Correct-by-Construction Reinforcement Learning of Cardiac Pacemakers from Duration Calculus Requirements*

Kalyani Dole[†1], Ashutosh Gupta[†1], John Komp[†2], Shankaranarayanan Krishna[†1], Ashutosh Trivedi[†2]

[1] Indian Institute of Technology Bombay
[2] University of Colorado Boulder
{kalyanid,akg,krishnas}@cse.iitb.ac.in, {john.komp,ashutosh.trivedi}@colorado.edu

Abstract

As the complexity of artificial pacemakers continues to grow, the importance of capturing its functional correctness requirement formally cannot be overestimated. The pacemaker system specification document by *Boston Scientific* provides a widely accepted set of specifications for pacemakers. As these specifications are written in a natural language, they are not amenable to automated verification, synthesis, or reinforcement learning of pacemaker systems. This paper presents a formalization of these requirements for a dual-chamber pacemaker in *duration calculus* (DC), a highly expressive real-time specification language. The proposed formalization allows us to automatically translate pacemaker requirements into executable specifications as stopwatch automata, which can be used to enable simulation, monitoring, validation, verification and automatic synthesis of pacemaker systems. The cyclic nature of the pacemaker-heart closed-loop system results in DC requirements that compile to a decidable subclass of stopwatch automata. We present shield reinforcement learning (shield RL), a shield synthesis based reinforcement learning algorithm, by automatically constructing safety envelopes from DC specifications.

Introduction

The human heart is arguably the most important real-time safety-critical system. In an average life-span of 70 years, the human heart beats more than 2.5 billion times. Each heartbeat is a complex chemical reaction that starts at the *sinoatrial node* in the right atrium and sweeps down through the ventricles. The cells of the myocardium have unique properties that regulate the speed of this chemical reaction to synchronize each chamber's beat with the hemodynamics of the cardiac cycle (Kay and Shepard 2017). Any disruption in the cellular chain can cause the heart to beat improperly.

A common cardiac arrhythmia, known as heart block or AV block, is a condition where the electrical signal controlling the ventricular contractions are partially or completely blocked (Vijayaraman and Ellenbogen 2017). Depending upon the severity of this block, pacemaker therapy may be indicated. A pacemaker is an implantable electronic device that sends electrical impulses to the heart to regulate the heartbeat. The correctness of the pacemaker-heart closed-loop system is time-critical: it should not only provide supportive pacing when needed, it must ensure that the heart is not sent into unnaturally fast heart beats. Furthermore, there are secondary requirements to improve the quality of life such as detecting and supplying increased rates when the patient exercises (rate-adaptive pacing) and maximizing battery longevity (2.2% risk of infection; 0.4% mortality rate per pacemaker replacement (Polyzos, Konstantelias, and Falagas 2015)).

Over sixty years of research and development in designing artificial cardiac pacemakers has resulted in remarkable robustness, convenience, and acceptance of these devices. As the number of patients using pacemakers continues to rise (Greenspon et al. 2012), so are the expectations on the invisibility of its design resulting in ever-increasing demands on improved functionality, size shrinkage, power consumption, battery performance and adaptability to individual cardiac arrhythmias (Gomes 2020; Finkelmeier 1991). The medical device industry continues to meet these challenges by extending its design with more sensors (e.g., for rate augmentation, sleep apnea, and haemodynamic status) (Lau, Siu, and Tse 2017) resulting in ever-increasing complexity of the device software.

While manual design, supported by exhaustive model-based development, has served well for previous generations of devices, the need for adaptability without sacrificing the basic functional correctness is ever-present (Maisel et al. 2001). This paper proposes a *correct-by-construction* reinforcement learning (RL) paradigm to design cardiac pacemakers by integrating formal requirements in RL.

Example 1 (The need for Adaptive Pacemaker)**.** *Mobitz II second-degree AV block (Langendorf and Pick 1968) is a*

*This material is based upon work supported by the National Science Foundation (NSF) under grant No. CCF-2009022 and NSF CAREER award CCF-2146563.

†These authors contributed equally.

condition where the atrioventricular node does not transmit all atrial depolarizations to the ventricle and is manifested as a pattern of dropped beats. A 3:2 *block drops every third. While the standard pacing algorithms fill in these missing beats providing patient support, the timing of the needed ventricular paces are aimed towards satisfying the safety requirements and do not attempt to match the intrinsic heart rate. As the patients can feel this difference in heartbeats, their quality of life can be enhanced by adapting the time to provide for a missing beat to the patient intrinsic rhythm.*

The overarching objective of this industrial collaboration is to formalize the requirements in a rigorous, formal language capable of precisely capturing the requirements of a cardiac pacemaker. The key requirements on a pacemaker include maintaining a minimum heart rate, avoiding pacing during the ventricular repolarization, not increasing the heart rate excessively, and not disturbing the natural rhythm of the heart. The precise algorithms to implement these requirements require robust detection of various events as well as a careful accounting of a number of timers to provide the control logic. *Pacemaker System Specification* document (Laboratory 2007) by Boston Scientific provides a refined version of the basic specifications of the pacemaker design in a natural language. A key contribution of this paper is capturing these time-critical requirements for the dual-chamber pacemaker in a real-time formal logic.

