Invariant verification of nonlinear hybrid automata networks of cardiac cells

Zhenqi Huang, Chuchu Fan, Alexandru Mereacre,
Sayan Mitra, and Marta Kwiatkowska

[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.

Abstract. Verification algorithms for networks of nonlinear hybrid automata (HA) can aid understanding and controling of biological processes such as cardiac arrhythmia, formation of memory, and genetic regulation. We present an algorithm for over-approximating reach sets of networks of nonlinear HA which can be used for sound and relatively complete invariant checking. First, it uses automatically computed input-to-state discrepancy functions for the individual automata modules in the network $\mathcal{A}$ for constructing a low-dimensional model $\mathcal{M}$. Simulations of both $\mathcal{A}$ and $\mathcal{M}$ are then used to compute the reach tubes for $\mathcal{A}$. These techniques enable us to handle a challenging verification problem involving a network of cardiac cells, where each cell has four continuous variables and 29 locations. Our prototype tool can check bounded-time invariants for networks with 5 cells (20 continuous variables, 29^5 locations) typically in less than 15 minutes for up to reasonable time horizons. From the computed reach tubes we can infer biologically relevant properties of the network from a set of initial states.
Keywords. biological networks; hybrid systems; invariants; verification.

1 Introduction

Central to understanding and controlling behavior of complex biological networks are invariant properties. For example, synchronization of the action potentials of cardiac cells and neurons is responsible for normal functioning of the heart and for formation of memory [7, 17], and maintenance of synchrony is an invariant property. Real-time prediction of loss of synchrony can enable automatic deployment of counter-measures. For instance, embedded defibrillator devices are being designed to preempt possible cardiac arrest that arises from loss of synchrony. Offline invariant checks can aid in debugging pacemakers and brain-machine interfaces. Checking invariant properties for networks of dynamical systems is challenging. Analytical results exist only for modules with relatively simple dynamics and on special types of topologies such as scale-free

and random graphs [8, 10, 41, 43, 46]. These approaches cannot be applied to modules with nonlinear and hybrid dynamics such as the models of cardiac cells in [12, 21]. Aside from the nonlinearities in the modules, the complete network model involves shared continuous variables between modules (ion-channels) which have limited support in analytical and verification approaches. In the absence of analytical approaches, one performs simulation experiments which are computationally inexpensive but fall short of providing guarantees and are of limited utility in studying invariants for sets of initial states or parameter values. For example, if we wanted to know if the voltage of an action potential stays within some range from a set of initial states, then a finite number of simulations cannot give us a provably correct answer.

In this paper, we present an algorithm for verifying bounded-time invariant properties of networks of deterministic nonlinear hybrid automata. The underlying principle is simulation-based verification which combines numerical simulations with formal analysis [5, 15, 16]. First, a simulation ψ is computed from a single initial state $\mathbf{v}$. This ψ is then bloated by *some factor* to over-approximate all executions from a neighborhood $B_{\mathbf{v}}$ of $\mathbf{v}$ of non-zero measure. By repeating this process for different $\mathbf{v}$'s, all behaviors from a set of initial states can be over-approximated and robust invariants can be checked. In [16], we used user-provided model annotations (*discrepancy functions*) to statically compute the bloating factor in a way that can make the over-approximations arbitrarily precise. The resulting algorithm enjoys scalability and relative completeness: if the system satisfies the invariant robustly, then the algorithm is guaranteed to terminate. The burden of finding discrepancy functions for large models is partly alleviated in [27] for nonlinear differential equations. That paper proposes *input-to-state (IS)* discrepancy functions for each module $\mathcal{A}_i$ of a larger system $\mathcal{A} = \mathcal{A}_1 \| \ldots \| \mathcal{A}_N$. These user-provided, albeit modular, annotations are used to construct a lower-dimensional nonlinear time-varying system whose trajectories give the necessary bloating factor for the trajectories of the system $\mathcal{A}$.

These previous results do not extend to hybrid systems with guards and resets, and their applicability is still limited by the annotation required from the user. One challenge is that individual simulations capture a particular sequence of locations. However, the states reached in a bloated version of the simulation may intersect with many other guards and visit a completely different sequence of locations. Our contributions address this and other technical hurdles, demonstrating a promising approach for invariant verification of nonlinear hybrid networks. (a) We present a new simulation-based verification algorithm for nonlinear hybrid networks that uses modular input-to-state (IS) discrepancy functions. Modular annotations and the simulation-based approach make it scalable. The algorithm is sound; it systematically discovers possible transitions and then generates new simulations for different location sequences. We identify general robustness conditions that yield relative-completeness. (b) We develop a set of techniques for automatically computing input-to-state discrepancy functions for a general class of nonlinear hybrid models. (c) The performance of our prototype implementation in checking bounded-time invariants of complex Simulink

models of cardiac cell networks illustrate the promise of the approach [26]. For networks with 5 cells, each with 4 dimensions and 29 locations, and multi-affine dynamics (total of 20 continuous variables, 29^5 locations), invariants for up to reasonable time horizons are established typically in less than 15 minutes. In two minutes, it finds counter-examples of networks with 8 cells. All of this enables us to check biologically relevant properties for cardiac cells networks.

Section 2 provides background for hybrid automata, whereas Section 3 introduces IS discrepancy and techniques for computing them. Section 4 describes the main algorithm and Section 5 presents its applications in checking cardiac networks. Finally, Section 6 discusses related works and concludes the paper.

2 Hybrid Automata Modules and Networks

Hybrid Input/Output Automata (HA) is a framework for specifying interacting modules that evolve discretely and continuously and share information over continuous variables and discrete transitions [32,37,38]. Please see Appendix A for related definitions and notations.

For a variable v, its type, denoted by $\mathit{type}(v)$, is the set of values that it can take. For a set of variables $\mathcal{V}$, a *valuation* $\mathbf{v}$ maps each $v \in \mathcal{V}$ to a point in $\mathit{type}(v)$. Given a valuation $\mathbf{v}$ for $\mathcal{V}$, the valuation of a particular variable $v' \in \mathcal{V}$, denoted by $\mathbf{v}.v'$, is the restriction of $\mathbf{v}$ to v'; for a set $V \subseteq \mathcal{V}$, $\mathbf{v}.V$ is the restriction of $\mathbf{v}$ to V. $\mathit{Val}(\mathcal{V})$ is the set of all valuations for $\mathcal{V}$. A *trajectory* for $\mathcal{V}$ models continuous evolution of the values of the variables over a closed interval $[0, T]$ called the *domain*. A trajectory ξ is a map $\xi : [0, T] \to \mathit{Val}(\mathcal{V})$. Restriction of ξ to a subset of variables $\mathcal{X} \subseteq \mathcal{V}$ is denoted by $\xi \downarrow \mathcal{X}$. For a trajectory ξ of $\mathcal{V} \cup \mathcal{U}$ with domain $[0, T]$, we define $\xi.\mathsf{fstate}$ as $(\xi \downarrow \mathcal{V})(0)$ and $\xi.\mathsf{lstate}$ as $(\xi \downarrow \mathcal{V})(T)$. A variable is *continuous* if all its trajectories are piece-wise continuous and it is *discrete* if its trajectories are piece-wise constant. A HA has a set of continuous variables $\mathcal{X}$ that evolve along trajectories (defined by differential equations with inputs $\mathcal{U}$) and can be reset, and a set of discrete variables $\mathcal{L}$ that change with transitions.

Definition 1. *A* Hybrid I/O Automaton (HA) $\mathcal{A}$ *is a tuple* $(\mathcal{L}, \mathcal{X}, \mathcal{U}, \Theta, \mathcal{D}, \mathcal{T})$ *where (a)* $\mathcal{L}$ *is a set of discrete variables.* $\mathit{Val}(\mathcal{L})$ *is the set of* locations. *(b)* $\mathcal{X}$ *is a set of real-valued continuous variables.* $\mathcal{V} := \mathcal{X} \cup \mathcal{L}$ *is the set of* state variables*;* $\mathit{Val}(\mathcal{V})$ *is the* state space. *(c)* $\mathcal{U}$ *is a set of real-valued* input variables*;* $\mathit{Val}(\mathcal{U})$ *is the* input space. *(d)* $\Theta \subseteq \mathit{Val}(\mathcal{V})$ *is a set of* start states*; (e)* $\mathcal{D} \subseteq \mathit{Val}(\mathcal{V}) \times \mathit{Val}(\mathcal{V})$ *is a set of* discrete transitions. *(f)* $\mathcal{T}$ *is the set of* trajectories *for* $\mathcal{V} \cup \mathcal{U}$ *that is closed under* prefix, suffix, *and* concatenation *[32]. Over any trajectory* $\xi \in \mathcal{T}$, $\mathcal{L}$ *remains constant. For any state* $\mathbf{v}$ *and piece-wise continuous input trajectory* η*, there exists a state trajectory* ξ *such that* $\xi.\mathsf{fstate} = \mathbf{v}$ *and either (i)* $\xi \downarrow \mathcal{U} = \eta$*, or (ii)* $\xi \downarrow \mathcal{U}$ *matches a prefix of* η *with a transition enabled at* $\xi.\mathsf{lstate}$.

A transition $(\mathbf{v}, \mathbf{v}') \in \mathcal{D}$, for any two states $\mathbf{v}$, $\mathbf{v}'$, is written as $\mathbf{v} \to_{\mathcal{A}} \mathbf{v}'$ or as $\mathbf{v} \to \mathbf{v}'$ when $\mathcal{A}$ is clear from the context. The transitions of $\mathcal{A}$ are specified for pairs of locations in the guard-reset style. For each pair (ℓ, ℓ') of locations the *guard* $G_{\ell,\ell'} \subseteq \mathit{Val}(\mathcal{X})$ is the set of states from which a transition from location ℓ to ℓ' is enabled and the reset map is a continuous function $\mathit{Val}(\mathcal{X}) \to \mathit{Val}(\mathcal{X})$.

For location $\ell \in \mathit{Val}(\mathcal{L})$, the trajectories of $\mathcal{A}$ are defined by a *trajectory invariant* $I_\ell \subseteq \mathit{Val}(\mathcal{X})$ and a set of ordinary differential equations (ODEs) involving the variables in $\mathcal{X}$ and $\mathcal{U}$. The ODE is specified by a Lipschitz continuous function called *dynamic mapping* $f_\ell : \mathit{Val}(\mathcal{X}) \times \mathit{Val}(\mathcal{U}) \to \mathit{Val}(\mathcal{X})$. Given a input trajectory η of $\mathcal{U}$ and a state $\mathbf{v} \in \mathit{Val}(\mathcal{V})$, a state trajectory from $\mathbf{v}$ with η is a function $\xi_{\mathbf{v},\eta} : [0,T] \to \mathit{Val}(\mathcal{V})$ satisfying: (a) $\xi_{\mathbf{v},\eta}(0) = \mathbf{v}$, (b) for any $t \in [0,T]$, the time derivative of $\xi \downarrow \mathcal{X}$ at t satisfies the differential equation $\frac{d(\xi\downarrow\mathcal{X})(t)}{dt} = f_\ell((\xi \downarrow \mathcal{X})(t), \eta(t))$, and (c) $(\xi \downarrow \mathcal{X})(t) \in I_\ell$ and $(\xi \downarrow \mathcal{L})(t) = \ell$. As in the last two statements, we will drop the subscripts of a trajectory when the dependence on the initial state and the input is clear. Because of the invariant I_ℓ, in some location $\ell \in \mathit{Val}(\mathcal{L})$ all the trajectories might be of finite duration. Conditions (i) and (ii) in Definition 1 make the HA *input enabled*, that is, from any state $\mathcal{A}$ is able to consume any input η completely (i) or up to some time at which it reacts with a transition (ii).

