HW2: Model Checking Pacemaker Software

In this lab you will implement a model of a DDD pacemaker in UPPAAL, and use heart models to model check the pacemaker model against physiological requirements. Then you will use principles of risk management to analyze model checking results. You will either improve your design to further reduce risk, or justify the acceptability of remaining risk. The results from this homework will be part of your final report.

1. Understand the heart models

In Lecture 5, we discussed how to abstract heart models so that they can cover electrical behaviors observable to a pacemaker in a large variety of heart conditions. We have also covered how to use an abstraction tree of heart models to balance coverage and physiological context during closed-loop model checking of pacemaker software. In this section, you are given a simplified abstraction tree of heart models in UPPAAL (Fig. 1). By answering the following questions, you will develop a better understanding of behavior coverage using non-determinism.

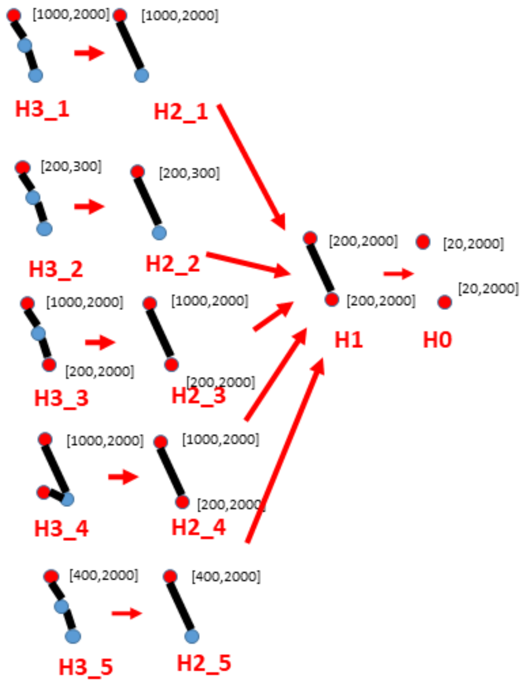


Fig. 1: Simplified abstraction tree of heart models. The numbers are the Rest period of the atria and the ventricles.

1.1 Simulation of refined heart models

There are 3 different heart model abstraction levels, and 5 heart models at abstraction level 3, which are named as H_3_n. In the first question, you will simulate these heart models in UPPAAL.

From their parameter settings and/or their exhibited behaviors, write down the name of the heart condition each of the model represents.

First, open “HW2.xml” in UPPAAL. In the “system declaration”, you will see the heart models starting with “system” declaration under the “HW2_1” section. There can only be one active system at the same time, so in order to simulate a heart model, delete the “//” in front of the respective “system” statement and comment out other heart models by adding “//” in front of their “system” declarations. Write down the name of the heart conditions corresponding to H_3_n in another Word file names “HW2_YourName.docx”.

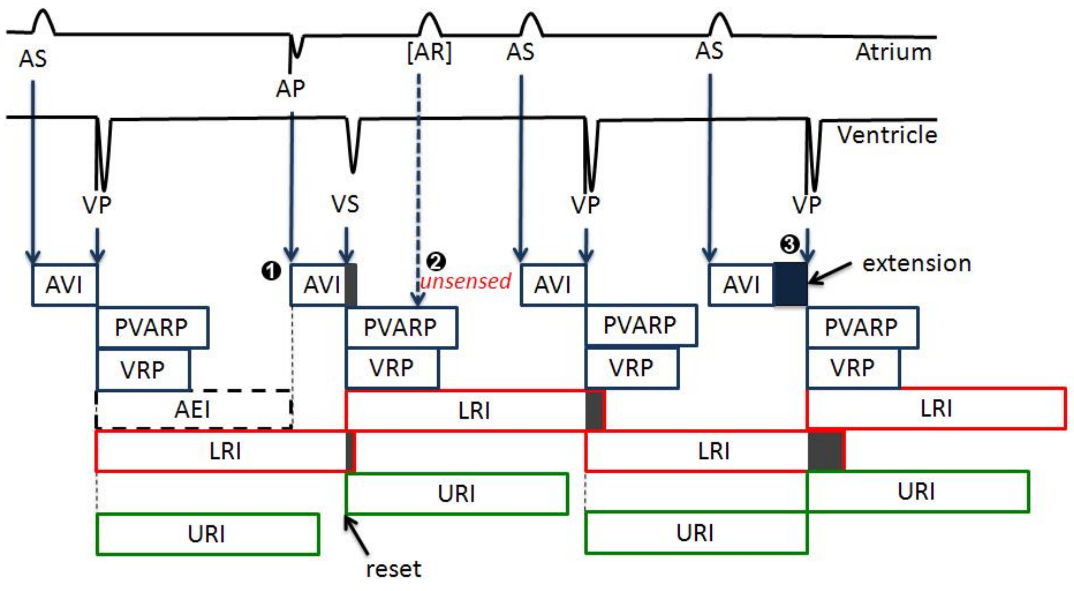
1.2 Symbolic simulation of more abstract models