Previous Efforts. Research into using pacemakers as a vehicle for applying formal methods to medical device design traces back to the *Pacemaker Grand Challenge* in 2007. The challenge was kicked off with the release of a generic dual chamber pacemaker specification (Laboratory 2007) from Boston Scientific detailing the functional correctness requirements on the system. Before release, Boston Scientific removed references to proprietary algorithms leaving only the high level intention of such features. *The Pacemaker Challenge: Developing Certifiable Medical Devices* was presented as a Schloss Dagstuhl seminar in 2014 (Méry, Schätz, and Wassyng 2014). In 2018, Bonfanti et al. (Bonfanti, Gargantini, and Mashkoor 2018) reviewed the usage of formal methods in medical devices. Table 4 of (Bonfanti, Gargantini, and Mashkoor 2018) provides a breakdown of 48 papers related to pacemaker research into categories of modeling, model verification and validation, software validation, and code generation. A number of efforts have focused on capturing these specifications using formal languages or extensions of those languages such as AADL (Larson 2014) and Z (Gomes and Oliveira 2009). There have been attempts to model a dual-chamber pacemaker with advanced features using timed automata (Jiang et al. 2012). Timed automata are useful in capturing key features of closed-loop systems and enable the use of tools like UPPAAL in verification. While timed automata based representations are amenable to formal analysis, the translation to such specifications is manual and in the process lose interpretability; moreover, timed automata are not expressive enough to capture several properties of interest. On the other hand, expressive specification languages limit automation.

Our Choice of Logic. We propose the use of *Duration Calculus* (DC) (Chaochen, Hansen, and Sestoft 1993; Chaochen, Hoare, and Ravn 1991) in expressing hard real-time constraints on pacemakers. DC is a real-time logic designed to express complex, time-critical properties of hybrid dynamical systems (Chaochen, Hoare, and Ravn 1991; Dole, Gupta, and Krishna 2020). The duration modalities $\int P \bowtie c$ as well as $\ell \bowtie c$ in DC allows one to capture the accumulated duration of time when some event P has been holding, as well the real time duration ℓ of interest. These modalities make DC very expressive. The expressiveness of DC, on the negative side, contributes to undecidability of key decision procedures such as emptiness and model checking against real-time models such as timed automata (Chaochen, Hansen, and Sestoft 1993); on the other hand, there has been work in exploring decidable sub classes of DC (Pandya 2002; Dole, Gupta, and Krishna 2020). There are also tools, such as DCVALID (Pandya 2001), to compile discrete DC specifications to automatic structures. To the best of our knowledge, most timed logics (other than DC) cannot model real time properties having durations. DC naturally has this modality. Circumventing the undecidability of DC by limiting the expressiveness which still is good enough to capture specifications of safety critical systems (such a the pacemaker) is a key challenge in developing correct-by-construction synthesis.

Correct-by-Construction Synthesis via RL. Correct-by-construction synthesis (Baier and Katoen 2008) is an approach to safety-critical system design that advocates the integration of the formal proof-of-correctness of the designed system by automatically refining formal specifications. It takes an adversarial view of the environment and employs tools from competitive game theory to design theoretically optimal, if over-cautious, systems. Reinforcement learning (Sutton and Barto 2018) (RL) paradigm offers an alternative view of the environment as a stochastic player with unknown strategy and proposes an adaptive sampling-based approach to converge towards the optimal policy. While RL adapts to the changes in the environment, it requires strong assumption on the nature of the environment (Markovian assumption) and lacks guarantees on the safety of the system during learning. Shielding (Alshiekh et al. 2018; Könighofer et al. 2020) is an approach to RL that combines guarantees from the correct-by-construction synthesis with adaptive nature of reinforcement learning. By focusing on cardiac pacemaker as our case-study, we develop shielding-based RL for a subclass of DC specifications.

Contributions. The contributions of this paper are summarized below.

1. We express DDD pacemaker requirements from (Laboratory 2007) in Duration Calculus.
2. We observe that due to the cyclic nature of the pacemaker-heart closed-loop system, the aforementioned requirements belong to bounded-time fragment of DC. We show that for bounded-time DC, the formulae can be compiled into a decidable class of (acyclic) stopwatch automata (Cassez and Larsen 2000), providing a set of *executable specification*. We show that the emptiness or

acyclic stopwatch automata is NP-COMPLETE.

3. We implemented these translations using an extension of DCVALID (dcvalid) that compiles DC specifications into stopwatch, timed, and finite-state automata.
4. We validate the behavior of the executable specifications (automata) obtained from the DC requirements for the pacemaker by showing their performance over two challenging scenarios.
5. We provide a proof-of-concept in using formal pacemaker requirements in DC to generate an RL-shield (Alshiekh et al. 2018) to restrict the behavior of the RL agent to enable safe learning. We sketch the design of an adaptive pacemaker via RL that adapts its behavior in accordance to the natural rhythm of the patient.

All of the relevant artifacts (tool and benchmarks) can be found at (Dole et al. 2022).

Preliminaries

Duration Calculus. Duration Calculus (Chaochen, Hansen, and Sestoft 1993) is a highly expressive and succinct logic, which can capture specifications involving durations. The chop modality ($\frown$) contributes to its succinctness and enables compositional specification, while the measurement constructs ($\ell \bowtie c$) and ($\int P \bowtie c$) enable duration measurements.

Definition 1 (DC: Syntax)**.** *Given a set Var of real time signals, we define the syntax of DC formulae as follows:*

$$\begin{aligned} P &::= \ \mathsf{false} \mid \mathsf{true} \mid x \in Var \mid P \wedge P \mid \neg P \\ D &::= \ \lceil P \rceil \mid \lceil P \rceil^{\bullet} \mid D \wedge D \mid \neg D \mid D^\frown D \mid M \\ M &::= \ \ell \bowtie c \mid \int P \bowtie c \mid \sum P \bowtie c \end{aligned}$$

where $\bowtie \in \{<, \leq, =, \geq, >\}$.