A HA without inputs ($\mathcal{U} = \varnothing$) is *closed*; otherwise, it is *open*. A HA with a single location and no transitions is called a *dynamical system*. We denote the components of HA $\mathcal{A}$ by $\mathcal{L}_\mathcal{A}, \mathcal{X}_\mathcal{A}, \mathcal{U}_\mathcal{A}, \Theta_\mathcal{A}, \mathcal{D}_\mathcal{A}, \to_\mathcal{A}$ and $\mathcal{T}_\mathcal{A}$, and for $\mathcal{A}_i$ its components are denoted by $\mathcal{L}_i, \mathcal{X}_i, \mathcal{U}_i, \Theta_i, \mathcal{D}_i, \to_i$ and $\mathcal{T}_i$.

Semantics. We assume that the discrete transitions are urgent and deterministic. That is, for any state $\mathbf{v} = (\mathbf{x}, \ell)$ at most one of following two cases are possible: (a) a transition to unique other state $(\mathbf{x}', \ell')$ is enabled, or (b) there is a trajectory $\xi_\mathbf{v}$ of non-zero duration. A bounded execution of $\mathcal{A}$ records the evolution of the variables along a particular run. A *bounded execution fragment* is a finite sequence of trajectories $\xi_{(0)}, \xi_{(1)}, \ldots$, such that, for each i, $\xi_{(i)} \in \mathcal{T}$ and $\xi_{(i)}.\mathsf{lstate} \to \xi_{(i+1)}.\mathsf{fstate}$. A *bounded execution* is an execution fragment with $\xi_{(0)}.\mathsf{fstate}$ in Θ. A state $\mathbf{v}$ is *reachable* if it is the last state of some execution. We denote the set of reachable states of $\mathcal{A}$ by $\mathsf{Reach}_\mathcal{A}$. The reachable states up to a bounded time horizon $T > 0$ are denoted by $\mathsf{Reach}_\mathcal{A}(T)$. The reachable states from a subset of initial states $\Theta' \subseteq \Theta$ up to T are denoted by $\mathsf{Reach}_\mathcal{A}(\Theta', T)$. A set $\mathit{Inv} \subseteq \mathit{Val}(\mathcal{V})$ is an *invariant* of a closed HA $\mathcal{A}$ if $\mathsf{Reach}_\mathcal{A} \subseteq \mathit{Inv}$. Checking invariants corresponds to verifying safety properties. Computing Reach_A exactly is undecidable but for the simplest classes of hybrid automata [1,24,33,47].

For relative completeness, we define robustness of HA. A HA $\mathcal{A}'$ is a *c-perturbation* of $\mathcal{A}$ if $\mathcal{A}'$ is obtained by perturbing the initial set and dynamical mappings of $\mathcal{A}$ by at most c. That is, $\mathcal{A}$ and $\mathcal{A}'$ are identical everywhere except that (i) $d(\Theta_\mathcal{A}, \Theta_{\mathcal{A}'}) \leq c$ where $d(\cdot,\cdot)$ is the Hausdorff distance and (ii) for every location ℓ and any continuous state $\mathbf{x} \in \mathit{Val}(\mathcal{X})$, the dynamical mappings of the two HA satisfy $|f_{\ell,\mathcal{A}}(\mathbf{x}) - f_{\ell,\mathcal{A}'}(\mathbf{x})| \leq c$. The *c-perturbed reach set* of $\mathcal{A}$, denoted by c-$\mathsf{Reach}_\mathcal{A}$, is the set of states reachable by some c-perturbation of $\mathcal{A}$. For a time bound $T > 0$, *Inv* is a *robust invariant up to time* T if there exists a positive constant $c > 0$ such that c-$\mathsf{Reach}_\mathcal{A}(T) \subseteq \mathit{Inv}$. In this paper we will present semi-decision procedures for bounded-time robust invariant checking of networks of deterministic nonlinear HA.

Composition. Large and complex models can be created by *composing* smaller automata. The composition operation identifies ("plugs-in") the input variables

of one automaton $\mathcal{A}_i$ with the state variables of another automaton[3]. A pair of HAs $\mathcal{A}_1$ and $\mathcal{A}_2$ are *compatible* if their state variables are disjoint $\mathcal{V}_1 \cap \mathcal{V}_2 = \varnothing$.

Definition 2. *Given a pair of compatible HAs $\mathcal{A}_1$ and $\mathcal{A}_2$ the* composed automaton $\mathcal{A} = \mathcal{A}_1 \| \mathcal{A}_2$ *is $\langle \mathcal{L}, \mathcal{X}, \mathcal{U}, \Theta, \mathcal{D}, \mathcal{T} \rangle$, where (a) $\mathcal{L} := \mathcal{L}_1 \cup \mathcal{L}_2$, (b) $\mathcal{X} := \mathcal{X}_1 \cup \mathcal{X}_2$, (c) $\Theta = \Theta_1 \times \Theta_2$, (d) $\mathcal{U} = U_1 \cup U_2 \setminus (\mathcal{V}_1 \cup \mathcal{V}_2)$, (e) $\mathcal{D}$: $\mathbf{v} \to \mathbf{v}'$ iff either $\mathbf{v}.\mathcal{V}_1 \to_1 \mathbf{v}'.\mathcal{V}_1$ and $\mathbf{v}.\mathcal{V}_2 = \mathbf{v}'.\mathcal{V}_2$, or $\mathbf{v}.\mathcal{V}_2 \to_2 \mathbf{v}'.\mathcal{V}_2$ and $\mathbf{v}.\mathcal{V}_1 = \mathbf{v}'.\mathcal{V}_1$, and (f) A trajectory ξ of $\mathcal{V} \cup \mathcal{U}$ is in $\mathcal{T}$ iff $\xi \downarrow (\mathcal{V}_i \cup \mathcal{U}_i) \in \mathcal{T}_i$ for each $i \in \{1, 2\}$.*

Note that the composition of two or more HA will define a *network*. $\mathcal{A}$ satisfies the requirements for Definition 1 and can be constructed by syntactically combining the guards, resets, and ODEs of its components.

Example 1. In the 2-dimensional FitzHugh-Nagumo (FHN) cardiac cell network, the i^{th} cell automaton $\mathcal{A}_i$ has a single location, two continuous variables $\mathcal{X} = \{x_{i1}, x_{i2}\}$ corresponding to fast and slow currents, and inputs (u_{i1}, u_{i2}), corresponding to diffusion from neighboring cells, and u_{i3}, a stimulus. The evolution is given by the ODEs (dynamic mapping): $\dot{x}_{i1} = (a - x_{i1})(x_{i1} - 1)x_{i1} - x_{i2} + u_{i3} + \frac{D}{h^2}(u_{i1} + u_{i2} - 2x_{i1})$, $\dot{x}_{i2} = \epsilon(\beta x_{i1} - \gamma x_{i2} - \delta)$, where $a, \beta, \delta, \gamma, \epsilon$ are parameters of the cell, the u_{i3} term models direct stimulus input, and the $\frac{D}{h^2}(.)$ term models the effect of the diffusion coupling with neighboring cells. In Figure 1, three FHN cells $\mathcal{A}_1, \mathcal{A}_2$ and $\mathcal{A}_3$ are interconnected in a ring and with a pulse generator. In each cycle, the pulse is activated for S_{on} time and stays off for S_{off} time. The composed system is defined by identifying input variables of one automaton with the state variables of another. For example, $u_{11} = x_{21}, u_{12} = x_{31}$ defines the part of the ring where $\mathcal{A}_1$ gets diffused current inputs from its neighbors and and $u_{13} = st$ connects the output of the pulse generator to $\mathcal{A}_1$.

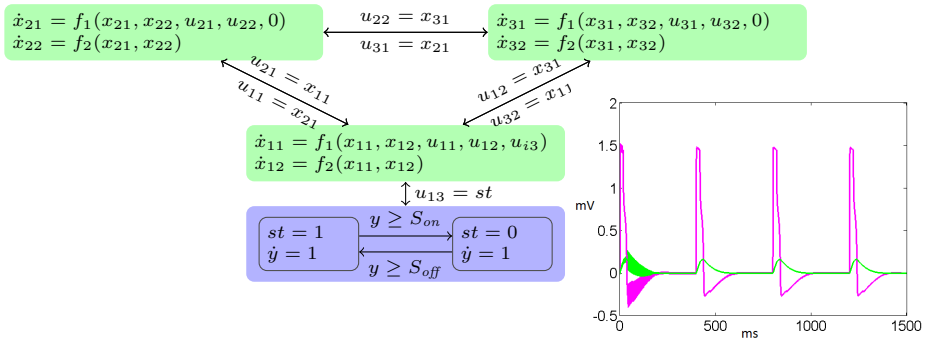

Fig. 1: Ring of 3 FHN-modules with a simple pulse generator. Reach set from a set of initial states projected on x_{11} and x_{12}.

[3] We do not allow HA to interact via transition synchronization as in [32, 38].

3 Annotations for Modules in a Network

We proposed simulation-based robust invariant verification of dynamical and switched systems in [16]. The approach requires the designers to provide special annotations called *discrepancy functions* for each location of the automaton. The algorithm first computes a validated numerical simulation from an initial state, say $\mathbf{v}$, and then bloats the simulation using the discrepancy function to compute arbitrarily precise over-approximations of $\mathsf{Reach}_{\mathcal{A}}(B_\delta(\mathbf{v}), T)$. Repeating this over a set of initial states $\mathbf{v}$ and with varying precision δ, one obtains a decision procedure for robust invariant checking. Towards our goal of verifying hybrid networks, in this section we present new techniques for computing discrepancy functions for such models.

3.1 IS Discrepancy and Approximations

First of all, we will use the definition of Input-to-State (IS) discrepancy function [27], which enables us to use annotations for individual modules in a dynamical system to then check invariants of the composed system. The IS discrepancy function for a location ℓ of $\mathcal{A}$ (or for a dynamical system) bounds the distance between two trajectories in location ℓ from different initial states, as a function of time and the inputs they receive.

Definition 3. *For a HA $\mathcal{A} = (\mathcal{L}, \mathcal{X}, \mathcal{U}, \Theta, \mathcal{D}, \mathcal{T})$, a continuous function $V : Val(\mathcal{X})^2 \to \mathbb{R}_{\geq 0}$ is an* input-to-state discrepancy function *for a location ℓ if*

(a) $\exists$ class-$\mathcal{K}$ functions (see [27]) $\underline{\alpha}, \overline{\alpha}$, s.t., $\forall\ \mathbf{x}, \mathbf{x}' \in Val(\mathcal{X})$, $\underline{\alpha}(|\mathbf{x} - \mathbf{x}'|) \leq V(\mathbf{x}, \mathbf{x}') \leq \overline{\alpha}(|\mathbf{x} - \mathbf{x}'|)$, and

(b) $\exists \beta : \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ and $\gamma \in \mathcal{K}$ such that for any $\mathbf{x}, \mathbf{x}'$, any pair of input trajectories $u, u' : \mathcal{U}$, and any $t \in \mathbb{R}_{\geq 0}$,

$$V(\xi_{\mathbf{x},\ell,u}(t), \xi_{\mathbf{x}',\ell,u'}(t)) \leq \beta(|\mathbf{x} - \mathbf{x}'|, t) + \int_0^t \gamma(|u(s) - u'(s)|)ds.$$

In addition, $\beta(\cdot, \cdot)$ is of class-$\mathcal{K}$ in the first argument and $\beta(\cdot, 0) = \overline{\alpha}(\cdot)$.

