Synthesis of Context-Aware Safety Monitors for Medical Cyber-Physical Systems: A Combined Model and Data Driven Approach

Xugui Zhou, Bulbul Ahmed, James H. Aylor, Philip Asare*, Homa Alemzadeh

Department of Electrical and Computer Engineering, University of Virginia, Charlottesville, VA 22904
`{xz6cz, ba3fw, jha, alemzadeh}@virginia.edu`
*Department of Electrical and Computer Engineering, Bucknell University, Lewisburg, PA 17837
`philip.asare@bucknell.edu`

Abstract. The increasing device complexity, shrinking technology sizes, and shorter time to market have resulted in significant challenges in ensuring the reliability, safety, and security of medical devices. This paper presents a combined model and data driven approach for the synthesis of context-aware safety monitors that can continuously predict and mitigate hazards in medical cyber-physical systems (MCPS). The synthesis process couples systems-theoretic modeling of hazards with formal specifications of safety requirements, particularly Signal Temporal Logic (STL), refined by data from closed-loop system simulations for the prediction of unsafe system contexts. The monitor code is synthesized from the final refined STL and is integrated with the target controller software as a wrapper that only has access to the input-output interface (sensor and actuator values). The monitor performs real-time execution of STL formulas for preemptive detection of unsafe controller commands *before* they are delivered to the actuators and cause harm to patients. We demonstrate the effectiveness of our approach in simulation using a closed-loop artificial pancreas system (APS). The results show a safety monitor developed with this approach demonstrates up to 15.6% increase in average prediction accuracy (F1 score) and 48.1% reduction in false-positive (FP) rate, and a slightly longer (1%) reaction time (time between prediction and actual occurrence of hazard) compared to a baseline.

Keywords: safety· resilience· run-time verification· online monitoring· error detection· hazard analysis· cyber-physical system · medical device.

1 Introduction

Medical Cyber-Physical Systems (MCPS) are increasingly deployed in various safety-critical diagnostic and therapeutic applications. Recent studies have shown the susceptibility of medical devices, such as patient monitors, infusion pumps, implantable pacemakers, and surgical robots to accidental faults or malicious attacks with potential adverse impacts on patients [1–5]. Although leveraging CPS correct by construction techniques like formal methods, model-based design, and automated synthesis can help to improve the resilience of designs, CPS are still vulnerable to residual faults and malicious attacks that can evade the most rigorous design and verification methods and appear during run time. Run-time verification and assurance of safety properties based on formal models of systems have been an active area of research in safety-critical systems [6–9]. However, these approaches often rely on ad-hoc safety properties and do not account for control system interactions and context in the CPS. Further, most of these solutions do not provide the ability for *early detection* of safety property violations, which would help with the *prevention* of hazards.

Great efforts have also been made to improve the security of MCPS using online safety monitoring and anomaly detection, including model-based approaches [10, 11], probabilistic models [12], fuzzy logic-based algorithms [13], and control invariant detection techniques [14]. However, these techniques often detect the safety violations too late, leaving little or no time for recovery and mitigation. Few exceptions, such as several previous works on CPS safety and security [15] [4], rely on complex dynamic models of physical system/environment for online safety verification. However, developing such dynamic models for MCPS is challenging because of the changes in patients' profiles over time and unpredictable variances in the human body affected by eating or doing exercise.

In this paper, we propose a system-resilience design methodology in which context-aware safety monitors can be synthesized from the safety requirements identified through a control-theoretic hazard analysis process, and be integrated with the control software of a target MCPS at design-time to continuously detect and mitigate the delivery of unsafe control commands to the patient at run-time (see Figure 1). The main contributions of the paper are as follows:

- Introducing context awareness as a mechanism for designing monitors that are capable of hazard prediction and mitigation.
- Proposing a framework for synthesis of context-aware monitors that:
  - Provides a formal specification of safety context for hazard prediction and mitigation.
  - Generates formal signal temporal logic (STL) specifications.
  - Leverages machine learning techniques for further quantitative refinement of formal specifications for the monitor.
- Developing a closed-loop simulation system and a fault injection framework for experimental evaluation of monitors in terms of timely and accurate prediction of hazards for a case study of Artificial Pancreas Systems (APS).

2 Context-Aware Safety Monitoring

CPS are designed by the tight integration of cyber components and software with the hardware devices and the physical world. The core of the MCPS are the autonomous controllers that connect the human operators (e.g., physicians, nurses) and cyber networks with the physical components (e.g., patient's body) (Figure 1). The controller's goal is to adapt to the constantly changing and uncertain physical environment and the operator's commands through estimating the current state of the system based on sensor measurements and changing the physical state by sending control commands to the actuators. In such systems,

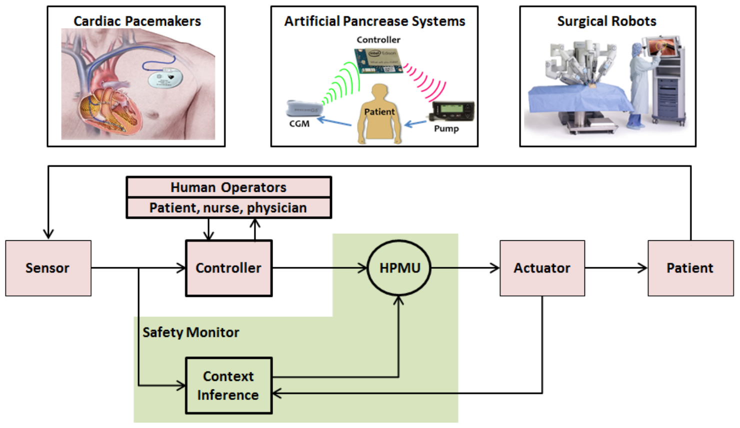


Fig. 1: Generic MCPS and Context-aware Monitor Integrated as a Wrapper

safety hazards and accidents might happen as a result of unsafe commands issued by the controller due to accidental faults and malicious attacks acting on the sensor data, the controller (algorithm, software, hardware), or the actuators.

