Continuous ASM, and a Pacemaker Sensing Fragment

Richard Banach[1⋆], Huibiao Zhu[2⋆⋆], Wen Su[2], and Xiaofeng Wu[2]

[1]School of Computer Science, University of Manchester,
Oxford Road, Manchester, M13 9PL, U.K.
banach@cs.man.ac.uk

[2]Software Engineering Institute, East China Normal University,
3663 Zhongshan Road North, Shanghai 200062, P.R. China.
{hbzhu,wensu,xfwu}@sei.ecnu.edu.cn

Abstract. The ASM framework is extended to include continuously varying quantities as well as conventional discretely changing ones. This opens the door to the more faithful modeling of many scenarios where digital systems have to interact with the continuously varying physical world. Transitions in the extended framework are thus either *moded* (for discontinuous changing quantities), or *pliant* (for smoothly changing quantities). Refinement and retrenchment are defined in the extended context. The framework is used to develop a fragment of a simple system for the sensing problem for cardiac pacemakers, in the context of the pacemaker verification challenge.

1 Introduction

Conventional model based formal refinement technologies (see for example [1, 26, 2, 9]) are based on discrete mathematical and logical concepts. These are typically ill suited to modeling and developing applications whose models are expressed in continuous mathematics. Nevertheless, many such applications are these days implemented using digital techniques. So there is a mismatch between the ideal of continuous modeling at the abstract level, and the discrete techniques used close to implementation.

In this paper we present an extension of the ASM formalism that enables us to treat continuously changing quantities fluently, and we develop the accompanying extension of ASM refinement and retrenchment to cope with it. The ASM extension is based on restricting the continuous behaviours to solutions of well posed initial value problems. The resulting framework includes all the behaviours needed for the kind of engineering problems that arise in practice.

We apply this framework to a simple version of the sensing problem for heart pacemakers. Pacemakers have been proposed as a study problem for the Verification Grand

⋆ Work partly done while the first author was a visiting researcher at the Software Engineering Institute at East China Normal University. The support of ECNU is gratefully acknowledged.

⋆⋆ Huibiao Zhu is supported by National Basic Research Program of China (No. 2011CB302904), National High Technology Research and Development Program of China (No. 2011AA010101 and No. 2012AA011205), National Natural Science Foundation of China (No. 61061130541 and No. 61021004).

Challenge [13, 24, 25]. Pacemakers are interesting from this viewpoint since the historical approach to their evolution and development is so deeply ingrained in clinical practice and experimentation [7, 11]. From the formal point of view, the most pressing question that this prompts is: "what exactly are the requirements?"

The rest of this paper is as follows. In Section 2 we briefly introduce pacemakers, the pacemaker challenge and work done to date, and our own focus: pacemaker sensing. In Section 3 we review ASM modeling, we give the extension to continuous phenomena, and we develop the relevant refinement and retrenchment machinery. Section 4 explores the sensing problem in more detail. Section 5 presents some ASM models for the sensing problem, starting with a simple reference model. Section 6 concludes.

2 Heart Pacemakers, the Pacemaker Challenge, and Sensing

The heart has two atria and two ventricles. Blood collects in the atria, and is decanted into the ventricles by a wave of muscular contraction stimulated by the sinoatrial node. A short time later, another powerful wave of contraction in the ventricles pumps the blood round the body.

The heart beats when it is told to do so (via an electrical pulse) by its environment, in this case the sinoatrial node which initiates atrial depolarization. In normal working, the atrial pulse has to be of the right characteristics to cause depolarization. Similarly, the ventricular pulse causes the ventricular depolarization to happen. Inevitably, various things can go wrong with the nervous mechanisms that cause all this to happen. Problems of different kinds can arise with atrial depolarization, with ventricular depolarization, with both, and with the relationship between them. Such deficiencies are collectively referred to as heart block. Heart block can be addressed by the implantation and configuring of pacemakers, which supply electrical pulses that substitute for ones that the body is unable to generate properly itself.

Over the years, pacemakers have evolved into very sophisticated devices. To facilitate research into the computing dimension of pacemaker technology, a public domain specification of a pacemaker system has been produced by Boston Scientific [10] for use in the Verification Grand Challenge [13, 24, 25].

A perusal of this document reveals that the reader has to rely on a very large amount of additional knowledge. The most self-contained and relatively complete part of [10] deals with the different "modes" of pacemaker working. The wide range of possible electrical stimulation defects gives rise to a corresponding range of modes of pacemaker operation, each designed to address a specific defect. Each of these modes is assembled out of a number of available pacemaker features, where each relevant feature of a mode is tunable by the physician within a given range. Aided by a moderate amount of supplementary information, this aspect of pacemaker operation becomes tractable, and has attracted some interest from the verification community [16, 12, 17].

Normally, the pacemaker listens to the heart's activity. When it detects that the heart has generated a depolarization, it suppresses the artificial pulse — this is sensing. The heart's electrical activity can be detected on the patient's skin (via an electrocardiogram, (ECG)), and inside the heart itself (via a cardiac electrogram (EG)). Fig. 1 illustrates. Since the pacemaker is implanted inside the patient, it is the EG that it must sense.

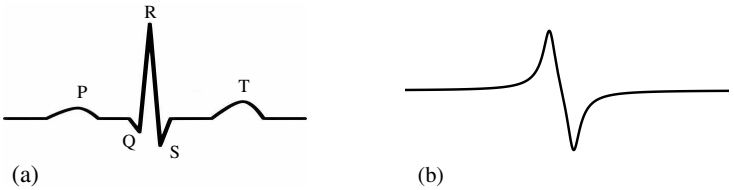

Fig. 1. (a) A surface electrocardiogram (ECG). (b) An internal cardiac electrogram (EG).