Here $\xi_{\mathbf{x},\ell,u}$ denotes the trajectory of the continuous variables $\mathcal{X}$ in location ℓ from state $\mathbf{x}$ and with the input trajectory u. The tuple $(\underline{\alpha}, \overline{\alpha}, \beta, \gamma)$ is called the *witness* of the discrepancy function V. The first condition merely bounds V in terms of the norm of its arguments. The more important second condition ensures that the distance between the trajectories is bounded as a function of β and γ, and can be reduced arbitrarily by making $\mathbf{x} \to \mathbf{x}'$ and $u \to u'$. IS discrepancy is related to integral input-to-state stability [2–4, 45]. However, for our verification algorithms, we do not require neighboring trajectories to converge over time. Using the IS discrepancy functions along with their witnesses, we construct a reduced order model M which can be employed to compute precise over-approximations of $\mathsf{Reach}_A(T)$. Given a dynamical system (HA with one location) $\mathcal{A} = \mathcal{A}_1 \| \mathcal{A}_2$ connected in a ring and IS discrepancy with witnesses for each of the modules, the IS approximation of $\mathcal{A}$ is a $(2+1)$-dimensional closed deterministic dynamical system M defined as follows.

Definition 4. *For a pair of nonnegative constants* (δ_1, δ_2)*, the* (δ_1, δ_2)-IS approximation *of* $\mathcal{A}$ *is a closed dynamical system with three variables* $\mathcal{X} = \{m_1, m_2, clk\}$ *initialized to* $\{\beta_1(\delta_1, 0), \beta_2(\delta_2, 0), 0\}$*, and dynamics* $\dot{\mathbf{x}} = f_M(\mathbf{x})$*, where*

$$f_M(\mathbf{x}) = \begin{bmatrix} \dot{\beta}_1(\delta_1, \mathbf{x}(clk)) + \gamma_1 \circ \underline{\alpha}_2^{-1}(\mathbf{x}(m_2)) \\ \dot{\beta}_2(\delta_2, \mathbf{x}(clk)) + \gamma_2 \circ \underline{\alpha}_1^{-1}(\mathbf{x}(m_1)) \\ 1 \end{bmatrix}. \tag{1}$$

The variable *clk* tracks the real time, and both the initial state and the dynamics of M depend on the choice of the parameters δ_1 and δ_2. It can be shown that the valuations of m_i along μ (the trajectory of M) give an upperbound on the distance between any trajectories of $\mathcal{A}_i$ that start from initial states and are at most δ_i apart. The following theorem establishes that the reach set of $\mathcal{A}$ from a set of states can be precisely over-approximated by bloating an individual execution ξ of $\mathcal{A}$ by a factor that is entirely determined by (a) a pair $V = (V_1, V_2)$ of IS discrepancy functions of $\mathcal{A}_1$ and $\mathcal{A}_2$ along with their witnesses, and (b) the trajectory μ.

Theorem 1 (Theorems 5.4 and 5.7 from [27]). *Let* $\xi_{\mathbf{v}}$ *be a trajectory of* $\mathcal{A}$*. For any nonnegative pair* $\delta = (\delta_1, \delta_2)$*, and any time* $T \geq 0$*, suppose* μ *is the trajectory of the* (δ_1, δ_2)*-IS approximation* M*. Then* $\mathsf{Reach}_{\mathcal{A}}(B_\delta(\mathbf{v}), T) \subseteq \bigcup_{t\in[0,T]} B^V_{\mu(t)}(\xi_{\mathbf{v}}(t))$*. Further, for any* $\epsilon > 0$ *and* $T > 0$*,* $\exists\ \delta_1, \delta_2 > 0$ *such that, for the* (δ_1, δ_2)*-IS approximation* M*,* $\bigcup_{t\in[0,T]} B^V_{\mu(t)}(\xi_{\mathbf{v}}(t)) \subseteq \epsilon\text{-}\mathsf{Reach}_{\mathcal{A}}(B_\delta(\mathbf{v}), T))$.

The precision of the over-approximation can be improved by reducing the parameters δ_1 and δ_2, and thus creating a finer covering of the initial set $\Theta_{\mathcal{A}}$. The result is generalized to dynamical systems with N modules connected in general network topologies [27], where the IS approximation is $(N+1)$-dimensional.

For a hybrid system $\mathcal{A} = \mathcal{A}_1 \| \mathcal{A}_2$, instead of providing annotations for the total of $|\mathit{Val}(\mathcal{L}_1)| \times |\mathit{Val}(\mathcal{L}_2)|$ locations of $\mathcal{A}$, the user has to provide IS discrepancy functions for location of $\mathcal{A}_1$ and $\mathcal{A}_2$. Then we can automatically construct $|\mathit{Val}(\mathcal{L}_1)| \times |\mathit{Val}(\mathcal{L}_2)|$ IS approximations corresponding to each possible location-pair. For cardiac cell networks, where all the automata modules are identical, this means working with $|\mathit{Val}(\mathcal{L}_1)|$ IS discrepancy functions. Next we present a new technique for computing such annotations in Proposition 1. The proof appears in Appendix B.

Proposition 1. *For a dynamical system with linear input* $\dot{\mathbf{x}} = f(\mathbf{x}) + B\mathbf{u}$*, where* B *is a matrix,* $V(\mathbf{x}_1, \mathbf{x}_2) = |\mathbf{x}_1 - \mathbf{x}_2|$ *is an IS discrepancy function with*

$$V(\xi_1(t), \xi_2(t)) \leq e^{\lambda_{max} t}|\mathbf{x}_1 - \mathbf{x}_2| + \int_0^t M|B||(\upsilon_1(\tau) - \upsilon_2(\tau))|d\tau,$$

where λ_{max} *is the largest eigenvalue of the Jacobian matrix* $J = \frac{1}{2}(\frac{\partial^T}{\partial \mathbf{x}} f(\mathbf{x}) + \frac{\partial}{\partial \mathbf{x}} f(\mathbf{x}) + I)$*,* $M = \sup_{s\in[0,t]} e^{\lambda_{max} s}$ *is the supremum of an exponential function of* λ_{max}*, and* ξ_i *is the state trajectory from* $\mathbf{x}_i$ *with input trajectory* υ_i*. Specifically, for a linear time invariant system* $\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}$*,* λ_{max} *is the largest eigenvalue of the matrix* A*, and* $M = \sup_{s\in[0,t]} |e^{As}|$.

For linear dynamical systems, we use the special case to obtain tight IS discrepancy functions by solving Linear Matrix Inequalities. The more general case establishes IS discrepancy functions for a larger class of non-linear systems for which the Jacobian matrix has bounded eigenvalues. For the nonlinear dynamic maps in this paper, computing the maximum eigenvalue of the Jacobian is solved using the MATLAB optimization toolbox or by a sum of squares solver [44].

4 Checking Bounded Invariants of HA Networks

First we define simulations for hybrid automata, and then we describe the verification algorithm that uses simulations and IS-discrepancy functions.

4.1 Simulations of Dynamical Systems

For a closed dynamical system $\mathcal{A}$ with an initial state $\mathbf{v}$, validated ODE solvers [11, 13, 40] can compute a sequence of sets $R_0, \ldots, R_l \subseteq \mathit{Val}(\mathcal{X})$ such that the trajectory $\xi_{\mathbf{v}}$ of $\mathcal{A}$ is contained in R_k over the interval $[(k-1)\tau, k\tau]$, where τ is the simulation time-step. We formalize this as follows:

Definition 5. *Consider a deterministic closed HA $\mathcal{A}$, an initial state $\mathbf{v}$, an error bound $\epsilon > 0$, and time step $\tau > 0$. Let the location of $\mathbf{v}.\mathcal{L}$ be ℓ and $\xi_{\mathbf{v}}$ be the execution of $\mathcal{A}$ starting from $\mathbf{v}$. A* $(\mathbf{v}, \epsilon, \tau)$-simulation fragment *is a finite sequence $\rho = (R_0, t_0), \ldots, (R_l, t_l)$ where, for each $k \in \{0, \ldots, l\}$, (a) $0 < t_k - t_{k-1} \leq \tau$, (b) R_k is contained in the invariant I_ℓ except possibly the last R_l, (c) $dia(R_k) \leq \epsilon$, and (d) for any time $t \in [t_{k-1}, t_k]$, $\xi_{\mathbf{v}}(t).\mathcal{X} \in R_k$.*

For relative completeness of verification, we will require that for a desired error bound $\epsilon > 0$ the diameter of R_k can be made smaller than ϵ by reducing the step size τ. A simulation for a HA is a sequence of simulation fragments (for different locations) that captures all the transitions of at least one execution.

Definition 6. *Consider a HA $\mathcal{A}$, an initial state $\mathbf{v}$, an error bound $\epsilon > 0$, a time bound $T > 0$, a transition bound l, and a time step $\tau > 0$. Let $\xi_{\mathbf{v}}$ be the execution from $\mathbf{v}$ with $\xi_{\mathbf{v}}.dur \leq T$, where $\xi_{\mathbf{v}}.dur$ is the time duration of the trajectory $\xi_{\mathbf{v}}$, and with l transitions at times $\sigma_1, \ldots, \sigma_l \in \mathbb{R}_{\geq 0}$; let $\sigma_0 = 0$. A* $(\mathbf{v}, \epsilon, \tau, T, l)$-simulation *is a finite sequence $\psi = \rho_0, \ldots, \rho_l$ where (a) each $\rho_k = (R_{k(1)}, t_{k(1)}), \ldots, (R_{k(m_k)}, t_{k(m_k)})$ is a $(\xi(\sigma_k), \epsilon, \tau)$-simulation fragment with m_k samples, (b) $t_0 = 0$, $t_{l(m_l)-1} = T$, and for each $k > 0$, $t_{k(m_k)} \geq t_{(k+1)(1)}$, and (c) $\sigma_k \in [t_{(k+1)(1)}, t_{k(m_k)}]$.*

A $(\mathbf{v}, \epsilon, \tau, T, l)$-simulation ψ is a sequence of l simulation fragments where each fragment ρ_k has m_k elements with indices $k(1), \ldots, k(m_k)$. The k^{th} transition on the actual execution $\xi_{\mathbf{v}}$ has to occur between the last sample period of ρ_{k-1} and the first sample interval of ρ_k (condition (c)). In addition, ρ_k is a $(\xi_{\mathbf{v}}(\sigma_k), \epsilon, \tau)$-simulation fragment, that is, it *contains* the trajectory of $\mathcal{A}$ starting from the post state $\xi_{\mathbf{v}}(\sigma_k)$ of the k^{th} transition. In Algorithm 2, the subroutine *Simulate* computes a simulation of HA of the above type. The simulation ψ represents

other executions that start near $\mathbf{v}$. Formally, we say an execution fragment ξ is *captured* by ψ if duration of ξ is at most T, ξ experiences exactly the same sequence of locations as recorded in some prefix of ψ, and its $k^{(th)}$ transition occurs in the intervals $[t_{k(m_k)}, t_{(k+1)(1)}]$.

