Learning and Detecting Emergent Behavior in Networks of Cardiac Myocytes

Radu Grosu
Department of Computer Science
Stony Brook University
Stony Brook, NY 11794-4400, USA
grosu@cs.sunysb.edu

Flavio Corradini
Department of Mathematics and Computer Science
University of Camerino
Camerino (MC), I-62032, Italy
flavio.corradini@unicam.it

Emilia Entcheva
Department of Biomedical Engineering
Stony Brook University
Stony Brook, NY 11794-8181, USA
emilia.entcheva@sunysb.edu

Scott A. Smolka
Department of Computer Science
Stony Brook University
Stony Brook, NY 11794-4400, USA
sas@cs.sunysb.edu

Anita Wasilewska
Department of Computer Science
Stony Brook University
Stony Brook, NY 11794-4400, USA
anita@cs.sunysb.edu

Ezio Bartocci*
Department of Mathematics and Computer Science
University of Camerino
Camerino (MC), I-62032, Italy
ezio.bartocci@unicam.it

ABSTRACT

We address the problem of specifying and detecting emergent behavior in networks of cardiac myocytes, spiral electric waves in particular, a precursor to atrial and ventricular fibrillation. To solve this problem we: (1) Apply discrete mode abstraction to the cycle-linear hybrid automata (CLHA) we have recently developed for modeling the behavior of myocyte networks; (2) Introduce the new concept of spatial-superposition of CLHA modes; (3) Develop a new spatial logic, based on spatial superposition, for specifying emergent behavior; (4) Devise a new method for learning the formulae of this logic from the spatial patterns under investigation; and (5) Apply bounded model checking to detect the onset of spiral waves. We have implemented our methodology as the EMERALD tool suite, a component of our EHA framework for specification, simulation, analysis and control of excitable hybrid automata. We illustrate the effectiveness of our approach by applying EMERALD to the scalar electrical fields produced by our CELLEXCITE simulation environment for excitable-cell networks.

1. INTRODUCTION

One of the most important and intriguing questions in systems biology is how to formally specify *emergent behavior in biological tissue*, and how to efficiently predict and detect its onset. A prominent example of such behavior is electrical *spiral waves* in spatial networks of cardiac myocytes (heart cells). Electrical impulses regularly circulate through cardiac tissue and cause the heart's muscle fibers to contract. In a healthy heart, these electrical impulses travel smoothly and unobstructed, like a water wave that ripples gently in a pond. These waves can, however, sometimes develop into troublesome, whirlpool-like spirals of electrical activity. Spiral waves of this nature are a precursor to a variety of cardiac disturbances, including *atrial fibrillation* (AF), an abnormal rhythm originating in the upper chambers of the heart. AF afflicts two-three million Americans alone, putting them at risk for clots and strokes. Moreover, the likelihood of developing AF increases with age.

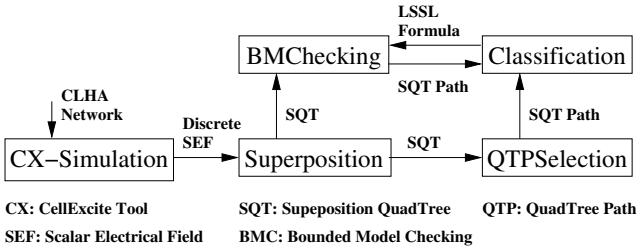

Figure 1: Overview of our method.

In this paper, we address this question by proposing a simple and efficient method for learning, and automatically detecting the onset of, spiral waves in cardiac tissue. See Figure 1 for an overview of our approach. Underlying our method is a *linear spatial-superposition logic* (LSSL) we have developed for specifying properties of spatial networks. LSSL is discussed in greater detail below. Our method also builds upon hybrid automata, image processing, machine learning, and model-checking techniques to first learn an LSSL formula that characterizes such spirals. The formula is then automatically checked against a *quadtree* representation [20] of the scalar electrical field (SEF) produced at each discrete time step by a simulation of a hybrid-automata network modeling the myocytes. A scalar field is a function that associates a scalar value, which in our case is an electric potential, to every point in space. The quadtree representation is obtained via discrete mode abstraction and hierarchical superposition of the elementary units within the SEF.

An earlier version of this paper appeared in *Proc. 11th International Conference on Hybrid Systems: Computation and Control* (HSCC'08), Springer, LNCS 4981, April 2008.

*Currently visiting the Department of Computer Science, Stony Brook University, Stony Brook, NY.

The electric behavior of cardiac myocytes is hybrid in nature: they exhibit an all-or-nothing electrical response, the so-called *action potential* (AP), to an external excitation. An AP can thus be viewed as triggering a discrete mode transition from the cell's resting mode of continuous behavior to its excited mode of continuous behavior. Despite their discrete-continuous hybrid nature, networks of myocytes have traditionally been modeled using nonlinear partial differential equations [13, 17]. While highly accurate in describing the molecular processes underlying cell behavior—nonlinear differential equations allow one to closely match the values of a multitude of state variables to their actual physical values—these models are not particularly amenable to formal analysis and typically do not scale well for the simulation of complex cell networks.

In [11], we showed that it is possible to automatically learn a much simpler *hybrid automaton* (HA) [12] model for cardiac myocytes, which explicitly captures, up to a prescribed error margin, the mixed discrete and continuous behavior of the AP. To highlight its *cyclic* structure and its *linear* dynamics, which may vary in interesting ways from cycle to cycle, we called it a *cycle-linear hybrid automaton* (CLHA). Moreover, as we showed in [24, 25, 2], one can use a variant of this CLHA model to efficiently (up to an order of magnitude faster) and accurately simulate the behavior of myocyte networks, and, in particular, induce spirals and fibrillation.

A key observation concerning our simulations, see Figure 3, is that mode abstraction, in which the AP value of each CLHA in the network is *abstracted* to its corresponding mode, faithfully preserves the network's waveform and other spatial characteristics. Hence, for the purpose of learning, and detecting the onset of, spirals within CLHA networks, we can exploit mode abstraction to dramatically reduce the system state space. A similar mode abstraction is possible for voltage recordings in live cell networks.

