Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

Quantitative verification of implantable cardiac pacemakers over hybrid heart models

Taolue Chen, Marco Diciolla, Marta Kwiatkowska*, Alexandru Mereacre

Department of Computer Science, University of Oxford, UK

ARTICLE INFO

Article history:
Available online xxxx

Keywords:
Hybrid automata
Quantitative verification
Approximate verification
Pacemaker

ABSTRACT

We develop a model-based framework which supports approximate quantitative verification of implantable cardiac pacemaker models over hybrid heart models. The framework is based on hybrid input–output automata and can be instantiated with user-specified pacemaker and heart models. For the specifications, we identify two property patterns which are tailored to the verification of pacemakers: “can the pacemaker maintain a normal heart behaviour?” and “what is the energy level of the battery after t time units?”. We implement the framework in Simulink based on the discrete-time simulation semantics and endow it with a range of basic and advanced quantitative property checks. The advanced property checks include the correction of pacemaker mediated Tachycardia and how the noise on sensor leads influences the pacing level. We demonstrate the usefulness of the framework for safety assurance of pacemaker software by instantiating it with two hybrid heart models and verifying a number of correctness properties with encouraging experimental results.

1. Introduction

Recent technological progress has led to widespread use of software controlled medical devices, for example, cardiac pacemakers [15], GPCA infusion pumps [16], and continuous glucose monitors [26]. Safety and reliability of such devices are of paramount importance, which calls for rigorous software development frameworks. The main difficulty posed by medical devices is that they must handle monitoring and control of nonlinear continuous flows, e.g., electrical signal or glucose level, in conjunction with discrete mode switching, while at the same time also dealing with stochasticity. The latter arises due to probabilistic switching between diseased and normal behaviours, resulting in randomness of the timing of the events, and also aspects such as device failure or noise on the sensor readings. Therefore, developing *quantitative*, *probabilistic verification* techniques for implantable devices is essential to ensure their safety and reliability.

We focus our attention on a model-based framework for the development and quality assurance of pacemaker software. Such a framework has to enable the specification of pacemaker models and heart models, support their composition (to allow the pacemaker to monitor the heart beats and, if they are missed, insert artificial paces), and provide a range of correctness checks that can be established on the composition of the pacemaker instantiated with a suitable heart model, for example one that exhibits behaviours specific to a given patient. Typical correctness properties include basic safety properties such as whether the pacemaker corrects Bradycardia (slow heart beat), maintaining normal heart rhythm of 60–100 beats per minute (BPM), as well as advanced properties, e.g., whether the pacemaker can correct pacemaker mediated Tachycardia (PMT), which arises when the pacemaker increases the heart rate to unsafe levels. Quantitative verification is also well suited to prediction and optimisation of energy usage of the pacemaker device, a key concern in view of surgical intervention being necessary due to battery depletion.

* Corresponding author.

0890-5401/$ – see front matter © 2014 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.ic.2014.01.014

There has been much interest in developing modelling and verification approaches for pacemaker software (see below for description of related work). Our work extends that of Jiang et al. who developed *timed automata* models for the dual-chamber pacemaker [15], as well as the Virtual Heart Model (VHM) [13,14], also based on timed automata. While the VHM system, now implemented on FPGA, is intended for simulation and testing, the pacemaker model of [15] has been verified using UPPAAL [3] against a random heart model. The random heart model simply generates signals *nondeterministically* within a real-time interval, and hence cannot represent a realistic heart rhythm, which is regular, varies from patient to patient, and exhibits randomness in the timing of the events. For the basic safety property, the random heart model yields an over-approximation of heart behaviours. Intuitively, using this model, when the verification outcome is affirmative, the pacemaker is deemed to be "safe". However, when the outcome is negative, the verification is inconclusive. Consequently, we desire a more refined heart model, which does not suffer from spurious verification results and is able to deal with advanced pacemaker algorithms.

For these purposes, we have developed two hybrid heart models, first presented in [4,5]: one adapted from a synthetic multi-channel electrocardiogram (ECG) signal based on nonlinear ordinary differential equations (ODEs) [6], and the other formed as a network of cardiac cells, inspired by [27,2]. The ECG model involves surface readings from the human chest, and its advantage is relative simplicity, at a cost of approximating the signal that the pacemaker reads by signals which can be regarded as the external behaviour of the heart. One of its shortcomings is that it does not capture the conduction structure of the heart, and, as a consequence, certain advanced properties, for instance the PMT, cannot be established in that model. In contrast, the cardiac cell network model is fine-grained, directly involves *action potential* (AP) readings from individual cells, and is capable of modelling conduction delays needed to establish advanced correctness properties. Both heart models incorporate *stochasticity* not considered in [14,27], enabling probabilistic switching between diseased and normal rhythms that can be adapted to patient data.

In this paper, we propose a generic model-based framework for quantitative verification of pacemaker software that incorporates stochasticity to model probabilistic switching and noise. The framework can be instantiated with a model for pacemaker software and a hybrid heart model. We implement the framework in Simulink based on discrete-time simulation semantics and endow it with a range of basic and advanced quantitative property checks. Typical examples of advanced properties include "what is the energy level of the battery after t time units?". We demonstrate the usefulness of the framework by instantiating it with two hybrid heart models and present encouraging experimental results for the extended pacemaker models of [14] using approximate quantitative verification.

1.1. Contributions

The contribution of this paper can be summarised as follows:

- We develop two hybrid heart models with stochastic features, one based on ECG and the other on a network of cardiac cells. The models are tailored for quantitative verification and can be made patient-specific.
- We formulate a generic model-based framework for pacemaker software which can be instantiated with hybrid heart models.
- We implement in Simulink an extensive approximate quantitative verification framework which supports a range of property patterns.
- We develop techniques to analyse in detail the effect of pacing noise and energy consumption.

This paper is an extended version of [4,5] which introduced the ECG and the cardiac cell models. In addition to the results presented there, we formulate the model-based framework, describe its implementation based on discrete-time simulation semantics, and provide experimental results for advanced properties, including PMT correction and detailed analysis of energy consumption.

1.2. Related work

Pacemaker software has been modelled and analysed using a number of approaches that employ formal methods. In [15], a dual-chamber pacemaker is modelled as a timed automaton, based on specifications provided by Boston Scientific, and verified using UPPAAL against a simple random heart model. Luu et al. [19] develop a real-time formal model for a pacemaker and verify it with the PAT model checker. Networks of timed automata are employed in the Virtual Heart Model [13,14,25,12] and hybrid automata are used in the model of [27,2], both analysed through simulation. No stochasticity is considered. Macedo et al. [21] develop and analyse a concurrent and distributed real-time model for pacemakers through a pragmatic incremental approach using VDM and scenarios. Gomes and Oliveira [8] present a formal specification of the pacemaker using the Z notation and employ theorem proving, whereas Méry and Singh [24] use Event-B and the ProB tool to validate their models.

Several works formulate heart models, but composition with pacemakers is not studied. In [9,18], the authors develop a model of the cardiac conduction system that addresses the stochastic behaviour of the heart, validated via simulation. However, the hybrid behaviour of the heart is not considered. Grosu et al. [11] carry out automated formal analysis of a realistic cardiac cell model, and Grosu et al. [10] propose a method to learn and to detect the emergent behaviour (i.e. the

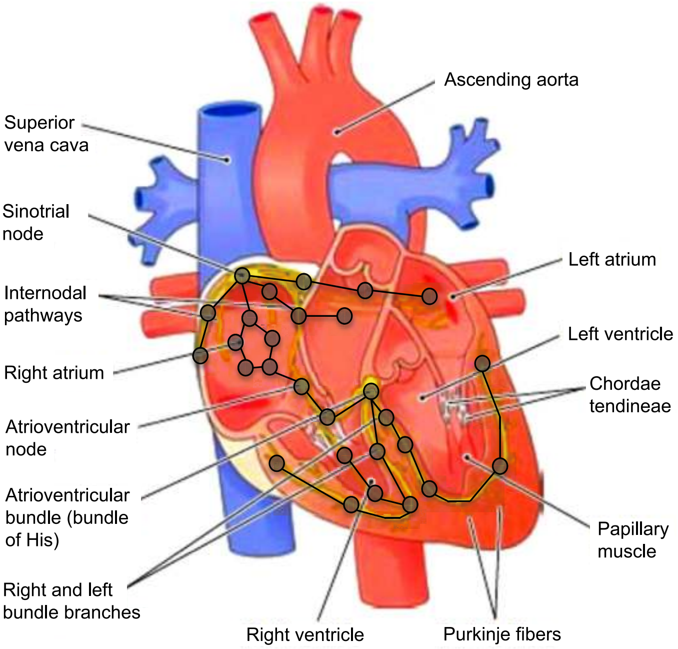


Fig. 1. Electrical conduction system of the heart.

spiral formation) that may lead to the onset of a ventricular fibrillation. Furthermore, risk analysis of glucose infusion pumps is performed with physiological models using statistical model checking in [26], but there is no stochasticity in the models.

1.3. Organisation

Section 2 introduces the necessary background material about the heart and its electrical conduction system. Section 3 describes the model-based verification framework. The details of the heart and pacemaker models are given respectively in Section 4 and Section 5. The property patterns and experimental results are presented in Section 6. Finally, Section 7 concludes the paper and gives a summary possible future directions.

2. Heart basics

In this section we describe the working of the heart, including its *electrical conduction system* (ECS). The main function of the human heart is to maintain blood circulation of the body. This rhythmic, pump-like function is driven by muscle contractions, and in particular the contraction of the atria and ventricles which are triggered by electrical signals.

