České vysoké učení technické v Praze
Fakulta informačních technology

Czech Technical University in Prague
Faculty of Information Technology

Dipl.-Ing. Dr.techn. Stefan Ratschan

Foundations of the Automatic Analysis of Cyber-Physical Systems

Summary

We address foundational questions of the automatic analysis of cyber-physical systems. Here we understand cyber-physical systems as systems that tightly integrate computation with physical processes. As a basis, we use the formalism of a hybrid dynamical system. We discuss the role of formal verification in the design of cyber-physical systems, and present a way of circumventing the fact that most formal verification problems for hybrid systems are undecidable.

Souhrn

Zabýváme se základními otázkami automatické analýzy takzvaných kyber-fyzikálních systémů. Zde chápeme kyber-fyzikální systém jako systém, který úzce integruje výpočty s fyzikálními procesy. Jako základ používáme formalizmus hybridního dynamického systému. Zvažujeme roli formální verifikace v návrhu kyber-fyzikálních systémů a prezentujeme způsob jak obejít problém, že většina úloh formální verifikace pro hybridní systémy je nerozhodnutelná.

Klíčová slova:

formální verifikace, kyber-fyzikální systémy, hybridní dynamické systémy, řešení omezujících podmínek

Keywords:

formal verification, cyber-physical systems, hybrid dynamical systems, constraint solving

Contents

1 Introduction

In Section 2 we will present the notion of a cyber-physical system, in Section 3 we will introduce hybrid dynamical systems as a formal model for cyber-physical systems, in Section 4 we will discuss the role of formal verification in the design of cyber-physical systems, in Section 5 we will present a way of circumventing the fact that most formal verification problems for hybrid systems are undecidable, and in Section 6 we will conclude the discussion.

2 Cyber-Physical Systems

In this section, we review the notion of a cyber-physical system and the role of models in the design of such systems. We present a specific class of models, hybrid dynamical systems, that will serve as a basis for the questions addressed in this document and discuss the automatic analysis of hybrid dynamical systems.

Citing Edward A. Lee [34, page 1], "a cyber-physical system (CPS) is an integration of computation with physical processes." [1] Among others, Lee states the following examples: information technological support for heart surgery, a non-centralized traffic control system based on inter-car interaction, electronic pilot-support systems in airplanes, and multiple autonomous quadrotors cooperating on a common task.

In current industrial practice, systems involving both computation and physical processes are more and more designed based on models, that is, a formal description of the given system that is abstract in the sense that it describes only those aspects of the system that are important in the current stage of the design process and ignores other aspects. To this end, certain modeling languages and tools are used (e.g., Matlab/Simulink, Modelica) that partially automize the design process using methods for the automatic simulation and analysis of the built models. The current research area of model based design [44, 29, 13] tries to advance the usage of models in systems design.

Modeling languages used in industrial practice are often very rich, allowing a broad range of expressive features and containing extensive libraries of modeling components. The downside of this expressivity is

[1] The notion of a cyber-physical system is often also used in a broader sense, or even as a keyword summarizing a certain vision and resulting research agenda (NSF, EU). Instead of this, in this document we will restrict ourselves to the simple technical definition given above.

the complexity of those formalisms and the lack of a precise formal semantics [8]. Since this document concentrates on foundational aspects, where a clean and elegant mathematical formalization is essential, we use a small and clean formalism instead of such a rich modeling language. Here we use the notion of a hybrid dynamical system that we will introduce in the next section.

3 Hybrid Dynamical Systems

The Turing Award winner Joseph Sifakis cites as one of the three grand challenges for rigorous system design [53]: "We need theory and models encompassing continuous and discrete dynamics to predict the global behavior of a [computational] system interacting with its physical environment." [2] The classical way of modeling computation are discrete formalisms such as finite automata. The classical way of modeling physical systems is based on continuous mathematics, especially differential equations. Hence, a natural way of modeling the integration of computation with physical processes is based on formalisms that integrate differential equations with finite automata. This gives rise to the notion of a hybrid dynamical system, often just called hybrid system, that combines continuous with discrete state and evolution. This ignores several aspects of cyber-physical systems (e.g., components [46], physical distribution [21, 15], process scheduling [35], network communication [40], and aspects resulting from specific application areas) but serves its role as a clean basis for studying foundational questions arising from the interaction between physical and computational components.

