Software Engineering Group
Department of Computer Science
Nanjing University
http://seg.nju.edu.cn

Technical Report No. NJU-SEG-2011-IW-001

From Offline Long-Run to Online Short-Run: Exploring A New Approach of Hybrid Systems Model Checking for MDPnP

Tao Li, Qixin Wang, Feng Tan, Lei Bu, Jian-nong Cao

Xue Liu, Yufei Wang , Rong Zheng

Postprint Version. Originally Published in: High Confidence Medical Devices, Software, and Systems, 2011

From Offline Long-Run to Online Short-Run: Exploring A New Approach of Hybrid Systems Model Checking for MDPnP

Tao Li*, Qixin Wang*, Feng Tan*, Lei Bu†, Jian-nong Cao*
Xue Liu‡, Yufei Wang*, Rong Zheng**
* Department of Computing, The Hong Kong Polytechnic University
Email: {cstaoli, csqwang, csftan, csjcao, csyufewang}@comp.polyu.edu.hk
† State Key Laboratory for Novel Software Technology,
Department of Computer Science and Technology, Nanjing University
Email: bulei@nju.edu.cn
‡ School of Computer Science, McGill University
Email: {xue.liu}@mcgill.ca
** Department of Computer Science, University of Houston
Email: rzheng@uh.edu

Abstract

Hybrid systems model checking is a great success in guaranteeing the safety of computerized control cyber-physical systems *(CPS). However, when applying hybrid systems model checking to MDPnP CPS, we encounter two challenges due to the complexity of human body: i) there is no good differential equation based models for many human body parameters; ii) the complexity of human body can easily cause verification state space explosion. In attempt to address the challenges, we propose to alter the traditional approach of* offline *hybrid systems model checking of time-*un*bounded (i.e.,* long-run*) future. Instead, we propose to carry out* online *(real-time) hybrid systems model checking of time-*bounded *(i.e.,* short-run*) future. We carry out a case study on laser tracheotomy MDPnP, which shows the necessity of our proposed approach. We also carry out emulations based on real-world vital sign traces to show the feasibility of our proposed approach.*

1. Introduction

Thanks to the rapid development of embedded systems technology, we now have thousands of kinds of embedded medical devices. So far, these devices are designed for isolated use. However, people have envisioned that by coordinating these devices, we can significantly increase medical treatment safety, capability, and convenience/efficiency. This vision resulted in the launch of the *Medical Device Plug-and-Play* (MDPnP) [1] effort, which aims to enable the safe composition and collaboration of disparate embedded devices in medical contexts. An MDPnP system is a typical *Cyber-Physical System* (CPS) [15]. On the one hand, it involves cyber-world discrete computer logic of various embedded medical devices. On the other hand, it involves physical-world patient-in-the-loop, which is a continuous complex biochemical system.

The prerequisite and top concern of any MDPnP system is safety. In the cyber-world, for safety-critical systems, people commonly carry out model checking [5] *before* the system is put online. Typically, model checking builds an *offline* model (usually with automata) of the system, and then proves that the model can *never* reach unsafe states. Only after passing model checking is the system allowed to run.

This practice is a great success. Today, model checking is an obligatory procedure required by law for many safety-critical computer systems, such as avionics. For verification of the combined cyber and *physical* systems (i.e., CPS), the most promising family of model checking tools are the *hybrid systems* model checking tools [16], which integrate the discrete automata models with the continuous differential equation (and other control theory) models. Today, hybrid systems model checking is mature enough to deal with many computerized control systems, an important category of CPS.

The success of hybrid systems model checking in computerized control CPS naturally piqued curiosity on its application to MDPnP CPS. However, this faces two major

challenges, mainly due to the complexity of human body (the patient).

Challenge 1 : In most cases, there are *no* good offline models to describe the complex biochemical system of human body (the patient) [11]. Even if some vital signs can be modeled offline, the models may not (with some exceptions [10]) fit into existing hybrid systems model checking tools, which are mainly designed to work with linear differential equation based control models.

Challenge 2 : The verification state space can easily explode due to the uncountable combination possibilities of subsystems and parameters. This is a long-time problem for model checking already. But for MDPnP CPS, the problem becomes even more prominent, as it now involves the numerous genes, tissues, organs, and biochemical processes of the patient's human body. These subsystems/parameters can all directly or indirectly affect each other, creating astronomical possibilities of combinations.

In attempt to address the above challenges for MDPnP CPS, we propose to alter the traditional practice of *offline* model checking of hybrid system's behavior in the time-*un*bounded (i.e., *long-run*) future; instead, we shall carry out *online* model checking of hybrid system's behavior in the time-*bounded* (i.e., *short-run*) future.