2.1. SA node

The *sinoatrial* (SA) node (a special tissue in the heart, see Fig. 1) spontaneously produces an electrical signal, which is the *natural pacemaker* of the heart. On each heart beat, it generates the control electrical signal which is conducted through prescribed *internodal pathways* into the atrium causing its contraction. The signal then passes through the slow conducting *atrioventricular (AV) node*, allowing the blood to empty out the atria and fill the ventricles. The fast conducting *Purkinje* system spreads the electricity through the ventricles, causing all tissues in both ventricles to contract simultaneously and to force blood out of the heart. At the cellular level, the electrical signal is a change in the potential across the cell membrane, which is caused by the flow of ions between the inside and outside of the cell. It is known that sodium, potassium and calcium are the major ion species involved in this process; they flow through multiple voltage-gated ion channels. Excitation disturbances can occur in the behaviour of these ion channels at the cellular level, or in the propagation of the electrical waves at the cell network level [27].

Abnormalities in the electrical signal generation and fast or slow propagation can cause different types of arrhythmias, such as *Tachycardia* and *Bradycardia*, which require medical intervention in the form of medication, surgery or implantable pacemakers.

2.2. Action potential

At the cellular level, the heart tissue is activated by an external voltage applied to the cell or the SA node. After the activation, a transmembrane voltage change over time can be sensed due to ion channel activities, which is referred to as

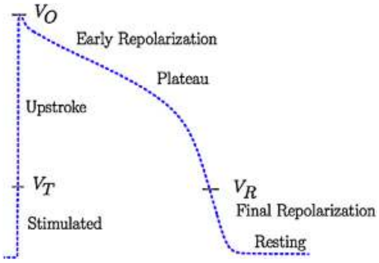


Fig. 2. Action potential [28].

an *action potential* (AP). This is also the signal that an implantable pacemaker will receive or generate. A simplified version of the *ventricular* AP (generated according to the dynamic Luo–Rudy model [28] from the Simulink implementation via MATLAB) is shown in Fig. 2. The AP is fired as an all-or-nothing response to a supra-threshold electrical signal, and each AP follows roughly the same sequence of events and has the same magnitude regardless of the applied stimulus. In general, APs exhibit the following major phases:

- *Stimulated*. This is the phase where the cell is triggered by a voltage spike from the AP of its neighbouring tissue or from an artificial pacing signal (pacemaker signal). However, if the current does not reach the threshold, then the cell cannot get stimulated, and consequently it goes to the resting phase.
- *Rapid upstroke*. If the received voltage spike is high enough, the upstroke indicates the depolarisation of the cell and the time when the muscle contracts.
- *Plateau and ER* (early repolarisation). This is a plateau phase during which calcium influx facilitates the muscle contraction.
- *Resting and FR* (final repolarisation). This is the last phase which features faster repolarisation that brings the potential back to the resting phase.

After the initial increase in the membrane potential, an AP lasts for a couple of hundred milliseconds (for most mammals including human beings). The early portion of an AP is known as the *effective refractory period* (ERP), due to its non-responsiveness to further stimulation, and the latter portion is known as the *relative refractory period* (RRP), during which an altered secondary excitation event is possible if the stimulation threshold is raised. Excitations during the RRP period will produce a slightly shorter subsequent ERP period.

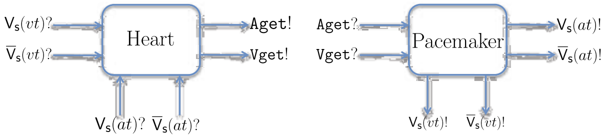
3. Model-based framework for the verification of pacemakers

In this section, we introduce a formal framework for the modelling and quantitative verification of pacemaker models. The framework is based on hybrid input–output automata [20] and supports the composition of a heart model and a pacemaker model on which verification is performed. We describe specific models that we have implemented and used to validate the framework. This includes two heart models (see Section 4) and the pacemaker model (see Section 5) based on [15]. We remark that our framework is generic and can be instantiated with user-specified heart models and pacemaker models.

We start with some notations. Let $\mathcal{X} = \{x_1, \ldots, x_d\}$ be a set of variables in $\mathbb{R}$. An $\mathcal{X}$-valuation is a function $\eta : \mathcal{X} \to \mathbb{R}$ assigning to each variable $x \in \mathcal{X}$ a real value $\eta(x)$. Let $\mathcal{V}(\mathcal{X})$ denote the set of all valuations over $\mathcal{X}$. A *constraint* on $\mathcal{X}$, denoted by *grd*, is a conjunction of expressions of the form $x \bowtie c$ for variable $x \in \mathcal{X}$, comparison operator $\bowtie \in \{<, \leqslant, >, \geqslant\}$ and $c \in \mathbb{R}$. Let $\mathcal{B}(\mathcal{X})$ denote the set of constraints over $\mathcal{X}$. An $\mathcal{X}$-valuation η satisfies constraint *grd*, denoted $\eta \models grd$, if and only if $(\eta(x_1), \ldots, \eta(x_d)) \in grd$. For $\delta \in \mathbb{R}$ and $\mathcal{X}$-valuation η, $\eta + \delta$ is the $\mathcal{X}$-valuation η' such that $\forall x \in \mathcal{X}.\ \eta'(x) := \eta(x) + \delta$, which implies that all variables proceed at the same speed. Let $\mathcal{Y}(\mathcal{X})$ denote the set of all real-valued functions over $2^{\mathcal{X}}$. We define $\mathcal{L}(\mathcal{X}) := \{x := u \mid x \in \mathcal{X} \wedge u \in \mathcal{X} \cup \{0\}\}$ to be the set of *update* assignments over the set of variables $\mathcal{X}$. For an assignment $\lambda = \{x := u\} \in \mathcal{L}(\mathcal{X})$ we write $\eta[\lambda]$ to be the valuation η' such that $\eta'(x) = \eta(u)$ and $\eta'(y) = \eta(y)$ for all $y \in \mathcal{X}$ and $y \neq x$. We also define $\mathrm{Distr}(A)$ to be the set of probability distributions over the finite set A.

Definition 1 *(Hybrid I/O automaton).* A hybrid I/O automaton (HA) $\mathcal{A} = (\mathcal{X}, Q, q_0, E_1, E_2, \mathrm{Inv}, \to, \mathcal{D})$ consists of: a finite set of variables $\mathcal{X}$; a finite set of modes Q, with the initial mode $q_0 \in Q$; a finite set E_1 of *input* actions and a finite set E_2 of *output* actions with $\mathcal{E} = E_1 \cup E_2$; an invariant function $\mathrm{Inv} : Q \to \mathcal{B}(\mathcal{X})$; a transition relation $\to \subseteq Q \times (\mathcal{E} \cup \{\varsigma\}) \times \mathcal{B}(\mathcal{X}) \times 2^{\mathcal{L}(\mathcal{X})} \times Q$, where ς is the internal action; a derivative function $\mathcal{D} : Q \times \mathcal{X} \to \mathcal{Y}(\mathcal{X})$ that assigns a function to a variable $x \in \mathcal{X}$.

The above definition is adapted from [20]. We impose some restrictions on HAs, which are described when we introduce the network of HAs. The state space of an HA is $\mathcal{S} = Q \times \mathcal{V}(\mathcal{X})$. A state $s \in \mathcal{S}$ is a pair $s = (q, \eta)$ where $q \in Q$ is a mode and η is the continuous state denoting a valuation of all variables $\mathcal{X}$. The initial state is $s_0 = (q_0, \vec{0})$ where $\vec{0}$ is the valuation


$\mathsf{V_s}(vt)$?
$\overline{\mathsf{V}}_\mathsf{s}(vt)$?
Heart
Aget!
Vget!
$\mathsf{V_s}(at)$? $\overline{\mathsf{V}}_\mathsf{s}(at)$?
Aget?
Vget?
Pacemaker
$\mathsf{V_s}(at)$!
$\overline{\mathsf{V}}_\mathsf{s}(at)$!
$\mathsf{V_s}(vt)$! $\overline{\mathsf{V}}_\mathsf{s}(vt)$!

(a) Heart model. (b) Pacemaker model.

Fig. 3. Interfaces for the heart and pacemaker models.

which assigns 0 to each variable. Notice that we only consider transitions with at most one input or output action. Let $\Phi : Q \times \mathcal{V}(\mathcal{X}) \times \mathbb{R}_{\geqslant 0} \to \mathcal{V}(\mathcal{X})$ be the flow function defined by $\mathcal{D}$. The (unlabelled) transition relation of $\mathcal{A}$ is a set $\mathcal{J} \subseteq \mathcal{S} \times \mathcal{S}$ that defines transitions between states of $\mathcal{A}$.

We now define the discrete time simulation semantics for a hybrid automaton. The main idea is to discretise the flow functions of the HA using an integration routine $\mathbb{S}$. There are several standard approaches that can be applied here, for instance the Runge–Kutta method which has a total accumulation error of $\mathcal{O}(h^4)$, where h is the time step. More specifically, we use $\Phi^{\mathbb{S}}$ to denote the flow function obtained by using the integration routine $\mathbb{S}$.

