10-2012

Model-Driven Safety Analysis of Closed-Loop Medical Systems

Miroslav Pajic  
*University of Pennsylvania*, pajic@seas.upenn.edu

Rahul Mangharam  
*University of Pennsylvania*, rahulm@seas.upenn.edu

Oleg Sokolsky  
*University of Pennsylvania*, sokolsky@cis.upenn.edu

David Arney  
*University of Pennsylvania*, arney@seas.upenn.edu

Julian M. Goldman  
*Massachusetts General Hospital & CIMIT*, jmgoldman@partners.org

See next page for additional authors

Follow this and additional works at: [https://repository.upenn.edu/mlab_papers](https://repository.upenn.edu/mlab_papers)

Part of the Electrical and Computer Engineering Commons

**Recommended Citation**

Miroslav Pajic, Rahul Mangharam, Oleg Sokolsky, David Arney, Julian M. Goldman, and Insup Lee, "Model-Driven Safety Analysis of Closed-Loop Medical Systems", *IEEE Transactions on Industrial Informatics* 10(1), 3-16. October 2012. [http://dx.doi.org/10.1109/TII.2012.2226594](http://dx.doi.org/10.1109/TII.2012.2226594)

An earlier version of this article can be found [here](https://repository.upenn.edu/mlab_papers/52).

This paper is posted at ScholarlyCommons. [https://repository.upenn.edu/mlab_papers/52](https://repository.upenn.edu/mlab_papers/52)  
For more information, please contact repository@pobox.upenn.edu.
Model-Driven Safety Analysis of Closed-Loop Medical Systems

Abstract
In modern hospitals, patients are treated using a wide array of medical devices that are increasingly interacting with each other over the network, thus offering a perfect example of a cyber-physical system. We study the safety of a medical device system for the physiologic closed-loop control of drug infusion. The main contribution of the paper is the verification approach for the safety properties of closed-loop medical device systems. We demonstrate, using a case study, that the approach can be applied to a system of clinical importance. Our method combines simulation-based analysis of a detailed model of the system that contains continuous patient dynamics with model checking of a more abstract timed automata model. We show that the relationship between the two models preserves the crucial aspect of the timing behavior that ensures the conservativeness of the safety analysis. We also describe system design that can provide open-loop safety under network failure.

Keywords
closed-loop medical systems, model-based development, safety analysis, high-confidence medical device systems

Disciplines
Electrical and Computer Engineering

Comments
An earlier version of this article can be found here.

Author(s)
Miroslav Pajic, Rahul Mangharam, Oleg Sokolsky, David Arney, Julian M. Goldman, and Insup Lee
Model-Driven Safety Analysis of Closed-Loop Medical Systems

Miroslav Pajic, Student Member, IEEE, Rahul Mangharam, Member, IEEE, Oleg Sokolsky, Member, IEEE, David Arney, Julian Goldman, and Insup Lee, Fellow, IEEE

Abstract—In modern hospitals, patients are treated using a wide array of medical devices that are increasingly interacting with each other over the network, thus offering a perfect example of a cyber-physical system. We study the safety of a medical device system for the physiologic closed-loop control of drug infusion. The main contribution of the paper is the verification approach for the safety properties of closed-loop medical device systems. We demonstrate, using a case study, that the approach can be applied to a system of clinical importance. Our method combines simulation-based analysis of a detailed model of the system that contains continuous patient dynamics with model checking of a more abstract timed automata model. We show that the relationship between the two models preserves the crucial aspect of the timing behavior that ensures the conservativeness of the safety analysis. We also describe system design that can provide open-loop safety under network failure.

Index Terms—closed-loop medical systems, model-based development, safety analysis, high-confidence medical systems

I. INTRODUCTION

Clinical scenarios for critical care patients often involve large numbers of medical devices. Some of these devices, such as bedside monitors, provide vital information about the state of the patient. Other devices (e.g., infusion pumps) provide treatment. That is, they affect the state of the patient (e.g., by infusing medication). Medical device systems, considered together with the patient and caregivers, represent an important class of cyber-physical systems. Patient safety is the primary concern in such systems, yet reasoning about patient safety is very difficult because of insufficient understanding of the dynamics of the human body’s response to treatment. Human errors, another important source of patient safety problems, are also difficult to reason about in the framework of conventional embedded system development.

It is natural to view a clinical scenario as a control system, in which the patient is a plant, bedside monitors are sensors and infusion pumps are actuators. Traditionally, caregivers perform the role of the controller in such a system. This means that the caregiver needs to continuously monitor all sensor devices and apply appropriate treatment. The large number of devices to monitor and control makes the job of the caregiver very difficult. On top of that, a caregiver is typically responsible for several patients. An emergency may divert the caregiver’s attention elsewhere, making him or her miss an important event. As a result, patient safety may suffer. Multiple such occurrences are documented in the clinical literature.

The human caregiver will always play an indispensable part of most clinical scenarios. However, in many cases automatic controllers can play important roles, reducing the burden on the caregiver and avoiding the possibility of human errors. A number of such cases was documented in the ASTM standard for the Integrated Clinical Environment [2], developed by the Medical Device Plug-and-Play Interoperability program (mdpnp.org) [3]. Although many medical devices today have network interfaces and can send sensed data across the network, few can be controlled remotely. Vendors of medical equipment continue to stay away from closed-loop scenarios.

The rationale is that it is difficult to reason about patient safety in closed-loop scenarios. Such reasoning is necessary for medical devices, which have to be approved for use by government regulators who assess their safety and effectiveness. A particular challenge arises from the complexity of a complex interplay between the continuous dynamics of the patient reaction to treatment and the discrete nature of the controller and communication network. The dynamics of the patient body is not well understood and exhibits parametric uncertainty and high variability between different patients.

To overcome this difficulty, we explore a model-driven approach that allows us to prove safety properties of devices on the modeling level and ensures that abstract models used in the verification process are sound with respect to the actual dynamics of the system. Both the abstract, formal model and the detailed, informal model are needed in the process of verification, validation, and regulatory approval of closed-loop medical device systems. On the one hand, formal models allow us to exhaustively explore the possible behaviors of the system and prove its safety. On the other hand, detailed models allow us to use high-fidelity simulation that take real system dynamics into account. Both kinds of results can be used to make the case for regulatory approval.

We start with a detailed model of the system in Simulink. The model contains a patient model that reflects continuous pharmacokinetic dynamics of drug absorption by the patient...
body. To reflect the variation between different patient populations, the patient model incorporates uncertainty in the values of its parameters. The Simulink model enables high-fidelity simulations of the system in various scenarios. Yet, currently there are no tools supporting the verification of hybrid systems (i.e., systems consisted of both discrete-time components and continuous-time dynamics) modeled in Simulink that are both nondeterministic and with uncertain parameters. Most existing tools for Simulink are limited to testing-based analysis that does not guarantee the coverage of parameter space.

To prove safety of the system, we express the safety property as a timing relationship between system components. We then model the system using the formalism of timed automata in the UPPAAL tool [4]. The model abstracts away continuous dynamics of the system, replacing it with timing constraints. The values for the timing constraints are obtained from the continuous dynamics of system components using the detailed Simulink model. It is necessary to guarantee that the patient model in UPPAAL over-approximates the patient’s dynamics with respect to the utilized control algorithm. For that, we present a procedure that utilizes convex optimization to calculate an upper bound on critical timing values for a linear time-invariant (LTI) model with uncertain parameters and show that behaviors of the UPPAAL model is always within this bound.

There is a need to develop a methodology for the analysis of safety properties of closed-loop medical device systems [5], [6]. In this paper we present a case-study focused on a system of clinical importance. In the case study, we prove that the system is safe under a set of assumptions. We then demonstrate that a violation of these assumptions (e.g., by a more realistic fault model) can make the system unsafe. Finally, we propose a solution to restore system safety under the new fault model. The approach presented in this paper is intended to serve as a reference point, a step toward a model-based methodology for safety analysis of closed-loop medical device systems.

The rest of the paper is organized as follows. Section II introduces the medical case study and presents the system architecture. Section III explains the use of Simulink models of the system to analyze safety with no failure assumption. It also describes a procedure used to translate safety requirements into timing constraints, when the patient is modeled as a continuous-time LTI system with uncertain parameters. Section IV describes the system model in UPPAAL and safety properties we verified. Section V proposes modifications to the system to deal with failures and provides an argument that the new system guarantees open-loop safety. The last section summarizes the paper and identifies avenues for future work.