In the clinical view of sensing, e.g. in [21, 3, 15], it is assumed that sensing is done by a filter circuit. This works in the frequency domain, registering electrical activity in the relevant frequency range, suppressing other frequencies. The circuit outputs a discrete "Yes" or "No" according to the power spectrum of the filtered input.

The majority of the literatures in this area simply assume that the pacemaker knows whether or not a pulse has occured. Nevertheless, the clinical literature always cautions that there is a significant risk of sensing circuits confusing a true depolarization with other electrical activity. The source of this problem is the following.

The circuit senses only the power spectrum, so the phase information in the signal is discarded. Consequently, signals having the right power spectrum, but with a shape very different from Fig. 1.(b) can be erroneously identified with a depolarization.

In this paper, we illustrate our continuous extension of the ASM formalism by supplementing frequency domain sensing with a time domain tracking of the EG. This seeks to distinguish genuine depolarizations from spurious other signals sharing a similar power spectrum. Since pacemakers are capable of measuring the internal EG ([10] explicitly demands such a capability), this is an entirely credible strategy.

3 ASM, Discrete and Continuous

In this section we review the essentials of ASM [9, 8], and extend the formalism to cope with continuously varying quantities, extending its reach to address many problems not treatable using the purely discrete theory alone.

3.1 Continuous ASM Models

To keep things simple, we assume that the states of an ASM model are given by valuations of the tuple of variables relevant to the model, i.e. functions from the tuple of variables to the tuple of the variables' types. To extend such models to include continuously varying phenomena, we partition the variables into two subsets: the **mode variables**, whose types are discrete sets, and which are therefore only permitted to change discontinuously and discretely, and the **pliant variables**, whose types include topologically dense sets, and which are permitted to evolve both continuously and via discrete changes. By restricting to mode variables alone, we recover the conventional discrete ASM framework.

We model time as an interval $\mathcal{T}$ of the real numbers $\mathbb{R}$, with a finite left endpoint representing the time at which the initial state of the model is created, and with a right

endpoint which is either finite or infinite, depending on whether the dynamics is finite or infinite. Now, the values of all variables become functions of $\mathcal{T}$. For the mode variables, this function is a piecewise constant function, constant on each element of a sequence of left-closed right-open intervals. Thus $\mathcal{T}$ itself also partitions into a sequence of left-closed right-open intervals, $\langle [t_0 \ldots t_1), [t_1 \ldots t_2), \ldots \rangle$, the coarsest partition of $\mathcal{T}$ such that all discontinuous changes take place at some boundary point t_i.

In a typical interval $[t_i \ldots t_{i+1})$, the mode variables will be constant, but the pliant variables will change continuously. However, merely insisting on continuity still allows for a wide range of mathematically pathological behaviours. To eliminate these, we make the following restrictions:

I Zeno: there is a constant δ_{Zeno}, such that for all i needed, $t_{i+1} - t_i \geq \delta_{\mathsf{Zeno}}$.[1]

II Limits: for every variable x, and for every time $t \in \mathcal{T}$, the left limit $\lim_{\delta \to 0} x(t-\delta)$ written $\overrightarrow{x(t)}$ and right limit $\lim_{\delta \to 0} x(t+\delta)$, written $\overleftarrow{x(t)}$ (with $\delta > 0$) exist, and for every t, $x(t) = \overleftarrow{x(t)}$. [N. B. At the endpoint(s) of $\mathcal{T}$, any missing limit is defined to equal its counterpart.]

III Differentiability: The behaviour of every pliant variable x in the interval $[t_i \ldots t_{i+1})$ is given by the solution of a well posed initial value problem $\mathcal{D}\, xs = \phi(xs, t)$ (where xs is a relevant tuple of pliant variables and $\mathcal{D}$ is the time derivative). "Well posed" means that $\phi(xs, t)$ has Lipschitz constants which are uniformly bounded over $[t_i \ldots t_{i+1})$ bounding its variation with respect to xs, and that $\phi(xs, t)$ is measurable in t.

It is recognised that ASM types can be mathematically complex entities. Therefore it is intended that I-III above apply to variables with as general a type as might be needed, provided that the concepts required in I-III (left/right limits, initial value problem, Lipschitz constants, uniform boundedness, measurability) make sense for them.

With I-III in place, the behaviour of every pliant variable is piecewise absolutely continuous, with the variation being described by a suitable differential equation (DE).

Accompanying the distinction between mode and pliant variables, is a distinction between mode and pliant transitions. Mode transitions are just like conventional ASM transitions in that they record a discrete transition from before-values to after-values of the mode variables, albeit that these are the values of piecewise constant functions of time. A rule for a mode transition OP can be written using familiar ASM notation:

$$\begin{array}{l} \textsc{Op}(\textbf{in } \overrightarrow{is}, \textbf{out } \overleftarrow{os}) \;= \\ \textbf{if } \; guard(\overrightarrow{xs}, \overrightarrow{is}) \;\; \textbf{then choose } \; \overleftarrow{xs}, \overleftarrow{os} \\ \quad \textbf{with } \; rel(\overleftarrow{xs}, \overrightarrow{xs}, \overrightarrow{is}, \overleftarrow{os}) \;\; \textbf{do } \; xs, os \;:=\; \overleftarrow{xs}, \overleftarrow{os} \end{array} \tag{1}$$

In (1) we single out is and os, the inputs and outputs (read-only and write-only respectively), while xs are the state variables (accessed in read/write manner). Note that the

[1] Our approach to the Zeno problem contrasts with many others, which demand that any finite time interval contains only a finite number of transitions, or that the sequence of transition times contains no accumulation points. But this permits the sequence $t_{i+1} - t_i = 1/i$, which satisfies the mentioned restrictions, yet allows transitions to get arbitrarily close together.

choice of left limit for before-values and right limit for after-values makes (1) into the kind of instantaneous transition that we would expect. Also, if the after-values for xs and os are available explicitly, the relevant expression can be assigned in the **do** clause, and the **choose** and **with** clauses can be omitted.