Definition 2 *(Discrete-time semantics).* (See [1].) Consider an HA $\mathcal{A}$, an integration routine $\mathbb{S}$, a time step h and a time bound T. Let $k = \lfloor \frac{T}{h} \rfloor$. The set of *k-step trajectories* of $\mathcal{A}$ consists of sequences (discrete paths) of the form $\sigma = s_0, s'_0, \ldots, s_{k-1}, s'_{k-1}$, where the states $s_i = (q_i, \eta_i)$ and $s'_i = (q'_i, \eta'_i)$ are defined as follows:

- the state $s_0 = (q_0, \eta_0)$ is the initial state;
- for each $0 \leqslant i \leqslant k-1$, (q'_i, η'_i) is the state after the continuous state evolution of the system from (q_i, η_i) with $q'_i = q_i$ and $\eta'_i = \Phi^{\mathbb{S}}(q_i, \eta_i, h)$;
- for each $0 \leqslant i \leqslant k-2$, $((q'_i, \eta'_i), (q_{i+1}, \eta_{i+1})) \in \mathcal{J}$.

We define $\sigma[i]$ to be the i-th state in σ, and $|\sigma|$ to be the length of the discrete path, i.e., the number of states $s_i \in \sigma$.

We use a *network of HAs* for the composition of more than one HA. The (discrete-time simulation) semantics of a network of HAs is the same as for a single HA. In order to obtain a deterministic network we impose some restrictions on HAs as follows:

- they must be *input enabled*, meaning that, for each mode and each input action, there is an edge labelled by the input action;
- the output actions have the highest priority, meaning that they are always *urgent*, i.e., if at any state the output action is enabled, the system must execute that action;
- the input actions are never enabled unless the corresponding output actions from the environment synchronise with them: once they can be synchronised, they are urgent;
- for each mode, there is a self-loop labelled by the internal action.

Definition 3 *(Network of hybrid automata).* Let m be the number of HAs in the network. A state of the network is $((q^{(1)}, \eta^{(1)}), \ldots, (q^{(m)}, \eta^{(m)}))$. There is a transition

$$\big((q_i^{(1)}, \eta_i^{(1)}), \ldots, (q_i^{(m)}, \eta_i^{(m)})\big) \to \big((q_{i+1}^{(1)}, \eta_{i+1}^{(1)}), \ldots, (q_{i+1}^{(m)}, \eta_{i+1}^{(m)})\big),$$

where

- either, for each $1 \leqslant k \leqslant m$, $(q_i^{(k)}, \eta_i^{(k)})$ has a continuous evolution;
- or, for each $1 \leqslant k \leqslant m$, $(q_i^{(k)}, \eta_i^{(k)})$ has a discrete transition. If, for some k, $(q_i^{(k)}, \eta_i^{(k)})$ enables an output action $a \in E_2^{(k)}$, then all of the other $(q_i^{(k')}, \eta_i^{(k')})$ must take a corresponding input action $a \in E_1^{(k')}$ (notice that this is guaranteed by input enabledness); otherwise, each state evolves by taking the internal action.

We assume in our framework that both the heart model and the pacemaker model are specified as hybrid input–output automata (in fact, we work with timed automata models for pacemakers which are subsumed by hybrid automata). To allow user-specified models, we have assumed fixed component interfaces for the heart and pacemaker models shown in Fig. 3. The heart and the pacemaker communicate via input and output actions which are marked by ? and ! respectively. The pacemaker communicates with the heart through four output actions, $\mathsf{V_s}(at)!$, $\overline{\mathsf{V}}_\mathsf{s}(at)!$, $\mathsf{V_s}(vt)!$ and $\overline{\mathsf{V}}_\mathsf{s}(vt)!$. The actions $\mathsf{V_s}(at)!$ and $\overline{\mathsf{V}}_\mathsf{s}(at)!$ denote the beginning and the end of the *atrial* stimulus, respectively, while $\mathsf{V_s}(vt)!$ and $\overline{\mathsf{V}}_\mathsf{s}(vt)!$ denote the beginning and end of the *ventricle* stimulus (see Section 4 for detailed descriptions). The heart communicates with the pacemaker using two output actions `Aget!` and `Vget!` (again, their meaning is given in Section 5).

4. Hybrid heart models

We describe two heart models that we have developed to validate our framework. The first one is based on ECG, while the second one is based on the propagation of the cardiac AP through the electrical conduction system of the heart by

Please cite this article in press as: T. Chen et al., Quantitative verification of implantable cardiac pacemakers over hybrid heart models, Inform. and Comput. (2014), http://dx.doi.org/10.1016/j.ic.2014.01.014

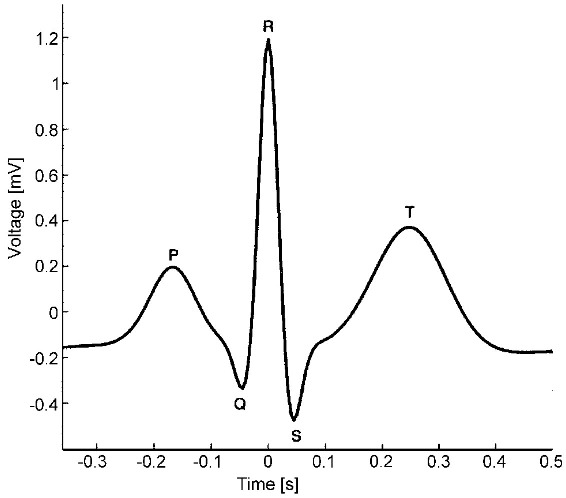


Fig. 4. Example electrocardiogram [23].

means of a network of heart cells. We also show how to map from an ECG heart model to a two-cell heart model, and how to generate different heart behaviours.

4.1. The ECG heart model

This heart model is based on ECG rhythms developed by Clifford et al. [6]. An ECG is a signal recorded from the surface of the human chest, which describes the activity of the heart. The ECG signal is an over-approximation of the electrical activity inside the human heart. An example ECG is given in Fig. 4.

Typically, an ECG signal describes a cardiac cycle composed of three main waves, P, QRS and T. The P wave denotes the *atrial depolarisation*. The QRS wave reflects the rapid *depolarisation of the right and left ventricles*. The T wave denotes the *repolarisation of the ventricles*. [6] presents a mathematical model for generating ECGs based on a system of nonlinear ODEs, which is given as follows:

$$\dot{\theta} = \omega, \qquad \dot{x} = -\sum_i \frac{\alpha_i^x \omega}{(b_i^x)^2} \Delta\theta_i^x \, exp\left[-\frac{(\Delta\theta_i^x)^2}{2(b_i^x)^2}\right]. \tag{1}$$

Here α_i^x and b_i^x, respectively, are the amplitude and width of the Gaussian functions used to model the ECG, $\theta \in [-\pi, \pi]$ is the *cardiac phase*, $\Delta\theta_i^x = (\theta - \theta_i^x) \bmod 2\pi$, and $\omega = \frac{2\pi h}{60\sqrt{h_{av}}}$ is the *angular velocity*, where h is the *instantaneous* (*beat-to-beat*) *heart rate* in BPM and h_{av} is the mean of the last n heart rates (typically with $n = 6$) normalised by 60 BPM.

To use Eq. (1) one has to define the instantaneous (beat-to-beat) heart rate function $h(t)$ ($t \in \mathbb{R}_{\geqslant 0}$), which specifies the distance between two consecutive R-events (highest peak in Fig. 4). Technically, it is equivalent to the so called *RR-series* $\chi(n)$, $n \in \{1, \dots, N\}$, where N denotes the length of the series. The value of $\chi(n)$ denotes the time between two consecutive heart beats. The RR-series can be generated by constructing the power spectrum $S(f)$ as a sum of two Gaussian distributions

$$S(f) = \frac{\sigma_1^2}{\sqrt{2\pi c_1^2}} e^{(\frac{(f-f_1)^2}{2c_1^2})} + \frac{\sigma_2^2}{\sqrt{2\pi c_2^2}} e^{(\frac{(f-f_2)^2}{2c_2^2})},$$

which have means f_1 with power σ_1, f_2 with power σ_2 and standard deviations c_1 and c_2 respectively. The RR-series $\chi(n)$ is obtained by taking the inverse Fourier transform of $S(f)$. More details on the construction of the function $h(t)$ can be found in [23].

4.2. The cardiac cell heart model

This heart model is based on modelling the ECS of the heart (see Section 2). The ECS (see Fig. 1) is a network of nerves whose role is to propagate the AP through the heart tissue. Modelling every single cell of the ECS is computationally intensive. Thus, we abstract the conduction system as a network of cells in order to achieve a good trade-off between the complexity of the model and the running time of the experiments. In our experiments each cell is connected to neighbouring cells, forming a graph of 33 cells as shown in Fig. 1. The ECS of the heart consists of conduction pathways with different

Please cite this article in press as: T. Chen et al., Quantitative verification of implantable cardiac pacemakers over hybrid heart models, Inform. and Comput. (2014), http://dx.doi.org/10.1016/j.ic.2014.01.014