II. CASE STUDY

In this section, we describe a case study that represents one of the MD PnP interoperability clinical scenarios [2]. In the past, we built a demonstration of this scenario [7] using the Integrated Clinical Environment (ICE) architecture, developed by the MD PnP project. Figure 1 shows the main components of an ICE-compliant system. The patient and caregiver are the human elements of the system. The Supervisor is the computer system that runs the control algorithm. Medical devices are connected, through adapters where necessary, to the Network Controller, which keeps track of connected devices and their capabilities. The Data Logger records pertinent network traffic for later forensic analysis and external networks such as the hospital information system are connected through an external interface. In our case study, we did not model the data logger since it does not affect our runtime safety analysis.

A. Clinical Use Case

The selected scenario involves a patient connected to a patient-controlled analgesia (PCA) infusion pump. PCA infusion pumps are commonly used to deliver opioids for pain management, for instance after surgery. Patients have very different reactions to the medications and require very different dosages and delivery schedules. PCA pumps give the patient a button to press to request a dose when they decide they want it rather than using a schedule fixed by a caregiver. Some patients may decide they prefer a higher level of pain to the nausea the drugs may cause and can press the button less often, while patients who need a higher dose can press it more often.

A major problem with opioid medications in general is that an excessive dose, or overdose, can cause respiratory failure. A properly programmed PCA system should not allow an overdose because it is programmed with limits on the number of doses it will deliver, regardless of how often the button is pushed. However, this safety mechanism is not sufficient to protect all patients. Some patients still receive overdoses if the pump is misprogrammed, if the pump programmer overestimates the maximum dose a patient can receive, if the wrong concentration of drug is loaded into the pump, or if someone other than the patient presses the button (PCA-by-proxy), among other causes. PCA pumps have been involved in a large number of adverse events [8], and existing safeguards such as drug libraries and programmable limits are not adequate to address all the scenarios seen in clinical practice [9].

The system we are considering aims to improve patient safety in such scenarios by introducing a supervisor that monitors patient data for the early signs of respiratory failure and can stop the infusion and sound an alarm if the patient experiences an adverse event. We use a pulse oximeter device that receives physiological signals from a clip on the patient’s finger and processes them to calculate heart rate and SpO₂.
outputs, where SpO$_2$ is the measure of blood oxygenation. Note that, at the time of writing, there are no PCA pumps on the market that are capable of being remotely controlled. In the demonstration system, we used the PCA pump prototype we have built during the Generic Infusion Pump project [10].

B. System Architecture

Figure 2 shows the components of the PCA safety system. The system components are described below. Figure 3 shows the devices and essential data flow in this control loop. In the case-study pulse oximeter acts as the Monitoring System from Figure 2. It receives physiological signals from the patient and processes them to produce heart rate and SpO$_2$ outputs. The Supervisor gets these outputs and makes a control decision, possibly sending a stop signal to the PCA Pump. Unless it is stopped by the Supervisor, the PCA pump delivers a drug to the patient at its programmed rate that corresponds to the selected operating mode. The patient model gets the drug rate as an input and updates the drug level in the patient’s body. This in turn influences the physiological output signals through a drug absorption function.

1) PCA Infusion Pump: The PCA pump in the case study operates in the following way. Before operation, the pump is programmed by the caregiver, who sets the pump’s operating mode, normal rate of infusion, the increased rate of a bolus, and bolus duration. Some PCA pumps also can be programmed to limit the total amount of drug to be infused. Once programmed and started, the pump delivers the drug at the normal rate until it is stopped or the bolus button is pressed. From that moment, it delivers drug at the bolus rate for the specified duration and then returns to the normal rate.

The pump is equipped with a number of built-in sensors that detect internal malfunctions such as the presence of air in the tubes that deliver the drug. When a problem is detected, the pump is stopped. We do not consider such malfunctions in this case study and do not represent the built-in alarm mechanism.

Finally, the pump is equipped with a network interface, which allows the pump to transmit its status across the network to other devices such as the logger. For the purpose of our scenario, we assume that the network interface allows the pump to accept control signals. A stop control signal will set the current infusion rate to zero, while the start signal will set the normal infusion rate (regardless of the state of the pump before it was stopped).

2) Monitoring System: Patients using a PCA pump are usually also attached to patient monitors that record the patient’s EKG, blood pressure, respiratory rate, and SpO$_2$. These monitors sound alarms if the measured values are outside thresholds set by the caregivers, but do not stop the infusion. Thus, the patient continues to receive more of an overdose while the caregiver responds, assesses the patient, and if a real problem occurred, finally stops the pump.

In this study, we look at using SpO$_2$ and heart rate measurements as the basis for a physiologic closed-loop control system that can stop the PCA pump and halt the dose of opioid while sounding an alarm if respiratory distress is detected. Both of these measurements can be produced by a device called Pulse Oximeter. This device is equipped with a finger clip sensor that shines two wavelengths of light through the patient’s finger. The measured light intensity reflects the blood oxygen content, which can change rapidly.

The pulse oximeter samples the patient’s SpO$_2$ at regular intervals, processes them, and outputs an averaged result [11]. It calculates the average using a variable-sized sliding window. The window size varies with the last output value. The reason for changing the window size is that a smaller sample size gives faster, but potentially less accurate results. When SpO$_2$ values are low, quick response is more important than filtering out transient noise. When SpO$_2$ is high, increasing the window size helps to filter out transient low values at the expense of less frequent updates. Since the samples are at regular intervals and a varying number of samples are used to calculate the output, the output is updated irregularly. The size of the sliding window used in the case study is determined using a simple table shown in Table I. Note that this table does not reflect the details of any real implementation but rather attempts to capture the essential behavior of a typical pulse oximeter.

3) Caregiver Model: The caregiver in this system programs the PCA pump and reacts to alarms. The control system is closed loop, so no intervention by the caregiver is necessary to stop the infusion when a problem is detected. The caregiver can react to restart the system if it has stopped in reaction to a false alarm, or when a problem such as a slipped patient sensor is fixed.

![Fig. 2. Hardware for the PCA Demo System.](image-url)

![Fig. 3. PCA System Control Loop.](image-url)

<table>
<thead>
<tr>
<th>Sliding Window Size for Pulse Oximeter</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>new window size</td>
<td>10</td>
</tr>
<tr>
<td>last output value</td>
<td>97 - 100</td>
</tr>
</tbody>
</table>
4) **Patient Model**: To model the Patient we use a pharmacokinetic patient model for intravenous delivery of anesthetic drugs presented in [12] (module 12). The model utilizes a common 3-compartment model to describe the changes in drug concentration in blood and tissue. The patient is described as a continuous-time LTI system with a state space representation:

\[
\begin{bmatrix}
C_1 \\
C_2 \\
C_3
\end{bmatrix} =
\begin{bmatrix}
-(k_{12} + k_{13} + k_{10}) & k_{21} & k_{35} \\
k_{12} & -k_{12} & 0 \\
k_{13} & 0 & -k_{31}
\end{bmatrix}
\begin{bmatrix}
C_1 \\
C_2 \\
C_3
\end{bmatrix} + \begin{bmatrix}
\frac{1}{V_1} \\
0 \\
0
\end{bmatrix}I
\]

\(dl = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} C\)

where \(C_i, i \in \{1, 2, 3\}\) is the drug concentration in the \(i^{th}\) compartment, \(I\) is the mass infusion rate (in mass/unit of time) of the drug, \(k_{ij}\) are patient specific constants and \(V_1\) is the volume of the blood plasma compartment. Since the first compartment presents blood plasma, its concentration is mapped into the drug level concentration \(dl\) as in (1).

Using the above model, we represent the instantaneous level of medication in the patient’s body as a single variable \(dl\). This variable is linked to the patient’s heart rate and \(SpO_2\) level by the drug absorption function, which represents how the patient reacts to the dose received over time. Using \(dl\) value, the state space of the patient is partitioned into three regions: in pain (under-medicated), pain-controlled (adequate medication), or over-medicated. These regions are defined as (Figure 4):

- The **Safe** region is defined as the region where the Patient’s readings are below some predefined threshold values that guarantee that Patient’s vitals are not endangered.
- If the patient is over-medicated to the point that he or she starts experiencing respiratory distress, we consider it an overdose. Thus, we define the **Critical** region where Patient’s life is in danger or there is a chance that irreparable damage can occur. The overdose condition is referred as the border of the Critical region.
- The **Alarm**ing region defined as the region where Patient’s vitals are not endangered but there is a reasonable concern that Patient can be forced to the Critical region.

