our approach has focused on the most abstract level of hybrid systems design where all transitions are instantaneous, that is, duration issues are not considered. It consists of three models (see Fig. 1). *Event-Triggered* model (*EventTriggered_Ctx* and *EventTriggered_M* of Fig. 1) is a generic abstract model designed to specify and prove systems with continuous controllers in Event-B. It defines a formula *safe* used in dR $\mathcal{L}$ to represent the system's safety envelope. To model the alternation between the controller and the physical part as represented in HPs $((ctrl; plant)^*$, where *ctrl* denotes the discrete evolution, followed by the continuous evolution *plant*), *Event-Triggered* model defines a variable *exec* that can take two values *ctrl* and *plant*. In Event-B, the time must be explicitly handled. To be sure that this explicit time will only be updated after the execution of the controller, we added another value, *prg*, to *exec*. Therefore, our model follows the following structure: *init; (ctrl; plant; prg)**.

To model *Time-Triggered* systems in Event-B, we designed a new model, named *Time-Triggered* (*TimeTriggered_Ctx* and *TimeTriggered_M* of Fig. 1), that refines the previous one. The sensors of such systems take periodic measurements of physical state variables and the longest time between these sensors updates is bounded by a symbolic duration ϵ . For this purpose, *Time-Triggered* model defines a variable *d* to know whether the duration ϵ is reached or not. Since the controller of such systems must make a choice that will be safe for up to ϵ time, we replaced the formula *safe* by a new safety envelope named *safeEpsilon* that depends on both the current discrete state and the time duration ϵ , in addition to the current physical state. Finally, to prove that *TimeTriggered_M* refines *EventTriggered_M*, we establish that, during a control period ϵ , *safeEpsilon* implies *safe*. Moreover, we must guarantee that the continuous controller is able to execute exactly when *safe* is no longer satisfied. In addition, in the *Time-Triggered* model, we must also guarantee that the system will not exceed the domain of the safety envelope within time ϵ .

As future work, we plan to integrate Mathematica [15], a symbolic mathematical computation system that resolves differential equations, as a back-end tool in the Rodin platform to resolve differential equations. We also plan to define a refinement of the *Time-Triggered* model to take into account duration between the sending of continuous measurements by sensors and their processing by the controller as well as duration between the sending of actions by the

Fig. 1. Approach schema.

controller and their execution in order to get a model that corresponds better to real CPSs, and consequently implements this concrete model.

References

1. Lee, E.A.: Cyber physical systems: design challenges. In: 2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC), pp. 363–369. IEEE (2008)
2. Kopetz, H.: Event-triggered versus time-triggered real-time systems. In: Karshmer, A., Nehmer, J. (eds.) *Operating Systems of the 90s and Beyond*. LNCS, vol. 563, pp. 86–101. Springer, Heidelberg (1991). <https://doi.org/10.1007/BFb0024530>
3. Platzer, A.: Differential dynamic logic for hybrid systems. *J. Autom. Reason.* **41**(2), 143–189 (2008)
4. Loos, S.M., Platzer, A.: Differential refinement logic. In: 2016 31st Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), pp. 1–10. IEEE (2016)
5. Abrial, J.-R.: *Modeling in Event-B: System and Software Engineering*. Cambridge University Press, Cambridge (2010)
6. Henzinger, T.A., Ho, P.-H., Wong-Toi, H.: HyTECH: a model checker for hybrid systems. In: Grumberg, O. (ed.) *CAV 1997*. LNCS, vol. 1254, pp. 460–463. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-63166-6_48
7. Frehse, G.: PHAVer: algorithmic verification of hybrid systems past HyTech. In: Morari, M., Thiele, L. (eds.) *HSCC 2005*. LNCS, vol. 3414, pp. 258–273. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-31954-2_17
8. Butler, M., Maamria, I.: Practical theory extension in Event-B. In: Liu, Z., Woodcock, J., Zhu, H. (eds.) *Theories of Programming and Formal Methods*. LNCS, vol. 8051, pp. 67–81. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39698-4_5
9. Banach, R., Butler, M., Qin, S., Verma, N., Zhu, H.: Core hybrid Event-B I: single hybrid Event-B machines. *Sci. Comput. Program.* **105**, 92–123 (2015)
10. Dupont, G., Aït-Ameur, Y., Pantel, M., Singh, N.K.: Proof-based approach to hybrid systems development: dynamic logic and Event-B. In: Butler, M., Raschke, A., Hoang, T.S., Reichl, K. (eds.) *ABZ 2018*. LNCS, vol. 10817, pp. 155–170. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-91271-4_11
11. Sanfelice, R., Copp, D., Nanez, P.: A toolbox for simulation of hybrid systems in Matlab/Simulink: Hybrid equations (HyEQ) toolbox. In: *Proceedings of the 16th International Conference on Hybrid Systems: Computation and Control*, pp. 101–106 (2013)
12. Fulton, N., Mitsch, S., Quesel, J.-D., Völpl, M., Platzer, A.: KeYmaera X: an axiomatic tactical theorem prover for hybrid systems. In: Felty, A.P., Middeldorp, A. (eds.) *CADE 2015*. LNCS (LNAI), vol. 9195, pp. 527–538. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21401-6_36
13. ANR-17-CE25-0005: DISCONT ANR project (2017). <https://discont.loria.fr>
14. Quesel, J.-D., Mitsch, S., Loos, S., Aréchiga, N., Platzer, A.: How to model and prove hybrid systems with KeYmaera: a tutorial on safety. *Int. J. Softw. Tools Technol. Transfer* **18**(1), 67–91 (2016)
15. Wolfram, S.: *The Mathematica*, 5th edn. Wolfram Media, Champaign (2003)