Proceedings of the
1st Workshop on Artificial Intelligence and Formal Verification, Logics, Automata and Synthesis (OVERLAY),
Rende, Italy, November 19–20, 2019

Control Software Synthesis for Cyber-Physical Systems with QKS*

Vadim Alimguzhin[1], Federico Mari[2], and Igor Melatti[1]

[1]Sapienza University of Rome, Italy
[2]University of Rome Foro Italico, Italy

Abstract

The integration of artificial intelligence and formal methods has been proved to be a winning binomial for the analysis of cyber-physiscal systems. Many of such systems are indeed closed loop control systems where a software guides a plant through satisfying safety requirements. For safety- and mission-critical contexts, such requirements are strictly important, thus motivating research on methods for the automatic generation of correct control software. In this paper we review recent results obtained with QKS, a tool exploiting artificial intelligence and formal methods technologies in order to compute control software for cyber-physical systems. Namely, we sketch the QKS workflow and we show how QKS has been applied to interesting cases for control and automation.

1 Introduction

Cyber-Physical Systems (CPSs) have become increasingly important in safety- and mission-critical contexts such as, for example, smart grids, avionics, automotive, and Industry 4.0. Thanks to their practicality and adaptability, Artificial Intelligence (AI) techniques are gaining considerable attention also in the CPSs community. The application of AI to safety- and mission-critical systems motivates further research on automatic (formal) analysis methods for CPSs.

Many CPSs are indeed closed loop control systems (Figure 1) which consist of two main subsystems: the controller and the *plant* (i.e., the controlled system). The plant is a physical system composed, for example, of electrical or mechanical devices whereas the controller is a control software executed on a micro-controller. In an infinite loop, the controller reads the Analog-to-Digital (AD) conversion of plant sensors output and sends back the Digital-to-Analog (DA) conversion of commands to plant actuators so that the closed loop system is guaranteed to satisfy safety requirements. Many works in the literature study how formal methods may be combined with AI techniques in order to both verify correctness of CPSs (see, e.g., [6, 12, 14, 8, 11, 10, 13] and citations thereof) and synthesise automatic control software for CPSs (see, e.g., [2, 5, 15, 25, 9, 16] and citations thereof). In this paper, we focus on automatic control software synthesis for CPSs by integrating formal methods and AI techniques.

In this context, the CPS plant is modeled as a Discrete-Time Hybrid System (DTHS). The dynamics of a DTHS is defined as a boolean combination of (possibly nonlinear) constraints on integer and real

*This work was partially supported by INDAM, GNCS 2019, "Formal Methods for Mixed Verification Techniques".

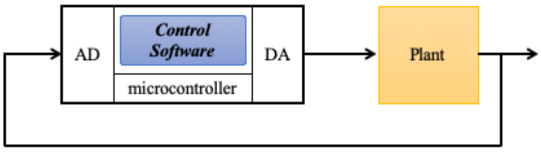


Figure 1: Closed loop control system

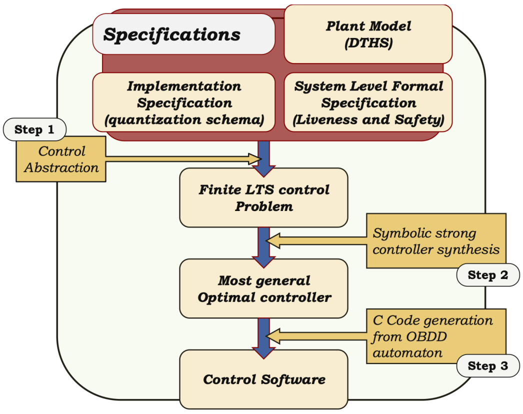

Specifications
Plant Model (DTHS)
Implementation Specification (quantization schema)
System Level Formal Specification (Liveness and Safety)
Step 1
Control Abstraction
Finite LTS control Problem
Symbolic strong controller synthesis
Step 2
Most general Optimal controller
C Code generation from OBDD automaton
Step 3
Control Software