Any treatment needs to make sure that the patient stays out of the critical region, and we use this requirement as the main safety property of the system that needs to be ensured. In this case study, since the drug level is estimated from the patient’s \(SpO_2\) and heart rate, we defined the boundary of the Critical region in terms of these values and set it to \(H_2^{safe} = 70\%\) for \(SpO_2\) (and \(H_2^{pulse} = 11.5\) beats/min for heart rate), a clear indication of respiratory failure.

Some patients react very quickly to a drug dose, while others react more slowly. Furthermore, a mapping between the drug level on one side and measured \(SpO_2\) level and heart rate is also patient specific. Therefore, by adjusting this function, we can tune the model to different patient types. However, in all these cases patient reaction to drug can be described as in (1), where patient specific coefficients belong to predefined regions and fully determine the patient’s behavior.

To analyze the dynamics of the patient’s pharmacokinetic model, we have designed the system model in Simulink (the model is described in next section). This enables simulation and analysis of the system’s behavior under different scenarios.

5) **Supervisor**: The supervisor is a running program in the system that communicates with other devices and executes clinical application scripts (CAS’s). A CAS is a script that implements a particular clinical use case. The clinical application in this case study is to control the loop shown in Figure 2. The Supervisor receives the patient’s heart rate and \(SpO_2\) measurements from the pulse oximeter and uses this information to decide whether the PCA infusion pump should be allowed to run or immediately stopped.

In the case study, two simple Supervisor’s control algorithms were designed. In the first algorithm, after the pump is activated the decision to stop the pump is made as soon as the patient heart rate or \(SpO_2\) readings fall below a fixed threshold. The threshold value needs to ensure that the patient does not enter the Critical region despite the delay in detecting the problem and delivering the control signal to the pump. For the case study, we defined the threshold as \(H_1^{safe} = 90\%\) for the \(SpO_2\) and \(H_1^{pulse} = 57\) beats/min for heart rate. Values below these thresholds typically indicate “a clinical concern” ([13], p. 45), meaning that a caregiver needs to be notified. The supervisor notifies the caregiver when the threshold is crossed, as it sends the message to stop the pump. Values between \(H_1\) and \(H_2\) are thus referred as the Alarm region. The width of the alarming region is denoted as \(\Delta H = |H_2 - H_1|\).

The second control algorithm is similar, with a small difference that in this case the pump is always activated for a fixed, predefined duration of time. Therefore, the Supervisor does not need to send commands to stop the pump unless, as in the first design, it detects that the Patient have entered the Alarm region. In this case, the Supervisor again sends a command to stop the pump and alerts the caregiver.

### III. **Modeling System Dynamics in Simulink**

The overall structure of the Simulink model follows that of the model shown in Figure 2. The Simulink blocks are used to capture the dynamics of the PCA infusion pump, Pulse Oximeter (PO), Patient model, and Supervisor, described in Section II. In addition, they are used to define the notion of the safe, critical, and alarming regions precisely. Using the overall structures and timing properties of the components, this section describes when the system is safe and when it is not safe. Finally, it identifies the limitations of the model with respect to its use in system verification.

#### A. Simulink Models

The PO is implemented as described in Section II (a more detailed description of the PO design is presented in
Section IV). It monitors the Patient’s HR and SpO2 level and informs the Supervisor about these values. Figure 5 presents the Patient’s model, where effects of the drug flow are depicted using the Patient model described in Section II. Therefore, the Patient is modeled as a continuous-time LTI system where matrices \( \mathbf{A}, \mathbf{B} \) and \( \mathbf{C} \) are defined as in (1). For the Patient model that describes a particular population of subjects, we use the parameters for a scenario of intravenous delivery of anesthetic drugs ([12], page 694). Here, the kinetic values in (1) are initialized as:

\[
\begin{align*}
\hat{k}_{10} &= 0.152 \text{ min}^{-1}, \quad \hat{k}_{12} = 0.207 \text{ min}^{-1}, \quad \hat{k}_{13} = 0.040 \text{ min}^{-1} \\
\hat{k}_{21} &= 0.092 \text{ min}^{-1}, \quad \hat{k}_{31} = 0.048 \text{ min}^{-1}, \quad \hat{V}_1 = 12 \text{ liters}.
\end{align*}
\]

(2)

In the general case, to be able to express Patient dynamics for different types of patient populations, it is not possible to fix a value for each of the parameters \( k_{ij} \). Therefore, to model the Patient’s dynamics in the general case, each parameter \( k_{ij} \) is initialized with a value within a closed region:

\[
\begin{align*}
\hat{k}_{ij} &\in \left[ \hat{k}_{ij} - \Delta k_{ij}, \hat{k}_{ij} + \Delta k_{ij} \right] \\
\hat{V}_1 &\in \left[ \hat{V}_1 - \Delta V, \hat{V}_1 + \Delta V \right],
\end{align*}
\]

(3)

which is a simplified version of the function presented in [14]. This effectively means that matrices \( \mathbf{A}, \mathbf{B} \), defined in (1), in the general case (i.e., for all types of patients) belong to a specific polyhedron. The patient’s Heart Rate (HR) and SpO2 level are extracted from its drug level using a linear mapping, as shown in Figure 5. As the mapping between the drug level, measured SpO2 level and heart rate is patient-specific, it is necessary to include this additional level of uncertainty into the model. Therefore, in the implemented model, the gain in matrix \( \mathbf{C} \) (i.e., element \( C_{11} \)) can also take values within a pre-specified region \([1 - \Delta c, 1 + \Delta c]\).

To analyze the patient’s dynamics for parameter values from (2), the patient’s behavior was simulated for a case when the drug is repeatedly delivered for one hour followed by one hour pause. Figure 6 presents the obtained changes in HR and SpO2 levels during the period of almost three hours.

\[ t_{POdel} + t_{net} + t_{sup} + t_{net} + t_{Pump} + t_{PCA2P} \leq t_{crit} \]  

(4)

where:

- \( t_{POdel} \) - worst case delay caused by the PO; it can be calculated from the PO specification, see Table I,
- \( t_{net} \) - worst case network delay; the value depends on the used network protocol,
- \( t_{Sup} \) - worst case delay introduced by the Supervisor; it can be obtained from the Supervisor model,
- \( t_{Pump} \) - worst case delay introduced by the PCA pump; the value can be derived from the pump model,
- \( t_{PCA2P} \) - worst case delay between the moment the drug flow is activated and the moment it reaches the

- \( t_{POdel} + t_{net} + t_{sup} + t_{net} + t_{Pump} + t_{PCA2P} \leq t_{crit} \)
patient; it depends on the type of used connection.
• \( t_{crit} \) - Patient’s critical time, a shortest time that the patient spends in the Alarming region before entering the Critical region; it can be computed from the patient’s model.

If (4) is satisfied, we can guaranteed that the Supervisor will be able to determine that the patient has entered the Alarming zone and stop the PCA pump before the patient switches from the Alarming to Critical zone. The values for these time constants are shown in Table II. All values, beside \( t_{net} \), can be obtained from the system specification. Here, for safety analysis we assume that the network latency \( t_{net} \) is bounded by 0.5s and we will try to relax the bound later.

Remark 1: Currently there exist several Simulink-based network simulators that can be used for network modeling (e.g., TrueTime). However, from the perspective of safety analysis and relation (4), the network can be modeled as a random delay with delay distribution that matches the application scenario. It is worth noting here that the network induced delay is not fixed and it depends on the underlying network controller (e.g., 802.11 MAC with distributed (DCF) or point (PCF) coordination function) and environmental conditions (e.g., traffic, presence of interference). Also, although implemented network protocols can improve the overall Quality of Service, in general case the delay cannot be bounded when wireless networks are used, no matter which communication protocol is employed. Hence, even though we start with the assumption on bounded network delay, we also consider scenarios with unbounded network delays (i.e., messages can be lost), along with its consequences on the system’s safety.