4.2 Verification Algorithm

We sketch the key ideas that enable the checking of bounded-time invariants of closed networks of hybrid automata. The main inputs of *InvVerify* (Algorithm 1) are the specification of the composed automaton $\mathcal{A} = \mathcal{A}_1 \| \ldots \| \mathcal{A}_N$, the open unsafe set $\mathbb{U}$, and the collection of discrepancy functions and witnesses *ISD* for every location of each subsystem. The variable $\mathcal{C}$ (line 2) is initialized to a

Algorithm 1: *InvVerify*$(\mathcal{A}, ISD, \mathbb{U}, T, \epsilon_0, \delta_0, n_0)$: Verifies invariants of hybrid networks.

```
1  R ← ∅; δ ← δ0; ε ← ε0; τ ← τ0; n ← n0;
2  C ← {(v, δ, ε, τ) | {v} is a δ-Cover(Θ)};
3  while C ≠ ∅ for each (v, δ, ε, τ) ∈ C do
4  |   (flag, S) ← ReachFromCover(A, v, δ, ε, τ, T, ISD, n);
5  |   switch flag do
6  |   |   case SAFE: C ← C\{(v, δ, ε, τ)}; R ← R ∪ S ;
7  |   |   case UNSAFE: return (UNSAFE, R) ;
8  |   |   case REFINE:
9  |   |   |   C ← C\{(v, δ, ε, τ)}; δ ← δ/2; ε ← ε/2; τ ← τ/2; n ← 2n;
10 |   |   |   C ← C ∪ {(v, δ, ε, τ) | {v} is a δ−Cover(Θ ∩ Bδ(v))};
11 |   |   end
12 |   end
13 end
14 return (SAFE, R);
```

collection of tuples $\{(\mathbf{v}_k, \delta, \epsilon, \tau)\}_{k \in |\mathcal{C}|}$, such that $\{\mathbf{v}_k\}$ is a δ-cover of Θ, that is, $\Theta \subseteq \cup_{k \in |\mathcal{C}|} B_\delta(\mathbf{v}_k)$, and (δ, ϵ, τ) are parameters. For each $(\mathbf{v}, \delta, \epsilon, \tau)$ in $\mathcal{C}$, the subroutine *ReachFromCover* (Algorithm 2) computes *flag* and a set S. The *flag* is set to SAFE if all executions from $B_\delta(\mathbf{v})$ are disjoint from $\mathbb{U}$ up to time T and in that case $\mathbf{v}$ is removed from $\mathcal{C}$. The *flag* is set to UNSAFE if at least one execution reaches $\mathbb{U}$, and in that case *InvVerify* returns UNSAFE and $\mathcal{R}$. Finally, if the *flag* is set to REFINE then $\mathbf{v}$ is replaced by a finer cover of $\Theta \cap B_\delta(\mathbf{v})$. In addition to having $\delta/2$-radius balls covering $B_\delta(\mathbf{v})$, the parameters ϵ and τ are also halved to compute more precise over-approximations. The sets S and $\mathcal{R}$ compute over-approximations of $\mathsf{Reach}_{\mathcal{A}}(B_\delta(\mathbf{v}), T)$ and $\mathsf{Reach}_{\mathcal{A}}(T)$, respectively.

ReachFromCover checks safety with respect to $\mathbb{U}$ of the states reachable from $B_\delta(\mathbf{v})$ up to time T and with at most $n > 0$ transitions. First, it computes an over-approximation ($\mathcal{R}$) of $\mathsf{Reach}_A(B_\delta(\mathbf{v}), T)$ with certain precision (determined by the parameters δ, ϵ, and τ). If this over-approximation is sufficient

Algorithm 2: *ReachFromCover*$(\mathcal{A}, ISD, \mathbb{U}, \mathbf{v}, \delta, \epsilon, \tau, T, n)$: Over-approx $\mathsf{Reach}_{\mathcal{A}}(B_\delta(\mathbf{v}))$.

1 $\mathcal{R} \leftarrow \varnothing$; $\mathcal{C} \leftarrow \{(\mathbf{v}, 0)\}$; *count* $\leftarrow 0$;
2 **while** $\mathcal{C} \neq \varnothing$ **for each** $(\mathbf{v}, t_0) \in \mathcal{C}$ **do**
3 $\quad r \leftarrow \delta$; $\psi \leftarrow$ *Simulate*$(\mathcal{A}, \mathbf{v}, T - t_0, \epsilon, \tau)$; *count* $\leftarrow$ *count* $+ 1$;
4 $\quad$ **for** $k = 0 : l$, **where** $\psi = \rho_0, \rho_1, \ldots, \rho_l$ **do**
5 $\quad\quad (S_k, r) \leftarrow$ *BloatWithISD*$(\rho_k, ISD, r, \epsilon, \tau, \mathcal{A})$;
6 $\quad$ **end**
7 $\quad$ **if** *a transition* (ℓ, ℓ') *is enabled from* S_j *but is not captured by* ψ **then**
8 $\quad\quad \mathcal{C} \leftarrow \mathcal{C} \cup \{(\mathbf{v}, t_0) \mid \{\mathbf{v}\}$ is the δ-*Cover*$(R_{\ell,\ell'}(S_k \cap G_{\ell,\ell'}))$,
9 $\quad\quad\quad t_0$ is the first time(ℓ, ℓ') is enabled$\}$;
10 $\quad$ **if** *a transition is captured by* ψ *but is not enabled for a subset* $S'_k \subseteq S_k$ **then**
11 $\quad\quad \mathcal{C} \leftarrow \mathcal{C} \cup \{(\mathbf{v}, t_0) \mid \{\mathbf{v}\}$ is the δ-*Cover*(S'_k),
12 $\quad\quad\quad t_0$ is the first time the transition is captured$\}$;
13 $\quad$ **end**
14 $\quad$ **if** $(\cup_j s_j \cap \mathbb{U} = \varnothing) \wedge (\mathit{count} < n)$ **then** $\mathcal{R} \leftarrow \mathcal{R} \cup (\cup_j s_j)$; $\mathcal{C} \leftarrow \mathcal{C} \backslash \{(\Theta, t_0)\}$;
15 $\quad$ **else if** $(\exists\ R_j \subseteq \mathbb{U})\ \wedge\ (\mathit{count} = 1)$ **then return** (UNSAFE, $\mathcal{R}$) ;
16 $\quad$ **else return** (REFINE, $\mathcal{R}$) ;
17 **end**
18 **return** (SAFE, $\mathcal{R}$);

to prove/disprove safety with respect to $\mathbb{U}$ then it sets the *flag* to SAFE or UNSAFE, and otherwise it returns REFINE. If it detects that more than n transitions are possible within time T, then also it returns REFINE.

In computing $\mathcal{R}$ in *ReachFromCover*, the set $\mathcal{C}$ stores a set of state-time pairs that are yet to be processed. If $(\mathbf{v}, t_0) \in \mathcal{C}$ then $\mathsf{Reach}_{\mathcal{A}}(B_\delta(\mathbf{v}), T - t_0)$ is yet to be evaluated and added to $\mathcal{R}$. For each $(\mathbf{v}, t_0)$ in the cover $\mathcal{C}$, a $(\mathbf{v}, \epsilon, \tau, T, l)$-simulation $\psi = \rho_0, \ldots \rho_l$ is computed. The variable *count* tracks the number of new simulation branches initiated in a run of the algorithm. Let ξ be the actual execution starting from $\mathbf{v}$ and $G_{\ell,\ell'}$ be the guard from location ℓ to ℓ'. By Definition 6, each ρ_k of ψ is a simulation fragment. By Definition 4, the IS-approximation is a (small) dynamical system, whose trajectory gives an upper bound of the distance between continuous trajectories of $\mathcal{A}$. The subroutine *BloatWithISD*$(\rho_k, ISD, r, \epsilon, \tau, \mathcal{A})$ (i) creates an IS-Approximation M of $\mathcal{A}$ using the discrepancy functions in *ISD* that correspond to the location of ρ_k, (ii) generates a $(r, \epsilon, \tau, T, 0)$-simulation of M, say μ, (iii) bloats each set R_j in ρ_k with the valuation of $\mu(t_j)$ to obtain a set s_j, (iv) returns the sequence of sets $S_k = (s_{k(1)}, t_{k(1)}), \ldots, (s_{k(m)}, t_{k(m)})$, and finally (v) applies the transition between ρ_k and ρ_{k+1} on the set $s_{k(m)}$ and returns r as the radius of image of the reset function. From Theorem 1, S_k contains all continuous trajectories of $\mathcal{A}$ that start from $B_r(R_{k(1)})$. It can be checked that $\cup_j s_j$ precisely over-approximates all the executions from $B_r(\mathbf{v})$ that are captured by ψ (Proposition 2-3).

To over-approximate the states reached via executions from $B_\delta(\mathbf{v})$ that are *not* captured by ψ, the algorithm generates new simulations (line 7-13) and adds up *count*. The algorithm transverses S_k and generates and checks two possible

cases as described in line 7 and 10. Then the algorithm decides whether the computed over-approximation $\mathcal{R}$ is safe, unsafe, or needs further refinement.

4.3 Soundness and Relative Completeness

We will sketch the interesting parts of the correctness argument and the details of the proofs are given in the technical report [26]. In what follows, all the program variables refer to their valuations at the p^{th} iteration of the **while** loop of *ReachFromCover*, unless otherwise stated. That is, $(\mathbf{v}, t_0)$ is the time-state pair being explored in the p^{th} iteration. Propositions 2 and 3 follow from straightforward inductive application of Theorem 1, the fact that ψ is a simulation from $\mathbf{v}$ with the properties stated in Definition 6, and the continuity of the reset functions for all location pairs.

Proposition 2. *Let ψ be a simulation from $\mathbf{v}^{(p)}$. For any execution fragment ξ starting from a state in $B_\delta(\mathbf{v}^{(p)})$, if the transition sequence of ξ is captured by ψ then, for any $t \in [0, T - t_0^{(p)}]$, $\xi(t) \in \cup_{j=1}^{l(m_l)} s_j$. Recall that $l(m_l)$ denotes the last index of ρ_l and thus the total number of elements in ψ.*

Proposition 3. *For the execution $\xi_{\mathbf{v}}$ from $\mathbf{v}$, and any $r > 0$, there exists sufficiently small δ, ϵ, τ, such that $\cup_{j=1}^{l(m_l)} s_j \subseteq \cup_{t\in[0,T-t_0]} B_r(\xi_{\mathbf{v}}(t))$.*

Lemma 1. *If ReachFromCover returns* (SAFE, $\mathcal{R}$)*, then $\mathcal{R} \supseteq \mathsf{Reach}_{\mathcal{A}}(B_\delta(\mathbf{v}), T)$.*

The proofs of Lemma 2 can be found in Appendix B. In the poof, we show that every execution can be decomposed into execution segments such that each of such segments is captured by some simulation generated during the **while** loop of *ReachFromCover*. Thus, by combining the above observation with Proposition 2, we establish Lemma 2.

