# Operational Safety Verification of AI-Enabled Cyber-Physical Systems

by

Imane Lamrani

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved August 2020 by the
Graduate Supervisory Committee:

Sandeep Gupta, Chair
Armando Rodriguez
Ayan Banerjee
George Runger
Yi Zhang

ARIZONA STATE UNIVERSITY

December 2020

ABSTRACT

One of the main challenges in testing artificial intelligence (AI) enabled cyber physical systems (CPS) such as autonomous driving systems and internet-of-things (IoT) medical devices is the presence of machine learning components, for which formal properties are difficult to establish. In addition, operational components interaction circumstances, inclusion of human-in-the-loop, and environmental changes result in a myriad of safety concerns all of which may not only be comprehensibly tested before deployment but also may not even have been detected during design and testing phase. This dissertation identifies major challenges of safety verification of AI-enabled safety critical systems and addresses the safety problem by proposing an operational safety verification technique which relies on solving the following subproblems:

1. Given Input/Output operational traces collected from sensors/actuators, automatically learn a hybrid automata (HA) representation of the AI-enabled CPS.

2. Given the learned HA, evaluate the operational safety of AI-enabled CPS in the field.

This dissertation presents novel approaches for learning hybrid automata model from time series traces collected from the operation of the AI-enabled CPS in the real world for linear and non-linear CPS. The learned model allows operational safety to be stringently evaluated by comparing the learned HA model against a reference specifications model of the system. The proposed techniques are evaluated on the artificial pancreas control system.

*To my beloved parents El Mokhtar and Hakima*

ACKNOWLEDGEMENTS

I was very fotunate to have Dr. Sandeep Gupta and Dr. Ayan Banerjee advise me throughout this journey towards achieving deeper levels of knowledge and experience. I would like to express my sincere gratitude for all their endeavors. During my internship at the Food and Drug Administration, I was very lucky to be under the direct supervision of Dr. Yi Zhang. I am grateful for his support, encouragement, motivation, and mentoring that extended beyond the internship experience. This journey allowed me to meet and work alongside my dear colleagues at the iMPACT Laboratory: Vinaya, Azemat, Javad, Koosha, Apu, Jung, Prajwal, and Bernard. It was a great pleasure to get to know each one of you and I wish you all the best in your career. I am grateful for the support and advice of my beloved auntie Dr. Chadia Affane and my dear uncle Dr. Saad Biaz. They are my idols and the main motivators to embark in this challenging experience. I want to thank my back-bones and lovely sisters Asmae, Sanae, and Soukaina who were always there to support me emotionally during hardships along the way. They contributed significantly to my personal growth and pushed me to become the best version of myself. My deepest love goes to my dear fiancee Hammed whose presence in my life makes it more beautiful. Finally, I would like to thank myself for not giving up on my dreams and for always seeking knowledge. You did it, Dr. Lamrani.

# GLOSSARY

**Accident**: undesired event that results in a harm.

**Functional failure**: faults associated with logical components of the system.

**Functional safety**: safe function of a device or a system focusing on electronics and related software.

**Harm**: direct or indirect physical injury or damage to the health of people. Indirect injury can be a result of damage to the environment or loss/damage of equipment or property.

**Hazard**: a potential source of harm.

**Operational safety**: assurance that the system is behaving out in the field as designed.

**Physical faults**: faults associated with mechanical, electrical, or electronic components.

**Rectangular and Diagonal Guards:** Guards are thresholds on the continuous variables of a cyber-physical system. Guards are rectangular if they are represented as $x\{\leq, \geq\}m$, where $m$ is a real number and $x$ is a continuous variable. They are diagonal if they are represented as $Ax + B\{\leq, \geq\}m$, where $A$ and $B$ are real constants.

**Risk**: a combination of the probability of occurrence of harm and the severity (level) of that harm.

**Safety-critical sytem**: a system whose malfunction or failure is catastrophic or critical.

**Safety/System safety**: freedom from unacceptable risk of injury or damage to humans, environment, and the system itself.

**Systematic faults**: faults associated with the development mistakes at the specification, design, or implementation phase.

**System safety engineering**: an engineering discipline that employs knowledge from system engineering, management principles, and systems theory to identify and eliminate hazards or reduce the associated risks when the hazards cannot be eliminated.

**Tolerable (acceptable) Risk**: a risk that the appropriate acceptance authority is willing to accept without additional mitigation.

**Under-determined CPS and internal variables:** In most practical scenarios CPS controllers are partially observable systems. This is because during operational deployments, not all continuous parameter evolution used by the controllers can be monitored. This results in hidden variables. Deriving temporal evolution of system variables from far lesser number of observed parameters thus results in an under-determined CPS problem.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

# Chapter 1

# INTRODUCTION

## 1.1 Safety Engineering of AI-enabled CPS

The increasing number of recent cases of fatal accidents of safety-critical cyber-physical systems (CPS) have renewed the discussion on the verification, validation, and certification of these systems. The novel nature of CPS which embodies artificial intelligence (AI) and machine learning (ML) components requires the development of novel rigorous safety verification and validation techniques that can cope with the complex nature of the cutting-edge CPS technology being developed. The V-Model, shown in Fig. 3.2, is widely used in system's safety engineering by autonomous driving, medical, and aerospace industries. Initially, a simple high-level model is developed using predictive environment and user models, which are often incomplete. The simple model is then modified throughout the system's development lifecycle to account for previously ignored or unknown physical phenomena. At the operation phase, user behavior contingencies, environmental changes, and components interaction circumstances which only manifest in-the-field of operation may put the AI-enabled CPS's safety in jeopardy. In other words, the AI-enabled CPS operation in the real world tends to diverge from the safety assured design of the system [1] . In addition, the actual deployed system's model may not conform with the system's knowledge mental model of the user/operator and operational testing agent. This discrepancy in the system's knowledge across system's stakeholders is one of the highlighted operational safety issues regarding Watchkeeper accidents (DSA (2019)), which was later one of the potential causes of Boeing 737 Max 8 aircraft crash where operators lacked crucial infor-

[1] Throughout this dissertation, we refer to this problem as operational safety verification problem.

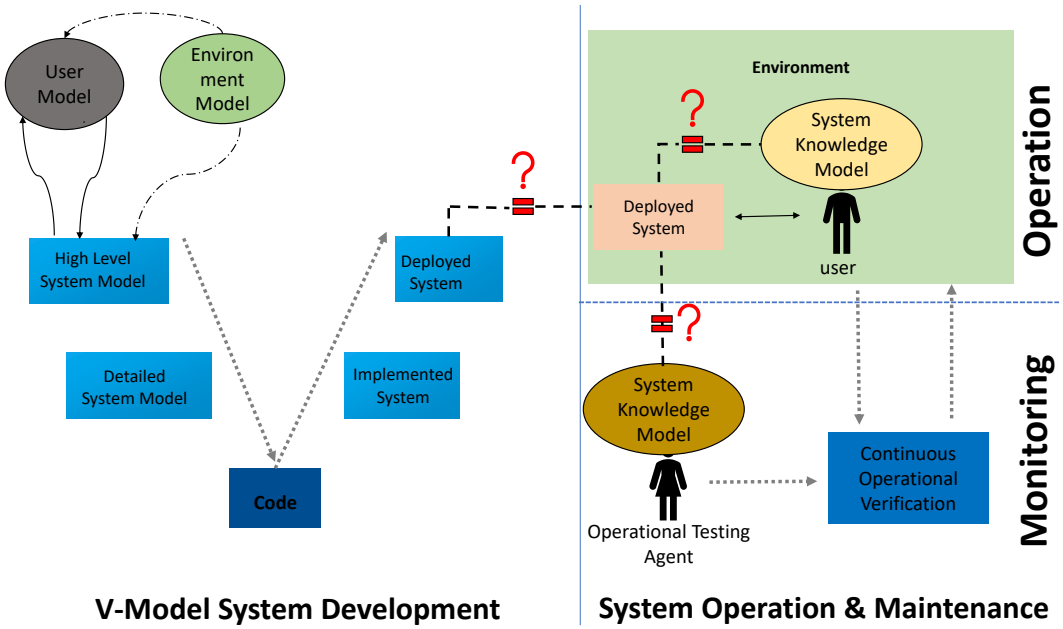


**Figure 1.1:** V-Model Based System Engineering and Operation.

mation about the MCAS system (Transportasi (2018)). The dissonance between ”what the system is designed to do”, ”what the operator thinks the system is doing”, and ”what the system is actually doing” is a crucial safety problem. In other words, some control components of the CPS may not have been explicitly declared (intentionally or unintentionally) in the specifications model. On the other hand, safety-critical CPS should meet government regulatory requirements before marketing. Due to production pressure and conflicting goals and tradeoffs, organizations tend to migrate to a state of higher risk (Leveson (2011)) and sometimes they tend to conceal crucial information about the system's inner workings during the regulatory process. For example, the Volkswagen's defeat device that allowed vehicles to improperly meet US standards during regulatory testing (Contag *et al.* (2017)). These concerns motivate a need for a continuous rigorous operational safety verification technique to help monitor the system's operation in the real world.

## 1.2 Examples of Operational Safety Violation

A case in point is three separate instances of problems caused by Maneuvering Characteristics Augmentation System (MCAS) sub-component in Boeing 737 Max 8 aircraft. The MCAS system was self-certified to be safe under certain scenarios and investigators are still examining why the U.S. Federal Aviation Administration failed at detecting the problems during the plane's certification. According to recent reports (Hatton and Rutkowski (2019); Johnston and Harris (2019)), all three cases were caused by sensor failures. Two of the three cases resulted in fatal disasters but in one case, the presence of a third co-pilot (a rare presence) helped to override the MCAS system and recover from a potential nose dive. In the two failure cases, the MCAS system was engaged during take-off which gives very little time for the pilots to properly react. This clearly shows that the MCAS was potentially used in practice under very different scenarios than it was tested for. As such, the coverage problem for AI-enabled safety critical CPS can potentially encounter combinatorial explosion due to the presence of significant number of interacting external sub-components and the ever-changing operational context (Leveson (1986); Leveson (2011)).

Another example of safety violation is the fatal crash of the autonomous driving Uber vehicle that caused death to a pedestrian. Every component of the system is claimed to be operating properly including the software, yet the environmental context was overlooked in the requirement and design phases of the system safety development.

The Volkswagen's cheating device that allowed vehicles to improperly meet US standards during emission regulatory testing is also considered an operational safety violation (Contag *et al.* (2017)).

## 1.3 Contributions

### *1.3.1 List of Publications*

This proposal consists of an overview of the following publications.

**I:** Imane Lamrani, Ayan Banerjee, and Sandeep K.S Gupta. ”Co-simulation of Physical Model and Self-Adaptive Predictive Controller Using Hybrid Automata” Workshop on Formal Co-Simulation of Cyber-Physical Systems CoSim-CPS 2018

**II:** Imane Lamrani, Ayan Banerjee, and Sandeep K.S Gupta. ”N-HyMn: Mining Non-Linear Hybrid Systems from Input Output Traces of Cyber-Physical Systems.” IEEE Industrial Cyber-Physical Systems ICPS 2018

**III:** Ayan Banerjee, Imane Lamrani, Prajwal Paudyal, Sandeep Gupta. ”Generation of Movement Explanations for Testing Gesture Based Co-Operative Learning Applications.”IEEE International Conference On Artificial Intelligence Testing AITEST 2019

**IV:** Imane Lamrani, Ayan Banerjee, and Sandeep K.S Gupta. ”Operational Safety Verification Via Hybrid Automata Mining Using I/O Traces of AI-Enabled CPS.”Artificial Intelligence Safety Workshop SAFEAI 2020

**V:** Imane Lamrani, Ayan Banerjee, and Sandeep K.S Gupta. ”Toward Operational Safety Verification of AI-Enabled CPS (student abstract).” AAAI Conference on Artificial Intelligence 2020

**VI:** Imane Lamrani, Ayan Banerjee, and Sandeep K.S Gupta. ”N-HyMn: Mining Non-Linear Hybrid Systems from Input Output Traces of Cyber-Physical Systems.” IEEE Transactions On Industrial Informatics IEEE-TII 20

### *1.3.2 Author's Contribution*

**Publication I**: The author developed a co-simulation framework for self-adaptive predictive (SAP) control systems. As an initial step, the author proposed the framework and discussed preliminary encouraging results for the formal verification of SAP systems.

**Publication II**: The author developed the proposed model mining technique using Input/Output (I/O) traces collected from the operation of the hybrid system. The author implemented the proposed technique, performed its experimental evaluation on a simple artificial pancreas (AP) control system, and wrote the paper. The proposed technique is developed on the basis of the assumption that an initial formal model of the simple AP exists. Encouraging design safety conclusion results are presented in the published paper.

**Publication III**: The ideas and approaches introduced in this work were proposed and implemented by Dr. Ayan Banerjee. This work proposes an explanation framework for machine learning (ML) based gesture recognition applications, wherein the proposed formal model mining approaches presented in **publication I** and **II** were applied.

**Publication IV and V**: The author presents a novel safety verification technique based on learning a hybrid automaton model and using it to evaluate the safety of the system during its operation in the field.

**Publication VI**: This work concentrates on showing the efficacity and effectiveness of the proposed model mining technique for non-linear complex AI-enabled systems, where an initial formal model exists.