This approach has following merits that respectively correspond to the aforementioned challenges:

(1) Though many human body parameters are hard to model offline, their online behavior in short-run future is quite describable and predictable. For example, after injecting 1ml of morphine, it is hard to accurately predict the blood oxygen level curve in the next 10 minutes, as it depends on too many factors, such as the patient's gender, age, weight, genes etc. [8][12]. However, at the granularity of a few seconds into the future, the blood oxygen level curve is quite predictable. It cannot plunge from 99% to 59% in just 3 seconds, or show a saw-toothed wave form. Instead, the curve has to be smooth, which can be perfectly predicted with existing tools, such as linear regression.

(2) Because the modeling is conducted online, many parameters' values become fixed numbers: sampled at the moment of modeling. This cuts the need to traverse the domains of these parameters, hence greatly reduces verification state space. The state space is further reduced because we only look into *short-run* future. That is, we only verify *time-bounded* behavior of the system, rather than all possible behavior of the system in the (time-unbounded) future.

Details of the proposed approach is like follows. Given an MDPnP system $\mathcal{S}$, we shall periodically sample the observable state parameters every T seconds. At time instance kT ($k = 0, 1, 2, \ldots$), we shall build a hybrid systems model (referred to as an "online model") of $\mathcal{S}$ with the observed numerical values of state parameters, and verify its safety in the time interval $(kT, (k+1)T]$ (i.e., within only *short-run* future). If the online model is proven safe, the system can run for another T seconds. Otherwise, the system must immediately switch to an application dependant fall-back plan.

Such modeling and verification must be finished within bounded and short time (i.e., in *real-time*), to allow decision making (on whether run the system for another T seconds or switch to fall-back plan) *before* any fault happens. A more detailed and practical design may involve *pipelining*: model and verify the safety of next period before the current verified safe period ends. For simplicity, we are not going to discuss the detailed design of pipelining in this paper.

In the rest of the paper, we carry out a case study on a representative MDPnP system, laser tracheotomy, to demonstrate our proposed approach and show the approach's feasibility. Section 2 demonstrates how to model the laser tracheotomy MDPnP; Section 3 evaluates the approach; Section 4 discusses related work; and Section 5 concludes the paper.

2. Modeling Laser Tracheotomy MDPnP

Laser tracheotomy MDPnP is a representative MDPnP application [10], where interlocking of various medical devices increases the safety of the surgery. Laser tracheotomy MDPnP involves the following entities (see Fig. 1):

Patient : the patient that receives the surgery;

O_2 **Sensor** : the patient's windpipe oxygen level sensor;

SpO_2 **Sensor** : the patient's blood oxygen level sensor;

Ventilator : the medical device that administrates the patient's respirations;

Surgeon : the doctor that conducts the surgery;

Laser Scalpel : the medical device that the surgeon uses to cut the patient's windpipe;

Supervisor : the central computer that connects all medical devices and make decisions to guarantee safety.

The application context is as follows. In the surgery, due to general anesthesia, the patient is paralyzed, hence have to depend on the ventilator to breath. The ventilator has three modes: pumping out (the patient inhales oxygen), pumping

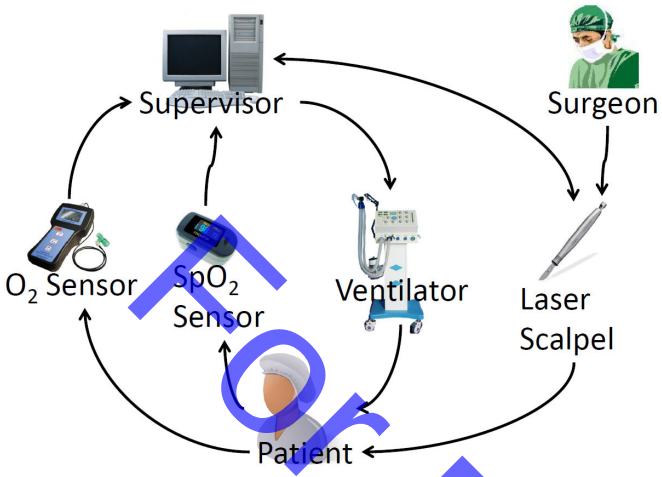

Figure 1. Layout of Laser Tracheotomy MDPnP

in (the patient exhales), and halt (the patient exhales naturally due to chest weight). However, when the laser scalpel is to cut the patient's windpipe, the oxygen level inside the windpipe must be lower than a threshold. Otherwise, the laser may trigger fire. Therefore, before the laser scalpel is allowed to emit laser, the ventilator must have stopped pumping out (oxygen) for a while. On the other hand, the ventilator can neither stop pumping out for too long, or the patient will suffocate due to too low blood oxygen level.