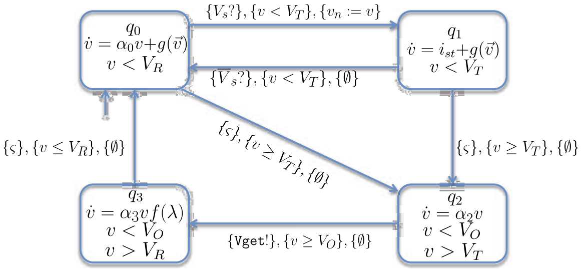
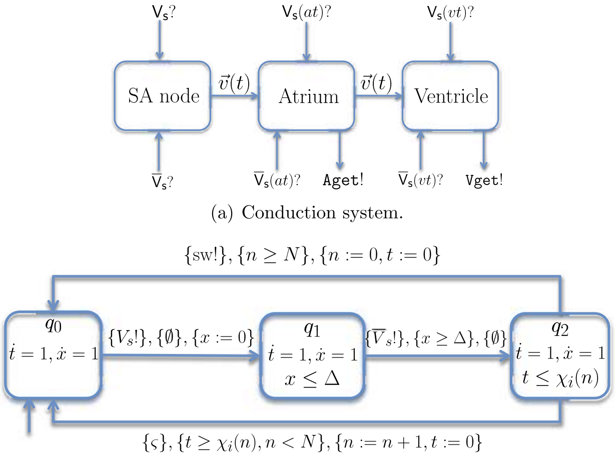

q_0
$\dot{v} = \alpha_0 v + g(\vec{v})$
$v < V_R$
$\{V_s?\}, \{v < V_T\}, \{v_n := v\}$
q_1
$\dot{v} = i_{st} + g(\vec{v})$
$v < V_T$
$\{\overline{V}_s?\}, \{v < V_T\}, \{\emptyset\}$
$\{\varsigma\}, \{v \le V_R\}, \{\emptyset\}$
$\{\varsigma\}, \{v \ge V_T\}, \{\emptyset\}$
$\{\varsigma\}, \{v \ge V_T\}, \{\emptyset\}$
q_3
$\dot{v} = \alpha_3 v f(\lambda)$
$v < V_O$
$v > V_R$
$\{\texttt{Vget!}\}, \{v \ge V_O\}, \{\emptyset\}$
q_2
$\dot{v} = \alpha_2 v$
$v < V_O$
$v > V_T$

Fig. 5. Hybrid automaton for a ventricular cardiac cell.


$\mathsf{V_s}?$
$\mathsf{V_s}(at)?$
$\mathsf{V_s}(vt)?$
SA node
$\vec{v}(t)$
Atrium
$\vec{v}(t)$
Ventricle
$\overline{\mathsf{V}}_\mathsf{s}?$
$\overline{\mathsf{V}}_\mathsf{s}(at)?$
Aget!
$\overline{\mathsf{V}}_\mathsf{s}(vt)?$
Vget!
$\{\text{sw!}\}, \{n \ge N\}, \{n := 0, t := 0\}$
q_0
$\dot{t} = 1, \dot{x} = 1$
$\{V_s!\}, \{\emptyset\}, \{x := 0\}$
q_1
$\dot{t} = 1, \dot{x} = 1$
$x \le \Delta$
$\{\overline{V}_s!\}, \{x \ge \Delta\}, \{\emptyset\}$
q_2
$\dot{t} = 1, \dot{x} = 1$
$t \le \chi_i(n)$
$\{\varsigma\}, \{t \ge \chi_i(n), n < N\}, \{n := n + 1, t := 0\}$

(a) Conduction system.

(b) SA node stimulation automaton.

Fig. 6. Electrical conduction system (ECS) model.

conduction delays. Cells are connected by pathways. The delays of the pathways depend on the physiology of the tissue considered. Moreover, it is possible to use the pathway delays to model various known tissue diseases.

Our model consists of the SA node, whose role is to generate sequences of AP signals which are propagated through the ECS of the heart, and 32 cells that share similar properties.

The cell model in Fig. 5, taken from [27], consists of four (discrete) modes, each associated with an AP phase: *resting and final repolarisation* (q_0), *stimulated* (q_1), *upstroke* (q_2), and *plateau and early repolarisation* (q_3). The variables of the model are: v, which is the membrane voltage to control mode switches; a restitution-related variable v_n, which is used to modify the next ERP phase upon a new round of excitation; and i_{st}, which is the stimulus current. Notice that the variable v_n serves as the "memory". This is crucial to capture the proper response of AP to pacing frequency, which is an essential feature of cardiac excitation. Accordingly, [27] defines the parameter $\lambda = \frac{v_n}{V_R}$, where V_R is a model-specific constant called *repolarisation voltage*, and incorporates the function $f(\lambda) = 1 + 13\sqrt[6]{\lambda}$ into mode q_3 (see Fig. 5).

Given N cells, $\vec{v}$ is the vector of all membrane voltages such that, for $i \leqslant N$, v_i denotes the voltage of cell i. In the model, we also have the function $g(\vec{v})$ denoting the voltage contribution from the neighbouring cells. Let $N - 1$ be the total number of cells connected to the current cell k. The function $g_k(\vec{v})$ for the k'th cell is defined as:

$$g_k(\vec{v}) = \sum_{i=1,\, i \neq k}^{N} v_i(t - \delta_{ki}) \cdot a_{ki} - v_k \cdot d_k, \tag{2}$$

where a_{ki} is the gain applied to the potential v_i from cell i, δ_{ki} is the time it takes for the potential to reach cell k, and d_k is the distance coefficient. These coefficients depend on the conduction system, and in particular the conduction delays.

The mode invariants, defined according to Definition 1, are given by linear inequalities describing the AP. They depend on three model-specific constants: threshold voltage V_T, overshoot voltage V_O, and repolarisation voltage V_R. Initially, the cell starts in q_0. When (externally) stimulated by the event $\mathsf{V_s}?$ (input action), it enters the *stimulated* mode (q_1) and updates its voltage according to the stimulus current (i_{st}). Upon termination of the stimulation, via event $\overline{\mathsf{V}}_\mathsf{s}?$ (input action), with a sub-threshold voltage, the cell returns to resting without firing an AP. If the stimulus is supra-threshold, i.e., $v \geqslant V_T$ holds, the excited cell will generate an AP by progressing to mode *upstroke* (q_2). From the *upstroke* mode the cell will go to the *plateau and ER* mode (q_3) generating the event `Vget!` (output action). Then the cell transitions to mode *resting and FR* (q_0).

In Fig. 6(a) we depict three blocks representing the connection of cells in the ECS. The atrium block consists of 14 cells and the ventricle block consists of 18 cells. The connections of these cells are illustrated in Fig. 1, where atrium cells are

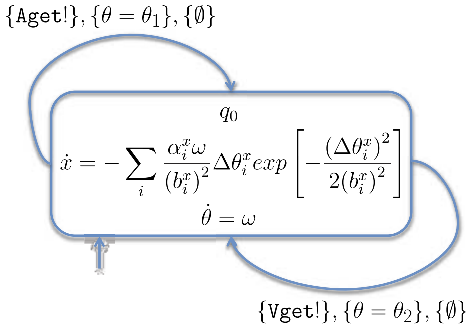

$\{\mathtt{Aget!}\}, \{\theta = \theta_1\}, \{\emptyset\}$
q_0
$\dot{x} = -\sum_i \frac{\alpha_i^x \omega}{(b_i^x)^2} \Delta\theta_i^x exp\left[-\frac{(\Delta\theta_i^x)^2}{2(b_i^x)^2}\right]$
$\dot{\theta} = \omega$
$\{\mathtt{Vget!}\}, \{\theta = \theta_2\}, \{\emptyset\}$

Fig. 7. ECG hybrid automaton.

```
Pick $i \in \{1, 2, 3\}$ according to $\alpha$;
while $k < M$ do
  if sw? then
    Pick a $j$ according to $\mathbf{P}_i$;
    $i := j$; $k := k + 1$;
    Use $\chi_i$ as the new RR-series (see Fig. 6(b));
  end if
end while
```

Fig. 8. Mode switching.

shown in the upper part of the figure, while ventricle cells are in the lower part. Every cell in the atrium and the ventricle can be stimulated by the pacemaker using the input actions $\mathsf{V_s}(at)?$, $\overline{\mathsf{V}}_\mathsf{s}(at)?$ and $\mathsf{V_s}(vt)?$, $\overline{\mathsf{V}}_\mathsf{s}(vt)?$, respectively. The output actions $\mathtt{Aget}!$ and $\mathtt{Vget}!$ notify the pacemaker that the AP in the atrium and the ventricle (where the pacemaker leads are inserted) have reached a given threshold. The function $\vec{v}(t)$ is the output voltage from a given cell.

Recall that the SA node is the self-firing cell of the heart, stimulated by the central nervous system. The frequency of the stimulation is given by the hybrid automaton depicted in Fig. 6(b). The main parameter of this automaton is the function $\chi_i(n)$, $i \in \{1, 2, 3\}$, which represents the RR-series. Here the index i denotes three different types of RR-series corresponding to Normal, Tachycardia and Bradycardia heart rhythms. The automaton starts in mode q_0. Then it moves to mode q_1 by generating an output action $\mathsf{V_s}!$ representing the start of the stimulus. In mode q_1 the automaton waits for Δ units of time (which denotes the width of the stimulus) before moving to mode q_2 and generating the end of the stimulus action $\overline{\mathsf{V}}_\mathsf{s}!$. In mode q_2 the automaton waits for at most $\chi_i(n) - \Delta$ units of time. There are two possible transitions from mode q_2. When $n < N$, the value of n is incremented and a new value of the RR-series is chosen. When the automaton reaches the end of the RR-series, i.e., $n \geqslant N$, a new type of RR-series is picked by generating the action $\mathsf{sw}!$.