In this question, we will see that the more abstract heart model (H_1) covers the observable behaviors of the refined heart models (H_3_n). By manually select “enabled transitions” during simulation of H_1, you will simulate the model 5 times and you should be able to reproduce execution traces during the 5 heart conditions you saw in the last question. Save the execution traces.

First, open “HW2.xml”. In the “system declaration”, you will see heart model H_1 under comment “//H_1”. Make the system active by uncommenting the system and save the current model as “HW2_1_2_YourName.xml”. Then don’t modify any parameters in this file so that generated traces can be properly loaded. Go to the simulation panel. Choose the enabled transitions from the upper left panel and click “next” to progress the simulation. At the end of a simulation, save the trace with name “HW2_1_2_n_YourName.xtr” by clicking the “save” button on bottom left, in which n ranges from 1-5. To start a new simulation, click “reset” and choose an enabled transition again. The number n corresponds to the 5 heart conditions in Question 1.1. The simulation should have similar observable behaviors (in terms of Ain and Vin patterns, and if possible, timing) as you saw in Question 1.1.

2. Pacemaker modeling

In Lecture 4 we introduced the basic 5 timing cycles of a DDD pacemaker as shown below:



Open “HW2_1_1_YourName.xml” and save it as “HW2_2_YourName.xml”. Use the information in the lecture slides and reading material, implement a DDD pacemaker model in UPPAAL. The pacemaker should take Ain and Vin from the heart as input events, and outputs AP and VP events. You can add internal events like (AS, VS) as necessary.

3. Model Checking the pacemaker model

In Lecture 5 and 6 we have identified several physiological requirements that a pacemaker should satisfy. In this section you will check your pacemaker model against these properties and identify potential risks. The physiological properties are as follows:

3.1 **No deadlock:** We need to ensure that the pacemaker can always progress instead of stuck in some states. The TCTL query for this property is A[] not deadlock.

3.2 **Lower Rate Limit:** The pacemaker should fulfil its intended use, which is maintaining a minimum ventricular rate. So the maximum interval between two consecutive ventricular events (Vin, VP) should be less or equal to TLRI.

3.3 **Upper Rate Limit:** The pacemaker should not pace to increase the heart rate too high. So the minimum interval between a ventricular event (VS,VP) and a ventricular pacing (VP) should be no less than TURI.

***Note:** Satisfying these 3 properties alone will not guarantee the correctness of you design. You can create additional properties to check whether certain functions of a pacemaker is fulfilled.

Pair your pacemaker model with heart model H_0, which covers all possible observable behaviors of the heart, as well as corresponding monitors. Use “Open Queries” in the File tab to load TCTL queries and check the first 3 properties. Your pacemaker design should be able to satisfy all 3 properties above. If not, use the counter-examples returned by the model checker to identify the problem and fix the bugs. Save your pacemaker design as “HW2_3_YourName.xml”.

In Lecture 5 and 6 we discussed Atrial Tachycardia Response (ATR) and Endless-loop Tachycardia (ELT), which are two cases in which the DDD pacemaker inappropriately increases the heart rate. In the following questions, we will use model checking to identify these two scenarios.