Definition 2 (DC: Semantics)**.** *Let σ be a timed trace $\langle (s_0, \tau_0), (s_1, \tau_1), \ldots, (s_n, \tau_n) \rangle$ where each s_i is a state (a set of propositional variables Var) and τ_i is its time stamp. For the timed trace σ and a propositional logic formula P over Var, we say $(\sigma, i) \models P$ iff $s_i \models P$. A timed trace σ satisfies a DC formula ψ in an interval $I = [b, e]$ (where $b \leq e$ and $b, e \in \mathbb{N}$ range over the indices of a timed trace), and we write $(\sigma, [b, e]) \models \psi$, if:*

- $(\sigma, [b, e]) \models \lceil P \rceil$ *iff* $b<e$, *and* $(\sigma, t) \models P$ *for all* $b<t<e$;
- $(\sigma, [b, e]) \models \lceil P \rceil^{\bullet}$ *iff* $b=e$ *and* $(\sigma, b) \models P$;
- $(\sigma, [b, e]) \models D_1 \wedge D_2$ *iff* $(\sigma, [b, e]) \models D_1, D_2$;
- $(\sigma, [b, e]) \models \neg D$ *iff* $(\sigma, [b, e]) \not\models D$;
- $(\sigma, [b, e]) \models D_1^\frown D_2$ *iff there is* $b \leq z \leq e$ *s.t.* $(\sigma, [b, z]) \models D_1$ *and* $(\sigma, [z, e]) \models D_2$;
- $(\sigma, [b, e]) \models \ell \bowtie c$ *iff* $(\tau_e - \tau_b) \bowtie c$ *holds;*
- $(\sigma, [b, e]) \models \int P \bowtie c$ *iff* $\sum \{\tau_{i+1} - \tau_i \mid (\sigma, i) \models P\} \bowtie c$;
- $(\sigma, [b, e]) \models \sum P \bowtie c$ *iff* $|\{i : (\sigma, i) \models P\}| \bowtie c$;

One can derive operators $\Rightarrow$ (conditional) and $\Leftrightarrow$ (biconditional) in the usual manner. Moreover, the temporal logic modalities eventually $\Diamond D \overset{\text{def}}{=} \mathsf{true}^\frown D^\frown \mathsf{true}$ and the globally $\Box D \overset{\text{def}}{=} \neg \Diamond \neg D$ can be derived from the basic syntax.

Stopwatch Automata. Alur and Dill (Alur and Dill 1994) generalized the theory of finite state automata to model time-constrained evolution of systems. The resulting formalism, known as timed automata, express time constraints by using a finite set of non-negative real-valued variables called *clocks* that work as timers in that they grow with uniform rate and can be reset to 0 to remember the time since some given event. These clocks can be used in *guard expressions* on the transitions using the following grammar:

$$\varphi := x \bowtie c \mid x - x' \bowtie c \mid \varphi \wedge \varphi \tag{1}$$

where $\bowtie \in \{\leq, <, =, >, \geq\}$, x and x' are clock variables and c is a natural number. Let $\mathcal{G}(X)$ be the set of guard expressions over the set of clocks X. The stopwatch automata (Cassez and Larsen 2000) generalize timed automata by allowing the clocks to be paused (it is customary to refer to pausable clocks as stopwatches); however, adding this expressiveness results in undecidability of the emptiness problem. See Figure 2 for a stopwatch automata having a clock y and a stopwatch variable x_p.

Definition 3 (Stopwatch Automata)**.** *A stopwatch automaton is a tuple $\mathcal{T} = (Q, q_0, \Sigma, X, \gamma, E, \delta)$ where: Q is the finite set of* locations*; $q_0 \in Q$ is the initial* location*; Σ is a finite alphabet of actions; X is the finite set of clock* variables*; $\gamma : Q \to 2^X$ is the set of paused clocks per location; $E : Q \times \Sigma \to \mathcal{G}(X)$ is the* action guard*, and $\delta : Q \times \Sigma \to \mathcal{D}(2^X \times Q)$ is the* transition function*.*

The DC specifications can compiled into an "executable specification" of stopwatch automata. We omit the details of this translation due to a lack of space.

Theorem 2. *For every DC formula ϕ, one can effectively construct a stopwatch automaton $\mathcal{A}_\phi$ that accept the same set of timed traces.*

On a negative side, the satisfiability of DC (Chaochen, Hansen, and Sestoft 1993) as well as its emptiness (Henzinger et al. 1995) of stopwatch automata is undecidable.

Bradycardia Pacing in DC

The goal of cardiac pacing is to replace a biological component of the cardiac cycle that has failed, is operating intermittently, or to mitigate an incorrect conduction pathway that is causing irregular heartbeats. We provide a brief overview of the hear conduction system (Kay and Shepard 2017).

A heart beat begins at the sinoatrial node (SA node), causing a depolarization of cells in atrium, which compresses the atrium and forces blood into the ventricle. This phase is depicted as P-wave in Figure 1. The atrioventricular node (AV node) then delays the passage of repolarization signal to ventricle. This prevents the premature ventricular contraction (PVC) and allows blood to fill the ventricle. Then, the depolarization continues down the ventricle, causing it to compress and push the blood through the body. On the EGM, this depolarization is seen as the QRS complex. Repolarization of the ventricular cells produces a larger signal in surface EGM and is called a T-wave.

Irregular heartbeats caused by a malfunctioning of SA or AV are known as bradycardia. A malfunction of the SA can

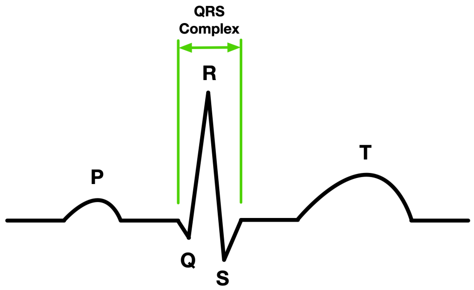

Figure 1: EGM: Cardiac Segments

lead to conditions where the heart beats slowly, irregularly, drops beats, beats rapidly, or pauses. When the AV node fails to pass the signal through the ventricle, it can result in heart block, resulting in dropped beats to complete blockage of all signals. Finally, myocardial infarction (heart attacks) can lead to the death of heart cells along the conduction path.