Formally, the behavior of laser tracheotomy MDPnP must comply with the following safety rules:

Safety Rule 1 : when the laser scalpel emits laser, the patient's windpipe oxygen level must not exceed a threshold Θ_{O_2};

Safety Rule 2 : the patient's blood oxygen level never reaches below a threshold Θ_{SpO_2}.

Note the setting of constant thresholds Θ_{O_2} and Θ_{SpO_2} are medical experts' responsibility and are beyond the coverage of this paper.

Also, for better human-computer-interface friendliness, we add a third rule: an optional rule that does not have to be strictly followed but is preferred.

(Optional) Safety Rule 3 : once the supervisor approves the laser scalpel to emit, the approval lasts for at least $T^{min}_{approve}$ seconds, unless the laser scalpel requests to stop emission by itself.

2.1. Offline Modeling

Because the laser tracheotomy MDPnP involves both discrete medical device logic and physical world patient, it is a typical hybrid system. Therefore we try to model laser tracheotomy MDPnP with hybrid automata [9], the modeling language of hybrid systems model checking.

The traditional way of model checking (including hybrid systems model checking) is done offline. That is, the model is built and its time-unbounded (long-run) behavior is verified *before* the system runs. We choose to start with this approach. As a common practice, our offline modeling of laser tracheotomy MDPnP assumes a global variable t representing the global clock: t is initialized to 0 second, and $\dot{t} \equiv 1$.

Intuitively, we intend to start with modeling the patient, the core entity of the laser tracheotomy MDPnP. However, the patient's behavior is directly influenced by the ventilator, which has to be understood first.

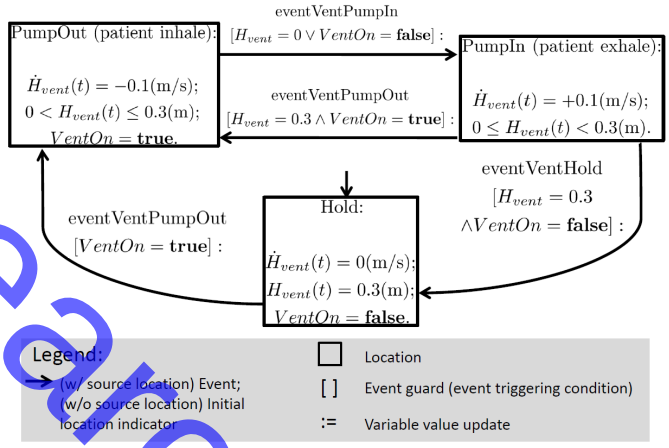

Figure 2. Offline hybrid automaton of Ventilator

The ventilator is basically a compressible air reservoir: a cylinder of height $H_{vent}(t)$ ($0 \leq H_{vent}(t) \leq 0.3$(m)). The movement of ventilator cylinder (indicated by $\dot{H}_{vent}(t)$) pumps out/in oxygen/air to/from patient, thus helping the patient to inhale/exhale. The ventilator behavior is defined by the hybrid automaton in Fig. 2. The automaton has three locations: PumpOut, PumpIn, and Hold. When the supervisor (will be discussed later, see Fig. 7) allows the ventilator to work (i.e., when global variable $VentOn$ is set to **true**), the ventilator switches between pumping out (where $\dot{H}_{vent} = -0.1$m/s) and pumping in (where $\dot{H}_{vent} = +0.1$m/s). This causes the patient to inhale oxygen and exhale respectively. When the supervisor pauses the ventilator (i.e., when $VentOn$ is set to **false**), the ventilator cylinder will try to restore to its maximum height (0.3m) and holds there until the ventilator is allowed again ($VentOn$ set to **true**).

With the ventilator hybrid automaton at hand, we can now start modeling the patient. The patient hybrid automa-

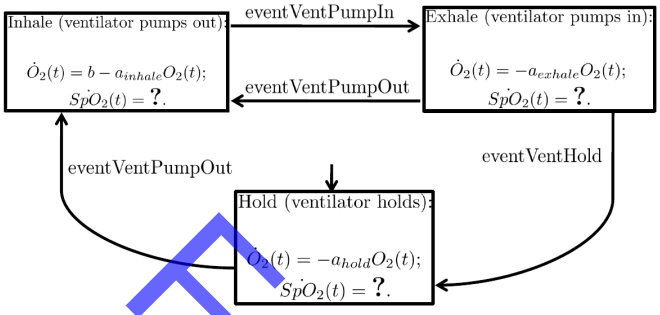

Figure 3. Offline hybrid automaton of Patient. Note in location Hold (which corresponds to ventilator Hold), the patient still exhale due to chest weight.