Our goal is to detect unsafe control commands, regardless of their originating causes, and stop or mitigate them before execution on the actuators to prevent safety hazards. A simple monitoring mechanism might involve checking the values of control commands based on ad-hoc safety rules or medical guidelines [10]. However, such a monitoring system does not take into account the current status of the system and the dynamics of the patient and might incorrectly classify safe commands, leading to a large number of false alarms, or, more importantly, might miss the detection of unsafe control actions before they are delivered to patient. For instance, in a surgical robot, setting fixed constraints on the position and velocity of the robotic instrument end-effectors might lead to a large number of false alarms [4]. There are several examples of adverse events previously reported in the literature [1] [16] where such inadequacies in the controller and monitor design in medical devices (e.g., patient monitors) led to false alarms or missed detection and potential harm to patients.

Safety as an emergent property of CPS is context-dependent and should be controlled by enforcing a set of constraints on the behavior of system and control actions given the current system state [17]. Recent systems-theoretic approaches to safety, such as the Systems-Theoretic Accident Model and Processes (STAMP) [18], try to identify unsafe context-dependent control actions and interactions between system components that will lead to safety hazards. However, attempts at providing formal frameworks for models such as STAMP [19] [20], have still left gaps between the high-level safety requirements identified from hazard analysis and the low-level formal specification of safety properties that can be used for run-time monitoring and safety assurance. We leverage the control-theoretic notion of system context from STAMP accident model and develop a formal framework for context-aware detection and mitigation of hazards. Based on this framework, we develop an approach for synthesizing context-aware monitors that take into account the current system context under which a control action is issued to predict whether that control action is potentially unsafe.

As shown in Figure 1, the proposed monitor can be integrated with the target MCPS controller as a wrapper that only has access to the input-output interface for observing the sensor data and actuator commands of the controller and making context inference. The Hazard Prediction and Mitigation Unit (HPMU) then evaluates whether the action decided by the controller given the inferred context might result in any hazards and stops delivering unsafe commands by issuing corrective actions to mitigate hazards. We **assume** the sensor data received by the controller and the monitor is fault-free or protected using mechanisms such as quickest change detection and automatic response proposed in [21] and only focus on the faults/attacks targeting the controller itself. Also, the monitor is made tamper-proof using protective memories or hardware isolation.

3 Monitor Synthesis

Our overall methodology for the synthesis of context-aware safety monitors starts with the specification of system context driven by aspects of the STAMP accident model, formalization of the context specification using Signal Temporal Logic (STL), and its optimization through learning from system simulation traces. Our combined model and data-driven approach provides a common framework to engineers and clinicians for the specification of safety requirements based on domain knowledge and the automated refinement of safety properties to be checked at run-time based on real or simulated patient data.

3.1 Model of System Dynamics

State Space and Control Actions. We let the dynamics of the MCPS be captured by its generalized state space x, including continuous state variables, possibly their derivatives (capturing the phase space), and any discrete states. We denote the set of all possible values the states can take as X. The MCPS can take control actions, $u = (x_u, v_u(x_u))$, which acts on a subset of the state variables x_u, and has an effect on those variables defined by $v_u(x_u)$, which can denote instantaneous changes in the case of discrete actions and effects on rates of change for continuous states.

Regions of Operation. We assume there are two mutually exclusive regions of the state space or some transformation of it: (i) the hazardous region X_h, which could be further partitioned into regions associated with specific hazards, and (ii) the safe/desirable region X_*.

Unsafe Control Actions. A control action u is unsafe if upon execution of u in state x at time t the system will transit to a state in X_h at a future time $t' > t$.

3.2 Safety Context Specification

The safety context specification (SCS) consists of two parts: (i) the unsafe control action specification used by the context-aware monitor to detect unsafe control actions and predict hazards; (ii) the preemptive mitigation specification that provides at least one mitigation option in the event that the controller has issued an unsafe control action in order to prevent hazards.

Unsafe Control Action Specification. Usually, the controller and monitor only have an estimate of the system state through a subset of variables in the state space. In addition, they may be making decisions based on some transformation of this subset of state variables. This subset and its possible transformation is denoted by $\mu(X)$. More specifically, if $x = (x_1, \ldots, x_n)$, then $\mu(X) = (\mu_1(x),\ \ldots, \mu_m(x))$. To reduce complexity in identifying unsafe control actions, the *contexts* are described as specific ranges of variables in $\mu(X)$, which can be thought of as partitions $\rho(\mu(X))$ of the space $\mu(X)$.

To define the nature of a control action in a particular partition, we assume that the action is applied in the partition until the system transitions into a state outside that partition. The region of operation of the new state outside the partition is then considered, which corresponds to the function $(\rho(\mu(X)), u) \mapsto \{X_*, X_h\}$. An unsafe control action specification (UCAS) is the set, $U^{\rho}_{\neg s}$, of all tuples $(\rho(\mu(X)), u)$ such that $(\rho(\mu(X)), u) \mapsto X_h$. This definition explicitly identifies an unsafe control action. The unsafe control action specification is generated using the following steps:

1. Define the set of accidents (A) and hazards (H) of interest for the system.
2. Identify the observable variables of interest related to the hazards and define the two regions $\{X_*, X_h\}$ based on them.
3. Decide on the various variables (and possible transformations) of interest $\mu(X)$ to use for partitions and define the partitions $\rho(\mu(X))$ of interest as completely as possible. The thresholds for all variables that define the partition need not be known.
4. List all the combinations of $\rho(\mu(X))$ and $u \in U$.
5. Identify the combinations that might result in transitions to hazardous region X_h, and add them into set $U^{\rho}_{\neg s}$.