4.3. *Mapping from ECG to action potential*

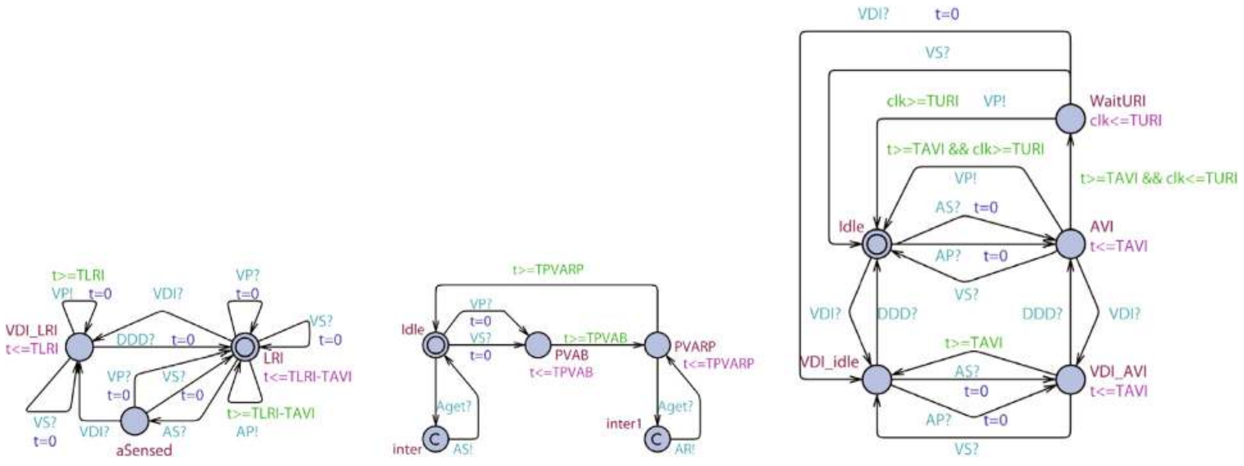
We show how to map from an ECG to the atrium and ventricle AP signals. In our framework, the mapping can be implemented as a hybrid automaton which is depicted in Fig. 7.

Here θ_1 represents the beginning of the P wave, whereas θ_2 represents the beginning of the Q wave. By this mapping we can create a model consisting of two cardiac cells (one for the atrium and one for the ventricle) in which the propagation delay δ (see Eq. (2)) is proportional to the angular velocity ω. We do not use the value of the $x(t)$ function in the current analysis based on the ECG heart model. However, the function of $x(t)$ is crucial when one wants to define a patient-specific heart model. In this case, from the given patient ECG data one can learn the parameters of the function $x(t)$ [7]. This is one of the advantages of the ECG model compared to the cardiac cell model.

4.4. *Switching between different heart behaviours*

The two heart models described above can exhibit only a single heart behaviour, such as Normal, Bradycardia or Tachycardia. However, a real human heart exhibits several spontaneous behavioural changes. In this paper we consider three of them: (1) the malfunction of the SA node; (2) the malfunction of certain cardiac cells; and (3) the malfunction of the conduction system.

Let χ_i be the set of RR-series, $\alpha \in \mathrm{Distr}(\{1, 2, 3\})$ be an initial distribution and let $\mathbf{P}_i \in \mathrm{Distr}(\{1, 2, 3\})$, for $i \in \{1, 2, 3\}$, denote the transition probabilities between different heart modes. The malfunctioning of the SA node can be modelled through the mode switching algorithm in Fig. 8. We remark that the initial distribution and the transition probabilities can be learned from patient data, although we do not use personalised patient data in this paper. The other heart malfunctions can be modelled by varying the cell model parameters.



(a) LRI component (b) PVARP component (c) AVI component

Fig. 9. LRI, PVARP, AVI components.

5. Pacemaker model

The pacemaker is implanted under the chest skin and sends impulses to the heart at specific time intervals. It has two leads: one for the atrium and one for the ventricle. Each lead has the ability to sense or deliver an electrical signal. As mentioned, the authors in [15] develop a pacemaker model based on timed automata (TAs) which we use to validate our framework. For completeness of the presentation, below we reproduce this model and its variants developed for our framework.

The pacemaker model in [15] consists of five basic TA components (Fig. 9, Fig. 10(a)) and additional three advanced components (Fig. 10(b), Fig. 10(c)). The basic components are: the lower rate interval (LRI) component, the atrio-ventricular interval (AVI) component, the upper rate interval (URI) component, the post ventricular atrial refractory period (PVARP) component and the ventricular refractory period (VRP) component. The LRI component (see Fig. 9(a)) has the function of keeping the heart rate above a given minimum value. The AVI component (see Fig. 9(c)) has the purpose to maintain the synchronisation between the atrial and the ventricular events. An event is when the pacemaker senses or generates an action. The AVI component also defines the longest interval between an atrial event and a ventricular event. The PVARP component (see Fig. 9(b)) notifies all other components that an atrial event has occurred. Notice that there is no AR signal in the PVARP and Interval components as we are not using the advanced algorithms given in [15]. The URI component (see Fig. 10(a)) sets a lower bound on the times between consecutive ventricular events. The VRP component (see Fig. 10(a)) filters noise and early events that may cause undesired behaviour.

The advanced components, Interval, Counter and Duration, are used for detection and correction of pacemaker mediated Tachycardia. The components switch the functioning modes of the pacemaker from DDD (pacing and sensing of the atrium and ventricle) to VDI (pacing and sensing only the ventricle). More details will be given in Section 6.3.6. Note that in all components the locations labelled with **C** do not allow time to elapse.

There are four actions in the pacemaker model that will be considered in the remainder of the paper. The input actions Aget? and Vget? will notify the pacemaker when there is an AP from the atrium or from the ventricle, respectively. The output actions AP! and VP! are responsible for pacing the atrium and the ventricle, respectively. After receiving an AP? event, the component in Fig. 11 generates a cell stimulus of duration Δ_a using the two actions $\mathrm{V_s}(at)!$ and $\overline{\mathrm{V}}_\mathrm{s}(at)!$. There is a similar component for the ventricle stimulus generation.

5.1. Enhanced pacemaker model

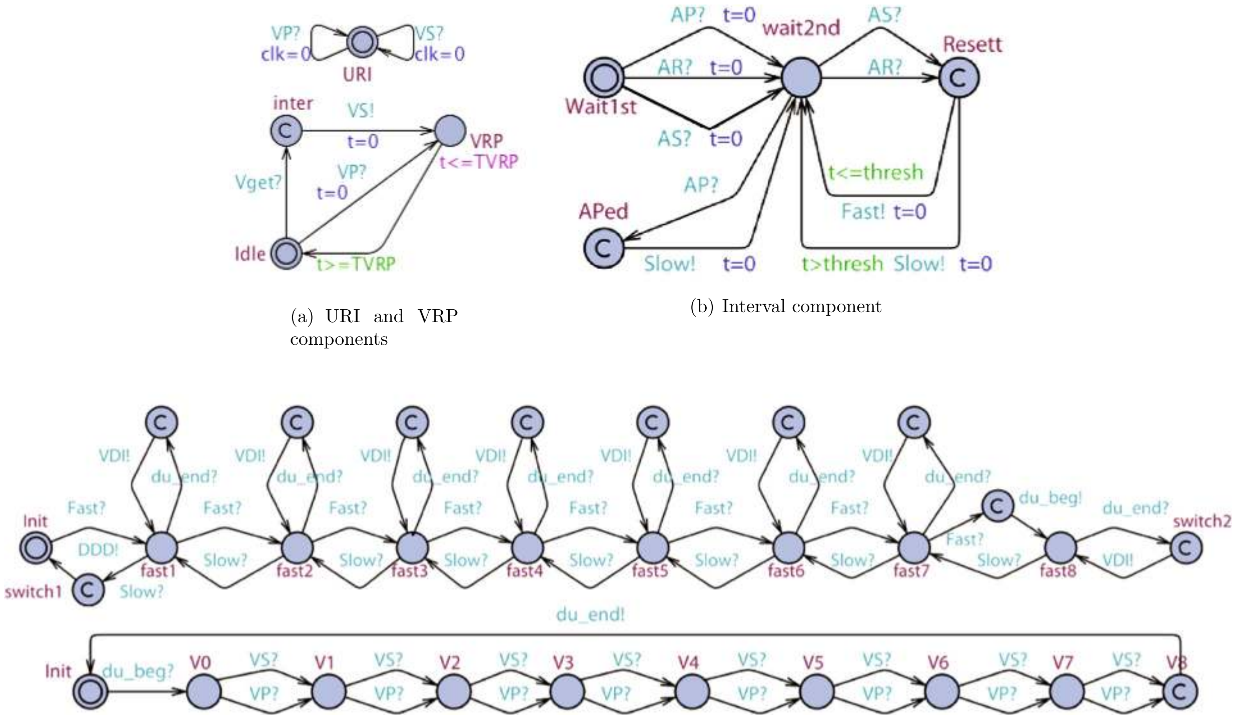
In this section, we enhance the pacemaker model in [15] by considering noise and energy consumption.

5.1.1. Pacing noise

One of the important design issues of pacemakers is the need to tolerate noise. For instance, when the pacemaker tries to deliver a beat, the beat might get lost due to noise on the channel. For the pacemaker model presented in [15] the assumption is that the pacemaker can pace the heart perfectly. This can simplify the modelling considerably, but is not realistic.

To remedy this, in this paper we consider so called “failure-to-capture”, which in practice is equivalent to insufficient contact between the lead and the myocardium, or lead fracture [9]. In particular, we add to the fixed stimulus current i_{st} (cf. Fig. 5) a *normally distributed noise* with mean μ and variance σ^2 each time the pacemaker wants to pace the cell. Intuitively, if the noise added to the channel is too high, the stimulus from the pacemaker will *not* be high enough to stimulate the

Please cite this article in press as: T. Chen et al., Quantitative verification of implantable cardiac pacemakers over hybrid heart models, Inform. and Comput. (2014), http://dx.doi.org/10.1016/j.ic.2014.01.014



(a) URI and VRP components

(b) Interval component

(c) Counter and Duration component

Fig. 10. URI, VRP, Interval, Counter and Duration component for PMT analysis.