Figure 2: QKS flow

variables. The passage from discrete- to continuous-time is left to the plant modeler (e.g., using Runge-Kutta methods). The quantised control synthesis problem for CPSs may be stated as follows. Given: (*i*) the DTHS model for the plant of a CPS $\mathcal{C}$; (*ii*) the number of bits used for AD and DA conversions (*quantisation schema*), describing how the control software may read values from the plant and send commands to the plant; (*iii*) formal specifications describing functional requirements of the closed loop system (*safety* and *liveness*); we want to automatically generate a correct-by-construction control software for $\mathcal{C}$ (if any) which satisfies the given functional requirements. Such a problem is undecidable [18].

In Section 2 we describe Quantized feedback Kontrol Synthesizer (QKS) [2, 5], an algorithm (along with a tool implementing it) to practically solve the quantised control synthesis problem for CPSs stated above. Given that the problem is undecidable, QKS computes a sufficient and a necessary condition for the solution of the control problem. To achieve such goal, QKS exploits AI and formal methods (in particular Model Checking) technologies. QKS has been effectively applied to several interesting use cases for control and automation, as for example the Buck DC/DC Converter (BDC) (Figure 3) and the Inverted Pendulum (IP) (Figure 4), as we show in Section 3.

2 Quantized feedback Kontrol Synthesizer (QKS)

In this section, we describe the QKS execution flow, which is sketched in Figure 2.

Step 1 Starting from the input DTHS, QKS generates a finite control abstraction. Such abstraction depends on the quantisation scheme, which defines the precision of the plant (state) sensors (AD conversion) and of the plant (action) actuators (DA conversion).

Step 2 Given a finite control abstraction G, representing *safety* and *liveness* requirements, QKS generates a controller K for the input DTHS. Starting from any initial state, K is able to drive G so that both safety and liveness requirements are met, notwithstanding non-deterministic behaviours of G.

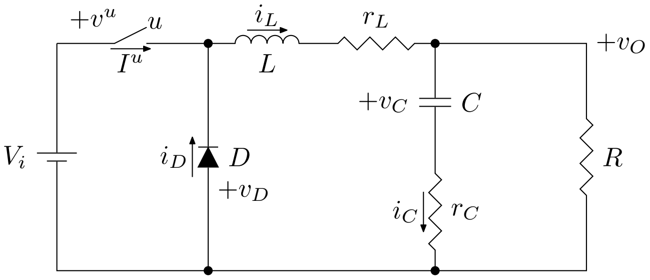


Figure 3: Buck DC-DC converter

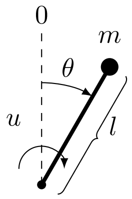
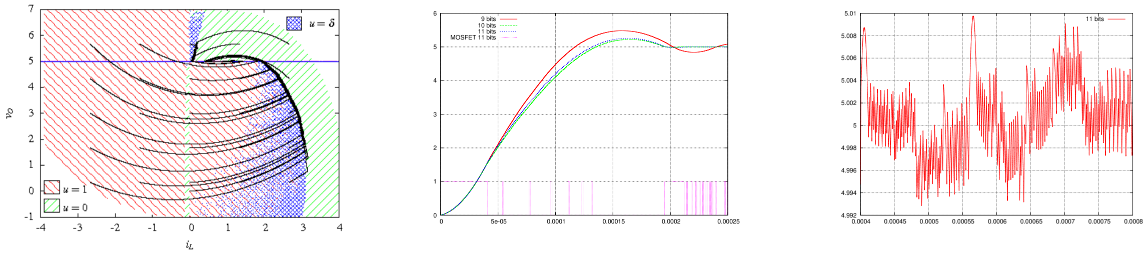


Figure 4: Inverted Pendulum (IP)



(a) Controlled region and trajectories (b) Setup time (c) Ripple for best controller

Figure 5: BDC: Controlled region and performance

Step 3 QKS translates K in an actual control software in the C language.