The author developed an advanced model mining technique for non-linear complex AI-enabled systems when an initial formal model does/doesn't exist. The proposed technique learns a formal model using only I/O traces and a very limited knowledge about the system under learning. The author implemented the proposed technique, performed its experimental evaluation on Medtronic Minimed 670G, and wrote the paper. Thanks to the US

Food and Drug Administration (FDA) and Mayo Clinic for collaborating in this research. Encouraging operational safety verification results are presented in the pending journal submission.

| System Model Mining Of | Initial Formal Model Provided | Safety Conclusions At | Publications |
|---|---|---|---|
| Linear Hybrid Systems | Yes | Design Phase | **II** |
| ML-Based Cooperative Learning Applications | No | Operation Phase | **III** |
| Non-Linear Complex AI-Enabled Systems | Yes | Operation Phase | **IV and V** |
| Non-Linear Complex AI-Enabled Systems | Yes/No | Operation Phase | **VI** |

**Table 1.1:** Proposal Contributions in Model Mining and Safety Conclusions

Chapter 2

# PRELIMINARIES

## 2.1 AI-based CPS

As shown in Fig. 2.1, an **AI-enabled CPS** a system that is engineered to map percepts into actions in order to control some physical aspect of the environment. An AI-enabled CPS is composed of an intelligent agent that interacts with the physical environment through sensors and actuators. The performance analyzer can be a simple reflex agent, a goal-based agent, a model-based agent, or a utility-based agent (Russell and Norvig (2016)). Given a set of actions, a simple reflex agent uses a basic condition-action rule to perform an action, whereas a goal-based agent incorporates a set of goals to achieve, a model-based agent involves a model of the internal aspects of the environment that are unperceived, and a utility-based agent uses a utility function to choose the optimal action if many actions satisfy the goals. The intelligent agent is considered a learning agent if it incorporates a learning

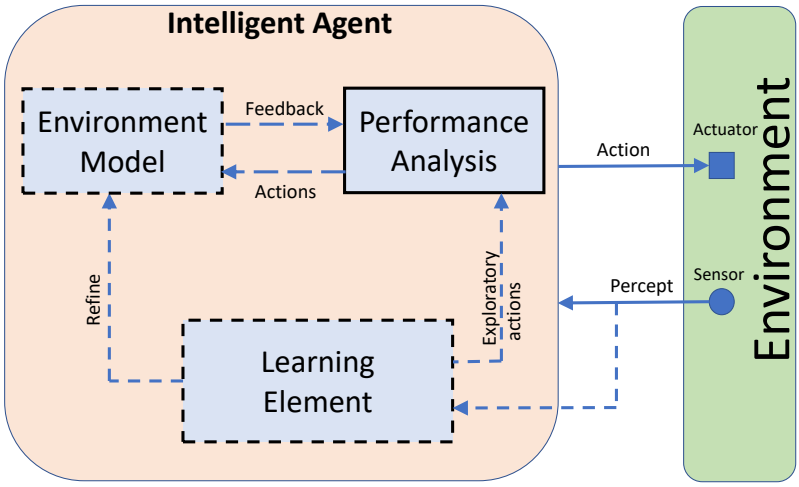


**Figure 2.1:** AI-enabled CPS.

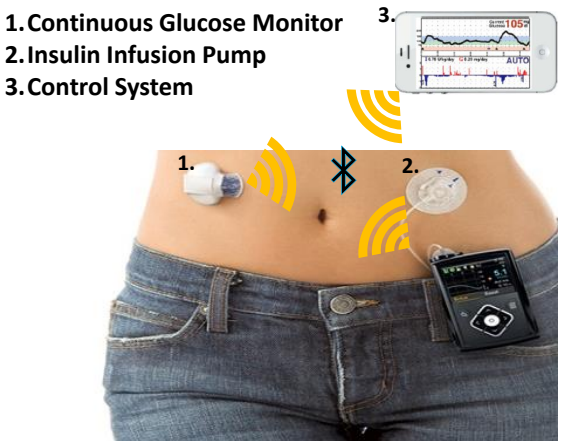


**Figure 2.2:** Artificial Pancreas Control System. (Photo: Medtronic)

element that allows it to learn from historical states of the environment and to improve its control structure.

### *2.1.1 Example of an AI-Enabled CPS: Artificial Pancreas*

Type 1 diabetes (T1D) is a chronic metabolic disease caused by the autoimmune destruction of the pancreas $\beta$ cells. In healthy subjects, pancreatic $\beta$ cells are responsible for the release of insulin to regulate the blood glucose (BG) variations, most commonly due to carbohydrates intake or physical activity. This regulatory process aims to maintain BG within the safe euglycemic range [70-180] mg/dl. The CLAP control system, shown in Fig. 2.2, is used for automated control of blood glucose level for T1D patients (Haidar (2016)). The controller running inside the insulin pump or on a device that is wirelessly connected to the insulin infusion pump receives glucose-meter value every 5 minutes from the continuous glucose monitoring (CGM) sensor. The controller carefully chooses the amount of insulin infusion rate $I_t$ to maintain a safe level of blood glucose, thereby avoiding occurrence of hypoglycemic and hyperglycemic events. These dangerous events happen as a result of an inaccurate infusion of insulin that can induce hypoglycemia i.e. BG $<$ 70 mg/dL, which can be potentially associated with serious threats to the subject including

coma and death. Conversely, prolonged hyperglycemia i.e. BG $>$ 180 mg/dL, can lead to critical health conditions including cardiovascular diseases. In 2016, FDA approved the first hybrid CLAP system Medtronic Minimed 670G that monitors BG and automatically adjusts the basal or bolus insulin delivery on the basis of the CGM readings and the user meal input (FDA (2016)). Following the Seridan system levels of automation, AP falls in the fifth and sixth automation levels (Sheridan and Parasuraman (2000)). An example of model-based CLAP is described in Appendix A.

### *2.1.2 Example of an AI-Enabled CPS: Advanced Driver Assist Systems*

Advanced driving assist systems (ADAS) are safety-critical AI-enabled CPS that are designed to work in uncertain environments. ADAS include automatic emergency braking, automatic parking, and auto-passing. The vehicle can be controlled through two control outputs: the throttle (acceleration or brake) and the steering speed. The autonomous driving control system consists of several control modes, *Cruise* where the car moves forward at a constant speed, *Brake* where a constant slow deceleration is applied, *HardBrake* where a constant hard deceleration is applied, *Speedup* where a constant acceleration is applied, and the lane switching modes *ShiftLeft* and *ShiftRight* in which the acceleration and steering are controlled to switch the vehicle to its left and right lane respectively (Fan *et al.* (2017)). The controller switches between these modes based on inputs from the sensors and the driver. The Society of Automotive Engineers (SAE) derived from the Sheridan automation hierarchy five discrete levels of automation specific to developing automated vehicles (Alves *et al.* (2018)).

## 2.2 AI-Enabled CPS's Input/Output Traces

I/O operational traces are time series data representing closed-loop operation of AI-based CPS. I/O traces comprise system's perceived variables that are collected from sensors

and actuators during the operation of the AI-based CPS and unperceived (internal) variables that are collected through simulations of the environment model used by the agent. Each trace may encompass one or more agent's modes.

## 2.3 Hybrid Automata

A **hybrid automaton** is a formal model of a closed-loop control system (Henzinger (2000)). A controller measures values of the continuous variables representing the plant using a sensor and decides to switch mode if a certain condition is satisfied. This decision is transmitted to the actuator that performs the desired change. As shown in Figure 2.3, a linear HA is a tuple of the following components (Alur *et al.* (1992); Alur *et al.* (1995)).

-$\mathcal{M} = \{m_0 \dots m_q\}$ is a set of discrete states or *control modes* where $m_0$ is the initial mode.

-$X$ is the continuous state space in which the continuous variables representing the physical system or the controller inputs $\overrightarrow{x} = \{x_1, x_2, \dots, x_n\}$ take their values. Hence $X \subset \mathcal{R}^n$, where $\mathcal{R}$ is the set of real numbers.

-a finite set of *Control Switches* in $\mathcal{M} * \mathcal{M}$, where $(m_i, m_j)$ defines the control switch from source mode $m_i$ to target mode $m_j$.

-a *Flow* function that assigns to each control mode $m \in M$ a set of linear differential algebraic equations that relates the continuous state space variables $\overrightarrow{x}$ to its derivatives and the controller outputs. For every discrete mode $m$, the equation takes the following form: $\frac{d\overrightarrow{x}}{dt} = A_m \overrightarrow{x} + B_m \overrightarrow{o} + C_m$, where $A_m$ is an $n \times n$ matrix, $B_m$ is an $n \times p$ matrix, whereas $C_m$ is an $n \times 1$ column vector.

-a *Guard condition* is a function that maps every control switch to a guard condition. A control mode change takes place when the corresponding guard condition is satisfied.

-a *Reset* function that maps every control switch to a reset condition. In this paper, $\dot{x}$ and $\frac{dx}{dt}$ both mean differential of $x$ w.r.t time $t$.

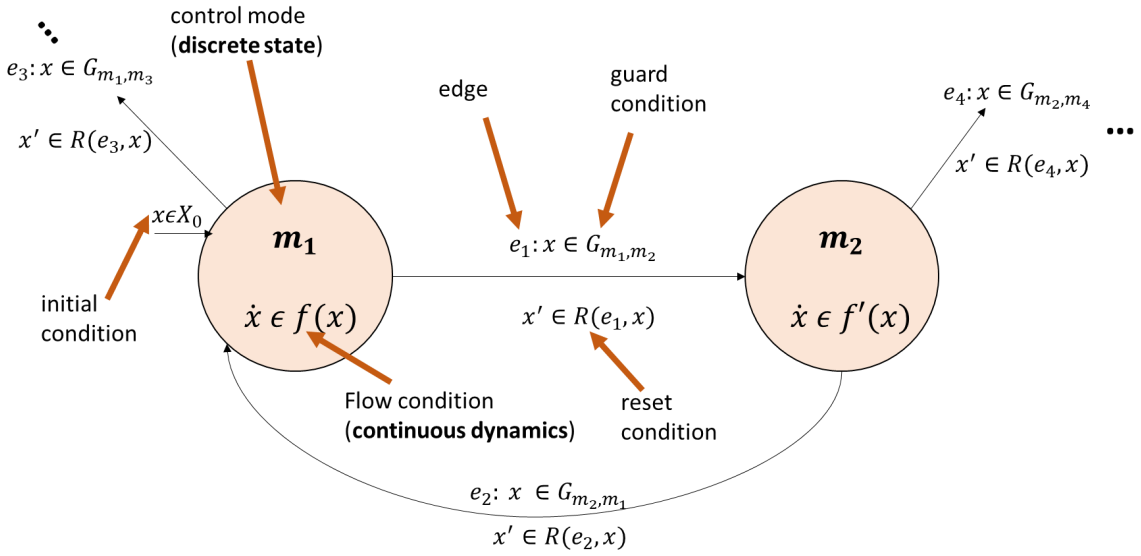


**Figure 2.3:** Hybrid Automaton.

### *2.3.1 Reachability Analysis*

Reachability analysis is formal verification technique over control system models that explores all possible trajectories of operation (a.k.a reachable states) as the result of hybrid automata model execution and starting from a pre-specified initial range of parameters of environmental conditions and users' behavior. At every time step, reachability analysis calculates the set of reachable states (continuous and discrete successor sets) via the evaluation of the differential equations for every point in the original set and within each control mode region. Since reachability analysis is intractable, its solution is always an over-approximation of the system's operating envelope (Alur *et al.* (1995)). The final reach set is a convex polytope enclosing all reachable sets. Figure 2.4 shows an example of evolution of reachable sets starting from an initial set $X_0$ within two control mode $m_1$ and $m_2$ with different flow dynamics.

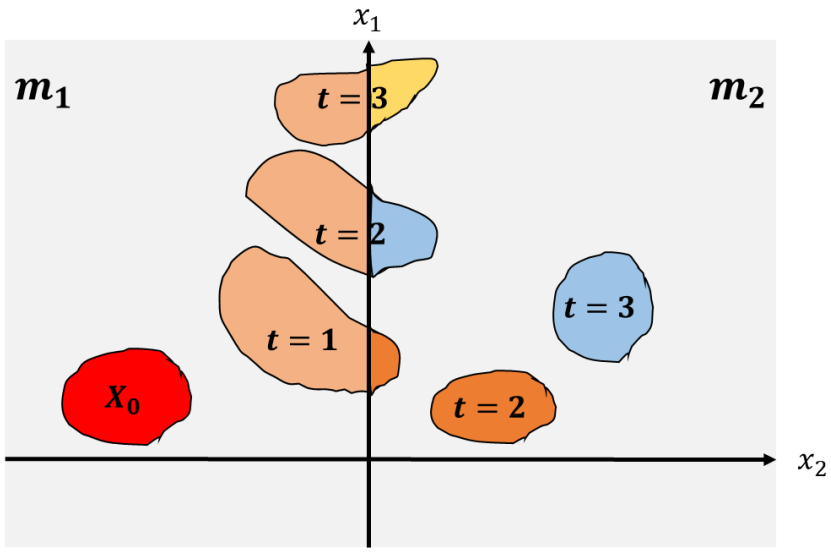


**Figure 2.4:** Example of Reachable Sets Computation.

**Linear Hybrid Automaton of a Simple-Reflex Artificial Pancreas System**

The dynamics of the AP are represented by nonlinear equations 10.1, 10.2 and 10.3, where $\dot{X}$ represents the rate of the variation in the interstitial insulin concentration, $\dot{G}$ is the rate of change of blood glucose concentration ($G$) for the infused insulin concentration $X$ and $\dot{I}$ is the variation in plasma insulin concentration ($I$) (Bergman *et al.* (1979)). Note that here only the blood glucose and insulin levels are the observed parameters from the operation of the AP in field. The parameter X is not observed but plays a significant role in relating blood glucose and insulin. The AP device has three control modes: 1- basal, where the insulin infusion rate $I_t = 5$, 2- braking, where $I_t = 0.5G + 44.75$, and 3- correction bolus, where $I_t = 50$ (Banerjee *et al.* (2013)).