In the literature, definitions of the notion of hybrid system come in several flavors, that we roughly classify as follows:

- automata based [3, 51, 31] (very often called "hybrid automaton")
- set/function based [22, 54]
- constraint/logic based [45]

In this document we mainly use the set and function based approach, sometimes using constraints/logical formulas to describes those sets. More specifically, we use variants of the following definition:

[2] The word in brackets was added by the author of this document for clarification purposes.

Definition 1 *A* hybrid system *is a tuple* $(M, \Phi, \text{Init}, \text{Flow}, \text{Jump})$ *where*

- M *is a finite set, whose elements are usually called* modes*,*
- $\Phi \subseteq M \times \mathbb{R}^n$*,*
- $\text{Init} \subseteq \Phi$*,*
- $\text{Flow} \subseteq \Phi \times \mathbb{R}^n$*,*
- $\text{Jump} \subseteq \Phi \times \Phi$*.*

The intuition is that Φ describes the state space of the system, with discrete part M and continuous part $\mathbb{R}^n$. Init describes the set of initial states, Flow describes the continuous behavior of the hybrid system, and Jump describes the discrete behavior of the hybrid system.

Before making precise, how those elements determine the dynamical behavior of a hybrid system, we illustrate this definition based on the widely used example of a thermostat controlling the heating of a room. Here, the elements of the tuple $(M, \Phi, \text{Init}, \text{Flow}, \text{Jump})$ are as follows:

- $M = \{on, off\}$
  (the heating can either be on or off)
- $\Phi = M \times \mathbb{R}$, i.e. $n = 1$
  (we model the continuous behavior using one real variable, the temperature in the room)
- $\text{Init} = \Big\{(m, x) \mid 10 \leq x \wedge x \leq 25\Big\}$
  (initially, the temperature is between 10 and 25 degrees, independently of whether the heating is on or off)
- $\text{Flow} = \left\{ (m, x, \dot{x}) \mid \begin{array}{c} [m = on \Rightarrow [x \leq 21 \wedge \dot{x} = 30 - x]] \\ \wedge \\ [m = off \Rightarrow [x \geq 19 \wedge \dot{x} = -x]] \end{array} \right\}$
  (evolution of the temperature follows two differential equations, depending on whether the heating is on or off; in addition, continuous evolution is only possible if the temperature is not higher than 21 when the heating is on, or less then 19, when the heating is off, respectively)
- $\text{Jump} = \left\{ (m, x, m', x') \mid \begin{array}{c} [m = on \wedge x \geq 21] \Rightarrow [m' = off \wedge x' = x] \\ \vee \\ [m = off \wedge x \leq 19] \Rightarrow [m' = on \wedge x' = x] \end{array} \right\}$
  (the heating switches off if the temperature is at least 21 degrees and on if the temperature is 19 degrees or less)

Often such hybrid systems are visualized using graphs such as the one in Figure 1.

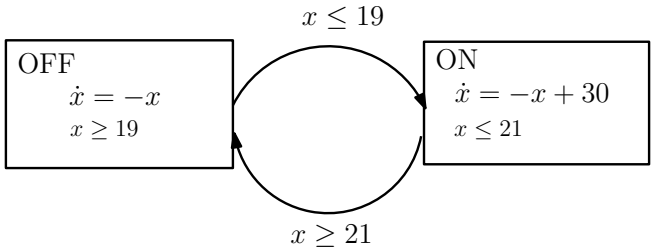

Figure 1: Visualization of Heating Example

The sets $\Phi, \text{Init}, \text{Flow}, \text{Jump}$ are usually infinite. So, we cannot directly represent them on computers using their elements. Instead, we work with a finite, symbolic representation of these sets. The constraints used for describing those sets in the above example (e.g., $10 \leq x \wedge x \leq 25$), form such a representation. In the document we will often call the result a *hybrid system description* which then falls into the class of constraint/logic based descriptions of hybrid systems mentioned above.

Widespread special cases of hybrid systems include switched systems [37, 36], timed automata [2], rectangular automata [27, 28], differential equations with dis-continuous right-hand side [17], and differential inclusions [4].