Steps 1 and 2 require medical domain knowledge and input from clinicians. Steps 1 through 3 need to be defined manually by an analyst. Step 4 can be automated based on definitions in steps 2 and 3 [20]. Step 5 can be done manually, but can also be automated using simulation [22].

Preemptive Mitigation Specification. The preemptive mitigation specification is a set U^{ρ}_{s} of tuples of the form $(\rho(\mu(X)), \boldsymbol{u}^{\rho})$, where $\boldsymbol{u}^{\rho}$, is the set of safe control actions in the the context $\rho(\mu(X))$ that result in transition to X_*. Since the preemptive mitigation specification only applies in contexts where an unsafe control action can be issued, the partitions considered are the same partitions in $U^{\rho}_{\neg s}$, but here, $u \in \boldsymbol{u}^{\rho}$ is such that $(\rho(\mu(X)), u) \mapsto X_*$. The preemptive mitigation specification is generated using the following steps:

1. Generate the unsafe control action specification, $U^{\rho}_{\neg s}$ as above.
2. For each context in $U^{\rho}_{\neg s}$, find all control actions u such that $(\rho(\mu(X)), u) \mapsto X_*$ and set these to $\boldsymbol{u}^{\rho}$ for that context.

Step 2 may be done manually or can be learned from simulations as well.

3.3 Formalization of SCS in Temporal Logic

For online monitoring of safety requirements and detection of unsafe control actions, we need to describe their specification using a machine-checkable language/logic that can express complex policies in MCPS. Signal Temporal Logic (STL) is a formal specification language that is often used for rigorous specification and run-time verification of requirements in CPS. Although there has been considerable interest in the use of STL for specification based monitoring, most previous works relied on the specification of ad-hoc rules or clinical guidelines using STL [23]. This paper is the first attempt towards converting the high-level safety properties generated using a control-theoretic accident model into STL formalism and synthesizing the generated STL formulas as an online context-aware monitor for MCPS.

Conversion of SCS to STL. We use the bounded-time variant of STL, where all temporal operators are associated with lower and upper time-bounds. We refer to [23] for a more detail description of STL. Since we want our STL formula to ensure safety, we would like the formula to evaluate to true as long as the unsafe control action is not issued in the context where this would lead to a hazard. The STL formula for a specific context, $(\rho(\mu(X)), u)$, such that $(\rho(\mu(X)), u) \mapsto X_h$, has the form:

$$G_{[t_0,t_e]}(\varphi_1(\mu_1(x)) \wedge \ldots \wedge \varphi_m(\mu_m(x)) \implies \neg u) \tag{1}$$

where G is the globally operator that ensures the formula holds always during time window $[t_0, t_e]$, representing the initial time and end time when we run the control system, and $(\varphi_1(\mu_1(x)) \wedge \ldots \wedge \varphi_m(\mu_m(x)))$ represent the partition $\rho(\mu(X))$, where each $\varphi_i(\mu_i(x))$ can take the form $\mu_i(x)\{<, \leq\}\bar{\beta}_i$, $\mu_i(x)\{>, \geq\}\underline{\beta}_i$, $\underline{\beta}_i\{<, \leq\}\mu_i(x) \wedge \mu_i(x)\{<, \leq\}\bar{\beta}_i$, or $\mu_i(x)\{<, \leq\}\underline{\beta}_i \vee \mu_i(x)\{>, \geq\}\bar{\beta}_i$ for continuous variables, in which $\bar{\beta}_i$ or $\underline{\beta}_i$ defines the boundary of the partition in that dimension $\rho(\mu_i(x))$, or the form $(\mu_i(x) = \alpha_1) \vee \ldots \vee (\mu_i(x) = \alpha_p)$ for discrete variables, which defines a specific state or set of states.

Similarly, STL formula for mitigation specification $(\rho(\mu(X)), u) \mapsto X_*$ is

$$G_{[t_0,t_e]}((F_{[0,t_s]}(u))\mathcal{S}(\varphi_1(\mu_1(x)) \wedge \ldots \wedge \varphi_m(\mu_m(x)))) \tag{2}$$

where F is the eventually operator indicating $u \in u^\rho$ should be taken within period t_s since (denoted by S operator) the system enters context $(\varphi_1(\mu_1(x)) \wedge \ldots \wedge \varphi_m(\mu_m(x)))$. This should hold globally during $[t_0, t_e]$.

The time parameter t_s specifies the requirements for the latest possible time mitigation action should be initiated after a context with potential unsafe control action is detected to prevent hazards. This time is highly dependent on many factors, including the specific value of the state, x, when the system enters the context $\rho(\mu_m(x))$, and the nature of the various safe control actions $u \in \boldsymbol{u}^\rho$. The specifics of determining this time requirement, in general, is beyond the scope of this paper. The time between activation of a fault in the system and the occurrence of a hazard (defined as Time-to-Hazard in Section 4.2) can provide an upper bound for specifying this time requirement.

Optimization of STL Formulas. In some cases, although the values of some of the boundary variables $\bar{\beta}_i$ or $\underline{\beta}_i$ in the STL formulas are known (e.g., based on medical guidelines or past experience), the appropriate values for some others might not be easily known. In such cases, the partition boundaries can be learned from actual or simulated data from the system using machine learning methods [24] [25]. The values of these boundaries have implications for the amount of time between the prediction of a hazard and its actual occurrence (defined as Reaction Time in Section 4.2) and successful hazard mitigation. Hence the learning problem can be designed to maximize reaction time while keeping the boundaries meaningful. In order to learn these boundaries, we would need data from both safe and hazardous contexts of interest. In the absence of such data, learning from positive examples (non-anomalous data) is an approach proposed by previous works [24] [25], which we adopt for our case study in Section 4.1.

