Simulation-based Verification of Cardiac Pacemakers with Guaranteed Coverage

Zhenqi Huang [1], Chuchu Fan [1], Alexandru Mereacre [2],
Sayan Mitra [1], Marta Kwiatkowska [2]
[1] {zhuang25,cfan10,mitras}@illinois.edu
Department of Electrical and Computer Engineering,
University of Illinois at Urbana-Champaign.
[2] {Marta.Kwiatkowska,Alexandru.Mereacre}@cs.ox.ac.uk
Department of Computer Science,
University of Oxford.

Design and testing of pacemaker is challenging because of the need to capture the interaction between the physical processes (e.g. voltage signal in cardiac tissue) and the embedded software (e.g. a pacemaker). At the same time, there is a growing need for design and certification methodologies that can provide quality assurance for the embedded software. We describe recent progress in simulation-based techniques that are capable of ensuring guaranteed coverage. Our methods employ discrepancy functions, which impose bounds on system dynamics, and proceed through iteratively constructing over-approximations of the reachable set of states. We are able to prove time bounded safety or produce counterexamples. We illustrate the techniques by analyzing a family of pacemaker designs against time duration requirements and synthesize safe parameter ranges. We conclude by outlining the potential uses of this technology to improve the safety of medical device designs.

***Index Terms*—biological networks; hybrid systems; invariants; verification; safety; pacemakers.**

I. Introduction

Computer simulations are an all-powerful tool for understanding the behavior of complex systems such as implantable medical devices. In many cases, the design and test processes are centered around creating models and prototypes, and then studying them through simulations. The term "simulation" covers considerable ground, ranging from hardware-in-the-loop (HIL) simulations where some of the components are real artifacts, to numerical simulation of mathematical models, with other approaches in between. While simulations are relatively easy to compute, it can be hard to draw useful inferences from them. One crucial problem is that of *coverage*. A single simulation of a model, done carefully, informs us about one behavior of the system. Most systems produce uncountably many behaviors with different inputs from the environment and different choices of model parameters. Any study involving a finite number of simulations necessarily leaves out a large proportion of these behaviors, and this lack of coverage makes it impossible to draw robust and useful conclusions about the behavior of the system. Some studies [8], [11] exploit stochastic optimization techniques to improve the possibility of catching a bug, however these approaches cannot provide a sure guarantee of safety.

Consider the coverage problem in the context of design and testing of implantable cardiac pacemakers. Pacemakers normally have sensors to detect events directly in the heart tissue, such as atrial sense and ventricular sense. Using the information from the sensor, the pacemaker decides whether and when to generate a stimulus to the heart. In order to test the pacemaker in its "natural" environment, we have to close the loop with an appropriate model of the human heart. This by itself is a challenging problem, although there are significant benefits to this approach and remarkable progress has been made in the past decade [1]–[3]. For instance, the model presented by Grosu et al [3] describes the heart as a network of automata, where each automaton models the currents and voltages in a fragment of the cardiac tissue. Each automaton model has several parameters, such as the initial membrane voltage and the resistance of the conduction channels. In addition, the pacemaker itself has parameters, e.g., the frequency and the magnitude of the stimulation pulses. Many of these parameters cannot be measured precisely due to the uncertainty in their values. Even if we assign specific values to all parameters and generate a simulation, to draw inferences about the behavior of the whole system we need to cover sets of behaviors that arise from parameter ranges. On the other hand, by excessive inference, we may over-approximate the behavior of the system and produce false negative. This "accurate coverage" problem is particularly challenging.

Software bugs and failures resulted in 24% of all medical devices recalls in 2011. According to the US Food and Drug Administration (FDA), errors in embedded software have led to a substantial increase in safety alerts or even patient death. Combined with the relative lack of standardization in the field of medical devices, there is an urgent need to develop methodologies for ensuring correct behavior of embedded pacemaker software and their certification. In this work we focus on effectively solving the coverage problem for medical cyber-physical systems.