The state space of a 400×400 CLHA network is still prohibitively large, even after applying the above-described abstraction: it contains $4^{160,000}$ modes, as each CLHA has four mode values. To combat state explosion, we use a spatial abstraction inspired by [14]: we regard the mode of each automaton as a probability distribution and define the *mode superposition* of a set of CLHAs as the probability that an arbitrary CLHA in this set is in a particular mode. By successively applying superposition to the network, we obtain a tree structure, the root of which is the mode superposition of the entire CLHA network, and the leaves of which are the modes of the individual CLHA. The particular structure we employ, quadtrees, is inspired by image-processing techniques [20]. We shall refer to quadtrees obtained in this manner as *superposition-quadtrees* (SQT).

Our LSSL logic is an appropriate logic for reasoning about *paths* in SQTs, and the spatial properties of CLHA networks in which we are interested, including spirals, can be cast in LSSL. For example, we have observed that the presence of a spiral can be formulated in LSSL as follows: Given an SQT, is there a path from its root leading to the core of a spiral? Based on this observation, we build a machine-learning classifier, the training-set records for which correspond to the probability distributions associated with the nodes along such paths. Each distribution, for mode value *stimulated*, corresponds to an attribute of a training-set record, with the number of attributes bounded by the depth of the SQT. An additional attribute is used to classify the record as either spiral or non-spiral. For spiral-free SQTs, we simply record the path of maximum distribution.

For training purposes, our CellExcite simulator [2] generates, upon successive time steps, snapshots of a 400×400 CLHA network and their mode abstraction; see Figures 1,3. Training data for the classifier is then generated by converting the abstracted snapshots into SQTs and selecting paths leading to the core of a spiral (if present). The resulting table is input to the decision-tree algorithm of the Weka machine-learning tool suite [8]. This produces a classifier, in the form of a path-predicate, constraining the distribution of the attribute *stimulated* in each node along the path.

The syntax of LSSL is similar to that of linear temporal logic, with LSSL's Next operator corresponding to *concretization* (anti-superposition). Moreover, a sequence of LSSL Next operators corresponds to a path through an SQT. The classifier produced by Weka can therefore be regarded as an LSSL formula. An SQT path can be thought of as a magnifying glass, which starting from the root, produces an increasingly detailed but more focused view of the image (i.e. abstracted snapshot). This effect is analogous to *concept hierarchy* in data mining [16] and arguably similar to the way the brain organizes knowledge: a human can recognize a word or a picture without having to look at all of the characters in the word or all of the details in the picture, respectively.

Although the LSSL logic and its underlying semantics (Kripke structures) allow us to reason about infinite paths through recursive structures (fractals), physical considerations—such as the number of myocytes in a cardiac tissue or the screen resolution—impose a maximum length k on such paths. We therefore maintain k as a parameter in LSSL's semantic definition, permitting us to accommodate any finite number of myocytes or screen resolution. Defining LSSL's semantics in this manner places us within the framework of bounded model checking [3].

Our spatial-superposition logic might also be understood as a *Scale logic*, as it allows us to examine an image at various scales or levels of detail. The notion of scale is prevalent in biological systems, ranging from genetic scale to societal scale. The built-in notion of scale in our logic therefore makes it well suited for reasoning about biological systems.

We are now in a position to view spiral detection as a bounded-model-checking problem [3]: Given the SQT Q generated from the discrete SEF of a CLHA network and an LSSL formula φ learned through classification, is there a finite path π in Q satisfying φ? We use this observation to check every time step during simulation whether or not a spiral has been created. More precisely, the LSSL formula we use states that no spiral is present, and we thus obtain as a counterexample one or all the paths leading to the core of a spiral. In the latter case, we can identify the number of spirals in the SEF and their actual position.

The above-described method, including user-guided path selection, has been fully implemented as the Emerald tool suite for automated spiral learning and detection. Emerald is written in Java, and it is a new component of our EHA environment for the specification, simulation, analysis and control of CLHA networks. EHA stands for Excitable Hybrid Automata, as we have used CLHA to model various types of excitable cells, including neurons and cardiac myocytes [25]. The EHA environment is freely available from [10].

The rest of the paper is organized as follows. Section 2 reviews excitable-cell networks and their modeling with CLHA.

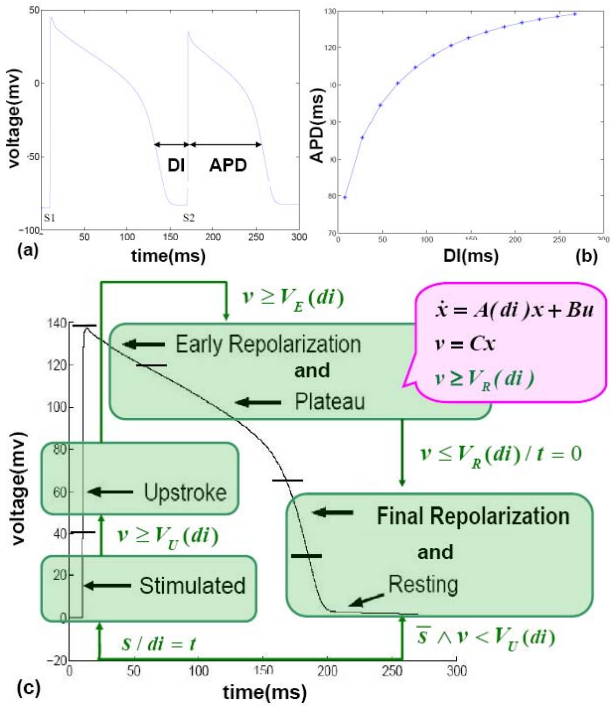

Figure 2: CLHA model of cardiac myocytes.

Section 3 defines superposition and quadtrees, the essential ideas underlying linear spatial-superposition logic, the topic of Section 4. Section 5 describes our learning and bounded-model-checking techniques; their implementation is considered in Section 6, along with our experimental results. Section 7 discusses related work. Section 8 offers our concluding remarks and directions for future research.

2. BIOLOGICAL BACKGROUND

An *excitable cell* (EC) has the ability to propagate an electrical signal, known at the cellular level as the *action potential* (AP), to the neighboring cells. An AP corresponds to a change of potential across the cell membrane, and is caused by the flow of ions between the inside and outside of the cell through the membrane's ion channels.

An AP is an externally triggered event (with duration): an EC fires an AP as an all-or-nothing response to a suprathreshold stimulus, and each AP follows the same sequence of phases (described below) and exhibits roughly the same waveform regardless of the applied stimulus. During most of the AP no re-excitation can generally occur (the EC is in a refractory period).

Despite differences in duration, morphology and underlying ion currents, the following major AP *phases* can be identified across different species and EC types: *resting*, *stimulated*, *upstroke*, *early repolarization*, *plateau* and *final repolarization*. We abbreviate them as `r` (resting and final repolarization), `s` (stimulated), `u` (upstroke), and `p` (plateau and early repolarization).