Pliant transitions do the corresponding job for pliant variables. While a mode transition is a single before-/after-value pair, a pliant transition is a family of before-/after-value pairs parameterized by the relevant time interval $[t_i \ldots t_{i+1})$. Moreover, instead of the change from before-values to after-values taking place instantaneously, the before-value refers to the initial value at t_i while the after-value refers to an arbitrary time in the interval, so the before-value and after-value are separated in time. To reflect the constraints that apply to pliant transitions, we write rules for them thus:

$$\begin{array}{l} \textsc{PliOp}(\textbf{in}\ is(t \in (t_{\mathrm{L}(t)} \ldots t_{\mathrm{R}(t)})), \textbf{out}\ os(t \in (t_{\mathrm{L}(t)} \ldots t_{\mathrm{R}(t)}))) \stackrel{\mathrm{c}}{=} \\ \textbf{if}\ \ IV(xs(t_{\mathrm{L}(t)})) \wedge guard(xs(t_{\mathrm{L}(t)}))\ \ \textbf{then with}\ \ rel(xs, is, os, t) \\ \quad \textbf{do}\ \ xs(t), os(t)\ :=\ \textbf{solve}\ DE(xs(t), is(t), os(t), t) \end{array} \tag{2}$$

In (2), the symbol $\stackrel{\mathrm{c}}{=}$ signals the presence of a pliant transition, distinguishing it from the instantaneous kind. The inputs is and outputs os are continuously absorbed from and emitted to the environment, as indicated in the signature. For an arbitrary t, we let $\mathrm{L}(t) = \max\{i \,|\, t_i \leq t\}$ and $\mathrm{R}(t) = \min\{i \,|\, t_i > t\}$ so that: (a) we do not have to know the index i explicitly in $[t_i \ldots t_{i+1})$; (b) we can refer to the beginning and end of the interval during which the pliant event runs in a generic manner in the syntactic description of the event. Note that the initial values IV and guard $guard$ depend only on the before-value of the state, and not on the input, whereas rel, which expresses any additional constraints that must hold beyond the differential equation DE itself, can depend on all state and input values from the start of the interval $t_{\mathrm{L}(t)}$ up to the current time t. The assignment in (2) says that the after-state and output at t should satisfy the differential equation DE (as well as rel). As for the instantaneous case, if the continuous functions of t to be assigned to xs, os are known explicitly, we can omit the **with** and/or **solve** clauses as approporiate, and just assign xs, os to the relevant expression.

As mentioned earlier, pliant variables can undergo instantaneous discontinuous transitions as well as continuous ones. For such transitions, the structure in (1) is sufficient. We continue to call instantaneous transitions involving both kinds of variable **mode transitions**, introducing the term **pure mode transitions** for the former kind.

We say that a continuous ASM ruleset is **well formed** iff:

- Every enabled mode transition is feasible, i.e. has an after-state, and on its completion enables a pliant transition (but does not enable any mode transition). (3)
- Every enabled pliant transition is feasible, i.e. has a time-indexed family of after-states, and EITHER: (4)

  (i) During the run of the pliant transition a mode transition becomes enabled. It preempts the pliant transition, defining its end. ORELSE
  (ii) During the run of the pliant transition it becomes infeasible: finite termination. ORELSE
  (iii) The pliant transition continues indefinitely: nontermination.

A **run** of a continuous ASM system starts with a mode transition which creates the initial state, and then, pliant transitions alternate with mode transitions. The last transition (if there is one) is a pliant transition (whose duration may be finite or infinite).

3.2 Continuous ASM Refinement and Retrenchment

Now we develop our continuous ASM framework to encompass refinement and retrenchment of continuous ASM models. We start by describing the usual formulation, appropriate to pure mode transitions, and then show how to extend this to encompass the new kinds of transition.

In general, to prove a conventional ASM refinement or retrenchment, we verify so-called (m, n) diagrams, in which m abstract steps simulate n concrete ones in an appropriate way. This means that there is nothing that the n concrete steps can do that is not suitably reflected in m appropriately chosen abstract steps, where both m and n can be freely chosen to suit the application, and the meaning of "suitably reflected" depends on whether we are dealing with refinement or retrenchment. For this paper, it will be sufficient to focus on the refinement and retrenchment proof obligations (POs) which are the embodiment of this policy. The situation for refinement is illustrated in Fig. 2, in which we suppress input and output for clarity.

In Fig. 2 the retrieve relation $R_{\mathrm{A,C}}$, between abstract and concrete states, holds at the beginning and end of the (m, n) pair. This permits us to "glue together" such (m, n) diagrams to create relationships between abstract and concrete runs in which $R_{\mathrm{A,C}}$ is periodically re-established. [N. B. In much of the ASM literature, the main focus is on an *equivalence*, usually written $\equiv$, between abstract and concrete states. This is normally deemed to contain a "practically useful" subrelation $R_{\mathrm{A,C}}$, chosen to be easier to work with. The approach via $R_{\mathrm{A,C}}$ will be the focus of our treatment, and is also focus of the KIV [14] formalization in [19, 20].]

The first PO is the initialization PO, common to both refinement and retrenchment:

$$\forall\, y' \bullet CInit(y') \Rightarrow (\exists\, x' \bullet AInit(x') \wedge R_{\mathrm{A,C}}(x', y')) \tag{5}$$

In (5), it is demanded that for each concrete initial state y', there is an abstract initial state x' such that the retrieve relation $R_{\mathrm{A,C}}(x', y')$ holds.