The main problem in determining whether the safety condition is satisfied is to find the value for \( t_{crit} \). If a mathematical model for the patient is known, and patient-specific (fixed) parameters can be obtained, this value can be analytically or numerically determined. In our model, for a specific population of subjects we can use fixed parameters as in (2). Since the Patient model is modeled as an LTI process when the drug flow is on, time-responses for drug level, HR and SpO2 level have the same general form:

\[
e_{min} + a_1 e^{-\lambda_1 t} + a_2 e^{-\lambda_2 t} + a_3 e^{-\lambda_3 t}
\]

where \( \lambda_1 = -0.0079 \), \( \lambda_2 = -0.0023 \), \( \lambda_3 = -0.0007 \) are eigenvalues of matrix \( \mathbf{A} = \hat{\mathbf{A}} \) (from (2)) and constants \( e_{min}, a_1, a_2 \) and \( a_3 \) belong to different sets of constants (depending whether we consider the drug level, HR or SpO2) for a fixed drug infusion rate.

To calculate \( t_{crit} \) consider time instances, \( t_1 \) and \( t_2 \), when the Patient enters the Alarming and Critical regions, respectively. If the Patient is continuously pushed toward the Critical region (which in this case means that the pump is continuously delivering drug to the Patient), then \( t_{crit} = t_2 - t_1 \). Considering the Alarming region boundary for SpO2 it holds that:

\[
H_i^{SpO_2} = e_{min} + a_1 e^{-\lambda_1 t_1} + a_2 e^{-\lambda_2 t_1} + a_3 e^{-\lambda_3 t_1}
\]

Thus, we can compute \( t_{crit} \) from \( H_i^{SpO_2} = \Delta H_i^{SpO_2} \).

For our model the Patient’s dynamics is dominated by the maximal eigenvalue \( \lambda_3 \), which is approximately \(-0.0007s^{-1}\), but the other eigenvalues also have some impact. Thus, \( t_{crit} \) is a couple of orders of magnitude bigger than the sum of all other timing parameters from (4). For \( H_i^{SpO_2} = 90\% \) and \( H_i^{SpO_2} = 70\% \) we have \( t_{crit} \approx 581.2 s \approx 9.68 min \). Therefore, for these patient parameters that correspond to a particular population, our system always satisfies the safety requirement if the assumption that all messages are delivered is valid.

2) Safety analysis for the model with uncertain parameters:
The aforementioned procedure for \( t_{crit} \) calculation can be used only when exact values for the parameters in matrices \( \mathbf{A}, \mathbf{B} \) and \( \mathbf{C} \) are known. As specified earlier in the section, in the general case we can only claim that matrices \( \mathbf{A}, \mathbf{B} \) and \( \mathbf{C} \) belong to polyhedrons \( \mathbb{R} \{ \mathbf{A} \} \), \( \mathbb{R} \{ \mathbf{B} \} \) and \( \mathbb{R} \{ \mathbf{C} \} \), respectively. For example, a polyhedron \( \mathbb{R} \{ \mathbf{A} \} \) can be defined as:

\[
\mathbb{R} \{ \mathbf{A} \} = \{ \mathbf{A} | -\Delta \mathbf{A} \leq \mathbf{A} - \hat{\mathbf{A}} \leq \Delta \mathbf{A} \}
\]

where \( \leq \) denotes element-wise inequality and for all \( \Delta k_{ij} \) from (3) the matrix \( \Delta \mathbf{A} \) is defined as

\[
\Delta \mathbf{A} = \begin{bmatrix}
-\Delta k_{12} & \Delta k_{13} & \Delta k_{10} \\
\Delta k_{12} & -\Delta k_{12} & 0 \\
\Delta k_{13} & 0 & -\Delta k_{13}
\end{bmatrix}
\]

Polyhedrons \( \mathbb{R} \{ \mathbf{B} \} \) and \( \mathbb{R} \{ \mathbf{C} \} \) can be similarly defined.

To model different types of patient populations, in the Patient model each of the predefined parameters can vary up to 10% from its initial value (e.g., \( \Delta k_{12} = 0.1k_{12} \)). In this case, to check the safety condition from (4) it is not possible to use the time response from (5) since matrices \( \mathbf{A}, \mathbf{B} \) and \( \mathbf{C} \) are unknown. To deal with this type of uncertainty, when matrices belong to a predefined set of polyhedrons, the following theorem can be utilized:

**Theorem 1:** Consider an LTI system with uncertain parameters of the form:

\[
\begin{align*}
\dot{x}(t) &= \mathbf{A}x(t) + \mathbf{B}u(t) \\
y(t) &= \mathbf{C}x(t)
\end{align*}
\]

where \( \mathbf{A} \in \mathbb{R} \{ \mathbf{A} \}, \mathbf{B} \in \mathbb{R} \{ \mathbf{B} \} \) and \( \mathbf{C} \in \mathbb{R} \{ \mathbf{C} \} \). If a constant input vector \( u_i \) is connected to the system’s input at time \( t_0 \) and \( x_0 = x(t_0) \), then for all \( t \geq 0 \):

\[
t \geq \frac{1}{||\mathbf{A}||} \ln \left( \frac{||y(t_0 + t) - y(t_0)||}{||\mathbf{C}|| \cdot (||x_0|| + ||\mathbf{B}|| + ||\mathbf{C}|| ||\mathbf{B}|| |x_0|)} + 1 \right)
\]

where:

\[
\mathbf{A} = \arg \max_{\mathbf{A} \in \mathbb{R} \{ \mathbf{A} \}} ||\mathbf{A}||, \quad \mathbf{B} = \arg \max_{\mathbf{B} \in \mathbb{R} \{ \mathbf{B} \}} ||\mathbf{B}||, \quad \mathbf{C} = \arg \max_{\mathbf{C} \in \mathbb{R} \{ \mathbf{C} \}} ||\mathbf{C}||
\]

\[
\mathbf{A}_{min} = \arg \min_{\mathbf{A} \in \mathbb{R} \{ \mathbf{A} \}} ||\mathbf{A}||
\]

2 Although we cannot guarantee that the uncertainty bound (i.e., 10%) can be used to model all types of uncertainties in the patient model, we have opted to use it as it provides a way to model different types of patients. However, if the patient’s dynamics (pharmacokinetics) is better understood for particular types of patient populations, a new uncertainty bound can be easily adopted.
Proof: See Appendix A

If $x_0 = x(t_0)$ is known, the bound for $t_{crit}$ can be computed as shown in (8). However, in cases where only values $y(t_0)$ and $y(t_0 + t)$ are known it is not possible to exactly determine $x_0$, since $y(t_0) = Cx_0$ and matrix $C$ might not be invertible (as in our case). In this case, to compute a lower bound $\tilde{t}_{crit}$ we consider a vector norm $||\tilde{x}_0||$ that is an upper bound on the maximal norm vector $x$ can take. To obtain this value we simulated the patient’s dynamics when the parameter values are initialized as $A = \hat{A}$ and $B = \hat{B}$ (from (9)), since this configuration at each time instance maximizes the value for $||\dot{x}||$ from (1). Note that the proposed bound on $||x_0||$ is very conservative as it provides an upper bound on $||x(t)||$ in the general case (i.e., for the reachable domain) and not only in points where $y(t_0) = Cx_0$. However, as we will see later, even with this bound safety can be guaranteed.

When the pump is continuously delivering the drug, a lower bound for $t_{crit}$ (i.e., $\tilde{t}_{crit}$) can be derived using Theorem 1. In this case, the safety conditions will be satisfied if:

$$t_{Podel} + t_{Sup} + t_{net} + t_{Pump} + t_{PCA2P} \leq \tilde{t}_{crit}$$ (11)

where:

$$\tilde{t}_{crit} = \frac{1}{|A|} \ln \left( \frac{|\Delta H_{SpO_2}|}{SpO_2_{gain}} \cdot \frac{|C||B_{u_1}|}{|A_{max}|} \right) + 1$$ (12)

and $SpO_2_{gain} = 0.35 \cdot 5.8 = 2.03$ is the gain from the linear mapping between the drug level and SpO_2 level (Figure 5).

