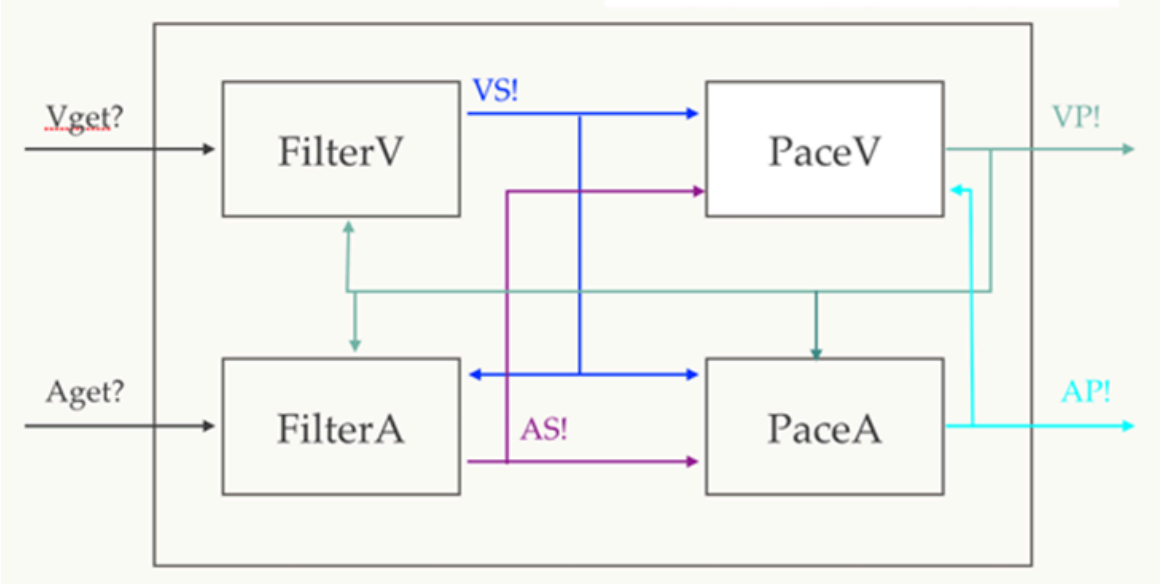
Pacemaker Report

By ZHANG Boyuan - CPS M1

0. Understanding the request from TD3 and TD 4



0.1 TD3: Model Design Based on Recommended Structure

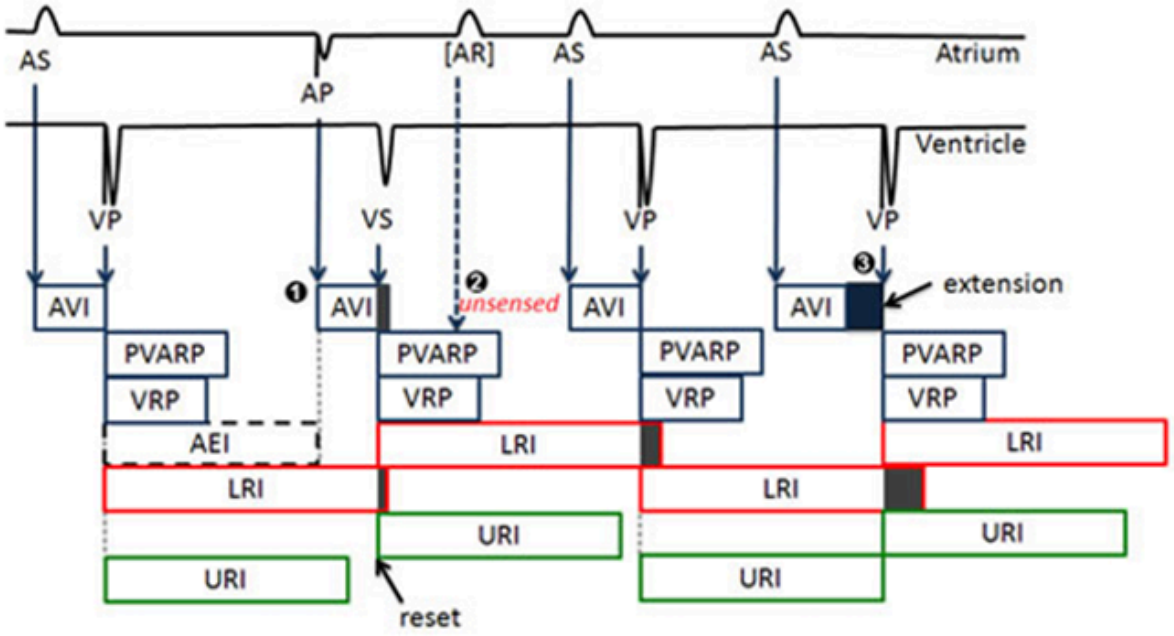
The task for TD3 involves designing a pacemaker model following the teacher's recommended structure. The model comprises two primary components:

- **Heart:** Represented by two automata:
  - **HeartVentricle (HV)**
  - **HeartAtrium (HA)**
- **Pacemaker:** Represented by four automata:
  - **FilterV (FV)**
  - **FilterA (FA)**
  - **PaceA (PA)**
  - **PaceV (PV)**

Due to the complexity introduced by multiple clocks within the **PaceV** automaton, which can lead to confusion and potential errors, the decision was made to split **PaceV** into two separate automata:

- **PaceV (PV):** Handles ventricular pacing logic.
- **PaceVclk (PVclk):** Manages the timing and synchronization related to ventricular pacing.

This division simplifies the model by isolating the timing mechanisms, thereby enhancing readability and maintainability.



0.2 TD4: Verification and Observer Design

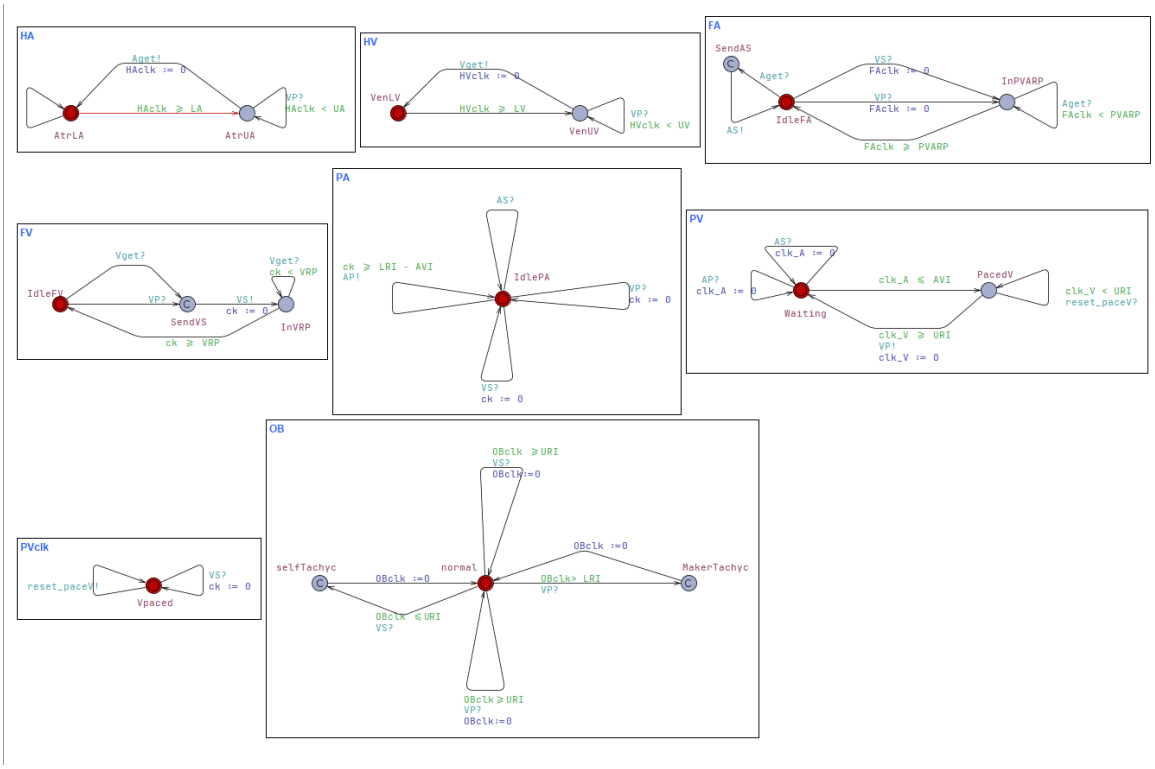
TD4 focuses on the verification of the pacemaker model to ensure its correctness and reliability. The approach comprises two main steps:

1. **Verification of Common Properties:**
    - **Objective:** Validate fundamental properties to guarantee the pacemaker's correct operation.
    - **Methods:** Utilize UPPAAL queries to check for absence of deadlocks, reachability of critical states, and adherence to timing constraints.
2. **Design and Integration of Observer Automaton:**
    - **Objective:** Provide external validation of the pacemaker's behavior by monitoring for abnormal conditions.
    - **Design:** Develop an **Observer** automaton that detects tachycardia arising from two sources:
        - **Patient-Induced Tachycardia:** Abnormal heart rhythms originating from the patient's heart.
        - **Pacemaker-Induced Tachycardia:** Excessively rapid pacing signals generated by the pacemaker.

The Observer ensures that the system maintains safe operating conditions by promptly identifying and responding to potential tachycardia events.

1. Choices on All 7 Automata (Full Pacemaker with Observer)

In this project , I designed a pacemaker model consisting of seven automata, including an observer. Below, we explain the state choices for each automaton, the role of time, and the design rationale based on the provided UPPAAL model.



1.1 HeartVentricle (HV)

State Choices:

- **VenLV:** Represents the ventricle's lower bound interval.
- **VenUV:** Represents the ventricle's upper bound interval.

Role of Time:

- A clock `HVclk` is used to simulate the natural beating interval of the ventricle, with timing constraints between `LV` (600) and `UV` (1000).

Design Explanation:

- The ventricle contracts naturally within the time interval `[LV, UV]`.
- When `HVclk` reaches `LV`, the automaton emits a `Vget!` signal, indicating a ventricular event.
- The clock `HVclk` is reset upon sending `Vget!` or upon receiving a pacing signal `VP?`, ensuring accurate timing between ventricular events.

1.2 HeartAtrium (HA)

State Choices:

- **AtrLA:** Represents the atrium's lower bound interval.
- **AtrUA:** Represents the atrium's upper bound interval.

Role of Time:

- A clock `HAclk` is used to simulate the natural beating interval of the atrium, with timing constraints between `LA` (600) and `UA` (1000).

Design Explanation:

- Similar to the ventricle, the atrium contracts naturally within the time interval `[LA, UA]`.
- When `HAclk` reaches `LA`, the automaton emits an `Aget!` signal, indicating an atrial event.
- The clock `HAclk` is reset upon sending `Aget!` or upon receiving a pacing signal `AP?`, ensuring accurate timing between atrial events.

1.3 FilterV (FV)

State Choices:

- **IdleFV:** Waiting for ventricular signals.
- **SendVS (Committed):** Immediately sends a `VS!` signal upon receiving a valid ventricular event.
- **InVRP:** Ventricular Refractory Period, during which additional `Vget?` signals are ignored.

Role of Time:

- A clock `FVclk` tracks the duration of the Ventricular Refractory Period (`VRP`).

Design Explanation:

- Upon receiving a ventricular event (`Vget?` or `VP?`), the automaton transitions to `SendVS`, sends `VS!`, resets `FVclk`, and enters `InVRP`.
- In the `InVRP` state, any additional `Vget?` signals are ignored until `FVclk` exceeds `VRP`.
- After the refractory period, the automaton returns to `IdleFV`, ensuring that noise and rapid unintended ventricular events are filtered out.