One can define the evolution of a hybrid system as follows:

Definition 2 *Given a hybrid system* $H = (M, \Phi, \text{Init}, \text{Flow}, \text{Jump})$*, a* flow *of length* l *of* H *is a pair* (m, φ) *where* $m \in M$ *and* $\varphi : [0, l] \to \mathbb{R}^n$*. By abuse of notation, for a flow* $\phi = (m, \varphi)$ *of length* l*, and* $t \in [0, l]$*, we denote by* $\phi(t)$ *the pair* $(m, \varphi(t))$*. A* trajectory *of* H *is a sequence of flows* $\phi_0, \ldots, \phi_p$ *of lengths* $l_0, \ldots, l_p$ *such that*

- $r_0(0) \in \text{Init}$*,*
- *if* $i > 0$ *then* $(r_{i-1}(l_{i-1}), r_i(0)) \in \text{Jump}$*,*
- *if* $l_i > 0$ *then for all* $t \in [0, l_i]$*,* $(r(t), \dot{r}(t)) \in \text{Flow}$*.*

For example, Figure 2 illustrates a trajectory of the thermostat example given above, where each element of the sequence of flows is drawn into one and the same diagram. Note also, that each flow has

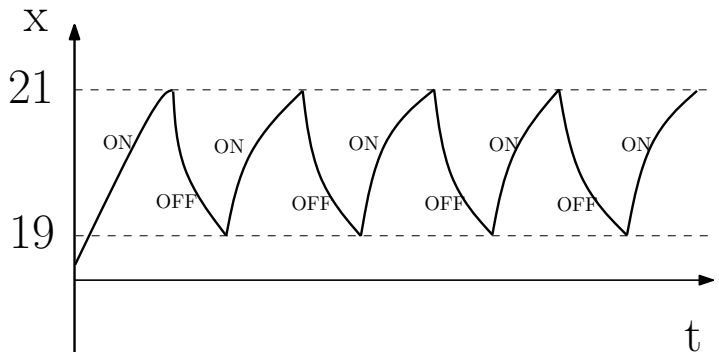

Figure 2: Example Trajectory

a local notion of time starting from zero that the diagram joins into a global time axis.

The above definitions are powerful enough to allow the modeling of common features of hybrid systems such as invariants or forced/non-forced jumps. The fact that Flow is based on a relation instead of a function $\Phi \to \mathbb{R}^n$ allows us to define non-deterministic hybrid systems, that is, hybrid systems for which a given initial point does not determine a unique trajectory starting from that point, but that allows many different trajectories to start in a given initial point. This permits the modeling of uncertainty or leaving open behavior that one does not want to specify in more detail at a given point of time in the design process. Constraints defining this relation Flow can model such non-deterministic behavior using differential inequalities, instead of differential equations. For example, the constraint $1 \leq \dot{x} \wedge \dot{x} \leq 2$, restricts the derivative to the interval $[1, 2]$, but allows any derivative in this interval.

Note that we use the term "system" in "hybrid system" in an essentially different way from its usage in "cyber-physical system": A hybrid system is an abstract mathematical formalism whereas a cyber-physical system is a concrete object from reality. Especially, it is not necessarily the case that the continuous part of a hybrid system models the physical and the discrete part models the computational part of a cyber-physical system. This is just a design choice that often turns out to be convenient. Still, there are many exceptions. For example, the gears of a motor or physical impact (bouncing ball) are often modeled discretely. For details, we refer to several textbooks [38, 45, 51, 54, 22] and surveys [9, 47, 24, 25, 52, 31] in the area.

4 System Design, Analysis, and Verification

When designing cyber-physical systems, two activities play a major role:

- Modeling: creating a model of the desired system.
- Analysis: analyzing the behavior of the created model.

These two activities are repeatedly iterated. One of the advantages of the usage of models is the fact that already early in the design process models are available whose analysis can be supported by computer tools. The most widely used tool for analysis is simulation, that is, the computation of (approximations of) possible behavior of the model. For hybrid systems, this means the computation of (approximations of) possible trajectories. Today, the simulation of hybrid systems is mature technology [43, 16, 1].

