Requirement-Guided Model Refinement

Zhihao Jiang

University of Pennsylvania, Philadelphia PA, USA**

1 Introduction

Medical device is a typical Cyber-Physical System and ensuring the safety and efficacy of the device requires closed-loop verification. Currently closed-loop verifications of medical devices are performed in the form of clinical trials in which the devices are tested on the patients. Using clinical trials as closed-loop verification has several problems:

- The cost of clinical trials are high
- Due to the cost, the scale of the clinical trials cannot be large, thus affecting patient generality and the effectiveness of the trials.
- It is the last step of the design process thus discovering a bug at this stage is very costly to fix.

In [1] we addressed the problems above by proposing a model-based design framework to enable closed-loop verification earlier in the design stage. The increased confidence in safety and efficacy of the device can potentially reduce the scale of the clinical trials, thus reduce cost.

The biggest challenge for closed-loop evaluation is to bridge the domain knowledge gap between the domain experts and software developers. These domain knowledge are two folds:

1. How the physical plant works?
2. Evaluation of the well-being of the patient (Performance)

From tool developer's perspective, both of the aspects need to be addressed.

1.1 How the physical plant works?

The first aspect can be addressed by developing appropriate model(s) of the physical plant. The word "appropriate" means two things:

- The model is validated, such that it captures the correct input-output of the system that it models.
- The model has the correct level of abstraction, such that it has to be complex enough to model the required details of the patients' condition, and at the same time avoid over-complexity.

** This document serves as the final report for Zhihao Jiang's internship at Mathworks during Summer 2014, under supervision of Pieter Mosterman from Mathworks and Prof. Rahul Mangharam from University of Pennsylvania.

In [1] we developed the Virtual Heart Model (VHM), which is an electrophysiological heart model structure that captures the timing behaviors of the electrical conduction system of the human heart. The model structure can be used to represent different patient conditions and those conditions are validated by the physicians.

In [2], we formally defined a series of heart models using the VHM structure. The heart models are at different abstraction levels and can be used for model checking of pacemaker software. We also adapted the Counter-Example Guided Abstraction and Refinement (CEGAR) framework [3] to select the most appropriate model during model checking.

1.2 Evaluation of the well-being of the patient (Performance)

Having the models for the physical plant is important, thus this alone is not enough to bridge the gap between domain expert and software developers. These additional domain knowledge lies mostly in:

- The requirements specified by the domain experts to evaluate the safety and efficacy of the device.
- Assumptions made during heart model abstractions

The connection between these two aspects and the heart models are not well formalized, which significantly limiting the ability of the software developers to use the heart models.

In this project, we propose a Matlab toolbox for EP heart modeling to address the problem above during pacemaker software design. We formalized the abstraction rules and their effects on the model behaviors. Each heart model documents all the rules that were applied and all observable behaviors. These information can be used to 1) validate the model and 2) evaluate whether the model has enough information to express the heart conditions in the requirement.

This report is arranged as follow. Section 2 introduces the observable behaviors of the heart. Section 3 formalizes the abstraction rules and their effects on the heart behaviors. Section 4 introduces the heart structure and validation. Section 5 introduces the requirements and how to use heart behaviors to detemine whether a heart model can be used to verify certain requirement.

2 Observable Behaviors of The Heart

The observable EP heart behaviors are local depolarizations that can be recorded by placing electrodes against the inner heart wall. With these recordings, the physicians developed their own perspective of describing heart behaviors in terms of generation and conduction of depolarizations throughout the heart. In terms of modeling, the behaviors are state transitions of the model.

The following are the most fundamental behaviors to describe the behavior of the heart.

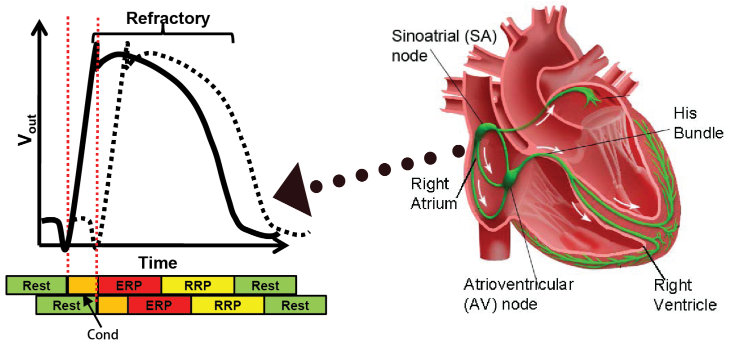

Fig. 1. Heart Model Abstractions

– Self-activation (self):increase the number of activations
– Conduction (cond): maintain the number of activations
– Refractory Blocking (block): reduce the number of activations
– External Pacing (pace): increase the number of activations

We name the behavior of depolarization of certain heart tissue *activation (act)*. It can be triggered by both *self-activation* and *conduction*. Ideally one activation generated either by self-activation or pacing will activate all the activatable tissue in the heart **once**. First there are identified structures within the electrical conduction system of the heart. Without forming a circle in the heart, the self-depolarization always happens in the *SA node (SA)*. After depolarizing the whole atria, the depolarization signal reaches the *AV node (AV)*. Then the signal travels through the *His Bundle (His)* and depolarize the whole ventricle, including the *Right Ventricular Apex (RVA)* through the *Pukinje Fibers (Puk)*.

Let S be the set of structures and B be the set of behaviors. We represent the behavior associated with a structure $s \in S$ using $s.b$.