Modern pacemakers have mechanisms for monitoring heart intrinsic activity, transmitting generated pacing pulses and sensing the synchronization on intrinsic heart activity. A pacemaker that adds sensing and synchronization in the ventricle has the code *VVI* (for pace in the ventricle (V), sense intrinsic cardiac activity in the ventricle (V), and inhibit (I) a scheduled pace if a valid intrinsic ventricular event occurs). Dual chamber pacemakers (NASPE (Bernstein et al. 2002) with code *DDD*) pace in both the atrium and ventricle (D), sense in both chambers (D), and individual paces may be inhibited or triggered based on intrinsic activity (D). In standard pacing therapy, it is desired to pace only for the minimum required and rely on the patient's natural heart rhythm when present. This is both physiologically better for the patient and conserves the pacemaker battery for better longevity. In the absence of intrinsic activity, the pacemaker outputs a pulse to the relevant chamber at a specified rate.

At the core of pacemaker lies its ability to detect improper activity by comparing sensed intrinsic activity with the status of various period and interval timers. This helps it to decide when intrinsic events should be accepted, or to generate a pace in the future to maintain proper heart rhythm.

Due to the cyclic nature of the heart and pacemaker interactions (every atrial pace or sense restarts the timing obligations and each duration is bounded by an upper-rate interval), we observe that the pacemaker requirements fall into time-bounded fragment of DC. Motivated by the general principle of time bounded verification, (Ouaknine, Rabinovich, and Worrell 2009), and the nature of specification needed for the DDD pacemaker, we study duration calculus whose models are evaluated over bounded timed traces.

Bounded Semantics for Duration Calculus

In this section, we focus on a subclass of DC whose models are evaluated over bounded timed traces. That is, DC formulae are evaluated over words of the form $\sigma = (s_0, \tau_0)(s_1, \tau_1) \dots (s_n, \tau_n)$ for some fixed and bounded n and an interval $[b, e]$. Apart from this, the DC semantics is as before. We refer to this semantics *bounded semantics*.

A key step in developing correct-by-construction learning (and synthesis) for this subclass is its reduction to an executable specification (stopwatch automata). From this reduction, we derive the decidability of the satisfiability (reachability) and controller synthesis (safety game) problems.

From DC to Stopwatch Automata

For a DC formula D under the bounded semantics for a bound n, we construct a timed or stopwatch automaton $\mathcal{A}_D^{[0,n]}$ which accepts all timed traces of length n satisfying D. This is done inductively by building automata $\mathcal{A}_\varphi^{[i,j]}$ for $0 \leq i \leq j \leq n$ and subformulae φ of D. These automata accept timed traces of length $j - i$ satisfying φ. Depending on the subformula φ, $\mathcal{A}_\varphi^{[i,j]}$ is either a timed or a stopwatch automata. For instance if φ has the duration construct $\int P \bowtie c$, then we require stopwatches to measure accumulated durations, and otherwise, a timed automata suffices. In all cases, $\mathcal{A}_\varphi^{[i,j]}$ is an acyclic since it accepts behaviours of bounded length. The automaton construction is inductive, and we sketch key steps next.

Let Var be the set of propositional variables in D. For a Boolean combination P of variables in Var, let Var_P denote all subsets of Var which satisfy P. For example, if $Var = \{p, q, r\}$, and $P = \neg p \wedge q \Rightarrow r$, then $\{q, r\}, \{p\}, \{r\}, \{p, r\}, \emptyset \in Var_P$. We illustrate the construction of $\mathcal{A}_D^{[i,j]}$ for the cases when D is one of $\int P \bowtie c$ or $D_1 {}^\frown D_2$. The other cases are omitted for lack of space.

1. Let $D = \int P \bowtie c$. Here, $\mathcal{A}_D^{[i,j]}$ is a stopwatch automata. Let x_P be the stopwatch variable used. From each location, we have two transitions : one on Var_P and another, on $Var_{\neg P}$. If we have a transition decorated with Var_P, then the rate of x_P is set to 1 in the target location and the rate of x_P is set to 0 in the target otherwise. It is easy to see that x_P accumulates the real time duration of P being true. After $j - i$ transitions along a path, we take the last transition to the final location, and check for $x_P \bowtie c$. If $(x_P \bowtie c)$ does not hold, we do not reach the final location. The length to any accepting location is exactly $j - i$ and each location has a single incoming transition.
2. Let $D = D_1 {}^\frown D_2$. In this case $\mathcal{A}_D^{[i,j]}$ is given by $\bigcup_k [\mathcal{A}_{D_1}^{[i,k]} \cdot \mathcal{A}_{D_2}^{[k,j]}]$ for $i \leq k \leq j$. Here $\mathcal{A}_{D_1}^{[i,k]} \cdot \mathcal{A}_{D_2}^{[k,j]}$ fuses the final state of $\mathcal{A}_{D_1}^{[i,k]}$ with the initial location of $\mathcal{A}_{D_2}^{[k,j]}$, and the idea is to obtain the union of such concatenations for all $k \in [i, j]$. The concatenation of all behaviours of length $k - i$ satisfying D_1 over an interval I_1 followed by all possible behaviours of length $j - k$ satisfying D_2 over an interval I_2 gives all behaviours of length $j - i$ satisfying $D_1 {}^\frown D_2$ over an interval I obtained as the fusion of intervals I_1, I_2.