In order to generate controller K at step 2, QKS employs a symbolic algorithm based on Ordered Binary Decision Diagrams (OBDDs) [19]. Such an algorithm outputs a time-optimal solution, thus K is able to drive the closed-loop system G always following minimal paths. Note that, given that K is represented by an OBDD with known depth, the control software resulting from K yields a Worst-Case Execution Time (WCET) which is linear in the number of bits in the quantisation schema (non-functional requirements).

Our approach is on the edge between formal methods and AI On the one hand, our setting and theoretical background are typically used in formal methods. Mainly, in [19] we define the formalization to prove that a controller for the abstract finite-state control problem can be suitably transformed into a controller for the concrete discrete-time control problem. On the other hand, steps of Figure 2 exploit many different AI techniques. Namely, explicit search-based methods and heuristics, Constraint Satisfaction Problems (CSPs) solved through Mixed Integer Linear Programming (MILP) solvers, and symbolic search-based methods on decision diagrams (OBDD).

QKS variants Besides the sequential version for linear DTHSs [17, 18, 19] and non-linear DTHSs [1, 5], QKS also has the following enhancements.

Compact control software. A method which trades-off temporal optimality with control software compactness is presented in [2]. To this aim, such method looks for maximal regions (step 3, Figure 2), which may be controlled executing the same action.

Parallel generation of control abstraction. In [3] a parallel version of QKS is presented, based on the *Map-Reduce* workflow (step 1, Figure 2).

On-the-fly generation of control abstraction. In [4] an *on-the-fly* version of QKS is presented, which allows faster answers for problems which do not admit solutions (steps 1 and 2, Figure 2).

3 Results

In this section we show some results obtained with QKS, in particular on the Buck DC/DC Converter (BDC) of Figure 3 and on the Inverted Pendulum (IP) of Figure 4.

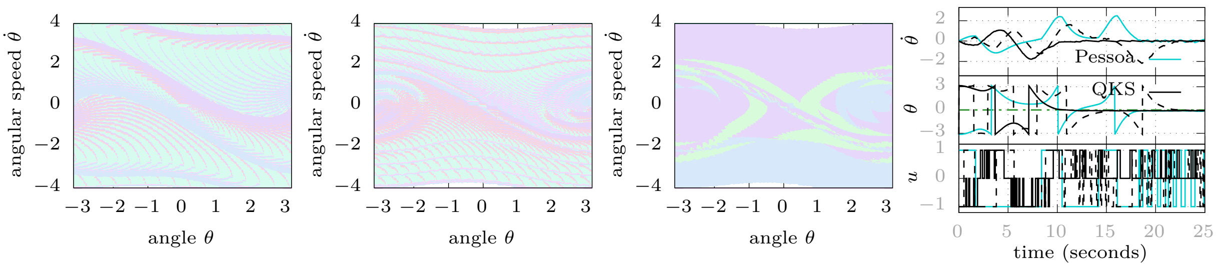
Buck DC/DC Converter (BDC) The BDC, depicted in Figure 3, is a mixed-mode analog circuit converting the DC input voltage (V_i in Figure 3) to a desired DC output voltage (v_O in Figure 3). As an example, buck DC-DC converters are used off-chip to scale down the typical laptop battery voltage (12-24) to the just few volts needed by the laptop processor (e.g., see [22]) as well as on-chip to support Dynamic Voltage and Frequency Scaling (DVFS) in multicore processors (e.g., see [21]). Because of its widespread use, control schemas for buck DC-DC converters have been widely studied (e.g., see [26]). The typical software based approach (e.g., see [22]) is to control the switch u in Figure 3 (typically implemented with a MOSFET) with a microcontroller. In our context, we use QKS to automatically synthesise the control software for such a microcontroller, with the goal to keep the output voltage v_O close enough to a given reference value (liveness) when both the power supply V_i and the load R may vary up to 25%.

