Quantitative Verification of Implantable Cardiac Pacemakers

Taolue Chen Marco Diciolla Marta Kwiatkowska Alexandru Mereacre

Department of Computer Science, University of Oxford, United Kingdom
{firstname.lastname}@cs.ox.ac.uk

***Abstract*—Implantable medical devices, such as cardiac pacemakers, must be designed and programmed to the highest levels of safety and reliability. Recently, errors in embedded software have led to a substantial increase in safety alerts, costly device recalls or even patient death. To address such issues, we propose a model-based framework for quantitative, automated verification of pacemaker software. We adapt the electrocardiogram model of Clifford *et al*, which generates realistic normal and abnormal heart beat behaviours, with probabilistic transitions between them, to produce a timed sequence of action potential signals that serve as pacemaker input. Working with the timed automata model of the pacemaker by Jiang *et al*, we develop a methodology for deriving the composition of the heart and the pacemaker, based on discretisation. The main correctness properties we consider include checking that the pacemaker corrects Bradycardia (slow heart beat) and does not induce Tachycardia (fast heart beat), for a range of realistic heart behaviours. We also analyse undersensing, through considering noise on sensor readings, and energy usage. We implement the framework using the probabilistic model checker PRISM and MATLAB and demonstrate encouraging experimental results. Our approach can be adapted to individual patients and is applicable to other pacemaker models.**

***Keywords*-Verification; Timed Automata; Probabilistic and Hybrid Models; Implantable Pacemakers**

I. Introduction

Implantable cardiac pacemakers (pacemakers for short) are the most commonly used cardiac rhythm management devices. The primary purpose of a pacemaker is to maintain an adequate heart rate through electrical impulses delivered to the heart. Nowadays, millions of implantable pacemakers are used worldwide. Because of safety-critical implications, they must be designed and programmed to the highest levels of safety and reliability. Unfortunately, according to the US Food and Drug Administration (FDA) [1], errors in embedded software have led to a substantial increase in safety alerts, costly device recalls or even patient death. Combined with the relative lack of standardisation in the field of medical devices, there is an urgent need to develop methodologies for ensuring correct behaviour of embedded pacemaker software.

Recently, Jiang *et al* [8] developed a model-based framework for automatically verifying cardiac pacemakers in the real-time setting. Working from descriptions by Boston Scientific, a leading manufacturer, [8] develop a detailed model of a basic dual chamber pacemaker as a network of timed automata ([2], TAs). They also provide a model of the heart behaviour as a TA, and perform verification of the composition in UPPAAL [10]. While the pacemaker model of [8] is rather extensive, their (random) heart model is arguably too simplistic, since it generates signals non-deterministically within a real-time interval. As a consequence, the model admits unrealistic and extremely irregular heart behaviours, which would hinder the verification of certain correctness properties. Indeed, it is explicitly mentioned in [8] that a more physiologically-relevant heart model is needed for more complex properties.

In this paper we contribute to the effort initiated in [8], aimed at developing a model-based verification methodology for cardiac pacemakers, and specifically address the issue of a realistic heart model, as well as their suggestion that incorporating costs and probabilities could lead to a more realistic and detailed quantitative analysis. We adapt a sophisticated heart model based on nonlinear ordinary differential equations (ODEs) from medical engineering due to Clifford *et al* [4]. Developed for different purposes, this is an artificial model for generating multi-channel electrocardiogram (ECG) to provide simulations of normal and abnormal cardiac rhythms. The electrical activity of the heart can be monitored externally by an ECG, which measures the voltage difference between two leads placed on the skin of the torso and provides a general view of the electrical activities of the heart. The heart model from [4] uses a three dimensional vectorcardiogram (VCG) formulation to generate the normal and/or abnormal cardiac dipole for a patient using a sum of Gaussian kernels, fitted to real VCG recordings. Switching between normal and abnormal beat types is achieved using a Markov chain. Transition probabilities of the Markov chain can be learned from patient data, and hence adapted to an individual.

The main idea is to convert the ECG signals to action potential signals used as input to the pacemaker, which can then be composed with the pacemaker model for analysis, similarly to the approach of [8]. However, since the heart model can probabilistically switch to different mode behaviours, we need to develop quantitative, probabilistic analysis methods. A typical correctness requirement for the

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

pacemaker is that it maintains the rate of 60-100 beats per minute (BPM), which is checked against the signals generated from the realistic heart model. We target two main types of properties: (i) whether the pacemaker corrects faulty heart beats, and (ii) that the pacemaker does not induce erroneous heart behaviours (that is, it does not overpace the heart unless necessary). We utilise the pacemaker model presented in [8], to demonstrate the working of the framework. The use of probabilities also allows us to carry out performance evaluation of the pacemaker in terms of energy consumption during its life time.

Contributions: In this paper, we develop a methodology for quantitative, automated verification of pacemaker software models composed with a physiologically-relevant model of the heart based on [4], enhancing the results of [8]. We apply discretisation techniques in order to obtain the composed heart and pacemaker behaviour. We formulate and implement algorithms for quantitative, probabilistic analysis, for properties such as whether the pacemaker corrects Bradycardia (slow heart beat) and does not induce Tachycardia (fast heart beat), and analyse the effect of undersensing (due to noise on sensor readings) and energy consumption. We implement the framework using the probabilistic model checker PRISM [9] and MATLAB. PRISM, which supports *probabilistic* TA, is used for the discretisation of the pacemaker model. The complex heart model of [4] is implemented instead using MATLAB. We develop functions to export PRISM models into MATLAB, and to combine the pacemaker specification with the heart model, and hence carry out the verification in MATLAB. We provide experimental results which are encouraging.

Related work: Jiang *et al* [8] formulate a model for a basic dual chamber pacemaker in terms of a network of TAs. The pacemaker is verified using UPPAAL against a simple random heart model. [6], [7] develop a real-time Virtual Heart Model (VHM) which can be used for simulation and testing, whereas [7] devise a framework for testing and validation of implantable cardiac devices. In all these works, probabilities are not considered. Tuan *et al* [11] propose a real-time formal model for a pacemaker and its environment. A number of time constraints are verified using the PAT model checker. However, even though they have modelled all the operating modes described in Boston Scientific, they do not take into account probabilities and complex behaviours of the pacemaker, e.g. mode switching. Macedo *et al* [12] develop a concurrent and distributed real-time model for cardiac pacemakers through a pragmatic incremental approach. The models are expressed in VDM (Vienna Development Method) and checked using a scenario-based approach. Gomes *et al* [5] present a formal specification of the pacemaker using the Z notation. Theorem proving approaches are employed to validate some informal requirements defined by Boston Scientific. However, no validation experiments regarding safety conditions were performed. Mert *et al* [14] formally design all the operational modes of a single electrode pacemaker system using Event-B and prove them. They use the ProB tool to validate their models in different situations, such as the absence of input pulses.

Organization: This paper is organized as follows. Sect. II presents the necessary background on human heart beats, their model and a pacemaker model. Sect. III shows how we discretise and compose the (continuous) heart and pacemaker models. Sect. IV demonstrates the verification process and the experimental results. The paper is concluded with Sect. V, where possible future work is also discussed.