Using the AP phases as a guide, we have developed, for several representative EC types, *cycle-linear hybrid automata* (CLHA) models that approximate the AP and other bio-electrical properties with reasonable accuracy. Their derivation was first performed manually [24, 25]. We subsequently showed in [11] how to fully automate this process by learning various biological aspects of the AP of different cell types.

Intuitively, a hybrid automaton [12] is an extended finite-state automaton, the states of which encode the various phases of continuous dynamics a system may undergo, and the transitions of which are used to express the switching logic between these dynamics. The CLHA we obtained are fairly compact in nature, employing two or three continuous state variables and four to six modes. The term *cycle-linear* is used to highlight the cyclic structure of CLHA, and the fact that while in each cycle they exhibit linear dynamics, the coefficients of the corresponding linear equations and mode-transition guards may vary in interesting ways from cycle to cycle.

Figure 2 presents one of our CLHA models. To understand the model, first note that when an EC is subjected to repeated stimuli, two important time periods can be identified: AP *duration* (APD), the time the cell is in an excited state, and *diastolic interval* (DI), the time between the "end" of the AP and the next stimulus. Figure 2(a) illustrates the two intervals. The function relating APD to DI is called the APD *restitution function*. As shown in Figure 2(b), the relationship is nonlinear and captures the phenomenon that a longer recovery time is followed by a longer APD. A physiological explanation of a cell's restitution is rooted in the ion-channel kinetics as a limiting factor in the cell's frequency response.

The CLHA model itself, superimposed over the image of a typical AP, is given in Figure 2(c). Each mode has associated a linear dynamics $\dot{x} = Ax + Bu$, $v = Cx$, where x is the CLHA state, u is the input, and v (for voltage) is the output. A mode also has an associated invariant in v, forcing the outgoing transition to be eventually taken. The concept of mode dynamics and invariant is illustrated in Figure 2(c) for mode `p` (plateau and early repolarization); see that mode's callout. Transition labels are of the form $e \wedge g/a$, where e is an (optional) event, g is a guard, and $/a$ is an optional set of assignments. The only events in the model, representing the start and end of stimulation, are denoted by s and $\overline{s}$, respectively. Observe the per-mode and transition-guard dependence on the DI, which is measured with the help of clock variable t.

The dynamics of excitable-cell networks play an important role in the physiology of many biological processes. For cardiac cells, on each heart beat, an electrical control signal is generated by the sinoatrial node, the heart's internal pacemaking region. Electrical waves then travel along a prescribed path, exciting cells in atria and ventricles and assuring synchronous contractions. Of special interest are cardiac arrhythmias: disruptions of the normal excitation process due to faulty processes at the cellular level, single ion-channel level, or at the level of cell-to-cell communication. The clinical manifestation is a rhythm with altered frequency, tachycardia (rapid heart beat) or bradycardia (slow heart beat), or the appearance of multiple frequencies, polymorphic ventricular tachycardia, with subsequent deterioration to a chaotic signal, ventricular fibrillation (VF). VF is a typically fatal condition in which there is uncoordinated contraction of the cardiac muscle of the ventricles in the heart. As a result, the heart fails to adequately pump blood. and hypoxia (lack of oxygen) may occur.

In order to simulate the emergent behavior of cardiac tissue, we have developed CellExcite [2], a CLHA-based simulation environment for excitable-cell networks. CellExcite allows the user to sketch a tissue of excitable cells, plan the stimuli to be applied during simulation, and customize

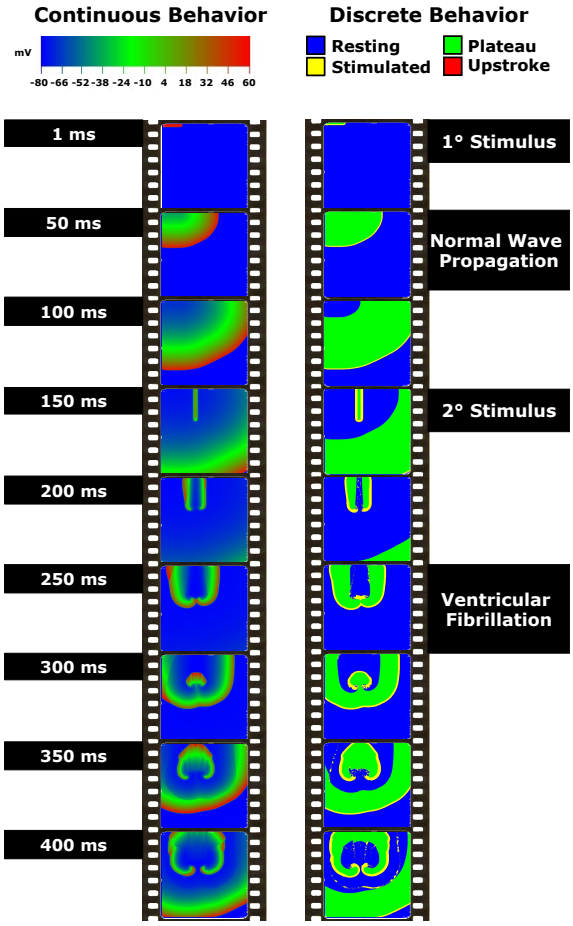

Figure 3: Simulation of continuous and discrete behavior of a CLHA network.

the arrangement of cells by selecting the appropriate lattice. Figure 3 presents our simulation results for a 400×400 CLHA network of cardiac myocytes. Nine 50-ms. simulation steps are shown, during which (steps 1 and 4) the network was stimulated twice, at different regions. The results we obtain demonstrate the feasibility of using CLHA networks to capture and mimic different spatiotemporal behavior of wave propagation in 2D isotropic (homogeneous) cardiac tissue, including normal wave propagation (1-150 ms); the creation of spirals, a precursor to fibrillation (200-250 ms); and the break-up of such spirals into more complex spatiotemporal patterns, signaling the transition to ventricular fibrillation (250-400 ms).

As can be clearly seen in Figure 3, mode abstraction, in which the action-potential value of each CLHA in the network is *discretely abstracted* to its corresponding mode, faithfully preserves the network's waveform and other spatial characteristics. Hence, for the purpose of learning and detecting spirals within CLHA networks, we can exploit mode abstraction to dramatically reduce the system state space.