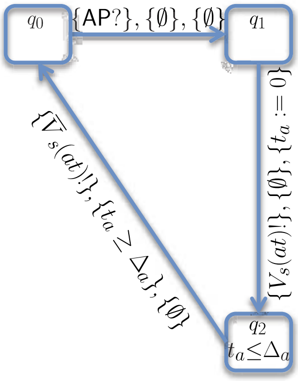

q_0
$\{AP?\}, \{\emptyset\}, \{\emptyset\}$
q_1
$\{V_s(at)!\}, \{\emptyset\}, \{t_a := 0\}$
q_2
$t_a \leq \Delta_a$
$\{\overline{V}_s(at)!\}, \{t_a \geq \Delta_a\}, \{\emptyset\}$

Fig. 11. Atrium pacing.

cell. In such a case that stimulus is considered “missing”. We remark that there exist different causes of noise that affect pacemakers, for instance lead displacement, which we leave as future work.

5.1.2. Energy

Pacemaker’s life time is limited and is crucially dependent on the battery embedded into the devices. The pacemaker must be re-implanted when the battery depletes, and hence energy usage analysis is indispensable.

We use the Kinetic Battery Model (KiBaM) [22] to describe energy consumption of the pacemaker over time. The model is given as a system of ODEs:

$$\frac{dy_1(t)}{dt} = -\iota(t) + k\left(\frac{y_2(t)}{1-c} - \frac{y_1(t)}{c}\right), \qquad \frac{dy_2(t)}{dt} = -k\left(\frac{y_2(t)}{1-c} - \frac{y_1(t)}{c}\right). \tag{3}$$

The battery charge is distributed in two wells: *available-charge* $y_1(t)$ and *bound-charge* $y_2(t)$. The function $\iota(t)$ denotes the current applied to the battery. When the value of $\iota(t)$ is zero the battery enters the recovery mode, where the energy from the bound-charge well flows to the available-charge well. The recovery effect of the battery allows a nearly discharged battery to recover in a period of zero or low current by increasing its available-charge. When the current $\iota(t)$ is not zero, both $y_1(t)$ and $y_2(t)$ decrease over time. If C [Ah] (ampere-hour) is the initial *total capacity* of the battery then $y_1(0) = c \cdot C$ and $y_2(0) = (1 - c) \cdot C$, where c is a fraction of the total charge. The *conduction* parameter k represents the flow rate of charge from the bound-charge well to the available-charge well. The battery is considered to be empty when there is no charge in the available-charge well, i.e., $y_1(t) = 0$.

We compose the KiBaM with the pacemaker model as follows. The pacemaker has in total eight components. Each of them uses $\iota_j(t)$ µA (micro-ampere), $j \in \{1, \dots, 8\}$. The total current applied at any time to the battery will be $\iota(t) = \sum_{j=1}^{8} \iota_j(t)$. When the pacemaker is in the aSensed state of the LRI component, Idle or VDI_idle states of the AVI component, Idle state of the PVARP component, URI state of the URI component, Idle state of the VRP component, Wait1st or wait2nd states of the Interval component, Init or all fast states of the Counter component, and any states except V8 of the Duration component, then the respective current $\iota_j(t)$ is zero. Technically, when the pacemaker is in the sensing mode or the idle mode, the current applied to the battery is very small. On the other hand, when the pacemaker is pacing, counting the duration between successive pacing events or sending a signal, the current applied to the battery increases.

6. Verification methods and experimental results

In this section, we describe our implementation of the pacemaker verification framework and present experimental results. First, we identify property patterns tailored to the quantitative verification of pacemakers. Next, we instantiate the framework with the cardiac cell model and the (enhanced) pacemaker model described in the previous section, and summarise the results of the verification for a broad range of properties, including known physiological scenarios that are problematic for pacemaker designs.

6.1. Property patterns

In this section we present two important property patterns which are used to analyse the pacemaker. The first one specifies the key safety property of the pacemaker, namely, whether it maintains the number of heart beats in the normal range of 60–100 BPM, and the second specifies whether the energy consumed by the pacemaker at a given time point is less than a specific value.

Definition 4 *(Duration of a path).* Given a discrete path σ (see Definition 2) and a simulation step h, we define the *duration* of σ in milliseconds as $\mathtt{Dur}(\sigma) = h \cdot |\sigma|$.

Definition 5 *(Heart beats).* Given a discrete path σ, we define the number of heart beats of σ as $\mathtt{Heart\_beats}(\sigma) = \sum_{i=0}^{|\sigma|-1} \mathbb{1}(\sigma, i)$, where $\mathbb{1}(\sigma, i)$ is the characteristic function of transitions such that $\mathbb{1}(\sigma, i) = 1$ if $\sigma[i] = (q_2, \cdot)$ and $\sigma[i+1] = (q_3, \cdot)$, and 0 otherwise. Here q_2 and q_3 refer to the upstroke and plateau modes in Fig. 5 respectively.

We are now ready to define “normal paths”, namely, paths corresponding to normal heart behaviours.

Definition 6 *(Normal path property).* Given a discrete path σ we say that σ is *normal* if, for any i, j,

$$\big(\mathtt{Dur}\big(\sigma[i..j]\big) = 1\ \mathtt{minute}\big) \quad \Rightarrow \quad \big(\mathtt{Heart\_beats}\big(\sigma[i..j]\big) \in [60, 100]\big).$$

Definition 7 *(Energy property).* Let σ be a discrete path, T a finite time bound, h the simulation step and V an energy level bound. We define the *time bound energy property* to be satisfied if the following expression is true: $y_1(\lfloor \frac{T}{h} \rfloor \cdot h) \leqslant V \wedge T \leqslant |\sigma| \cdot h$, where $y_1(\cdot)$ is computed by Eq. (3).

Intuitively the energy property pattern checks that the energy at time T is less or equal to V. Both the energy and normal path properties are encoded as deterministic automata where each transition corresponds to h time units.

6.2. Approximate quantitative verification

The complexity of the heart models that we have developed, including non-linearity of the electrical signal and the large number of cells, makes automatic symbolic verification using hybrid automata tools infeasible. An additional complication is that we also model stochastic features, and specifically noise and probabilistic switching. We therefore employ *approximate* quantitative verification methods based on finitely many simulation runs. The derived simulation trajectories are used to estimate the probability of the satisfaction or violation of the specification expressed as a property pattern. Such a method is necessarily approximate, and so the property can only be established up to a given confidence level, but its advantage is that quantitative properties such as expected energy usage can also be handled.

There are a number of approaches that can be applied to estimate the probability of a property being satisfied based on the set of randomly generated paths according to Definition 2. These include statistical model checking based on hypothesis

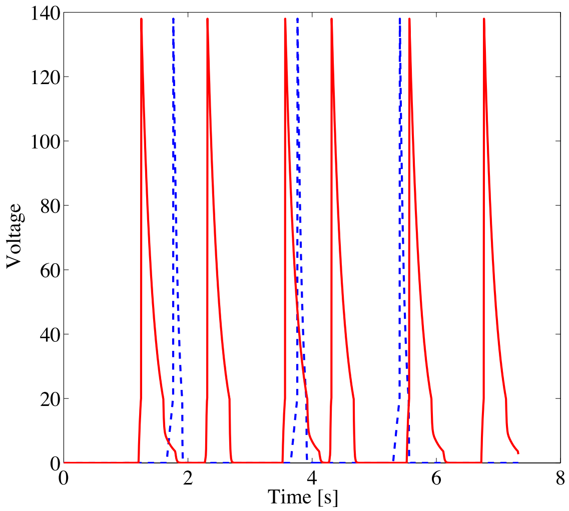


Fig. 12. Bradycardia correction experiment.

testing [29], Bayesian methods [30], and probability estimation [17]. We use the latter method, which we now explain. Let $0 < \varepsilon < 1$ be the error bound, $1 - \delta$ with $0 < \delta < 1$ be the confidence level, T be the time bound and h be the simulation step. Let $k = \frac{T}{h}$ be the discrete path length, p be the probability of all "normal paths" of length k and $N = \log(\frac{2}{\delta})/2\varepsilon^2$ be the number of random paths of length k. Let N' be the number of random paths which are "normal". By simple results from probability theory (mainly the Chernoff bound), one can show that the probability $Prob[|\frac{N'}{N} - p| \leqslant \varepsilon] \geqslant 1 - \delta$, which intuitively means that, with a very high probability (i.e., $1 - \delta$), the probability $\frac{N'}{N}$ that we compute is ε-close to the real probability p. We refer the readers to [17] for further details.

We exploit this approach in our experiments involving heart models with probabilistic switching. It should be emphasised that the verification method introduced in this section is not limited to the property patterns described in Section 6.1, but applies also to other classes of properties, e.g., Metric Temporal Logic or Durational Calculus, that can be checked on paths of finite length.

6.3. Experimental results

We implement both heart models and the pacemaker model in Simulink with the simulation step $h = 3.7$ ms and time bound $T = 1$ min, yielding a path length $k = 16\,217$. These parameters are good enough to capture the physiological behaviour of the heart. We run the experiments on a 2.83 GHz 4 Core(TM)2 Quad CPU with 3.7 GB of memory. All experiments run in less than one hour. The model files can be accessed at http://qav.cs.ox.ac.uk/subm/icsb13.zip.