The value for $|\hat{A}_{min}|$ can be efficiently obtained from the convex optimization problem, using existing methods/tools for convex optimization. We used CVX, a package for specifying and solving convex programs [15], to solve this optimization problem for the predefined region $\mathcal{R}(\hat{A})$, and obtained the value $|\hat{A}| = 7.34 \cdot 10^{-3}$. On the other hand, it is not possible to utilize efficient methods for convex optimization to obtain $|\hat{A}|$, $|\hat{B}_{u_1}|$ and $|C|$, because $A_{max}$, $B$, $C$ are arguments in maximization over convex functions. However, to compute these values the following theorem can be employed:

Theorem 2: Consider the optimization problem:

$$\max_{A,E} \mathcal{R}(A)|A|$$, where $\mathcal{R}(A)$ is defined as in (6).^3

Then, there exists an optimal solution $A_{max}$ which is a vertex of the polyhedron $\mathcal{R}(A)$.

Proof: See Appendix B.

From the previous theorem, to compute $|\hat{A}|$ it is necessary to calculate $|\hat{A}|$ in all $2^{n^2}$ vertices of the polyhedron. In our case, after computing the norm in $2^9 = 512$ points, we obtained $|A| = 8.97 \cdot 10^{-3}$. Similarly, after computing the norm in 2 points for both $|\hat{B}_{u_1}|$ and $|\hat{C}|$, we obtained $|\hat{B}_{u_1}| = 0.0917$, $|\hat{C}| = 1.100$. Furthermore, after matrices $A$, $B$, $C$ had been determined, using the previously described approach we obtained the value $||\tilde{x}_0|| = 27.08$. Therefore, in our model $t_{crit} = 24.83 s \approx 0.41$ min. Note that this bound is significantly lower than $\tilde{t}_{crit}$ calculated for the ‘fixed’ patient model (which was 9.68 minutes). As mentioned before, this bound can be improved if $||\tilde{x}_0||$ is calculated only at points where $C\hat{x}_0 = y(t_0)$. However, even with this conservative bound, the value for $t_{crit}$ satisfies the safety requirement if communication delay imposed by the network is bounded.

Remark 2: The bound provided in (12) can be improved using the approach presented in [16]. In addition, this approach can be utilized in cases where a non-constant drug flow is delivered to the patient. However, the flexibility comes with a higher computational cost required by this procedure. Finally, for complex Simulink patient models (e.g., nonlinear systems with uncertain parameters or nondeterministic systems) it might not be possible to analytically determine $\tilde{t}_{crit}$. In this case patient’s behavior can be simulated, and since simulation results depend on input signals and the model’s initial state, it is essential that the worst-case scenario is known. Otherwise, methods similar to the one from [17] can be used.

3) Safety conditions for the second Supervisor design: As described in Section II, for the second Supervisor design the pump is configured to stay activated only for a predefined fixed duration of time. In addition, Supervisor stops the pump only if the patient is in the Alarming region. In this case, the safety requirement is satisfied if at least one of the two conditions are met. The first condition is described in (4) and it is equivalent to the safety condition for the first Supervisor design. The second safety requirement is that the PCA activation period $t_{dur}$ (i.e., duration of the drug flow) satisfies the requirement that it can not drive the Patient from the Safe region to the Critical region. This condition can be specified as $t_{dur} \leq t_{crit}$. Therefore, the second safety requirement is satisfied if:

$$t_{dur} \leq t_{crit},$$ (13)

with $t_{crit}$ defined as in (12).

4) Limitations of system verification in Simulink: Currently there does not exist a holistic approach for verification of nondeterministic hybrid systems, with uncertain parameters, modeled in Simulink. Most of the existing tools use procedures like Instrumentation Based Verification [18] where monitors are designed to check for violations of the safety conditions. However, this approach employs coverage-based testing, and thus can be used only in combination with tools for automatic test generation. Several such commercial (e.g., Reactis, Design Verifier) and non-commercial (e.g., [17]) tools exist, but they do not guarantee full coverage of the design state-space.

From the safety conditions described in (4) and (13), we infer that safety of the closed-loop system can be mapped into analysis of timing relations between the system’s components. Thus, we have opted to use UPPAAL [4] for verification of the closed-loop system. UPPAAL also facilitates development of more detailed system models since it inherently supports composition of asynchronous components. However, when UPPAAL is used for the system’s verification it is necessary to guarantee that the patient model in UPPAAL over-approximates the patient’s dynamics with respect to the utilized control algorithm (as described in the next section).

IV. SYSTEM MODELING AND VERIFICATION IN UPPAAL

This section describes UPPAAL modeling of the closed-loop system. The structure of the model follows the architecture of the system. For each component in Figure 2, the model
includes a separate automaton. The automata communicate using synchronization channels and shared variables. Figure 7 shows the network of automata and communication between them. Solid arrows represent communication channels, while dashed arrows denote shared variables.

### A. UPPAAL Component Models

1) **PCA Automaton**: The automaton, which models the pump, is shown in Figure 8. When the pump is operational, it is either in the state running, with the shared variable pca_rate set to default rate, or in the state bolusing, when pca_rate is increased by the bolus rate. Both rates are specified as parameters of the model. Furthermore, by setting the appropriate value for time_mode the pump can be set (at configuration, not runtime) to work in the both previously mentioned modes. In the first mode the pump is kept continuously on (i.e., in bolusing state) until stopped by the Supervisor, while in the second mode the pump can be bolusing for a fixed duration given by the value of the bolus_time parameter. The pump transitions to the bolusing state upon the signal received from the patient only if it is in the running state; in all other states, the signal is ignored. From either running or bolusing state, the pump can move to a stopped state (Rstopped or Bstopped, respectively) upon a signal from the network.

2) **PO Automaton**: The automaton, which represents the pulse oximeter, is shown in Figure 10. The operation of the automaton proceeds in rounds. Each round begins by setting the window size for the round based on the last sampled value. Then, the automaton collects the number of samples to fill the window. Samples are obtained periodically with the interval of 1 time unit, which corresponds to 100 ms. Finally, the result is stored in the po_result variable and delivered to the supervisor using the resultready channel.

3) **Network and Caregiver Automata**: The network is modeled using two automata. Communication from the Supervisor to the pump is modeled with automaton from Figure 9. The automaton implements a two-place buffer, which means that there may be two network messages in transit. The stop message may be dropped by the network, if the boolean parameter drop is set to true. We do not model dropping of the restart message, since the loss of these messages does not affect the safety of the patient. If messages are not dropped, they are delivered by the network in order. A similar automaton (not shown here) is used to introduce a bounded delay for PO measurements (i.e., po_result). Again, messages can be dropped if drop is set to true. The Caregiver automaton, not shown here, contains one state and can send any of the messages at any time. The messages include PCA pump and Supervisor configuration commands (e.g., CG2PCA_start, CG2S_clear). A more detailed model of the caregiver may include data-dependent behaviors, for example, the clear signal may be sent only if the SpO₂ reading is high enough. However, any other model will have fewer behaviors than the caregiver model used here, and thus the safety property will still hold.

4) **Supervisor Automaton**: The Supervisor automaton, shown in Figure 11, implements the simple control algorithm. Upon receiving a SpO₂ reading from the pulse oximeter, the supervisor compares it with the pre-defined threshold value and, if the result is too low, sends the stop message to the pump across the network. The model also incorporates a delay, which represents the worst-case execution time of the supervisor algorithm. Then, once the caregiver resolves the problem, the supervisor sends another message to restart the pump. For simplicity of the presentation, the Supervisor automaton only deals with SpO₂, not heart rates.

5) **Patient Automaton**: The Patient automaton, shown in Figure 12, periodically updates the drug level based on the
flow rate of the pump and drug absorption rate. In addition, at any time, the model can deliver a sample as the function of the current drug level. The designed model implements a linear patient dynamics, which does not match the more realistic Simulink model. An important question that needs to be addressed is the consistency between the two models. Since the UPPAAL model of the controller uses significantly simplified patient dynamics, there is the possibility that verification results obtained on the UPPAAL model would not apply to the more detailed model and thus the system itself. Therefore, in the remaining of this subsection we address initialization of the Patient’s and PCA pump automata.

6) Initializing model parameters: To determine values for the parameters of Patient and PCA pump automata we consider the Patient’s behavior with respect to the safety requirements. From safety conditions from Section III it follows that the focus of interest for the safety analysis is with the Patient’s behavior in the Alarming region. For both Supervisor designs it is required that the minimal time that the Patient can spend in the region is bounded by \( t_{\text{crit}} \) (from (12)). Therefore, the value of \( t_{\text{crit}} \), the minimal time it takes for the patient to be overdosed, is an important consistency check for the models.