II. Preliminaries

In this section, inspired by [6], we first briefly describe the electrical system of the heart. The human heart maintains blood circulation of the body by contraction of the atria and ventricles. A special tissue in the *Sinus node* (see Fig. 1)

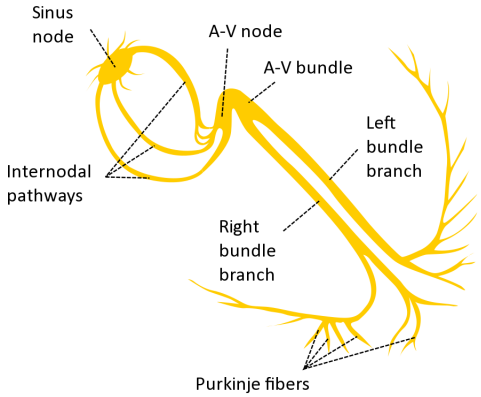


Figure 1. Electrical conduction system of the heart.

generates the electrical signal which is the primary pacemaker of the heart. The Sinus node spontaneously produces an electrical signal, which is conducted through *Internodal pathways* into the atrium causing its contraction. The signal then passes through the slow conducting *atrioventricular (A-V) node*, allowing the blood to empty out the atria and fill the ventricles. The fast conducting *His-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. This electrical system is the *natural pacemaker* of the heart. Abnormalities in the electrical signal generation and 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. The heart normally has around 60-100 *beats per minute* (BPM) and a heart beat is essentially a contraction of the ventricle.

The electrical signal that passes through the heart is known as *action potential*. This is the signal that an implantable pacemaker will receive or generate. A typical ventricular action potential is shown in Fig. 2. The action

potential is triggered by a voltage spike from the action potential of its neighbouring tissue or from an artificial pacing signal (pacemaker signal). The upstroke indicates the depolarization of the cell and the time when the muscle contracts. This is followed by the plateau, which allows the muscle to hold its contraction and fully eject blood. The downstroke is repolarisation, when the muscle relaxes and refills. A more gradual upstroke of depolarization will slow the conduction velocity of the tissue, as it will take more time to reach the voltage threshold necessary to trigger a neighbour tissues to depolarise. Thus, the shape and timing of the action potential determines the conduction velocity and refractoriness of the heart.

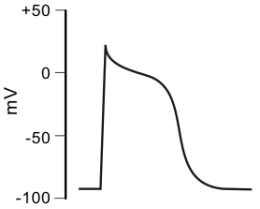


Figure 2. Action potential.

A. Basic models

We introduce two basic models that will be used later in the paper.

Definition 1 [Labelled Transition System] A *labelled transition system* (LTS) $\mathcal{S}$ is a tuple $\mathcal{S} = (S, s_0, A, L, Act, \rightsquigarrow)$ where: S is a finite set of states, $s_0 \in S$ is the initial state, A is the set of atomic propositions, $L : S \to 2^A$ is the labelling function which assigns to each state $s \in S$ a set of atomic propositions, *Act* is a set of actions and $\rightsquigarrow$: $S \times Act \times S$ is the transition relation.

A (discrete) path σ of an LTS is defined as a finite sequence $\sigma = s_0, a_0, s_1, a_1, \ldots, a_{n-1}, s_n$ of states and actions such that $(s_i, a_i, s_{i+1}) \in \rightsquigarrow$ for any $0 \leq i < n$. We define $\sigma[i]$ to be the i-th state in σ, $|\sigma|$ to be the length of the path, i.e., the number of states $s_i \in \sigma$, and $\sigma[i \ldots j]$ for any $i \leq j < |\sigma|$ to be the sub-path of σ starting in $\sigma[i]$ and ending in $\sigma[j]$.

Definition 2 [Markov Chains] A (time-inhomogeneous) *Markov* chain $\mathcal{D}$ is a tuple $\mathcal{D} = (S, \alpha, A, L, \mathbf{P})$ where: S, A and $L : S \to 2^A$ are defined as in Def. 1, $\alpha : S \to [0, 1]$ is the initial distribution, and $\mathbf{P} : S \times S \times \mathbb{N} \to [0, 1]$ is the step-dependent *transition probability* matrix.

The definition of paths carries over from LTSs to Markov chains. We also define the probabilistic distribution over the set of paths in a standard way (by considering the topology generated from the cylinder sets, cf., e.g., [3]). Given a finite path σ, its probability is given as $\Pr(\sigma) = \alpha(\sigma[0]) \prod_{i=0}^{|\sigma|-1} \mathbf{P}(\sigma[i], \sigma[i+1], i)$.

B. The heart model

We present the artificial vector model for generating ECG rhythms developed by Clifford *et al* [4]. An ECG is a signal recorded from the surface of the human chest, which describes the activity of the heart. An example ECG is given in Fig. 3. Typically, an ECG signal describes a cardiac cycle, which is of three main waves, P, QRS and T. The P wave denotes the *atrial depolarization*. The QRS wave reflects the rapid *depolarization of the right and left ventricles*. The T wave denotes the *repolarisation of the ventricle*. [4] presents a mathematical model given as a system of nonlinear ODEs. The model is the following:

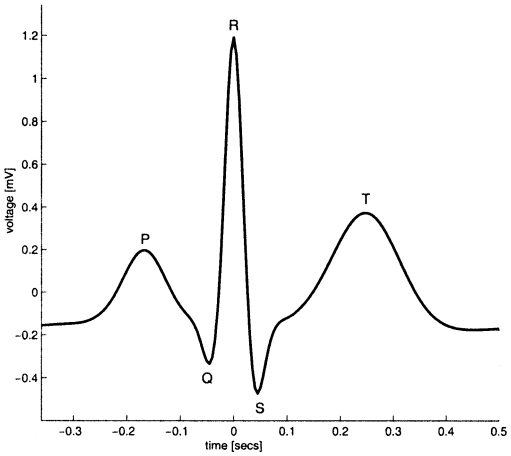


Figure 3. Example electrocardiogram [13].

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

where $\theta \in [-\pi, \pi]$ is the *cardiac phase*, $\Delta\theta_i^x = (\theta - \theta_i^x) \bmod 2\pi$, $\omega = \frac{2\pi h}{60\sqrt{h_{av}}}$, 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 $n = 6$) normalized by 60 BPM. The model generates an ECG signal for each cartesian coordinate $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$. In this paper we only need to consider the $\mathbf{x}$ coordinate, since the signal $\mathbf{x}$ is sufficient to generate the action potential.

To use Eq. (1) one has to define the *instantaneous (beat-to-beat) heart rate* function $h(t)$ ($t \in \mathbb{R}_{\geq 0}$), which specifies the distance between two consecutive R-events (highest peak in Fig. 3). Technically, it is equivalent to the so called *RR-series*, which can be generated by first constructing the power spectrum $S(f)$ as a sum of two Gaussian distributions $S(f) = \frac{\sigma_1^2}{\sqrt{2\pi c_1^2}} e^{\left(\frac{(f-f_1)^2}{2c_1^2}\right)} + \frac{\sigma_2^2}{\sqrt{2\pi c_2^2}} e^{\left(\frac{(f-f_2)^2}{2c_2^2}\right)}$, which have means f_1, f_2 and standard deviation c_1, c_2. By multiplying this time series by an appropriate scaling constant and adding an offset value, the resulting time series can give any required mean and standard deviation. More details on the construction of the RR-series are given in [13].