The second PO is correctness. The PO is concerned with the verification of (m, n) diagrams. For this, we have to have some way of deciding which (m, n) diagrams are sufficient for the application. Let us assume that we have done this. Let $CFrags$ be the set of fragments of concrete runs that we have previously determined will permit a covering of all the concrete runs of interest for the application. We write $y :: ys :: y' \in CFrags$ to denote an element of $CFrags$ starting with concrete state y, ending with concrete state y', and with intervening concrete state sequence ys. Likewise we write $x :: xs :: x' \in AFrags$ for abstract fragments. Let is, js, os, ps denote the sequences of abstract inputs, concrete inputs, abstract outputs, concrete outputs, respectively, belonging to $x :: xs :: x'$ and $y :: ys :: y'$ and let $In_{\mathrm{AOPS,COPS}}(is, js)$ and $Out_{\mathrm{AOPS,COPS}}(os, ps)$ denote suitable input and output relations. The specific form of the correctness PO now

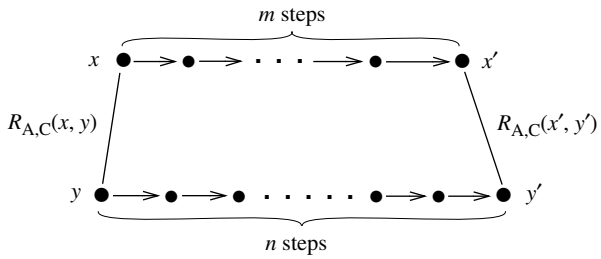

Fig. 2. An ASM (m, n) diagram, showing how m abstract steps, going from state x to state x' simulate n concrete steps, going from y to y'. The simulation is embodied in the retrieve relation $R_{\mathrm{A,C}}$, which holds for the before-states of the series of steps $R_{\mathrm{A,C}}(x, y)$, and is re-establshed for the after-states of the series $R_{\mathrm{A,C}}(x', y')$.

differs in form depending on whether we are dealing with refinement or retrenchment. We start with refinement. Then the correctness PO reads:

$$\begin{array}{l} \forall\, x, is, y, ys, y', js, ps \bullet y :: ys :: y' \in CFrags \wedge \\ \quad R_{\mathrm{A,C}}(x, y) \wedge In_{\mathrm{AOPS,COPS}}(is, js) \wedge \mathrm{COPS}(y :: ys :: y', js, ps) \;\Rightarrow \\ \quad\quad (\exists\, xs, x', os \bullet x :: xs :: x' \in AFrags \wedge \mathrm{AOPS}(x :: xs :: x', is, os) \wedge \\ \quad\quad\quad R_{\mathrm{A,C}}(x', y') \wedge Out_{\mathrm{AOPS,COPS}}(os, ps)) \end{array} \tag{6}$$

In (6), it is demanded that whenever there is a concrete run fragment of the form $\mathrm{COPS}(y :: ys :: y', js, ps)$, carried out by a sequence of concrete operations[2] COPS, with state sequence $y :: ys :: y'$, input sequence js and output sequence ps, such that the retrieve and input relations $R_{\mathrm{A,C}}(x, y) \wedge In_{\mathrm{AOPS,COPS}}(is, js)$ hold between concrete and abstract before-states and inputs, then an abstract run fragment $\mathrm{AOPS}(x :: xs :: x', is, os)$ can be found to re-establish the retrieve and output relations $R_{\mathrm{A,C}}(x', y') \wedge Out_{\mathrm{AOPS,COPS}}(os, ps)$.

The ASM refinement policy also demands that non-termination be preserved from concrete to abstract, but we will not need that in this paper.

Assuming that (5) holds, and that we can prove enough instances of (6) to cater for the application of interest, then the concrete model is a **correct refinement** of the abstract model. In a correct refinement, all the properties of the concrete model (that are visible through the retrieve and other relations), are suitably reflected in properties of the abstract model (because of the direction of the implication in (6)). If in addition, the abstract model is also a correct refinement of the concrete model (using the converses of the same relations), then the concrete model is a **complete refinement** of the abstract model. In a complete refinement, all relevant properties of the abstract model are also present in the concrete model (because of the direction of the implication in the modified version of (6)). Therefore, to ensure that the complete set of requirements

[2] We define an operation as a maximal enabled set of rules — provided its updates are consistent. Enabled inconsistent updates cause abortion of the run.

of an intended system is faithfully preserved through a series of refinement steps, it is enough to express them all in a single abstract model, and then to ensure that each refinement step is a complete refinement. We now turn to the retrenchment version of the correctness PO.

For retrenchment, [6, 5] give definitive accounts; latest developments are found in [18]. See also [4] for formulations of retrenchment adapted to several specific model based refinement formalisms including ASM. The retrenchment correctness PO weakens (6) by inserting *within*, *output* and *concedes* relations, $W_{\text{AOPS,COPS}}$, $O_{\text{AOPS,COPS}}$, $C_{\text{AOPS,COPS}}$ respectively into (6), to give extra flexibility and expressivity. In particular, the concession $C_{\text{AOPS,COPS}}$ weakens the conclusions of (6) disjunctively, giving room for many kinds of "exceptional" behaviour. The result is:

$$\begin{array}{l}\forall\, x, is, y, ys, y', js, ps \bullet y :: ys :: y' \in CFrags \wedge \\ \quad R_{\text{A,C}}(x, y) \wedge W_{\text{AOPS,COPS}}(is, js, x, y) \wedge \text{COPS}(y :: ys :: y', js, ps) \;\Rightarrow \\ \quad\quad (\exists\, xs, x', os \bullet x :: xs :: x' \in AFrags \wedge \text{AOPS}(x :: xs :: x', is, os) \wedge \\ \quad\quad\quad ((R_{\text{A,C}}(x', y') \wedge O_{\text{AOPS,COPS}}(x, x', is, os, y, y', js, ps)) \vee \\ \quad\quad\quad\quad C_{\text{AOPS,COPS}}(x, x', is, os, y, y', js, ps))) \end{array} \tag{7}$$

To ensure that retrenchment only deals with well defined transitions, and to ensure smooth retrenchment/refinement interworking, in retrenchment we also insist that $R_{\text{A,C}} \wedge W_{\text{AOPS,COPS}}$ always falls in the domain of the requisite operations, though this is another thing not needed here.