The constructed $\mathcal{A}_D^{[i,j]}$ is (i) acyclic, (ii) each location has at most one incoming transition, and (iii) the length of any path in $\mathcal{A}_D^{[i,j]}$ is $j - i$. Our goal is to obtain $\mathcal{A}_D^{[0,n]}$ for some bound n. This final automaton is obtained by taking the product of component automata, or fusing them as above, or in some cases, taking a complement. Complementation is

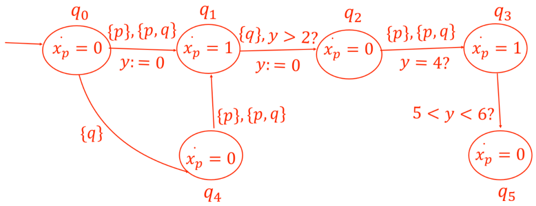

Figure 2: A stopwatch automaton over the alphabet $\Sigma = 2^{Var}$ where $Var = \{p, q\}$. We have a stopwatch variable x_p and a clock y. The rate of x_p is 0 at locations q_0, q_2, q_4, q_5. It can be seen that the accumulated duration of time for which p is true is some $t > 2$ on reaching q_2, q_3 via q_1. Thereafter, on reaching q_5, this duration is in $(t + 1, t + 2)$ for $t > 2$. Note that no duration is accumulated in location q_2 where 4 units of time are spent.

not a problem here since we deal only with acyclic automata and bounded length behaviours.

Emptiness Problem

In this section, we show that checking emptiness of an acyclic stopwatch automaton is decidable. The key idea is to encode the transitions and paths of such an automaton as QF_LRA formulae, which are Boolean combinations of propositional variables and linear constraints over real variables, and check for satisfiability.

Theorem 3. *Acyclic stopwatch (timed) automata as obtained above have a decidable reachability.*

Consider an acyclic stopwatch automata $\mathcal{A}$ as above, consisting of k locations $s_1, \ldots, s_k$. Each of the $k - 1$ edges in any path is decorated by some Var_P. Being acyclic, the whole automaton can be encoded as a disjunction of finitely many QF_LRA formulae, depending on the number of branches/paths the automaton has : each path leading to an accepting state is encoded by a QF_LRA formula. Then $\mathcal{A}_D^{[i,j]}$ has an accepting path iff we can satisfy the QF_LRA formula.

The linear constraints used in the QF_LRA formulae are $x - y \bowtie c$, $\sum_i x_i \bowtie c$ for real variables x_i, x, y and $c \in \mathbb{N}$. $x - y \bowtie c$ are useful in encoding the difference between real valued variables representing time elapses while $\sum_i x_i \bowtie c$ is useful in encoding the accumulated duration of a proposition P across several states (variable x_i encodes time elapse in a state i). The Boolean combinations of propositional variables allowed in our QF_LRA formulae are handy to encode the labels on the transitions.

Checking the satisfaction of the QF_LRA formula amounts to checking if we have an interpretation which satisfies all the constraints. Substituting a fresh propositional variable for each distinct constraint in this QF_LRA formula, we obtain a propositional logic formula ζ over an extended set of variables $Var' \supset Var$. If we obtain a satisfying assignment for the variables in Var', which satisfies this formula, then we also know that the original QF_LRA formula is satisfiable. Guessing such an assignment non-deterministically, we can verify if it forms a satisfying assignment or not. The size of ζ is linear in the size of our QF_LRA formula, hence checking if the guessed assignment satisfies ζ takes time linear in the size of the QF_LRA formula. This way we get a non-deterministic polynomial time procedure to check satisfiability of the QF_LRA formula, and hence the non emptiness of our automaton also. It is easy to see that we cannot have a polynomial time algorithm to check satisfiability of QF_LRA, since it subsumes propositional logic, whose satisfiability is known to be NP-complete.

Theorem 4. *DC under the bounded semantics has a decidable satisfiability.*

Starting from a formula in DC under the bounded semantics, we first construct the timed or stopwatch automaton corresponding to it as described above. This construction is such that any bounded timed trace satisfying the formula is accepted by the constructed automata. The decidability of emptiness checking in the constructed automata is shown using Theorem 5, and is NP-complete.

Safety Games

When DC specifications concern the choices of two agents (the controller and the environment), the set of signals can be paritioned into controllable and uncontrollable signals. The corresponding translation to stopwatch automata gives rise to two-player safety game (or minimax reachability games) on stopwatch automata. In order to compute the maximally-permissive set of actions of a pacemaker, we need to solve a safety game on the resulting stopwatch automata. Similar to Theorem 4, we get the decidability of the reachability games

Theorem 5. *Safety games on acyclic stopwatch (timed) automata can be solved effectively.*

This safety region can be computed using controllable-predecessor opertaor and defines a set of maximally permissive actions for the pacemaker. When the underlying system model is timed automata, the winning region can be computed using UPPAAL-Tiga (Behrmann et al. 2007), and for a stopwatch automata it can be computed using Phaver+ (Benerecetti, Faella, and Minopoli 2013).

Once the safety region for the DC requirement has been computed, it can be used to block unsafe actions of the RL agent. This is the crux of shielding based reinforcement learning (Alshiekh et al. 2018).

Experimental Results

We have implemented DC to stopwatch automata construction in an extension of DCVALID (dcvalid) with some optimizations and derived operators. DCVALID in turn uses MONA (Klarlund and Møller 2001) and provides a validity checker and visualizer for DC Formulae.

Validation of Pacemaker Requirements

We have written the specification of pacemaker in DC logic. We present the full list of requirements. We used DCVALID to compile the pacemaker specification into automata and validated their correctness using two requirements (see extended version at (Dole et al. 2022)). In order to simulate the