The system of ODEs from Eq. (1) models a single mode of heart behaviour, such as Bradycardia or Tachycardia, but not both. In order to model the case when the heart changes its behaviour, [4] introduce a probabilistic framework in

terms of Markov chains. The state space of the Markov chain denotes different *modes* of the functioning of the heart, and the transition from one state to another is given by the transition probability matrix $\mathbf{P}$. The state switch is made at the point when the cardiac phase θ jumps from π to $-\pi$. The probabilistic nature of the model hinges on our probabilistic analysis, provided in Sect. IV-C1. We refer the reader to [4] for more details.

C. The pacemaker model

The pacemaker is implanted under the chest skin and it 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 [8] develop a pacemaker model based on TAs [2]. For completeness, below we present this model and its variants.

Let $\mathcal{X} = \{x_1, \ldots, x_n\}$ be a set of *nonnegative* real-valued variables, called *clocks*. An $\mathcal{X}$-valuation is a function $\eta : \mathcal{X} \to \mathbb{R}_{\geqslant 0}$ assigning to each variable x a nonnegative real value $\eta(x)$. Let $\mathcal{V}(\mathcal{X})$ denote the set of all valuations over $\mathcal{X}$. A *clock constraint* on $\mathcal{X}$, denoted by g, is a conjunction of expressions of the form $x \bowtie c$ for clock $x \in \mathcal{X}$, comparison operator $\bowtie \in \{<, \leq, >, \geq\}$ and $c \in \mathbb{N}$. Let $\mathcal{B}(\mathcal{X})$ denote the set of clock constraints over $\mathcal{X}$. An $\mathcal{X}$-valuation η *satisfies* a constraint $x \bowtie c$, denoted $\eta \models x \bowtie c$, if and only if $\eta(x) \bowtie c$; it satisfies a conjunction of such expressions if and only if η satisfies all of them. Let $\mathbf{0}$ denote the valuation that assigns 0 to all clocks. For a subset $X \subseteq \mathcal{X}$, the reset of X, denoted $\eta[X := 0]$, is the valuation η' such that $\forall x \in X.\ \eta'(x) := 0$ and $\forall x \notin X.\ \eta'(x) := \eta(x)$. For $\delta \in \mathbb{R}_{\geq 0}$ 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 clocks proceed at the same speed.

Definition 3 [Timed I/O Automata] A *timed I/O automaton* (TA) is a tuple $\mathcal{A} = (\Sigma, \mathcal{X}, Q, I, q_0, \rightarrow)$, where:

- $\Sigma = \Sigma^{\text{in}} \cup \Sigma^{\text{out}} \cup \{\tau\}$ is the union of the set of *input* actions Σ^{in}, the set of *output* actions Σ^{out} and the *internal* action $\{\tau\}$;
- $\mathcal{X}$ is a finite set of clocks;
- Q is a nonempty, finite set of locations;
- $I : Q \to \mathcal{B}(X)$ assigns invariants to locations;
- $q_0 \in Q$ is the initial location;
- $\rightarrow \subseteq Q \times \Sigma \times \mathcal{B}(\mathcal{X}) \times 2^{\mathcal{X}} \times Q$ is the edge relation.

Semantics: Given a TA $\mathcal{A} = (\Sigma, \mathcal{X}, Q, I, q_0, \rightarrow)$, a location $q \in Q$ and a valuation η, the semantics of TA is given as a transition system where states are pairs $\langle q, \eta \rangle$ and transitions are defined by the rules:

- $\langle q, \eta \rangle \xrightarrow{d} \langle q, \eta + d \rangle$ if $\eta \in I(q)$ and $(\eta + d) \in I(q)$ for a non-negative real d;
- $\langle q, \eta \rangle \xrightarrow{a} \langle q', \eta' \rangle$ if $q \xrightarrow{g,a,X} q',\ \eta \in g,\ \eta' = \eta[X := 0]$ and $\eta' \in I(q')$.

We use a *network of timed (I/O) automata* for the composition of more than one TA. The semantics of a network of TAs is given as for a single TA in terms of transition systems. A state of the network is a pair $\langle \mathbf{q}, \eta \rangle$, where $\mathbf{q}$ denotes a vector of current locations of the network, one for each TA, and η is as usual a clock assignment storing the current values of the clocks in the system. A network may perform two types of transitions, delay transitions and discrete transitions. The rule for delay transitions is similar to the case of a single TA, where the invariant of a location vector is the conjunction of the location invariants of the TAs. There are two rules for discrete transitions, defining local actions where one of the TAs makes a move on its own, and synchronising actions where two processes synchronise on a channel and move simultaneously. Let q_i stand for the i'th element of a location vector $\mathbf{q}$ and $\mathbf{q}[q_i'/q_i]$ for the vector $\mathbf{q}$ with q_i being substituted by q_i'. The transition rules are as follows:

- $\langle \mathbf{q}, \eta \rangle \xrightarrow{d} \langle \mathbf{q}, \eta + d \rangle$ if $\eta \in I(\mathbf{q})$ and $(\eta + d) \in I(\mathbf{q})$, where $I(\mathbf{q}) = \bigwedge I(q_i)$;
- $\langle \mathbf{q}, \eta \rangle \xrightarrow{\tau} \langle \mathbf{q}[q_i'/q_i], \eta' \rangle$ if $q_i \xrightarrow{g,\tau,X} q_i',\ \eta \in g, \eta' = \eta[X := 0],\ \eta' = I(\mathbf{q}[q_i/q_i'])$;
- $\langle \mathbf{q}, \eta \rangle \xrightarrow{\tau} \langle \mathbf{q}[q_i'/q_i][q_j/q_j'], \eta' \rangle$ if $q_i \xrightarrow{g_i,a,X_i} q_i'$, $q_j \xrightarrow{g_j,b,X_j} q_j',\ a \in \Sigma^{out} \wedge b \in \Sigma^{in},\ i \neq j,\ \eta \in g_i \wedge g_j, \eta' = \eta[X_i \cup X_j := 0]$ and $\eta' = I(\mathbf{q}[q_i'/q_i][q_j/q_j'])$.

The pacemaker model in [8] consists of five TA components: the lower rate interval (LRI) component, the atrioventricular 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. 4)

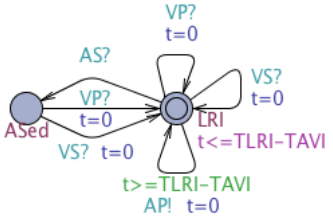


Figure 4. LRI component [8].

has the function to keep the heart rate above a given minimum value. The AVI component (see Fig. 5) 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. 6) notifies all other components that an atrial event has occurred. Notice that there is no AR signal as we are not using the advanced algorithms given in [8].