Lemma 2. *If Algorithm 2 returns (SAFE, $\mathcal{R}$), then all executions fragment ξ starts from $B_\delta(\mathbf{v})$ with duration $\xi.\mathsf{dur} \leq T$ are safe, moreover $\cup_{t\in[0,\xi.\,\mathsf{dur}]}\xi(t) \subseteq \mathcal{R}$.*

Theorem 2. *InvVerify is sound.*

Proof. Suppose *InvVerify* returns SAFE. Then, for any cover $(\mathbf{v}, \delta, \epsilon, \tau) \in \mathcal{C}$, *ReachFromCover* terminates and returns SAFE. This can happen only if all initial state-time pairs $(\mathbf{v}, t_0)$ are removed from $\mathcal{C}$. It follows from line 14 of Algorithm 2 that $\mathcal{R} \cap \mathbb{U} = \varnothing$. From Lemma 2, we have $\mathsf{Reach}_{\mathcal{A}}(T, B_\delta(\mathbf{v})) \cap \mathbb{U} = \varnothing$.

Now suppose the *InvVerify* returns UNSAFE. Then, for a cover $(\mathbf{v}, \delta, \epsilon, \tau) \in \mathcal{C}$, *ReachFromCover* terminates and returns UNSAFE. From line 15, *ReachFromCover* returns UNSAFE only if, in the simulation ψ computed in the first loop, there exists some R_j such that $R_j \subseteq \mathbb{U}$. Then, for the execution $\xi_{\mathbf{v}}$, and for $t \in [t_{j-1}, t_j]$, we have $\xi(t) \in R_j \subseteq \mathbb{U}$, and therefore $\mathcal{A}$ is unsafe. □

In this paper, we present a termination argument for *InvVerify* under the following robustness assumption.

Assumption 1. *(i) $\mathcal{A}$ has an average dwell time [25]. That is, there exists $N' \geq 0$ and $\tau' > 0$ such that, for any execution fragment ξ of $\mathcal{A}$, the number of transitions occurring in ξ is upperbounded by $N' + \frac{\xi.\mathsf{dur}}{\tau'}$. (ii) Either of the following two conditions holds: (a) $\bar{\mathbb{U}}$ is a robust invariant up to time T. (b) There exists $c > 0$ such that all c-perturbations of $\mathcal{A}$ are unsafe.*

Assumption 1(i) is standard for well-designed systems and can be automatically checked for certain model classes [39]. Part (ii) is a robustness condition with respect to the invariant $\bar{\mathbb{U}}$ (complement of $\mathbb{U}$) such that the satisfaction of the invariant remains unchanged under sufficiently small perturbations on the models.

Theorem 3. *InvVerify terminates and outputs the correct answer.*

Proof. Under Assumption 1(iia), we will show that, after sufficiently many refinements, for any cover $(\mathbf{v}, \delta, \epsilon)$ in the set $\mathcal{C}$, *ReachFromCover* returns SAFE. From Assumption 1(i), there exist finitely many different sequences of transitions for all the execution segments with duration no longer than T. Thus, line 9 and 12 in *ReachFromCover* can be executed for finitely many times such that the counter *count* is bounded. Therefore, for sufficiently large n, $count < n$ holds. For sufficiently small $\epsilon, \delta, \tau > 0$, the overapproximated reach set of an execution $\xi_{\mathbf{v}}$ can be computed arbitrarily precise (Proposition 3). From Assumption 1(iia), if $\xi_{\mathbf{v}}$ is safe, for sufficiently small perturbation c the c-perturbations are also safe. Thus, lines 14-16 are executed for every $\mathbf{v}$ after sufficiently many refinements. After all covers $(\mathbf{v}, \delta, \epsilon)$ in the set $\mathcal{C}$ are explored by *InvVerify*, it returns SAFE.

Otherwise, suppose Assumption 1(iib) holds. First of all, as $\mathcal{R}$ always contains the unsafe execution, denoted $\xi_{\mathbf{v}}$, SAFE is never returned. From Assumption 1(iib), there exists $c > 0$ such that all c-perturbations of $\mathcal{A}$ are unsafe. It follows that all the executions from $B_c(\mathbf{v})$ are unsafe since otherwise we can construct a c-perturbation $\mathcal{A}'$ that satisfies the invariant by setting $\Theta_{\mathcal{A}'} = \Theta_{\mathcal{A}} - B_c(\mathbf{v})$. After sufficiently many refinements, a state $\mathbf{v}' \in B_c(\mathbf{v}')$ will be explored by *ReachFromCover* with high enough precision. Then, on the $(\mathbf{v}', \delta, \epsilon, \tau, T, l)$-simulation ψ computed in the first while loop in *ReachFromCover*, a set R_j will be contained in the unsafe set. Then *ReachFromCover* returns UNSAFE, and therefore *InvVerify* returns UNSAFE.

5 Checking Invariants for Cardiac Cell Networks

We present a challenging case study modeling a cardiac cell that involves nonlinear HA networks. The purpose of the case study is to demonstrate the feasibility of the algorithms developed in the previous section. Note that we do not focus on the biological relevance of the model, and instead present experimental results showing the computation of the reach sets as well as the verification time. Our case study is the minimal ventricular (MV) model of Bueno-Orovio et al. [12] that generates action ptential (APs) on cardiac rings [18]. Unlike the FHN model, the MV model can reproduce realistic and important AP phenomena, e.g. alternans [23], and yet is computationally more efficient than some of

the other models in the literature. Using the techniques from Grosu et al. [20], we abstract the MV model into a network of multi-affine hybrid (MAH) automata (see Figure 2). On the resulting network we check a key invariant property.

5.1 The MAH cardiac cell network model

The MV model describes the flow of currents through a cell. The model is defined by four nonlinear PDEs representing the transmembrane potential $x_1(\boldsymbol{d}, t)$, the fast channel gate $x_2(\boldsymbol{d}, t)$, and two slow channel gates, $x_3(\boldsymbol{d}, t)$ and $x_4(\boldsymbol{d}, t)$. All of the four variables are time and position $\boldsymbol{d} := (d_x, d_y, d_z) \in \mathbb{R}^3$ dependent. For one dimensional tissue, i.e., $\boldsymbol{d} := d_x$, the evolution of transmembrane potential is given by:

$$\frac{\partial x_1(d_x, t)}{\partial t} = D\frac{\partial^2 x_1(d_x, t)}{\partial d_x^2} + e(x_1, t) - (J_{\text{fi}} + J_{\text{so}} + J_{\text{si}}), \tag{2}$$

where $D \in \mathbb{R}$ is the diffusion coefficient, $e(\boldsymbol{d}, t)$ is the external stimulus applied to the cell, J_{fi} is the fast inward current, J_{si} is the slow inward current and J_{so} is the slow outward current. The currents J_{fi}, J_{so} and J_{si} are described by Heaviside function. For more details see Appendix C. To define the propagation of the action potential on a cardiac ring of length L, we set the boundary conditions to: $x_i(0, t) = x_i(L, t)$ for all $i \in \{0, \ldots, 4\}$ and $t \in \mathbb{R}$.

MAH approximation. One alternative to solving these highly nonlinear PDEs is to discretize space and hybridize the dynamics. The result is the MAH model. Following the approach of [20] we first hybridize the dynamics and obtain a HA with 29 locations. The basic idea is to approximate the Heaviside function from J_{fi}, J_{so} and J_{si} with a sequence of ramp functions. Each location of the resulting HA contains a multi-affine ODE such as:

$$\dot{x_1} = -0.935x_1 + 12.70x_2 - 8.0193x_1x_2 + 0.529x_3x_4 + 0.87 + st$$
$$\dot{x_2} = -0.689x_2;\ \dot{x_3} = -0.0025x_3;\ \dot{x_4} = 0.0293x_1 - 0.0625x_4 + 0.0142,$$

where st is the time-varying stimulus input. Urgent transitions from each location ℓ_i to the next (and predecessor) location ℓ_{i+1}, $i \in [29]$, are enabled by the guards of the form $x_1 \geq \theta_i'$ and $x_1 < \theta_i$, where θ_i, θ_i' are the constants arising from ramp approximations of the Heaviside functions.

Next, we discretize the 2nd order derivative $D\frac{\partial^2 x_1(d_x,t)}{\partial d_x^2}$ from Eq. 2 with a discretization step of Δ using 2nd order central difference method and obtain $D\frac{x_1(d_x+\Delta,t)-2x_1(d_x,t)+x_1(d_x-\Delta,t)}{\Delta^2}$. Informally, Δ represents the spatial discretization and corresponds to the length of the cell in the ring. This 2nd order central difference term is added to the right hand side of the dynamic mapping for x_1 (in each location) to obtain the final MAH model of a single cardiac cell. Note that by using the central difference method the approximation error for the original MV model is of the order $\mathcal{O}(\Delta^2)$. To check invariants of a cardiac ring of length L, we connect all of the $\lfloor \frac{L}{\Delta} \rfloor$ HA into a network such that the input variables of

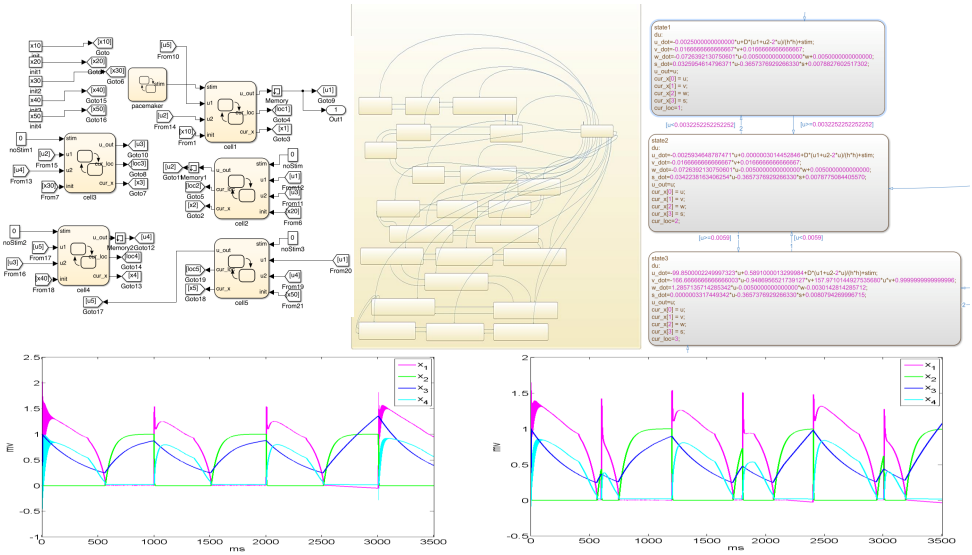

Fig. 2: Top left: top-level Simulink/Stateflow model for a ring of five MAH cells; the *Pacemaker* block stimulates one cell. Center: Stateflow model of a single MAH cell. Top right: dynamics and guards in 3 locations of a single cell. Bottom: reach set projected on x_{11} (AP) for stimulation period of 1000 msec (left) and 600 msec (right) with x-axis for time and y-axis for voltage.

every HA $\mathcal{A}_i$, $i \in \left[\lfloor \frac{L}{\Delta} \rfloor\right]$, are identified with the variable $x_{(i+1)1}$ of the successor $\mathcal{A}_{i+1}$ and the state variable $x_{(i-1)1}$ of $\mathcal{A}_{i-1}$ in the ring. We consider scenarios where one HA in the ring gets a sequence of stimuli from a pulse generator and for the remaining HA $st(t) := 0$.