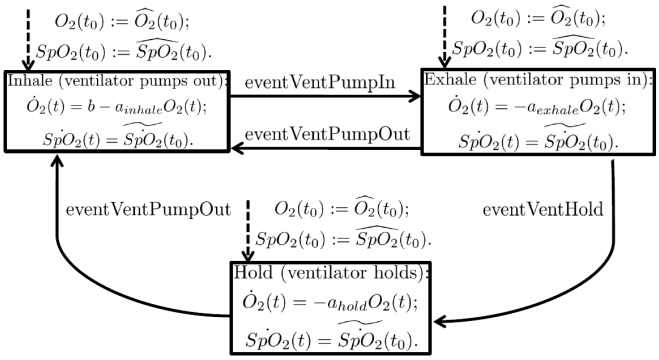

Figure 4. Online hybrid automaton of Patient.

ton (see Fig. 3) is tightly coupled with the ventilator hybrid automaton (see Fig. 2). It also has three locations: Inhale, Exhale, and Hold, which respectively correspond to the ventilator hybrid automaton's locations of PumpOut, PumpIn, and Hold. The events between the three locations are also triggered by corresponding events from the ventilator hybrid automaton.

Inside of each location are the offline continuous time models for windpipe oxygen level $O_2(t)$ and blood oxygen level $SpO_2(t)$. Unfortunately, though there are good offline models for $O_2(t)$ [10], there is *no* accurate enough offline model for $SpO_2(t)$. This is because blood oxygen level are strongly affected by complex human body biochemical reactions. To our best knowledge, its behavior is hard to be accurately described/predicted offline by any known tools [8][12].

Therefore, we failed to model $SpO_2(t)$ offline, and hence failed to model the patient offline. What is worse, as the patient model is an indispensable component of the holistic offline model, *the offline model checking of laser tracheotomy MDPnP fails*.

2.2. Online Modeling

The failure of offline approach forces us to consider the proposed online approach (see Section 1) instead. Specifically, we sample the patient's windpipe/blood oxygen level periodically, with a period of $T = 3$ (second). Suppose at $t_0 = kT$ ($k \in \mathbb{N}$), we got the most up-to-date windpipe oxygen level sensor reading $\widehat{O_2}(t_0)$ and blood oxygen level sensor reading $\widehat{SpO_2}(t_0)$, we can then build the hybrid systems model for the interval of $(t_0, t_0 + T]$.

Same as the offline model checking, we use global variable t to represent the global clock, except that now t is initialized to t_0 and stops at $(t_0 + T)$ as we only care about the system's *short-run* safety until $(t_0 + T)$.

The patient hybrid automaton now looks like Fig. 4. The biggest change is the continuous time model for the blood oxygen level $SpO_2(t)$. In offline model checking, we have to describe the long-run behavior of $SpO_2(t)$, which is too difficult. However, in online model checking, we only have to describe $SpO_2(t)$'s behavior in interval $(t_0, t_0+T]$, where $T = 3$ (second). If we only look into such short-run future, blood oxygen level curve $SpO_2(t)$ is very describable and predictable. It cannot plunge from 99% to 59% within just 3 seconds, neither can it show a saw-toothed wave form. Instead, it must be smooth; in fact smooth enough to be safely predicted with standard tools (such as linear regression) based on its past history.

In Fig. 4, we use a simple way to predict/describe $SpO_2(t)$ in $t \in (t_0, t_0 + T]$:

$$\dot{SpO}_2(t) \equiv \widehat{\dot{SpO}_2}(t_0), \qquad \forall t \in (t_0, t_0 + T],$$

where $\dot{SpO}_2(t)$ is the derivative of $SpO_2(t)$ at time t; and $\widehat{\dot{SpO}_2}(t_0)$ is the estimation (e.g., via linear regression) of $\dot{SpO}_2(t_0)$ based on $SpO_2(t)$'s history recorded during $(t_0 - T_{past}, t_0)$. T_{past} is a configuration constant picked empirically offline; it can be $+\infty$, which implies the estimation considers all history data available. In our case study, we pick $T_{past} = 30$ seconds.

Also, depending on the patient's state at time t_0, the initial location can be Inhale, Exhale, or Hold. Whichever location it is, the initial value of windpipe/blood oxygen value should be $\widehat{O_2}(t_0)$ and $\widehat{SpO_2}(t_0)$ respectively.

We now look at other entities in the laser tracheotomy MDPnP: O_2 Sensor, SpO_2 Sensor, Ventilator, Laser Scalpel, Surgeon, and Supervisor.