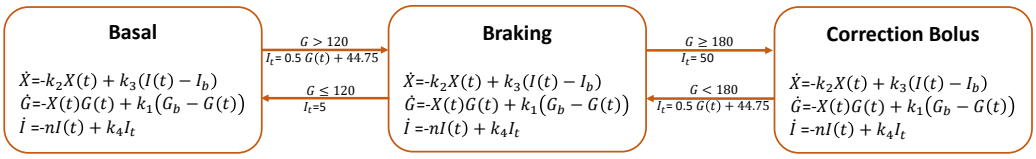


**Figure 2.5:** Hybrid Automaton Model of AP.

Figure 2.5 shows the hybrid automaton model of AP.

$$\dot{X} = -k_2X(t) + k_3(I(t) - I_b), \tag{2.1}$$

$$\dot{G} = -X(t)G(t) + k_1(G_b - G(t)), \tag{2.2}$$

$$\dot{I} = -nI(t) + k_4I_t(t). \tag{2.3}$$

The aim the AP control system is to maintain the prescribed level of blood glucose and avoid occurrence of hypoglycemic/hyperglycemic events. These dangerous events happen as a result of inaccurate infusion rates of insulin, e.g. if the glucose concentration $G$ goes above $180mg/dl$, it can lead to hyperglycemia while low glucose level i.e. below $60mg/dl$ can cause hyperglycemia. The hyperglycemia ($G > 180mg/dl$) and hypoglycemia ($G < 60mg/dl$) sets are considered the unsafe sets of the AP system. Safety of AP can be verified through reachability analysis on the HA of AP to verify that the reach set does not intersect with the unsafe sets (Asarin *et al.* (2000); Kong *et al.* (2015); Frehse *et al.* (2011)). On the other hand, operational safety verification of the AP control system considers verifying that the certified design model of AP (FDA (2016)) conforms with the operation results of AP out in the field. A hybrid automaton of the model-based CLAP of Medtronic Minimed 670G is illustrated in Appendix A.

## 2.4 Technical Preliminaries

### *2.4.1 Fisher Information and Cramer Rao Bound*

We consider the problem of deriving an unbiased estimator of a continuous variable $v$ from a series of observations. The estimator has design parameters expressed as a vector $\overrightarrow{\theta} = \{\theta_1, \theta_2, \ldots \theta_k\}$. The term unbiased indicates that the expected value of the output of the estimator is the true value of $v$. Fisher information provides a measure of the information carried by $v$ about an unknown design parameter $\theta_i$. Given a series of observations of the variable $v$ and executions of the estimator, the Fisher information is given by $\frac{\delta ln P(v|\theta_i)}{\delta \theta_i}$, where $P(v|\theta_i)$ is the conditional probability of the observation $v$ given the value of the design parameter $\theta_i$. Larger the value of this Fisher information, larger is the contribution of $\theta_i$ in determining the value of $v$. Hence, an effective method to reduce the number of design parameters that make significant contribution in the estimator for $v$ is to order them in decreasing order of Fisher information and only consider those design parameters that have significantly higher Fisher information. Once, the most significant design parameters are identified, the next logical step is to derive the minimum variance unbiased estimator (MVUE), such that the mean value of the estimator output is the true value of $v$ and the variance of the output of the estimator is minimized. In general, deriving MVUE of a system from a set of observations is an extremely difficult proposition. However, if the underlying design model is linear, then the Cramer Rao Lower Bound (CRLB) theorem can be used to derive the MVUE (Milanese and Belforte (1982)) . The CRLB considers a linear estimator for $v$ such that: $\overrightarrow{v_o} = HD + w$, where $\overrightarrow{v_o}$ is a set of observations for the variable $v$, H is a set of observations for the design parameters $\overrightarrow{\theta}$, $D$ is the matrix of coefficients for the linear estimator, and $w$ is the observation noise.

The CRLB states that the Fisher information matrix is given by:

$$I = \frac{H^T H}{\sigma^2}, \tag{2.4}$$

where $\sigma$ is the variance in the observation noise, while the MVUE is given by:

$$D = (H^T H)^{-1} H^T v. \tag{2.5}$$

This result will be used in our HyMn algorithm for two purposes: a) to derive flow equations in modes of hybrid system using input output observations, and b) to derive non-rectangular guards which are expressed as linear combinations of continuous state variables of the hybrid system.

### *2.4.2 Pearson's Divergence (PE) Score*

: This metric is used to compute a difference between two probability distributions $P$ and $P'$ of samples in two consecutive windows $\mathcal{Y}(t)$ and $\mathcal{Y}(t+w)$, respectively. The Pearson (PE) divergence is defined as:

$$PE(P, P') = \frac{1}{2} \int p'(Y) \left( \frac{p(Y)}{p'(Y) - 1} \right)^2 dY \tag{2.6}$$

where $p(Y)$ and $p'(Y)$ are probability density function of $P$ and $P'$, respectively.

### *2.4.3 DBSCAN Clustering*

DBSCAN is a density based clustering technique that uses three parameters, $MinPoints$, $Epsilon$, and a distance metric. Using the distance metric, it defines density as the number of points present in $Epsilon$ neighborhood of a given point. DBSCAN iterates over each data point to classify it as core point, $\geq MinPoints$ points in its $Epsilon$ neighborhood,

border points, not core points but in $Epsilon$ neighborhood of core points, and noise points, which are neither. The core points which are in $Epsilon$ neighborhood of each other are connected to form clusters. We used DBSCAN to cluster observed control mode changes in the input/output data.

### *2.4.4 RuLSIF Change-Point Detection in Time Series Data*

The goal of the change-point detection technique is to discover control mode changes lying behind time series data. Recent efforts within this line of research introduced a new strategy, the relative unconstrained least-squares fitting (RulSIF), which was reported to outperform competitive non-parametric change-point detection approaches (Liu *et al.* (2013)). Let $y(t) \in \mathbb{R}^m$ a m-dimensional time series at time $t$ and $Y(t) = [y(t)^T, y(t+1)^T, ..., y(t+c-1)^T]^T$ be a subsequence (sample) of time series at time $t$ with length $c$ where $T$ represents the transpose, and $\mathcal{Y}(t) = [Y(t), Y(t+1), ..., Y(t+w-1)]$ a set of retrospective subsequence samples starting at time $t$, which forms a sliding window (SW) where $w$ is the window size. The RuLSIF change-point strategy considers computing the Pearson $(PE)$ divergence as a dissimilarity measure between two consecutive SWs $\mathcal{Y}(t)$ and $\mathcal{Y}(t+w)$. The higher the dissimilarity value, the more the point is considered a potential control mode change-point. RuLSIF uses the following approximator of the $PE$ divergence.

$$\widehat{PE}_\alpha = -\frac{1}{2n}\sum_{j=1}^{w}\hat{g}(Y'_j)^2 - \frac{1}{n}\sum_{i=1}^{w}\hat{g}(Y_i)^2 - \frac{1}{2} \tag{2.7}$$

where $\left\{Y_i\right\}_{j=1}^{w}$ and $\left\{Y'_j\right\}_{j=1}^{w}$ are samples from SWs $\mathcal{Y}(t)$ and $\mathcal{Y}(t+w)$ respectively. $\hat{g}(Y)$ is a density-ratio estimator, and is defined as:

$$\hat{g}(Y) = \sum_{l=1}^{n}\hat{\theta}_l K(Y, Y_l) \tag{2.8}$$

where $K(Y, Y')$ is a kernel basis function and $(\hat{\theta}_1, ..., \hat{\theta}_n)$ are parameters to be learned from data samples through RuLSIF optimization problem (Liu *et al.* (2013)).

### *2.4.5 Multivariate Non-Linear Polynomial Regression Analysis*

We consider the problem of estimating a non-linear relationship among the set of continuous variables $\mathcal{X}$ from time series data. Multivariate polynomial regression analysis (Agrawal *et al.* (2014)) can be performed on multidimensional data to model non-linear variables that depend on more than one variable by fitting data to high order multidimensional non-linear polynomials. For example, a quadratic non linear regression polynomial, which aims to capture non-linear regression relationship between one dependent variable $z$ and two independent (or dependent) variables $x$ and $y$ from time series data $x_i, y_i, z_i; i = 1, ..., n$ where n represents the number of data points, has the following form: $z = a_1 + a_2x + a_3y + a_4x^2 + a_5xy + a_6y^2$. In this paper, each variable $x_i \in \mathcal{X}$ with respect to time is regressed on powers of the variables in $\mathcal{X}$ while fitting the data into the high order non-linear polynomial regression model to find the best fit curve.

### *2.4.6 Cross Validation Mean Absolute Error*

: In order to assess the accuracy of the estimated non-linear equation, we use the cross validation mean absolute error (CVMAE). It considers leaving out one data point in a given signal and obtaining the parameters of the multivariate polynomial. Then the error in predicting the left out data point is estimated. We do this for all the samples in the signal and compute the mean error. The most accurate estimated non-linear relationship is the one with the least CVMAE. Sometimes, the CVMAE error can be infinite due an increased number of data points used in the estimation process. In this case, we use the root mean square error (RMSE) between the output of the estimated non-linear equation run test results and the actual output signal from the observed traces.

# Chapter 3

# SAFETY VERIFICATION OF AI-ENABLED CPS

## 3.1 Safety Aspects

ISO/IEC define safety of a system as freedom from unacceptable risk of injury or damage to humans, environment, and the system itself. In reliability engineering, safety is assured through the management of failures caused by physical component failures at run time. Functional safety aims at detecting random physical (mechanical, electrical, and electronic) or functional (logical) component failure during run-time and enabling corrective actions to avoid or reduce accident risk down to a tolerable level (Instruments (2011); Ladkin (2008)). For example, ISO 26262 provides standards for functional safety management of automotive applications through a definition of standards for a safety life cycle of the development and production of automotive applications (ISO (2011)). The challenge with functional safety is that it becomes impractical to determine every functional potential failure scenario because of the high complexity and non-determinism of AI-enabled CPS. In fact, any behavior of AI-enabled CPS that cannot be analyzed through system design and training would need to be monitored (Haugh *et al.* (2018)). Leveson describes safety as an *emergent property* from the compound behavior of the system' components interaction and needs to be assured throughout the life cycle of engineered systems (Leveson (2011)). System safety engineering is an important part of the overall system safety that focuses on optimizing safety through the application of system engineering, management principles, and systems theory. It aims at identifying and managing hazards at every stage of the system development life cycle (Leveson (2011); Leveson (1986)), as depicted in Figure 3.1.

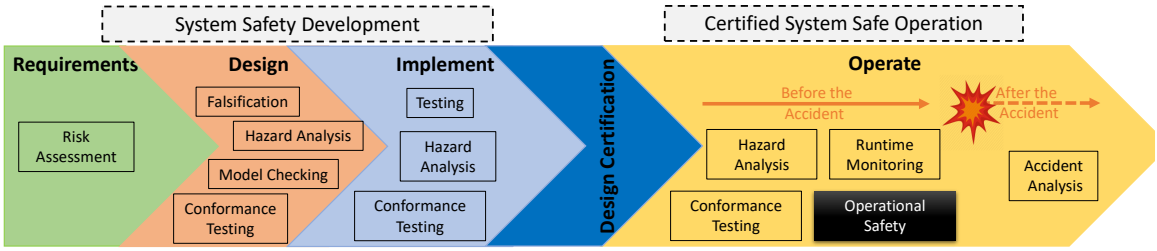


**Figure 3.1:** AI-enabled CPS Safety Life Cycle.

## 3.2 Safety Verification of AI-Enabled CPS

Many techniques have been developed for safety verification of CPS. In the following, we will discuss some of the general and commonly used safety verification techniques during the system safety engineering lifecyle.

### *3.2.1 Safety Analysis at Design Phase*

**Risk assessment and hazard analysis** is a crucial step in the development of safety-critical CPS and is applied to identify hazards, investigate their root cause, and embody a mitigating approach at the system's design stage. Traditional risk and hazard analysis techniques relate safety to a component reliability. For example, FMEA hazard analysis aims at identifying hazards caused by a chain of occurring events subsequent to a single component failure. FTA identifies leading factors of hazards as a combinations of components failures in a top-down search manner starting from undesirable hazardous events. Unlike traditional hazard analysis reliability-based techniques, STPA considers safety as an emergent system property that must be built into the design of the system. STPA uses the system's functional control diagram to analyze the interaction between the system components. It considers that hazardous situations are a result of inadequate control actions of the safety constraints, which can occur because of: 1- A required safety control action is not provided, 2- An unsafe control action is provided, 3- A control action provided too late or too early, and 4- A control action is stopped too soon or applied too early. All these techniques are performed

manually by engineers and require a detailed and complete knowledge of the CPS under scrutiny. However, the new technology being developed such as autonomous vehicles and IoT medical devices once deployed in the real world may exhibit unknown paths to hazards that were overlooked in the requirements and design phases due to the high complexities of the system, uncertainties of the human-in-the-loop behavior, and the effect of unobserved environment's internal variables. For example, every component of the autonomous driving Uber vehicle involved in the fatal crash causing a death of a pedestrian is claimed to be operating properly including the software, yet the environmental context was overlooked in the requirement and design phases.

Another group of verification approaches are called **Model Checking** where formal properties of the state of the system are verified via exhaustive state space analysis (Asarin *et al.* (2000); Kong *et al.* (2015); Frehse *et al.* (2011)). Formal models are suitable for modeling continuous and discrete dynamics of complex physical systems (Henzinger (2000)). One of the main challenges to formally verifying AI-Enabled CPS is the unavailability or incompleteness of the environment and user mental models. For example, dynamical variations between different and same individual as well as the nonlinear nature of the dynamics of the human body pose a major challenge in testing medical intelligent systems. Another challenge is the formal specification of the learning component of the AI-enabled CPS (Seshia *et al.* (2016)).