4 Case Study of Artificial Pancreas System (APS)

To evaluate the effectiveness of our approach, we applied our methodology to the case of developing a run-time monitor for an Artificial Pancreas System (APS). This system is responsible for regulating Blood Glucose (BG) dynamics by monitoring BG concentration in the patient's body through sensor data collected from a Continuous Glucose Monitor (CGM) and providing insulin at the right rate to the patient through a pump. The control software estimates the current patient status (e.g., glucose value, insulin on board (IOB)) and calculates the next recommended insulin rate value to be delivered to the patient (Figure 2a).

The particular APS system we considered here is OpenAPS [26]. To evaluate the effect of the system on patients through simulation, we developed a closed-loop system by integrating the OpenAPS control software with the Glucosym patient simulator [27] (Figure 2b). This simulator contains patient models derived from data collected from 10 actual adult patients with Type I diabetes mellitus aged 42.5 ± 11.5 years, with their glucose dynamics predicted using Bergman & Sherwin model [28].

4.1 Monitor Synthesis for APS

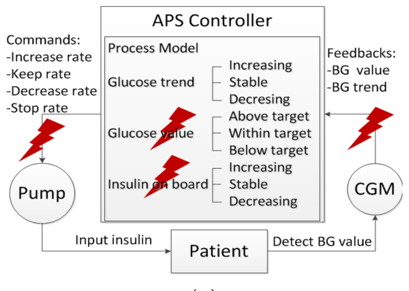
We first identified the set of accidents and the safety hazards that might happen due to unsafe control actions issued by the APS controller.

Accidents: In Type I diabetes, which the APS is designed for, there are two main accidents that we are concerned about:

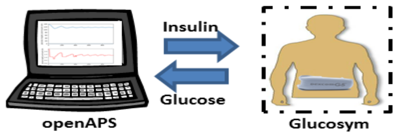
- **A1:** Complications from hypoglycemia (BG level too low), including death.
- **A2:** Complications from hyperglycemia (BG level too high), including morbidities such as retinopathy and in extreme cases, death.

Hazards: The set of system states under the control of the APS that together with the other conditions might lead to accidents include:

- **H1:** Too much insulin is infused, which will reduce the BG and might lead to **A1**.
- **H2:** Too little insulin is infused, which causes the BG to increase and could lead to **A2**.



(a)



(b)

Algorithm 1: Mitigation algorithms

```
initialization;
while not end of simulation do
    Read current BG value;
    Read output of monitor and controller;
    if Alerts for extra insulin input then
        $mitigation\_H1 = True$;
    else if Alerts for insufficient insulin input then
        $mitigation\_H2 = True$;
    if $mitigation\_H1$ then
        $insulin = 0$;
    else if $mitigation\_H2$ then
        $insulin += f(IoB)$;
    if $insulin' < 0$ or $BG > BGT$ then
        mitigation_H1 = False;
    else if $insulin' > 0 or BG < BGT$ then
        mitigation_H2 = False;
end
```

(c)

Fig. 2: (a) Typical APS Control Structure; (b) Closed Loop System with OpenAPS Controller and GlucoSym; (c) Example Mitigation Algorithm.

To better describe these hazardous states and to be able to annotate them in the collected data automatically, we adopted the concept of Risk Index (RI) [29,30] that measures the risk of hyper- or hypo-glycemia based on estimated BG levels.

Following steps in the previous section, the formalized safety context specifications for APS is shown in Table 1 as a set of rules to be checked by the monitor. For example, the first row is the formal representation of UCAS, $(\rho(\mu(X)) = (\text{BG}>\text{HBGT, BG'}>0, \text{IOB'}<0, \text{IOB}<\beta_1), u = u_1) \mapsto X_{h=H2}$, requiring that under such system context/partition $\rho(\mu(X))$, the unsafe control action u_1 (decrease insulin) should not be issued at anytime during $[t_0, t_e]$. Otherwise, an H2 hazard happens. The β_i are the thresholds of IOB in different system contexts. Here we used a temporal logic extraction and learning tool called TeLEx [33] for learning the IOB thresholds from the non-anomalous simulation data (positive examples) collected from the non-faulty controller. We had to learn the