Figure 5 shows QKS controllers performance for BDC. Figure 5a shows controlled region and some trajectories of the closed loop system (with time increasing counter-clockwise) using quantization scheme with 10 bits. Different colors correspond to different sets of actions u allowed by the controller. As a result, all states in the desired controllable region $I \equiv (|i_L| \leq 2) \wedge (0 \leq v_O \leq 6.5)$ are in the actually controlled region. Figure 5b shows v_O trajectories using quantization schemes with number of bits $b \in \{9, 10, 11\}$ and the control signal (square wave) for $b = 11$. All trajectories stabilize after only 0.0003 secs (setup time). Figure 5c shows ripple of v_O after stabilization using quantization scheme with $b = 11$ bits. Such ripple turns out to be about 0.01 V, that is 0.2% of the reference value $V_{ref} = 5$ V. Both setup time and ripple compare well with typical figures of commercial high-end buck DC-DC converters [24] and with the results available from the literature [23, 26].

Inverted Pendulum (IP) An IP [7], shown in Figure 4, is modeled by taking the angle θ and the angular velocity $\dot{\theta}$ as state variables. The input of the system is the torquing force u, that can influence the velocity in both directions. In our setting, we use QKS to compute the following for the torque actuator: “apply the force clockwise” ($u = 1$), “apply the force counterclockwise” ($u = -1$), or “do nothing” ($u = 0$). The intensity of the force will be given as a constant F.

Figure 6 shows comparison between QKS and state-of-the-art method and tool for control software synthesis for switched nonlinear systems Pessoa [20], on the IP model. Results are reported with quantization scheme 9 bits, where *liveness* & *safety* properties are to reach and hold the pendulum upward equilibrium. In this setting, in order to build the control abstraction and to generate the controller, QKS takes 1h:30m while Pessoa 2h. Figure 6a shows controlled region for controller generated with Pessoa. Also in this setting, different colors correspond to different sets of actions u allowed by the controller. Controller OBDD has 19,994 nodes. Figure 6b shows controlled region for controller generated with QKS. Note that there is a different control software strategy w.r.t. Figure 6b. In this case, the controller OBDD is bigger w.r.t. the one generated by Pessoa, namely it has 21,629 nodes. Figure 6c shows controlled region for controller generated with the compact version of QKS. On the one hand, allowed actions are more homogeneous and minimal w.r.t. the non-compact case. On the other hand, controller size is smaller, namely the OBDD has 5,031 nodes. Figure 6d shows a closed loop system simulation with the generated control softwares, starting from the unstable lower equilibrium point of the pundulum. Compact QKS is dashed.

As for generated control software performance (in this setting), the following considerations hold: (*i*) Compact QKS controller has a dimension which is the 23% of non-compact one (Figure 6c vs Figure 6b) but has a worse set-up time (23 seconds the compact one, dashed, vs 10 seconds the non-compact one, Figure 6d). (*ii*) Non-compact QKS controller stabilizes the pendulum faster than Pessoa controller (10 vs 18 seconds, Figure 6d) but has a higher switching change speed.



(a) Pessoa (b) QKS (c) Compact QKS (d) Setup time

Figure 6: IP: Controlled regions (Figures 6a to 6c): each color is a combination of actions allowed by the controller. Performance (Figure 6d): Pessoa is cyan, QKS is black whereas compact QKS is dashed black.

4 Conclusions and Future Work

In this paper we reviewed the main results concerning the quantised control software synthesis for CPSs using QKS. Results show that QKS compares well with state-of-the-art algorithms and tools.

There are many future research directions in this area. One of the most interesting is usage of AI and formal methods techniques in order to extract models for complex CPSs from simulators and execution logs.