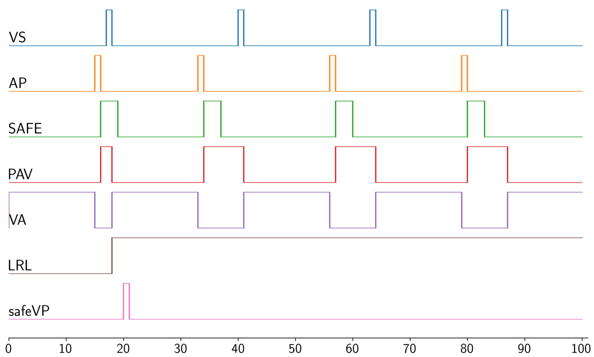

Figure 3: Ventricular Safety Pacing

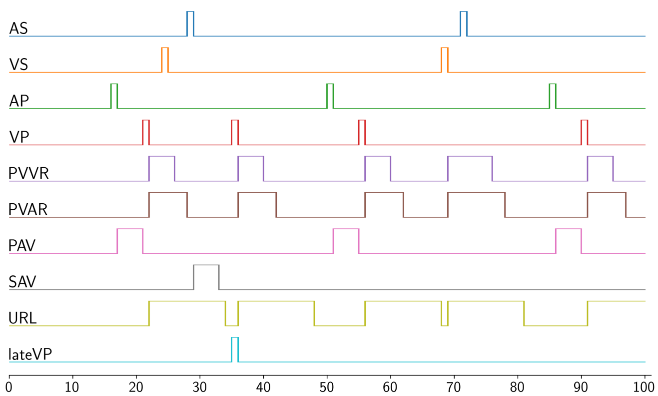

Figure 4: Upper Rate Holdoff

pacemaker, we compiled each requirement into an automaton, resulting in a large number of implicitly conjuncted automata. Since explicitly computing their product is neither practical nor necessary for the simulation, we implemented an on-the-fly simulation algorithm.

The simulation algorithm begins by determining a topology of the automata through ordering them with respect to their inter-dependency. The inputs to the system are provided by simulating the intrinsic atrial and ventricular heart beats, and each automaton is simulated in the order determined by the topology, which in turn generates the output events of atrial and ventricular pacing. Here are the results of our simulation on two scenarios found at (Dole et al. 2022).

Test Case 1 (Ventricular Safety Pacing). Atrial paced events are followed by a Ventricular Safety Pace (VSP) period (shown as signal $SAFE$ in Figure 3). Any VS detected while this signal is asserted will trigger a VP when the signal ends. As seen in the figure, $safeVP$ indeed occurs on the falling edge of $SAFE$ showing the VSP functionality.

Test Case 2 (Upper Rate Hold-off). The scenario depicted in Figure 9 in (Dole et al. 2022) is slightly complex: if VP is triggered at the precise time, the upper and lower rate requirements could be violated. The presented test case in Figure 4 shows satisfaction of this complicated situation.

Since the first ventricular sense in the example occurs within the refractory period, it can be ignored and thus it does not change any pacing timing. Meanwhile, the subsequent atrial sensed event, AS, occurs quite early. If the ventricular sensed event was not ignored, a new VA interval would have started causing the AS to fall into refractory interval PVARP. This would have ignored the AS setting the subsequent VP to occur at LRL. However, as seen in Figure 4, $PVAR$ does not get restarted after the refractory VS (the first pulse on the VS trace.) Additionally, the AS did cause assertion of the SAV signal showing the start of a new AV interval. This too would not have occurred if the refractory VS had been misclassified. Finally, because of the early arrival of the AS, the upper rate URL signal is still asserted and thus, the scheduled VP must be held off till the URL ends. Once URL completes, signal $lateVP$ is asserted showing correct hold-off.

Design of an Adaptive DDD Pacemaker

Consider the Mobitz II second-degree AV discussed in Example 1. Figure 5a shows a short timeline of 3:2 heart block pacing. The pacemaker does not look for the underlying intrinsic rate and will pace at the end of the lower rate interval. Since the patients can feel this difference in heartbeats, we design an adaptive pacemaker that finds an optimal time to provide for a missing beat matching the intrinsic rhythm.

For this case-study, we start from the DC specification of the DDD pacemaker with one exception. In a standard pacemaker, the timing of the ventricular pace is fixed in time, occurring at the termination of the AV interval. For this example, this requirement was relaxed, allowing the pace to occur any time in the AV interval. The modified pacemaker specification is available at (Dole et al. 2022). By embodying all the pacemaker requirements, the automaton guards against incorrect RL actions, rejecting those that would violate a non-permissive requirement. We designed an RL agent with the information of the safety region of the resulting specification, thus eliminating unsafe choices from the RL agent. The RL agent was rewarded based on how closely it matched the intrinsic rhythm of the heart model. To generalize the learning to different environments, we extended the state information with the history of the last 10 AV intervals.

For this case-study, we created a simple heart model with a 3:2 heart block. In addition, the heart model was permitted to randomly accelerate and decelerate with a 10% chance in either direction. Figure 6a summarizes the timing of the intrinsic and paced events by a standard pacemaker over a run of 5000 time points. In this figure, regardless of the varying intrinsic ventricular rate represented by the dark grey bars, all standard pacemaker provided beats (VP) occur at the lower rate time period 15 which is represented by the single light grey bar. Figure 6b summarizes the timing of the intrinsic and paced events by a permissive pacemaker. The black bar represents the total atrial contractions. The dark grey and light grey bars are the total ventricular intrinsic beats (VS) and the pacemaker induced (VP) respectively over 5000 time points. The pacemaker lower rate was set to pace to 15 in lieu of any intrinsic ventricular activity. The RL agent learned that the optimal time to pace was two time points after the missing intrinsic beat and on the chart each