Table 1: Safety Context Specification for APS described in STL formalism

Rule No.	**Description**	**Hazard** if violated
1	$G_{[t_0,t_e]}((\text{BG}>\text{HBGT}\wedge \text{BG'}>0)\wedge (\text{IOB'}<0\wedge \text{IOB}<\beta_1)\Rightarrow \neg\, u_1)$	H2
2	$G_{[t_0,t_e]}((\text{BG}>\text{HBGT}\wedge \text{BG'}>0)\wedge (\text{IOB'}=0\wedge \text{IOB}<\beta_2)\Rightarrow \neg\, u_1)$	H2
3	$G_{[t_0,t_e]}((\text{BG}>\text{HBGT}\wedge \text{BG'}<0)\wedge (\text{IOB'}>0\wedge \text{IOB}<\beta_3)\Rightarrow \neg\, u_1)$	H2
4	$G_{[t_0,t_e]}((\text{BG}>\text{HBGT}\wedge \text{BG'}<0)\wedge (\text{IOB'}<0\wedge \text{IOB}<\beta_4)\Rightarrow \neg\, u_1)$	H2
5	$G_{[t_0,t_e]}((\text{BG}>\text{HBGT}\wedge \text{BG'}<0)\wedge (\text{IOB'}=0\wedge \text{IOB}<\beta_5)\Rightarrow \neg\, u_1)$	H2
6	$G_{[t_0,t_e]}((\text{BG}<\text{LBGT}\wedge \text{BG'}<0)\wedge (\text{IOB'}>0\wedge \text{IOB}>\beta_6)\Rightarrow \neg\, u_2)$	H1
7	$G_{[t_0,t_e]}((\text{BG}<\text{LBGT}\wedge \text{BG'}<0)\wedge (\text{IOB'}<0\wedge \text{IOB}>\beta_7)\Rightarrow \neg\, u_2)$	H1
8	$G_{[t_0,t_e]}((\text{BG}<\text{LBGT}\wedge \text{BG'}<0)\wedge (\text{IOB'}=0\wedge \text{IOB}>\beta_8)\Rightarrow \neg\, u_2)$	H1
9	$G_{[t_0,t_e]}((\text{BG}>\text{HBGT}\wedge \text{IOB}<\beta_9)\Rightarrow \neg\, u_3)$	H2
10	$G_{[t_0,t_e]}((\text{BG}<\beta_{21})\Rightarrow u_3)$	H1
11	$G_{[t_0,t_e]}((\text{BG}>\text{HBGT}\wedge \text{BG'}>0)\wedge (\text{IOB'}<=0\wedge \text{IOB}<\beta_{10}))\Rightarrow \neg\, u_4)$	H2
12	$G_{[t_0,t_e]}((\text{BG}<\text{LBGT}\wedge \text{BG'}<0)\wedge (\text{IOB'}>=0\wedge \text{IOB}>\beta_{11})\Rightarrow \neg\, u_4)$	H1

$^*t_0, t_e=$ start time and end time of the simulation

$*HBGT = 95mg/dL, LBGT = 160mg/dL$ [31] [32]

$*\ u_{1,2,3,4}$ =decrease_insulin, increase_insulin, zero_insulin, keep_insulin boundaries β_i for IOB since this variable is critical for hazard prevention but, unlike the blood glucose (BG) value, has no established boundaries in the literature. We learned these boundaries from positive examples (e.g., non-anomalous data) since this is the data we could obtain from simulations of the unmodified OpenAPS behavior and nominal behavior of the GlucoSym, which are the most realistic and hence the most reliable to learn from.

The learning approach involves first identifying the complementary partition to the one in the formula with the unknown boundary. We assume that taking the action, u, is safe in this complementary partition. For example, assume $\varphi_m(\mu_m(x))$ has an unknown boundary in the formula $G_{[t_0,t_e]}(\varphi_1(\mu_1(x)) \wedge \ldots \wedge \varphi_m(\mu_m(x)) \implies \neg u)$ and that $\varphi_m(\mu_m(x)) = \mu_m(x) < \beta_m$. We learn the threshold β_m from the positive examples such that we ensure $G_{[t_0,t_e]}(\varphi_1(\mu_1(x)) \wedge \ldots \wedge \varphi_m{}'(\mu_m(x)) \implies u)$, where $\varphi_m{}'(\mu_m(x)) = \mu_m(x) \geq \beta_m$ is the complementary partition. Our experiments show that although this approach may create false positives for $\varphi_m(\mu_m(x)) = \mu_m(x) < \beta_m$, it will not create any false negatives.

Preemptive Mitigation. When the monitor detects any unsafe control actions issued by the controller, it will mitigate the potential hazards by correcting the command and delivering a new command ($u \in u^{\rho}$) to the actuator. For example, it can decrease the insulin when it is more than needed, or add suitable insulin when the provided command is insufficient. Designing a mitigation mechanism that can prevent as many hazards as possible while not causing any new hazards is a challenging task. Here we implemented one possible mitigation algorithm to prevent upcoming hazards in APS, as shown in Figure 2c. Further investigation of mitigation algorithms based on formal specification and learning from simulation data is beyond the scope of this paper.

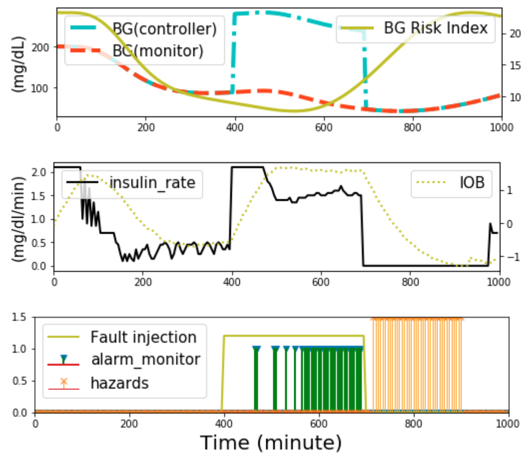
4.2 Experimental Evaluation

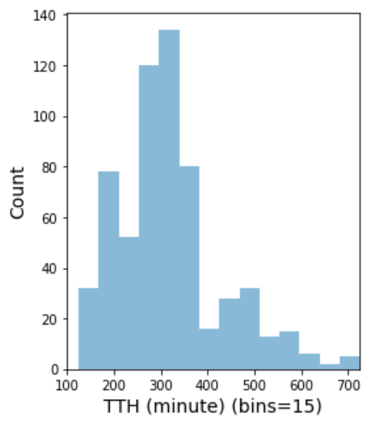
We evaluated the impact of context-awareness in safety monitoring by comparing the performance (accurate and timely prediction of hazards) of our context-aware monitor to a baseline monitor designed based on the medical literature and without considering the notion of safety context. We considered a context-aware monitor without any specific thresholds (CAWOT) and with the thresholds learned from simulation data (CAWT). We used the data authenticity monitor proposed in [10] and described in Table 2 as the baseline.