References

[1] V. Alimguzhin, F. Mari, I. Melatti, I. Salvo, and E. Tronci. Automatic control software synthesis for quantized discrete time hybrid systems. In *Proceedings of 51th IEEE Conference on Decision and Control (CDC 2012)*, pages 6120–6125. IEEE, 2012.

[2] V. Alimguzhin, F. Mari, I. Melatti, I. Salvo, and E. Tronci. On model based synthesis of embedded control software. In *Proceedings of 12th International Conference on Embedded Software (EMSOFT 2012)*, pages 227–236. ACM, 2012.

[3] V. Alimguzhin, F. Mari, I. Melatti, I. Salvo, and E. Tronci. A map-reduce parallel approach to automatic synthesis of control software. In *Proceedings of 20th International SPIN Symposium on Model Checking of Software (SPIN 2013)*, volume 7976 of *Lecture Notes in Computer Science*, pages 43–60. Springer, 2013.

[4] V. Alimguzhin, F. Mari, I. Melatti, I. Salvo, and E. Tronci. On-the-fly control software synthesis. In *Proceedings of 20th International SPIN Symposium on Model Checking of Software (SPIN 2013)*, volume 7976 of *Lecture Notes in Computer Science*, pages 61–80. Springer, 2013.

[5] V. Alimguzhin, F. Mari, I. Melatti, I. Salvo, and E. Tronci. Linearizing discrete-time hybrid systems. *IEEE Transactions on Automatic Control*, 62(10):5357–5364, 2017.

[6] A. Bobbio, E. Ciancamerla, S. Di Blasi, A. Iacomini, F. Mari, I. Melatti, M. Minichino, A. Scarlatti, E. Tronci, R. Terruggia, and E. Zendri. Risk analysis via heterogeneous models of scada interconnecting power grids and telco networks. pages 90–97, 2009.

[7] G. Kreisselmeier and T. Birkhölzer. Numerical nonlinear regulator design. *IEEE Trans. on on Automatic Control*, 39(1):33–46, 1994.

[8] T. Mancini, F. Mari, A. Massini, I. Melatti, F. Merli, and E. Tronci. System level formal verification via model checking driven simulation. In *Proceedings of 25th International Conference on Computer Aided Verification (CAV 2013)*, volume 8044 of *Lecture Notes in Computer Science*, pages 296–312. Springer, 2013.

[9] T. Mancini, F. Mari, A. Massini, I. Melatti, I. Salvo, S. Sinisi, E. Tronci, R. Ehrig, S. Röblitz, and B. Leeners. Computing personalised treatments through in silico clinical trials. A case study on downregulation in assisted reproduction. In *Proceedings of 25th RCRA International Workshop on Experimental Evaluation of Algorithms for Solving Problems with Combinatorial Explosion (RCRA 2018)*, 2018.

[10] T. Mancini, F. Mari, A. Massini, I. Melatti, I. Salvo, and E. Tronci. On minimising the maximum expected verification time. *Information Processing Letters*, 122:8–16, 2017.

[11] T. Mancini, F. Mari, A. Massini, I. Melatti, and E. Tronci. Anytime system level verification via parallel random exhaustive hardware in the loop simulation. *Microprocessors and Microsystems*, 41:12–28, 2016.

[12] T. Mancini, F. Mari, A. Massini, I. Melatti, and E. Tronci. SyLVaaS: System level formal verification as a service. *Fundamenta Informaticae*, 1–2:101–132, 2016.

[13] T. Mancini, F. Mari, I. Melatti, I. Salvo, and E. Tronci. An efficient algorithm for network vulnerability analysis under malicious attacks. In *Proceedings of The 24th International Symposium on Methodologies for Intelligent Systems (ISMIS 2018)*. Springer, 2018.

[14] T. Mancini, F. Mari, I. Melatti, I. Salvo, E. Tronci, J. Gruber, B. Hayes, and L. Elmegaard. Parallel statistical model checking for safety verification in smart grids. In *Proceedings of 2018 IEEE International Conference on Smart Grid Communications (SmartGridComm 2018)*. IEEE, 2018.