When it is not possible to formally verify the CPS model against the safety requirements, **Test-based Falsification** is used to check whether the model satisfies a property of interest (system's safety requirement) by searching for a behavior that violates it (Abbas *et al.* (2013)). However, for AI-enabled CPS, the task of mapping a given safety property (e.g. avoiding reward hacking) in terms of temporal logic can be very challenging (if that is even possible) (Amodei *et al.* (2016)).

### *3.2.2 Safety Analysis at Implementation Phase*

**Testing** is one of the fundamental approaches in the verification and validation of safety-critical systems at the implementation stage, which relies on ”test oracles”. An oracle or a test oracle is a reference for checking the actual system behavior observed during tests (Ammann and Offutt (2016)). Testing requires that the behavior of the system is unambiguous and well defined. Also, it can only prove the presence and not the absence of errors and require the existence of complete test cases. In well-established areas such as mechanical and physical systems engineering, different approaches were developed to address the discussed drawbacks. These approaches include, but are not limited to, black-box testing, automated test cases generation, heuristics for test case selection, and metamorphic testing (Myers *et al.* (2011); Zhang *et al.* (2014); Chen *et al.* (1998)).

System engineering involve a hierarchy of system models with rising layers of complexity to finally achieve a modified calibrated deployed system. For example, the effects of transport delays and controller sampling frequency are not considered in an initial specifications model. **Conformance testing** is a technique that aims at verifying correctness of a system model w.r.t the previously developed models and ensuring that the final deployed system follows the behavior of the initial specifications model. This verification is performed through a distance measure between simulated trajectories of the two models, which can consider differences in trajectories' timing (Henzinger *et al.* (2005)), trajectories' values (Girard *et al.* (2008)), or both (Abbas (2015)). Woehrle *et al.* presented a conformance testing method that relies on mapping the specifications of the system and its implementation generated traces to timed automata and verifying whether each generated implementation trace is included in the traces of the specifications timed automaton. As opposed to this conformance notion, other works define conformance testing as a *closeness measure* between an implementation and the specifications model, whose computation

solely relies on system traces (Abbas (2015); Araujo *et al.* (2018)). However, even for simple linear systems, providing guarantees about the conformance degree remains a challenge. In addition, the conformance testing result is a pass-fail output and when a failure occurs, safety engineers are confined to a violating pair of traces along with the different models of the system that may have been developed using different formalisms, languages, and tools. As such, performing a root-cause analysis using only the violating pair of traces is an arduous and cost-ineffective task.

### *3.2.3 Safety Verification at Operation Phase*

**Runtime monitoring** is a safety verification approach applied at the system's operation phase. It initially requires safety/correctness specifications to be expressed in terms of formal logic, which is not always viable since some of the system requirements can only be expressed in natural language. In addition, formally expressing and monitoring specifications while considering all complexities of the system (if that is even feasible) will impose an inevitable runtime overhead. It should also be noted that correctness specifications are developed based on assumptions about the dynamical partially observed environment and the human-in-the-loop behavior and these assumptions may not hold once the system is deployed in the real world environment. Thus, ensuring completeness and correctness of these specifications is a crucial and arduous task. An example of an offline runtime monitoring method used for flight operation is *exceedance analysis*. It consists on monitoring whether a set of parameters modeling hazardous event exceeds a certain threshold. The problem with such a technique is that it lacks contextualization, making it hard and time consuming for experts to analyze and interpret data correctly. State of the art runtime monitoring techniques may cope effectively with interaction complexities of traditional embedded systems of well-established areas, but may fall short when dealing with high and interleaving complexities of innovative novel AI-enabled CPSs.

Interactive, dynamic, and non-linear complexities of AI-based CPS may jeopardize their ”safety-assured” operation once deployed in the real world environment. I.e. AI-based CPS such as automonous driving vehicles and IoT medical devices are developed using an incomplete environment model due to its high complexities and are verified for safety using assumptions about the operator's behavior which is potentially subject to contingencies. As a result, the system may exhibit hazardous situations under a certain context that was never detected previously in the development or the certification phase of the CPS. Hence, the certified AI-enabled CPS may be subject to mishaps once deployed in the real world environment, a case example is the Boeing 737 max 8 crash (Johnston and Harris (2019)). This requires the development of new safety verification techniques capable of coping with the nature of new innovative technology being developed. **Operational safety** of AI-based systems is performed at the operation stage and aims at detecting deviations in the system's components (e.g human operator, environment, and intelligent agent) and verifying whether these deviations may jeopardize the safe operation of the system in the future. Early detection of operational deviations may be an efficient way to detect additional hazards' leading factors that were overlooked in the development phase and proactively prevent occurrence of accidents. We propose an operational safety verification technique that uses the proliferation of operational time series data generated during the AI-based CPS operation in the real world to gain a better understanding of the system's operation and enable the refinement of system's safety conclusions.

## 3.3 Arising Safety Issues

Operational safety has achieved an enormous progress in well established areas including aircraft and nuclear engineering. However, limited work has been performed in the field of cutting-edge AI-based systems, such as IoT medical devices and autonomous driving vehicles.

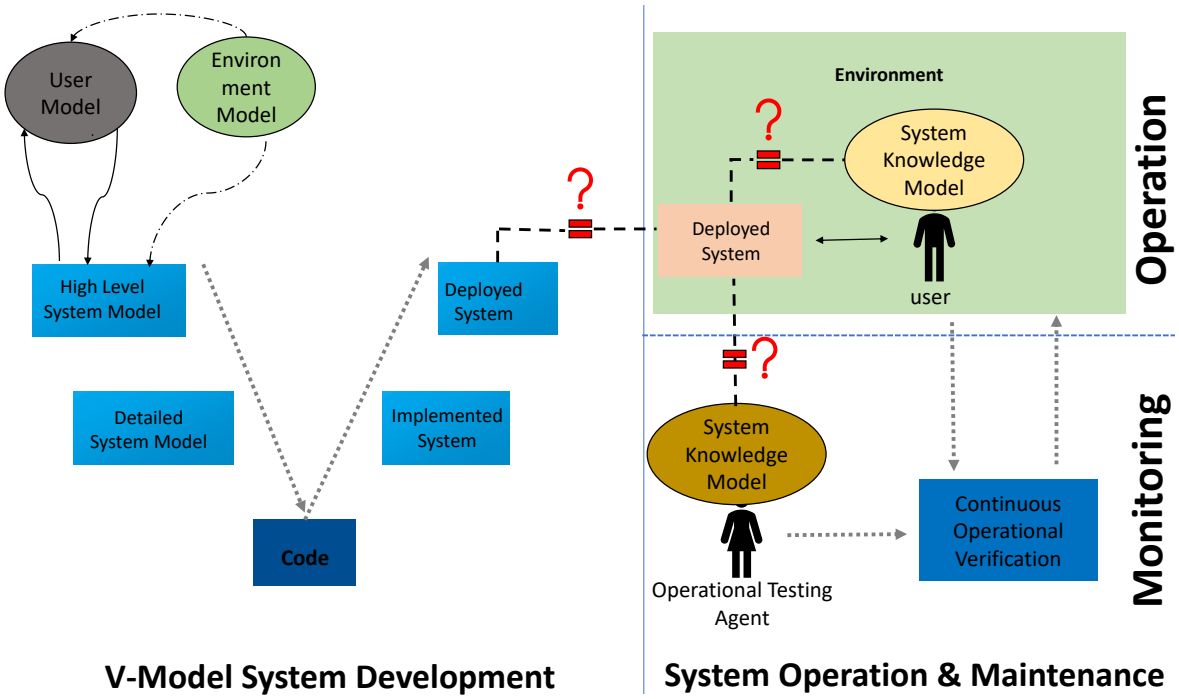


**Figure 3.2:** V-Model Based System Engineering and Operation.

In this section, we will provide a landscape of arising safety issues during the development, certification, and operation of AI-based systems.

**Ensuring system's knowledge consistency between system's components:**

One of the operational safety requirements of AI-based systems is a complete and consistent understanding of the system components and their interaction outcomes. I.e., dissonance of the system's operation knowledge between the operator, designer, operational testing agent, and legal agency is one of the potential leading factors of overlooked hazards. For example, pilots of the Boeing 787 max 8 failed at mitigating MCAS's failure because they were not properly au fait with the inner workings of the component. This decision of skipping pilots' training phase was due to production deadlines pressure in the competitive and aggressive environment of aircraft industry. The competition pressure and goals trade-off often lead manufacturers to migrate to a state of higher risk. This will potentially cause consumers to lose trustworthiness in manufacturers and trust in using AI-based systems, which is not desirable.

**Detection of corruption scenarios:**

New approaches for better and optimized control are developed at a fast pace whereas little attention is given to safety verification approaches. This will continue to intensify the gap between productivity and safety. For example, formally verifying neural networks (NN) performance and their interaction with the remaining components of the AI-based CPS remains a challenge for large NN controllers and formally verifying self-adaptive controllers for complex AI-based CPS is still an ongoing research problem (Ivanov *et al.* (2019)). At the certification process, manufacturers may tend to conceal the inner workings of some components for which safety verification approaches as yet unestablished.

**Monitoring learning agent's behavior:**

A learning agent has the ability to explore actions in order to optimize its objective function. However, safe actions in some contexts may be unsafe if performed in different contexts. The operational safety monitoring becomes crucial for safety-critical AI systems such as robots interacting with humans. As the functionalities and complexity of the AI-based CPS increases, it may become unfeasible to define the set of all possible safe actions for every context and system's state. Hence, an overlooked combination of the system's subcomponent states in a specific scenario is possible, which may pose the human-in-the-loop at risk (Levin (2018)). Hence, we need to monitor the operation of the learning agent and ensure that the agent does not perform unsafe exploratory actions. This safety problem has been referred to as *reward hacking* and has been considered in the context of a cleaning robot (Amodei *et al.* (2016)).

# Chapter 4

# OPERATIONAL SAFETY VERIFICATION OF AI-ENABLED CPS

The coverage problem for safety verification of AI-based safety critical CPS can potentially encounter combinatorial explosion due to the presence of significant number of interacting external sub-components and environmental conditions of use cases. In addition, a complete environment's model of complex dynamical physical systems is ultimately not available. As an example, the blood glucose system model used in the development of artificial pancreas does not encompass all variations including human behavior, mental state, and physical activity. The research question is whether these variations and incompleteness of the environment's model may guide towards misleading safety conclusions about the system and whether its specified safety assurances will hold once the system is operating within the real environment.

## 4.1 Operational Safety Verification Overview

We define **operational safety** as the detection of situations where the operation of the AI-enabled CPS in the field deviates from the safety certified operation of the system. The proposed approach as depicted in Fig. 4.1 consists on a hybrid automata (HA) mining algorithm, which takes the following inputs: 1- Input/Output traces collected from the operation of the AI-enabled CPS and 2- Limited information collected from the specifications document of the system. The output of the learning algorithm is a learned hybrid automaton comprising agent's control logic along with the environmental model of the CPS. The learned formal model is used to gain an insight into the safety of system by comparing the specifications of the system and the learned formal properties in order to detect the presence of inconsistencies. The proposed operational safety verification technique can also be

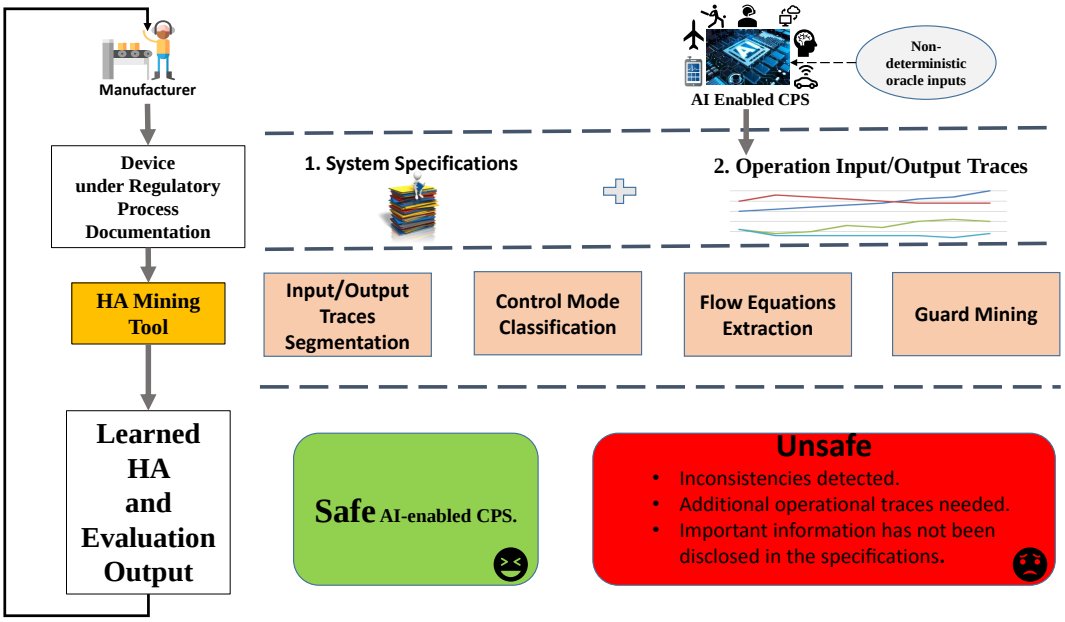


**Figure 4.1:** Operational Safety Verification: Proposed Approach.

utilized by certifiers or regulatory agencies to automatically investigate the safety of the system by detecting intentional or unintentional corruption scenarios during the certification process.

## 4.2 Learning Scenarios

There are two HA learning scenarios:

**Model-Agnostic:** In this case, a complete reference specifications model is not available. In this learning scenario, multiple learned HA with different number of agent's modes are learned to find the most approximative specifications model of operation of the AI-enabled CPS.