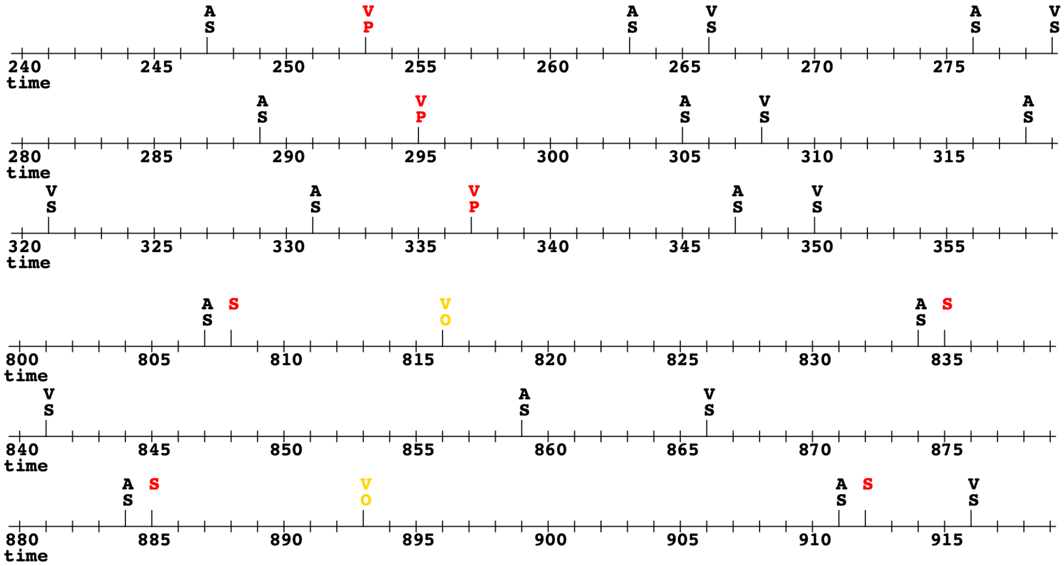

Figure 5: Timeline segments of cardiac activity during test runs (a) Non-permissive pacemaker synthesized via RL pacing heart with 3:2 heart block; (b) An adaptive pacemaker learned via RL, shield from safety specifications, heart with 3:2 heart block.

dark grey bar is paired with a light grey bar approximately half its size and to the right. It is desired to always allow the heart to intrinsically beat if possible, thus, two time periods of delay was enforced by the reward generator to assure the pacemaker does not begin to pace over the heart. The pace totals are not exactly half as the agent needs to recognize the intrinsic rate change. The optimal time for an intrinsic beat at 6 here is at 9 because pacing at 8 would violate the pacemaker's URL requirement so it had to be held off until the Upper Rate Limit had been satisfied.

Figure 5 shows part of a timeline of the experiment run. An example of the optimal timing can be seen by looking at a sequence starting at time 834 with an atrial sense. The ventricle intrinsically beats at 841. This pattern repeats at times 859 and 866. The next atrial event at 884 does not have an intrinsic beat at 891. This is the dropped beat. The RL agent provides support at 893 or two increments later as expected. This timeline also demonstrate the utility of the safety shield in action. In many cases, atrial senses are followed immediately by the agent attempting to pace. This time is the blanking period and is not permitted by the pacemaker requirements. These actions, denoted by an 'S' symbol, were prevented from occurring by the shield.

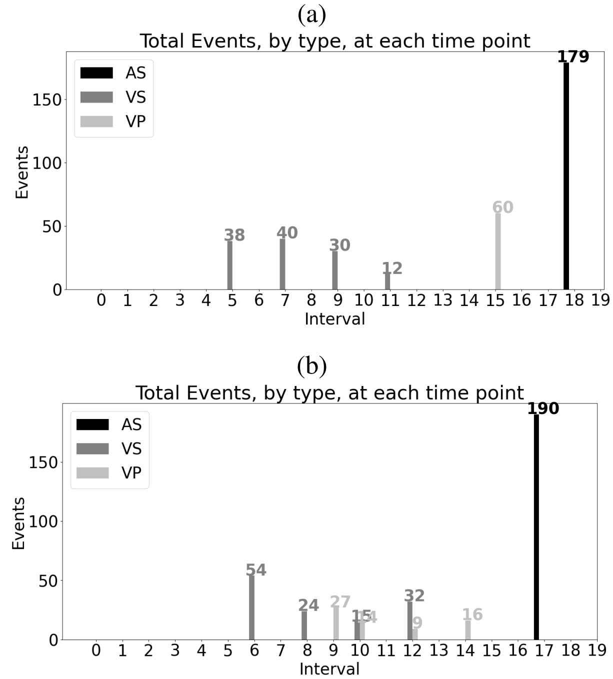

Figure 6: Timing with a) Standard Pacemaker and b) Permissive Pacemaker; 3:2 heart block

Discussions

We looked at the generalizing nature of RL to create new and unique features in medical devices and other real-time control systems. We presented an improvement in patient comfort by regulating pacing to match the intrinsic heart rate of heart block patients. A major concern while deploying RL in safety critical systems is that allowing unfettered exploration to find optimal paths may exceed safety parameters. Shielding the exploration by limiting the RL's action choice to remain in safety zones allows the RL agent to safely train. Once trained, the agent can be deployed with no further learning while still constrained by the shield to prevent any possible future erroneous actions. When constrained to specific problems, RL agents trained using locally permissive requirements guarded from exceeding safety parameters can provide a way to add new features to safety critical systems.

References

Alshiekh, M.; Bloem, R.; Ehlers, R.; Könighofer, B.; Niekum, S.; and Topcu, U. 2018. Safe reinforcement learning via shielding. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Alur, R.; and Dill, D. L. 1994. A Theory of Timed Automata. *Theoretical Computer Science*, 126(2): 183–235.

Baier, C.; and Katoen, J. 2008. *Principles of model checking*. MIT Press.

Behrmann, G.; Cougnard, A.; David, A.; Fleury, E.; Larsen, K. G.; and Lime, D. 2007. UPPAAL-Tiga: Time for Playing Games! In *Proceedings of the 19th International Conference on Computer Aided Verification*, 121–125.

Benerecetti, M.; Faella, M.; and Minopoli, S. 2013. Automatic synthesis of switching controllers for linear hybrid systems: Safety control. *Theoretical Computer Science*, 493: 116–138.

Bernstein, A. D.; Daubert, J.-C.; Fletcher, R. D.; Hayes, D. L.; Lüderitz, B.; Reynolds, D. W.; Schoenfeld, M. H.; and Sutton, R. 2002. The revised NASPE/BPEG generic code for antibradycardia, adaptive-rate, and multisite pacing. *Pacing and clinical electrophysiology*, 25(2): 260–264.