In UPPAAL, the value for \( t_{\text{crit}} \) may be different (we denote it as \( t_{\text{crit}}^U \)), and can be obtained from the Patient model as:

\[
 t_{\text{crit}}^U = \left\lceil \frac{\Delta H^{\text{SpO}_2}/\text{SpO}_2\text{gain}}{\text{PCA rateON} - \text{ab rate}} \right\rceil, \tag{14}
\]

where \( \text{PCA rateON} = \text{default bolus rate} + \text{default rate} \) and \( \lceil x \rceil \) denotes the smallest integer greater than or equal to \( x \). To guarantee consistency between the UPPAAL and Simulink models, the value for \( t_{\text{crit}}^U \) has to be no greater than the previously obtained value \( t_{\text{crit}} \). Consequently, from condition \( t_{\text{crit}}^U \leq t_{\text{crit}} \) and (14) we can derive the UPPAAL model parameters (default bolus rate and default rate from the PCA automaton and ab rate from the Patient automaton).

In our case, \( t_{\text{crit}}^U \) turns out to be 20 seconds for the initial values of the parameters that describe the patient dynamics and drug flow (\( \text{pca rate} = 3, \text{ab rate} = 2 \), which is an order of magnitude smaller than \( t_{\text{crit}} \). Clearly, the UPPAAL model overestimates the rate of change for the drug level in the patient’s body. And, since the system has been proven safe in this case, it would also be safe in a more realistic case. However, since we want to more realistically mimic the patient’s behavior in the Alarming region it is necessary to set the parameters in a way that:

\[
 \left\lceil \frac{\Delta H^{\text{SpO}_2}/\text{SpO}_2\text{gain}}{\text{PCA rateON} - \text{ab rate}} \right\rceil = \lceil t_{\text{crit}} \rceil \tag{15}
\]

This also allows for relaxation of timing requirements imposed on other components in the control loop. For example, with this value it is possible to relax the bound on the network imposed delay (\( t_{\text{net}} \)) from Table I.

Furthermore, it is necessary to achieve consistency between the UPPAAL and physical patient models in cases when the pump is turned off. Here, the goal is to minimize the difference between the times (obtained from UPPAAL and Simulink models) that the patient would spend in the Alarming zone if...
the pump is turned off at the boundary of the Critical zone. As in the previous case, the condition that relates UPPAAL model parameters with the patient model from (1) can be obtained from (12) where \( u_i = 0 \) (i.e., drug delivery is turned off) and rate\(_{\text{down}} = ab\_rate - \text{default\_bolus\_rate}\):

\[
\frac{\Delta H^\text{rate}/\text{SpO}_2\_\text{gain}}{\text{rate\_down}} = \frac{1}{||A||} \ln \left( \frac{\Delta H^\text{rate}/\text{SpO}_2\_\text{gain}}{||C|| \cdot |x_0|} + 1 \right)
\]

From (15) and (16) the parameters for PCA and Patient automata can be derived.

**B. Verifying Safety Properties of the System**

The main safety property that needs to be verified on the UPPAAL model is whether or not the patient can enter the Critical region, where SpO\(_2\) and heart rate are low enough to indicate a respiratory arrest. Before verifying safety, however, we perform several auxiliary checks to ensure model sanity.

We express properties we verify in the subset of the Computational Tree Logic (CTL) [19] used by UPPAAL. The main temporal operators of this logic that we use are \( A\bigotimes\phi \), which means that \( \phi \) is satisfied in every state along every execution path from the current state, and \( A\bigotimes\phi \), meaning that \( \phi \) is satisfied eventually along every path.

The first sanity check is the absence of deadlocks in the model. Another sanity check is that once the SpO\(_2\) level goes below the pain threshold, it eventually goes up. This property is captured by the temporal formula

\[
A\bigotimes(\text{samplebuffer} < \text{pain\_thresh} \Rightarrow A\bigotimes(\text{samplebuffer} \geq \text{pain\_thresh})).
\]

Note that the property is defined in terms of the true SpO\(_2\) level as defined by the patient model, not the sensor reading obtained by the supervisor. Intuitively, this property should hold, because the normal infusion rate is lower than the drug absorption rate. Once the patient stops requesting new boluses and the last bolus infusion is over, the drug level will start decreasing and thus SpO\(_2\) and heart rate levels should increase, until they reach pain threshold again. We also check that the pump is stopped if the patient ever enters the Alarming region:

\[
A\bigotimes(\text{samplebuffer} < \text{alarm\_thresh} \Rightarrow A\bigotimes(\text{PCA.Rstopped} \lor \text{PCA.Bstopped})).
\]

We consider this property to be a sanity check rather than a safety requirement, because wrong parameters of the model – for example, too short bolus duration or too high drug absorption rate – can make the system appear safe (that is, SpO\(_2\) level never goes too low), but it would be safe for the wrong reason. All sanity checks were passed by the UPPAAL model described above when no dropped messages are allowed. Clearly, property (18) does not hold if messages can be dropped.

Finally, we turn to checking the main safety property. With the threshold for the Critical region set to 70%, the property

\[
A\bigotimes(\text{samplebuffer} \geq \text{critical})
\]

is satisfied if the stop message cannot be dropped. However, if losing messages is enabled in the network automaton, the property is not satisfied.

**V. Failures and Fail-Safe PCA system**

We have seen in Section IV that the system does not satisfy its safety property if network messages can be lost. Also, if any of the delays on the left side of the safety condition (4) is significantly increased, the condition would not be satisfied. Increase in delays can be caused by a component or network failure, effectively resulting in an open-loop system.

To provide safety assurance, realistic scenarios have to be taken into account, where network failures occur or PO accidentally gets detached from the patient. Thus, the Supervisor’s control algorithm and PCA pump’s designs have to guarantee the system’s open-loop safety. For the case study, this denotes that even if the Supervisor does not receive the right values for HR and/or SpO\(_2\), or the pump does not receive the command to disable the drug flow, the system design has to ensure that the patient would not enter the Critical region. In addition, open-loop safety implies that even if the patient keeps pressing the button, no drug flow would be enabled if there is a chance that the amount of infused drug could harm the patient.

One way to design the system that complies with the open-loop safety requirement is to make changes in the closed-loop system from Figure 2. In the modified system, the pump only receives activation commands from the Supervisor (not the Patient), along with the duration of the drug flow (which does not have to be fixed). When the Patient presses the button, if the Supervisor is informed about the current HR and SpO\(_2\) levels it is able to determine a duration of the pump’s activation (\(\Delta t_{\text{safe}}\)) that guarantees the patient’s safety. Additional condition is that the Supervisor disregards if the button is pressed during \(t_{\text{del}}\) units of time after the pump has stopped, where \(t_{\text{del}}\) accounts for all the delays in the loop:

\[
t_{\text{del}} = t_{\text{Podel}} + t_{\text{Sup}} + t_{\text{net}} + t_{\text{Pump}} + t_{\text{PCA2P}}
\]

The imposed condition ensures that the last drug delivery has taken full effect before the next ‘button pressed’ command is sent to the Supervisor. It implies that the drug level will reach its local maximum (i.e., HR and SpO\(_2\) level will reach local minima) before HR and SpO\(_2\) measurements sent to the Supervisor are obtained. All constituents of \(t_{\text{del}}\), except the network delay, can be calculated as previously described. However, we assume that if the message is delivered, the underlying real-time network provides a guaranteed bound on the network delay, as described in [7].

To calculate \(\Delta t_{\text{safe}}\) the following parameters are used:

- \(h_{\text{cur}}\) - last received SpO\(_2\) level;
- \(h'_{\text{cur}}\) - the actual SpO\(_2\) when the drug reaches the body;
- \(\Delta h(h_0, \Delta t)\) - the maximal SpO\(_2\) decrease caused by the drug flow of duration \(\Delta t\), if the initial SpO\(_2\) level is \(h_0\).