**Model-Aware:** A certified reference specifications model is available. In this case, the number of agent's modes is known and fixed a-priori. A model of the internal unobserved variables of the system is simulated to generated time series data for the unperceived variables.

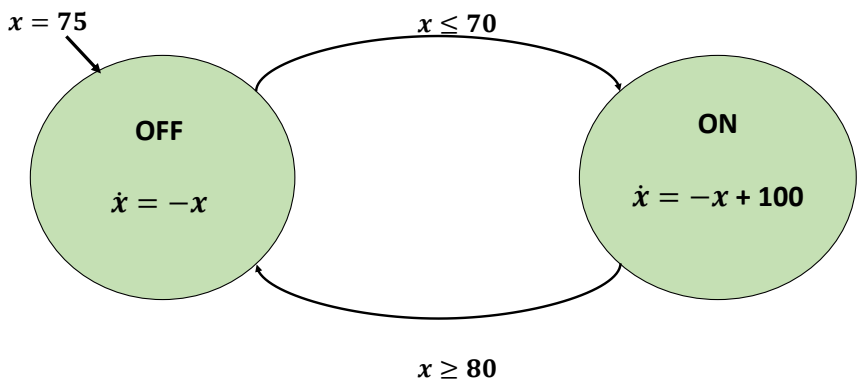


**Figure 4.2:** Temperature Control System Hybrid Automata.

## 4.3 Operational Safety Through HA Learning Overview

### *4.3.1 Ensuring Correctness of the Learned Model*

As shown in Figure 4.1, the learned HA model is used to analyze the safety of the AI-enabled CPS. In order to increase the accuracy of the learned model, we must ensure correctness of the HA learning technique through implementing the following requirements in each of the learning steps:

**I/O traces collection - Ensuring Data Unbias**: It is crucial that the data collected for learning (training and testing) is collected from different regions of interest of the operation of the AI-enabled CPS. For example, for medical devices, data collected from the operation of the system during different user's activities, mental changes, physical activity, day time, and environmental conditions.

**I/O traces partitioning - Avoiding False Positives**: One important initial step in learning an HA is to initially evaluate the type of data we possess. For example, the stationarity of time series is an important factor to assess change-point detection techniques which are more suitable for partitioning the I/O time series data. Even if it has proven that RuLSIF change-point detection technique reaches over $0.89\%$ accuracy in detecting changes

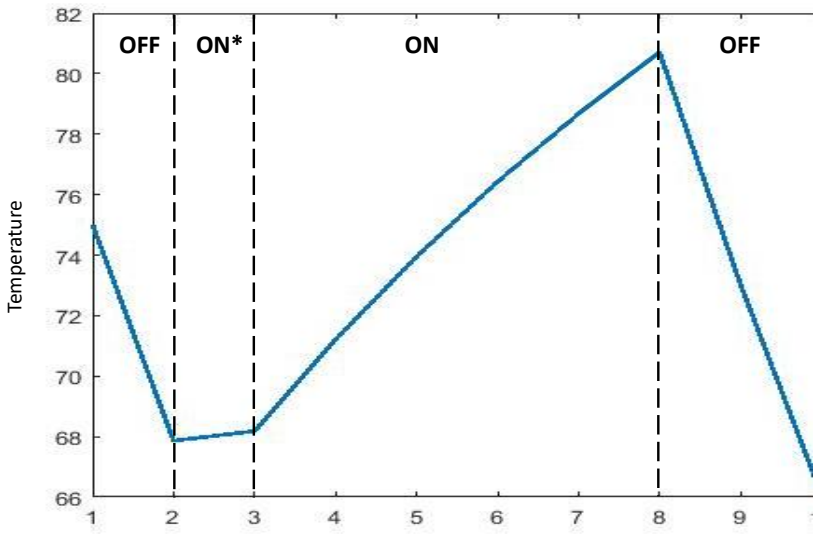


**Figure 4.3:** Temperature Control System Sample Execution.

for different non-stationary datasets (Aminikhanghahi and Cook (2017)), our data may still exhibit false positive changes due to a significant change in the unobserved internal environment's variables or the user's behavior that may have a drastic change effect on the controlled physical property. For example, if we consider the temperature control system with two agent modes $ON$ and $OFF$, as depicted in Figure 4.2. Figure 4.3 shows a trace partitioning of a sample operational trace of the temperature control system. $ON$ and $ON^*$ are partitioned as distinct agent's modes whereas the $ON^*$ mode is an operational behavior of the system in mode $ON$ affected by an external environment input. Thus, the trace partionning at potential mode change points is not definitive but will be refined in the agent's mode clustering/merging step of the HA learning technique.

**Agent Mode Clustering**: Figure 4.3 shows an example execution of the temperature control system in the real world. $ON$ and $ON^*$ are classified as distinct agent's modes from the previous partitioning step. In the following, we analyze the agent's modes clustering for the two HA learning scenarios:

In the model-aware learning scenario, the number of control modes is known and fixed a-

priori. For the temperature control system, $n = 2$. Hence, the agent modes clustering will lead to one of the following cases:

**1-** If multiple instances of $ON^*$ are present in the collected traces, then mode $ON$ and $ON*$ will be clustered as a distinct control mode $ON'$. Such operational behaviors of $ON^*$ will be reflected on the learned automaton through the parameters of the learned flow equations of mode $ON'$.

**2-** If only few instances of $ON^*$ are present in the collected traces, the $ON^*$ operational subtraces will represent noise and will not have an effect on the flow equations learning.

**Learning Flow Equations - Avoiding underfitting/overfitting**: We use cross-validation as a means to learn a model whose complexity leads to neither underfitting nor overfitting.

**Guard Mining**: We consider AI-enabled CPS with urgent guards (Minopoli and Frehse (2016a)). For example, the transition from mode $OFF$ to $ON$ and vice-versa occurs as soon as the guard condition ($x \leq 70$) and ($x \geq 80$) respectively is satisfied. If the temperature value reaches the guard value $70$ before the controller sampling frequency, then the mode change will occur when the temperature is slightly below $70$.

In the field of operation, novel physical phenomena that may have been overlooked in the system's development stage and new system's interactions and evolution will be reflected in the learned model. The learned HA can be used to re-evaluate the safety of system out in the field, refine safety conclusions, and provide safety feedback to the holistic system's stakeholders (standard organization, legal agency, user/operator, and manufacturer) in an iterative manner, as shown in Figure 4.4.

## 4.4 Safety Conclusions

The learned HA model accuracy is assessed by measuring the root mean square error (RMSE) between the generated data using the learned model and the testing data set, as

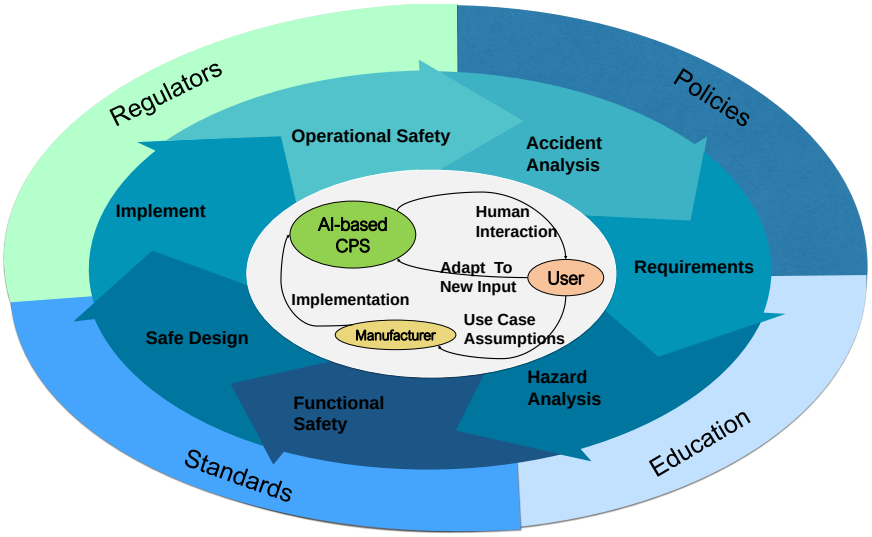


**Figure 4.4:** Holistic AI-enabled CPS Overview.

defined in Equation 4.1.

$$Error = \sqrt{\frac{\sum_{t=1}^{T}(\hat{x}_t - x_t)^2}{T}} \tag{4.1}$$

$\hat{x}_t$ represents the generated output variable value using the learned model at time instant t, $x_t$ represents the operational output variable value from the testing data set, and $T$ represents the total number of data points. Please note that we assume that the AI-enabled CPS has been certified for safe operation by a legal agency or that the manufacturer has already verified and proven the safety of the system. This means that the reach set of the reference formal model of the AI-enabled CPS does not intersect with the unsafe set, as shown in Figure 4.5. With respect to the safety evaluation using the learned model, there can be four distinct safety guarantee cases:

**Case 1)** The reach set of the learned model is an over-approximation of the specified system and encompasses the reach set of the specified system but it does not intersect the unsafe set. In such a case, we can guarantee that the system is operating within the safety envelope. This represents the *safety guarantee 1* of our proposed operational safety verification technique.

**Case 2)** The reach set of the learned model is an underapproximation of the reach set of the reference system or intersects it and learned system does not intersect the unsafe state. This

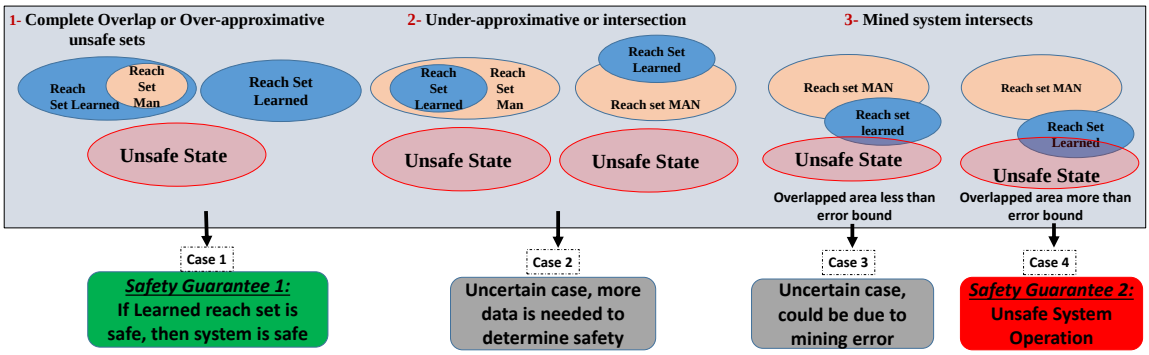


**Figure 4.5:** Operational Safety Analysis Result Cases.

is an uncertain scenario, because the learned model is incomplete. In this case, additional traces are needed to accurately learn the reference model.

**Case 3)** The reach set of the mined system intersects unsafe set but the area of intersection is within error bound of the learning technique. This case is also an uncertain case, because the intersection with unsafe set can be either due to a problem with the system operation or due to an error in the mining.

**Case 4)** The reach set of the mined system intersects unsafe set and area of intersection is greater than the error bound of the mining technique. In such a scenario, we can guarantee that the system is unsafe. This represents the *safety guarantee 2* of our proposed operational safety verification technique.

# Chapter 5

# LEARNING HYBRID AUTOMATA FROM I/O TRACES

## 5.1 Problem Statement

From I/O timeseries data collected from the operation of an AI-enabled CPS, automatically learn a HA representation of the system. The learning technique should:

- Infer linear flow equations representing the dynamics of the system.
- Infer controller modes.
- Infer guard and reset conditions.

## 5.2 HyMn: Linear Hybrid System Mining

Hybrid systems are versatile in modeling the interaction between the cyber and physical components of cyberphysical control systems (CPS) such as artificial pancreas (AP). They are typically used for analysis of safety of the human centric control systems which have serious consequences of failure. As such hybrid systems are parameterized and the variables often depend on the subject on which the control system is deployed. Traditionally, control systems are initially developed using average statistical estimates of the subject specific parameters. However, such excursions may lead to suboptimal designs. Publication I proposes HyMn, a hybrid system parameter estimation tool, where the subject specific parameters in a hybrid system are automatically learned from input/output traces collected from the run-time behavior of an AI-Enabled CPS.

**Data Collector/Generator**: Input/Output traces are collected from the operation of CPS. In this work, we assume that the traces are noiseless. Collected I/O traces are divided into

two sets: traces that are used for the inference technique and traces employed to verify the accuracy of the inferred HA.

**HyMn**: The HyMn algorithm takes the observed continuous states or controller inputs $\overrightarrow{x}$ and the controller outputs $\overrightarrow{o}$ as input and extracts a hybrid system of the form of the tuple $\{M, X, W, E, Inv, flow\}$ according to HA Definition (section 2.3). Figure 4.1 shows the main steps of the proposed HA mining technique:

*1- I/O Segmentation*: The first step is to segment the input output traces considering times at which there is a potential discrete mode change. Whenever there is a discrete transition due to a controller mode change, the controller output changes according to the decisions of the controller. There can be two types of controller outputs for a given mode: a) a step output, where after a transition the controller output changes levels and stays at a given level unless there is another transition, and b) the output is a linear function of the continuous state variables of the physical system. For both types of output, a sudden change in the slope of the controller output indicates change in mode. The timestamps $\overrightarrow{T} = \{t_1, t_2, \ldots t_k\}$ at which such jumps occur are considered to segment the controller inputs $\overrightarrow{x}$ and are marked to be potentially different controller modes. As shown in Figure 5.1, modes where controller output is constant is characterized by a sharp change in the differential of the outputs. HyMn employs peak detection algorithm on the differential of the outputs and derives the modes that have a constant level as controller output. This gives the time stamps of some of the mode transitions as shown in Figure 5.1. The time difference between two inflection points comprises of a controller mode.