3. SUPERPOSITION AND QUADTREES

A key benefit of using hybrid automata compared to nonlinear ODEs is their explicit support for finitary abstractions: the infinite range of values of a hybrid automaton's continuous state variables can be abstracted to the automaton's discrete finite set of modes. As discussed in Sections 1 and 2, abstracting the AP (voltage) of the constituent CLHA in a CLHA network to their corresponding mode (`s`, `u`, `p` or `r`) turns out to faithfully preserve the network's waveform and other spatial characteristics. This simplifying approximation allows us to reduce the spiral-onset verification problem to a finite-state verification problem.

Although in this paper we consider a CLHA network an execution at a time, our ultimate goal is exhaustive simulation, i.e. model checking. Within this context, the state space of a 400×400 CLHA network, which would be necessary to simulate the behavior of a tissue of about 16 cm^2 in size, is still too large for analysis purposes: it has $4^{160,000}$ mode values! To combat state explosion, we use a spatial abstraction inspired by [14]. Consider the state space of a CLHA network. We regard the current mode of each CLHA in the network as a degenerate probability distribution, and define the *superposition* of a set of (possibly superposed) modes as the mean of their distributions. By successively applying superposition to the CLHA network, we obtain a tree whose root is the mode superposition of the entire network, and whose leaves are the individual modes of the component CLHA. The particular superposition tree structure we employ, the quadtree, was inspired by image-processing techniques [20].

Let $\mathcal{A}$ be a $2^k \times 2^k$ matrix of CLHA modes. A quadtree $Q = (V, R)$ representation of $\mathcal{A}$ is a quaternary tree, such that each vertex $v \in V$ represents a sub-matrix of $\mathcal{A}$. For example, the root v_0 of the quadtree in Figure 4 represents the entire matrix; child v_1 represents the matrix $\{2^{k-1}, \ldots, 2^k\} \times \{0, \ldots, 2^{k-1}\}$; child v_6 represents the matrix $\{2^{k-1}, \ldots, 3 * 2^{k-2}\} \times \{0, \ldots, 2^{k-2}\}$; etc.

Due to superposition, a quadtree is in general a more efficient data structure than the matrix it represents: if the subtree rooted at a node of a quadtree is of one "color" (mode in our case), then there is no need to descend into the node's subtree as no additional information can be gleaned by doing so. Moreover, given a quadtree representation of an image and a property of the image in which one is interested—such as determining whether a mode-abstracted snapshot of a CLHA network contains a spiral—it may only be necessary to follow a *path* through the quadtree (as opposed to exploring the entire tree) to determine if the property holds. Moreover, the path need not necessarily descend all the way to the leaf level, but rather may terminate at an interior node. See Sections 4 and 5 for a further discussion of such quadtree properties.

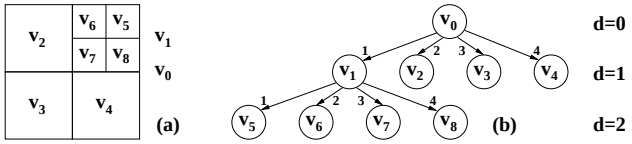

Figure 4: Quadtree representation.

Definition *1.* **(Distributions).** Let $\mathcal{N}$ be a CLHA network whose constituent CLHA have (ordered) modes $M = \{\mathtt{s}, \mathtt{u}, \mathtt{p}, \mathtt{r}\}$, and let Q be the quadtree representation of $\mathcal{N}$. Then each leaf node $l \in Q$ has an associated degenerate *leaf distribution* D_l whose probability mass function (also sometimes known as the point mass function, and in either case abbreviated as PMF) p_l is such that $\exists! m \in M.\ p_l(m) = 1$. Also, let $i \in Q$ be an interior node with children $i_1, \ldots, i_4$. Then i has an associated *superposition distribution* D_i whose PMF p_i is such that $\forall m \in M. p_i(m) = \frac{1}{4} \sum_{j=1}^{4} p_{ij}(m)$.

The intuition is as follows. If a leaf node occurs at the maximum depth of the quadtree, then it corresponds to the current mode of a CLHA. As CLHA are deterministic, they assume one of the values in M with probability 1. (We will weaken this restriction at the end of the section when we consider superposition quad-graphs.) If the leaf does not occur at the maximum depth of the quadtree, then it corresponds to the superposition of identical degenerate distributions, and no additional information is obtained by decomposing the leaf into its four superposition components. The visual interpretation is that a pixel has one definite color, and nothing is learned by decomposing an area in which all pixels have the same color.

As for the distribution of an interior node i, if all of i's children are leaves, then, for each mode value m, i's superposition is the mean of the occurrences of m. Hence, the probability that the mode of the parent is m is the probability that the mode of an arbitrary child is m. If i's children are interior nodes, it still holds that the probability that i's mode is m is the probability that the mode of an arbitrary leaf below i's children is m.

We call a quadtree whose nodes are labeled with leaf and interior-node distributions a *superposition quadtree* (SQT). The distributions in an SQT are not known in advance; our learning algorithm seeks to determine them for what we perceive to be spirals. The use of probability distributions is justified by the fact that different spirals might have slightly different shapes; i.e., slightly different distributions of values for the leaf nodes of their associated quadtrees.

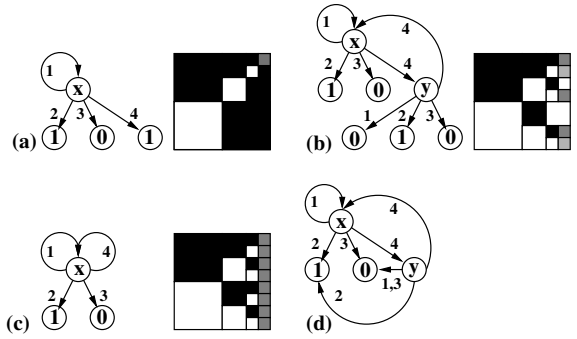

Figure 5: Fractals as finite-state SQGs: (a) $x=2/3$, (b) $x=5/11$, $y=4/11$, (c) $x=1/2$.

The SQTs presented so far were constructed over a finite matrix $\mathcal{A}$ containing $2^k * 2^k$ elements. In general, SQTs can be obtained via the finite unfolding of a *quad-graph*.

