Abstraction Trees for Closed-Loop Model Checking of Medical Devices*

Zhihao Jiang[1], Houssam Abbas[1], Pieter J. Mosterman[2,3], and Rahul Mangharam[1]

[1] Department of Electrical and Systems Engineering, University of Pennsylvania
[2] School of Computer Science, McGill University, Canada
[3] MathWorks, USA

Abstract. This paper proposes a methodology for closed-loop model checking of medical devices, and illustrates it with a case study on implantable cardiac pacemakers. To evaluate the performance of a medical device on the human body, a model of the device's physiological environment must be developed, and the closed-loop consisting of device (e.g., pacemaker) and environment (e.g., the human heart) is model-checked. Formal modeling of the environment and its application in model checking pose several challenges that are addressed in this paper. Pacemakers should guarantee safe operations across large varieties of heart conditions, which are represented by an incomplete set of timed automata models. A set of domain-specific abstraction rules are developed that can over-approximate the timing behaviors of a heart model or a group of heart models, such that the new behaviors introduced by abstraction are mostly physiologically meaningful. The rules serve as a systematic method to cover heart conditions that may not be explicitly accounted for in the initial set of heart models. Closed-loop model checking is systematically performed using the heart models in the abstraction tree, to obtain the most concrete counter-example(s) that correspond to property violation. These counter-examples, along with their physiological context, are then presented to the physician to determine their physiological validity. The proposed methodology creates a separation between steps requiring physiological domain expertise (model creation and abstraction rules definition) and steps that can be automated (rule application, model checking, and abstraction refinement). While the methodology is illustrated for pacemaker verification, it is more broadly applicable to the verification of other medical devices.

1 Introduction

Implantable medical devices such as pacemakers are designed to improve physiological conditions with very little human intervention. Their ability to autonomously affect the physiological state of the patient makes the medical devices safety-critical, and sufficient evidence for their safety and efficacy should be provided before the devices can be implanted in the patients. Medical devices increasingly rely on software, and device function and their clinical performance can be affected by seemingly minor changes to software. [4]

* This work was supported by NSF CAREER 1253842, NSF MRI 1450342 and STARnet, a Semiconductor Research Corporation program, sponsored by MARCO and DARPA.

[4] In what follows, the word 'device' is used to refer to the software of the device.

Over the course of the past four decades, cardiac rhythm management devices such as pacemakers and implantable cardioverter defibrillators (ICD) have grown in complexity and now have more than 80,000 to 100,000 lines of software code [2]. In 1996, 10% of *all* medical device recalls were caused by software-related issues and this rose to 15% between 2003-2012 [20,18]. There is currently no standard for testing, validating, and verifying the software for implantable medical devices [3,4].

There are two categories of device bugs: 1) the device may fail to conform to its *specifications*, that is, the prescription of how it should react to certain inputs. 2) the device may fail to improve the conditions of the patient as promised, even if it conforms to its specifications. The desired physiological conditions that the closed-loop system should achieve are captured in the *physiological requirements*; for example, for a pacemaker, the heart rate should always be maintained above a certain threshold.

Bugs in the first category (non-conformance to specification) can be detected via systematic and extensive open-loop testing in which a set of input sequences is fed to the device, and its output is compared with the expected output. Bugs in the second category (violation of physiological requirements), on the other hand, require the availability of the *closed-loop system*, which consists of the device and its environment. For instance, the pacemaker and the heart as its environment. In the medical device industry, closed-loop verification of the physiological requirements is mostly performed in terms of clinical trials, in which the actual devices are implanted in human subjects over an extended duration. Unfortunately, because of the extremely high cost of clinical trials (several million dollars and spanning several years [21]), the amount and variety of human subjects during the clinical trials are limited, which reduces the opportunity to find bugs. Moreover, clinical trials are often conducted at the final design stage. Fixing bugs at this stage is very costly.

Closed-loop model checking enables closed-loop evaluation of the physiological requirements at an earlier design stage, which requires formal model(s) of the physiological environment. In closed-loop model checking, there is only one device model. However there can be a large number of environmental conditions which require different models to represent them. For instance, a heart with atrial flutter has an additional conduction pathway that is not present in a healthy heart, causing fast atrial rate. The timing and structural differences of different heart conditions should be distinguished in corresponding heart models. A set of initial models of the environment can be constructed but the set is inherently incomplete because of the large number of environment conditions and their combinations. As a result, performing model checking using every model in the set cannot ensure full coverage of the environmental conditions.

In this paper, domain-specific over-approximation rules are developed that produce abstract models that not only cover explicitly modeled environment conditions, but also cover timing behaviors and conditions not modeled in the set of initial models. The abstract models can be then used for closed-loop model checking of the device model. If the closed-loop system satisfies a requirement, the device under verification satisfies the requirement under environment conditions covered by the abstract models. However, if the requirement is not satisfied, the model checker returns a counter-example. In device modeling, the counter-example is considered *spurious* if it can not be produced by the device (as shown in (Fig. 1(a)). However

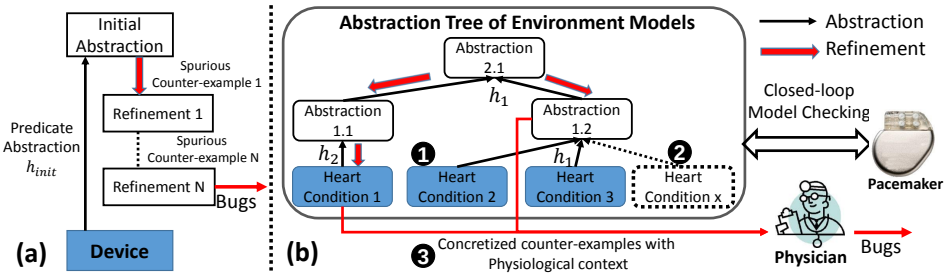

Fig. 1. (a) Device modeling with CEGAR framework (b) Closed-loop model checking with environment abstraction tree.

in environment modeling, even if the counter-example can not be produced by any of the initial environment models, it might still be a physiologically valid behavior. Thus the validity of a counter-example cannot be determined by refining the environment model, but can ultimately only be determined by domain experts.