Until recently, deriving coverage guarantees was only pos-

The authors are supported by NSA SoS grant (W911NSF-13-0086), AFOSR YIP grant (FA9550-12-1-0336), NSF CAREER grant (CNS 10-54247), ERC AdG VERIWARE (246967) and ERC PoC VERIPACE (620196).

sible for microprocessors and programs. While massive, these essentially digital systems lack the continuous nonlinear dynamics typical of electrical conduction in a cardiac cell. Nor do they involve the "hybrid" behavior that arises in the interaction of a physical process (say, voltage oscillations) with embedded software (pacemaker). Over the last few years we have begun to witness the development of techniques that combine computer simulations with algorithmic static analysis to obtain guarantees about behaviors of these types of complex systems. A number of recent studies, by us and others, demonstrate the promise of these techniques in finding design flaws and uncovering model inconsistencies [4]–[8].

Our approach builds on algorithms that cover a set of model behaviors from a single simulation of the model. Given a model, to check if it violates the property, we over-approximate its behaviors with increasingly higher precision until we reach a conclusion. For constructing over-approximations, we exploit continuity of the trajectories. First, a numerical simulation of a single behavior of the model is computed for a particular choice of the parameters. This simulation is then "bloated" by a factor to over-approximate all behaviors that arise from similar choices. Repeating this simulate-and-bloat process for different choices of the initial states and parameter values, we can over-approximate all possible behaviors that can arise from all parameter choices. Further, it is important that the over-approximation can be made more precise so that false positives can be eliminated. To turn this idea into an algorithm, we introduced the notion of *discrepancy* which (a) upper bounds the distance between two neighboring behaviors and (b) the bound converges to zero as the parameter choices for the two behaviors get closer and closer [4], [6]. It has been shown that, for an expressive class of models, indeed one can find discrepancy functions that meet these criteria [5].

For the pacemakers, a key property or requirement is that the cardiac cell voltage oscillations (action potentials) induced by the pacemaker are regular (as in Figure 2a) and not alternating (as in Figure 2c). The alternating behavior is well-known to be a precursor for cardiac arrhythmic disorders, including ventricular tachycardia and fibrillation [3]. Another requirement is that, even if initially the oscillations are irregular, they then converge to a regular pattern. Our simulation-based verification algorithms can automatically check that these properties hold not just for a specific choice of the model parameters, but for the whole range of possible values. This provides assurance about uncountably large sets of behaviors from a finite number of simulations.

While we believe that the simulation-based verification technology for nonlinear hybrid systems is poised to take-on industrial challenges, we conclude this article with an overview of the open research questions and by highlighting the more practical issues that need to be addressed for this technology to be adopted in the design and certification of implanted medical devices.

II. Models and Requirements for Embedded Medical Devices

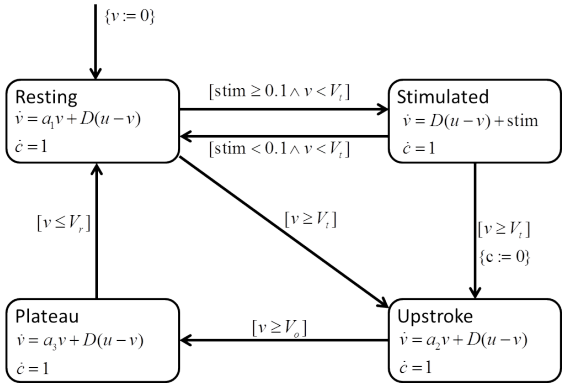
The first step for model-based design and analysis is to construct a model of the system. Thanks to vigorous research activities over the past two decades, now there are several good choices for modeling cyber-physical systems. Although we use the language of the *Hybrid Input/Output Automata* framework [9], the analysis methods that we develop are not particularly sensitive to this choice.

The key component of our framework is a *hybrid I/O automaton (HIOA)*—a (possibly nondeterministic and nonlinear) state machine whose state variables evolve discretely and continuously. The discrete transitions are written in the usual precondition-effect style and the continuous evolution is described by differential and algebraic equations. The solutions of these equations are called *trajectories*. Furthermore, an HIOA can share information over continuous variables and discrete transitions with other automata in the system. This is useful for modeling large systems in a modular or compositional fashion.