Definition *2.* **(SQG).** A *superposition quad-graph* (SQG) is a 4-tuple $G = (V, v_0, R, L)$ consisting of:

- A finite set of *vertices* V with *initial vertex* $v_0 \in V$,
- A *transition relation* $R \subseteq V \times [1..4] \times V$ s.t. $\forall v \in V, i \in [1..4]. \exists\, u \in V.\ (v, i, u) \in R$,
- A *probability-distribution labeling* L s.t. $\forall v \in V.\ L(v) = 1/4 \sum_{u \in R(v)} L(u)$.

The condition on R ensures that each vertex in V has precisely four successors in R. The condition on L ensures that the probability distributions are related through superposition. The manner in which we construct SQTs as finite unfoldings of SQGs can be extended to support the definition of *infinite* SQTs generated by *recursion*. That is, it supports the definition of *fractals*. Furthermore, just as we use SQTs to represent finite images, SQGs can be used to represent *infinite* images; i.e. fractals.

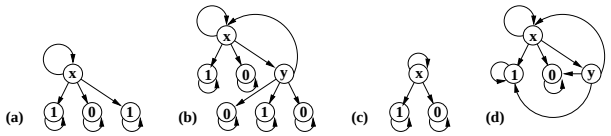

Figure 6: Kripke structures for SQGs of Figure 5.

Figure 5(a-c) gives SQGs representing the recursive specification of three fractals and a graphical depiction of the unfolding of these SQGs up to depth 3. (The SQG of Figure 5(d), for which no depiction is given, is considered below.) Note the fractal-like nature of these pictures: the *gray* areas represent recursion and correspond to recursive nodes in the SQGs. Such nodes are labeled by *distribution variables*, the values for which can be computed by solving a *linear system*. For example, x and y in Figure 5(b) are computed by solving the linear system $x = 1/4\,(x + 1 + y)$ and $y = 1/4\,(1 + x)$. The four self-loops of the leaves are not shown for simplicity. Note that leaves may now be associated with any constant distribution. Also note that the finite-state SQGs of Figure5 (b) and (d) yield equivalent infinite SQTs.

4. LINEAR SUPERPOSITION LOGIC

In this section, we define *linear spatial-superposition logic* (LSSL). Although the LSSL logic—especially its spatial analogues of the temporal fixpoint operators of LTL [18]—and its underlying semantics (Kripke structures) allow us to reason about infinite paths, physical considerations such as the number of myocytes in a cardiac tissue or screen resolution, impose a maximum length k on paths. We therefore maintain k as a parameter in LSSL's semantic definition, placing us within the framework of bounded model checking [3].

Every finite SQT can be transformed into an SQG by adding to each leaf node a self-loop labeled by i, $i \in [1..4]$. Moreover, an SQG can be transformed into a *Kripke structure* by erasing (forgetting) the transition labeling, collapsing all resulting transitions that share the same source and target nodes into one transition, and assuming nondeterminism among transitions emanating from the same node. For example, applying this forgetful transformation to the SQGs of Figure 5 yields the Kripke structures of Figure 6, where the self-loops are made explicit. The Kripke structure of Figure 6(d) can be seen as the minimal-state equivalent of the one of Figure 6(b) where nodes labeled by 0 or 1 are shared.

Definition *3.* **(Kripke structure).** A *Kripke structure* (KS) over a set of atomic formulas AF is a four-tuple $M = (S, I, R, L)$ consisting of:

- A countable set of *states* S, with *initial states* $I \subseteq S$;
- A *transition relation* $R \in S \times S$ with $\forall s \in S. \exists\, t \in S.\ (s, t) \in R$;
- A *labeling (or interpretation) function* $L : S \to 2^{AF}$.

The condition associated with the transition relation R ensures that every state has a successor in R. Consequently, it is always possible to construct an infinite path through a KS, an important property when dealing with reactive systems. In our case, it means that we can reason about recursive SQTs, i.e. fractals.

The labeling function L defines for each state $s \in S$ the set $L(s)$ of atomic formulas that are valid in s. Atomic formulas are *inequalities over distributions* of the form $P[D = m] \sim d$,

where D is a distribution function, $m \in M$ is a discrete value (e.g. a mode), $d \in [0..1]$, and $\sim$ is one of $<, \leq, =, \geq$, or $>$. We use $P[D=m]$ as a more intuitive notation for $p(m)$, where p is the PMF associated with D. (This notation is also more reminiscent of $P[X=m]$, where X is a random variable.) It should thus be noted that the 0-1 state labels used in Figure 6, where the mode in question is `s`, are shorthand for the atomic proposition $P[D=\mathtt{s}]=0$ or $P[D=\mathtt{s}]=1$.

In order to verify properties of a reactive system modeled as a KS K, it is customary to use either a linear-time or a branching-time temporal logic. A model for a linear-time (LTL) formula is an infinite path π in K. A model for a branching-time formula is K itself; given a state s of K, this allows one to quantify over the paths originating from s. For our current purposes of specifying and detecting the onset of, spirals, LTL suffices.

Strictly speaking, our logic is a *linear spatial-superposition logic* (LSSL), as a path π in K represents a sequence of *concretizations* (anti-superpositions). Syntactically, however, our temporal-logic operators are the same as in LTL: the *next* operator X, with $X\varphi$ meaning that φ holds in a concretization of the current state; its inverse operator B; the *until* operator U, with $\varphi U \psi$ meaning that φ holds along a path until ψ holds; and the *release* operator R, with $\psi R \varphi$ meaning that φ holds along a path unless released by ψ.

Definition 4. (LSSL Syntax). The syntax of *linear space-superposition logic* is defined inductively as follows:

$$\begin{array}{lll} \varphi & ::= & \top \mid \bot \mid P[D=m] \sim d \mid \neg\phi \mid \varphi \vee \psi \mid X\varphi \mid \\ & & B\varphi \mid \varphi U \varphi \mid \varphi R \varphi \\ \sim & ::= & < \mid \leq \mid = \mid \geq \mid > \end{array}$$

As discussed above, a bound k on the path length is maintained as a parameter in LSSL's semantic definition.