Counter-examples returned from abstract models can be difficult to interpret by domain experts. One abstract counter-example could be produced by multiple physiologically valid conditions, which causes ambiguity. Thus, a rigorous framework is necessary to balance the need to cover a wide range of environmental conditions and the need to provide counter-examples to the physicians within their physiological context.

Another challenge for closed-loop model checking of medical devices is the amount of domain expertise needed during: 1) physiological modeling, 2) model abstraction and refinement, and 3) checking the validity of counter-examples. Thus the framework must also allow non-domain experts to perform verification (item 2 above), and establish 'hand-off' points where the results of verification can be handed back to the experts for interpretation.

1.1 Contributions

In this paper a framework is proposed for environment modeling in closed-loop model checking of medical device software. The cardiac pacemaker is used as an example of applying this framework. An expandable set of timed-automata heart models are first developed to represent different physiological conditions (Fig. 1 Marker 1). A set of domain-specific abstraction rules are then developed based on physiological knowledge, which help ensure the physiological relevance of the behaviors introduced into the abstract models (Fig. 1 Marker 2). Then the rules are applied to the initial set of physiological models to obtain an abstraction tree, which will be used for closed-loop model checking of the pacemaker. A straightforward search procedure is then used to conduct model checking using suitable heart models and return the most concrete and unambiguous counter-examples to the physicians for analysis (Fig. 1 Marker 3). In this framework, physiological knowledge is only needed when constructing the initial model set and when analyzing counter-examples. The application of the physiological abstraction rules and the verification procedure can be automated. The proposed method can potentially be generalized to other domains in which the device operates in a large variety of environmental conditions.

1.2 Related Work

Counter-Example Guided Abstraction Refinement (CEGAR) [8] has been proposed to over-approximate the behaviors of the device using predicate abstraction (Fig. 1.(a)). CEGAR works well during device modeling, however, it cannot be applied to environment modeling for two reasons: 1) predicate abstraction does not guarantee the validity of behaviors introduced into the model. In fact, for device modeling, all additional behaviors introduced into the abstract model are spurious. 2) the validity of a counter-example cannot be checked automatically as in device modeling.

Proof-based approaches have also been applied to verify abstractions and refinements of pacemaker specification using Event-B [9]. However, the authors did not take into account environment behaviors thus the framework cannot be used for verifying physiological requirements.

Physiological modeling of cardiac activities has been studied at various levels for different applications. In [19], the electrical activity of the heart is modeled with high spatial fidelity to study the mechanisms of cardiac arrhythmia. In [10], formal abstractions of cardiac tissue have been studied to reduce the complexity of the heart tissue model. However, these two models do not focus on the interaction with the pacemaker, therefore cannot be used for closed-loop model checking. In [7], hybrid automata models of the heart has been used to capture the complex beat-to-beat dynamics of the heart tissue. However the model cannot be used to cover behaviors across different heart conditions.

In previous work [12] a set of formal heart models covering various heart conditions at different abstraction levels was developed, and closed-loop model checking was performed on models of implantable pacemakers. However, the physiological knowledge required during each step of closed-loop model checking prevents the method to be practical.

2 Technical preliminaries

Timed automata [6] are an extension of finite automata with a finite set of real-valued clocks. A timed automaton $\mathbf{G}$ is a tuple $\langle S, S_0, \Sigma, X, inv, E\rangle$, where S is a finite set of locations. $S_0 \subset S$ is the set of initial locations. Σ is the set of events. X is the set of clocks. inv is the set of invariants for clock constraints at each location. E is the set of edges. Each edge is a tuple $\langle s, \sigma, \psi, \lambda, s'\rangle$ which consists of a source location s, an event $\sigma \in \Sigma$, clock constraints ψ, λ as a set of clocks to be reset and the target location s'. For the clock variables X, the clock constraints Ψ can be inductively defined by $\Psi := x \perp c \| \Psi_1 \wedge \Psi_2$, where $\perp \in \{\leq, =, \geq\}$, and $c \in \mathbb{N}$.

Semantics of Timed Automata A state of a timed automaton is a pair $\langle s, v\rangle$ which contains the location $s \in S$ and the valuation $v : X \to \mathbb{R}$ for all clocks. The set of all states is Ω. For all $\lambda \subseteq X$, $v[\lambda := 0]$ denotes the valuation which sets all clocks $x \in \lambda$ as zero and the rest of the clocks unchanged. For all $t \in \mathbf{R}$, $v + t$ denotes the valuation which increases all the clock value by t. There are two kinds of transitions between states. The *discrete transition* happens when the condition of an edge has been met. So we have:

$$\langle s, \sigma, \psi, \lambda, s'\rangle \in E, v \models \psi, v[\lambda := 0] \models inv(s') \Rightarrow (s, v) \xrightarrow{\sigma} (s', v[\lambda := 0])$$

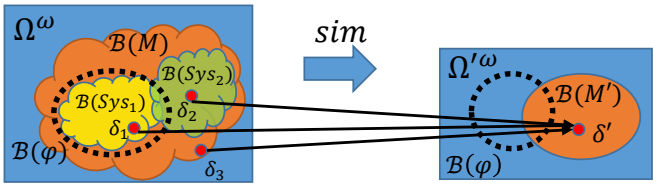

Fig. 2. Two models $Sys1, Sys2$ are over-approximated by model M such that $\{Sys1, Sys2\} \preceq_t M$. M' time-simulates M such that $M \preceq_t M'$. For a property φ if $M' \not\models \varphi$, δ' is returned as counter-example. However, δ' corresponds to 3 different behaviors in the original behavior space: δ_1 satisfies φ and is produced by $Sys1$, δ_2 falsifies φ and is produced by $Sys2$, and δ_3 falsifies φ and belongs to neither behavior space.