3.4 **Persistent fast ventricular events:** This property requires that the ventricular rate should not be faster or equal to URL for more than 30 beats. The monitor and TCTL property for this requirement are given to you.

3.4.1 First check your pacemaker model against this property with heart model H_0. The property should not satisfy and a counter-example will be returned. Analyze the counter-example. What heart condition does it represent and why?

3.4.2 Refine the heart models along the abstraction tree, until the returned counter-example can be identified as ELT and ATR. Save the models and counter-example traces for both ATR and ELT cases. i.e. “HW 2_3_4_ATR_YourName.xml” and “HW2_3_4_ATR_YourName.xtr”

3.4.3 Can you modify parameters in H_1 to identify ELT and ATR from counter-examples? If yes, write down the parameters for the heart model. If not, explain why.

Submission

In the homework submission, submit a .zip file with name “HW2_YourName.zip”. The .zip file should include:

- “HW2_YourName.docx”, which has answers for Question 1.1, 3.4.1, 3.4.3
- “HW2_1_2_YourName.xml”, which will be used to load traces for Question 2.1.2
- 5 traces with name “HW2_1_2_n_YourName.xtr”, which should be able to load properly in model “HW2_1_2_YourName.xml”.
- “HW2_3_YourName.xml”, which includes your pacemaker design.
- “HW 2_3_4_ATR_YourName.xml” and “HW 2_3_4_ELT_YourName.xml”, which will be used to load traces for Question 3.4.2.
- 2 traces with name “HW2_3_4_ATR_YourName.xtr” and “HW2_3_4_ELT_YourName.xtr”, which should be able to load properly in model “HW 2_3_4_ATR_YourName.xml” and “HW 2_3_4_ELT_YourName.xml”, respectively.

Deadline

The deadline for this homework is Oct 18th at the end of day. Please start early and ask questions during office hours.

Extra Credit

Students who like challenges can try to implement the mode-switch algorithm and ELT termination algorithm, and verify 1) whether their intended use has been fulfilled, and 2) whether new risks have been introduced. The algorithm description is as follows.

Warning: if you choose to implement these two algorithms, you will have to use the resulting pacemaker model as your design throughout this class. This means that for other activities like risk management, testing and code generation, you will have to take into account these algorithms as well, which requires additional effort. But the reward will be high if you do it right, so think about it.

Mode-Switch Algorithm

The algorithm first measures the interval between atrial events (AS, AR). The interval is considered as **fast** if it is above a threshold (**Trigger Rate**) and **slow** otherwise. A counter increments for fast events and decrements for slow events. After the counter value reaches the **Entry Count**, the algorithm will start a **Duration** which is a time interval used to confirm the detection of SVT. In the **Duration**, the counter keeps counting. If the counter value is still positive after the **Duration**, the pacemaker will switch to the VDI mode (Fallback mode). In the VDI mode, the pacemaker only senses and paces the ventricle. At any time if the counter reaches zero, the **Duration** will terminate and the pacemaker is switched back to DDD mode. In our mode-switch algorithm, we use nominal parameter values from the clinical setting. We define **trigger rate** at 170 bpm (350 ms), **entry count** at 8, **duration** for 8 ventricular events and fallback mode as VDI.

Endless-loop Tachycardia termination algorithm

The ELT persists without intervention and the patient’s heart is forced to beat at a fast rate approaching the Upper Rate Limit. Thus, device manufacturers require a way to identify ELT and terminate it despite of the limited information the pacemaker can get. The ELT detection algorithm by Boston Scientific utilizes these three features:

– Ventricular rate at Upper Rate Limit

– VP-AS pattern

– Fixed V–A conduction delay

The pacemaker first monitors VP-AS pattern with ventricular rate at upper rate limit. Then it compares the VP-AS interval with previous intervals. ELT is confirmed if the difference between the current VP-AS interval and the first VPAS interval are within ±32ms for 8 consecutive times. Then the pacemaker increases the PVARP period to 500 ms once so that the next AS will be blocked and will not trigger a VP. ELT will then be terminated.