1.4 FilterA (FA)

State Choices:

- **IdleFA:** Waiting for atrial signals.
- **SendAS (Committed):** Immediately sends an `AS!` signal upon receiving a valid atrial event.
- **InPVARP:** Post Ventricular Atrial Refractory Period, during which additional `Aget?` signals are ignored.

Role of Time:

- A clock `FAclk` tracks the duration of the Post Ventricular Atrial Refractory Period (`PVARP`).

Design Explanation:

- Upon receiving a ventricular event (`VP?` or `VS?`), the automaton resets `FAclk` and enters `InPVARP` to prevent false atrial sensing due to retrograde conduction.
- In the `InPVARP` state, any additional `Aget?` signals are ignored until `FAclk` exceeds `PVARP`.
- In the `IdleFA` state, upon receiving `Aget?`, the automaton transitions to `SendAS`, sends `AS!`, and returns to `IdleFA`.
- This design ensures proper filtering of atrial signals and prevents erroneous pacing due to noise or premature atrial events.

1.5 PaceA (PA)

State Choices:

- **IdlePA:** Waiting for conditions to pace the atrium.

Role of Time:

- A clock `PAclk` measures the time since the last atrial or ventricular event.

Design Explanation:

- The automaton ensures atrial pacing occurs if no atrial event is sensed within `LRI - AVI` time units, maintaining the Lower Rate Interval (`LRI`).
- The guard condition `PAclk >= LRI - AVI` triggers atrial pacing by emitting an `AP!` signal.
- The clock `PAclk` is reset upon receiving `AS?`, `VP?`, or `VS?`, ensuring accurate timing based on cardiac activity.
- By resetting `PAclk` on ventricular events, the automaton accounts for situations where ventricular events indicate cardiac activity, thus preventing unnecessary atrial pacing.

1.6 PaceV (PV)

State Choices:

- **Waiting:** Awaiting conditions to pace the ventricle.
- **Ready:** Prepares for ventricular pacing, ensuring timing constraints are met.
- **PacedV (Committed):** Represents the state where ventricular pacing occurs.

Role of Time:

- Two clocks are used:
  - `clk_A`: Measures time since the last atrial event (`AS?` or `AP?`).
  - `clk_V`: Measures time since the last ventricular event (`VS?` or `VP?`).

Design Explanation:

- Upon receiving an atrial event (`AS?` or `AP?`), the automaton resets `clk_A`.
- From `Waiting`, the automaton transitions to `Ready` when `clk_A <= AVI`, indicating the pacing window is open.
- In the `Ready` state, the automaton waits until `clk_V >= URI` before pacing the ventricle, ensuring the heart does not beat too rapidly.
- The guard `clk_V >= URI` with synchronization `VP?` ensures pacing occurs only when the `URI` constraint is met.
- When conditions are satisfied, the automaton transitions to `PacedV`, sends `VP!`, resets `clk_V`, and returns to `Waiting`.
- This design enforces both the `AVI` and `URI` constraints, ensuring safe and effective ventricular pacing.

1.7 PaceVclk (PVclk)

State Choices:

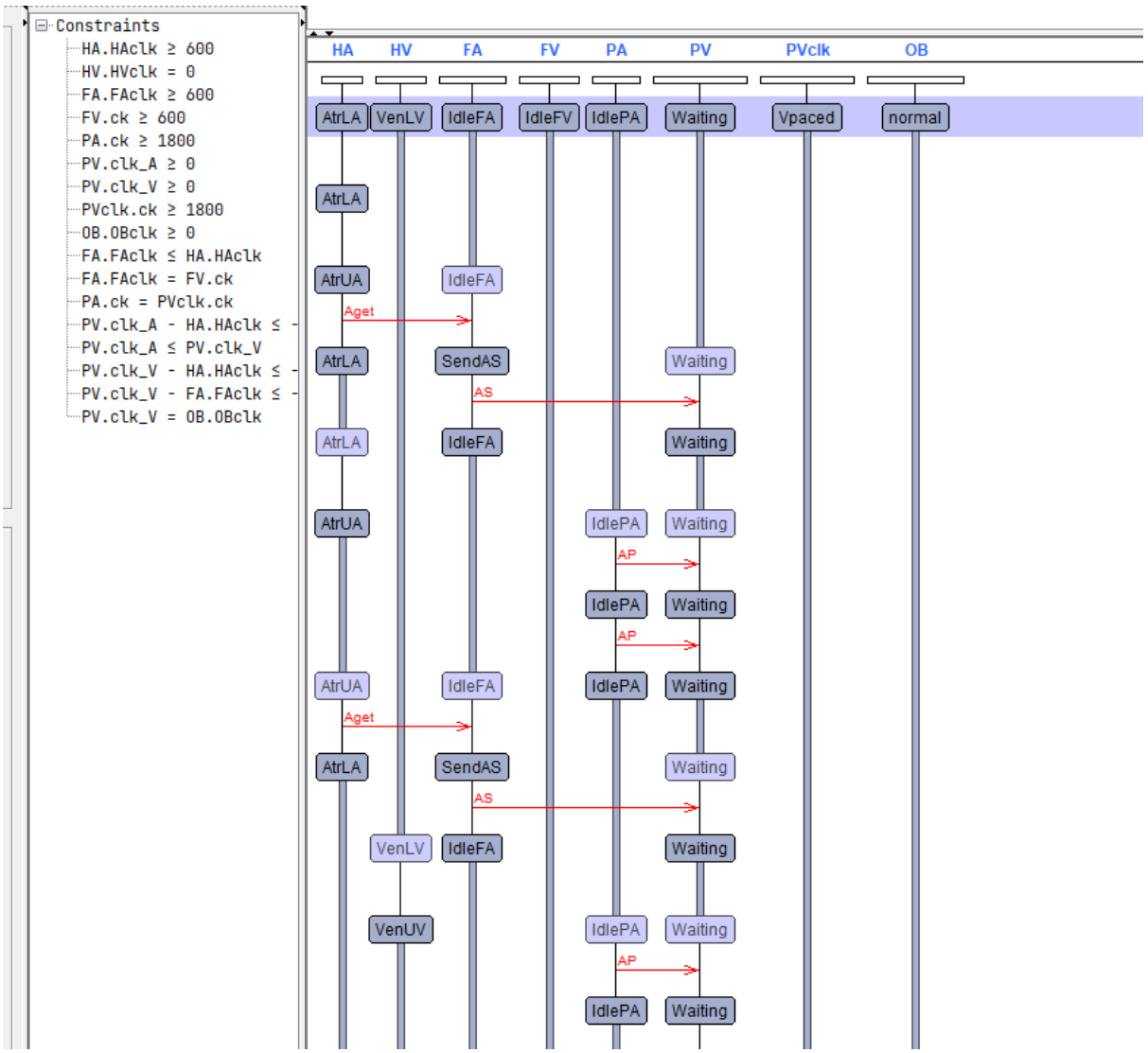
- **Vpaced:** Indicates that ventricular pacing has occurred.