All of the preceding was still formulated for the exclusively discrete world. However, to extend it to the continuous world too, is simplicity itself. We just have to reinterpret the paths, $x :: xs :: x'$ and $y :: ys :: y'$ appearing in (6) and (7) appropriately, and we are done.

More precisely, a path like $x :: xs :: x'$ say, can consist of interleavings of pliant and mode transitions. If $x :: xs :: x'$ starts with a pliant transition, then the value x used in the POs (6) and (7) is the right limit at its initial point x. Similarly, if $x :: xs :: x'$ ends with a pliant transition, then the value x' used in the POs is the left limit at its endpoint x'. Similar remarks apply to the concrete path $y :: ys :: y'$. The fact that the POs are largely insensitive to what goes on in the interior of the paths, makes them equally applicable to paths that interleave pliant and mode transitions, as to paths that just have a sequence of discrete states in their interior.

4 Pacemaker Sensing

Our objective for the rest of this paper is to design an ASM system to track a signal that represents the continuous internal electrical activity of the heart. A theoretical treatment of the expected shape of the signal detected by an electrode inside the heart as a wave of depolarization passes over it has been carried out some time ago, based on the idea of a dipole of charge passing a detector [22, 23]. This is the basic shape in Fig. 1.(b).

Thus, our abstract model focuses on Fig. 1.(b), allowing for an acceptable margin of error. This is shown in Fig. 3. There, the basic shape $pulse(t)$ (solid curve in Fig. 3) is given by (8), while the dashed error curves surrounding it are given by adding (or

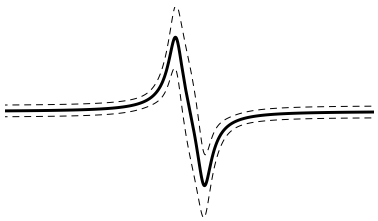

Fig. 3. Schematic of the EG of a single depolarization, with a margin of error to allow for noise.

subtracting) $err(t)$, which is a small positive constant augmented by a strongly peaked Gaussian centred at the point of maximum variability of $pulse(t)$:

$$pulse(t) \;=\; -\,K\left(\frac{1}{\sqrt{(t+a)^2+b^2}} - \frac{1}{\sqrt{(t-a)^2+b^2}}\right) \tag{8}$$

$$err(t) \;=\; c + r\exp(-p\,t^2) \tag{9}$$

In real pacemakers, the completion of the previous heartbeat initiates the start of a fresh cycle. Let us assume that this is at time $t = 0$. The new cycle begins with a so called refractory period, in which the heart signal is ignored, allowing the transients of the previous heartbeat to die off, and for myocardial polarization to be re-established. Say this refractory period lasts till T_0. Then, from T_0 to T_{MAX}, the heart signal is monitored for the presence of a shape similar to $pulse$. Assuming the heart is working normally, if the patient is being physically active, then the patient's heartrate will be higher and the pulse occurs sooner; whereas if the patient is being inactive then the heartrate will be lower and the pulse will occur later. Thus, we can specify a typical function $egm_A(t)$ that constitutes an acceptable electrogram shape as follows:

$$\begin{aligned}&(\exists\, R \bullet T_0 < R_0 \le R \le R_{MAX} < T_{MAX} \wedge \\ &\quad (\forall\, t \bullet T_0 < t < T_{MAX} \Rightarrow egm_A(t) \in egmWin_R(t)))\end{aligned} \tag{10}$$

where

$$egmWin_R(t) = [egmWin^-_R(t) \ldots egmWin^+_R(t)] \tag{11}$$

where

$$egmWin^-_R(t) = pulse(t-R) - err(t-R) \tag{12}$$

and

$$egmWin^+_R(t) = pulse(t-R) + err(t-R) \tag{13}$$

In (10), $egmWin_R(t)$ is set valued, and at each relevant time t, it specifies the allowable cardiac electrogram window within which a normally operating heart's electrogram behaviour is expected to fall. A minor complication is that the precise moment within the allowed timeframe at which the electrogram undergoes the pulse is not known exactly, but is merely constrained by $R_0 \le R \le R_{MAX}$, where, like T_0 and T_{MAX}, the constants R_0 and R_{MAX} are statically determined.

5 Sensing Models

5.1 The Reference Model

At the abstract level, we can view $egmWin_R(t)$ as as static function, depending on two parameters: time and the chosen value of R. With this, we can specify a reference model for cardiac behaviour using a pliant rule that generates acceptable electrograms as follows, where any call of EGM_A produces an electrogram egm_A which stays within the acceptable window $egmWin_R$ during the period of interest, $T_0 < t < T_{MAX}$:

$$\begin{array}{l}
\text{EGM}_A(\textbf{out}\ ego(t)) \stackrel{\text{c}}{=} \\
\textbf{choose}\ eg'(t) \\
\textbf{with}\ (\exists\, R \bullet T_0 < R_0 \leq R \leq R_{MAX} < T_{MAX} \wedge T_0 < t < T_{MAX} \wedge \\
\qquad eg'(t) \in egmWin_R(t)) \\
\quad \textbf{do}\ egm(t)\ :=\ eg'(t), ego(t)\ :=\ eg'(t)
\end{array} \tag{14}$$

5.2 Heartrate-Aware ASM Sensing

We now present the continuous variable heartrate sensing model, the CVH model. Initialisation (assumed to be at time 0), resets the "verdict" variables $pulseGOOD$ and $pulseBAD$, and initialises the pliant variables dev, $egirise$ and $egifall$ which will measure the quality of the electrogram. Then it models the refractory period.