So we have the following behaviors: SA.self, SA.block, SA_AV.cond, AV.block, His.cond, His.block, Puk.cond, Puk.block, RVA.block. Some tissue in the left atrium and the ventricles can self-depolarize prematurely. They are called *Premature Atrial Contraction (PAC)* and *Premature Ventricular Contraction (PVC)* respectively. So we have two more behavior: PAC.self and PVC.self. A typical heart cycle can be represented by a sequence of behaviors: SA.self$\rightarrow$SA_AV.cond $\rightarrow$ His.cond$\rightarrow$Puk.cond$\rightarrow$Vent.act

3 Abstraction Rules for the Heart

Abstraction rules are based on both domain knowledge and reasoning. The rules may have dependencies among each other, and for rules that are independent of each other, applying them in different order on a heart model will yield the same result.

In the next sub-section I will introduce the abstraction rules that we use for the toolbox and their brief justifications.

3.1 Abstraction Rules

The rules that we used to abstract the heart models are as follow:

1. Remove all the nodes which do not generate electrical pulses
2. Merge sources of electrical pulses to the nearest pace node
3. Replace the blocking property of ERP period with non-deterministic conduction
4. Replace activation from conduction with self-activation

3.2 Behavior Abstraction

During the abstractions the structure of the model changes, while at the same time the behaviors of the model changed as well. Although the total number of behaviors will not decrease during overapproximation abstractions, specific behaviors may merge with other behaviors thus will not be distinguishible anymore. These are the type of changes that apply to the behaviors during abstraction:

– Renaming: sometimes behaviors are assigned to another structure and their old names no longer make sense. Or we can say that Renaming is a special case for Inclusion
– Inclusion: can also be used for constructing higher-level behaviors. e.g. Atrial.self={SA.self,PAC.self}
– Remove: Behaviors like Block can be removed without **reducing** the observable behaviors

The result of the abstractions can be seen in Fig. 1.

Abstraction 1: HM1 SA.self={SA.self}
PAC.self={PAC.self}
AV.block={AV.block}
P2.cond={SA_AV.cond}
P3.cond={His.cond}
RVA.act={PVC.self,P3.cond}

Abstraction 2: HM2 N1.self={SA.self,PAC.self}
N2.block={AV.block}
P4.cond={P2.cond}
P5.cond={P3.cond}
N3.act={RVA.act}

Abstraction 3: HM3 N4.self={N1.self}
N5.self={N4.self}
P6.cond={P4.cond,P5.cond}
P6.block={N2.block}

Abstraction 4: HM4 N6.self={N3.self,P6.cond}
N7.self={N5.self,P6.cond}

3.3 High-level behaviors of the heart

We can use the behavior abstractions to construct high-level behaviors of the heart to represent different heart conditions.
Intrinsic atrial activations:
Atrial.self={SA.self,PAC.self}
Intrinsic Ventricular activations:
Vent.self={PVC.self}

3.4 Heart Model Validation

In [1] our heart models were validated by physicians. With the rule-based heart model abstraction, each abstract heart model can be validated using the following sequence:

1. Validate the initial heart model (structure)
2. Validate all the abstraction rules applied.

4 The Heart Modeling Tool Box

In the toolbox we have 4 data structures:

– Heart models
– Physiological behaviors of the heart
– Abstraction rules for the heart
– Physiological requirements

The heart model structure have following fields:

– Heart parameter sets
– Rules applied to the model
– Behavior set

Functions associated with the heart model:

– Display
– Heart Model validation

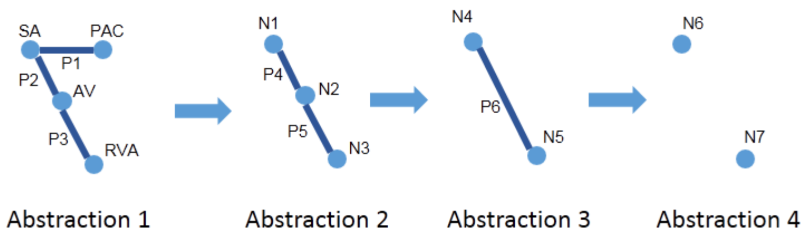

Fig. 2. Stroke Volume as a function of heart rate and AV interval

5 Requirement-Guided Heart Model Selection

Closed-loop requirements are mostly **conditional**, meaning that it should hold under certain environmental condition. For example, the following requirement:
R1: When the intrinsic activation in the atria and ventricles are both less than 100bpm, the ventricular rate is less than 100bpm
The requirement constrains on the Atrial.self and Vent.self, and reason about Vent.act

5.1 Rules for Heart Model Eligibility Check

– The behavior (or its renames) constrained in the condition of the requirement should not be abstracted with other behaviors in the heart model
–

```
function [status,message]=eligible(HM,Req)
        for all BehaveObj1 in Req.condition
            for all the BehaveObj2 in HM.behavior
                while (BehaveObj1 not in BehaveObj2)||(BehaveObj2 not rename of BehaveObj1)
                    if
```

5.2 Example

We perform eligible(HM4,R1) and the check fails, since Atrial.self=N1.self=N4.self∈N6.self. The check also suggest that N3.self and P6.cond should be seperated. If we go to HM3 and eligible(HM3,R1) is successful. So HM3 is the suggested heart model for requirement R1.

References

[1] Zhihao Jiang, Miroslav Pajic, and Rahul Mangharam. Cyber-Physical Modeling of Implantable Cardiac Medical Devices. *Proceeding of IEEE Special Issue on Cyber-Physical Systems*, 2011.

[2] Zhihao Jiang, Miroslav Pajic, Rajeev Alur, and Rahul Mangharam. Closed-loop verification of medical devices with model abstraction and refinement. *International Journal on Software Tools for Technology Transfer*, pages 1–23, 2013.

[3] Edmund Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith. Counterexample-guided abstraction refinement for symbolic model checking. *J. ACM*, 50(5):752–794, 2003.