First, since the online model only looks into the short-run future of $(t_0, t_0 + T]$, where T is also the sensor sampling period, there is no interactions with sensors throughout the interval of $(t_0, t_0 + T]$. Therefore, in online model checking, the hybrid automata of O_2 sensor and SpO_2 sensor are unnecessary. This helps shrink the verification state space.

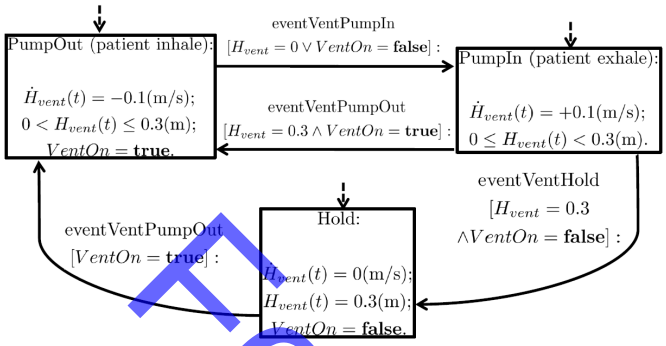

Figure 5. Online hybrid automaton of Ventilator.

Next, the ventilator hybrid automaton in online model (see Fig. 5) is almost the same as its offline model counterpart (see Fig. 2) A main difference is that the online model's initial location can be any location depending on the ventilator's state at t_0.

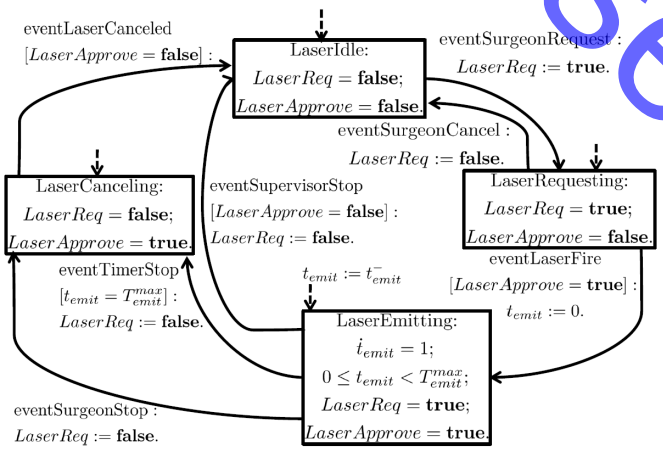

Figure 6. Online hybrid automaton of LaserScalpel. This is the only automaton that sets the value of global variable $LaserReq$.

The last entity that directly interacts with the patient is the laser scalpel. We can actually model the laser scalpel and the surgeon with one hybrid automaton: the laser scalpel hybrid automaton (see Fig. 6). The automaton has four locations, defined by the four possible combinations of two global boolean variables: $LaserApprove$ and $LaserReq$. The laser scalpel emits laser if and only if both $LaserApprove$ and $LaserReq$ are **true** (i.e., in location LaserEmitting). The meanings of $LaserApprove$ and $LaserReq$ are as follows.

$LaserApprove$ indicates whether the supervisor (see Fig. 1) allows the laser scalpel to emit laser (**true** for yes and **false** for no). Its value can only be set by the supervisor hybrid automaton (see Fig. 7), which is to be further explained later.

$LaserReq$, however, indicates whether the laser scalpel wants to emit laser (**true** for yes and **false** for no). Its value can only be set by the laser scalpel hybrid automaton. The value setting takes place in the following events: i) when in LaserIdle, the surgeon can request emitting laser through eventSurgeonRequest, which sets $LaserReq$ to **true**; ii) when in LaserRequesting or LaserEmitting, the surgeon can request stopping laser emission through eventSurgeonCancel and eventSurgeonStop respectively, which both set $LaserReq$ to **false**; iii) when in LaserEmitting, the supervisor can stop the laser emission at any time by setting $LaserApprove$ to **false**, which triggers eventSupervisorStop and sets $LaserReq$ to **false**; iv) for additional safety, when in LaserEmitting, timer t_{emit} prevents emitting laser continuously for too long (for more than T^{max}_{emit} seconds, to be exact) by triggering eventTimerStop, which sets $LaserReq$ to **false**.

The laser scalpel hybrid automaton's initial location can be anywhere depending on the laser scalpel's state at t_0. Special care is needed for the case when the initial location is LaserEmitting. In this case, the initial value of t_{emit} should be the value of t_{emit} at time t_0^- (i.e., the t_{emit} clock reading from the end of last sampling period), denoted as t^-_{emit}. This is to monitor the *continuous* duration of laser emission *across* the temporal boundaries of periodical online model checking.