5.2 Experimental Results

For understanding the effect of stimuli on cardiac tissue (cell networks) the key invariant properties of interest are of the form $x_1 \leq \theta_{max}$, where θ_{max} is a threshold voltage value. Other properties about timing of action potentials can be constructed using these building block invariants and additional timers. We implemented the algorithms of Section 4 in MATLAB programs that take as input Simulink/Stateflow models of FHM and MAH networks (see Figure 2). For all the locations the IS-discrepancy functions are computed using the techniques of Section 3. The cells being identical, essentially $Val(\mathcal{L})$ IS discrepancy functions are sufficient. The results presented here are based on experiments performed on a Intel Xeon V2 desktop computer using Simulink's ode45 simulation engine. Table 1(a) shows typical running times of our prototype on MAH networks of size 3, 5, and 8 cells with different invariant properties (defined by $x_1 \leq \theta_{max}$). These are for a time horizon (T) of 1200 ms with a stimulus of 5 ms exciting one of the cells every 600 ms. The uncertainty in the initial set is ± 0.0001 mV

N	θ_{max}	Sims	Refs	RT(s)	$x_{i1} \leq \theta_{max}$
3	2	16	0	104.8	✓
3	1.65	16	0	103.8	✓
3	1.55	17	1	110.6	✓
3	1.5	NA	NA	9.0	×
5	2	3	0	208.0	✓
5	1.65	5	1	281.6	✓
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	×

(a)

N	θ_{max}	Sims	Refs	RT	$x_{i1} \leq \theta_{max}$
3	1.5	1	0	1.5	✓
3	1.0	16	1	20.4	✓
5	1.5	8	2	9.2	✓
5	1.0	NA	NA	1.1	×
8	1.5	1	0	1.8	✓
8	1.0	24	3	33.5	✓

(b)

T	S-taliro	Our tool
100	24.2	3.1
1000	27.4	9.3
10000	55.5	62.9

(c)

Table 1: (a) Scaling with network size. N: number of cells in the ring network, θ_{max}: threshold voltage defining invariant, Sims: number of simulations, Refs: max. number of refinements, RT: running time in seconds. (b) Verification of FHM networks with time horizon T=1200 ms, initial set uncertainty ±0.01 mV. (c) Comparison of running time with S-taliro over 3 cell MAH networks for cases where both tools find counter-examples.

for each of the cells in the network (for comparison, the invariant ranges for the first few locations are 0.003 mV), except for the 2^{nd} network with 5 cells, where the initial set has higher uncertainty of ±0.0001 mV. With this larger initial set, even with the same threshold, the algorithm requires many more refinements and simulations to prove the invariant. Analogous results for 3, 5, and 8 cell FHN networks are shown in Table 1(b), with the longer time horizon $T = 10000$ ms and greater uncertainty in the initial set of ±0.01 mV. The two orders of magnitude faster running time (even for the same number of simulations) can be explained by the lower dimension (2) of FHN cells, and the absence of any transitions which spawn new branches in the execution of MAH -simulations. A comparable tool that can check for counter-examples in this class of HA models models is S-taliro [5]. We were able to find counter-examples using S-taliro for the 3 cell MAH networks with similar initial states (running times shown in Table 1(c)). On average, for smaller time horizons (T) S-taliro found counter-examples faster, but for longer T (and appropriate initial sets) the running times were comparable to our prototype.

It is known that electrical alternans initiate and destabilize reentrant waves which my induce cardiac arrhythmia such as ventricular fibrillation [28]. The electrical alternans involve long-short beat-to-beat alternation of AP duration at fast pacing rates. In Figure 2 (bottom left) we plot the reach set from a set of initial states with pacing rate of 1000 msec and observe that the AP durations do not change, whereas at a pacing rate of 600 msec (bottom right) the AP durations alternate. The reach set approximations computed by our tool enable

us to prove absence of alternans over bounded-time horizons and also to find initial states from which they may arise.

6 Related Work, Discussion and Conclusions

Networks of timed automata to model the propagation of APs in human heart are employed in the Virtual Heart Model [29–31, 42] and hybrid automata are used in [9, 48]. In [19, 34], the authors develop a model of the cardiac conduction system that addresses the stochastic behavior of the heart, validated via simulation. However, the hybrid behavior of the heart is not considered. Grosu et al. [22] carry out automated formal analysis of a realistic cardiac cell model. In [20] a method to learn and to detect the emergent behavior (i.e. the spiral formation) is proposed. Simulation-based analysis of general nonlinear HA has been investigated in [5], where a search for counter-examples is carried out using sampling and stochastic optimization. Our approach is designed to prove bounded-time invariants. Other promising tools include Breach [15] and *Flow** [14]; their application to cardiac cell networks will be an interesting direction to explore once support for these types of Simulink/Stateflow models is established.

In this paper, we present an algorithm to check robust bounded-time invariants for networks of nonlinear hybrid automata. We used automatically computed input-to-state discrepancy functions for individual locations of individual automata modules to over-approximate reachable states of the network. All of the developed techniques and the symmetry in the network of cells enabled us to check key invariants of networks of nonlinear cardiac cells, where each cell has four continuous variables and 29 locations. We will extend our algorithms to support richer classes of properties specified in metric or signal temporal logic. These results also suggest new strategies for pacemaker control algorithms, for example, for avoiding alternans and other undesirable behavior.

Acknowledgements. This work is supported by the ERC AdG VERIWARE, ERC PoC VERIPACE and the Institute for the Future of Computing, Oxford Martin School.

References

1. Rajeev Alur, Costas Courcoubetis, Nicolas Halbwachs, Thomas A. Henzinger, P.-H. Ho, Xavier Nicollin, Alfredo Olivero, Joseph Sifakis, and Sergio Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138(1):3–34, 1995.
2. D. Angeli. Further results on incremental input-to-state stability. *Automatic Control, IEEE Transactions on*, 54(6):1386–1391, 2009.
3. David Angeli. A lyapunov approach to incremental stability properties. *Automatic Control, IEEE Transactions on*, 47(3):410–421, 2002.
4. David Angeli, Eduardo D Sontag, and Yuan Wang. A characterization of integral input-to-state stability. *Automatic Control, IEEE Transactions on*, 45(6):1082–1097, 2000.
5. Yashwant Annapureddy, Che Liu, Georgios Fainekos, and Sriram Sankaranarayanan. S-taliro: A tool for temporal logic falsification for hybrid systems. In *TACAS*, 2011.
6. Vladimir Igorevich Arnol'd. *Mathematical methods of classical mechanics*, volume 60. Springer, 1989.
7. Nikolai Axmacher, Florian Mormann, Guillen Fernández, Christian E Elger, and Juergen Fell. Memory formation by neuronal synchronization. *Brain research reviews*, 52(1):170–182, 2006.
8. Alain Barrat, Marc Barthelemy, and Alessandro Vespignani. *Dynamical processes on complex networks*, volume 1. Cambridge University Press Cambridge, 2008.
9. Ezio Bartocci, Flavio Corradini, Maria Rita Di Berardini, Emilia Entcheva, Scott A. Smolka, and Radu Grosu. Modeling and simulation of cardiac tissue using hybrid I/O automata. *Theor. Comput. Sci.*, 410(33-34):3149–3165, 2009.
10. Stefano Boccaletti, Vito Latora, Yamir Moreno, Martin Chavez, and D-U Hwang. Complex networks: Structure and dynamics. *Physics reports*, 424(4):175–308, 2006.
11. Olivier Bouissou and Matthieu Martel. Grklib: a guaranteed runge kutta library. In *Scientific Computing, Computer Arithmetic and Validated Numerics, 2006. SCAN 2006. 12th GAMM-IMACS International Symposium on*, pages 8–8. IEEE, 2006.
12. Alfonso Bueno-Orovio, Elizabeth M. Cherry, and Flavio H. Fenton. Minimal model for human ventricular action potentials in tissue. *Journal of Theoretical Biology*, 253(3):544 – 560, 2008.
13. CAPD. Computer assisted proofs in dynamics, 2002.
14. Xin Chen, Erika brahm, and Sriram Sankaranarayanan. Flow*: An analyzer for non-linear hybrid systems. In Natasha Sharygina and Helmut Veith, editors, *Computer Aided Verification*, volume 8044 of *Lecture Notes in Computer Science*, pages 258–263. Springer Berlin Heidelberg, 2013.
15. Alexandre Donzé. Breach, a toolbox for verification and parameter synthesis of hybrid systems. In *C.A.V.* 2010.
16. Parasara Sridhar Duggirala, Sayan Mitra, and Mahesh Viswanathan. Verification of annotated models from executions. In *EMSOFT*, 2013.
17. Flavio Fenton and Alain Karma. Vortex dynamics in three-dimensional continuous myocardium with fiber rotation: filament instability and fibrillation. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 8(1):20–47, 1998.
18. Alejandro Garzón, Roman O. Grigoriev, and Flavio H. Fenton. Model-based control of cardiac alternans on a ring. *Physical Review E 80*, 2009.
19. Saul Greenhut, Janice Jenkins, and Robert MacDonald. A stochastic network model of the interaction between cardiac rhythm and artificial pacemaker. *IEEE Transactions on Biomedical Engineering*, 40(9):845–858, 1993.

20. Radu Grosu, Grégory Batt, Flavio H. Fenton, James Glimm, Colas Le Guernic, Scott A. Smolka, and Ezio Bartocci. From cardiac cells to genetic regulatory networks. In *CAV*, pages 396–411, 2011.
21. Radu Grosu, Gregory Batt, FlavioH. Fenton, James Glimm, Colas Guernic, ScottA. Smolka, and Ezio Bartocci. From cardiac cells to genetic regulatory networks. In Ganesh Gopalakrishnan and Shaz Qadeer, editors, *Computer Aided Verification*, volume 6806 of *Lecture Notes in Computer Science*, pages 396–411. Springer Berlin Heidelberg, 2011.
22. Radu Grosu, Scott A. Smolka, Flavio Corradini, Anita Wasilewska, Emilia Entcheva, and Ezio Bartocci. Learning and detecting emergent behavior in networks of cardiac myocytes. *Commun. ACM*, 52(3):97–105, 2009.
23. M. R. Guevara, G. Ward, A. Shrier, and L. Glass. Electrical alternans and period-doubling bifurcations. *Computers in Cardiology*, pages 167–170, 1984.
24. Thomas A. Henzinger, Peter W. Kopke, Anuj Puri, and Pravin Varaiya. What's decidable about hybrid automata? In *ACM Symposium on Theory of Computing*, pages 373–382, 1995.
25. J.P. Hespanha and A. Morse. Stability of switched systems with average dwell-time. In *Proceedings of 38th IEEE Conference on Decision and Control*, pages 2655–2660, 1999.
26. Zhenqi Huang, Chuchu Fan, Sayan Mitra, Alexandru Mereacre, and Marta Kwiatkowska. Invariant verification of nonlinear hybrid automata networks of cardiac cells, 2014. Online supporting material: models, code, and data. https://wiki.cites.illinois.edu/wiki/display/MitraResearch/Complex+Networks+of+Nonlinear+Modules.
27. Zhenqi Huang and Sayan Mitra. Proofs from simulations and modular annotations. In *In 17th International Conference on Hybrid Systems: Computation and Control*, Berlin, Germany. ACM press.
28. Raymond E. Ideker and Jack M. Rogers. Human ventricular fibrillation: Wandering wavelets, mother rotors, or both? *Circulation*, 114(6):530–532, 2006.
29. Eunkyoung Jee, Shaohui Wang, Jeong-Ki Kim, Jaewoo Lee, Oleg Sokolsky, and Insup Lee. A safety-assured development approach for real-time software. In *RTCSA*, pages 133–142, 2010.
30. Zhihao Jiang, Miroslav Pajic, Allison Connolly, Sanjay Dixit, and Rahul Mangharam. Real-Time Heart model for implantable cardiac device validation and verification. In *ECRTS*, pages 239–248. IEEE Computer Society, 2010.
31. Zhihao Jiang, Miroslav Pajic, Salar Moarref, Rajeev Alur, and Rahul Mangharam. Modeling and verification of a dual chamber implantable pacemaker. In *TACAS*, pages 188–203, 2012.
32. Dilsun K. Kaynar, Nancy Lynch, Roberto Segala, and Frits Vaandrager. *The Theory of Timed I/O Automata*. Synthesis Lectures on Computer Science. Morgan Claypool, November 2005. Also available as Technical Report MIT-LCS-TR-917.
33. Gerardo Lafferriere, George J. Pappas, and Sergio Yovine. A new class of decidable hybrid systems. In *In Hybrid Systems : Computation and Control*, pages 137–151. Springer, 1999.
34. Jie Lian, Hannes Krätschmer, and Dirk Müssig. Open source modeling of heart rhythm and cardiac pacing. *The Open Pacing, Electrophysiology & Therapy Journal*, pages 28–44, 2010.
35. Daniel Liberzon. *Switching in Systems and Control*. Systems and Control: Foundations and Applications. Birkhauser, Boston, June 2003.
36. Winfried Lohmiller and Jean-Jacques E Slotine. On contraction analysis for nonlinear systems. *Automatica*, 34(6):683–696, 1998.