The URI component (see Fig. 7(a)) sets a lower bound on the times between consecutive ventricular events. The

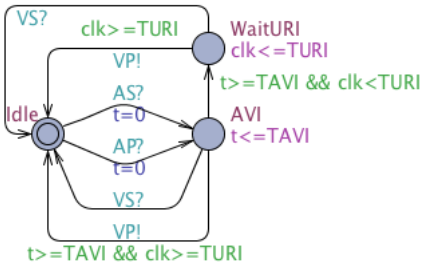


Figure 5. AVI component [8].

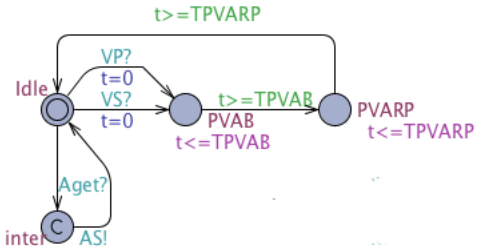
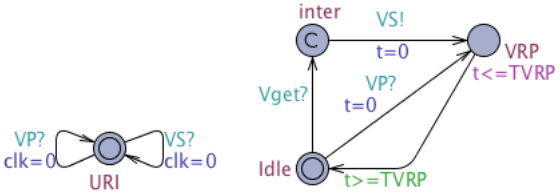


Figure 6. PVARP component [8].

VRP component (see Fig. 7(b)) filters noise and early events that may cause undesired behaviour. There are four actions in the pacemaker model that will be considered in the rest of the paper. The input actions Aget and Vget will notify the pacemaker when there is an action potential from the atrium or from the ventricle, respectively. The output actions AP and VP are responsible for pacing the atrium and the ventricle, respectively. Notice that in a real pacemaker device the input will be a signal. The pacemaker will have a voltage threshold that will be used to decide whether the signal yields an Aget or a Vget action. It is important to remark that all transitions from the pacemaker model that are not labelled with an input or output action are assumed to be labelled with the internal action τ. The locations that have transitions labelled with τ as well as the locations labelled with **C** do not allow the time to elapse.

III. Discretisation

As can be seen from Sect. II-C, to verify the pacemaker model we need to provide a sequence of actions consisting of Aget and Vget. Moreover, the pacemaker should be able to output actions AP and VP when needed. It is crucial to know *when* these actions happen, i.e., *timed* sequences are required. As stated earlier, the heart model consists of a system of nonlinear ODEs. To extract the (timed) sequence of Aget and Vget actions the system of ODEs in Eq. (1) needs to be solved, for which we use discretisation, resulting in an LTS. Accordingly, we also discretise the pacemaker model in order to obtain another LTS such that they can be composed and verified together. We now present two discretisation algorithms for the heart model and the pacemaker model, respectively. For the rest of the section,



(a) URI component [8] (b) VRP component [8]

Figure 7.

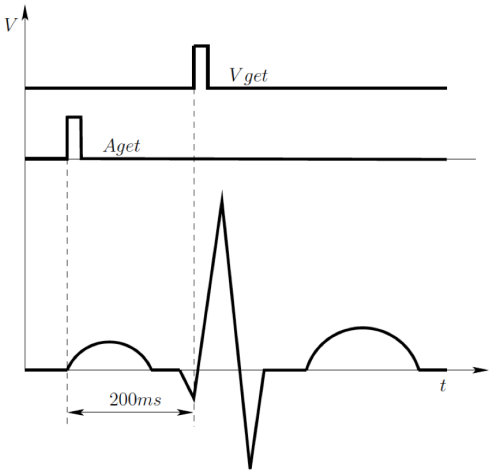


Figure 8. Mapping from ECG to action potential.

let $T \in \mathbb{N}$ be a time-bound, e.g. the life time of the battery of the pacemaker, let $\delta \in \mathbb{N}$ be a discretisation step (usually $\delta = 2$ms in our examples) and let $N = \lfloor \frac{T}{\delta} \rfloor$, the integral part of $\frac{T}{\delta} \in \mathbb{R}_{\geq 0}$, be the number of discretisation steps.

A. Discretisation of the heart model

The first step of the discretisation algorithm for the heart model is to generate the instantaneous (beat-to-beat) heart rate $h(t)$, $t \in \mathbb{R}_{>0}$, described in Sect. II-B. Then we use numerical techniques such as Runge-Kutta [15] to obtain the set of voltages $x(t_i)$ at times $t_i = i\delta$ for $i \leq N$. The values $x(t_i)$ will be used to generate the sequence of actions Aget and Vget, which are given as inputs to the pacemaker. In order to obtain the Aget or Vget action we use the mapping from Fig. 8. The Aget action is generated at the start of the P-event, and the Vget action is generated at the start of the Q-event. In a normal heart, there is usually a difference of 200 ms between consecutive Aget and Vget actions.

The mapping from Fig. 8 is used to generate the LTS from Fig. 9. The important characteristic of the LTS is that every transition corresponds to δ ms. Therefore, in order

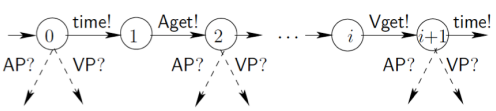


Figure 9. LTS modelling the action potentials of the heart.

to estimate the time difference between an Aget and a Vget event, one can count the number of states between these two events and then multiply it by δ. (For instance, to generate 5 min of heart behaviour using the discretisation step of 2 ms one would require to generate an LTS with $150,000$ states.) The last state from the generated LTS (not shown in Fig. 9) has a transition to the initial state labelled with 0. Note that, for the sake of simplicity, the LTS in Fig. 9 contains only one Aget and one Vget event, which corresponds to a single ECG cycle. A 5 min heart behaviour contains several hundreds of ECG cycles. Between the Aget and Vget actions there are time actions, which denote the passage of time. Fig. 9 also shows two input actions, AP and VP, enabled in all states except for the ones that have the output actions Aget and Vget. The purpose of these actions is to allow the pacemaker to pace the heart. A more detailed explanation of these actions will be given in Sect. III-C.

B. Discretisation of the pacemaker model

In this section, we provide a discretisation of the pacemaker model given as a network of TAs. The discretisation is based on the well-known region techniques for TA which can be regarded as the discretisation with step size 1. Here, to comply with the discretisation of the heart model, we will adapt to step size $\delta = 2$ ms.

A region [2] is an equivalence class under $\cong$, an equivalence relation on clock valuations, which can be characterised by a specific form of a clock constraint. Let c_{x_i} be the largest constant with which $x_i \in \mathcal{X}$ is compared in some guard in the TA. Clock valuations $\eta, \eta' \in \mathcal{V}(\mathcal{X})$ are *clock-equivalent*, denoted $\eta \cong \eta'$, if and only if either

1) for any $x \in \mathcal{X}$ it holds: $\eta(x) > c_x$ and $\eta'(x) > c_x$, or
2) for any $x_i, x_j \in \mathcal{X}$ with $\eta(x_i), \eta'(x_i) \leq c_{x_i}$ and $\eta(x_j)$, $\eta'(x_j) \leq c_{x_j}$ it holds: $\eta(x_j) \leq \eta'(x_j)$ iff $\lfloor \eta(x_i) \rfloor = \lfloor \eta'(x_i) \rfloor$ and $\{\eta(x_i)\} \leq \{\eta'(x_i)\}$.