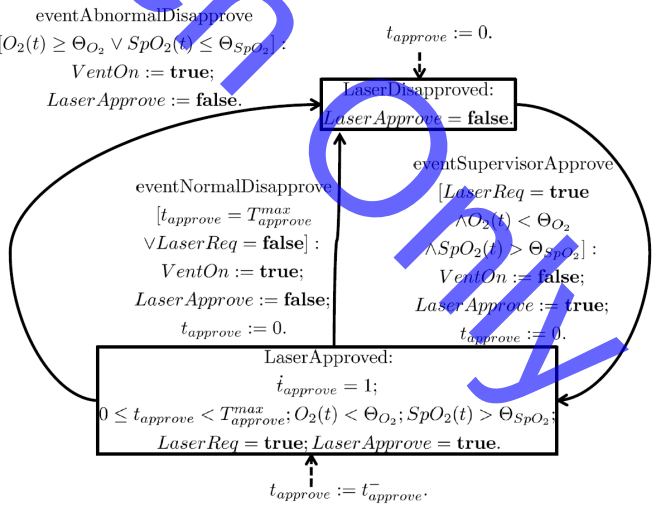

Figure 7. Online hybrid automaton of Supervisor. This is the only automaton that sets the value of global variable $VentOn$ and $LaserApprove$.

Finally, all medical device entities are interlocked by the supervisor, the central decision making computer (see Fig. 1). The supervisor maneuvers two global variables: $VentOn$ and $LaserApprove$. The setting of the two variables to **true/false** determines the on/off of ventilator and the permission/denial of emitting laser respectively.

The value setting decisions are made dependent on the most up-to-date information on the patient's windpipe oxygen level $O_2(t)$ and blood oxygen level $SpO_2(t)$. Since we know $O_2(t_0) = \widehat{O_2}(t_0)$ and $SpO_2(t_0) = \widehat{SpO_2}(t_0)$, based on the models given in the patient hybrid automaton (see Fig. 4), we can predict $O_2(t)$ and $SpO_2(t)$ for any t in our intend-to-verify interval of $[t_0, t_0 + T]$. Therefore, we can construct the supervisor hybrid automaton as Fig. 7, which directly uses $O_2(t)$ and $SpO_2(t)$ predicted by the patient hybrid automaton (see Fig. 4) for decision making.

The supervisor hybrid automaton has two locations: LaserDisapproved and LaserApproved. When in LaserDisapproved, the supervisor needs eventSupervisorApprove to move to LaserApproved. This event is triggered when the following three prerequisites hold:

Prerequisite 1 : the laser scalpel is requesting emitting laser (i.e., $LaserReq =$ **true**).

Prerequisite 2 : the windpipe oxygen level is less than threshold Θ_{O_2};

Prerequisite 3 : the blood oxygen level is greater than threshold Θ_{SpO_2}.

Through eventSupervisorApprove, the supervisor approves the emission of laser by setting $VentOn$ to **false** and $LaserApprove$ to **true**. This event also resets a clock $t_{approve}$ to monitor the continuous duration of the supervisor's stay in location LaserApproved.

When the supervisor stays in LaserApproved for too long (i.e., when $t_{approve}$ reaches $T^{max}_{approve}$) or when Prerequisite 1 no longer holds (i.e., when $LaserReq$ becomes **false**), the eventNormalDisapprove is triggered. This event moves the supervisor back to location LaserDisapproved and resets $VentOn$ to **true**, $LaserApprove$ to **false**, and $t_{approve}$ to 0.

In contrast to eventNormalDisapprove, event*Abnormal*Disapprove is triggered when the supervisor is in LaserApproved while Prerequisite 2 or 3 stops to hold. This event also moves the supervisor back to location LaserDisapproved and resets $VentOn$ to **true**, $LaserApprove$ to **false**. But unlike eventNormalDisapprove, eventAbnormalDisapprove does not reset $t_{approve}$. This allows us to check whether the optional Safety Rule 3 is violated. Note there is no need to check Safety Rule 3 under event*Normal*Disapprove, as the event is triggered when $t_{approve} = T^{max}_{approve} > T^{min}_{approve}$ or the laser scalpel requests to stop emission by itself (i.e., when $LaserReq$ becomes **false**).