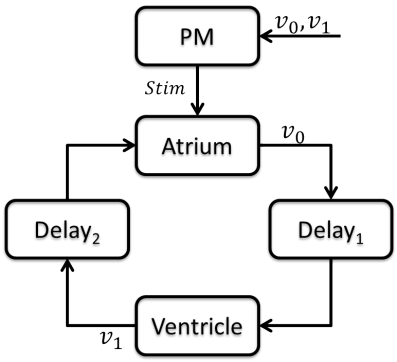
Example 1. *A HIOA model of a cardiac cell is shown in Figure 1a [10]. The four discrete locations* Resting*,* Stimulated*,* Plateau*, and* Upstroke *capture the four phases of the action potential cycle of the cell. The continuous variable v models the membrane voltage and c measures time elapsed since last* Upstroke*. The input u captures the diffusion effect of the neighboring cells and $stim$ is the input stimulation from the pacemaker. The evolution of these continuous variables is described by linear differential equations. The transitions define instantaneous changes. For example, preconditions on the membrane voltage such as $v \geq V_t$ determine when the automaton transitions from* Resting *to* Upstroke *to* Plateau *in the absence of pacing. The transitions to and out of* Stimulated *model the influence of $stim$ from the pacemaker.*

Figure 1b shows a larger HIOA obtained by composing five HIOAs: Atrium *and* Ventricle *are models of individual oscillators as in Figure 1a. Each* Delay *automaton produces a delayed impulse when it detects an input spike. The pacemaker (*PM*) records the intervals between two successive ventricle and atrium pulses and, if either of the intervals exceeds the threshold called* Lower Rate Interval (LRI)*, then it stimulates* Atrium*. Note that in Fig 1b there is a connection from the ventricle to the atrium. This allows modeling the so called retrograde conduction of the heart, which is the transmission of a cardiac impulse from the ventricles to the atria. However, to make full use of it, one has to employ a more physiologically accurate cell model (see Section IV).*

An *execution* of a hybrid automaton records the evolution of the variables for a particular choice of the initial state and all other parameters of the model. Formally, a *bounded execution fragment* is a finite sequence of trajectories $\xi_0, \xi_1, \ldots$, such that, from the last state of ξ_i to the first state of ξ_{i+1}, there is a valid transition of the automaton. In Figure 2a, we plot an execution in which the atrium (red) and the ventricle (green) stimulate each other and the pacemaker does nothing. If the delay between the atrium and the ventricle is higher,



(a) Model of a single cardiac cell oscillator. The hybrid automaton has four locations corresponding to four phases of the oscillator. It has two continuous state variables v, c and two input variables u and $stim$. The remaining symbols represent constant parameters.



(b) Model of the whole heart-pacemaker interface. The model has 5 components. The Atrium and Ventricle are HIOA presented in Fig. 1a. PM is the pacemaker module.

Fig. 1: Models of the cardiac cell and the heart-pacemaker interface.

the pacemaker (blue) activates periodically (Figure 2b). For even larger delays the atrium can be stimulated both by the pacemaker and the ventricle (Figure 2c).

A state of the automaton is *reachable* if it is reached at some point along some execution. Requirements can be stated in terms of *invariant* assertions involving the variables of the automaton model. For example, we examine an invariant that the heart rate remains in a proper interval. This requirement Inv_1 defines the set of states of the model that satisfy it. Then, to infer that all behaviors of the model satisfy this requirement, it suffices to check that the set of reachable states of the model is contained in the set defined by Inv_1.

The design task of the pacemaker is to choose the value of LRI such that the frequency of visiting Upstroke is always in some acceptable range. For a ventricle cell the frequency of visiting the Upstroke location is equivalent to the heart rate. This requirement can be stated as an invariant of the complete HIOA by adding auxiliary variables. More generally, LRI should be chosen so that it meets this requirement for a range of delay values that arise from the uncertainty in the physiological parameters. It is also useful to decide a range of safe values for LRI so that the clinician can make a choice from that range based on other health-related factors.