$$\begin{array}{l}
\text{INIT}_{CVH} = \\
\textbf{if}\ t = 0\ \textbf{then} \\
\quad \textbf{do}\ mode\ :=\ refrac, pulseGOOD\ :=\ \mathsf{false}, pulseBAD\ :=\ \mathsf{false}, \\
\qquad winR\ :=\ \emptyset, dev\ :=\ 0, egirise\ :=\ 0, egifall\ :=\ 0
\end{array} \tag{15}$$

$$\text{REFRACTORY}_{CVH} \stackrel{\text{c}}{=} \textbf{if}\ mode = refrac\ \textbf{then do}\ \mathsf{skip} \tag{16}$$

A mode transition signals the start of sensing:

$$\begin{array}{l}
\text{STARTSENSING}_{CVH} = \\
\textbf{if}\ mode = refrac \wedge t = T_0\ \textbf{then} \\
\quad \textbf{do}\ mode\ :=\ sensing, pulseGOOD\ :=\ \mathsf{false}, pulseBAD\ :=\ \mathsf{false}, \\
\qquad winR\ :=\ \emptyset, dev\ :=\ 0, egirise\ :=\ 0, egifall\ :=\ 0
\end{array} \tag{17}$$

The switch to sensing mode activates the sensing process.

$$\begin{array}{l}
\text{EGM}_{CVH}(\textbf{in}\ egi(t)) \stackrel{\text{c}}{=} \\
\textbf{let}\ wR(\bar{s}) = \textbf{\textit{if}}\ \bar{s} \notin \operatorname{dom} egi\ \textbf{\textit{then}}\ \emptyset\ \textbf{\textit{else}}\ \{R \mid R_0 \leq R \leq R_{MAX} \wedge \\
\qquad\qquad \forall \tilde{s} \bullet t_{\text{L}(t)} < \tilde{s} \leq \bar{s} \Rightarrow egi(\tilde{s}) \in egmWin_R(\tilde{s})\}\ \textbf{\textit{fi}}\ \textbf{in} \\
\textbf{let}\ s = \max\{\tilde{s} \mid t_{\text{L}(t)} < \tilde{s} \leq t \wedge wR(\tilde{s}) \neq \emptyset\}\ \textbf{in} \\
\textbf{if}\ mode = sensing\ \textbf{then} \\
\quad \textbf{do}\ winR(t)\ :=\ wR(s), \\
\qquad dev(t)\ :=\ \textbf{solve}\ \mathcal{D}\, dev(t) = \\
\qquad\qquad \min\{\text{dist}(egi(t), egmWin_R(t)) \mid R \in winR(t)\}, \\
\qquad egirise(t)\ :=\ \textbf{solve}\ \mathcal{D}\, egirise(t) = \mathcal{D}\, egi(t)\, \Theta(\mathcal{D}\, egi(t)), \\
\qquad egifall(t)\ :=\ \textbf{solve}\ \mathcal{D}\, egifall(t) = \mathcal{D}\, egi(t)\, \Theta(-\mathcal{D}\, egi(t))
\end{array} \tag{18}$$

In (18), Θ returns 1 if its parameter is positive, else 0. Sensing continues until a signal conforming to the region of high variation in $pulse$ has been detected (indicating a spontaneous heartbeat), or the timeout expires (indicating abnormal cardiac function).

Consider a normal heartbeat. If $egi(t)$ is close to zero near $t = T_0$, then a large range of R values will satisfy $egi(t) \in egmWin_R(t)$. As time progresses, the normal heartbeat reaches the point at which the double spike occurs. As the electrogram follows the shape of $pulse$, very soon the range of allowable R values is reduced to around $2t_0$ (where t_0 is the solution nearest to 0 to the equation $pulse(t_0) + err(t_0) = 0$), which is the width of the time window around the central point of $pulse$ in Fig. 3. Setting δ_{th} (the width of the window of allowable R values such that $egmWin_R$ contains the whole of the observed electrogram (which must be more than zero)) to slightly more than $2t_0$, gives us a feature to test for when confirming the presence of a normal heartbeat.

Also, since the whole of the observed electrogram is inside $egmWin_R$, $\mathcal{D}\,dev(t)$ remains at zero, and so dev stays at zero too, giving another feature to test for when confirming a normal heartbeat. And since $egirise$ and $egifall$ track the overall positive and negative change in the electrogram value, a normal heartbeat will also be characterised by $|egirise + egifall| < \delta_{rf}$ and $egirise > \Delta_{th}$, with δ_{rf} and Δ_{th} suitable constants (one small, one big). Observing also that the time for confirming a normal heartbeat will be less than T_{MAX}, a normal heartbeat enables the following mode transition:

$$
\begin{array}{l}
\textsc{PulseGood}_{CVH} = \\
\textbf{if}\ \ mode = sensing \wedge dev = 0 \wedge 0 < |\max\ winR - \min\ winR| < \delta_{th} \wedge \\
\quad |egirise + egifall| < \delta_{rf} \wedge egirise > \Delta_{th} \wedge T_0 < t < T_{MAX}\ \ \textbf{then} \\
\quad\ \textbf{do}\ \ mode\ :=\ refrac, pulseGOOD\ :=\ \mathsf{true}
\end{array}
\tag{19}
$$

Consider an abnormal heartbeat. By definition, an abnormal heartbeat does *not* conform to the envelope of permitted deviation around $pulse$. Thus $winR(t)$ will diminish over time, but instead of eventually remaining at a little less than δ_{th}, it will shrink to a single value, the last value $\bar{R}$ at the last time $\bar{t}$ for which the "electrogram-so-far remains in $egmWin_{\bar{R}}$" property still holds.