Now, the safety requirement can be specified as:

\[
h'_{\text{cur}} - \Delta h(h'_{\text{cur}}, \Delta t_{\text{safe}}) \geq H^2_{\text{SpO}_2}.
\]

It is worth noting that although we use SpO\(_2\) measurements to calculate \(\Delta t_{\text{safe}}\), it is also possible to use HR measurements, because the Critical region boundary (i.e., \(H^2_{\text{HR}}\) instead of \(H^2_{\text{SpO}_2}\)) corresponds to the same drug level.

The requirement that after the pump has been stopped the Supervisor would disregard if the button is pressed for \(t_{\text{del}}\)
time units, implies that $h'_{\text{cur}} \geq h_{\text{cur}}$ when the button pressed command is accepted (since \(\text{SpO}_2\) levels increase when the drug level decreases due to the fact that the pump is turned off). Thus, (19) is satisfied when the condition below is valid:

$$\Delta h(h'_{\text{cur}}, \Delta t_{\text{safe}}) \leq h_{\text{cur}} - H_2^{\text{SpO}_2}. \quad (20)$$

To determine $\Delta t_{\text{safe}}$ Theorem 1 can be utilized. In this case, the safety condition (19) is satisfied if:

$$\Delta t_{\text{safe}} \leq \tilde{t}_{\text{safe}} = \frac{1}{||A||} \ln \left( \frac{||H_2^{\text{SpO}_2} - h_{\text{cur}}/\text{SpO}_2\text{gain}}{||C|| \cdot \left( ||\tilde{x}_0|| + \frac{||B_u||}{||A_{\min}||} \right)} \right) \quad (21)$$

Obviously if $h_{\text{cur}} > H_1^{\text{SpO}_2}$ ($H_1^{\text{SpO}_2}$ is the \(\text{SpO}_2\) threshold value for the Alarming region) $\Delta t_{\text{safe}}$ would be greater than $\tilde{t}_{\text{crit}}$. Similarly, if the patient is already in the Alarming region ($h_{\text{cur}} \leq H_1^{\text{SpO}_2}$) then $\Delta t_{\text{safe}} \leq \tilde{t}_{\text{crit}}$.

The system guarantees open-loop safety since, even if some (or all) of the messages are dropped, the patient would never enter the Critical region. The reason is that the pump activation command also contains the duration of the flow drug; if the flow durations are properly computed, the pump stops before the safety requirement is breached.

To guarantee open-loop safety an assumption was made that the Supervisor is able to determine whether received measurements are valid. This is a reasonable assumption because modern POs send an invalid code when valid measurements can not be obtained.\(^4\) Note that in this work we do not consider failures where each component’s behavior differs from the behavior described by its model. For example, if the pump is active longer than requested, or if the PO does not send an invalid code when it is unable to obtain valid measurements.

The presented solution is not the only way to guarantee open-loop safety. An alternative approach, which requires a minimal change to existing PCA infusion pumps, is to have the Supervisor instruct the pump with the maximal amount of drugs that can be delivered. Using a procedure similar to (21) the Supervisor can easily derive the maximal allowed drug dosage. This system is also inherently fail-safe since, in the previous case, the worst consequence of failed network, Supervisor, or PO is no drug infusion.

VI. RELATED WORK

Formal methods have traditionally been used for verification of time-critical and safety-critical embedded systems [20]. For example, model-driven techniques were used for validation (e.g., [21]) and verification (e.g., [22], [23], [24]) of industrial control systems, but their use for medical device certification is quite recent. In this domain, formal techniques have been applied to improve medical device protocols [25] and safety [26]. In addition, in [10] the authors present the use of Extended Finite State Machines for model checking of the Computer Automated Resuscitation A medical device, while in [27] a model-driven development of cardiac pacemakers is described. Nonetheless, although these papers consider relevant clinical applications, in all of them either a simplified patient model was utilized or the patient was not modeled at all.

Continuous monitoring of the blood oxygenation of patients receiving PCA infusions has been done in the past and even commercially implemented. The Alaris 8210 \(\text{SpO}_2\) Module connects to the Alaris 8000 pump controller and adds the ability to pause infusions based on a target \(\text{SpO}_2\). Our approach shows how a similar system could be designed and validated. In particular, while the available commercial system is provided as a tightly-integrated system from a single vendor, our approach could be used to design and validate systems based on devices from multiple sources as long as the timing and other necessary information is available.

VII. CONCLUSION

We have presented a clinically relevant case study of the closed-loop control of a PCA infusion pump. To perform safety analysis of the closed-loop medical system we have used a model-driven approach that combines simulation-based validation of a continuous-time system model in Simulink with formal verification of a more abstract model using timed automata and the UPPAAL tool. The key to keeping the two models consistent has been the derivation of timing parameters of the system from the Simulink model and the use of these constants in the UPPAAL model.

For the case study, we have shown that the system is safe under no failure assumptions. We have identified how to deal with some of the failures that manifest as unbounded delays. The proposed method is based on the well-known notion of timed lease used in fault-tolerant distributed systems. We believe that such a technique can be applied to other tightly integrated medical systems in which fail safe is essential.

The dynamics in the considered case study is relatively simple. This choice is made on purpose, to better present the steps in our approach, including the analysis of a system with uncertain parameters. Given the simple case study, we believe that our approach allows us to construct safety cases for regulatory approval of closed-loop medical systems. With more complicated dynamics, some of the steps become more difficult; in particular, the derivation of timing parameters for patient models with uncertain parameters could require more sophisticated methods and further research may be required.

In this paper, we highlight the need for model-driven safety analysis of the closed-loop medical systems and, on a specific case study, we emphasize potential problems that might occur. However, it will be of interest to extend our approach to more general closed-loop medical scenarios.

APPENDIX A

PROOF OF THEOREM 1

Proof: Consider the LTI system from (7). The system state at time $t_0 + t$ for all $t \geq 0$ can be computed as ([28]):

$$x(t + t_0) = e^{A t} x_0 + \int_{t_0}^{t_0 + t} e^{A(t + t_0 - \tau)} B u(\tau) d\tau.$$  

Therefore, if $u(t) = u_i$ for all $t \geq t_0$, where $u_i$ is a constant vector, the system evolution can be described as:

$$x(t + t_0) - x_0 = (e^{A t} - I_n) x_0 + A^{-1} (e^{A t} - I_n) B u_i \quad (22)$$

\(^4\) Detachment of the probes from the patient’s body or movements of the patient’s hand are the main reasons why POs do not obtain valid values.
where $I_n \in \mathbb{R}^{n \times n}$ is the $n \times n$ identity matrix. Due to the fact that for matrices $P, P_1$ and $Q$ of the appropriate dimensions $||PQ|| \leq ||P|| \cdot ||Q||$ and $||P + Q|| \leq ||P|| + ||Q||$, from (22) it is possible
to provide the following bound:

$$|y(t + t_0) - y(t_0)| = |C(x(t + t_0) - x_0)| \leq ||C|| ||x(t + t_0) - x_0|| \quad (23)$$

Now, consider the term $|e^{At} - I_n|$. From the definition of the exponential function we can obtain:

$$||e^{At} - I_n|| = \left\| \sum_{k=0}^{\infty} \frac{(At)^k}{k!} - I_n \right\| = \left\| \sum_{k=1}^{\infty} \frac{(At)^k}{k!} \right\| \leq \sum_{k=1}^{\infty} \left( \frac{||A||^k}{k!} \right) = \sum_{k=0}^{\infty} \left( \frac{||A||^k}{k!} \right) - 1 = e^{||A||t} - 1 \quad (24)$$

Similarly, it can be shown:

$$||A^{-1}(e^{At} - I_n)|| \leq ||A||^{-1}(e^{||A||t} - 1) \quad (25)$$

From (23)-(25) it is possible to obtain a bound on the change in the system’s output:

$$|y(t + t_0) - y(t_0)| \leq ||C||(e^{||A||t} - 1)||x_0|| + ||A||^{-1}||B||u_1||$$

Therefore, from the previous equation:

$$t \geq \frac{1}{||A||} \ln \left( \frac{|y(t + t_0) - y(t_0)|}{||C|| ||x_0|| + ||B||u_1|| ||A||^{-1}} + 1 \right) \quad (26)$$

for all matrices $A$, $B$ and $C$ that belong to the polyhedrons $\mathbb{R}(A)$, $\mathbb{R}(B)$ and $\mathbb{R}(C)$ respectively.