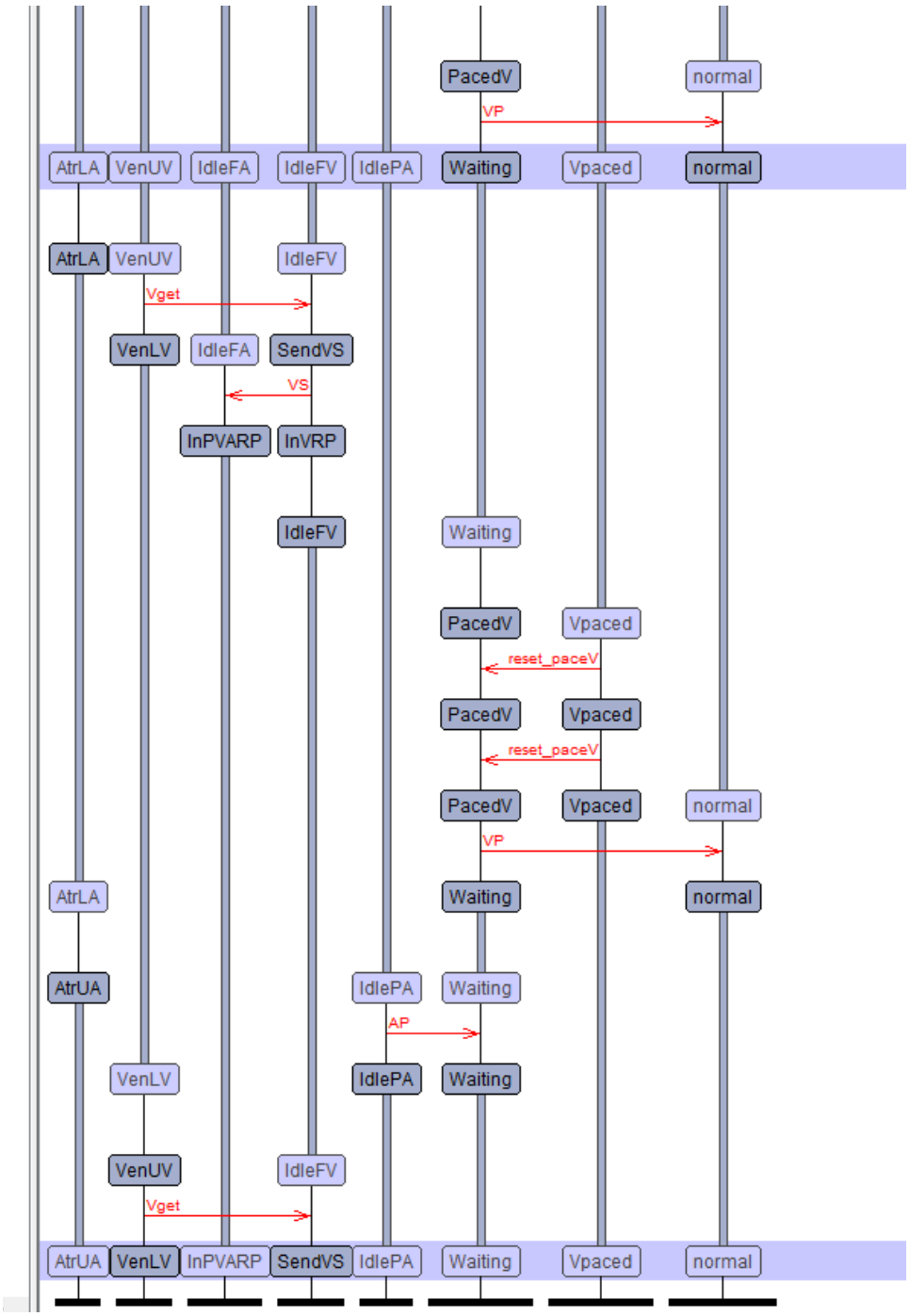
Role of Time:

- A clock `PVclk` measures the time since the last ventricular event.

Design Explanation:

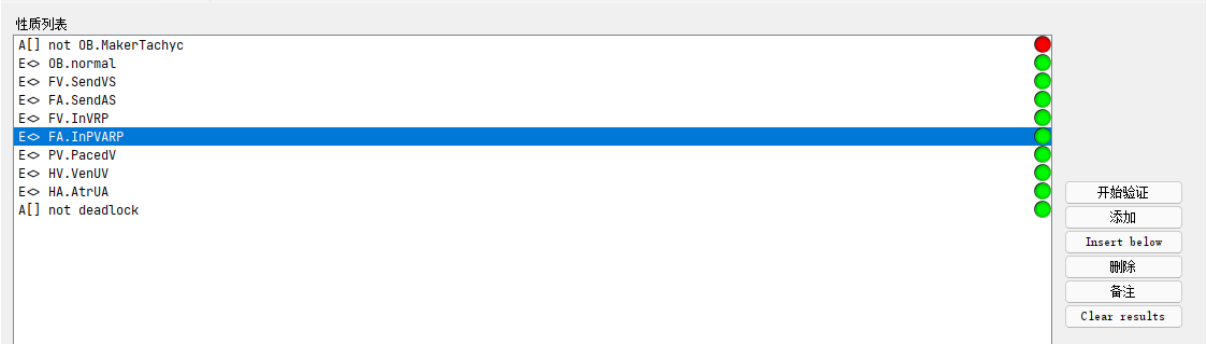
- The automaton remains in the `Vpaced` state, continuously monitoring ventricular events.
- Upon receiving a `VS?` signal, it resets `PVclk` to `0`, ensuring accurate tracking of ventricular activity.
- If a `reset_paceV!` signal is received, `PVclk` is also reset to maintain synchronization with pacing signals.
- This automaton aids in monitoring and controlling ventricular pacing to prevent inappropriate pacing intervals.


Constraints
HA.HAclk ≥ 600
HV.HVclk = 0
FA.FAclk ≥ 600
FV.ck ≥ 600
PA.ck ≥ 1800
PV.clk_A ≥ 0
PV.clk_V ≥ 0
PVclk.ck ≥ 1800
OB.OBclk ≥ 0
FA.FAclk ≤ HA.HAclk
FA.FAclk = FV.ck
PA.ck = PVclk.ck
PV.clk_A - HA.HAclk ≤ -
PV.clk_A ≤ PV.clk_V
PV.clk_V - HA.HAclk ≤ -
PV.clk_V - FA.FAclk ≤ -
PV.clk_V = OB.OBclk
HA
HV
FA
FV
PA
PV
PVclk
OB
AtrLA
VenLV
IdleFA
IdleFV
IdlePA
Waiting
Vpaced
normal
AtrUA
Aget
SendAS
AS
VenUV
AP



2. Choices on Verification Except Observer

In the verification process, we focused on testing the absence of deadlocks and the reachability of critical states in each automaton, based on the updated model.



Deadlock Testing

- **Query:** `A[] not deadlock`
- **Purpose:** To ensure that the system does not reach a state where no transitions are possible.
- **Result:** The query **succeeded**, indicating that the system is deadlock-free.

Reachability Testing

We tested the reachability of key states to confirm that under appropriate conditions, each component behaves as expected.

- **PaceV Ventricular Pacing:**
  - **Query:** `E<> PV.PacedV`
  - **Result:** The `PacedV` state is reachable, indicating ventricular pacing can occur.
- **PaceA Atrial Pacing:**
  - **Query:** `E<> PA.PacedA`
  - **Result:** The `PacedA` state is reachable, indicating atrial pacing can occur.
- **FilterA PVARP State:**
  - **Query:** `E<> FA.PVARP`
  - **Result:** The `PVARP` state is reachable, confirming the refractory period is entered as expected.
- **FilterV VRP State:**
  - **Query:** `E<> FV.VRP`
  - **Result:** The `VRP` state is reachable, confirming the ventricular refractory period is functioning.
- **PaceVclk Vpaced State:**
  - **Query:** `E<> PVclk.Vpaced`
  - **Result:** The `Vpaced` state is reachable, ensuring that ventricular pacing events are correctly tracked.

Verification Explanation:

- These tests confirm that each component can reach its critical states, ensuring that the pacemaker can perform pacing and sensing as designed.

- The absence of deadlocks confirms the system's liveness and responsiveness.
- The successful reachability of `PVclk.Vpaced` ensures that ventricular pacing events are appropriately monitored and controlled.

3. Observer Design

Purpose:

- To detect potential tachycardia caused by the pacemaker or the heart itself by monitoring the timing of ventricular events.