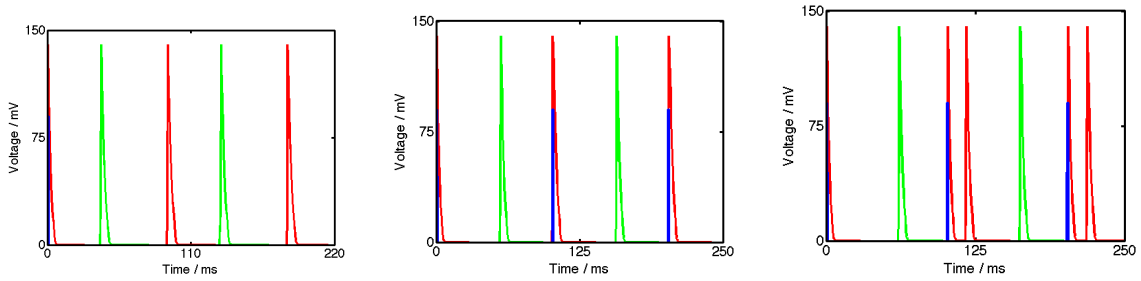
III. The Principle of Simulation-based Verification

For a given model we consider the bounded time invariant verification problem, which is parameterized by a time bound, a compact set of initial states and parameter ranges, and a set of unsafe states. The aim is to decide if there exists a particular choice of an initial state and parameter valuations that leads to a behavior that reaches an unsafe set. The simulation-based verification algorithm creates a finite cover of the set of initial states with a representative point for each cover. Next the simulator generates numerical approximations from all representative points. Then by *bloating* each simulation by an appropriate factor depending on the size of the cover it represents, the verification algorithm over-approximates all the behaviors from the cover. By doing so, the union of the bloated simulation is an over-approximation of the reachable states of the system. If the over-approximation proves the invariant or produces a counterexample the algorithm terminates. Otherwise, it creates finer covers of the initial states and repeats the earlier steps to compute more precise over-approximations.

In order to make this procedure sound, the bloating factor should be large enough so that each bloated simulation is an over-approximation of the reachable states[1] from a neighborhood of the initial state of the simulation. However, for relative completeness (modulo the precision of the machine and the robustness of the property), the computation can be made arbitrarily precise by refining the covers. These two opposing requirements are captured in the definition of a *discrepancy function* of [4]: for an n-dimensional dynamical system, it is any function $\beta : \mathbb{R}^{2n} \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ such that (a) it gives an upper-bound on the distance between any two trajectories $\xi(x,t)$ and $\xi'(x,t)$ of the system, namely $|\xi(x,t) - \xi(x',t)| \leq \beta(x,x',t)$, and (b) it vanishes for any t as x approaches x'. For linear ODEs, the discrepancy function β can be readily computed from the matrix norms, while sensitivity analysis yields an alternative approach [7]. For nonlinear models $\dot{y}(t) = f(t, y(t))$, the Lipschitz constant of f gives an exponential discrepancy, and if a contraction metric or incremental Lyapunov function can be computed then it gives a much tighter bound [4]. In more recent studies, the notion of discrepancy function is extended to capture the effect of different inputs, namely *input-to-state discrepancy* [5], [6]. For large models with many subsystems, such as the pacemaker network in Fig. 1b, the discrepancy of the overall system can be composed from the input-to-state discrepancy of the subsystems.

Simulations, discrepancy functions, and bloating yield over-approximations of the reachable states of a dynamical system from a set of initial states. By drawing more samples from the initial set, computing the reachtubes, and checking whether or not the reachtubes are contained in the invariant, we

[1]We call a bloated simulation a *reachtube*.



(a) The ventricle and atrium stimulate each other in turns with a proper rate. The pacemaker generates no impulse.

(b) The pacemaker generates periodic impulses to keep the lower rate of the atrium.

(c) The atrium is stimulated twice in a cycle, due to the pacemaker and the delayed pulse from the ventricle.