*2- Mode Classification*: The second step is to determine the total number of controller modes and cluster the segments into equivalence classes corresponding to each controller mode. The controller strategy or the *jump condition* for each mode can be computed using the following two steps:

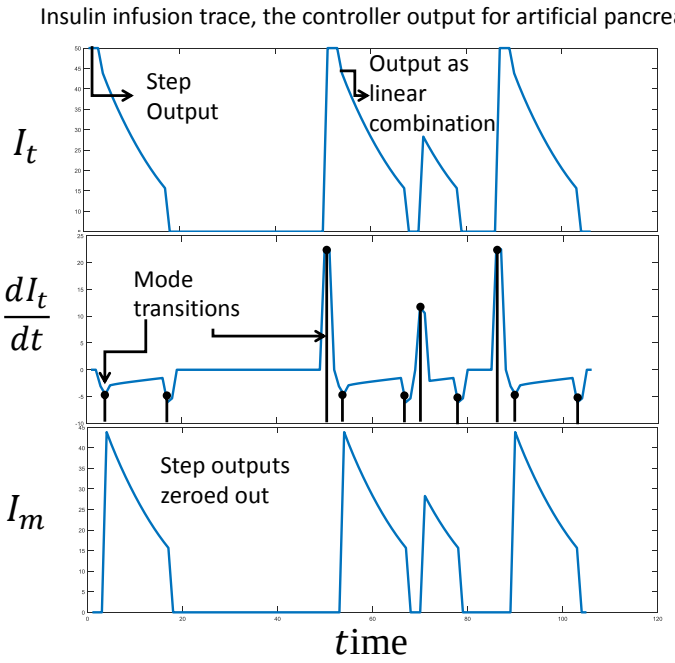


**Figure 5.1:** I/O Segmentation and Jump Condition Retrieval Example.

1- For each segment where the output differential is zero, the controller strategy is to provide a constant level of actuation obtained from the output trace $\overrightarrow{o}$.

2- For other segments, HyMn utilizes Fisher information theory to derive the linear equation connecting the controller output to the inputs.

For each output parameter, HyMn first uses Equation 2.4 to derive controller inputs whose linear combination gives the considered output. Then it uses Equation 2.5 to derive the estimator for the controller output. Segments are then grouped into classes based on the derived jump conditions. Each of this equivalent class is a composite mode and represents a unique strategy of the controller.

*3- Flow Extraction*: For each mode HyMn employs Fisher information and CRLB theorem to derive flow equations. The output of this re-classification are unique modes of the hybrid system, where two modes may have different jump conditions or flow equations.

*4- Guard Mining*: the guard mining approach takes as input the segmented input output traces where each segment is annotated with a controller mode. HyMn then considers ev-

ery possible mode transitions $(m \rightarrow m')$ and considers the values of the continuous state variables at the times of transitions, then develops the observation matrix $Go_{m\rightarrow m'}$. $Go_{m\rightarrow m'}$ is an $n \times d$ matrix, where each column corresponds to an observation of the continuous state variables at the time of transition from $m$ to $m'$, and there are $d$ such instances when the same mode transition is observed. In case $Go_{m\rightarrow m'}$ is full rank, HyMn obtains the rows that have constant values over all observation instances and the guard is expressed as a conjunction of equality condition $G_{m,m'} = \bigcap\{x_i = q_i\}$ on all such continuous state variables which have constant values, where $q_i$ is the constant value in the guard observation matrix. For non-rectangular guards, the guard observation matrix will not be full rank. In such a case we consider each continuous variable $x_i$ and express it as a linear combination of the other variables and a constant value, i.e., $x_i = A \begin{Bmatrix} x_1 & x_2 & \dots & x_n & 1 \end{Bmatrix}$, where $A$ is the coefficient matrix. We then use the same Fisher information-based analysis to derive the coefficient matrix $A$. The output of this step expresses guards in the form of equalities. However, we need the half planes which belong to each mode. This means for each transition $m, m'$ we need to find inequalities. For this purpose, for each transition observed, we consider the values of the differentials of the guard expressions. If we have a guard expression as $G_i = \bigcap\{x_i = q_i\}$, then if $\dot{x_i} > 0$, then the condition for $x_i$ is modified from $x_i = 0$ to $x_i \geq 0$. If the guard is expressed as $G_j = \bigcap\{x_i = \sum a_j x_{j\,j\neq i} + c_i\}$, then we consider the differential of the function $f = x_i - \sum a_j x_{j\,j\neq i}$. If from the observed I/O trace $\dot{f} > 0$ then the corresponding conjunction is modified as $x_i \geq \sum a_j x_{j\,j\neq i} + c_i$.

In the final step, for different observations of the same mode transitions $m, m'$ if there is a contradiction in any of the guard conjunction, then such conjunctions are eliminated from all guard expressions. This means that for two $m \rightarrow m'$ mode transition observations let us consider that the corresponding mined guards are:
$G_1 = \bigcap x_i \approx_1 c_i^1$ and $G_2 = \bigcap x_i \approx_2 c_i^2$, where $\approx_1, \approx_2 \in \{\geq, \leq\}$. Then the following rules must be applied:

-if $\approx_1 = \geq$ and $\approx_2 = \geq$, then the two terms can be replaced by the term $x_i \geq min(c_i^1, c_i^2)$,

-if $\approx_1 = \leq$ and $\approx_2 = \leq$, then the two terms can be replaced by the term $x_i \geq max(c_i^1, c_i^2)$,

-if $\approx_1 = \geq$ and $\approx_2 = \leq$, then the two terms can be eliminated from both the guard expressions if $c_i^2 \leq c_i^1$,

-if $\approx_1 = \leq$ and $\approx_2 = \geq$, then the two terms can be eliminated from both the guard expressions if $c_i^1 \leq c_i^2$,

Using the above-mentioned rules HyMn mines consistent guards from the observations.

**Verification and HA refinement**: Once the HA is generated through HyMn, its accuracy verification is crucial to the process. We compare collected I/O traces for verification to those generated using the inferred HA by calculating the root mean square error (RMSE) between the two sets of traces. The matching rate $\delta$ defines the accuracy of the inferred automaton that is evaluated according to some predefined rank $\alpha$ and used as a feedback to the HyMn Algorithm. The accuracy of the inferred automaton depends on the number and length of traces. For example, if the length of the trace is too short, then some of the modes can be missed, since these modes are not visible in the trace. HyMn algorithm uses the accuracy feedback to modify its inputs and refine the inferred automaton.

# Chapter 6

# EXPERIMENTS

## 6.1 HyMn Evaluation Results on Artificial Pancreas

### *6.1.1 Linearization of AP model*

The AP system is nonlinear in nature (see Section 2.1.1), hence it is necessary to linearize the system. To linearize the AP model we consider the difference in blood glucose, insulin concentration, and the interstitial insulin concentration. We consider a small time interval $h$ and rewrite $G(t+h) = G(t) + \Delta G$, $X(t+h) = X(t) + \Delta X$, and $I(t+h) = I(t) + \Delta I$. We can then ignore the non-linear terms that involve multiplication of $\Delta X$ and $\Delta G$. This results in the following linearized equations:

$$\begin{aligned}
\Delta\dot{X} &= -k_2(X(t) + \Delta X) + k_3(I(t) + \Delta I - I_b)), \\
\Delta\dot{G} &= -X(t).G - \Delta X G(t) - \Delta G X(t) \\
&\quad + k_1(G_b - G(t) - \Delta G), \\
\Delta\dot{I} &= -k_4 I - k_4 \Delta I + k_5 h G(t) - k_5 h G_0.
\end{aligned}$$

We simulate the hybrid system model of the AP for a given set of initial conditions to obtain input output traces. The simulations were carried out in Simulink and model based T1D simulator (Man *et al.* (2014)). From input output traces, we apply HyMn to obtain the hybrid system model and we compare the actual and inferred tuples for accuracy. In addition, we also evaluate the operation of the two hybrid system in terms of the results of reachability analysis. We use the SpaceEx (Frehse *et al.* (2011)) tool to derive the reach set for both the given and the inferred hybrid system and compare them to find differences.

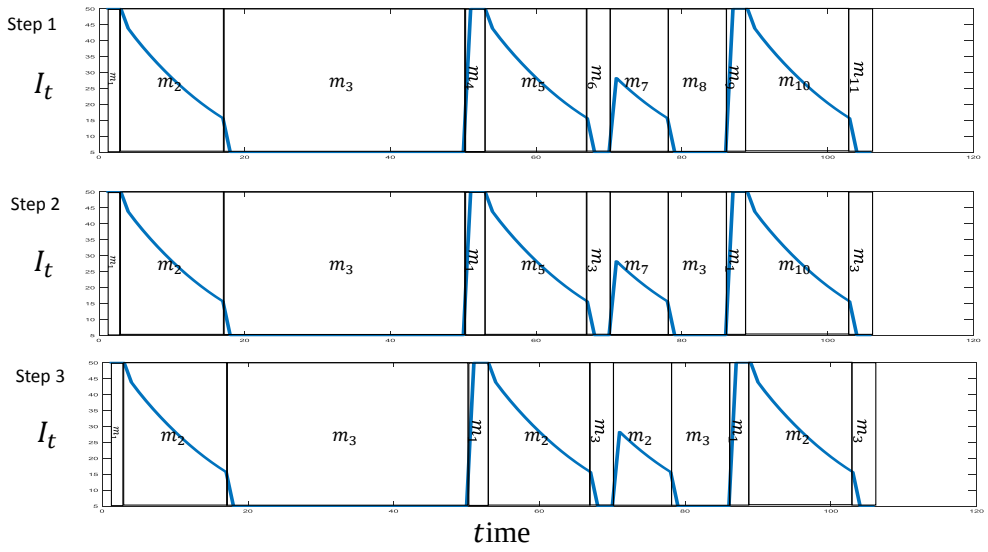


**Figure 6.1:** HyMn Mode Classification Execution Example for AP.

We use the AP example and show results of executing each step of HyMn. The first step of the HyMn algorithm is to consider the differential of the controller output $I_t$ as shown in the second part of Figure 6.1. Employing peak detection, the HyMn algorithm initially considers that there are as many modes as the number of peaks. From Figure 6.1, the HyMn will consider the mode set $M = \{m_1, m_2, \ldots m_{11}\}$, i.e., 11 distinct modes. The HyMn mode classification algorithm then considers the absolute value of $I_t$ to distinguish between modes where $I_t$ is constant or $\frac{dI_t}{dt} = 0$. As a result of this operation, HyMn finds that $m_1 = m_4 = m_9$ and $m_3 = m_6 = m_8 = m_{11}$. Hence it reduces the mode set to $M = \{m_1, m_2, m_3, m_5, m_7, m_{10}\}$. It then considers the segments where $I_t$ is not constant as shown in $I_m$ in Figure 6.1. It employs Equation 2.4 and 2.5 to derive the linear relation of $I_t$ with $G$ and $I$. The analysis results in the same equation for modes $\{m_2, m_5, m_7, m_{10}\}$: $I_t = 0.5B_g + 44.75$ Since the modes have the same linear equation relating controller output to the inputs, HyMn considers that $m_2 = m_5 = m_7 = m_{10}$. Hence, the total mode set is $M = \{m_1, m_2, m_3\}$. HyMn then considers all the mode transition times and develops the jump conditions. From the input output trace we see that $J_{m_1 \to m_2} = \{\{G; I; I_t\}, \{G; I; 0.5G + 44.75\}\}$, $J_{m_2 \to m_3} = \{\{G; I; I_t\}, \{G; I; 5\}\}$ ,

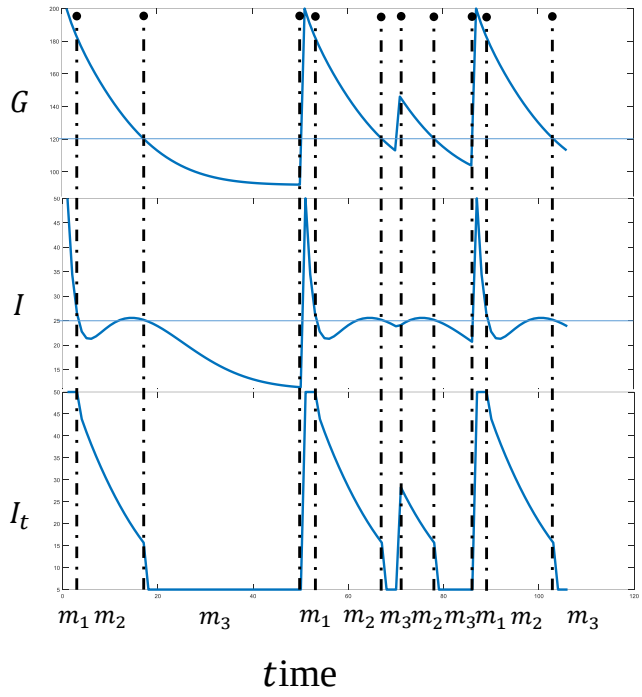


**Figure 6.2:** Input Traces ($G, I$) and Output Traces ($I_t$) for the AP System

$J_{m_3 \to m_1} = \{\{G; I; I_t\}, \{G; I; 50\}\}$, $J_{m_3 \to m_2} = \{\{G; I; I_t\}, \{G;\ I;\ 0.5G + 44.75\}\}$, and $J_{m_2 \to m_1} = \{\{G;\ I; I_t\}, \{G; I; 50\}\}$. The next step in HyMn is to find the flow equations in every segment using the traces of $I_t$, $G$, and $I$ (Figure 6.2).