Table 2: Rules of Baseline monitor

No.	Description
1	$\phi 1 = \Box(BG > 70) \wedge (BG < 180))$
2	$\phi 2 = \Box((\Delta BG > -5) \wedge (\Delta BG < 3))$
3	$\phi 3 = ((BG < \lambda_{10}) \Rightarrow \Diamond_{[0,\alpha]}(BG > \lambda_{10}))$
4	$\phi 4 = ((BG > \lambda_{90}) \Rightarrow \Diamond_{[0,\alpha]}(BG < \lambda_{90}))$

Patient Simulations. We ran the OpenAPS control system integrated with the Glucosym simulator on a virtual machine running Ubuntu 14.04 LTS. Each step/iteration in the simulation, representing 5 minutes in the real OpenAPS control system, took 3-5 seconds to run. Each experiment had the patient interacting with OpenAPS for 200 iterations (1000 minutes or about 16 hours). Simulations began with the patient at an initial glucose value between 80 and 200 mg/dl and had no meals or exercise, mimicking the patient potentially taking a meal at night, going to sleep, and having a meal after our simulation





(a) (b)

Fig. 3: (a) Hazard detection (b) Time to Hazard (TTH) Distribution

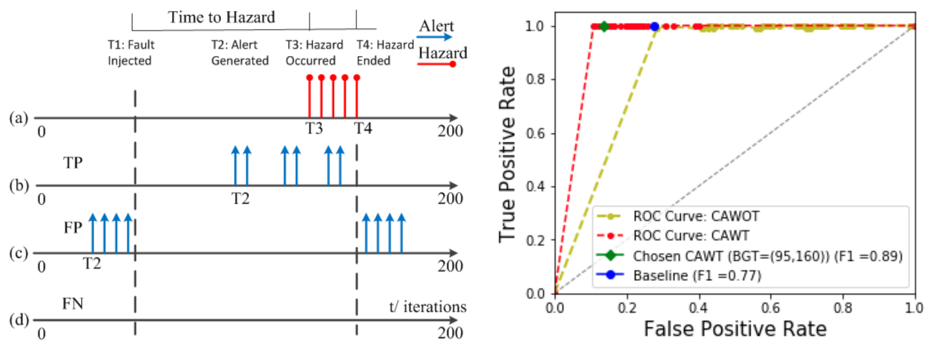
period the next day. We explored seven different initial glucose values for each patient. To account for some interpatient variability, each system version (without a safety monitor and with the three different safety monitors) was evaluated using three different patient profiles. The seven remaining patient profiles were used for learning the thresholds to synthesize the monitors.

To label data points as normal or hazardous in our simulation, we exploited the notion of the Risk Index (RI) [29,30] that captures both the glucose variability and its associated risks for hypo- and hyperglycemia. We marked that data point as hazardous if the potential risk of low- or high-glucose crosses a certain threshold (5 and 9) and keeps increasing. An example is shown in Figure 3a.

Fault Injection to Induce Hazards. To create situations that hazards occur during operation of the OpenAPS and to assess its performance both with and without safety monitoring, we developed a software fault injection framework for emulating faults and attacks targeting the controller that impact the sensor data (glucose values) and control commands (insulin rates). Our fault model considers hardware errors such as memory failures or poor interface connections, but it can also capture the consequence of software attacks on APS systems [3] [14]. In particular, we injected the following faults into the controller:

– Stop the refresh of the glucose value or insulin output, emulating poor connection, memory lock, or DoS attacks.
– Double/half the glucose value or insulin output, to simulate a single bit-flip error at the exponent section of the relative float memory register.
– Decrease/increase the glucose value or insulin output by $2^n (n \in [0, 1, 2, ..., 8])$, while keeping them inside the acceptable ranges, to simulate stealthy attacks.

The above results in 24 different fault scenarios. Each patient-initial glucose combination was tested with each of these scenarios. We assumed that a fault was transient (i.e., it appeared at some point in time, persisted for a particular duration, and disappeared afterward) and only occurred once per simulation. For each scenario, we randomly choose from 8 different start time and duration to inject the fault. The combination of these parameters resulted in a total of 4032



(a) (b)

Fig. 4: (a) Metrics for Hazard Detection Coverage; (b) ROC Curves for Monitors

simulations with faults. In addition, we ran 21 fault-free simulations (3 patients, each with seven initial values) to evaluate the OpenAPS system with no monitor, with the baseline monitor, and with the two context-aware monitors.

Fault Response of OpenAPS Without Safety Monitor. The resilience of the OpenAPS system without a monitor in the presence of faults was measured using **Time-to-Hazard (TTH)** metric. Time-to-Hazard represents the time between activation of a fault in the system until the occurrence of a safety hazard (Figure 4a). This is an important metric indicating the amount of time budget we have for the prediction and mitigation of safety hazards, if they are caused by any of the faults. We analyzed the distribution of this metric (Figure 3b) to help with the specification of time requirements for prediction and mitigation. It should be noted that our safety monitors are designed to detect unsafe control actions and predict hazards and not faults.

Experimental results show that out of 4032 simulated faults, only 613 (15.2%) led to hazards. Specifically, not refreshing sensor data or output commands has such a limited influence on the system that cannot lead to any unsafe state. However, attacks or hardware errors that directly change the value of insulin rate or blood glucose can potentially cause hazards. In addition, our results show that faulty insulin rate values delivered to the pump have a more adverse impact than the faulty glucose values that are potentially masked by the controller.