After this moment, the distance between $egi(t)$ and $egmWin_{\bar{R}}(\bar{t})$ will become positive, leading to a positive $\mathcal{D}\,dev(t)$ and thus positive dev, which remains until the deadline T_{MAX} expires, giving a feature to test for when confirming an abnormal heartbeat. This enables the following mode transition:

$$
\begin{array}{l}
\textsc{PulseBad}_{CVH} = \\
\textbf{if}\ \ mode = sensing \wedge dev > 0 \wedge t = T_{MAX}\ \ \textbf{then} \\
\quad\ \textbf{do}\ \ mode\ :=\ refrac, pulseBAD\ :=\ \mathsf{true}
\end{array}
\tag{20}
$$

To deal with successive heartbeats, we need to slightly alter the way we deal with time. There are two relatively straightforward approaches. In the first, we continue with the perspective that t refers to real time (measured from some arbitrary starting point). In this view, we would need $\textsc{PulseGood}_{CVH}$ and $\textsc{PulseBad}_{CVH}$ to re-assign the values of T_0 and T_{MAX} to appropriate future values (by adding to them the duration of the heartbeat that has just elapsed), ready for the next heartbeat. In the second, we would regard t as a *clock* — which is reset at the beginning of every heartbeat (via assignments $t := 0$ in $\textsc{PulseGood}_{CVH}$ and $\textsc{PulseBad}_{CVH}$) instead of referring to

real time. In this view, we would need to re-interpret the notations $t_{\mathrm{L}(t)}$ and $t_{\mathrm{R}(t)}$ so that they referred to the clock time at the beginning and end of the current invocation of the current pliant transition, rather than to real time. Both of these approaches are quite straightforward, allowing $\textsc{Refractory}_{CVH}$ to be re-enabled for a repetition of the sensing behaviour in the next heartbeat.

5.3 On $\textsc{Egm}_A$ and $\textsc{Egm}_{CVH}$

In a full-blown formal development, the relationship between $\textsc{Egm}_A$ and the more concrete $\textsc{Egm}_{CVH}$ would be of interest in monitoring how the pacemaker sensing requirements were being met through the development. Here, for lack of space, we just sketch an aspect of it as an illustration of the flexibility of our techniques.

The main use of pacemakers is in situations where cardiac behaviour is expected to *not* conform to the normal for a significant proportion of the time. This makes the prospects for a conventional kind of refinement futile. On the other hand, if we take advantage of the additional flexibility of retrenchment, then the prospects for setting up a formal relationship between $\textsc{Egm}_A$ and $\textsc{Egm}_{CVH}$ are improved. We now give one possible such retrenchment from $\textsc{Egm}_A$ to $\textsc{Egm}_{CVH}$.

The first concern in a retrenchment is the retrieve relation from the A model state space to the CVH model state space, $R_{A,CVH}$. Given that the two models operate on different timeframes, that the size of the allowed window of R values $egmWin_R$, varies over time, and recognising that the other relations of a retrenchment permit more finegrained control over states, it is acceptable to trivialise $R_{A,CVH}$:

$$R_{A,CVH}(egm, \langle winR, dev\rangle) \;\equiv\; \mathsf{true} \tag{21}$$

The within relation controls what is demanded of before-states and inputs in the hypotheses of the correctness PO (7):

$$\begin{aligned} &W_{\textsc{Egm}_A,\textsc{Egm}_{CVH}}(egi(t \in (T_0 \ldots T')), egm(T_0), \langle winR, dev\rangle(T_0)) \;\equiv \\ &\quad dev(T_0) = 0 \wedge egirise = 0 \wedge egifall = 0 \wedge |egi(T_0)| < \delta_{\mathrm{small}} \end{aligned} \tag{22}$$

In (22), T' refers to the time at which the after-state of the heartbeat is reached. We see that $dev(T_0) = 0$ and $egirise = 0$ and $egifall = 0$ are all recorded in (22), consistent with (17), and with the fact that these initial values are not mentioned in (18). We also record that the right limit of the sensed electrogram, $egi(T_0)$, is small enough to substantiate our earlier assumptions that egi is near zero at the start of sensing.

Now, either $\textsc{Egm}_{CVH}$ conforms to $\textsc{Egm}_A$ and we have a normal heartbeat, or not. In the former case, it is appropriate to describe the properties of the two executions using the output relation:

$$\begin{aligned} &O_{\textsc{Egm}_A,\textsc{Egm}_{CVH}}(egm(T_0, T'), ego(t \in (T_0 \ldots T')), \\ &\qquad\qquad \langle winR, dev\rangle(T_0, T'), egi(t \in (T_0 \ldots T'))) \;\equiv \\ &\quad dev(T') = 0 \wedge 0 < |\max\, winR - \min\, winR| < \delta_{th} \wedge \\ &\quad |egirise + egifall| < \delta_{rf} \wedge egirise > \Delta_{th} \wedge \\ &\quad (\forall\, t \bullet T_0 < t < T' \Rightarrow \textsc{Egm}_A(ego(t)) \wedge ego(t) = egi(t)) \end{aligned} \tag{23}$$

The gist of (23) is that the variables dev, $egirise$ and $egifall$ behaved as expected, and also that, since we had a normal heartbeat, the input electrogram $egi(t)$ always remained within the permitted envelope, $egmWin_R(t)$ for some R, and therefore that there is a possible abstract electrogram $ego(t)$ that followed it exactly.

If though, we are dealing with the case of an abnormal heartbeat, we can describe the properties of the two executions using the concession:

$$\begin{aligned} &C_{\text{EGM}_A,\text{EGM}_{CVH}}(egm(T_0,T'), ego(t \in (T_0 \ldots T')), \\ &\qquad\qquad \langle winR, dev\rangle(T_0,T'), egi(t \in (T_0 \ldots T'))) \ \equiv \\ &\quad T' = T_{MAX} \wedge dev(T') > 0 \wedge (\forall\, R \bullet R_0 \leq R \leq R_{MAX} \Rightarrow \\ &\qquad (\exists\, t \bullet T_0 < t < T' \wedge egi(t) \notin egmWin_R(t))) \end{aligned} \tag{24}$$

In (24), the first clause states that the deadline has expired, while the second states that the accumulated deviation from (even the best possible) permitted electrogram windows is positive. The third clause makes the preceding more explicit by asserting the existence of a value t at which the observed electrogram falls outside the permitted window $egmWin_R(t)$ (for even the best possible R).