Fig. 2: Three representative behaviors of the model. The red/green/blue curves correspond to the voltage of the atrium/ventricle/stimulus respectively.

not only can prove time bounded invariants, but also find counterexamples. That is, if the algorithm terminates and returns "safe" or "unsafe", then indeed the dynamical system satisfies or violates the given invariant. Furthermore, if the dynamical system *robustly* satisfies or violates the invariant, then the algorithm is guaranteed to terminate[2]. In contrast with statistical model checking [11], where the output is described by confidence intervals, this approach gives absolute guarantees about all possible behaviors that can arise from the choices in the model parameters.

In order to make this approach work for hybrid automata with guards, mode invariants, and resets, and in particular to produce counterexamples, we need one more key idea: we need to distinguish reachtubes that *must* contain actual executions of the automaton from those that are merely artifacts of over-approximation. For example, if a reachtube fragment is completely contained in the invariant of a location, it must contain at least one actual behavior of the automaton. This calculation of may and must reachtubes is propagated with respect to the guards, invariants, and the reset maps of the hybrid automaton to obtain the soundness and relative completeness guarantees.

The above algorithm has been implemented in the publicly available verification tool C2E2 [4], [12] and a compositional approach is implemented in a Matlab toolbox [5]. C2E2 takes as input a hybrid automaton model in an .xml format or as a Stateflow™ diagram and performs verification of time bounded invariants or temporal precedence properties.

IV. Applications and Experiments

Using C2E2 and the Matlab-based implementation of the above algorithms, we verify a number of properties for several different models of the pacemaker-heart system. We can also find safe parameter ranges for the pacemaker. We build a Simulink/Stateflow model of a network of cardiac cells composed with a pacemaker. Each cell or node in the network is modeled by an HIOA which interacts with the neighboring cells, and the pacemaker as described by Grosu et al. in [3]. This model is more physiologically accurate than the one we introduced in Example 1. It enables us to simulate the retrograde conduction. Each cell model has 4 continuous variables and 29 different sets of nonlinear ODE describing different modes; a network with 8 cells can have 32 continuous variables and 29^8 different locations. This model has a large set of behaviors that depend on the choice of the parameters (e.g., initial membrane voltages, resistance of channels, frequency of the pacemaker's impulse outputs, etc.). Analyzing these types of models is well beyond the capabilities of traditional verification algorithms; nevertheless, using the simulation-based approach we are able to verify whether the membrane voltage of the cell remains within some threshold θ_{max} [5]. Table I shows the typical verification times for different model sizes.

N	θ_{max}	Sims	Refs	RT(s)	$v \leq \theta_{max}$
3	2	16	0	104.8	✓
3	1.65	16	0	103.8	✓
3	1.5	NA	NA	9.0	×
5	2	3	0	208.0	✓
5	1.65	170	125	945.0	✓
5	1.5	NA	NA	63.4	×
8	2	3	0	240.1	✓
8	1.65	73	9	2376.5	✓
8	1.5	NA	NA	119.7	×

TABLE I: Scaling with model size. N: number of cells in the model, θ_{max}: threshold voltage defining invariant, Sims: number of simulations, Refs: number of refinements, RT: running time in seconds.

Simulation-based verification can also check duration properties (time difference between two events) for a bounded horizon. In Example 1, one design task is to determine the parameter LRI for a pacemaker such that the time difference between the Atrium and Ventricle events can be maintained within a desired range. The machine PM in Fig. 1b models a family of pacemaker designs with LRI as a parameter in the interval $[a, b]$. First, all instances of LRI in the model are replaced with a dummy variable l. We specify l as a constant described by the ODE $\dot{l} = 0$ in every location. We initialize

[2]A model $\mathcal{A}$ is said to robustly satisfy/violate a property P if there exists a small ϵ such that if the parameters of $\mathcal{A}$ are perturbed by ϵ then the resulting perturbed model $\mathcal{A}_\epsilon$ also satisfies/violates P.

LRI	Delay1	Delay2	Sims	RT(s)	$T \in Inv$
$[51, 53]$	$[50, 51]$	$[48, 49]$	6	5.7	✓
$[49, 51]$	$[45, 46]$	$[51, 52]$	14	21	✓
$[49, 51]$	$[49, 51]$	$[51, 52]$	27	76.2	×
$[43, 46]$	$[41, 44]$	$[39, 42]$	24	40.6	✓

TABLE II: Range of pacemaker designs (LRI values) for which the required invariant is maintained and violated. LRI/ Delay1 / Delay2: the range of values that LRI and delays of Delay can take, Sims: number of simulations, RT: running time in seconds.