The research area of model checking [11, 5, 23] tries to automatize the analysis of models at a higher level: Instead of computing a single behavior of the model, it tries to design algorithms that formally verify a given model in the following sense:

- Given: a model $\mathcal{M}$ and formal specification S.
- Either prove that *all* behaviors of the model fulfills the formal specification, or compute some example behavior (usually called counter-example) that shows that the model does not fulfill its specification.

Up to recent years, the model $\mathcal{M}$ was usually required to have finitely many states. Only recently, there has been major progress in the model checking of systems with infinitely many states, allowing the modeling and model checking of data structures [30, 42], time [6], probability [32], and continuous/hybrid dynamics [26, 50, 20, 7, 14].

There are various ways of providing the formal specification S. In this document we will mostly concentrate on safety verification, where the specification S requires that the model $\mathcal{M}$ always stays in a given set of states that we consider to be safe, that is, it never reaches a state that we consider to be not safe. In the case of hybrid systems, this can be formalized by extending the definition of a hybrid system with a set of unsafe states $\text{Unsafe} \subseteq \Phi$ and calling a trajectory $\phi_0, \ldots, \phi_p$ of lengths $l_0, \ldots, l_p$ an *error trajectory* if and only if there is an $i \in \{0, \ldots, p\}$ and

$t \in [0, l_i]$ such that $\phi_i(t) \in$ Unsafe. Then, for a given hybrid system H, we either prove that H is safe, that is, that it does not have an error trajectory, or compute a counter-example in the form of an error trajectory of H.

A major question in the formal safety verification of hybrid system is, how to analyze their continuous behavior. A widespread technique to do so, is to reduce the analysis of this continuous behavior, including the analysis of ordinary differential equations, to a constraint solving problem in a first-order theory of real numbers. We also follow this approach in this document, classifying the contributions of this document into formal verification and constraint solving.

5 From Undecidability to Quasi-decidability

Most formal verification tasks for hybrid systems are undecidable [27]. Positive decidability results exist only for very special cases [33, e.g.,]. Algorithms for more general cases are usually based on their ability to verify a wide class of benchmark problems efficiently. We showed [49, 12] that this problem of undecidability can be circumvented by providing a procedure for safety verification that may run forever, but that terminates successfully in all cases that are in a certain sense practically relevant. More specifically, such a procedure, that we call a *quasi-decision procedure*, terminates in all cases that are robust, that is, in all cases where small changes to the hybrid system do not change its safety. The fact that engineers usually design systems to be robust implies practical relevance of this class.

Algorithms for formal verification are often built upon constraint solvers, in the case of hybrid systems, for the real numbers. Due to a classical result by A. Tarski [55], the first-order predicate logical theory of real-closed fields, that formalizes the real numbers with addition and multiplication, allows quantifier elimination, and hence is decidable. However, adding a function symbol representing the sine function, makes the theory undecidable, since the sine function is periodic, and hence it allows encoding the theory of integers that is undecidable [41]. Still, in a similar way as described above for the safety verification of hybrid systems [49, 12], it is possible to find a quasi-decision procedure, that is, a procedure takes as input formulas with such function symbols, returns a correct result, if it terminates, and terminates in all cases, when the input formula is in a certain, precisely defined sense, robust.

Here, a major issue is the question of how to handle equalities. One

approach [48] is to handle equalities of the form $f = 0$ as a short-cut for two inequalities of the form $-f \leq 0 \wedge f \leq 0$. This may destroy robustness, since a solution of $f = 0$ may vanish if the two occurrences of f in $-f \leq 0 \wedge f \leq 0$ are slightly changed. Hence the algorithms do not provide any termination guarantees when proving satisfiability of equalities. In many applications this is not a problem, since either this case does not occur, or relaxation of equalities to $-f \leq \varepsilon \wedge f \leq \varepsilon$, for a small positive real number ε, is fine. Still, we also showed how to handle equalities directly, without rewriting to inequalities, for a certain class of formulas [19, 18], again terminating in all robust cases.