This completes the relationship bewtween EGM_A and EGM_{CVH}. As hinted above, the step from EGM_A to EGM_{CVH} constitutes but the first step of a formal development of the time domain sensing application. The remainder of the development throws up some fascinating technical challenges for our formal framework, which, unfortunately, we do not have the space to explore in the present paper.

6 Conclusion

In the preceding sections, we introduced the pacemaker sensing problem as a case study that was not only connected with the Verification Grand Challenge [13, 24, 25], but was also one that required continuous machinery to attain reasonably faithful modeling. We then gave an extension of the ASM framework to enable it to deal with continuous behaviours. These behaviours were not arbitrary, but were constrained by a Zeno condition, and the need to be describable using differential equations with right hand sides that are Lipschitz in the dependent variables and measurable in time. This class is flexible enough to include naturally occurring engineering discontinuities, without admitting unnecessarily pathological behaviours.

We then applied this framework to our case study. For lack of space, we were not able to pursue this beyond the first stage, giving merely a taste of both the richness of the case study, and of our framework's capabilities. A more extensive treatment, pursuing the development of the case study to near-implementation, will appear elsewhere.

References

1. Abrial, J.R.: The B-Book: Assigning Programs to Meanings. Cambridge University Press (1996)
2. Abrial, J.R.: Modeling in Event-B: System and Software Engineering. Cambridge University Press (2010)

3. Aubert, A., Goldreyer, B., Wyman, M., Jaquemlyn, E., Ector, H., de Geest, H.: Filter Characteristics of the Atrial Sensing Circuit of a Rate Responsive Pacemaker. To See or Not to See. PACE 12, 525–536 (1989)
4. Banach, R.: Model Based Refinement and the Design of Retrenchments. Available from [18].
5. Banach, R., Jeske, C., Poppleton, M.: Composition Mechanisms for Retrenchment. J. Log. Alg. Prog. 75, 209–229 (2008)
6. Banach, R., Poppleton, M., Jeske, C., Stepney, S.: Engineering and Theoretical Underpinnings of Retrenchment. Sci. Comp. Prog. 67, 301–329 (2007)
7. Barold, S., Stroobandt, R., Sinnaeve, A.: Cardiac Pacemakers and Resynchronization Step by Step: An Illustrated Guide. Wiley-Blackwell (2010)
8. Börger, E.: The ASM Refinement Method. FACJ 15, 237–257 (2003)
9. Börger, E., Stärk, R.: Abstract State Machines. A Method for High Level System Design and Analysis. Springer (2003)
10. Boston Scientific: PACEMAKER System Specification (2007), http://www.cas.mcmaster.ca/sqrl/_SQRLDocuments/PACEMAKER.pdf
11. Ellenbogen, K., Wood, M.: Cardiac Pacing and ICDs. Wiley-Blackwell (2008), 5th ed.
12. Gomes, A., Oliveira, M.: Formal Specification of a Cardiac Pacing System. In: Proc. FM-09. LNCS, vol. 5850, pp. 692–707. Springer (2009)
13. Jones, C., O'Hearne, P., Woodcock, J.: Verified Software: A Grand Challenge. IEEE Computer 39, 93–95 (2006)
14. Karlsruhe Interactive Verifier: http://www.informatik.uni-augsburg.de/lehrstuehle/swt/se/kiv/
15. Keinert, M., Elmqvist, H., Strandberg, H.: Spectral Properties of Atrial and Ventricular Endocardial Signals. PACE 2, 11–19 (1979)
16. Macedo, H., Larsen, P., Fitzgerald, J.: Incremental Development of a Distributed Real-Time Model of a Cardiac Pacing System Using VDM. In: Proc. FM-08. LNCS, vol. 5014, pp. 181–197. Springer (2008)
17. Méry, D., Singh, N.: Functional Behavior of a Cardiac Pacing System. Tech. rep., LORIA, Université Henri Poincaré - Nancy I (2011), http://www.loria.fr/~singhnne/Home_files/downloads/ijdecs2010.pdf, Int. J. Discrete Event Control Systems
18. Retrenchment Homepage: http://www.cs.man.ac.uk/retrenchment
19. Schellhorn, G.: Verification of ASM Refinements Using Generalized Forward Simulation. JUCS 7, 952–979 (2001)
20. Schellhorn, G.: ASM Refinement and Generalizations of Forward Simulation in Data Refinement: A Comparison. Theor. Comp. Sci. 336, 403–435 (2005)
21. Schuchert, A., Aydin, A., Israel, C., Gaby, G., Paul, V.: Arial Pacing and Sensing Characteristics in Heart Failure Patients Undergoing Cardiac Resynchronization Therapy. Europace 7, 165–169 (2005)
22. Wilson, F., Macleod, A., Barker, P.: The Distribution of the Action Currents Produced by Heart Muscle and Other Excitable Tissues Immersed in Extensive Conducting Media. J. Gen. Physiol. 16, 423–456 (1933)
23. Wilson, F., Macleod, A., Barker, P.: The Distribution of the Currents of Action and of Injury Displayed by Heart Muscle and Other Excitable Tissues, University of Michigan Studies. Scientific Series, vol. 10. Ann Arbor: University of Michigan Press (1933), Reprinted in: Lepeschkin and Johnston (eds.), Selected Papers of Frank N. Wilson. Ann Arbor, J.W. Edwards (1954)
24. Woodcock, J.: First Steps in the The Verified Software Grand Challenge. IEEE Computer 39, 57–64 (2006)
25. Woodcock, J., Banach, R.: The Verification Grand Challenge. JUCS 13, 661–668 (2007)
26. Woodcock, J., Davies, J.: Using Z, Specification, Refinement and Proof. Prentice Hall (1996)