Safety Monitor Performance. The system performance with the different safety monitors was evaluated using the following metrics:

- ***Detection Coverage*** represents the percentage of potential hazards that are detected by the monitor. We measure the detection coverage using the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN), as defined in Figure 4, as well as using average micro F1 score. Figure 4a(a) shows a scenario that a hazard happens at time T3 and ends at T4 after a fault was injected at T1. If an alert is raised by the monitor at time T2, it is considered a TP (Figure 4a(b)) if T2 $\in$ [T1, T4], a FP otherwise (Figure 4a(c)), and a FN if no alert is generated (Figure 4a(d)). On the other hand, if no hazards happen in the whole simulation and an alert is generated, it is considered as an FP and otherwise a TN.
- ***Reaction Time*** is the time difference between a monitor alert and the occurrence of a hazard, which is the maximum time we have for taking any

mitigation action before the hazard happens. Longer reaction times indicate earlier detection and a higher chance of timely and successful mitigation.

Coverage Analysis: Figure 4b shows the Receiver Operating Characteristic (ROC) curves for the context-aware monitors with (CAWT) and without learning thresholds (CAWOT) using different BG targets. The baseline monitor and a CAWT monitor with a specific BG Target (LBGT of 95mg/dl and HBGT of 160mg/dl) are shown as points on the ROC curves. For most of the target values, the CAWT monitor has a better performance than the baseline (blue dot) and the CAWOT monitor (yellow line). A more detailed summary of the results is shown in Table 3. We see that for the same number of simulations (4032), the proposed CAWT monitor generates a much fewer number of alarms than the baseline monitor (1076 vs. 3007), leading to about half reduction in false alarm rates while keeping the F1 score high (0.89). For the fault-free experiments, it is also the case that overall the CAWT monitor performs better than the baseline. The purpose of presenting the CAWOT monitor is to emphasize the importance of the learning process. Without learning the thresholds for glucose values and IOB, the CAWOT monitor exhibited the highest FPR among these monitors. To sum up, the CAWT monitor outperforms the baseline monitor with a 15.6% increase in average F1 score and 48.1% reduction in the FPR for faulty cases, and with a 16.3% increase in average F1 score for fault-free cases.

Time Analysis: Considering the low F1 score of the CAWOT monitor, we just present the reaction time of the Baseline and CAWT monitor in Table 3. We see that the CAWT monitor can detect unsafe control commands before hazard occurrence for nearly 3.5 hours on average (slightly longer than baseline) with a much less (58.7%) standard deviation than the baseline monitor, representing a more stable performance on ensuring safe reaction time for the patients. We should emphasize that the human body is a large lag system that usually takes hours to digest food and for the insulin to bring BG value back to normal from a severe situation. Therefore, it makes sense for APS to have the reaction time measured in the order of hours (instead of seconds or minutes in other CPS).

Preemptive Mitigation. When running the simulations with the CAWT and mitigation algorithm shown in Figure 2c, only 110 out of the 613 detected hazards still occurred (recovery rate of 82.1% in Table 3) and no new hazards were introduced into the system even for FP cases. Though there were still 110 hazards that we could not prevent, the average reaction time was extended for 106.2 minutes. In comparison, the recovery rate of the baseline monitor with the same

Table 3: Monitor Performance in Faulty and Fault-free Scenarios

Fault Scenarios	**Faulty**			**Fault-free**		
No. Simulations	4032			21		
No. Hazards (%)	613 (15.2%)			0		
Monitors	**Baseline**	**CAWOT**	**CAWT**	**Baseline**	**CAWOT**	**CAWT**
No. Alerts (%)	1551 (38.5%)	3007 (74.6%)	**1076 (26.7%)**	3 (14.3%)	13 (61.9%)	**0**
True Positive Rate	1	1	1	No Positive Case		
False Positive Rate	0.27	0.70	**0.14**	0.14	0.62	**0**
Micro F1 Score	0.77	0.41	**0.89**	0.86	0.38	**1**
Recovery Rate	49.8%		**82.1%**			
Reaction Time Avg±Std(min)	205.1±176.4		**207.2±72.8**			

$*HBGT = 95mg/dL, LBGT = 160mg/dL$ [31] [32]

mitigation algorithm was 49.8%, and it extended the average reaction time by 19.9 minutes. Also, in 4 cases a hazard happened 5 minutes earlier than when no mitigation strategy was used for the baseline monitor, which again demonstrates the disadvantages of not having awareness of the context.

5 Conclusion

This paper presented context-aware safety monitors that can detect, prevent, and mitigate hazards in MCPS along with a formal framework for the specification of safety context and an approach for synthesis of monitors. We developed a closed-loop APS simulation system as a case study to evaluate the proposed methods. Experimental results showed that our monitor outperforms a baseline monitor designed using medical guidelines in accurate and timely prediction of hazards and has stable performance in ensuring sufficient reaction time and mitigating hazards. Future work will focus on evaluating the applicability of this approach on a broader range of MCPS using a larger population of patients.

Code Availability

The code for the closed-loop APS system and the safety monitor is available at: `https://github.com/UVA-DSA/openAPS_GlucoSym_closed_loop_system`.

Acknowledgment

This work was supported by a grant from the U.S. National Science Foundation (No. 1748737) and a research innovation award from the University of Virginia.