[15] T. Mancini, E. Tronci, I. Salvo, F. Mari, A. Massini, and I. Melatti. Computing biological model parameters by parallel statistical model checking. In *Proceedings of 3rd International Conference on Bioinformatics and Biomedical Engineering (IWBBIO 2015)*, volume 9044 of *Lecture Notes in Computer Science*, pages 542–554. Springer, 2015.

[16] T. Mancini, E. Tronci, A. Scialanca, F. Lanciotti, A. Finzi, R. Guarneri, and S. Di Pompeo. Optimal fault-tolerant placement of relay nodes in a mission critical wireless network. In *Proceedings of 25th RCRA International Workshop on Experimental Evaluation of Algorithms for Solving Problems with Combinatorial Explosion (RCRA 2018)*, 2018.

[17] F. Mari, I. Melatti, I. Salvo, and E. Tronci. Synthesis of quantized feedback control software for discrete time linear hybrid systems. In *Proceedings of 22nd International Conference on Computer Aided Verification (CAV 2010)*, volume 6174 of *Lecture Notes in Computer Science*, pages 180–195. Springer, 2010.

[18] F. Mari, I. Melatti, I. Salvo, and E. Tronci. Undecidability of quantized state feedback control for discrete time linear hybrid systems. In *Proceedings of 9th International Colloquium on Theoretical Aspects of Computing (ICTAC 2012)*, volume 7521 of *Lecture Notes in Computer Science*, pages 243–258. Springer, 2012.

[19] F. Mari, I. Melatti, I. Salvo, and E. Tronci. Model based synthesis of control software from system level formal specifications. *ACM Transactions on Software Engineering and Methodology*, 23(1):1–42, 2014.

[20] M. Mazo, A. Davitian, and P. Tabuada. PESSOA: A tool for embedded controller synthesis. In *Proceedings of 22nd International Conference on Computer Aided Verification (CAV 2010)*, volume 6174 of *Lecture Notes in Computer Science*, pages 566–569. Springer, 2010.

[21] G. Schrom, P. Hazucha, J. Hahn, D. Gardner, B. Bloechel, G. Dermer, S. Narendra, T. Karnik, and V. De. A 480-mhz, multi-phase interleaved buck dc-dc converter with hysteretic control. In *PESC*, pages 4702–4707 vol. 6. IEEE, 2004.

[22] W.-C. So, C. Tse, and Y.-S. Lee. Development of a fuzzy logic controller for dc/dc converters: design, computer simulation, and experimental evaluation. *IEEE Trans. on Power Electronics*, 11(1):24–32, 1996.

[23] W.-C. So, C. K. Tse, and Y.-S. Lee. Development of a fuzzy logic controller for dc/dc converters : Design, computer simulation and experimental evaluation. *IEEE Trans. on Power Electronics*, 11(1):23–32, 1996.

[24] Slvp182: High accuracy synchronous buck dc-dc converter: `http://focus.ti.com.cn/cn/lit/ug/slvu046/slvu046.pdf`, 2001.

[25] E. Tronci, T. Mancini, I. Salvo, S. Sinisi, F. Mari, I. Melatti, A. Massini, F. Davi', T. Dierkes, R. Ehrig, S. Röblitz, B. Leeners, T. H. C. Krüger, M. Egli, and F. Ille. Patient-specific models from inter-patient biological models and clinical records. In *Proceedings of 14th International Conference on Formal Methods in Computer-Aided Design (FMCAD 2014)*, pages 207–214. IEEE, 2014.

[26] V. Yousefzadeh, A. Babazadeh, B. Ramachandran, E. Alarcon, L. Pao, and D. Maksimovic. Proximate time-optimal digital control for synchronous buck dc–dc converters. *IEEE Trans. on Power Electronics*, 23(4):2018–2026, 2008.