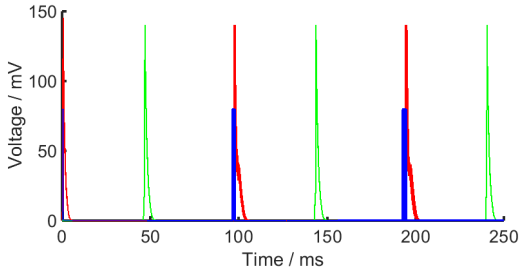


Fig. 3: Reach set of Example 1 with LR in the range $[49, 51]$ and the two delays in ranges $[45, 46]$, $[51, 52]$. The time duration property is satisfied in this case; however, we observe that, if we decrease LRI by 1, the atrium (red) will get excited twice in the last two cycles.

l to take a value nondeterministically in the range $[a, b]$. This modified version of PM captures all possible behaviors with LRI in the desired range. In a similar manner, Delay1 and Delay2 can be modified to model a range of values for the delay parameters between the Atrium and Ventricle.

With this model we can examine whether the family PM of pacemaker designs meets the time duration property. We are able the verify a range of design parameters for values ranging up to 3ms or find a counterexample in tens of seconds. Table II shows some of the ranges of parameter values for which the invariant is provably preserved or violated. Reach sets of the atrium (red), ventricle (green) and pacemaker (blue) are shown in Figure 3. This type of analysis can be used by a clinician to determine a safe range of customization parameters for the pacemaker based on the physiological characteristics of the patient. From experimental results we observe that our simulation-based verification technique scales to large models (tens of variables and millions of locations) with reasonable amount of uncertainty (several decision parameter).

V. Opportunities and Challenges

The simulation-based verification technology for nonlinear hybrid systems presented in this paper is poised to take on the challenges in design and analysis of medical devices, providing a rigorous foundation for methodologies needed for their verification and certification. We enumerate some of the open problems related to this approach.

In general, a nondeterministic model can have multiple transitions happen or trajectories starting from the same initial states. For example, the evolution of the continuous variables may be captured by a differential inclusion. Verifying such a model using simulation remains an open problem, since current solvers can only simulate a single trajectory captured by a differential equation.

While there is a rudimentary understanding of how to generalize the simulation-based approach to handle *unbounded time properties* for strictly periodic models, how to develop a general theory and algorithms for the verification of such properties is less clear, but also the logical next step.

A different application of bounded-time analysis arises in the context of real-time patient monitoring, where it could address the issue of (false) *alarm fatigue* for medical professionals. When can the reachtubes be computed fast enough and accurately enough to become a tool for detecting anomalies?

Example 1 demonstrated that our verification algorithms can be used to synthesize safe parameters for a given design. This can be a useful tool in customization of medical devices. The general problem of synthesis using simulations is only beginning to be researched and there is a panoply of interesting questions.

Of course, in addition, important questions have to be addressed by practitioners before we see the first real application: what is the technology insertion point in the design cycle? How to get suitable models of the device and its physiological environment? Where does the risk of failure justify the engineering cost of this rigorous analysis? If these questions are settled positively, at least for a few devices such as pacemakers and infusion pumps, then they can help usher a new generation of smart and algorithmically certified medical devices.

References

[1] Z. Jiang, M. Pajic, S. Moarref, R. Alur, and R. Mangharam, "Modeling and verification of a dual chamber implantable pacemaker," in *TACAS*, pp. 188–203, 2012.

[2] T. Chen, M. Diciolla, M. Z. Kwiatkowska, and A. Mereacre, "A simulink hybrid heart model for quantitative verification of cardiac pacemakers," in *HSCC*, pp. 131–136, 2013.

[3] R. Grosu, G. Batt, F. H. Fenton, J. Glimm, C. L. Guernic, S. A. Smolka, and E. Bartocci, "From cardiac cells to genetic regulatory networks," in *CAV*, pp. 396–411, 2011.