References

1. H. Alemzadeh, R. K. Iyer, and Z. Kalbarczyk et al., "Analysis of safety-critical computer failures in medical devices," *IEEE Security& Privacy*, vol. 11, 2013.
2. D. Halperin, T. S. Heydt-Benjamin, B. Ransford, S. S. Clark, B. Defend, W. Morgan, K. Fu, T. Kohno, and W. H. Maisel, "Pacemakers and implantable cardiac defibrillators: Software radio attacks and zero-power defenses," in *2008 IEEE Symposium on Security and Privacy (sp 2008)*. IEEE, 2008, pp. 129–142.
3. C. Li, A. Raghunathan, and N. K. Jha, "Hijacking an insulin pump: Security attacks and defenses for a diabetes therapy system," in *2011 IEEE 13th International Conference on e-Health Networking, Applications and Services*. IEEE, 2011.
4. H. Alemzadeh, D. Chen, and X. Li et al., "Targeted attacks on teleoperated surgical robots: Dynamic model-based detection and mitigation," in *2016 46th Annual IEEE/IFIP International Conference on DSN)*. IEEE, 2016, pp. 395–406.
5. H. Alemzadeh, C. Di Martino, and Z. Jin et al., "Towards resiliency in embedded medical monitoring devices," in *IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN 2012)*. IEEE, 2012, pp. 1–6.
6. I. Lee, S. Kannan, M. Kim, O. Sokolsky, and M. Viswanathan, "Runtime assurance based on formal specifications," *Departmental Papers (CIS)*, p. 294, 1999.
7. G. Eakman, C. Pit-Claudel, I. Lee, and O. Sokolsky, "Correct-by-construction implementation of runtime monitors using stepwise refinement," in *SETTA 2018, Beijing*, vol. 10998. Springer, 2018, p. 31.
8. Falcone, YliÃĺs, F. Jean-Claude, and M. Laurent, "What can you verify and enforce at runtime?" *International Journal on Software Tools for Technology Transfer*, vol. 14, no. 3, pp. 349–382, 2012.

9. J. V. Deshmukh, A. Donzé, and S. Ghosh et al., "Robust online monitoring of signal temporal logic," vol. 51, no. 1, pp. 5–30, 2017.
10. W. Young, J. Corbett, and M. S. Gerber et al., "Damon: A data authenticity monitoring system for diabetes management," in *2018 IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation (IoTDI)*, 2018.
11. S. Pinisetty, R. Partha S, and S. Steven et al., "Runtime enforcement of cyber-physical systems," *EMSOFT*, vol. 16, no. 5, 2017.
12. A. Rao, N. Carreon, R. Lysecky, and J. Rozenblit, "Probabilistic threat detection for risk management in cyber-physical medical systems," *IEEE Software*, vol. 35, no. 1, pp. 38–43, 2017.
13. C. C. Oliveira and J. M. da Silva, "A fuzzy logic approach for highly dependable medical wearable systems," in *IMSTW*. IEEE, 2015, pp. 1–5.
14. "ARTINALI: dynamic invariant detection for cyber-physical system security."
15. M. Althoff and J. M. Dolan, "Online verification of automated road vehicles using reachability analysis," *IEEE Transactions on Robotics*, vol. 30, no. 4, Aug 2014.
16. A. Roederer, J. Dimartino, J. Gutsche, M. Mullen-Fortino, S. Shah, C. W. Hanson, and I. Lee, "Clinician-in-the-loop annotation of icu bedside alarm data," in *2016 IEEE first international conference on connected health: applications, systems and engineering technologies (CHASE)*. IEEE, 2016, pp. 229–237.
17. N. Leveson, *Engineering a safer world: Systems thinking applied to safety*. MIT press, 2011.
18. N. Leveson and J. Thomas, "An stpa primer," *Cambridge, MA*, 2013.
19. P. Asare, J. Lach, and J. A. Stankovic, "Fstpa-i: A formal approach to hazard identification via system theoretic process analysis," in *ICCPS*, April 2013.
20. J. Thomas, "Extending and automating a systems-theoretic hazard analysis for requirements generation and analysis," *Technical Report SAND2012-4080*, 2012.
21. A. A. Cárdenas, S. Amin, and Z.-S. Lin et al., "Attacks against process control systems: Risk assessment, detection, and response," in *ASIACCS*, 2011, p. 12.
22. P. Asare, "A Framework for Reasoning about Patient Safety of Emerging Computer-Based Medical Technologies," Ph.D. dissertation, University of Virginia.
23. E. Bartocci, J. Deshmukh, and A. DonzÃľ et al., *Specification-Based Monitoring of Cyber-Physical Systems: A Survey on Theory, Tools and Applications*, 02 2018.
24. B. Zhang and W. Zuo, "Learning from positive and unlabeled examples: A survey," in *2008 International Symposiums on Information Processing*, May 2008.
25. S. Jha and S. A. Seshia, "A theory of formal synthesis via inductive learning," *Acta Informatica*, vol. 54, no. 7, pp. 693–726, Nov 2017.
26. "Openaps reference design," https://openaps.org/reference-design.
27. "Glucosym," https://github.com/Perceptus/GlucoSym.
28. S. Kanderian, S. Weinzimer, and G. Voskanyan et al, "Identification of intraday metabolic profiles during closed-loop glucose control in individuals with type 1 diabetes." *Journal of diabetes science and technology*, vol. 3, no. 5, 2009.
29. W. Clarke and B. Kovatchev, "Statistical tools to analyze continuous glucose monitor data," *Diabetes technology & therapeutics*, vol. 11, no. S1, pp. S–45, 2009.
30. B. P. Kovatchev, "Metrics for glycaemic controlâĂŤfrom hba 1c to continuous glucose monitoring," *Nature Reviews Endocrinology*, vol. 13, no. 7, p. 425, 2017.
31. A. M. Hersh, E. L. Hirshberg, and E. L. Wilson et al., "Lower glucose target is associated with improved 30-day mortality in cardiac and cardiothoracic patients." *Chest*, vol. 154, no. 5, pp. 1044–1051, 2018.
32. A. Association of Clinical Endocrinologists and A. Diabetes Association, "Standards of care in diabetes," *Diabetes Care*, vol. 33, no. 1, pp. 11–61, 2010.
33. "Temporal logic extractor," https://github.com/susmitjha/TeLEX.