The *timed transition* happens when the timed automaton can stay in the same location for certain amount of time. We have:

$$\tau \in \mathbb{R}, \forall \tau' \leq \tau, v + \tau' \models inv(s) \Rightarrow (s, v) \xrightarrow{\tau} (s, v + \tau)$$

Observable Timed Simulation For two timed automata $T^1 = \langle S^1, S_0^1, \Sigma^1, X^1, inv^1, E^1 \rangle$ and $T^2 = \langle S^2, S_0^2, \Sigma^2, X^2, inv^2, E^2 \rangle$, $\Sigma_o \subseteq \Sigma^1 \cap \Sigma^2$ is a set of *observable events*, an observable timed simulation relation is a binary relation $sim_o \subseteq \Omega^1 \times \Omega^2$ where Ω^1 and Ω^2 are sets of states of T^1 and T^2. We say T^2 simulates T^1 ($T^1 \preceq_t T^2$) if the following conditions holds:

- Initial states correspondence: $(\langle s_0^1, \mathbf{0} \rangle, \langle s_0^2, \mathbf{0} \rangle) \in sim_o$
- Timed transition: For every $(\langle s_1, v_1 \rangle, \langle s_2, v_2 \rangle) \in sim_o$, if $\langle s_1, v_1 \rangle \xrightarrow{\tau} \langle s_1, v_1 + \tau \rangle$, there exists $\langle s_2, v_2 + \tau \rangle$ such that $\langle s_2, v_2 \rangle \xrightarrow{\tau} \langle s_2, v_2 + \tau \rangle$ and $(\langle s_1, v_1 + \tau \rangle, \langle s_2, v_2 + \tau \rangle) \in sim_o$.
- Observable discrete transition: For every $(\langle s_1, v_1 \rangle, \langle s_2, v_2 \rangle) \in sim_o$, if $\langle s_1, v_1 \rangle \xrightarrow{\sigma_o} \langle s_1', v_1' \rangle$, in which $\sigma_o \in \Sigma_o$, there exists $\langle s_2', v_2' \rangle$ such that $\langle s_2, v_2 \rangle \xrightarrow{\sigma_o} \langle s_2', v_2' \rangle$ and $(\langle s_1', v_1' \rangle, \langle s_2', v_2' \rangle) \in sim_o$.

Observable timed-simulation can also be used to cover the behaviors of multiple models. A model M'simulates models M_1 and M_2 if $M_1 \preceq_t M'$ and $M_2 \preceq_t M'$, which is denoted as $\{M_1, M_2\} \preceq_t M'$.

A *trace* of a timed automaton M is a sequence δ with $\langle s_1, v_1 \rangle \cdots \langle s_n, v_n \rangle$ such that $\forall i \in [1 \cdots n-1], \langle s_i, v_i \rangle \xrightarrow{\alpha} \langle s_{i+1}, v_{i+1} \rangle \in E$ for $\alpha \in \Sigma \cup \mathbb{R}$. The reachable behavior space of M is denoted as $\mathbb{B}(M) \subset \Omega^{\omega}$. For a timed-automata M' such that $M \preceq_t M'$, for every trace $\delta \in \mathbb{B}(M)$, there exists a trace $\delta' \in \mathbb{B}(M')$ with $\langle s_1', v_1' \rangle \cdots \langle s_n', v_n' \rangle$ such that $\forall i \in [1 \cdots n-1], \langle s_i', v_i' \rangle \rightarrow \langle s_{i+1}', v_{i+1}' \rangle \in E'$, and also $(\langle s_i, v_i \rangle, \langle s_i', v_i' \rangle) \in sim_o$. We abuse the notation of sim_o and denote $(\delta, \delta') \in sim_o$. M' has more behaviors than M, and we say that M' over-approximates M.

Validity of a counter-example: Certain properties are preserved in timed simulation relation. For $\varphi \in ATCTL^*$, if $M \preceq_t M'$, we have $M' \models \varphi \Rightarrow M \models \varphi$ [8]. However, in general $M' \not\models \varphi \Rightarrow M \not\models \varphi$ does not hold. Violations of $ATCTL$ yield *counter-examples* and the validity of which need to be checked. For two models such that $M \preceq_t M'$, $\delta' \in \mathbb{B}(M')$ is *spurious* if $\nexists \delta \in \mathbb{B}(M)$ s.t. $\langle \delta, \delta' \rangle \in sim_o$. However, for environment models such that $\{M_1, M_2, ...M_n\} \preceq_t M'$, and an execution $\delta' \in$

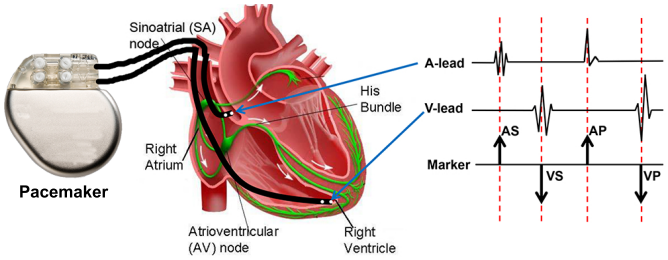

Fig. 3. (a) Lead placement for a dual chamber pacemaker (b) Electrogram (EGM) signals from pacemaker leads and corresponding internal event markers

$\mathbb{B}(M')$, the following condition is enough to prove δ' is spurious, but not necessarily invalid.

$$\forall i \ \nexists \delta \in \mathbb{B}(M_i) \text{ s.t. } \langle \delta, \delta' \rangle \in sim_o$$

Since there may be a valid environment model $M_c \notin \{M_1, M_2, ...M_n\}$ and $\mathbb{B}(M_c) \subset \mathbb{B}(M')$ and $\delta' \in \mathbb{B}(M_c)$. It is thus up to the domain experts to determine the validity of the counter-example.