The linearization method described in Section 6.1.1 results in a constant bias that depends on the sensed blood glucose, insulin concentration and interstitial concentration values. Hence the bias changes over time. However, the Cramer Rao based estimation only derives coefficients for the difference in the values of the continuous variables. Thus, it could not accurately estimate the time varying bias. To circumvent the problem, we add the bias to the estimated constant obtained using Cramer Rao bound each time instant. Based on Equation 2.4 and 2.5, HyMn derived the following set of equations:

$\Delta\dot{X} = 45.84\Delta X + 6.89^{-6}\Delta I + 2.47^{-8} - 0.021X(t) + 0.00001(G - 10)$

$\Delta\dot{G} = 80.77\Delta X + 45.49\Delta G + 1.21^{-5} - X(t).G + 0.031(Gb - G)$

$\Delta\dot{I} = 45.59\Delta I + 3.95^{-8} - 0.3I + 0.0033hG - 0.0033hG0.$

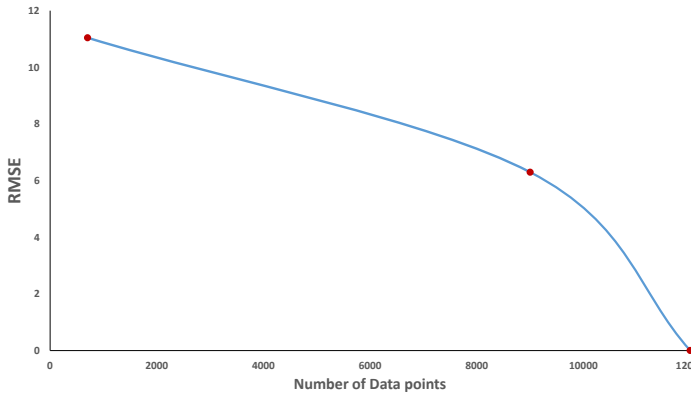


**Figure 6.3:** Variation of the RMSE W.R.T the Increase of Number of Collected Traces

For every segment we obtained the same equation resulting in the conclusion that $m_1, m_2, m_3$ are unique modes and are not composite. The next step is to determine the guards. Let us consider the transition from $m_2$ to $m_3$. The guard observation matrix can be obtained from the traces in Figure 6.2 as in Equation 6.1.

$$Go_{m_2 \to m_3} = \begin{Bmatrix} 118.07 & 118.07 & 118.07 & 113.14 \\ 24.8 & 24.8 & 24.8 & 23.8 \end{Bmatrix}. \tag{6.1}$$

$$Go_{m_1 \to m_2} = \begin{Bmatrix} 177.16 & 177.16 & 171.15.07 \\ 22.76 & 22.76 & 21.4 \end{Bmatrix}. \tag{6.2}$$

The matrix $Go_{m_2 \to m_3}$ is a full rank matrix. Hence, we only consider the row that is constant. However, there is no such row. Hence, we have four different expressions for the guard corresponding to each observation in Equation 6.1. HyMn then considers the derivative of $G$ and $I$ at the transition point and uses the rules discussed in Section 5.2. For all the transition points, from Figure 6.2, we see that $\dot{G} < 0$ indicating that the guard for transiting from $m_2$ to $m_3$ is $G \leq 118.07$. However, $\dot{I}$ had both positive and negative values resulting in a contradiction. Thus, the guard expression that uses $I$ is eliminated from the guard expression. The same operation results in the following guards: $G_{m_1 \to m_2} : G < 177.1$, $G_{m_3 \to m_2} : G > 118.07$, and $G_{m2 \to m1} : G > 177.1$.

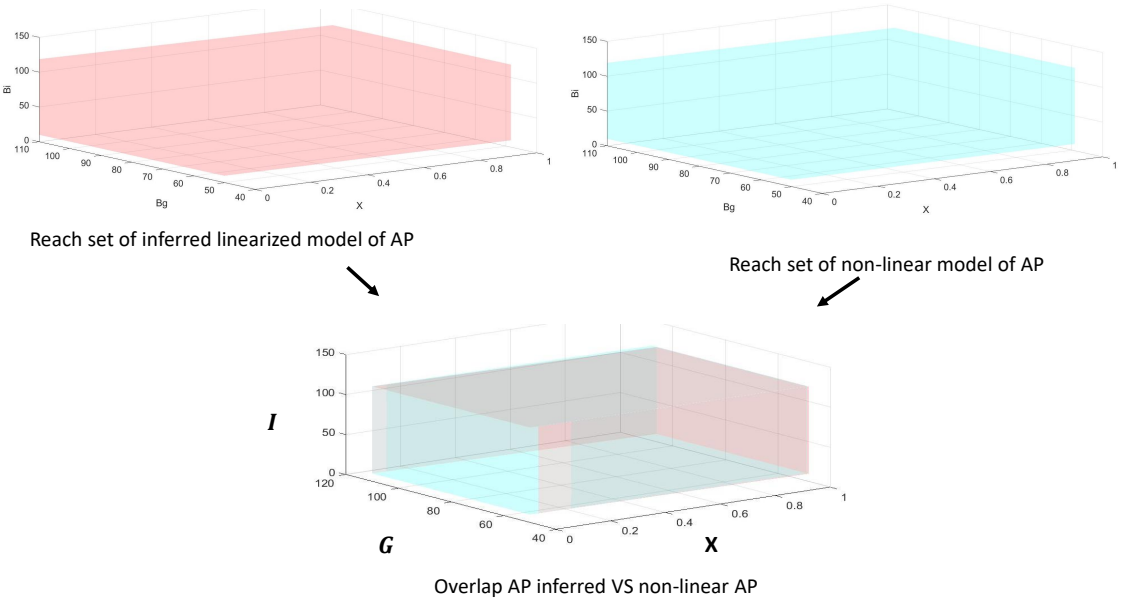


**Figure 6.4:** Reach set of the Non-Linear Model of AP vs Reach Set of Inferred Linear Model of AP.

The invariant set computation was trivial since it only required partitioning of $\mathcal{R}$ using the rectangular guards. The inferred AP hybrid system is almost similar to the given hybrid system of AP described in Section 2.3 (they only differ in the parameters' values $(k_1, k_2, k_3, k_4, n)$ of the continuous dynamics). We also used the inferred and the given hybrid system in reachability analysis using the SpaceEx tool. Figure 6.4 shows reach sets for both hybrid models starting from the same initial conditions set.

**Benefits of using HA learned patient-specific parameters**:

We executed the AP system with two parameter configurations: a) taking statistical average parameters obtained from a large pool of T1D subjects (Man *et al.* (2014)), and b) obtaining the patient specific parameters for a given subject using the HyMN approach. We kept the blood glucose profile the same for both the configurations. Figure 6.5 shows the plot of the insulin concentration over time for both the configurations. The results show that using patient specific parameters in this scenario reduces total insulin delivery by 5.29%.

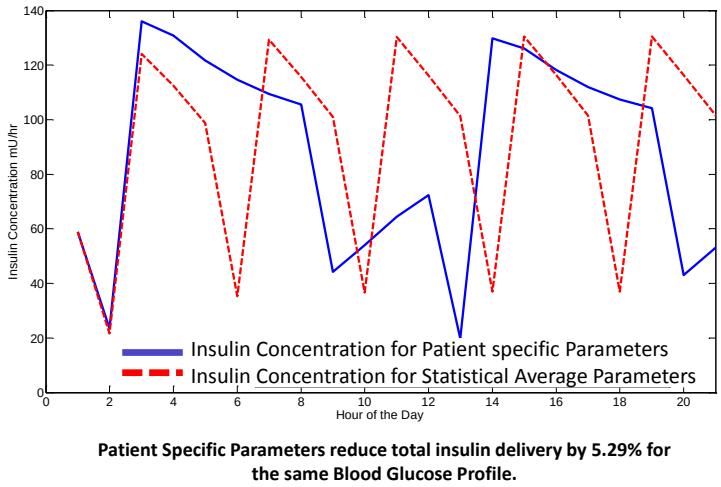


**Figure 6.5:** Comparison of Insulin Delivery using Patient Inferred Parameters from HyMn Versus Using Statistical Average Parameters.

This is a significant result because the aim of any controller is to achieve normal glucose levels with minimal insulin infusion.

### *6.1.2 Applications of HyMn*

HyMn has been applied to mine hybrid system models of gestures for ML-based cooperative gesture learning applications using a coalition of hand-shape recognition technology and explainable kinematic models. The mined model is utilized to provide explanation for recognition of continuous events, as depicted in Figure. 6.6. The technique was applied on 60 users for 20 ASL gestures and results show that the mined parameters of the kinematic equations can represent each gesture with precision of 83%, recall of 80% and accuracy of 82% (Publication III).

### *6.1.3 Limitations of HyMn*

When applying HyMn to traces collected from the operation of Medtronic Minimed 670G (Appendix A) through a collaborative work with Mayo clinic, it was subject to the following limitations:

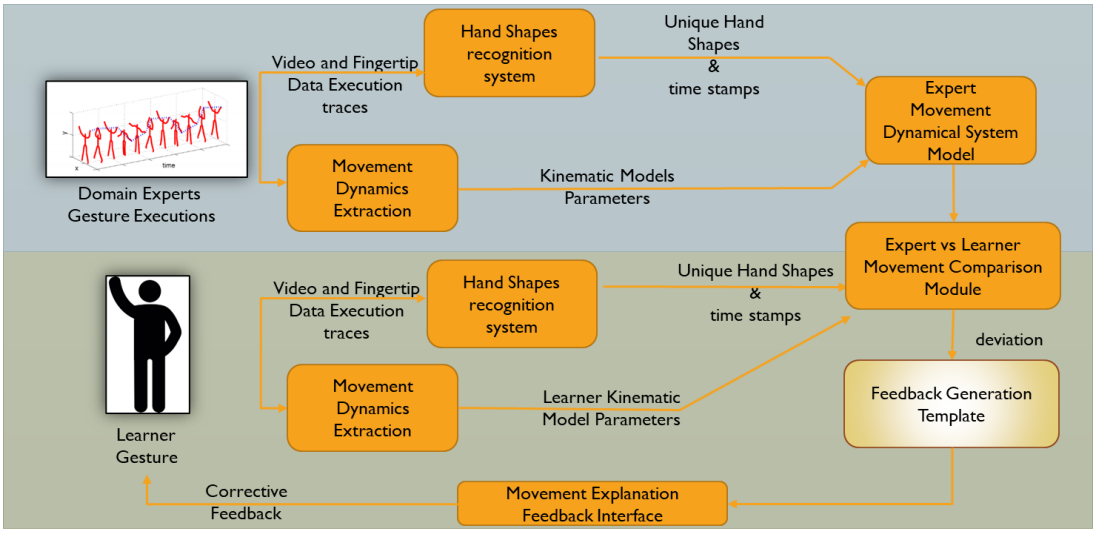


**Figure 6.6:** HyMn Application on Co-Operative Learning Systems

**Non-linearities**: The glucose subsystem model used in the control logic of the Medtronic Minimed 670G is non linear in nature. Hence, we need to update HyMn to learn non-linear differential equations.

**Input/Output traces Segmentation**: In HyMn, a sudden change in the slope of the controller output indicates a mode change. This is not necessarily true in the case of the Medtronic Minimed 670G due to the presence of smooth control mode changes and signal variations due to different physiological and operating conditions. This led to the generation of many false positive and false negative control mode changes.

**Complexity**: The control logic implemented in Medtronic Minimed 670G is a combination of PID and an insulin feedback (IFB) algorithm (Ruiz *et al.* (2012)). Since industrial control systems are moving towards employment of advanced and complex strategies of control logic such as learning agents or adaptive control which uses feedback from the environment to update the control logic or the environmental model used by the controller to estimate the current state of the environment (Leveson (2011); Rajkumar *et al.* (2010)).

Hence, additional and deeper safety analysis techniques must be developed as it is difficult for current safety verification methods to keep up with the increasing pace of technological change (Rajkumar *et al.* (2010); Scherer (2015); Kim and Kumar (2012); Lee *et al.* (2015)).

**Hidden control variables**: As shown in Figure A.1, the only observed signals from the operation of the Medtronic 670G are the insulin infusion rate $I_D$ and the CGM reading $S_G$. All remaining control and system variables are hidden. Hence, HyMn should be adapted with capabilities of learning an approximative hybrid automata model of the system using only observed I/O traces. In case the manufacturer does not allow full access to detailed specifications of the requirements and design of the system, HyMn should be able to learn the most approximative HA model of the system using only I/O traces and limited knowledge about the system.

**Limited amount of traces**: HyMn should be able to detect if additional traces or knowledge is mandatory for the operational safety certification process. This allows interaction with the manufacturer in an iterative operational safety verification of the system under regulatory process, as shown in Figure 4.1.

**Safety Comparison Metric**: HyMn should implement a safety assessment metric to detect dissonance between operational data and the safety assured design of the system.

Chapter 7

# N-HYMN: LEARNING NON-LINEAR HYBRID AUTOMATA

## 7.1 N-HyMn Algorithm

N-HyMn focuses on learning a non-linear HA representations of an agent-based CPS in the form of $< \mathcal{X}, \mathcal{M}, \mathcal{F}, \mathcal{E}, \mathcal{G}, \mathcal{R} >$ according to Definition of hybrid automata in Section 2.3. The input set consists of:

- set of input variables $\mathcal{I}$ and set of output variables $\mathcal{O}$, where $\mathcal{X} = \mathcal{I} \bigcup \mathcal{O}$.

- $Traces$, time series data of every continuous I/O trace collected from the operation of the CPS.

- $n$, the number of control modes in $\mathcal{M}$ of the CPS.

- $MinPts$, the minimum number of instances of the same control mode transition. In the Model-Aware scenario, $n$ is known a-priori. In the Model-Agnostic scenario, $n$ is not known, and can be varied to learn multiple HA representations, thus facilitating the identification of the most accurate learned HA.