Definition 5. (LSSL Semantics). Let K be a KS, π a path in K, and $f \in AF$ an atomic formula. Then, for $k \geq 0$, π *satisfies an* LSSL *formula* φ *with bound* k, written $\pi \models_k \varphi$, only if $\pi \models_k^0 \varphi$, where:

$$\begin{array}{lcl} \pi \models_k^i \top & \text{and} & \pi \not\models_k^i \bot \\ \pi \models_k^i f & \Leftrightarrow & f \in L(\pi[i]) \\ \pi \models_k^i \neg\varphi & \Leftrightarrow & \pi \not\models_k^i \varphi \\ \pi \models_k^i \varphi \vee \psi & \Leftrightarrow & \pi \models_k^i \varphi \text{ or } \pi \models_k^i \psi \\ \pi \models_k^i X\varphi & \Leftrightarrow & i < k \text{ and } \pi \models_k^{i+1} \varphi \\ \pi \models_k^i B\varphi & \Leftrightarrow & 0 < i \leq k \text{ and } \pi \models_k^{i-1} \varphi \\ \pi \models_k^i \varphi U \psi & \Leftrightarrow & \exists j.\ i \leq j \leq k.\ \pi \models_k^j \psi \text{ and} \\ & & \forall n.\ i \leq n < j.\ \pi \models_k^n \varphi \\ \pi \models_k^i \psi R \varphi & \Leftrightarrow & \forall j.\ i \leq j \leq k.\ \pi \models_k^j \varphi \text{ or} \\ & & \exists n.\ i \leq n < j.\ \pi \models_k^n \psi \end{array}$$

We say that $K \models_k \varphi$ if for all paths π in K, $\pi \models_k \varphi$.

Our until and release operators U and R are bounded versions of the LTL operators U and R. Similarly, the *globally* operator G, defined as $G\varphi \equiv \bot R \varphi$, is a bounded version of LTL's G operator. The *finally* operator F is defined as usual as $F\varphi \equiv \top U \varphi$. In general, the unbounded LTL version of G is assumed to not hold. For example, $G\varphi$ does not hold as φ could be violated at $k+1$; to decide $G\varphi$ in LTL with respect to a bound k, one needs a more sophisticated analysis of the KS K, as discussed in [3].

To illustrate LSSL, consider a k-unfolding of the KS of Figure 6(a), and assume the distributions labeling the states correspond to mode `s`. Then, this KS has a path π such that $\pi \models_k G(P[D=\mathtt{s}]=2/3)$ holds: the path that always returns to x. To automatically find π, we can model check the negation of this formula; as discussed in Section 5, π will be returned as a counterexample. Using the techniques in [3], one can show that π also satisfies the unbounded LTL version of the formula.

5. MODEL CHECKING AND LEARNING

Bounded model checking. Given a KS K, LSSL formula φ, and bound k, a *bounded model checker* (BMC) efficiently verifies if $K \models_k \varphi$. If not, it returns one or more paths π in K that violate φ (i.e., counterexamples); otherwise, it returns true. Intuitively, a BMC applies the LSSL semantics inductively defined in Section 4 to each path π in K. We have implemented a simple prototype BMC for KSs derived from SQTs and LSSL formulae, which first enumerates all paths in a KS and then for each path, applies the LSSL semantics. This approach is efficient enough to check within milliseconds the onset of spirals. We are currently improving our handling of safety formulae (those without the F operator) by pruning, during SQT traversal, all subtrees of a vertex as soon as we detect that the current path satisfies φ. A more ambitious SAT-based BMC is also under development.

Machine learning. Writing the LTL formulae that a reactive system should satisfy is a nontrivial task. Developers often find it difficult to specify the system properties of interest. The classification of LTL formulae into *safety* (something bad should never happen) and *liveness* properties (something good should eventually happen) provides some guidance, but the task remains difficult.

Writing LSSL formulae describing emerging properties of CLHA networks is even more difficult. For example, what is the LSSL formula for spiral onset? In the following, we describe a surprisingly simple, machine-learning-based approach that we have successfully applied to spiral detection. The main idea is to cast the onset property as follows: *Is there a path in the given* SQT *leading to the core of a spiral?*, where the core of a spiral is the central point from which the spiral emanates, getting progressively farther away as it revolves around the point.

The implementation of our approach is simple as well. For an SEF (a 400×400 array of AP values) produced by the CellExcite simulator (see Figure 1), our Emerald tool set allows the user to select a path through the SEF's corresponding SQT simply by clicking on a point in the SEF; e.g. in the core of a spiral. If no spiral is present, the SQT path with maximum PMF (probability mass function) is returned. Note that this method is not restricted to spirals: path selection via clicking on a representative point can be applied to normal wave propagation, wave collision, etc.

The paths so obtained are then used to learn the LSSL formula for the property in question, such as spiral onset. The learning algorithm works as follows: (1) For each path of length k, where k is the height of the SQT, we define k attributes $a_1, \ldots, a_k$ such that each a_i holds the PMF value of vertex v_i, for the mode we are interested in (for spirals, mode `s`). (2) Each path is classified by Emerald as spiral or non-spiral depending on whether or not the user clicked on a point (core); the classification is stored as an additional

classifier attribute c. (3) All records $(a_i, \ldots, a_k, c)$ are stored in a table, which is provided to the data-classification phase. (4) At the end of this phase, we obtain a path classifier which we translate into an LSSL formula.

Data classification [22] is generally a two-step process: training and testing. For *training*, we choose a classification algorithm that learns a set of descriptions of our training data set. The form of these descriptions depends on the type of classification algorithm employed. For *testing*, we use a test data set, disjoint from the training set, and containing the class attribute with a known value. The *accuracy* of the classifier on a given test set is the percentage of the test records that are correctly classified. Various techniques can be used to obtain test and training sets from an initial set of records, such as X-Cross Validation [8].

For classification purposes, we use a *descriptive classifier* (DC), which returns a set of if-then rules called *discriminant rules*. Underlying DCs are decision trees, rough sets, classification-by-association analysis, etc. A rule r has form

$$(\bigwedge_{i \in I} a_i = v_i) \Rightarrow (c = v)$$

where I is a subset of $[1..k]$. Usually, each class c has an associated set of rules $r_1, \ldots, r_n$; i.e. c is characterized by $\bigwedge_{i=1}^{n} r_i$. Using boolean arithmetic, this is equivalent to

$$(\bigvee_{i=1}^{n} \bigwedge_{j \in I_i} a_{ij} = v_{ij}) \Rightarrow (c = v)$$

The antecedent formula $\bigvee_{i=1}^{n} \bigwedge_{j \in I_i} a_{ij} = v_{ij}$ is called the *class description formula* of the class c.