The last thing to note is the initial location. Same as the other online hybrid automata, the initial location for the online supervisor automaton can be either LaserDisapproved or LaserApproved, depending on the state of the supervisor at time t_0. When LaserDisapproved is the initial location, $t_{approve}$ is initialized to 0. This will make pass the checking against Safety Rule 3 (as the check is the responsibility of the previous online model checking period). When LaserApproved is the initial location, $t_{approve}$ is initialized to $t_{approve}$'s value at time t_0^-, denoted as $t^-_{approve}$. This is to monitor the *continuous* duration of approving laser emission *across* the temporal boundaries of periodical online model checking.

3. Evaluations

To demonstrate the feasibility of our proposed approach, we carry out evaluations using real-world windpipe/blood oxygen level traces [1]. The traces are extracted from PhysioNet [2], a comprehensive online public database (set up by NIH, NIBIB, and NIGMS) of medical traces logged by various medical institutes in the United States.

Our evaluation aims to validate two claims:

Claim 1 : real-time (i.e., the model checking time cost is small and bounded) online hybrid systems model checking of short-run future is possible;

Claim 2 : online short-run modeling of complex medical parameters (such as blood oxygen level) is accurate.

To validate the first claim, we run our online hybrid systems model checking program $\mathcal{P}$ upon emulated windpipe/blood oxygen level sensors for 1200 seconds. Every $T = 3$ seconds during the 1200-second emulation period, our program $\mathcal{P}$ queries the emulated sensors for windpipe/blood oxygen level readings. The two emulated sensors reads corresponding real-world traces from PhysioNet respectively. Based on the readings, $\mathcal{P}$ builds online hybrid systems models as described in Section 2.2, and verifies it. The specific modeling and verification software used is PHAVer [7], a well-known hybrid systems model checking tool. Our computation platform is a Lenovo Thinkpad X201 with Intel Core i5 and 2.9G memory; the OS is 32-bit Ubuntu 10.10.

Throughout the 1200-second emulation period, program $\mathcal{P}$ carries out $1200/3 = 400$ rounds of online modeling and verifications. The statistics of execution time cost is described by Table 1. Through the statistics we see that all executions finish within $\frac{1}{2}T = 1.5$ second, and over 90%

[1] The windpipe oxygen level traces are derived from windpipe carbon dioxide level traces.

finish within 1 second. These time costs are all much shorter than the $T = 3$ (second) sampling/model-checking period.

If we have to strictly finish model checking before an interval starts, we can carry out 2-stage pipelining with phases of 0 and $\frac{1}{2}T = 1.5$ second (this implies our sensors are actually sampling every $\frac{1}{2}T$). With pipelining, real-time online hybrid systems model checking into the short-run future is possible.

Table 1. Statistics of execution time cost of online short-run hybrid systems model checking (unit: second)

Min	Max	Mean	Std
0.571	1.445	0.727	0.163

To validate the second claim, we carry out statistics on the prediction error of blood oxygen level curve. During the online verification, every time instance $t_0 = kT$ ($k \in \mathbb{N}$, $T = 3$ (second)), we sample the blood oxygen level and predict (see Fig. 4) the blood oxygen level curve in $(t_0, t_0 + T]$. Let the predicted blood oxygen level at time $(t_0 + T)$ be $\widehat{SpO}_2(t_0 + T)$. Let the PhysioNet trace reading of blood oxygen level at time $(t_0 + T)$ be $\widetilde{SpO}_2(t_0 + T)$. We define the relative prediction error at time $(t_0 + T)$ to be

$$ERR_{SpO_2}(t_0 + T) = \frac{|\widehat{SpO}_2(t_0 + T) - \widetilde{SpO}_2(t_0 + T)|}{\widetilde{SpO}_2(t_0 + T)}.$$

The statistics of the relative prediction errors throughout the 1200-second emulation period is described by Table 2. The statistics show that our online model checking's predictions on short-run behavior of blood oxygen level curve well match the real-world trace.

Table 2. Statistics of blood oxygen level online modeling relative errors (%)

Min	Max	Mean	Std
0	3.31	0.58	0.51

4. Related Work

Our approach is different from the well-known runtime verification [6]. Runtime verification aims to discover latent bugs of programs by logging and analyzing the programs' execution traces under varied inputs/configurations. It is not for *predicting/preventing* faults *before* they ever happen; whilst our approach is. Considering for many medical CPS systems, the cost/consequence of possible faults in test runs is high or even unbearable, our approach of predicting and preventing faults before they ever happen is necessary.

Qi et al. [13] propose combining model checking and runtime monitoring for web server dependability. But they are still focusing on discrete (automata) model checking, rather than hybrid systems model checking that this paper is about.

Also, our approach is not model-checker specific, though our evaluation in this paper uses PHAVer. In fact, we are considering integrating our approach with other well-known model checkers, such as Bogor [14], CellExcite [4] etc..