3 Formal Models of the Environment

To perform closed-loop model checking of medical devices, formal models of their physiological environment are needed to represent different physiological conditions the devices may encounter. In this section, timed-automata [6] models of the human heart are developed as the environment model for implantable pacemaker [12,14]. Physiological requirements are formalized with monitors and $ATCTL^*$ formula [5]. Model checking can then be performed on the closed-loop system in model checker UPPAAL [17].

3.1 Electrophysiological Heart Modeling Basics

A healthy heart generates periodic electrical impulses to control heart rates according to physiological needs. These impulses propagate through the heart, triggering coordinated muscle contractions and pump blood to the rest of the body. The underlying pattern and timing of these impulses determine the heart's rhythm and are the key to proper heart functions. Derangements in this rhythm are referred to as *arrhythmia*, which impair the heart's ability to pump blood and compromise the patients' health. Arrhythmias are categorized into so-called Tachycardia and Bradycardia. Tachycardia features undesirable fast heart rate which results in inefficient blood pumping. Bradycardia features slow heart rate which results in insufficient blood supply. Different heart conditions can be distinguished by the *timing* of the electrical conduction, and the *topology* of the electrical conduction system of the heart, which are researched in clinical setting referred to as *Electrophysiology (EP)*[16].

Implantable cardiac pacemakers are rhythm management devices designed to treat bradycardia. A typical dual chamber pacemaker has two leads inserted into the heart through the veins which can measure the local electrical activities of the right atrium and right ventricle, respectively. According to the timing between

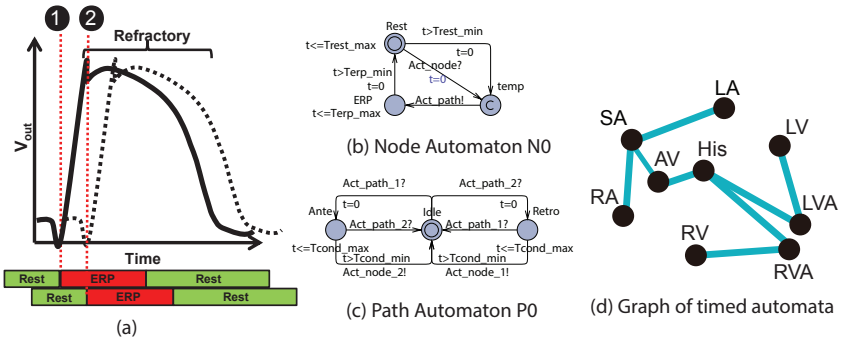

Fig. 4. (a) Action potential for a heart tissue and its tissue nearby (dashed). (b) Node automaton. (c) Path automaton. (d) An example model of the heart consist of a network of node and path automata. The nodes are labeled with the name of the corresponding physiological structures.

sensed impulses the pacemaker can deliver electrical pacing to the corresponding chamber to maintain proper heart rhythm (Fig. 3).

3.2 Timed Automata Models of the Heart

At cellular level, a heart tissue can be activated by an external voltage. Certain tissues also have the capability to self-activate, which contributes to natural heart beats. Once activated (Marker 1 in Fig. 4), the voltage outside the tissue changes over time, which is referred to as *Action Potential* (Fig. 4.(a)). The action potential can be divided into two functional timing periods: The *Effective Refractory Period (ERP)*, during which the tissue cannot be triggered by another activation; and the *Rest* period, during which the tissue can be activated and at the end of which the tissue will self-activate. The timing behaviors of the action potential are modeled as **node automaton** A_v (Fig. 4.(b)).

A node automaton initializes with the Rest state. From the Rest state, the node can either self-activate or be activated by external activations (indicated by Act_node). Upon activation the node transition to the ERP state and activate all the paths connecting to the node (indicated by Act_path). In the ERP state the node does not respond to external activations. At the end of ERP state the node transition to the Rest state. The duration a node automaton can stay in the Rest is in the range $[Trest\_min, Trest\_max]$, and the duration it can stay in ERP is in the range $[Terp\_min, Terp\_max]$. For heart tissue without the capability to self-activate, the parameters $Trest\_min$ and $Trest\_max$ are set to ∞. $Trest$ and $Terp$ are referred to as *parameters* of the automaton A_v.

The voltage change of the heart tissue will activate the tissue nearby with certain delay (Marker 2 in Fig. 4). This timing delay between heart tissue is modeled using **path automaton** A_e (Fig. 4.(c)). The initial state of a path automaton is Idle, which corresponds to no conduction. A path has two conduction directions, forward and backward. These are represented by the states Ante and Retro, named after their standard physiological terms Antegrade and Retrograde. If Act_path event is received from one of the nodes (1 or 2) connected to the path, the transition to Ante or Retro state will occur in the path automaton. At the end of Ante and Retro state the path will transition to Idle state and send Act_node signal to

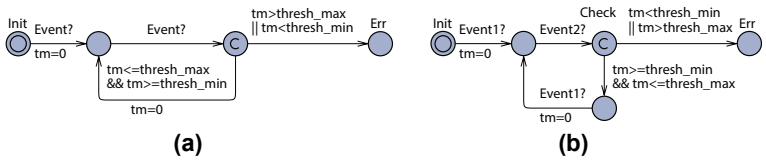

Fig. 5. (a) M_{sing} for single event; (b) M_{doub} for two events

the node automaton connected to the other end of the path (2 or 1). The spatial and temporal properties of a given human heart condition can be modeled by a network of node and path automata with different parameters (i.e., Fig. 4.(d)). Physiological structures of the heart are represented as node automata and the path automata specify the connectivities of the nodes and the conduction delays among them. The network can be viewed as a labeled directed graph:

Definition 31 *[Labeled graph] A **labeled graph** is a directed graph $G = (V, E, A)$ where V is a finite set of vertices, $E \subset V \times V$ is a finite set of directed edges, and A is a total labeling function $A : V \cup E \to TA$ where TA is the set of timed automata. The function A labels each vertex with a* node automaton*, and each edge with an* path automaton*. For a graph G, we write $\mathcal{E}(G) := V(G) \cup E(G)$.*

The heart model structure has been used to model various heart conditions and all of them have been validated by electrophysiologists [13].