Eq. (26) provides a lower bound on the value $t$ for which the system output can progress from point $y(t_0)$ to the point $y(t_0 + t)$ for any values of the aforementioned matrices. The bound is a function of the patient parameters (i.e., matrices $A$, $B$ and $C$). The goal is to provide the bound value that is the minimum of the term on the right side of (26) for all matrices $A$, $B$ and $C$ belonging to the corresponding polyhedrons.

The term on the right side of (26) decreases as the values $||A||$, $||C||$, $||x_0||$, $||A||^{-1}$ and $||B||u_1||$ increase. Note that it is not possible to calculate the value for $||A||$ that will minimize the term. Thus, for simplicity we consider two independent problems:

$$\max_{A \in \mathbb{R}(A)} ||A|| \quad \text{and} \quad \max_{A \in \mathbb{R}(A)} ||A||^{-1} \quad (27)$$

As $\max_{A \in \mathbb{R}(A)} ||A||^{-1} = \frac{1}{\min_{A \in \mathbb{R}(A)} ||A||}$, by defining $\hat{A}, \hat{B}, \hat{C}$ as in the Theorem statement, (8) can be obtained.

### Appendix B

**Proof of Theorem 2**

**Proof:** Assume that there exist a non-vertex matrix $\tilde{A}_0 = \arg \max_{A \in \mathbb{R}(A)} ||A||$, and that there does not exist a vertex matrix $A \in \mathbb{R}(A)$ such that $||A|| = ||\tilde{A}_0||$. Since $\tilde{A}_0$ is not a vertex of the bounded polyhedron $\mathbb{R}(A)$, there exists matrix element $\tilde{a}_{ij}$ such that $\tilde{a}_{ij} - \Delta \tilde{a}_{ij} < \tilde{a}_{ij} + \Delta \tilde{a}_{ij}$. Consider $\epsilon = \min(\tilde{a}_{ij} - \Delta \tilde{a}_{ij}, \tilde{a}_{ij} + \Delta \tilde{a}_{ij} - \tilde{a}_{ij})$. Since $\tilde{A}_0$ is not a vertex, then $\epsilon > 0$. We denote by $\tilde{A}_0^\epsilon$ and $\tilde{A}_0^\epsilon$ the matrices whose all elements are equal to the appropriate elements of $\tilde{A}_0$, except the elements from $i^{th}$ row and $j^{th}$ column that are $\tilde{a}_{ij}^\epsilon = \tilde{a}_{ij} - \epsilon$ and $\tilde{a}_{ij}^\epsilon = \tilde{a}_{ij} + \epsilon$. Thus, either $\tilde{a}_{ij}^\epsilon = \tilde{a}_{ij} - \Delta \tilde{a}_{ij}$, in which case we define a new matrix $\tilde{A}_1$ as $\tilde{A}_1 = \tilde{A}_0^\epsilon$, or $\tilde{a}_{ij}^\epsilon = \tilde{a}_{ij} + \Delta \tilde{a}_{ij}$ and we define $\tilde{A}_1 = \tilde{A}_0^\epsilon$.

Since $A_0$ maximizes the norm then $||A_0|| \geq ||A||$ and $||A_0|| \geq ||\tilde{A}_1||$. If any of $||\tilde{A}_1||, ||\tilde{A}_1||$ is strictly less than $||A_0||$ then:

$$||\tilde{A}_1|| > \frac{1}{2} \left(||\tilde{A}_1|| + ||\tilde{A}_1||\right) \quad (27)$$

On the other hand, since $||A||$ is a convex function and $A_0 = \frac{1}{2} (A_1 + A_2^\epsilon)$ it follows that: $||A_0|| \leq \frac{1}{2} \left(||A_1|| + ||A_2^\epsilon||\right)$, contradicting (27). Thus, $||A_0|| = ||A_1|| = ||\hat{A}_1||$.

By repeating the above procedure, we either obtain a contradiction or an array of matrices $\tilde{A}_0, \tilde{A}_1, ..., \tilde{A}_k$ that are solutions to the norm maximization problem. However, since at each iteration we set one of the matrix elements to be on its region boundary, the array can have at most $N$ elements (i.e., $k \leq N$), where $N$ denotes the number of elements of matrices $A \in \mathbb{R}(A)$. Furthermore, the final matrix $\hat{A}_k$ is a vertex of the polyhedron $\mathbb{R}(A)$ which is in contradiction with the initial assumption, implying that there exist an optimal solution $A_{max}$ that is a vertex of the polyhedron $\mathbb{R}(A)$.

### References


Miroslav Pajic is a Postdoctoral Fellow in the Department of Electrical & Systems Engineering at the University of Pennsylvania. He received the Ph.D. and M.S. in Electrical Engineering from the University of Pennsylvania in 2012 and 2010, and the M.S. and Dipl. Ing. degrees from the University of Belgrade, Serbia, in 2007 and 2003, respectively. His research interests include cyber-physical systems, real-time embedded systems, embedded and distributed/networked control systems and high-confidence medical device systems. He received several awards including 2011 ACM SIGBED Frank Anger Memorial Award and the best student paper award at RTAS 2012.

Rahul Mangharam is the Stephen J Angello Chair and Assistant Professor in the Department of Electrical & Systems Engineering and Dept. of Computer & Information Science at the University of Pennsylvania. His interests are in real-time scheduling algorithms for networked embedded systems with applications in automotive systems, medical devices and control networks. He received his BS, MS and Ph.D. in Electrical and Computer Engineering from Carnegie Mellon University in 2000, 2002 and 2008 respectively.

Oleg Sokolsky is a Research Associate Professor of Computer and Information Science at the University of Pennsylvania. His research interests include the application of formal methods to the development of cyber-physical systems, architecture modeling and analysis, specification-based monitoring, as well as software safety certification. He received his Ph.D. in Computer Science from Stony Brook University.

David Arney is a Systems Engineer for the Medical Device Plug and Play Program. He has been working on applying formal methods to medical device software since 2001 and was a scholar in residence at the FDA’s Center for Devices and Radiological Health in the Office of Science and Engineering Laboratories. He was involved in writing the ASTM 2761-09 ICE standard for interoperable medical devices. He started at the MD PhD program in September of 2010, and is currently writing his dissertation for a PhD in computer science under Professor Insup Lee at the University of Pennsylvania.

Julian M. Goldman MD is Medical Director of Biomedical Engineering for Partners HealthCare System, Founding Director of the Program on Medical Device Interoperability at CIMIT (Center for Integration of Medicine and Innovative Technology), and a practicing anesthesiologist at the Massachusetts General Hospital. Dr. Goldman founded the Medical Device "Plug-and-Play" (MD PnP) Interoperability Program in 2004 to promote innovation in patient safety and clinical care by leading the adoption of patient-centric medical device integration. He currently serves in leadership positions in several medical device standardization organizations, including Chair of ISO Technical Committee 121, Chair of the Use Case Working Group of the Continua Health Alliance, and User Vice Chair of ASTM Committee F29.

Insup Lee is Cecilia Fiter Moore Professor of Computer and Information Science and Director of PRECISE Center at the University of Pennsylvania. He also holds a secondary appointment in the Department of Electrical and Systems Engineering. He received the B.S. in Mathematics from the University of North Carolina, Chapel Hill and the Ph.D. in Computer Science from the University of Wisconsin, Madison. His research interests include cyber physical systems (CPS), real-time embedded systems, formal methods and tools, high-confidence medical device systems, and software engineering. The theme of his research activities has been to assure and improve the correctness, safety, and timeliness of life-critical embedded systems. Recently, he has been working in the area of medical cyber physical systems. He has served on many program committees and chaired many international conferences and workshops. He has also served on various steering and advisory committees of technical societies, including CPSWeek, ESWeek, ACM SIGBED, IEEE TC-RTS, RV, ATVA. He has served on the editorial boards of the several scientific journals and is a founding co-Editor-in-Chief of KIIEE Journal of Computing Science and Engineering (JCSE). He was Chair of IEEE Computer Society Technical Committee on Real-Time Systems (2003-2004) and an IEEE CS Distinguished Visitor Speaker (2004-2006). He with his student received the best paper award in RTSS 2003. He was a member of Technical Advisory Group (TAG) of President’s Council of Advisors on Science and Technology (PCAST) Networking and Information Technology (NIT), 2006-2007. He is IEEE fellow and received IEEE TC-RTS Outstanding Technical Achievement and Leadership Award in 2008.