As is customary, we built a classifier for one class only (the class c), called the *target class*, using all other classes as one contrasting class. Hence the classifier consists of only one class-description formula, describing the target class. We say that we *learned* that formula. We have used Weka's decision-tree algorithm, but any other rule-based algorithm could have been used as well. The classifier we have learned for spirals is as follows:

if a_7 <= 0.875 **then**
  if a_2 <= 0.048 **then** ∼c **else** c
else if a3 <= 0.078 **then**
  if a0 <= 0.025 **then** ∼c **else** c
else ∼c

Its translation into LSSL, where $\mathtt{X}^{\mathtt{k}}$ stands for k repetitions of X, generated the following formula:

$$\mathtt{X}^2\ \mathtt{P(D=s)} > 0.048 \wedge \mathtt{X}^7\,\mathtt{P(D=s)} \leq 0.875\ \vee$$
$$\mathtt{P(D=s)} > 0.025 \wedge \mathtt{X}^3\,\mathtt{P(D=s)} \leq 0.078\ \wedge\ \mathtt{X}^7\,\mathtt{P(D=s)} > 0.875$$

This formula is an approximate description of a spiral which we use together with EMERALD's BMC to detect spiral onset within milliseconds. In case the BMC returned a false positive, we add the corresponding record to the classification table as part of a retraining phase; see Figure 1.

6. IMPLEMENTATION

Our techniques of Sections 2-5 have been implemented as the EMERALD tool suite of the EHA environment. EMERALD is a Java application that can be used to learn an LSSL formula for a particular spatial pattern, and to check the formula against a set of images (of the kind pictured on the right-hand side of Figure 3) that reproduce the discrete

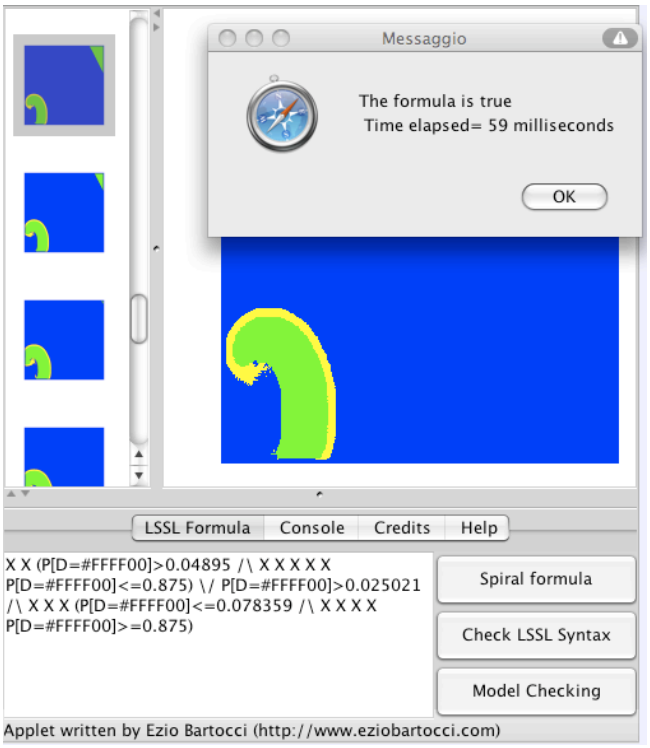

Figure 7: EMERALD Bounded Model Checker.

behavior of a CLHA network. For ease of use, EMERALD provides two graphical tools, one for *Preprocessing* (classification) and the other for *Bounded Model Checking*.

The Preprocessing tool enables users to browse the various collections of images they have assembled for machine-learning purposes, and to view their SQT representation. The user can select a path leading to a spiral core by clicking on an appropriate stimulated point (in yellow) of the image. If the image does not contain a spiral, the user can choose the maximum PMF path or a generic stimulated point. Each path selected is stored in a data table in the form of the PMF sequence of stimulated modes in each node of the traversed SQT. All such paths are subsequently exported to Weka in a common format. Presently, we have customized EMERALD for spiral detection, but we plan to extend the tool with the capability to classify any generic spatial pattern.

The BMC applet (Figure 7) enables the user to check an LSSL formula against the SQT representation of a specific image. As discussed in Section 5, the LSSL formula encodes the classifier for the spatial pattern under investigation. If the SQT in question fails to satisfy the formula, the resulting counter-examples (spirals) are reported to the user both as rows in the counterexample table and as red dots marking the core of the spiral contained in the image.

Table 1 contains our preliminary experimental results. For training and testing purposes, we used two different sets of images, each containing spirals and normal wave propagation. The first set of images was used to train the classifier; we supervised the training by discriminating between paths leading to a spiral core versus those (of maximum PMF) belonging to images that did not contain a spiral. From this first set we extracted 512 possible paths, and used Weka to build a ruled-based classifier with a very high prediction

Path Classifier	Test Set 550	Test Set 600	Test Set 650
Trained (512 Paths)	87.00%	88.83%	88.23%
Retrained (512 Paths + 67 Counterexamples)	97.10%	97.33%	93.07%

Table 1: Experimental Results

accuracy (99.25%).

The test set was divided into increasingly larger sets of images: 500, 550, 600 and 650 images. Applying the rule-based classifier on the first 500 images produced 67 wrongly classified paths. We used these paths to obtain a new, retrained classifier. We then used both classifiers on the remaining sets of images, and for each classifier and test set we computed the LSSL *formula accuracy*, as an estimate of how well the formula specifies the spatial pattern. As Table 1 shows, retraining considerably improves accuracy, and can be repeated each time a false classification is returned.

Weka's decision-tree algorithm took less than 9s to construct a rule-based classifier from the training (512 records) and retraining (579 records) tables, respectively. Our model checker took between 1.67s–7.09s, with an average of 4.72s to model-check the SQT for a 400×400 SEF if no spiral was present, and between 1ms and 4.64s, with an average of 230ms otherwise. All results were obtained on a PC equipped with a Centrino 2GHz processor with 1.5GB RAM.

7. RELATED WORK

The use of hybrid automata to model and analyze spatial networks is a relatively new subject area, and includes application to Delta-Notch signaling networks [9], coordinated control of autonomous underwater vehicles [19], and aircraft trajectories and landing protocols [7, 21]. In contrast, our focus is on emergent behavior (in the form of spiral waves) in networks of cardiac myocytes, and the use of spatial superposition as an abstraction mechanism. Predicting spirals [4] in pure continuous models [23] is a more complicated process than what is implemented in EMERALD, where discrete SQT structures, obtained via mode abstraction and superposition, are used. Several logics have recently been proposed for describing the behavior and spatial structure of concurrent systems [5, 6], and for reasoning about the topological aspects of modal logics and Kripke structures [1]. Unlike LSSL, these logics are not based on an abstraction mechanism like spatial-superposition that can be used to alleviate state explosion during model checking.