37. Nancy Lynch, Roberto Segala, and Frits Vaandrager. Hybrid I/O automata. *Information and Computation*, 185(1):105–157, August 2003.
38. Sayan Mitra. *A Verification Framework for Hybrid Systems*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA 02139, September 2007.
39. Sayan Mitra, Daniel Liberzon, and Nancy Lynch. Verifying average dwell time of hybrid systems. *ACM Trans. Embed. Comput. Syst.*, 8(1):1–37, 2008.
40. Nedialko S Nedialkov, Kenneth R Jackson, and George F Corliss. Validated solutions of initial value problems for ordinary differential equations. *Applied Mathematics and Computation*, 105(1):21–68, 1999.
41. R. Olfati-Saber, J.A. Fax, and R.M. Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, January 2007.
42. Miroslav Pajic, Zhihao Jiang, Insup Lee, Oleg Sokolsky, and Rahul Mangharam. From verification to implementation: A model translation tool and a pacemaker case study. In *IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 173–184, 2012.
43. Romualdo Pastor-Satorras and Alessandro Vespignani. Epidemic dynamics and endemic states in complex networks. *Physical Review E*, 63(6):066117, 2001.
44. Stephen Prajna, Antonis Papachristodoulou, and Pablo A Parrilo. Introducing sostools: A general purpose sum of squares programming solver. In *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on*, volume 1, pages 741–746. IEEE, 2002.
45. Eduardo D. Sontag. Comments on integral variants of iss. *Systems & Control Letters*, 34(1-2):93 – 100, 1998.
46. Steven H Strogatz. Exploring complex networks. *Nature*, 410(6825):268–276, 2001.
47. Vladimeros Vladimerou, Pavithra Prabhakar, Mahesh Viswanathan, and Geir E. Dullerud. Stormed hybrid systems. In *ICALP (2)*, volume 5126 of *LNCS*, pages 136–147. Springer, 2008.
48. Pei Ye, Emilia Entcheva, Radu Grosu, and Scott A. Smolka. Efficient Modeling of Excitable Cells Using Hybrid Automata. In *CMSB*, pages 216–227, 2005.

Acknowledgements. This project was partly supported by ERC AdG VERIWARE and PoC VERIPACE. We also thank Radu Grosu and Ezio Bartocci for helping us with the cardiac models.

A Definitions and Notations

Points, sets, vectors. For a vector or a point $s \in \mathbb{R}^n$ and a non-negative constant $\delta \geq 0$, $B_\delta(s)$ is the δ-ball around s with respect to the ℓ^2 norm. That is, $B_\delta(s) = \{y \mid |s - y| \leq \delta\}$. The definition is lifted to sets in $\mathbb{R}^n$ in the natural way. For a bounded set $S \subseteq \mathbb{R}^n$, its diameter $dia(S)$ is defined as $\max_{x,y \in S} |x - y|$.

For an positive integer n, $[n]$ denotes the set $\{0, \ldots, n\}$.

Functions and trajectories. A continuous function $f : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$ is *smooth* if all its higher derivatives and partial derivatives exist and are also continuous. It has a Lipschitz constant $K \geq 0$ if for every $x_1, x_2 \in \mathbb{R}^n$, $|f(x_1) - f(x_2)| \leq K|x_1 - x_2|$. Such functions are called Lipschitz continuous. The function x^2 over $\mathbb{R}$ is Lipschitz continuous but the exponential function e^x is not. A non-negative function $g : \mathbb{R}^n \to \mathbb{R}$ is a *class $\mathcal{K}$ function* if $g(x) \geq 0$ for $x \neq 0, g(0) = 0$ and $g(x) \to 0$ as $x \to 0$. A class $\mathcal{K}$ function g is called $\mathcal{K}_\infty$ if $g(x) \to \infty$ as $x \to \infty$. For example, the function $f(y_1, y_2) = y_1^2$ belongs to class $\mathcal{K}$ but not to $\mathcal{K}_\infty$. A function $g : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$ is called a $\mathcal{KL}$ function, if and only if (1) for each $t \in \mathbb{R}$, $g_t(x) \triangleq g(x,t)$ is a $\mathcal{K}$ function and (2) for each $x \in \mathbb{R}^n$, $g_x(t) \triangleq g(x,t) \to 0$ as $t \to \infty$ (see Appendix of [35] for these standard definitions).

For any function $f : A \to B$ and a set $S \subseteq A$, $f \lceil S$ is the restriction of f to S. That is, $(f \lceil S)(s) = f(s)$ for each $s \in S$. So, for a variable $v \in V$ and a valuation $\mathbf{v} \in \mathit{Val}(V)$, $\mathbf{v} \lceil v$ is the function mapping $\{v\}$ to the value $\mathbf{v}(v)$. In this paper, we write $\mathbf{v} \lceil v$ as $\mathbf{v}.v$. For any function $f : C \to [A \to B]$ and a set $S \subseteq A$, $f \downarrow S$ is the restriction of $f(c)$ to S. That is, $(f \downarrow S)(c) = f(c).S$ for each $c \in C$.

A *trajectory* for $\mathcal{V}$ with domain $[0, T]$ is a function $\xi : [0, T] \to \mathit{Val}(\mathcal{V})$. Its domain is denoted by $\xi.dom$. So, for a variable $v \in V$ and a trajectory τ of V, $\tau \downarrow v$ is the trajectory of v defined by τ. The valuation of the state variables ($\mathcal{V} = \mathcal{X} \cup \mathcal{L}$, not including the inputs $\mathcal{U}$) at the first and the last points in ξ are denoted by $\xi.\mathsf{fstate}$ and $\xi.\mathsf{lstate}$. That is, for a trajectory $\xi : [0, T] \to \mathit{Val}(\mathcal{V} \cup \mathcal{U})$, $\xi.\mathsf{fstate} = (\xi \downarrow \mathcal{V})(0)$ and $\xi.\mathsf{fstate} = (\xi \downarrow \mathcal{V})(T)$.

Hybrid automata. For an automaton $\mathcal{A}$, its components are denoted by $V_\mathcal{A}, X_\mathcal{A}, \Theta_\mathcal{A}, \mathcal{D}_\mathcal{A}, \mathcal{T}_\mathcal{A}$, etc. For an automaton $\mathcal{A}_i, \imath \in \{1, 2\}$, its components are denoted by $V_i, X_i, \Theta_i, \mathcal{D}_i, \mathcal{T}_i$, etc. For a given execution α of $\mathcal{A}$, $\alpha(t)$ is a shorthand for the lstate of the maximal prefix of α of duration t. The set of reachable states of automaton $\mathcal{A}$ from a arbitrary set of states $S \subseteq \mathit{Val}(V)$ is denoted by $\mathsf{Reach}_\mathcal{A}(S)$. The set of reachable states from the initial set $\Theta_\mathcal{A}$ is denoted by $\mathsf{Reach}_\mathcal{A}$. Similarly, the set of reachable states of from an arbitrary set S and the set of initial states Θ, within time bound $T \geq 0$ is denoted by $\mathsf{Reach}_\mathcal{A}(T, S)$ and $\mathsf{Reach}_\mathcal{A}(T)$. The type of the single argument will be used to disambiguate bounded time reach set from Θ ($\mathsf{Reach}_\mathcal{A}(T)$) from $\mathsf{Reach}_\mathcal{A}(S)$.

Operations with discrepancy functions. For a given function $V : Val(\mathcal{X}) \times Val(\mathcal{X}) \to \mathbb{R}_{\geq 0}$, a constant $\delta > 0$ and a point $\mathbf{x} \in Val(\mathcal{X})$, the set $B_\delta^V(\mathbf{x}) = \{\mathbf{x}' \in Val(\mathcal{X}) \mid V(\mathbf{v}, \mathbf{v}') \leq \delta\}$. The operation $B_\delta^V(\cdot)$ is lifted to subsets of $Val(\mathcal{X})$ in the usual way.

B Proofs of Propositions and Lemmas

Proof of Proposition 1:

Proof. As shown in Slotine's paper [36], considering the general deterministic systems:

$$\dot{x} = f(x) + Bu(t). \tag{3}$$

The plant equation (3) can be thought of as an n-dimensional fluid flow, where $\dot{x}$ is the n-dimensional "velocity" vector at the n-dimensional position x and time t. We can get that

$$\delta\dot{x} = \frac{\partial f}{\partial x}(x)\delta x + B\delta u,$$

where δx is a virtual displacement at fixed time. Formally, δx defines a linear tangent differential form, and $\delta x^T \delta x$ the associated quadratic tangent form [6], and they are both differentiable.