6 Conclusion

Computation devices are becoming deeper and deeper integrated into the physical world surrounding us. Hence it is essential to be able to design correctly functioning cyber-physical systems, that is, systems that integrate computation and physical processes. The essence of this integration can be captured by the notion of a hybrid dynamical system, that is, dynamical systems that combine discrete and continuous state and evolution. However, the formal safety verification of such systems is undecidable for almost all cases. This problem can be circumvented by taking into account system robustness, resulting in the notion of a quasi-decision procedure.

References

[1] Vincent Acary and Bernard Brogliato. *Numerical Methods for Nonsmooth Dynamical Systems*. Springer Verlag, 2008.

[2] R. Alur and D. L. Dill. "A Theory of Timed Automata". In: *Theoretical Computer Science* 126 (1994), pp. 183–235.

[3] Rajeev Alur, Costas Courcoubetis, Thomas A. Henzinger, and Pei-Hsin Ho. "Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems". In: *Hybrid Systems*. Ed. by Robert L. Grossman, Anil Nerode, Anders P. Ravn, and Hans Rischel. Vol. 736. LNCS. Springer Berlin Heidelberg, 1993, pp. 209–229.

[4] J.-P. Aubin and A. Cellina. *Differential Inclusions: Set-Valued Maps and Viability Theory*. Springer Verlag, 1984.

[5] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. The MIT Press, 2008.

[6] Johan Bengtsson and Wang Yi. "Timed Automata: Semantics, Algorithms and Tools". In: *Lectures on Concurrency and Petri Nets*. Vol. 3098. LNCS. Springer, 2004, pp. 87–124.

[7] Luca Benvenuti, Davide Bresolin, Alberto Casagrande, Pieter Collins, Alberto Ferrari, Emanuele Mazzi, Alberto Sangiovanni-vincentelli, and Tiziano Villa. "Reachability Computation for Hybrid Systems with Ariadne". In: *Proceedings of the 17th IFAC World Congress*. 2008.

[8] Olivier Bouissou and Alexandre Chapoutot. "An Operational Semantics for Simulink's Simulation Engine". In: *Proceedings of the 13th ACM SIGPLAN/SIGBED International Conference on Languages, Compilers, Tools and Theory for Embedded Systems*. LCTES'12. ACM, 2012, pp. 129–138.

[9] Michael S. Branicky. "Introduction to Hybrid Systems". In: *Handbook of Networked and Embedded Control Systems*. Ed. by Dimitrios Hristu-Varsakelis and William S. Levine. Control Engineering. Birkhäuser Boston, 2005, pp. 91–116.

[10] B. F. Caviness and J. R. Johnson, eds. *Quantifier Elimination and Cylindrical Algebraic Decomposition*. Wien: Springer, 1998.

[11] Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. *Model Checking*. MIT Press, 1999.

[12] Werner Damm, Guilherme Pinto, and Stefan Ratschan. "Guaranteed Termination in the Verification of LTL Properties of Non-linear Robust Discrete Time Hybrid Systems". In: *International Journal of Foundations of Computer Science (IJFCS)* 18.1 (2007), pp. 63–86.

[13] P. Derler, E.A. Lee, and A.S. Vincentelli. "Modeling Cyber-Physical Systems". In: *Proceedings of the IEEE* 100.1 (Jan. 2012), pp. 13–28.

[14] Andreas Eggers, Martin Fränzle, and Christian Herde. "SAT Modulo ODE: A Direct SAT Approach to Hybrid Systems". In: *Automated Technology for Verification and Analysis*. Vol. 5311. LNCS. 2008.

[15] J.C. Eidson, E.A. Lee, S. Matic, S.A. Seshia, and Jia Zou. "Distributed Real-Time Software for Cyber-Physical Systems". In: *Proceedings of the IEEE* 100.1 (Jan. 2012), pp. 45–59.

[16] Joel M. Esposito, Vijay Kumar, and George J. Pappas. "Accurate Event Detection for Simulating Hybrid Systems". In: *Hybrid Systems: Computation and Control, HSCC 2001*. Ed. by M.D. Di Benedetto and A. Sangiovanni-Vincentelli. Vol. 2034. LNCS. Springer, 2001, pp. 204–217.

[17] A. F. Filippov. "Differential equations with discontinuous right-hand side". In: *Mat. Sb.* 51 (1960). In Russian, pp. 99–128.