Arney et al. [3] propose using simple differential equations to model blood oxygen level. However, the paper just uses the simple model to demonstrate other designs. The accuracy of the model itself is not the focus of the paper.

Kim et al. [10] also studied the laser tracheotomy MDPnP. But their focus is not on hybrid systems model checking.

5. Conclusions and Future Work

Through our case study on laser tracheotomy MDPnP, we show that our online short-run approach can effectively address the two challenges in MDPnP CPS hybrid systems model checking. By focusing on online and short-run future, many originally hard to describe/predict human body parameters become describable and predictable; and many parameters become fixed numerical values, which greatly reduces verification state space. Our empirical evaluations based on real-world vital sign traces show that our approach is feasible in the sense of both execution time and modeling accuracy.

As future work, we will theoretically prove and improve our approach's real-time time cost bound. We will also propose a more comprehensive MDPnP development and runtime framework that integrates our proposed approach.

6. Acknowledgement

The research project related to this paper in Hong Kong Polytechnic University (HK PolyU) is supported in part by Hong Kong RGC General Research Fund (GRF) PolyU 5245/09E, The HK PolyU Internal Competitive Research Grant (DA) A-PJ68, HK PolyU Newly Recruited Junior Academic Staff Grant A-PJ80, HK PolyU Fund for CERG Project Rated 3.5 (DA) grant A-PK46, and Department of Computing start up fund. Lei Bu is supported by the National Natural Science Foundation of China (No.90818022, No.91018006) and National S&T Major Project (2009z01036-001-001-3). Xue Liu is supported in part by NSERC Discovery Grant 341823-07, and FQRNT grant 2010-NC-131844. Rong Zheng is supported in part by

the National Science Foundation (NSF) under award CNS-0832089.

References

[1] *Medical Device Plug-and-Play (MDPnP)*. `http://www.mdpnp.org`.

[2] *PhysioNet: the Research Resource for Complex Physiologic Signals*. `http://www.physionet.org`.

[3] D. Arney, M. Pajic, J. Goldman, I. Lee, R. Mangharam, and O. Sokolsky. Toward patient safety in closed-loop medical device systems. *Proceedings of the 1st International Conference on Cyber-Physical Systems*, Apr. 2010.

[4] E. Bartocci, F. Corradini, E. Entcheva, R. Grosu, and S. A. Smolka. Cellexcite: An efficient simulation environment for excitable cells. *BMC Bioinformatics*, 9(2):1–13, Mar. 2008.

[5] E. Clarke, O. Grumburg, and D. Peled. *Model Checking*. MIT Press, 1999.

[6] B. Finkbeiner, S. Sankaranarayanan, and H. Sipma. Collecting statistics over runtime executions. *ENTCS*, 70:4, 2002.

[7] G. Frehse. PHAVer: Algorithmic Verification of Hybrid Systems past HyTech. *Proc. of HSCC'05*, LNCS 2289:258–273, 2005.

[8] J. A. Grass. Patient-controlled analgesia. *Anesthesia & Analgesia*, 101(5S):S44–S61, Nov. 2005.

[9] T. Henzinger. The theory of hybrid automata. *Proc. of the 11th Annual IEEE Symposium on Logic in Computer Science*, pages 278–292, 1996.

[10] C. Kim, M. Sun, S. Mohan, H. Yun, L. Sha, and T. F. Abdelzaher. A framework for the safe interoperability of medical devices in the presence of network failures. *Proceedings of the 1st International Conference on Cyber-Physical Systems*, Apr. 2010.

[11] I. Lee and O. Sokolsky. Medical cyber physical systems. *Proc. of DAC*, 2010.

[12] J. X. Mazoit, K. Butscher, and K. Samii. Morphine in postoperative patients: Pharmacokinetics and pharmacodynamics of metabolites. *Anesthesia and Analgesia*, 105(1):70–78, 2007.

[13] Z. Qi, A. Liang, H. Guan, M. Wu, and Z. Zhang. A hybrid model checking and runtime monitoring method for c++ web services. *Proc. of the Fifth International Joint Conference on INC, IMS and IDC*, 2009.

[14] Robby, M. B. Dwyer, and J. Hatcliff. Bogor: An extensible and highly-modular software model checking framework. *Proc. of the 9th European Software Engineering Conference (ESEC/FSE-11)*, 2003.

[15] L. Sha, S. Gopalakrishnan, X. Liu, and Q. Wang. Cyber-physical systems: A new frontier. *Machine Learning in Cyber Trust: Security, Privacy, and Reliability*, pages 3–13, 2009.

[16] P. Tabuada. *Verification and Control of Hybrid Systems: A Symbolic Approach*. Springer, 2009.