8. CONCLUSIONS

In this paper, we have presented a framework for specifying and detecting emergent behavior in networks of cardiac myocytes. Our approach, which uses hybrid automata, discrete mode abstraction, and bounded model checking, is based on a novel notion of spatial-superposition and its related logic LSSL, and a new method for the automated learning of formulae in this logic from the spatial patterns under investigation. Our framework has been fully implemented in the EMERALD tool suite. Our preliminary experimental results are very encouraging, with a prediction accuracy of over 93% on a test set comprising 650 images. As future work, we plan to extend our framework to the learning of branching-time spatial-superposition properties, and the more intricate problem of specifying and detecting spatiotemporal emergent behavior.

We also experimented with the SIFT (Scale-Invariant Feature Transform) algorithm, which detects and matches interesting features in images while preserving invariance constraints for scaling, translation, and rotation [15]. We found that SIFT performed matching well on images of spirals that were related to one another through rigid transformations. It was less successful, due to an insufficient number of matching keypoints, on spirals with more markedly different shapes. Also, SIFT and other image-processing techniques tend to process the entire image. Our approach, in contrast, uses logical formulae over SQT *paths* and densities of a particular CLHA mode (stimulated) along such paths.

Acknowledgements.

We would like to thank the anonymous reviewers for their valuable comments. Research supported in part by NSF awards CCR-0133583, CNS-0509230 and CCF-0523863.

9. REFERENCES

[1] M. Aiello, J. Benthem, and G. Bezhanishvili. Reasoning about space: The modal way. *J. Log. Comput.*, 13(6):889–920, 2003.

[2] E. Bartocci, F. Corradini, E. Entcheva, R. Grosu, and S. A. Smolka. CellExcite: An efficient simulation environment for excitable cells. *BMC Bioinformatics*, **9**(Suppl 2):S3, 2008.

[3] A. Biere, A. Cimatti, E. Clarke, O. Strichman, and Y. Zhu. Bounded model checking. In *Adv. in Comp. vol. 58: Highly Depend. Software*. Acad. Press, 2003.

[4] M. A. Bray, S. F. Lin, R. R. Aliev, B. J. Roth, and J. P. J. Wikswo. Experimental and theoretical analysis of phase singularity dynamics in cardiac tissue. *J Cardiovasc Electrophysiol*, 12(6):716–722, 2001.

[5] L. Caires and L. Cardelli. A spatial logic for concurrency (part I). *Inf. Comput.*, 186(2):194–235, 2003.

[6] L. Caires and L. Cardelli. A spatial logic for concurrency (part II). *Theor. Comput. Sci.*, 322(3):517–565, 2004.

[7] I. deOliveira and P. Cugnasca. Checking safe trajectories of aircraft using hybrid automata. In *Proc. of SAFECOMP 2002*. Springer-Verlag, Sept. 2002.

[8] E. Frank, M. A. Hall, G. Holmes, R. Kirkby, B. Pfahringer, I. H. Witten, and L. Trigg. WEKA: A machine learning workbench for data mining. In *The Data Mining and Knowledge Discovery Handbook*, pages 1305–1314. Springer, 2005.

[9] R. Ghosh, A. Tiwari, and C. Tomlin. Automated symbolic reachability analysis; with application to Delta-Notch signaling automata. In *HSCC*, pages 233–248, 2003.

[10] R. Grosu, E. Bartocci, F. Corradini, E. Entcheva, S. A. Smolka, and P. Ye. EHA: An environment for the specification, simulation, analysis and control of networks of excitable hybrid automata. `http://www.cs.sunysb.edu/~eha`, 2008.

[11] R. Grosu, S. Mitra, P. Ye, E. Entcheva, I. V. Ramakrishnan, and S. A. Smolka. Learning

cycle-linear hybrid automata for excitable cells. In *Proc. of HSCC'07, the 10th International Conference on Hybrid Systems: Computation and Control*, volume 4416 of *LNCS*, pages 245–258, Pisa, Italy, April 2007. Springer Verlag.

[12] T. A. Henzinger. The theory of hybrid automata. In *Proceedings of 11th IEEE Symposium on Logic in Computer Science*, pages 278–293, 1996.

[13] A. L. Hodgkin and A. F. Huxley. A quantitative description of membrane currents and its application to conduction and excitation in nerve. *J Physiol*, 117:500–544, 1952.

[14] Y. Kwon and G. Agha. Scalable modeling and performance evaluation of wireless sensor networks. In *IEEE RT Tech. and App. Symp.*, pages 49–58, 2006.

[15] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision 2*, pages 1150–1157, 1999.

[16] Y. Lu. Concept hierarchy in data mining: Specification, generation and implementation. Master's thesis, Simon Fraser University, Dec. 1997.

[17] C. H. Luo and Y. Rudy. A dymanic model of the cardiac ventricular action potential: I. simulations of ionic currents and concentration changes. *Circ Res*, 74:1071–1096, 1994.

[18] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification*. Springer, 1992.

[19] F. Pereira and J. deSousa. Coordinated control of networked vehicles: An autonomous underwater system. *Aut. and Remote Ctrl*, 65(7):1037–1045, 2004.

[20] E. Shusterman and M. Feder. Image compression via improved quadtree decomposition algorithms. *IEEE Trans. on Image Processing*, 3(2):207–215, mar 1994.

[21] S. Umeno and N. Lynch. Safety verification of an aircraft landing protocol: A refinement approach. In *Proceedings of HSCC 2007*, Apr. 2007.

[22] A. Wasilewska and E. M. Ruiz. A classification model: Syntax and semantics for classification. In *RSFDGrC (2)*, pages 59–68, 2005.

[23] N. A. Wedge, M. S. Branicky, and M. C. Cavusoglu. Computationally efficient cardiac biolectricity models toward whole-heart simulation. In *Proc. of Intl. Conf. IEEE Engineering in Medicine and Biology Society*, pages 1–4, 2004.

[24] P. Ye, E. Entcheva, R. Grosu, and S. Smolka. Efficient modeling of excitable cells using hybrid automata. In *Proc. of CMSB'05, the 3rd Workshop on Computational Methods in Systems Biology*, pages 216–227, Edinburgh, Scotland, April 2005.

[25] P. Ye, E. Entcheva, S. A. Smolka, and R. Grosu. Modeling excitable cells using cycle-linear hybrid automata. *IET Syst Biol*, 2:24–32, Jan. 2008.