Let $\mathcal{R}e(\mathcal{X})$ be the set of regions over the set $\mathcal{X}$ of clocks. For $\Theta, \Theta' \in \mathcal{R}e(\mathcal{X})$, Θ' is the *successor region* of Θ if for all $\eta \models \Theta$ there exists $\delta \in \mathbb{R}_{>0}$ such that $\eta + \delta \models \Theta'$ and $\forall \delta' < \delta.\ \eta + \delta' \models \Theta \vee \Theta'$. The region Θ *satisfies* the guard g, denoted $\Theta \models g$, iff $\forall \eta \models \Theta.\ \eta \models g$. The *reset operation* on region Θ is defined as $\Theta[X := 0] := \{\eta[X := 0] \mid \eta \models \Theta\}$. Let $\overline{\Theta}$ be the closure of Θ, $\mathring{\Theta}$ the interior of Θ, and $\partial\Theta = \overline{\Theta} \setminus \mathring{\Theta}$ the boundary of Θ.

Given a tuple $v = (v_1, \cdots, v_n)$ with n components, we denote by $v\lfloor_k$ the k-th component of v.

Definition 4 [Region graph of TAs] The *region graph* of a TA $\mathcal{A} = (\Sigma, \mathcal{X}, Q, I, q_0, \rightarrow)$ is $\mathcal{G}(\mathcal{A}) = (\Sigma \cup \{\mathsf{time}\}, V, v_0, \hookrightarrow)$, where

- $V = Q \times \mathcal{R}e(\mathcal{X})$ is a finite set of *vertices* with *initial vertex* $v_0 = (q_0, \Theta_0)$, where Θ_0 is the initial region such that $\mathbf{0} \in \Theta_0$;
- $\hookrightarrow \subseteq V \times ((\Sigma \times 2^{\mathcal{X}}) \cup \{\mathsf{time}\}) \times V$ is the transition relation defined as follows:
  - $v \overset{\mathsf{time}}{\hookrightarrow} v'$ if $v\lfloor_1 = v'\lfloor_1$, and $v'\lfloor_2$ is the successor region of $v\lfloor_2$;
  - $v \overset{a,X}{\hookrightarrow} v'$ if $v\lfloor_1 \xrightarrow{g,a,X} v\lfloor_1{}'$ with $v\lfloor_2 \models g$, and $v\lfloor_2[X := 0] = v'\lfloor_2$, $a \in \Sigma$.

Any vertex in the region graph is a pair consisting of a location and a region. For a vertex $v \in V$ and clock valuation $\eta \in \mathcal{V}(\mathcal{X})$ we define the *boundary function* $\flat(v, \eta) = \inf\{t \mid \eta + t \in \partial v\lfloor_2\}$, which is the minimum time (if it exists) to "hit" the boundary of the region corresponding to vertex v starting from a clock valuation η. Given a time bound T, we usually introduce a "global" clock y which is never reset, and thus also define $\flat(v, \eta, y)$ to be $\min\{\flat(v, \eta), T - y\}$, i.e., the minimum time to hit the boundary $\partial v\lfloor_2$ *at time* $y \leq T$.

We proceed to define a *discretised region graph*, given the discretisation step δ and a time bound T, as follows.

Definition 5 Given a region graph $\mathcal{G}(\mathcal{A}) = (\Sigma \cup \{\mathsf{time}\}, V, v_0, \hookrightarrow)$, a discretisation step δ and a time-bound T, the discretised region graph $\mathcal{G}_\delta(\mathcal{A}) = (\Sigma \cup \{\mathsf{time}\}, S, s_0, \hookrightarrow_\delta)$ is defined as follows:

- $S = \{(v, \eta, y) \mid v \in V \wedge \eta \in \partial v\lfloor_2 \wedge y \leq T\}$ is the set of states with initial state $s_0 = (v_0, \mathbf{0}, 0)$;
- $\hookrightarrow_\delta \subseteq S \times (\Sigma \cup \{\mathsf{time}\}) \times S$ is the transition relation defined for every $(v, \eta, y) \in S$ as follows:
  - if $\delta < \flat(v, \eta, y)$ and $v \overset{a,X}{\hookrightarrow} v'$ then
    $$(v, \eta, y) \overset{a}{\hookrightarrow}_\delta (v', (\eta + \delta)[X := 0], y + \delta), \text{ and}$$
    $$(v, \eta, y) \overset{\mathsf{time}}{\hookrightarrow}_\delta (v, \eta + \delta, y + \delta)\ .$$
  - if $\delta \geq \flat(v, \eta, y)$ and $v \overset{\mathsf{time}}{\hookrightarrow} v'$ then
    $$(v, \eta, y) \overset{\mathsf{time}}{\hookrightarrow}_\delta (v', \eta + \delta, y + \delta)\ .$$

Remark 1 *In Def. 5, the reachable part of $\mathcal{G}_\delta(\mathcal{A})$ is an LTS and it is finite if one sets $\delta = \lfloor \frac{T}{M} \rfloor$ for some natural number M. In the verification, we always conform to this convention.*

C. Composition between the heart and the pacemaker

Alg. 1 describes the composition between the LTS $\mathcal{H}$ of the heart and the LTS $\mathcal{P}$ of the pacemaker. This results in the composed LTS $\mathcal{C}$. The number of states of the composed LTS is N. As input the algorithm takes the initial state of the heart $\mathcal{I}_H$ and the initial state of the pacemaker $\mathcal{I}_P$. $\mathcal{C}$ has the same structure as $\mathcal{H}$, i.e., there is a transition from the last state to the initial state, and also has the same number of transitions. Notice that the composed LTS has two additional actions AP and VP, which correspond to the case when the pacemaker is pacing the heart. The algorithm starts from the initial states $\mathcal{I}_H$ and $\mathcal{I}_P$. As the heart has a single transition from the initial state in line 3 the algorithm retrieves the action act to the next state. This action is used as a synchronisation action. If the action is Aget or Vget the

Algorithm 1 Composition algorithm

Require: LTS of the heart $\mathcal{H}$, LTS of the pacemaker $\mathcal{P}$, initial state of the heart $\mathcal{I}_H$, initial state of the pacemaker $\mathcal{I}_P$ and number of states N of the composition LTS