3.3 Formalizing Physiological Requirements

Physiological requirements must be formalized for closed-loop model checking. In the case of medical devices, the devices are designed to *improve* certain physiological conditions. Software developers are particularly interested in the scenario in which a healthy open-loop physiological condition became an unhealthy closed-loop condition due to device intervention, which is a bug in the device.

In general, a closed-loop requirement φ is in the form of $\varphi_E \Rightarrow \varphi_C$, in which φ_E is the open-loop physiological condition that the device encounters, often in form of parameter ranges in the environment models, and φ_C is the closed-loop physiological condition that the device should achieve. Then we have:

$$M_E \models \varphi_E \wedge M_E || M_D \models \varphi_C \Rightarrow M_E || M_D \models \varphi \tag{1}$$

The substates for a heart model are clocks and locations for each node and path automata. In [15], physiological heart conditions are mapped to constraints on substate variables of the heart models, which can be written as atomic propositions. General monitors are also developed in Stateflow [1] for closed-loop testing of physiological requirements. The timed-automata version are shown in Fig. 5. The $M_{sing}(event, thresh\_min, thresh\_max)$ enforces the time interval between two $event$ signals within $[thresh\_min, thresh\_max]$.
$M_{doub}(event1, event2, thresh\_min, thresh\_max)$ enforces the time interval between $event1$ and $event2$ signals within $[thresh\_min, thresh\_max]$. Model checking is performed on the closed-loop system including the heart model M_H, the pacemaker model M_P, and the monitor M. The requirement φ_P can be then represented with TCTL formula: A[] (not M.Err)

4 Physiological Abstraction rules

In this section, domain-specific abstraction rules are developed that can introduce new behaviors to a given heart model or a set of models. These new behaviors are physiologically meaningful and might be manifested by a heart condition not explicitly modeled in the initial set of models. The physician (or domain expert) remains the ultimate arbiter of what is physiologically meaningful. This is a peculiarity of environment modeling, borne out of the fact that the initial set of models is necessarily incomplete, and does not represent all valid behaviors.

Recall that heart conditions are modeled as finite directed labeled graphs as introduced in Def. 31. Rules operate on a graph only if it has the appropriate structure for that rule, and if its parameters meet certain rule-specific conditions (if any). Due to space limits only one abstraction rule is discussed in detail. The full set of rules and the proofs that these rules indeed produce over-approximations of the behavior space are relegated to the technical report [11].

Rule R7: Replace Conduction With Self-activation We describe Rule R7 as it illustrates both effects of an abstraction rule: structure change and modifications to the automata. The effect of a conduction path is to conduct electrical activity from a node. Since the pacemaker cannot distinguish self-activation of the node and activation triggered by path conduction, we can use self-activation to replace path conduction. If all self-activation nodes are allowed at any time by setting their minimum Rest period to 0, all the conduction paths can be removed, while preserving the original behaviors (where the Rest period was constrained to a finite interval).

Applicability conditions. This rule can only be applied after Rule 5 and Rule 6 have been applied.

Output graph. All edges are deleted: $G' = (V(G), \emptyset)$. The node automata are replaced with the one shown in Fig. 6.d.

Effect on parameters For every node automaton N in G', $N.Trest_{min} = 0$.

Consider Fig. 6.(a) showing an application of R7, $H'''_{vt} = N^1_A P^1 N^1_V$ is abstracted to $H_{all} = N^2_A N^2_V$. Here we prove that $H'''_{vt} \preceq_t H_{all}$ with observable events $\Sigma_o = \{N_A.Act\_path, N_V.act\_path\}$. The state of H'''_{vt} is represented by $(N^1_A.loc, P^1_1.loc, N^1_V.loc, N^1_A.t, P^1_1.t, N^1_V.t)$ and the state of H_{all} is represented by $(N^2_A.loc, N^2_V.loc, N^2_A.t, N^2_V.t)$. Due to space limit, only one transition from each category is presented:

Initial state: First for the initial state we have:

$$\langle (Rest, Idle, Rest, 0, 0, 0), (Rest, Rest, 0, 0) \rangle \in sim_o$$

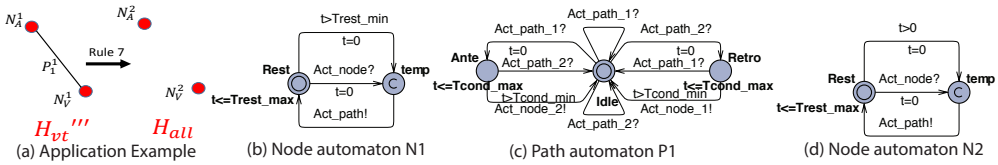

Fig. 6. (a) Rule 7 application example; (b)(c) Node and path automata used in H'''_{vt}; (d) Node automata used in H_{all}