6.3.1. Bradycardia correction

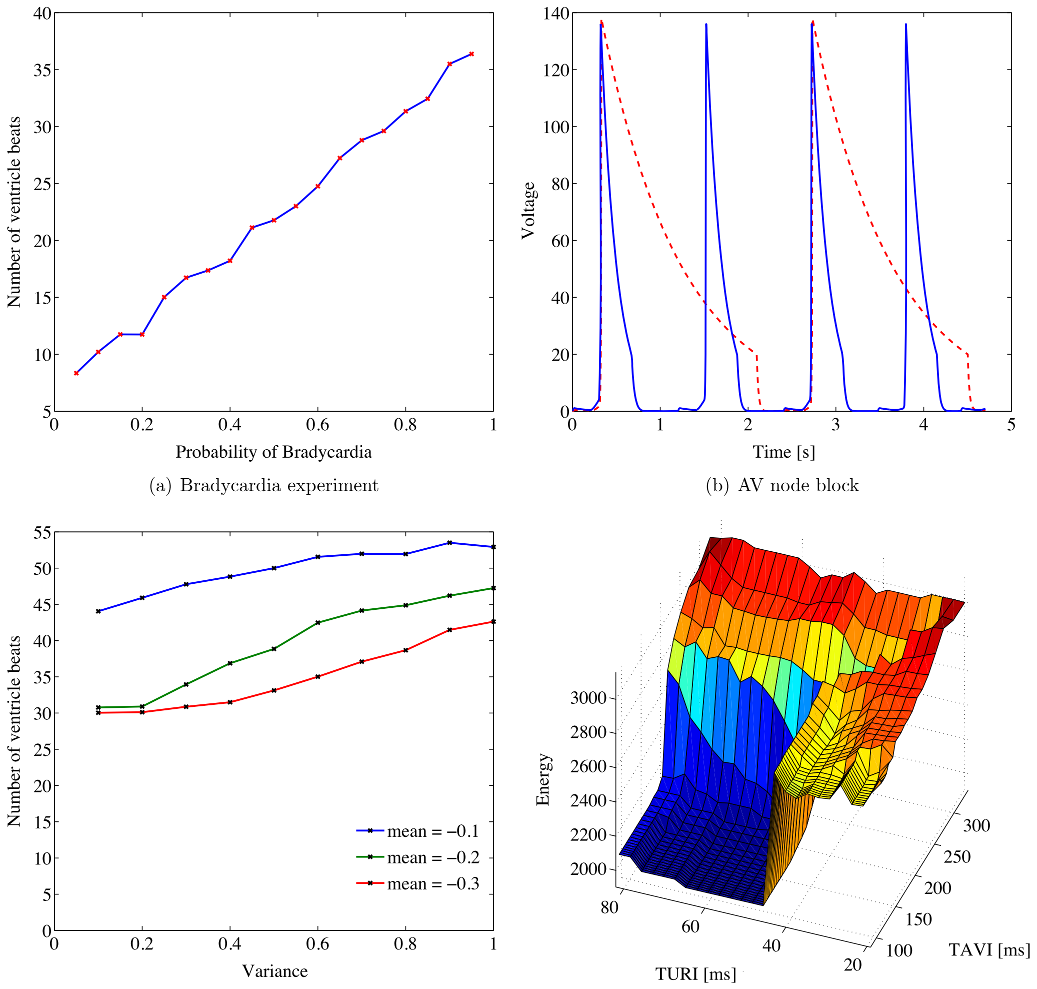
In Fig. 12 we depict two signals. The first one (in blue, dotted line) denotes the AP generated by the SA node. In this scenario the SA node is in the Bradycardia mode. More precisely, we have three beats in six seconds, which is approximately 30 BPM. The second signal (in red, continuous line) denotes the AP from one of the cells situated in the ventricle. This is the signal which is captured and paced by the pacemaker. Note that the pacemaker increases the number of beats per minute by first delivering a beat to the ventricle after approximately one second.

6.3.2. Probabilistic switching

We carry out experiments when the probabilistic switching between different heart behaviours is taken into account. Fig. 13(a) depicts the results on the relationship between the probability to generate Bradycardia and the number of pacemaker beats to the ventricle. We range the probability from 0.05 to 0.95 and run 40 experiments, each representing 8 minutes of the heart beat. As expected, by increasing the probability the pacemaker delivers more beats to the ventricle.

6.3.3. AV node block

In Fig. 13(b) we depict the case when the ERP value of the AV node is long enough, so that it filters out the signal from the SA node. As described in Section 2, a cell cannot be stimulated during its ERP phase. Thus, increasing the ERP value of the AV node results in filtering some of the signal that comes from the atrium. In this case, the SA node signal (in blue, dotted line) is being blocked by a high ERP value of the AV node (signal in red, continuous line). The factor is 2:1 (two beats in the atrium result in one beat in the ventricle). A long ERP value for the AV node induces Bradycardia in the ventricle.



(a) Bradycardia experiment

(b) AV node block

(c) Noise experiments

(d) Battery charge in 1 min period

Fig. 13. Experiments. (For interpretation of the references to colour in this figure, the reader is referred to the web version of this article.)

6.3.4. *Noise*

In this section we describe experiments that take into account noise on sensors. In the experiments that we run to simulate the noise we have 2 parameters: the mean μ and the variance σ^2 of the normally distributed noise. Fig. 13(c) shows the results of the experiments for different values of μ (red line with $\mu = -0.3$, green line with $\mu = -0.2$ and blue line with $\mu = -0.1$). We choose μ as negative in order to simulate the undersensing effect. In each experiment with fixed mean μ, we vary the variance from 0.1 to 1 with step of 0.1. Fig. 13(c) demonstrates that, when the noise with large mean (the red line with mean equal to -0.3 for example) is added to the stimulus, the number of beats in the ventricle decreases. This is due to the fact that more beats induced by the pacemaker will be lost. At the same time, increasing the variance of the normal distribution will produce more beats. The reason is that greater variance to the noise, when centred at negative mean, produces better chances of picking positive samples from the normal distribution. This, in turn, yields a better chance for the stimulus to be high enough to stimulate the cell.

6.3.5. *Energy*

In this section, we analyse energy consumption of the pacemaker. In Fig. 13(d) we depict the pacemaker battery charge in one minute period. In this experiment the SA node induces Bradycardia. We varied two parameters, TAVI and TURI, which are the default programmable parameters used by technicians to ensure a heart beat between 60 and 100 BPM. The value of TAVI is 70–300 ms with 10 ms increment and the value of TURI is 50–175 BPM with 5 BPM increment. All the time values are shown in ms. In Fig. 13(d) there is an energy rise when TURI < 50 or TAVI > 200. This is due to the fact that we are forcing the pacemaker to wait less between two consecutive ventricular events. Thus, the pacemaker will initiate most of the ventricular beats before a natural beat happens. The experimental results confirm our intuition that, by waiting less, the pacemaker will consume more energy, since it paces more frequently.

Please cite this article in press as: T. Chen et al., Quantitative verification of implantable cardiac pacemakers over hybrid heart models, Inform. and Comput. (2014), http://dx.doi.org/10.1016/j.ic.2014.01.014

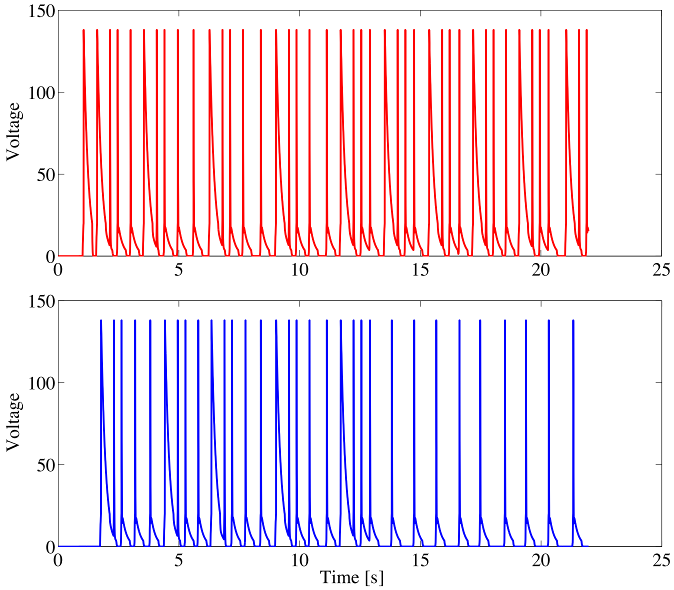


Fig. 14. PMT correction.

6.3.6. *Pacemaker mediated Tachycardia*

As mentioned before, in some cases the pacemaker can increase the heart rate inappropriately (i.e., pacemaker mediated Tachycardia), which is considered to be unsafe. In this section, we show that this scenario can be modelled in our framework. We then verify the advanced pacemaker model which can correct PMT (see Section 5).

In human hearts, the atrium can beat faster than the ventricle, at ratio 2:1 or 3:1. The resulting heart beat can still be regular due to a special cell called the AV node which has a refractory period longer than the other cells. The AV node connects the ECS of the atrium to the ECS of the ventricle. The pacemaker tries to maintain a 1:1 AV conduction through the AVI component. Thus, in the event of PMT, the pacemaker increases the beats in the ventricle inappropriately. In order to avoid this behaviour we need to switch the pacemaker from the DDD mode to the VDI mode after the PMT is detected. After a normal heart beat is re-established, the pacemaker can switch back to the DDD mode.