Ensure: Return the composition LTS $\mathcal{C}$

1: $n_C := 0$; $next_P := \mathcal{I}_P$;
2: **while** $n_C < N$ **do**
3: $act := getact(\mathcal{H}, \mathcal{I}_H)$;
4: **if** $act ==$ Aget **or** $act ==$ Vget **then**
5: $next_P := getnextstate(\mathcal{P}, \mathcal{I}_P, act)$;
6: **if** $next_P == \varnothing$ **then** $deadlock()$;
7: $\mathcal{C} := clts(\mathcal{C}, n_C, act)$;
8: $n_C := n_C + 1$; $\mathcal{I}_P := next_P$;
9: $\mathcal{I}_H := getnextstate(\mathcal{H}, \mathcal{I}_H, act)$;
10: **continue**;
11: **endif**
12: $next_P := getnextstate(\mathcal{P}, \mathcal{I}_P, \mathsf{AP})$;
13: **if** $next_P != \varnothing$ **then**
14: $\mathcal{C} := clts(\mathcal{C}, n_C, \mathsf{AP})$;
15: $n_C := n_C + 1$; $\mathcal{I}_P := next_P$;
16: $\mathcal{I}_H := getnextstate(\mathcal{H}, \mathcal{I}_H, \mathsf{AP})$;
17: **continue**;
18: **endif**
19: $next_P := getnextstate(\mathcal{P}, \mathcal{I}_P, \mathsf{VP})$;
20: **if** $next_P != \varnothing$ **then**
21: $\mathcal{C} := clts(\mathcal{C}, n_C, \mathsf{VP})$;
22: $n_C := n_C + 1$; $\mathcal{I}_P := next_P$;
23: $\mathcal{I}_H := getnextstate(\mathcal{H}, \mathcal{I}_H, \mathsf{VP})$;
24: **continue**;
25: **endif**
26: **if** $act ==$ time **then**
27: $next_P := getnextstate(\mathcal{P}, \mathcal{I}_P, act)$;
28: **if** $next_P == \varnothing$ **then** $deadlock()$;
29: $\mathcal{C} := clts(\mathcal{C}, n_C, act)$;
30: $n_C := n_C + 1$;
31: $\mathcal{I}_H := getnextstate(\mathcal{H}, \mathcal{I}_H, act)$;
32: **endif**
33: $\mathcal{I}_P := next_P$;
34: **endwhile**
35: **return** $\mathcal{C}$

pacemaker tries to synchronise on it by calling the function $getnextstate(\mathcal{P}, \mathcal{I}_P, act)$ (line 5). If the pacemaker cannot synchronise, i.e., there is no transition $\mathcal{I}_P \overset{act}{\rightsquigarrow} next_P$, then there is a deadlock and the whole process stops (line 6). If the pacemaker can synchronise with the heart then the function $clts$ creates a transition in the composed LTS $\mathcal{C}$ labelled with act. The function $getnextstate(\mathcal{H}, \mathcal{I}_H, act)$ (line 9) assigns to the current state of the heart the next state and the loop (line 2) starts again. If the pacemaker cannot synchronise on Aget or Vget, then the algorithm checks whether the pacemaker wants to pace the heart by synchronising on the actions AP or VP. If the pacemaker can pace the heart (lines $12 - 25$) then the function $clts$ creates a transition in the composed LTS $\mathcal{C}$ labelled with AP or VP. The next state of the heart will become the state after the first Aget or the first Vget if the AP or VP action is triggered, respectively. When there is no pacing required the heart and the pacemaker will try to synchronise on the time action (line 26). If no synchronisation is possible there is a deadlock and the whole loop stops. The loop will continue until there is a deadlock or the number of transitions in the composed LTS $\mathcal{C}$ is N.

IV. Verification and Experimental Results

In this section, we demonstrate how the verification is performed and show the experimental results. The implementation is available at http://www.veriware.org/.

A. Model construction

Typically the verification takes the following steps. First, the pacemaker model described in Sect. II is encoded into PRISM, i.e., each input-output TA in Fig. 4-7 is translated into a PRISM "timed automaton module" and synchronisation actions are used to simulate the inputs and outputs of the original pacemaker.

Second, we build the discretisation of the heart model and the pacemaker model (as a network of TAs), using the approaches described in Sect. III-A and Sect. III-B respectively. For the network of TAs, we use the PRISM digital clock engine. The discretisation step is chosen as $\delta = 2$ ms. Note that the theory behind the digital clock engine is given in Def. 5. As result, a MATLAB script containing all the necessary information is generated, which is later used for the composition of the heart and the discretised pacemaker model.

B. Property specification

Given a sequence of action potentials we need to specify a property that checks whether the sequence corresponds to a normal heart behaviour. Intuitively, this means that "*there are between 60 and 100 heart beats (ventricular events) in any interval of window time*". For this purpose, we introduce a *monitor* in MATLAB to check that the constraint is satisfied.

Definition 6 [Duration of a path] Given a discrete path σ and a discretisation step δ, we define the duration of σ in milliseconds as: $\texttt{Dur}(\sigma) = \delta \cdot |\sigma|$. The definition extends naturally to any sub-path of σ.

Definition 7 [Heart beats] Given a discrete path σ we define the number of heart beats of σ as: $\texttt{Heart\_beats}(\sigma) = \sum_{i=0}^{|\sigma|-1} \mathbb{1}(\sigma, i)$, where $\mathbb{1}(\sigma, i)$ is the characteristic function of transitions $\sigma[i] \overset{a_i}{\rightsquigarrow} \sigma[i+1]$ with respect to a ventricular

event, which means that $\mathbb{1}(\sigma, i) = 1$ if $a_i \in \{\mathsf{Vget}, \mathsf{VP}\}$, and 0 otherwise. We also define in the same way the function $\texttt{Pacemaker\_beats}(\sigma)$ except that $a_i \in \{\mathsf{AP}, \mathsf{VP}\}$.

We are now ready to define "good paths", namely, paths corresponding to good heart behaviours.

Definition 8 [Good path] Given a discrete path σ we say that σ is *good*, if $\forall i, j.\ \ (\texttt{Dur}(\sigma[i..j]) = \texttt{1 minute}) \ \Rightarrow \ (\texttt{Heart\_beats}(\sigma[\texttt{i..j}]) \in [\texttt{60}, \texttt{100}])$.

C. Verification and results

We present experiments for the quantitative verification of the pacemaker. All of them are run in MATLAB on a 2.83GHz 4 Core(TM)2 Quad CPU with 3.7Gb of memory. In general, the purpose of the verification is twofold: (1) to check that the pacemaker corrects the slow beating of the heart; and (2) to check that the pacemaker does *not* induce bad behaviours of the heart.

For (1), Fig. 10 shows a graph of Bradycardia. The blue lines represent the behaviour of the faulty heart without introducing the pacemaker. The green lines instead represent the behaviour of the faulty heart when equipped with the pacemaker. The x-axis shows the time at which events (beats) occur, whereas the y-axis shows the beat type. There are four kinds of beat type: 1 which corresponds to Aget events, 2 which corresponds to Vget events, 3 for AP events and, finally, 4 for VP events (not shown in the graph). The normal beats of the heart (blue lines) are slow, approximately one every two seconds. However, once the pacemaker is introduced, it will induce an AP beat (atrium beat) after the first second of non-sensing heart beats. This will in turn produce a normal ventricular beat (Vget event) slightly after correcting the Bradycardia.