In our case, the squared distance between two neighboring trajectories $|\xi_{\mathbf{x}_1} - \xi_{\mathbf{x}_2}|^2$ can be defined as $\delta x^T \delta x$ [36], which lead to the rate of change:

$$\begin{aligned}
\frac{d}{dt}(\delta x^T \delta x) &= \delta x^T \frac{\partial^T f}{\partial x}\delta x + \delta x^T \frac{\partial f}{\partial x}\delta x + (B\delta u)^T \delta x + \delta x^T (B\delta u) \\
&\leq \delta x^T (\frac{\partial^T f}{\partial x} + \frac{\partial f}{\partial x})\delta x + (\delta x^T \delta x) + ((B\delta u)^T (B\delta u)) \\
&= \delta x^T (\frac{\partial^T f}{\partial x} + \frac{\partial f}{\partial x} + I)\delta x + ((B\delta u)^T (B\delta u)) \\
&\leq 2\lambda_{max}(x)\delta x^T \delta x + ((B\delta u)^T (B\delta u)),
\end{aligned}$$

where $\lambda_{max}(x)$is the largest eigenvalue of the symmetric part of the Jacobian matrix, i.e., $\frac{1}{2}(\frac{\partial f}{\partial x} + \frac{\partial f^T}{\partial x} + I)$, then we have

$$\begin{aligned}
\delta x^T \delta x &\leq e^{\int_0^t 2\lambda_{max}(x)dt}(\delta x_0^T \delta x_0 + \int_0^t ((B\delta u(\tau))^T (B\delta u(\tau)))d\tau) \\
&= e^{2\lambda_{max} t}(\delta x_0^T \delta x_0 + \int_0^t ((B\delta u(\tau))^T (B\delta u(\tau)))d\tau), \\
|\delta x|^2 &\leq e^{2\lambda_{max} t}(|\delta x_0|^2 + \int_0^t |B\delta u(\tau)|^2 d\tau).
\end{aligned}$$

Use the inequality of norm,

$$
\begin{aligned}
|\delta x| &\leq e^{\lambda_{max} t}|\delta x_0| + e^{\lambda_{max} t}\int_0^t |B\delta u(\tau)|d\tau \\
&\leq e^{\lambda_{max} t}|\delta x_0| + \sup_{s\in[0,t]} e^{\lambda_{max} s}\int_0^t |B\delta u(\tau)|d\tau
\end{aligned}
$$

After getting the format of discrepancy function, the next goal is to find the smallest λ such that $J \leq \lambda I$. Thus computation of discrepancy function is to solve such an optimization problem:

$$
\begin{cases}
\min\ \lambda \\
such\ that (J - \lambda I) \leq 0.
\end{cases} \tag{2}
$$

Here the minimum λ should be the largest eigenvalue of J.

For the linear dynamics case, it is easy to get that $\dot{x}_1 - \dot{x}_2 = A(x_1 - x_2) + B(v_1 - v_2)$, then we can get that $V(x_1, x_2) = |\xi(x_1, t) - \xi(x_2, t)| = e^{At}|x_1 - x_2| + \int_0^t e^{A(t-\tau)}B(v_1(\tau) - v_2(\tau))d\tau$.

Proof of Lemma 2.

Proof. Assume that *ReachFromCover* returns *SAFE* and fix an execution $\xi'_{\mathbf{v}}$ starting from $\mathbf{v}' \in B_\delta(\mathbf{v})$ with $\xi.\mathsf{dur} \leq T$. From Proposition 2, it suffices to show that $\xi'_{\mathbf{v}}$ can be decomposed into a sequence of execution fragments $\xi = \xi_0, \xi_1, \ldots, \xi_l$, such that for each fragment ξ_k, there is a round $p(k)$ in which ξ_k gets captured by the simulation ψ of that iteration.

The proof is by induction on the number of fragments. The first continuous trajectory of $\xi'_{\mathbf{v}}$ is captured by $\psi^{(1)}$ computed in the first iteration. Let ξ_0 be the longest prefix of ξ that can be captured by $\psi^{(1)}$. From Proposition 2, $\xi_0.\mathsf{lstate} \in \cup_{j=1}^{l(m_l)} s_j^{(1)}$. Let ξ'' be the suffix of $\xi_{\mathbf{v}}$ such that $\xi_{\mathbf{v}} = \xi_0, \xi''$. Then, either (a) a transition occurs between ξ_0 and ξ'' ($\xi_0.\mathsf{lstate} \to \xi''.\mathsf{fstate}$) or (b) they concatenate in the middle of a continuous evolution ($\xi_0.\mathsf{lstate} = \xi''.\mathsf{fstate}$). Case (a) will be detected by line 7 and case (b) will be detected by line 10. It follows by line 9 and 12 that, in either case, new initial time-state pairs $(\mathbf{v}, t_0)$ will be added to the set $\mathcal{C}$ and there exists a pair $(\mathbf{v}'', t_0'')$ such that $\xi''.\mathsf{fstate} \in B_\delta(\mathbf{v}'')$. Since the algorithm returns SAFE, the pair $(\mathbf{v}'', t_0'')$ eventually gets explored from $\mathcal{C}$ in some iteration. Then, a prefix of ξ'' is captured by the simulation; let ξ_1 be the longest prefix that is captured. So, we can continue with the construction by defining $\xi_{\mathbf{v}} = \xi_0, \xi_1, \xi'''$ where the first two execution fragments are captured. Since *ReachFromCover* terminates and returns SAFE, the decomposition process terminates. □

C On minimal ventricular model and MAH model of cardic cells

The MV model [12] is defined by four nonlinear ODEs:

$$\begin{aligned}
\frac{\partial x_1(\boldsymbol{d},t)}{\partial t} &= e(\boldsymbol{d},t)-(J_{\text{fi}}+J_{\text{so}}+J_{\text{si}}), \\
\frac{\partial x_2(\boldsymbol{d},t)}{\partial t} &= \frac{(1-H(x_1(\boldsymbol{d},t)-\theta_u))\cdot(v_\infty-x_2(\boldsymbol{d},t))}{\tau_v^-} - H(x_1(\boldsymbol{d},t)-\theta_v)\cdot\frac{x_2(\boldsymbol{d},t)}{\tau_v^+}, \\
\frac{\partial x_3(\boldsymbol{d},t)}{\partial t} &= \frac{(1-H(x_1(\boldsymbol{d},t)-\theta_w))\cdot(w_\infty-x_3(\boldsymbol{d},t))}{\tau_w^-} - H(x_1(\boldsymbol{d},t)-\theta_w)\cdot\frac{x_3(\boldsymbol{d},t)}{\tau_w^+}, \\
\frac{\partial x_4(\boldsymbol{d},t)}{\partial t} &= \frac{1+\tanh(k_s\cdot(x_1(\boldsymbol{d},t)-u_s))}{2\cdot\tau_s} - \frac{x_4}{\tau_s},
\end{aligned} \tag{3}$$

where $H(x)=\int_{-\infty}^{x}\delta(\tau)d\tau$ is the Heaviside function and δ is Dirac delta function. The myocytes transmembrane potential $x_1(\boldsymbol{d},t)$ depends on the location $\boldsymbol{d}\in\mathbb{R}^3$ in the muscle tissue and time t. The dynamics of $x_1(\boldsymbol{d},t)$ is determined by an external stimulus $e(\boldsymbol{d},t)$ and by a sum of three currents: fast inward J_{fi}, slow inward J_{so} and slow outward J_{si}, where the three currents are given by the following equations:

$$\begin{aligned}
J_{\text{fi}} &= \frac{-x_2(\boldsymbol{d},t)\cdot H(x_1(\boldsymbol{d},t)-\theta_v)\cdot(x_1(\boldsymbol{d},t)-\theta_v)\cdot(u_u-x_1(\boldsymbol{d},t))}{\tau_{\text{fi}}}, \\
J_{\text{so}} &= (x_1(\boldsymbol{d},t)-u_o)\cdot\frac{(1-H(x_1(\boldsymbol{d},t)-\theta_w))}{\tau_o} + \frac{H(x_1(\boldsymbol{d},t)-\theta_w)}{\tau_{\text{so}}}, \\
J_{\text{si}} &= \frac{-H(x_1(\boldsymbol{d},t)-\theta_w)\cdot x_3(\boldsymbol{d},t)\cdot x_4(\boldsymbol{d},t)}{\tau_{\text{si}}}.
\end{aligned} \tag{4}$$

The flow of the currents is controlled by the fast channel gate $x_2(\boldsymbol{d},t)$ and two slow gates $x_3(\boldsymbol{d},t)$ and $x_4(\boldsymbol{d},t)$. The value for the rest of the parameters of Eq.3 and Eq.4 are given in Table 1 [12].

In Fig. 3 we depict the value of $x_1(\boldsymbol{d},t)$, $x_2(\boldsymbol{d},t)$, $x_3(\boldsymbol{d},t)$ and $x_4(\boldsymbol{d},t)$ over time triggered by an external stimulus $e(\boldsymbol{d},t)$. Note that the dynamics of the myocyte from Eq.3 does not depend on its location $\boldsymbol{d}$ in the muscle tissue. The location $\boldsymbol{d}$ will become crucial when we will extend the MV model to a (3D) tissue. The transmembrane potential $x_1(\boldsymbol{d},t)$ has four phases: stimulated, rapid upstroke, early repolarisation (plateau) and final repolarisation (resting). The stimulated phase starts when the myocyte is stimulated by an external stimulus or by neighbourhood myocytes. If the stimulus current does not reach the cell's stimulation threshold, then the cell cannot get stimulated, and consequently it goes to the resting phase. If the received current is high enough, the upstroke phase indicates the depolarisation of the cell. The early repolarisation phase indicates that the cell receives an influx of calcium. The final repolarisation is the last phase which features faster repolarisation that brings the transmembrane potential back to the resting phase.

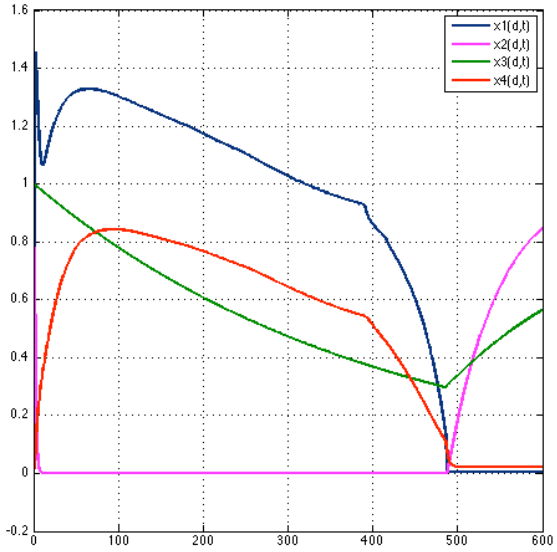

Fig. 3: The dynamics of the MV model over time.

As stated earlier, the behavior of the MV model from Eq.3 does not depend on the location $\boldsymbol{d}$. The MV model describes only the dynamics of a single cell. To incorporate the dependence on the location we extend the equation for the transmembrane potential $x_1(\boldsymbol{d},t)$ as follows $\frac{\partial u(\boldsymbol{d},t)}{\partial t} = \nabla(\boldsymbol{D}\nabla u(\boldsymbol{d},t)) + e(\boldsymbol{d},t) - (J_{\text{fi}} + J_{\text{so}} + J_{\text{si}})$, where $\boldsymbol{d} := (d_x, d_y, d_z)$, $\nabla x_1(\boldsymbol{d},t) := \left(\frac{\partial x_1(\boldsymbol{d},t)}{\partial d_x}, \frac{\partial x_1(\boldsymbol{d},t)}{\partial d_y}, \frac{\partial x_1(\boldsymbol{d},t)}{\partial d_z}\right)$ is the gradient of function $x_1(\boldsymbol{d},t)$ and $\boldsymbol{D} \in \mathbb{R}^{3\times 3}$ is the diffusion matrix. As we are interested only in myocytes connected in a ring we have that $\boldsymbol{d} := d_x$. Therefore, the above equation results in

$$\frac{\partial x_1(d_x,t)}{\partial t} = D\frac{\partial^2 x_1(d_x,t)}{\partial d_x^2} + e(x_1,t) - (J_{\text{fi}} + J_{\text{so}} + J_{\text{si}}), \tag{5}$$

where $D \in \mathbb{R}$ is a diffusion coefficient.