To accomplish this result we generate Tachycardia and assign a longer ERP value to the AV node. The pacemaker algorithm detects the PMT behaviour in the atrium. Then, after confirmed detection, the pacemaker switches from the DDD mode to VDI mode. During the VDI mode, the AV synchrony function of the pacemaker is deactivated, and thus the ventricular rate is decoupled from the fast atrial rate. When the components detect the end of the PMT, the pacemaker switches back to the DDD mode.

For our experiments, using the notation introduced in Section 6, we picked $\varepsilon = 0.01$ and $\delta = 0.01$ yielding $N = 11\,505$ sample paths. The advanced pacemaker model is capable of correcting $N' = 9017$ of them. Intuitively, this means that with confidence of 99% we are sure that the computed probability $\frac{N'}{N} = 0.783$ is within 0.01 radius from the real probability p. In Fig. 14 we show an example containing two graphs. The first graph depicts Tachycardia in the ventricle due to PMT. The second graph shows how the pacemaker switches its mode from DDD to VDI at time 13. As a result, the number of ventricle beats decreases.

7. Conclusion

In this paper, we proposed a generic model-based framework for quantitative analysis of pacemaker software that incorporates stochasticity to model probabilistic switching and noise. The framework can be instantiated with a hybrid automaton model for embedded pacemaker software and a hybrid heart model. We implemented the framework in Simulink based on discrete-time simulation semantics and endowed it with a range of quantitative property checks tailored to the verification of pacemakers and expressed as property patterns. The feasibility of automated quantitative verification for pacemakers was demonstrated by employing approximate verification that estimates the probability or expectation of a given property based on simulation runs up to a given confidence interval. Our methods are currently not practical for small confidence intervals and error bounds due to the high computational complexity of simulating sufficiently many paths.

For future work, we plan to represent the pacing noise by means of more realistic models. In this paper, we assume that the pacing noise can be modelled by a normal distribution. However, one can consider additional parameters, such as the influence of the body temperature, vibrations, or the position of the pacing lead. Similarly, we would like to consider extending the pacemaker specification with more detailed energy consumption parameters, compared to the rudimentary parameterisation afforded by the current battery model that we use. Additionally, we plan to explore the parameter synthesis problem for the pacemakers and, in the longer-term, employ symbolic model checking methods for hybrid automata. Another interesting future direction is to enhance the pacemaker specification by considering additional signals, e.g., blood pressure and temperature.

Acknowledgment

This work is supported by the ERC Advanced Grant VERIWARE and the Institute for the Future of Computing, Oxford Martin School.

References

[1] R. Alur, A. Kanade, S. Ramesh, K.C. Shashidhar, Symbolic analysis for improving simulation coverage of Simulink/Stateflow models, in: EMSOFT, ACM, 2008, pp. 89–98.

[2] E. Bartocci, F. Corradini, M.R.D. Berardini, E. Entcheva, S.A. Smolka, R. Grosu, Modeling and simulation of cardiac tissue using hybrid I/O automata, Theor. Comput. Sci. 410 (2009) 3149–3165.

[3] G. Behrmann, A. David, K.G. Larsen, J. Håkansson, P. Pettersson, W. Yi, M. Hendriks, UPPAAL 4.0, in: QEST, IEEE Computer Society, 2006, pp. 125–126.

[4] T. Chen, M. Diciolla, M. Kwiatkowska, A. Mereacre, Quantitative verification of implantable cardiac pacemakers, in: RTSS, IEEE Computer Society, 2012, pp. 263–272.

[5] T. Chen, M. Diciolla, M.Z. Kwiatkowska, A. Mereacre, A Simulink hybrid heart model for quantitative verification of cardiac pacemakers, in: C. Belta, F. Ivancic (Eds.), HSCC, ACM, 2013, pp. 131–136.

[6] G. Clifford, S. Nemati, R. Sameni, An artificial vector model for generating abnormal electrocardiographic rhythms, Physiol. Meas. 31 (2010) 595–609.

[7] G.D. Clifford, F. Azuaje, P. McSharry, Advanced Methods and Tools for ECG Data Analysis, Artech House Publishers, 2006.

[8] A.O. Gomes, M.V. Oliveira, Formal specification of a cardiac pacing system, in: Proceedings of FM '09, Springer-Verlag, Berlin, Heidelberg, 2009, pp. 692–707.

[9] S. Greenhut, J. Jenkins, R. MacDonald, A stochastic network model of the interaction between cardiac rhythm and artificial pacemaker, IEEE Trans. Biomed. Eng. 40 (1993) 845–858.

[10] R. Grosu, G. Batt, F.H. Fenton, J. Glimm, C.L. Guernic, S.A. Smolka, E. Bartocci, From cardiac cells to genetic regulatory networks, in: G. Gopalakrishnan, S. Qadeer (Eds.), CAV, Springer, 2011, pp. 396–411.

[11] R. Grosu, S.A. Smolka, F. Corradini, A. Wasilewska, E. Entcheva, E. Bartocci, Learning and detecting emergent behavior in networks of cardiac myocytes, Commun. ACM 52 (2009) 97–105.

[12] E. Jee, S. Wang, J.K. Kim, J. Lee, O. Sokolsky, I. Lee, A safety-assured development approach for real-time software, in: RTCSA, 2010, pp. 133–142.

[13] Z. Jiang, M. Pajic, A. Connolly, S. Dixit, R. Mangharam, Real-time heart model for implantable cardiac device validation and verification, in: ECRTS, IEEE Computer Society, 2010, pp. 239–248.

[14] Z. Jiang, M. Pajic, R. Mangharam, Cyber-physical modeling of implantable cardiac medical devices, Proc. IEEE 100 (2012) 122–137.

[15] Z. Jiang, M. Pajic, S. Moarref, R. Alur, R. Mangharam, Modeling and verification of a dual chamber implantable pacemaker, in: C. Flanagan, B. König (Eds.), TACAS, Springer, 2012, pp. 188–203.

[16] B. Kim, A. Ayoub, O. Sokolsky, I. Lee, P.L. Jones, Y. Zhang, R.P. Jetley, Safety-assured development of the GPCA infusion pump software, in: S. Chakraborty, A. Jerraya, S.K. Baruah, S. Fischmeister (Eds.), EMSOFT, ACM, 2011, pp. 155–164.

[17] R. Lassaigne, S. Peyronnet, Probabilistic verification and approximation, Ann. Pure Appl. Log. 152 (2008) 122–131.

[18] J. Lian, H. Krätschmer, D. Müssig, Open source modeling of heart rhythm and cardiac pacing, The Open Pacing, Electrophysiol. & Therapy J. (2010) 28–44.

[19] A.T. Luu, M.C. Zheng, Q.T. Tho, Modeling and verification of safety critical systems: a case study on pacemaker, in: SSIRI, IEEE Computer Society, 2010, pp. 23–32.

[20] N.A. Lynch, R. Segala, F.W. Vaandrager, H.B. Weinberg, Hybrid I/O automata, in: R. Alur, T.A. Henzinger, E.D. Sontag (Eds.), Hybrid Systems, Springer, 1995, pp. 496–510.

[21] H.D. Macedo, P.G. Larsen, J.S. Fitzgerald, Incremental development of a distributed real-time model of a cardiac pacing system using VDM, in: J. Cuéllar, T.S.E. Maibaum, K. Sere (Eds.), FM, Springer, 2008, pp. 181–197.

[22] J.F. Manwell, J.G. McGowan, Lead acid battery storage model for hybrid energy systems, Sol. Energy 50 (1993) 399–405.

[23] P. McSharry, G. Clifford, L. Tarassenko, L. Smith, A dynamical model for generating synthetic electrocardiogram signals, IEEE Trans. Biomed. Eng. 50 (2003) 289–294, http://dx.doi.org/10.1109/TBME.2003.808805.

[24] D. Méry, N.K. Singh, Pacemaker's functional behaviors in Event-B, rapport de recherche, http://hal.inria.fr/inria-00419973, 2009.

[25] M. Pajic, Z. Jiang, I. Lee, O. Sokolsky, R. Mangharam, From verification to implementation: A model translation tool and a pacemaker case study, in: M.D. Natale (Ed.), IEEE Real-Time and Embedded Technology and Applications Symposium, IEEE, 2012, pp. 173–184.

[26] S. Sankaranarayanan, G.E. Fainekos, Simulating insulin infusion pump risks by in-silico modeling of the insulin–glucose regulatory system, in: D. Gilbert, M. Heiner (Eds.), CMSB, Springer, 2012, pp. 322–341.

[27] P. Ye, E. Entcheva, R. Grosu, S.A. Smolka, Efficient modeling of excitable cells using hybrid automata, in: CMSB, 2005, pp. 216–227.

[28] P. Ye, E. Entcheva, S.A. Smolka, R. Grosu, Modelling excitable cells using cycle-linear hybrid automata, IET Syst. Biol. 2 (2008) 24–32.

[29] H.L.S. Younes, M.Z. Kwiatkowska, G. Norman, D. Parker, Numerical vs. statistical probabilistic model checking: An empirical study, in: K. Jensen, A. Podelski (Eds.), TACAS, Springer, 2004, pp. 46–60.

[30] P. Zuliani, A. Platzer, E.M. Clarke, Bayesian statistical model checking with application to Simulink/Stateflow verification, in: K.H. Johansson, W. Yi (Eds.), HSCC, ACM, 2010, pp. 243–252.