State Choices:

- **normal:** The default monitoring state, tracking time between ventricular events.
- **MakerTachyc (Committed):** Indicates pacemaker-induced tachycardia.
- **selfTachyc (Committed):** Indicates heart-induced tachycardia.

Role of Time:

- A clock `OBclk` measures the time since the last ventricular event.

Design Explanation:

- The observer checks if a ventricular event occurs too soon after the previous one.
- If `OBclk <= URI` and a `VS?` signal is received, it transitions to `selfTachyc`, indicating the heart is beating too fast.
- If `OBclk > LRI` and a `VP?` signal is received, it transitions to `MakerTachyc`, indicating the pacemaker is pacing too quickly.
- The use of committed states ensures immediate detection without delays, allowing for prompt identification of tachycardia conditions.

4. Observer Verifications

We tested the observer to ensure it correctly detects abnormal conditions.

- **Query:** `A[] not OB.MakerTachyc`
    - **Purpose:** To verify that the system does not enter the `MakerTachyc` state, ensuring the pacemaker does not cause tachycardia.
    - **Result:** The query **succeeded**, indicating that the `MakerTachyc` state is **not reachable**.

Verification Explanation:

- The success of this query confirms that the corrected `PaceV` automaton effectively prevents scenarios where the pacemaker might induce tachycardia.
- By using separate clocks for atrial and ventricular intervals, the timing constraints are accurately enforced, eliminating the previous issue.

- This ensures the pacemaker operates safely without causing excessively rapid heartbeats.

5. Difficulties and Further Development

Difficulties Encountered

- **Time Measurement Complexity:**
  - Initially, using a single clock in `PaceV` to measure both `AVI` and `URI` led to incorrect behavior.
  - Accurately representing multiple timing constraints required careful analysis and restructuring of the automaton.
- **Observer Limitations:**
  - After adding the observer, tachycardia caused by the pacemaker was still detectable.
  - It appears necessary to add additional constraints within the filters to prevent pacemaker-induced tachycardia, but determining the appropriate conditions was challenging.
- **Testing Error Interactions:**
  - Designing tests to detect the interplay of errors between components was uncertain.
  - Formulating queries using logical implications (e.g., `imply`) to test complex conditions proved difficult.

Further Development

- **Enhanced Filter Constraints:**
  - Introduce additional conditions within the `FilterV` and `FilterA` automata to further restrict and validate pacing signals.
  - For example, adding guards that ensure pacing signals adhere strictly to both `AVI` and `URI` constraints simultaneously.
- **Model Refinement:**
  - Continue refining the `PaceVclk` automaton to enhance its monitoring capabilities.
  - Ensure that all clocks are correctly reset and synchronized across automata to maintain accurate timing.
- **Advanced Testing:**
  - Develop more sophisticated queries that leverage logical operators to test complex interactions between automata.
  - Simulate pathological conditions to evaluate the pacemaker's response under various scenarios, ensuring robust performance.
- **Documentation and Naming Conventions:**
  - Improve naming conventions within the model for greater clarity and consistency.
  - Enhance documentation within each automaton to facilitate easier understanding and maintenance.

- **Future Extensions:**
    - Implement additional pacemaker modes and features, such as rate-responsive pacing or multi-chamber pacing.
    - Incorporate more detailed heart models to simulate different cardiac conditions and their interactions with the pacemaker.

6.Conclusion

This project provided valuable insights into modeling a pacemaker using UPPAAL. The design and verification process highlighted the importance of accurately representing timing constraints and the interactions between components. Although challenges were encountered, particularly in time management and observer limitations, these were addressed through careful analysis and model refinement. The introduction of separate clocks in the `PaceV` automaton successfully eliminated the risk of pacemaker-induced tachycardia, ensuring the system's safety and reliability. Future work will focus on enhancing the model's robustness, adding more sophisticated constraints, and extending its capabilities to cover a wider range of cardiac conditions and pacing scenarios.