Timed transitions: Consider a timed transition in H'''_{vt}

$$(Rest, Idle, Rest, t_1, t_2, t_3) \xrightarrow{\tau} (Rest, Idle, Rest, t_1 + \tau, t_2 + \tau, t_3 + \tau)$$

in which $(\tau \in \mathbb{R}) \wedge (t_1 + \tau \leq N_A^1.Trest\_max) \wedge (t_3 + \tau \leq N_V^1.Trest\_max)$. For a state in H_{all} such that $\langle (Rest, Idle, Rest, t_1, t_2, t_3), (Rest, Rest, t_1, t_3) \rangle \in sim_o$, there is a timed transition:

$$(Rest, Rest, t_1, t_3) \xrightarrow{\tau} (Rest, Rest, t_1 + \tau, t_3 + \tau)$$

and $\langle (Rest, Idle, Rest, t_1 + \tau, t_2 + \tau, t_3 + \tau), (Rest, Rest, t_1 + \tau, t_3 + \tau) \rangle \in sim_o$.
Discrete transitions: Consider a discrete transition in H'''_{vt}

$$(Rest, Ante, Rest, t_1, t_2, t_3) \xrightarrow[t_2 \in [P_1^1.Tcond\_min, P_1^1.Tcond\_max)]{N_V^1.Act\_path!} (Rest, Idle, Rest, t_1, t_2, 0)$$

in which $N_V^1.Act\_path! \in \Sigma_o$.
For a state in H_{all} such that $\langle (Rest, Idle, Rest, t_1, t_2, t_3), (Rest, Rest, t_1, t_3) \rangle \in sim_o$, there is a discrete transition:

$$(Rest, Rest, t_1, t_3) \xrightarrow[t_3 \in [0, N_V^2.Trest\_max)]{N_V^2.Act\_path!} (Rest, Rest, t_1, 0)$$

and $\langle ((Rest, Idle, Rest, t_1, t_2, 0)), (Rest, Rest, t_1, 0) \rangle \in sim_o$. Basically activation due to conduction is replaced by self-activation of the corresponding node automata.
Additional behaviors: The timed-simulation also allows additional behaviors into H_{all}. Consider a discrete transition in H_{all}

$$(Rest, Rest, t_1, t_3) \xrightarrow[t_3 \in [0, N_V^2.Trest\_min)]{N_V^2.Act\_path!} (Rest, Rest, t_1, 0)$$

However, for a state in H'''_{vt} such that $\langle (Rest, Idle, Rest, t_1, t_2, t_3), (Rest, Rest, t_1, t_3) \rangle \in sim_o$, when $t_3 \in [0, N_V^1.Trest\_min]$ there is no available discrete transitions. Physiologically, these implicitly included behaviors correspond to fast heart rate, premature heart events and even noise.

5 Closed-loop Model Checking With Abstraction Tree

A set of heart models corresponding to different heart conditions are first developed. The list can be expanded as new heart conditions are discovered. Because we start from a set of initial models, and each one may be abstracted using a number of abstraction rules, we have a choice of which rules to apply to which models, and the order in which to apply them. Depending on which rule is applied when, we end up with different abstract models. Thus an *abstraction tree* T_{HM} for the heart is created, as shown in Fig. 7. After the abstraction tree is built, it can be used for closed-loop model checking by non-domain experts. The question is which models to select from the abstraction tree to perform model checking, and provide the most concrete counter-examples. These counter-examples are provided to the

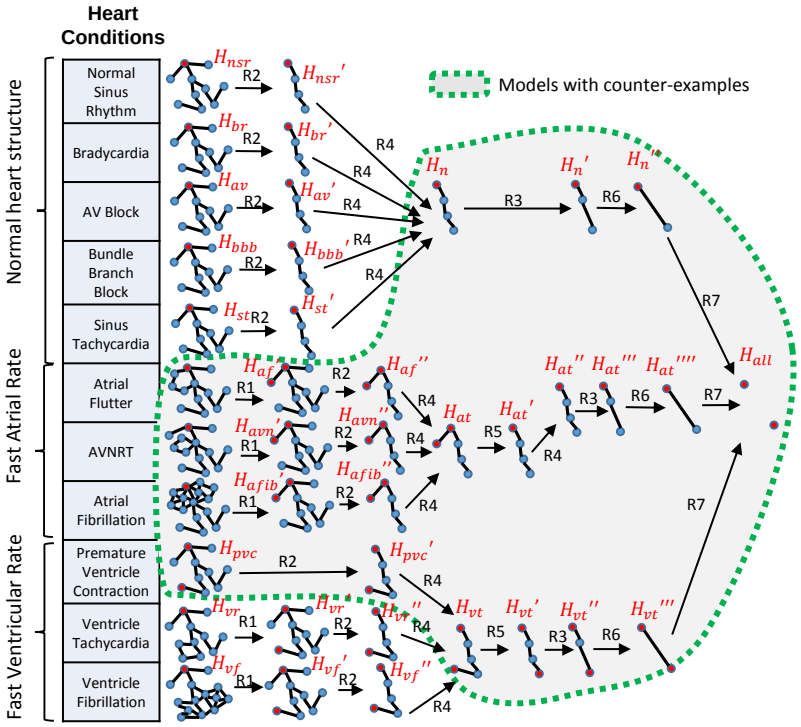

Fig. 7. Heart Model Abstraction Tree

physicians, along with the heart models that generated them, to determine their physiological validity.

We now summarize the model checking procedure, and give a detailed example in the next section. Recall the definition of appropriateness from Section 2. In the technical report [11], we show that a sufficient condition for a model M to be appropriate for a requirement φ with monitor Mon is $Var(\varphi) \subset Var(M) \cup Var(Mon)$.

A search algorithm is developed to select the most abstract models from the abstraction tree T_{HM} that are appropriate for a requirement Req. The detailed implementation of the algorithm can be found in [11]. Model checking is run on these models. When a run returns a violation, it returns an abstract counter-example δ_{cex}, along with the heart model M_{cex} that generated it. For each such violation, we search the tree for the most concrete descendant of M_{cex} that still displays a counter-example to Req. This is then provided to the physician along with the counter-example. The detailed implementation of the algorithm can be found in [11].