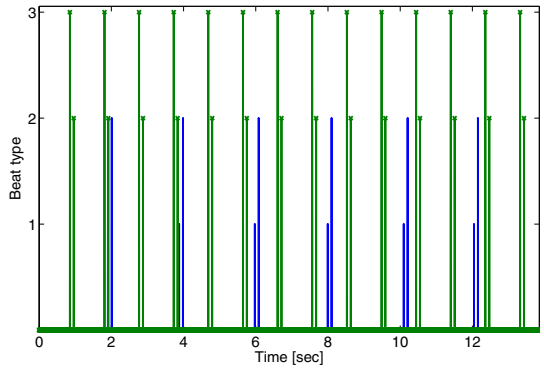


Figure 10. Correction of faulty heart with Bradycardia.

For (2), the idea is to give, as input, a perfect model of the heart to the pacemaker, i.e. a model that always produces between 60 and 100 BPM. We then check that, after the composition, the resulting behaviour is still normal. In Fig. 11 it is shown how the corrected heart behaviour (blue lines) has been corrupted by a pacemaker (green lines) for which the waiting time before pacing is too low. In this scenario, the pacemaker is inducing Tachycardia, in a heart which is functioning perfectly.

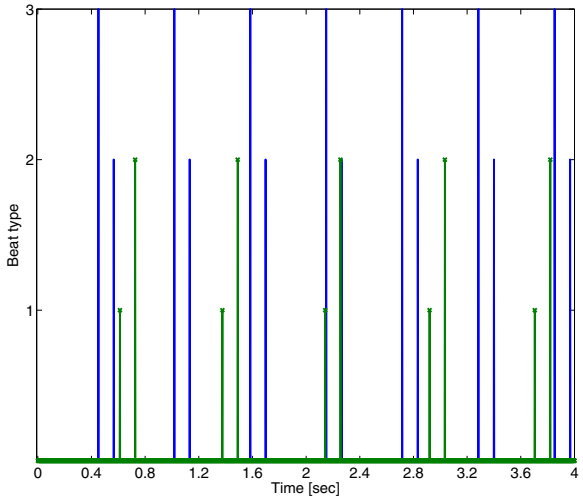


Figure 11. Faulty pacemaker inducing Tachycardia.

Based on this general picture, we summarise our results below. We separate the analysis into two parts: basic analysis and advanced analysis. The former concerns the basic function of the pacemaker (does it work properly?), while the latter concerns further aspects: here we consider the energy usage and undersensing.

1) Basic analysis:

Non-probabilistic analysis: The first set of experiments, presented in Tab. I, considers the heart model working in three different modes: Normal (**N**), Tachycardia (**T**) and Bradycardia (**B**). Here we do not consider the probabilistic transitions between different modes. We run four different experiments for each mode, the first one simulating one minute of heart beats, the second with two minutes of heart beats, the third with four minutes, and the fourth with eight minutes. For each experiment, we compute the state space of the model (#**N**), the verification time (**V_T**), the number of heart beats without the pacemaker (#**H_B**) and the number of beats induced by the pacemaker (#**P_B**).

In the case of normal heart behaviour the pacemaker produces 0 beats. The same happens during Tachycardia since the pacemaker cannot correct it. A more interesting case is when the heart is operating in Bradycardia mode. In this case the pacemaker will intervene, pacing the heart at normal rate. This is highlighted in Tab. I (fourth column) where the pacemaker forces the heart to beat at 69 BPM instead of 43 which was the original number of BPM of the slow heart without the pacemaker.

Probabilistic analysis: One advantage of the heart model in [4] is that it introduces probabilistic transitions between different "beat types" (in terms of modes) which can mimic the real heart by learning from the patient data.

	Minutes	#N	V_T	#H_B	#P_B
N:	1	33017	2.063	86	0
	2	65644	4.078	171	0
	4	131289	8.136	342	0
	8	262187	16.271	683	0
T:	1	32938	2.512	193	0
	2	65692	4.937	385	0
	4	131216	9.879	769	0
	8	262260	19.711	1537	0
B:	1	33023	1.688	43	69
	2	65997	3.330	86	137
	4	131258	6.603	171	274
	8	262504	13.196	342	548

Table I
EXPERIMENTAL RESULTS IN THE NON-PROBABILISTIC CASE.

This characteristic yields a much more precise heart model which might be *disease-dependent*, or even *patient-specific*, which in turn has the potential to improve the analysis and validation of the pacemaker. Below we demonstrate how to perform the verification using probabilistic model checking techniques. We construct a time-inhomogeneous Markov chain with three states $S = \{s_0, s_1, s_2\}$. The states are labelled as $L(s_0) = \mathbf{N}$, $L(s_1) = \mathbf{T}$ and $L(s_2) = \mathbf{B}$. Every fixed $\Delta = 30$ sec, probabilistically, the heart changes its state. The probability to switch between states can, in principle, be learned from the patient data. We use the following transition probability matrix $\mathbf{P}(s, s_0, i) = 0.9$ and $\mathbf{P}(s, S\backslash\{s_0\}, i) = 0.05$, $\forall s \in S$ and $i = 1, \ldots, \lfloor \frac{T}{\Delta} \rfloor$. The initial distribution is $\alpha(s_0) = 0.4$, $\alpha(s_1) = \alpha(s_2) = 0.3$. For the probabilistic analysis, we enumerate all paths and form the set $Paths$, where $|Paths| = |S|^{\lfloor \frac{T}{\Delta} \rfloor}$. Considering the high complexity of the heart model given by *nonlinear* ODEs and *time-inhomogeneous* Markov chains, path exploration offers a feasible solution. Finally, for each path $\sigma \in Paths$ we compute the probability $\Pr(\sigma)$.

We run seven experiments by considering different values for $\lfloor \frac{T}{\Delta} \rfloor$. Tab. II shows the probability of good paths, i.e. the paths returned by the monitor of good paths. The first probability value (**Pr_before**) refers to the heart model without the pacemaker, whereas the second (**Pr_after**) refers to the heart equipped with the pacemaker. Tab. II matches our intuitions. First, the probability of good paths increases when the heart and the pacemaker work together. This is due to the fact that the pacemaker can correct some Bradycardia behaviours. However, the heart is not perfect, which means that there are still "bad paths" even with the pacemaker. In fact, the pacemaker cannot correct Tachycardia. Thus, some paths will still be affected by such erroneous behaviour. Second, the probability of good paths decreases when increasing the length of the heart beats considered. This is because the monitor considers as "bad paths" any two consecutive cycles of time spent in Bradycardia or Tachycardia mode. Thus, increasing the length of the heart beats is equivalent to an increase in the number of switches between modes and the chance of getting two consecutive "faulty" (Bradycardia or Tachycardia) modes one after the other. Notice that for $\lfloor \frac{T}{\Delta} \rfloor = 1$ the initial energy consumption is approximately 20 whereas for $\lfloor \frac{T}{\Delta} \rfloor = 2, 3, \ldots$ the energy increases linearly with a factor of 2.4. This is due to the fact that in the first minute the heart induces Bradycardia with probability 0.3, i.e., $\alpha(s_2) = 0.3$, and the pacemaker beats around 69 times. For longer time periods the probability that the heart induces Bradycardia is smaller (0.05).