[18] Peter Franek, Stefan Ratschan, and Piotr Zgliczynski. "Quasi-decidability of a Fragment of the First-order Theory of Real Numbers". In: (2013). Under revision at the Journal of Automated Reasoning. URL: `http://arxiv.org/abs/1309.6280`.

[19] Peter Franek, Stefan Ratschan, and Piotr Zgliczynski. "Satisfiability of Systems of Equations of Real Analytic Functions is Quasi-decidable". In: *MFCS 2011: 36th International Symposium on Mathematical Foundations of Computer Science*. Vol. 6907. LNCS. Springer, 2011, pp. 315–326.

[20] Goran Frehse. "PHAVer: algorithmic verification of hybrid systems past HyTech". In: *International Journal on Software Tools for Technology Transfer (STTT)* 10.3 (2008), pp. 263–279.

[21] Sukumar Ghosh. *Distributed Systems: An Algorithmic Approach*. 2nd edition. CRC Press, 2014.

[22] Rafal Goebel, Ricardo G. Sanfelice, and Andrew R Teel. *Hybrid Dynamical Systems: Modeling, Stability, and Robustness*. Princeton University Press, 2012.

[23] Orna Grumberg and Helmut Veith, eds. *25 Years of Model Checking*. Vol. 5000. Lecture Notes in Computer Science. Springer, 2008.

[24] Hervé Guéguen, Marie-Anne Lefebvre, Janan Zaytoon, and Othman Nasri. "Safety verification and reachability analysis for hybrid systems". In: *Annual Reviews in Control* 33.1 (2009), pp. 25–36.

[25] W. P. M. H. Heemels, D. Lehmann, J. Lunze, and B. De Schutter. "Introduction to Hybrid Systems". In: *Handbook of Hybrid Systems Control*. Ed. by Jan Lunze and Françoise Lamnabhi-Lagarrigue. Cambridge University Press, 2009.

[26] Thomas A. Henzinger, Pei-Hsin Ho, and Howard Wong-Toi. "HYTECH: a model checker for hybrid systems". In: *International Journal on Software Tools for Technology Transfer (STTT)* 1 (1997), pp. 110–122.

[27] Thomas A. Henzinger, Peter W. Kopke, Anuj Puri, and Pravin Varaiya. "What's Decidable about Hybrid Automata". In: *Journal of Computer and System Sciences* 57 (1998), pp. 94–124.

[28] Thomas A. Henzinger and Rupak Majumdar. "Symbolic Model Checking for Rectangular Hybrid Systems". In: *Proc. of the Sixth Workshop on Tools and Algorithms for the Construction and Analysis of systems (TACAS'00)*. LNCS 1785. 2000, pp. 142–156.

[29] J.C. Jensen, D.H. Chang, and E.A Lee. "A model-based design methodology for cyber-physical systems". In: *Wireless Communications and Mobile Computing Conference (IWCMC), 2011 7th International*. July 2011, pp. 1666–1671.

[30] Ranjit Jhala and Rupak Majumdar. "Software model checking". In: *ACM Comput. Surv.* 41.4 (2009), pp. 1–54.

[31] S. Kowalewski, M. Garavello, H. Guéguen, G. Herberich, R. Langerak, B. Piccoli, J. W. Polderman, and C. Weise. "Hybrid automata". In: *Handbook of Hybrid Systems Control*. Ed. by Jan Lunze and Françoise Lamnabhi-Lagarrigue. Cambridge University Press, 2009.

[32] M. Kwiatkowska, G. Norman, and D. Parker. "Stochastic Model Checking". In: *Formal Methods for the Design of Computer, Communication and Software Systems: Performance Evaluation (SFM'07)*. Ed. by M. Bernardo and J. Hillston. Vol. 4486. LNCS (Tutorial Volume). Springer, 2007, pp. 220–270.

[33] Gerarde Lafferriere, George J. Pappas, and Sergio Yovine. "Symbolic Reachability Computation for Families of Linear Vector Fields". In: *Journal of Symbolic Computation* 32.3 (2001), pp. 231–253.

[34] Edward A. Lee and Sanjit A. Seshia. *Introduction to Embedded Systems, A Cyber-Physical Systems Approach*. `http://LeeSeshia.org`, 2011.