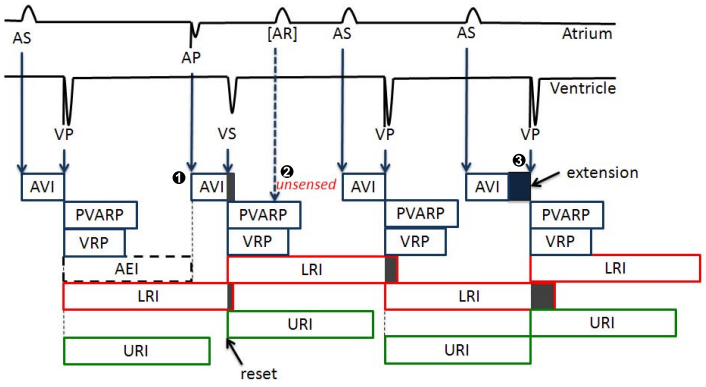

Fig. 8. Basic timers for a dual chamber pacemaker. AS: Atrial Sense, VS: Ventricular Sense, AP: Atrial Pacing, VP: Ventricular Pacing.

6 Case Study: Closed-loop Model Checking of a Dual Chamber Pacemaker

Step 1: The Pacemaker Model

A pacemaker diagnoses heart conditions and delivers electrical pacing to the heart according to the timing intervals between events from the heart and the pacemaker itself. In this section we use a simple dual chamber pacemaker as example for closed-loop model checking. The detailed UPPAAL timed automata implementation of the model can be found in [12]. A dual chamber pacemaker has several basic timers, which are shown in Fig. 8:

Atrial Escape Interval (AEI) defines the maximum interval between the last ventricular event (VS,VP) to an atrial event (AS,AP). If no AS happened before the AEI timer expires, atrial pacing (AP) is delivered to the heart (Marker 1 in Fig. 8).

Atrio-Ventricular Interval (AVI) defines the maximum interval between the most recent atrial event (AS,AP) to a ventricular event (VS,VP). If no VS happened before AVI timer expires, and the time since the most recent ventricular event (VS,VP) is no less than URI, ventricular pacing (VP) is delivered to the heart (Marker 3 in Fig. 8).

Post-Ventricular Atrial Refractory Period (PVARP) and Ventricular Refractory Period (VRP) define the minimum period that a AS or VS can happen since the most recent ventricular event (VS,VP).

Step 2: Requirement Encoding

The following requirement is designed to prevent the pacemaker from pacing too fast: "If the intervals between self-activations of the atria are between 300ms to 1000ms (60bpm-200bpm), the intervals between ventricular paces should be no shorter than 500ms." Self-activation of the atria can be expressed using the location and clock of of node automaton N_A. The requirement can be formalized using the monitor $M_{sing}(VP, 500, \infty)$:

$$Req1 : N_A.loc = Rest \wedge N_A.t \in [300, 1000] \Rightarrow \neg M_{sing}.loc == Err$$

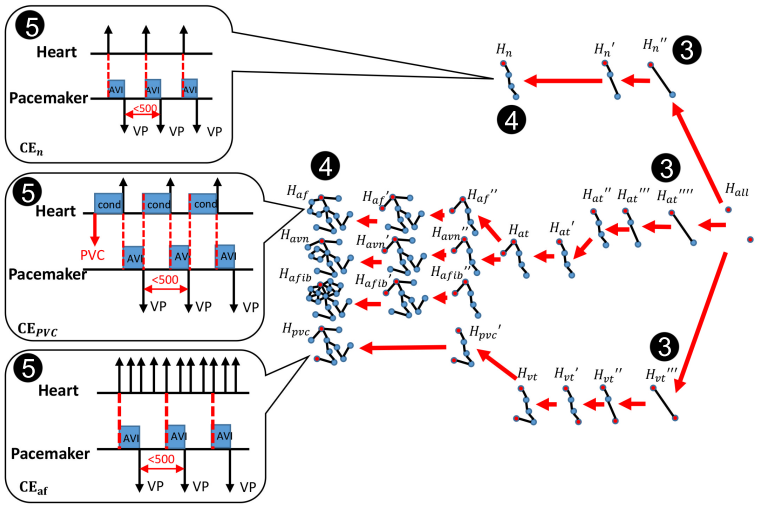

Fig. 9. Finding the most concrete counter-examples using the abstraction tree

Step 3: Choosing Appropriate Heart Models For the Requirement

To verify the closed-loop system with pacemaker model PM and abstraction tree T_{HM} (Fig. 7) against requirement $Req1$, the most abstract appropriate models are selected from the tree. The single event monitor M_{sing} from Fig. 5.a with variables $Var(M_{sing}) = \{M_{sing}.t, M_{sing}.loc\}$ is used for this requirement. The variables in the requirement are: $Var(Req1) = \{N_A.t, N_A.loc, M_{sing}.loc\}$.

At the root of the tree H_{all}, we have $\{N_A.t, N_A.loc\} \not\subset Var(H_{all}) \cup Var(M_{sing})$. So H_{all} is not appropriate for $Req1$. All the children of H_{all}: $H''_n, H''''_{at}, H'''_{vt}$ are appropriate for $Req1$, thus these 3 heart models are output as the most abstract models that are appropriate for $Req1$.

Step 4: Return the most Concrete Counter-Examples

After the appropriate models for $Req1$ are selected, we have the initial set $HM = \{H''_n, H''''_{at}, H'''_{vt}\}$. Then we run Algorithm 2. By model checking on all 3 initial models in UPPAAL we have:

$$H''_n||PM \not\models Req1;\ H''''_{at}||PM \not\models Req1;\ H'''_{vt}||PM \not\models Req1$$

The abstraction tree is then further explored. The heart models with counter-examples are illustrated in Fig. 7, and the most refined heart models with counter-examples are: $H_n; H_{pvc}; H_{af}; H_{avn}; H_{afib}$.

Step 5: Analysis of the Counter-examples

The counter-examples are then shared with physicians for analysis. In Fig. 9 we demonstrate 3 counter-examples. In the counter-examples, the first signal shows the intrinsic heart signals over time with up arrows as atrial activations and down arrows as ventricular activations. The second signal shows the pacemaker outputs with up arrows as atrial pacing and down arrows as ventricular pacing.