Bonfanti, S.; Gargantini, A.; and Mashkoor, A. 2018. A systematic literature review of the use of formal methods in medical software systems. *Journal of Software: Evolution and Process*, 30(5): e1943.

Cassez, F.; and Larsen, K. 2000. The Impressive Power of Stopwatches. In Palamidessi, C., ed., *CONCUR 2000 — Concurrency Theory*, 138–152. Berlin, Heidelberg: Springer Berlin Heidelberg.

Chaochen, Z.; Hansen, M. R.; and Sestoft, P. 1993. Decidability and Undecidability Results for Duration Calculus. In *STACS 93, 10th Annual Symposium on Theoretical Aspects of Computer Science, Würzburg, Germany, February 25-27, 1993, Proceedings*, 58–68.

Chaochen, Z.; Hoare, C. A. R.; and Ravn, A. P. 1991. A Calculus of Durations. *Inf. Process. Lett.*, 40(5): 269–276.

dcvalid. 2021. DCVALID–A tool for modelchecking Duration Calculus Formulae. accessed: 05/28/2021.

Dole, K.; Gupta, A.; Komp, J.; Krishna, S.; and Trivedi, A. 2022. Technical Appendix. https://github.com/kalyanidole/DDD-Pacemaker-Synthesis. Accessed: 2022-12-01.

Dole, K.; Gupta, A.; and Krishna, S. N. 2020. Robust Controller Synthesis for Duration Calculus. In *International Symposium on Automated Technology for Verification and Analysis*, 429–446. Springer.

Finkelmeier, N. E. 1991. Pacemaker technology: an overview. *AACN Advanced Critical Care*, 2(1): 99–106.

Gomes, A. O.; and Oliveira, M. V. M. 2009. Formal Specification of a Cardiac Pacing System. In Cavalcanti, A.; and Dams, D. R., eds., *FM 2009: Formal Methods*, 692–707. Berlin, Heidelberg: Springer Berlin Heidelberg.

Gomes, J. A. 2020. The Artificial Pacemaker: A Historical Overview. In *Heart Rhythm Disorders*, 449–460. Springer.

Greenspon, A. J.; Patel, J. D.; Lau, E.; Ochoa, J. A.; Frisch, D. R.; Ho, R. T.; Pavri, B. B.; and Kurtz, S. M. 2012. Trends in permanent pacemaker implantation in the United States from 1993 to 2009: increasing complexity of patients and procedures. *Journal of the American College of Cardiology*, 60(16): 1540–1545.

Henzinger, T. A.; Kopke, P. W.; Puri, A.; and Varaiya, P. 1995. What's decidable about hybrid automata? In *Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing, 29 May-1 June 1995, Las Vegas, Nevada, USA*, 373–382.

Jiang, Z.; Pajic, M.; Moarref, S.; Alur, R.; and Mangharam, R. 2012. Modeling and verification of a dual chamber implantable pacemaker. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, 188–203. Springer.

Kay, G. N.; and Shepard, R. B. 2017. Stimulation and excitation of cardiac tissues. In *Clinical Cardiac Pacing, Defibrillation and Resynchronization Therapy*, 61–113. Elsevier.

Klarlund, N.; and Møller, A. 2001. *MONA Version 1.4 User Manual*. BRICS, Department of Computer Science, University of Aarhus. Notes Series NS-01-1. Available from `http://www.brics.dk/mona/`.

Könighofer, B.; Lorber, F.; Jansen, N.; and Bloem, R. 2020. Shield synthesis for reinforcement learning. In *International Symposium on Leveraging Applications of Formal Methods*, 290–306. Springer.

Laboratory, S. Q. R. 2007. Pacemaker System Specification. ehttp://sqrl.mcmaster.ca/pacemaker.htm. Accessed: 2022-08-01.

Langendorf, R.; and Pick, A. 1968. Atrioventricular block, type II (Mobitz)—its nature and clinical significance. *Circulation*, 38(5): 819–821.

Larson, B. R. 2014. Formal semantics for the pacemaker system specification. *ACM SIGAda Ada Letters*, 34(3): 47–60.

Lau, C.-P.; Siu, C.-W.; and Tse, H.-F. 2017. Sensors for Implantable Cardiac Pacing Devices. *Clinical Cardiac Pacing, Defibrillation and Resynchronization Therapy*, 281–312.

Maisel, W. H.; Sweeney, M. O.; Stevenson, W. G.; Ellison, K. E.; and Epstein, L. M. 2001. Recalls and safety alerts involving pacemakers and implantable cardioverter-defibrillator generators. *Jama*, 286(7): 793–799.

Méry, D.; Schätz, B.; and Wassyng, A. 2014. The pacemaker challenge: developing certifiable medical devices (dagstuhl seminar 14062). In *Dagstuhl Reports*, volume 4.2. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Ouaknine, J.; Rabinovich, A.; and Worrell, J. 2009. Time-Bounded Verification. In *CONCUR 2009 - Concurrency Theory, 20th International Conference, CONCUR 2009, Bologna, Italy, September 1-4, 2009. Proceedings*, 496–510.

Pandya, P. K. 2001. Specifying and Deciding Quantified Discrete-time Duration Calculus Formulae using DCVALID: An Automata Theoretic Approach. In *Proceedings of RTTOOLS*.

Pandya, P. K. 2002. Interval Duration Logic: Expressiveness and Decidability. *Electron. Notes Theor. Comput. Sci.*, 65(6): 254–272.

Polyzos, K. A.; Konstantelias, A. A.; and Falagas, M. E. 2015. Risk factors for cardiac implantable electronic device infection: a systematic review and meta-analysis. *Ep Europace*, 17(5): 767–777.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

Vijayaraman, P.; and Ellenbogen, K. A. 2017. Atrioventricular Conduction System Disease. In *Clinical Cardiac Pacing, Defibrillation and Resynchronization Therapy*, 399–453. Elsevier.