[35] Qiao Li. "Scheduling in Cyber-Physical Systems". PhD thesis. Carnegie Mellon University, 2012. URL: `http://repository.cmu.edu/dissertations/91`.

[36] Zhengguo Li, Yengchai Soh, and Changyun Wen. *Switched and Impulsive Systems—Analysis, Design and Applications*. Vol. 313. LNCIS. Springer, 2005.

[37] Daniel Liberzon. *Switching in Systems and Control*. Birkhäuser Basel, 2003.

[38] Hai Lin and Panos J Antsaklis. “Hybrid dynamical systems: An introduction to control and verification”. In: *Found. Trends Syst. Control* 1.1 (2014), pp. 1–172.

[39] Jan Lunze and Françoise Lamnabhi-Lagarrigue, eds. *Handbook of Hybrid Systems Control*. Cambridge University Press, 2009.

[40] John Lygeros. “Handbook of Networked and Embedded Control Systems”. In: Springer, 2005. Chap. An Overview of Hybrid Systems Control, pp. 519–537.

[41] Yuri Matiyasevich. “Enumerable sets are Diophantine”. In: *Doklady Akademii Nauk SSSR* 191 (1970), pp. 279–282.

[42] Steven P. Miller, Michael W. Whalen, and Darren D. Cofer. “Software Model Checking Takes Off”. In: *Commun. ACM* 53.2 (2010), pp. 58–64.

[43] P. J. Mosterman. “An Overview of Hybrid Simulation Phenomena and Their Support by Simulation Packages”. In: *HSCC’99*. Ed. by F.W. Vaandrager and J.H. van Schuppen. LNCS 1569. Springer, 1999.

[44] Gabriela Nicolescu and Pieter J. Mosterman, eds. *Model-Based Design for Embedded Systems*. CRC Press, 2009.

[45] André Platzer. *Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics*. Heidelberg: Springer, 2010.

[46] Akshay Rajhans and Bruce H. Krogh. “Compositional Heterogeneous Abstraction”. In: *Proceedings of the 16th International Conference on Hybrid Systems: Computation and Control*. HSCC ’13. ACM, 2013, pp. 253–262.

[47] Jean-François Raskin. “An Introduction to Hybrid Automata”. In: *Handbook of Networked and Embedded Control Systems*. Ed. by Dimitrios Hristu-Varsakelis and William S. Levine. Control Engineering. Birkhäuser Boston, 2005, pp. 491–517.

[48] Stefan Ratschan. “Efficient Solving of Quantified Inequality Constraints over the Real Numbers”. In: *ACM Transactions on Computational Logic* 7.4 (2006), pp. 723–748.

[49] Stefan Ratschan. “Safety Verification of Non-linear Hybrid Systems is Quasi-decidable”. In: *Formal Methods in System Design* 44.1 (2014), pp. 71–90.

[50] Stefan Ratschan and Zhikun She. “Safety Verification of Hybrid Systems by Constraint Propagation Based Abstraction Refinement”. In: *ACM Transactions in Embedded Computing Systems* 6.1 (2007), pp. 1–23.

[51] A. J. van der Schaft and J. M. Schumacher. *An Introduction to Hybrid Dynamical Systems*. Springer, 2000.

[52] B. De Schutter, W. P. M. H. Heemels, J. Lunze, and C. Prieur. “Survey of Modeling, Analysis, and Control of Hybrid Systems”. In: *Handbook of Hybrid Systems Control*. Ed. by Jan Lunze and Françoise Lamnabhi-Lagarrigue. Cambridge University Press, 2009.

[53] Joseph Sifakis. “A vision for computer science—the system perspective”. In: *Central European Journal of Computer Science* 1.1 (2011), pp. 108–116. ISSN: 1896-1533.

[54] Paolo Tabuada. *Verification and Control of Hybrid Systems*. Springer Verlag, 2009.

[55] A. Tarski. *A Decision Method for Elementary Algebra and Geometry*. Also in [10]. Berkeley: Univ. of California Press, 1951.