CE_n is returned by H_n and none of its children models violate the requirement. By careful analysis we found that CE_n features the combination of fast intrinsic atrial rate and prolonged A-V conduction delay, which is the combination of heart conditions H_{st} and H_{av}. This scenario shows that the abstraction rules can introduce physiological heart conditions that were not explicitly modeled in the initial model set. The pacemaker improved the open-loop heart condition by pacing the ventricles AVI after each atrial event, which is a correct operation of the pacemaker despite the requirement violation.

CE_{pvc} has a very similar execution to CE_n. However, the activations of the atrial node are triggered by retrograde conduction from ventricular paces (marker cond). The atrial activations trigger another ventricular pace after AVI, which will trigger another retrograde conduction. In this case the heart rate is inappropriately high, which corresponds to a dangerous closed-loop behavior referred to as *Endless Loop Tachycardia*.

In CE_{af} the atrial rate is very high, which is also a sub-optimal but not dangerous heart condition. However, the ventricular rate can stay normal due to the blocking property of the AV node. Despite the filters in the pacemaker, the pacemaker still paces the ventricle for every 3 atrial activations, which extends fast atrial rate to more dangerous fast ventricular rate. This scenario is referred to as Atrial Tachycardia Response of a pacemaker.

From the analysis, pacemaker operations in CE_{pvc} and CE_{af} must be revised. However, the revision should not affect the behavior in CE_n. This example demonstrates that counter-examples from refined models provide more physiological context of the requirement violations, and distinguish the physiological conditions that can trigger the violations. The information is helpful for debugging and improving the algorithm. The physicians can also improve the physiological requirement so that these heart conditions can be then considered case by case.

7 Conclusion

In this paper we addressed the challenges for environment modeling during closed-loop model checking of medical device software. Physiological abstraction rules can be defined to increase coverage of physiological conditions beyond explicitly modeled conditions. By using the abstraction tree constructed by applying the physiological abstraction rules, closed-loop model checking returns the most concrete counter-examples with physiological context to the physician when a physiological requirement is violated. In this paper we use implantable pacemaker as case study but the framework can extend to other domains.

In the future, application of the abstraction rules will be automated. The application of this framework can also be extended to other Cyber-Physical domains.

References

[1] Matlab R2013a Documentation → Stateflow. http://www.mathworks.com /help/toolbox/stateflow.

[2] Personal communication with Paul L. Jones, Senior Systems/Software Engineer, Office of Science and Engineering Laboratories, Center for Devices and Radiological Health, US FDA. August, 2010.

[3] US FDA, General Principles of Software Validation; Final Guidance for Industry and FDA Staff, 2002.

[4] US FDA, Guidance for the Content of Premarket Submissions for Software Contained in Medical Devices, 2005.

[5] R. Alur, C. Courcoubetis, and D. Dill. Model-checking for real-time systems. In *Logic in Computer Science, 1990. LICS '90, Proceedings., Fifth Annual IEEE Symposium on e*, pages 414–425, 1990.

[6] R. Alur and D. L. Dill. A Theory of Timed Automata. *Theoretical Computer Science*, 126:183–235, 1994.

[7] T. Chen, M. Diciolla, M. Kwiatkowska, and A. Mereacre. Quantitative verification of implantable cardiac pacemakers over hybrid heart models. *Information and Computation*, 236(0):87 – 101, 2014.

[8] E. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith. Counter Example-Guided Abstraction Refinement for Symbolic Model Checking. *J. ACM*, 50(5):752–794, 2003.

[9] Dominique Mery and Neeraj Kumar Singh. Pacemaker's Functional Behaviors in Event-B. *Research Report*, 2009.

[10] M. A. Islam, A. Murthy, A. Girard, S. A. Smolka, and R. Grosu. Compositionality results for cardiac cell dynamics. In *Proceedings of the 17th International Conference on Hybrid Systems*, HSCC '14, pages 243–252, 2014.

[11] Z. Jiang, H. Abbas, P. Mosterman, and R. Mangharam. Tech Report: Abstraction-Tree For Closed-loop Model Checking of Medical Devices. *http://repository.upenn.edu/mlab_papers/73*, 2015.

[12] Z. Jiang, M. Pajic, R. Alur, and R. Mangharam. Closed-loop verification of medical devices with model abstraction and refinement. *International Journal on Software Tools for Technology Transfer*, pages 1–23, 2013.

[13] Z. Jiang, M. Pajic, A. Connolly, S. Dixit, and R. Mangharam. Real-Time Heart Model for Implantable Cardiac Device Validation and Verification. In *22nd Euromicro Conference on Real-Time Systems (ECRTS)*, pages 239 –248, July 2010.

[14] Z. Jiang, M. Pajic, and R. Mangharam. Cyber-Physical Modeling of Implantable Cardiac Medical Devices. *Proceeding of IEEE Special Issue on Cyber-Physical Systems*, 2011.

[15] Z. Jiang, M. Pajic, and R. Mangharam. Model-based Closed-loop Testing of Implantable Pacemakers. In *ICCPS'11: ACM/IEEE 2nd Intl. Conf. on Cyber-Physical Systems*, 2011.

[16] M. Josephson. *Clinical Cardiac Electrophysiology.* Lippincot Williams and Wilkins, 2008.

[17] K. Larsen, P. Pettersson, and W. Yi. Uppaal in a Nutshell. *International Journal on Software Tools for Technology Transfer (STTT)*, 1997.

[18] K. Sandler, L. Ohrstrom, and R. McVay. Killed by Code: Software Transparency in Implantable Medical Devices. *Software Freedom Law Center*, 2010.

[19] N. A. Trayanova and P. M. Boyle. Advances in modeling ventricular arrhythmias: from mechanisms to the clinic. *Wiley Interdisciplinary Reviews: Systems Biology and Medicine*, 6(2):209–224, 2014.

[20] US FDA. Medical Device Recall Report FY2003 to FY2012. 2012.

[21] US FDA. Human Subject Protection; Acceptance of Data from Clinical Studies for Medical Devices; Proposed Rule. 2013.