[4] P. S. Duggirala, S. Mitra, and M. Viswanathan, "Verification of annotated models from executions," in *EMSOFT*, 2013.

[5] Z. Huang, C. Fan, A. Mereacre, S. Mitra, and M. Kwiatkowska, "Invariant verification of nonlinear hybrid automata networks of cardiac cells," in *Computer Aided Verification* (A. Biere and R. Bloem, eds.), vol. 8559 of *Lecture Notes in Computer Science*, pp. 373–390, Springer International Publishing, 2014.

[6] Z. Huang and S. Mitra, "Proofs from simulations and modular annotations," in *Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control*, pp. 183–192, ACM, 2014.

[7] A. Donzé and O. Maler, "Systematic simulation using sensitivity analysis," in *Hybrid Systems: Computation and Control*, pp. 174–189, Springer, 2007.

[8] Y. Annpureddy, C. Liu, G. Fainekos, and S. Sankaranarayanan, *S-taliro: A tool for temporal logic falsification for hybrid systems*. Springer, 2011.

[9] N. Lynch, R. Segala, and F. Vaandrager, "Hybrid i/o automata," *Information and Computation*, vol. 185, no. 1, pp. 105–157, 2003.

[10] P. Ye, E. Entcheva, R. Grosu, and S. A. Smolka, "Efficient modeling of excitable cells using hybrid automata," in *Proc. of CMSB*, vol. 5, pp. 216–227, 2005.

[11] A. Legay, B. Delahaye, and S. Bensalem, "Statistical model checking: An overview," in *Runtime Verification*, pp. 122–135, Springer, 2010.
[12] P. S. Duggirala, S. Mitra, M. Viswanathan, and M. Potok, "The C2E2 user's guide," 2014. Available from: http://publish.illinois.edu/c2e2-tool/.

Zhenqi Huang is a PhD candidate in Electrical and Computer Engineering at University of Illinois at Urbana-Champaign. He received his BS in Mechanical Engineering from Tsinghua University (China) in 2010. His research focuses on verification and controller synthesis of cyber-physical systems.

Chuchu Fan is a PhD candidate at Electrical and Computer Engineering Department, University of Illinois at Urbana-Champaign. She receives her Bachelor degree from Automation Control Department, Tsinghua University in 2013. Her research interests are in verification of nonlinear hybrid systems.

Alexandru Mereacre received his B.S. degree in Computer Science from the Technical University of Moldova in 2005, the M.S. degree in Computer Science from RWTH Aachen University (Germany) and the University of Trento (Italy) in 2007, and his Ph.D. degree from RWTH Aachen University. Currently he is a research assistant at University of Oxford (UK). His research interests are in formal methods and parameter synthesis for embedded systems.

Sayan Mitra is an Associate Professor of Electrical and Computer Engineering at the University of Illinois at Urbana-Champaign. His research interests are in formal methods, distributed systems and hybrid control systems with applications in automotive, medical, and robotic systems. He received a PhD from MIT and held a post-doctoral fellowship at California Institute of Technology. He has held visiting positions at Oxford University and the Air Force Research Laboratory at New Mexico. He received the National Science Foundation's CAREER Award, Air Force Office of Scientific Research Young Investigator Award, and the IEEE-HKN C. Holmes MacDonald Outstanding Teaching Award.

Marta Kwiatkowska is a Professor of Computing Systems and Fellow of Trinity College, University of Oxford. She holds a PhD from the University of Leicester. In 2014 she was awarded an honorary doctorate from KTH Royal Institute of Technology in Stockholm. Her research has focused on modeling and verification for probabilistic, timed and hybrid systems, with applications to complex systems, network protocols, medical devices and DNA computing. Kwiatkowska led the development of the PRISM model checker, the leading probabilistic model checker. She gave the Milner Lecture in 2012 in recognition of "excellent and original theoretical work which has a perceived significance for practical computing". Kwiatkowska is a member of Academia Europea and Fellow of the BCS. She serves on editorial boards of several journals, including IEEE Transactions on Software Engineering and Formal Methods in System Design.