Dipl.-Ing. Dr.techn. Stefan Ratschan

Education

1998, Ph.D. in Computer. Science., Joh. Kepler Univ., Linz, Austria
1995, Diploma Degree in Comp. Sc., Joh. Kepler Univ., Linz, Austria
1989, High School Exam (Matura), Schärding, Austria

Employment History

2010–	Odborný asistent, Faculty of Information Technology, Czech Technical University in Prague
2006–	Researcher, Institute of Computer Science of the Czech Academy of Sciences
2002–2006	Researcher, Max-Planck-Institut für Informatik, Saarbrücken, Germany
2001–2002	Marie Curie Postdoctoral Fellow, Universitat de Girona, Spain
1999–2001	Postdoc, Johannes Kepler University, Linz, Austria
1998–1999	National Social Service, Bosnia and Herzegovina
1994–1998	Research Assistant, Joh. Kepler University, Linz, Austria

Selected Invited Talks

- Invited Plenary Talk, Sixth International Conference on Mathematical Aspects of Computer and Information Sciences (MACIS), Berlin, 2015
- Panel Session Speaker, 17th International Conference on Hybrid Systems: Computation and Control (HSCC), Berlin, 2014
- Invited Plenary Talk, SNC 2009, 3rd International Workshop on Symbolic-Numeric Computation, Kyoto, Japan, 2009

Selected Publications

- Peter Franek and Stefan Ratschan: Effective Topological Degree Computation Based on Interval Arithmetic, Mathematics of Computation, Volume 84, 2015, 1265–1290
- Stefan Ratschan: Safety Verification of Non-linear Hybrid Systems is Quasi-Decidable, Formal Methods in System Design, Volume 44, Issue 1, 2014, pp. 71-90

- Lijun Zhang, Zhikun She, Stefan Ratschan, Holger Hermanns, and Ernst Moritz Hahn: Safety Verification for Probabilistic Hybrid Systems, European Journal of Control, Volume 18, Number 6, 2012, pp. 572–587

- Stefan Ratschan and Zhikun She: Providing a Basin of Attraction to a Target Region of Polynomial Systems by Computation of Lyapunov-like Functions, SIAM J. Control and Optimization, Volume 48, Number 7, 2010, pp. 4377–4394

- Martin Fränzle, Christian Herde, Tino Teige, Stefan Ratschan, and Tobias Schubert: Efficient Solving of Large Non-linear Arithmetic Constraint Systems with Complex Boolean Structure, Journal on Satisfiability, Boolean Modeling and Computation, Special Issue on SAT/CP Integration, Volume 1, 2007, pp. 209–236

- Stefan Ratschan and Zhikun She: Safety Verification of Hybrid Systems by Constraint Propagation Based Abstraction Refinement, ACM Transactions on Embedded Computing Systems, Volume 6, Number 1, 2007, pp. 1–23

- Stefan Ratschan, Efficient Solving of Quantified Inequality Constraints over the Real Numbers, ACM Transactions on Computational Logic, Volume 7, Number 4, 2006, pp. 723–748

Teaching

- Systems Theory, semester course WS 2012/2013, 2013/14, 2014/15 (2+1 hours, 200-250 students each), Czech Technical University, Prague

- Formal Methods and Specification, semester course SS 2011, 2012, 2013, 2015 (2+1 hours, appr. 80 students each), Czech Technical University, Prague

- Systems Modeling and Analysis, semester course WS 2010/2011, 2011/12 (2+1 hours), Czech Technical University, Prague

- Non-linear Numerical Constraint Solving, Automatic Verification and Analysis of Complex Systems, 1st AVACS Spring School, Universität Oldenburg, March 2010

- Hybrid Systems: Modeling, Simulation, Verification, Semester course WS 2008/2009 (2 hours), Charles University, Prague

- Global Optimization, semester course WS 2007/2008 (2 hours) Charles University, Prague
- Solving Constraints over the Real Numbers, semester course 2003/04 (2 hours of lecture and 2 hours of exercises per week) Universität des Saarlandes, Germany
- Solving Quantified Real-Number Constraints, RAAG Summer School on Tools for Real Algebraic Geometry, 2003 Rennes, France
- Foundations of Computer Geometry, 1 year course 1998/1999 (2 hours of lectures and 2 hours in lab per week) Faculty of Natural Sciences and Mathematics University of Sarajevo, Bosnia and Hercegovina