$\lfloor \frac{T}{\Delta} \rfloor$	#N	V_T [s]	Pr_before	Pr_after	Energy
3	99030	27.283	0.324	0.632	25.637
4	132040	109.246	0.292	0.600	28.037
5	165050	409.065	0.262	0.570	30.426
6	198060	1161.823	0.2361	0.5416	32.804
7	231070	5208.8	0.213	0.514	35.173
8	264080	14280.94	0.1913	0.4888	37.531
9	297090	59935	0.172	0.464	39.886

Table II
EXPERIMENTAL RESULTS IN THE PROBABILISTIC CASE.

2) Advanced analysis:

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 analysis is indispensable. In the probabilistic experiments presented in Tab. II we also calculate the expected energy consumption of the pacemaker expressed as units of energy per experiment. Formally, we define the expected energy as $\mathcal{E} = \sum_{\sigma \in Paths} \Pr(\sigma) \times \texttt{Pacemaker\_beats}(\sigma)$. The consumption grows approximately linearly with time. This matches our intuition, as during all the experiments we did not change the switching probabilities between different modes. Changing those probabilities yields different results. For example, increasing the probability of switching to Bradycardia mode would increase the expected energy consumption since the pacemaker would then be expected to pace more often.

The reader should be aware that, in all the experiments, for simplicity (and also because of the lack of real data) we assume that the pacemaker consumes one unit of energy every time it paces the atrium or the ventricle. This is by no means a genuine shortcoming of our analysis, as real data can be fed to our framework once the users have collected them from the pacemaker manufacturer. The analysis remains the same.

Undersensing: In previous analysis, we assume that the pacemaker can sense the action-potential perfectly. This can simplify the modelling considerably, but is not a realistic assumption. Indeed, sensing is usually subject to certain noise, which means that the obtained action-potential of the pacemaker deviates from its real value. This might generate undersensing. It means that, when the heart generates an Aget or a Vget, the pacemaker may not sense it. In fact,

when there is noise present, the pacemaker might have problems in detecting these events. In order to make the verification of pacemakers more realistic we enhance the model of the pacemaker with undersensing. The undersensing results in the real action potential signal having an additive noise. The additive noise follows a Gaussian distribution $\mathcal{N}(0, \sigma^2)$, where 0 is the mean and σ^2 is the variance. The pacemaker has a threshold value U to decide when the action potential signal results in an Aget or a Vget. The real signal sensed by the pacemaker can be expressed as $s(t) + w(t)$, where $s(t)$ is the signal without noise and $w(t)$ is the noise. We are interested in computing $p = \text{Prob}(s(t) + w(t) \geq U)$, which is the probability of the pacemaker sensing an Aget or a Vget generated by the heart. Using standard probability theory we obtain that $p = 1 - \frac{1}{\sqrt{2\pi\sigma^2}} \int_{-\infty}^{U} e^{-\frac{\tau^2}{2\sigma^2}} d\tau$. In Fig. 12, we depict the probability of not inducing Tachycardia (i.e., $1 - p$) in a normal heart by varying the value $1 - p$ in the pacemaker from 0 to 0.5. As the intuition suggests, the probability of inducing Tachycardia is high when the pacemaker fails to sense many events, i.e., $1 - p = 0.5$.

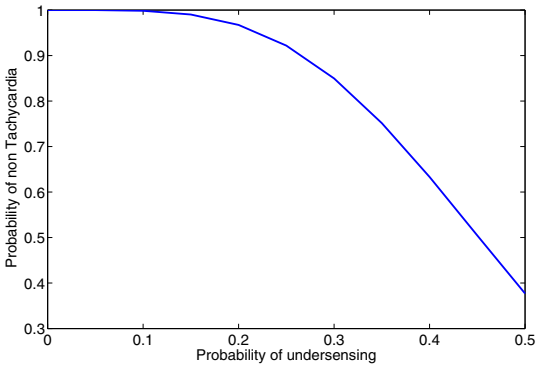


Figure 12. Probability of not inducing Tachycardia by undersensing.

V. Conclusion

In this paper we formulated quantitative verification algorithms for implantable pacemakers. We adapted the realistic heart model of Clifford *et al* [4] and utilised the network of TAs model of the pacemaker by Jiang *et al* [8]. A methodology for deriving the composition of the heart and the pacemaker, based on discretisation, was developed. We considered the main correctness properties, including that the pacemaker corrects Bradycardia and does not induce Tachycardia, for a range of realistic heart behaviours. We also analysed undersensing and energy usage. We implemented the framework using the probabilistic model checker PRISM and MATLAB, and demonstrated encouraging experimental results.

There are several interesting directions for future work. For instance, we plan to explore the parameter synthesis problem of the pacemaker. We also plan to perform more advanced energy analysis. Moreover, considering a failure model for the pacemaker seems to be a promising direction.

Acknowledgment

We would like to thank Gari Clifford and Julian Oster for sharing their insights on the heart model. Also, we are grateful to Rajeev Alur, Zhihao Jiang, and Miroslav Pajic for discussions of the pacemaker model.

References

[1] List of Device Recalls, U.S. Food and Drug Admin.

[2] R. Alur and D. L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126(2):183–235, 1994.

[3] C. Baier, J. Katoen. Principles of model checking. The MIT Press, 2008.

[4] G. Clifford, S. Nemati, and R. Sameni. An Artificial Vector Model for Generating Abnormal Electrocardiographic Rhythms. *Physiological Measurements*, 31(5):595–609, May 2010.

[5] A. O. Gomes and M. V. Oliveira. Formal specification of a cardiac pacing system. In Proc. *FM'09*, pages 692–707. Springer, 2009.

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

[7] Z. Jiang, M. Pajic, and R. Mangharam. Cyber-physical modeling of implantable cardiac medical devices. *Proceedings of the IEEE*, 100(1):122–137, 2012.

[8] Z. Jiang, M. Pajic, S. Moarref, R. Alur, and R. Mangharam. Modeling and verification of a dual chamber implantable pacemaker. In Proc. of *TACAS*'12, LNCS 7214, pages 188–203. Springer, 2012.

[9] M. Z. Kwiatkowska, G. Norman, and D. Parker. Prism 4.0: Verification of probabilistic real-time systems. In Proc. of *CAV*'11, LNCS 6806, pages 585–591. Springer, 2011.

[10] K. G. Larsen, P. Pettersson, and W. Yi. UPPAAL in a Nutshell. *International Journal on Software Tools for Technology Transfer*, 1(1-2): 134–152. Springer, 1997.

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

[12] H. Macedo, P. Larsen, and J. Fitzgerald. Incremental development of a distributed real-time model of a cardiac pacing system using VDM. In Proc. of *FM*'08, LNCS 5014, pages 181–197. Springe, 2008.

[13] P. McSharry, G. Clifford, L. Tarassenko, and L. Smith. A dynamical model for generating synthetic electrocardiogram signals. *IEEE Transactions on Biomedical Engineering*, 50(3):289–294, march 2003.

[14] D. Méry and N. K. Singh. Pacemaker's Functional Behaviors in Event-B. Rapport de recherche, 2009.

[15] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical Recipes in C*. Cambridge University Press, Cambridge, UK, 2nd edition, 1992.