In the following, we present the outline of the algorithm:

1) Segment each trace at time stamps of potential control mode changes:

**for each** $tr$ **in** $Traces$

$Timestamps$**(**$tr$**) = RuLSIF_Change_Point_Dectection(**$tr$**)**

The output of the $RuLSIF\_Change\_Point\_Dectection$ function is a list of timestamps [$start$, $ts_1$, $ts_2$, ..., $ts_f$, $end$], where $start = 0$ and $end = last\_value(tr)$ represent the start and end time value of each trace, respetively.

2) Cluster unique control mode changes:

2.a) Collect data at each potential control mode change timestamp for all traces:

$clustering\_data$ =[]

**for each** $tr$ **in** $Traces$

**for each** $ts$ **in** $Timestamps$

$clustering\_data$.**append(**$y$**(**$ts$**))**

$y(ts)$ is a $m$-dimensional time series sample at time $ts$.

2.b) Density-based clustering of unique control mode changes:

$id$ = **DBSCAN(**$clustering\_data, MinPts, n$**)**

The output of DBSCAN is a $p \times 1$ vector $id$ containing cluster indice of each data point, where $p$ represent the number of data points in the $clustering\_data$ matrix.

Each cluster represents a control mode change $e_i \in \mathcal{E}$.

3) Learn the reset condition of each control mode change:

3.a) For each $cluster$, collect data sample before and at the time occurence of the corresponding mode change:

**for** $i$ **in** $id$

**transition_data(**$cluster$**).** **append([**$y$**(**$ts-1$**),** $y$**(**$ts$**)])**

3.b) For each $cluster$, find the reset condition:

**MultiVar_Polynomial_Regression(**$transition\_data$**(**$cluster$**))**

The output of $MultiVar\_Polynomial\_Regression$ is a linear polynomial regression of the reset condition $\mathcal{R}_{cluster}$.

4) For each $cluster$, learn the guard condition:

4.a) For each $cluster$, collect input data sample one step and two steps back in time at the time occurence of corresponding mode change:

**guard_data(**$cluster$**).** **append([**$y_{in}$**(**$ts-1$**),** $y_{in}$**(**$ts-2$**)])**

$y_{in}(ts)$ is a $q$-dimensional time series sample at time $ts$, where $q$ represents the number of input continuous variables.

4.b) For each $cluster$, learn the polynomial regression of the guard condition $\mathcal{G}_{cluster}$:

**MultiVar_Polynomial_Regression(**$guard\_data$**(**$cluster$**))**

5) Learn non-linear flow equations for every unique control mode:

**MultiVar_Polynomial_Regression(**$flow\_data$**(**$mode$**))**

$flow\_data$**(**$mode$**)** is data comprised between a unique control mode interval.

## 7.2 N-HyMn Implementation Details

### *7.2.1 I/O Segmentation:*

Every control mode change is not always conditioned on an external input variable. For example, the CLAP Suspend before Low control mode is conditioned on the predicted value of blood glucose $BG(t)$, which is an internal variable of the CLAP. In addition, if the effect of the control mode changes that are conditioned on internal continuous variables is smooth without any drastic change on the observed continuous variables of the CPS, then these control modes will go undetected. Thus, it is important to perform control mode change detection on operational I/O traces to find control mode changes that are conditioned on internal continuous variables. On the other hand, an actuator action of an agent-based CPS does not always imply a control mode change. For example, within the Auto Basal Mode of the CLAP, the insulin infusion rate (actuator action) may change at every time step. In the Model-Aware scenario, this step will allow detection of unspecified control mode changes. In the Model-Agnostic scenario, this step is essential since a specifications reference model is not available. We apply RuLSIF change-point detection method on the CLAP input/output traces, as shown in Fig. 7.1, to determine time stamps at which there is a potential control mode change-point. At every sliding time step, the initial sample data from SW1 is removed and an initial sample from SW2 is included at the end of SW1 while a new sample data is added at the end of SW2. The $PE$ score between SW1 and SW2 will start increasing when SW2 starts including samples after a meal intake and a bolus infusion

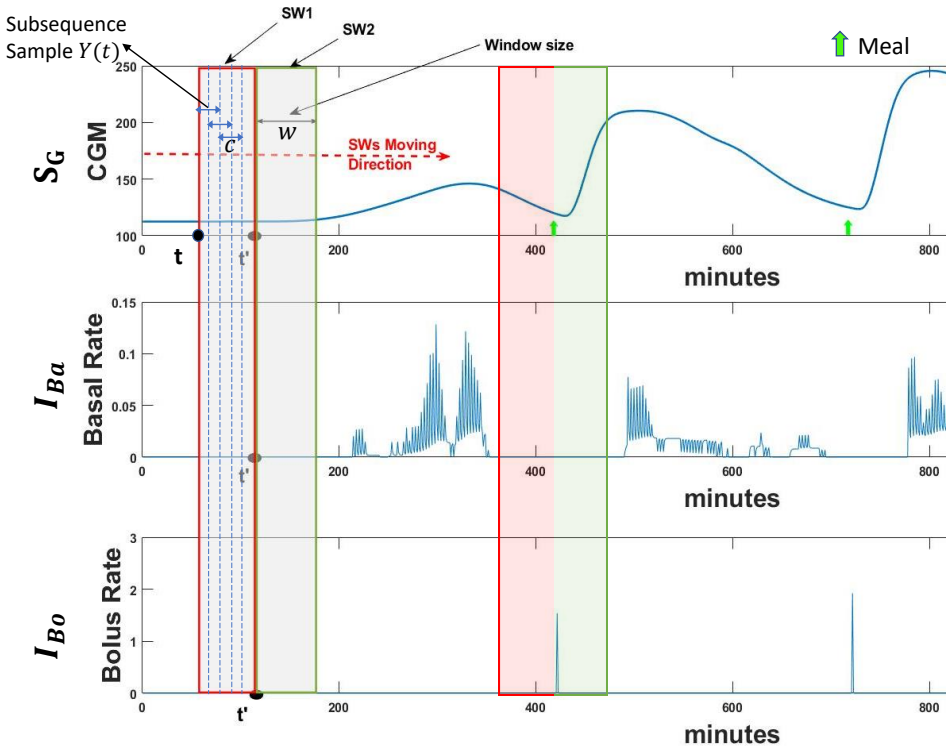


**Figure 7.1:** Input Trace (CGM readings $S_G(t)$) and Output Traces (Basal $I_{Ba}$ and Bolus $I_{Bo}$ Infusion Rates) of CLAP.

(if we assume that a meal input and the corresponding bolus infusion start their effect on the human body directly after occurence). The average $PE$ score will reach its peak when SW2 contains mostly samples from the effect time interval of the meal bolus injection on the human body, while SW1 does not contain samples from the effect time interval. Thus, the window size should be the estimated peak time interval of the actuation on the physical environment. For CLAP, the insulin pump uses rapid acting insulin which starts to take effect on the subject 10 minutes after injection with a peak time of 30 minutes. In our experiment, the window size is 20 minutes.

### *7.2.2 Control Modes Clustering:*

The second step is to cluster unique control mode transitions. N-HyMn employs density-based spatial clustering of applications with noise (Matlab R2019 DBSCAN) on time series

I/O traces to find clusters of unique control mode transitions. The output of this clustering is unique control modes. In this experiment, $MinPts$ is set to 10 which means that at least 10 instances of the same control mode transition should be available in each cluster. N-HyMn can learn at most $((p \textbf{ div } MinPts) - 1)$ HA representations with $\delta$ control modes where $\delta \in [2, (p \textbf{ div } MinPts)]$ and $p$ represents the total number of available control mode change instances, assuming that any CPS has at least two distinct control modes. For example, if we have $n = 50$ control mode change instances and $MinPts$ is set to 10, thus N-HyMn can learn at most 5 clusters, where each cluster contains 10 instances.

### *7.2.3 Reset Conditions Learning:*

For the reset condition mining, collected I/O operational traces should be classified into two sets: a training set for learning the HA components and a testing set for verifying the accuracy of the learned component. For each cluster of unique control mode transition, N-HyMn derives the corresponding reset condition using the following strategy: 1-If the output variable value after the control transition occurs remains constant for every mode transition in the cluster, then the reset condition is a constant value of actuation. 2- If the controller output is varying, then the reset condition is a linear function connecting outputs variables (e.g CLAP output traces $\{I_{Ba}(n), I_{Bo}(n))\}$ with inputs (e.g CLAP input traces $\{SG(n), CHO(n)\}$). N-HyMn find multiple reset conditions for the same control mode transition considering every possible combination of input variables since we do not know which input variables make a significant contribution in estimating the linear function. N-HyMn then finds $\sum_{\vartheta=1}^{\theta} \frac{\theta!}{\vartheta!(\theta-\vartheta)!}$ possible reset conditions for the same control mode transition, where $\theta$ represents the total number of input variable and $\vartheta$ the number of variables being chosen at a time. For that, N-HyMn considers the values of the input variables $\mathcal{I}$ at times of mode transitions then develops a $n \times d$ observation matrix $\mathcal{O}_{\mathcal{R}_{m_i,m_j}}$, where each column corresponds to an observation of the continuous state variables at the time of the mode

transition from mode $m_i$ to mode $m_j$ and there are $d$ such instances when the same mode transition is observed. N-HyMn applies the multivariate non-linear polynomial regression on the observation matrix generated for each cluster to estimate the possible reset conditions for every unique control mode transition. N-HyMn then chooses the reset condition with the least CVMAE to be considered the most accurate reset condition. This CVMAE has to be extremely small ($\approx 0$), otherwise N-HyMn delivers an error message that no accurate reset condition can be learned for the corresponding control mode transition (a feedback message is provided that in this case additional $Traces$ may provide additional information to the reset condition mining process).

### *7.2.4 Guard Conditions:*

As discussed in the definition of hybrid automata in Section 2.3, there can be two types of guard constraints: a) rectangular constraints, which are expressed as simple threshold on input continuous state variables (e.g $S_G(n) > 180$ ) and b) diagonal constraints, which are expressed as linear combination of continuous state variables (e.g the guard condition $((s_{xa} - s_{xb} > 10)$ & $(s_{ya} - s_{yb} > 10))$ of the driver assist systemDuggirala *et al.* (2015a)). N-HyMn considers the values of the input control variables $\mathcal{I}$ one step and two steps back in time at times of mode transitions (the mode transition occurs as a result of a guard condition that was satisfied one step back in time, but it was not satisfied two steps back in time). Then, N-HyMn develops a $n \times d$ observation matrix $\mathcal{O}_{\mathcal{G}_{m_i,m_j}}$, where each column corresponds to an observation of the input control variables one step and two steps back in time at times of mode transitions and there are $d$ such instances when the same mode transition is observed. For rectangular guards, the columns of the observation matrix will be linearly independent and hence will have full rank. Thus, in case $\mathcal{O}_{\mathcal{G}_{m_i,m_j}}$ is full rank, N-HyMn obtains the rows that have constant values over all observation instances and the guard is expressed as a conjunction of equality condition on all such continuous state variables which have constant

values. Hence the output of this stage is an expression of the form $\mathcal{G}_{m_i,m_j} = \bigcup\{x_i = q_i\}$, where $q_i$ is the constant value in the guard observation matrix. For non-rectangular guards, the guard observation matrix will not be full rank. In such a case, N-HyMn considers each continuous variable $x_i$ and express it as a linear combination of the other variables using the multivariate polynomial regression method. i.e., $x_i = A \begin{Bmatrix} x_1 & x_2 & \dots & x_n & 1 \end{Bmatrix}$, where $A$ is the coefficients matrix. The output of this step expresses constraints in the form of equalities. Thus, we consider the values of the differentials of the guard condition; that is to say if we have a guard expression as $\mathcal{G}_{m_i,m_j} = \{x_1 = q_1\}$. We check $\dot{x_1}$ for all observed instances before the mode change, then the condition for $x_1$ is modified to $x_1 \geq q_1$ if $\dot{x_1} > 0$ or to $x_1 \leq q_1$ if $\dot{x_1} < 0$. If the guard is expressed as $\mathcal{G}_{m_i,m_j} = \{ax_1 + bx_2 = c\}$, then we consider the differential of the function $f = ax_1 + bx_2$. Based on $\dot{f}$, the corresponding relation operator is modified accordingly. Finally if the sign of $\dot{x_i}$ is not consistent among all observed instances, it confirms that no consistent guard conditions exist for the partial learned hybrid automata. Using the above-mentioned rules, N-HyMn mines consistent guards from the observations.

### *7.2.5 Learning Flow Equations:*

N-HyMn applies multivariate non-linear polynomial regression analysis to estimate the non-linear ordinary differential equations that represent the environment's predictive model. As shown in Fig. B.1, the hybrid automaton of the AP control system uses the non-linear Bergman minimal model to estimate the predicted value of blood glucose level $B_G(t)$.

### *7.2.6 N-HyMn Complexity*

The runtime of the I/O segmentation step in HyMn uses RuLSIF algorithm for change detection (Liu *et al.* (2013)). The RulSIF divides each I/O trace for a variable into windows of a given size $w$. It then takes two consecutive windows and performs a density difference

estimate. This step requires $O(w^2)$ computation of the density function. Each sample point needs a weight parameter and this involves solving a convex optimization problem using gradient descent with complexity $O(2B/e)$, where $B$ is the upper bound on the absolute values of the variable and $e$ is the error tolerance (Bubeck *et al.* (2015)). For each window the complexity of the density estimate based difference computation is $O(w^2 + 2wB/e)$. This density estimate is performed for $k/w$ windows, where $k$ is the total number of samples in the I/O trace of a variable. Hence for $n$ dimensions, the complexity of the RuLSIF algoritm is $O(\frac{nk}{w}(O(w^2 + 2wB/e)))$. The worst case runtime for control mode clustering and mining reset condition is $O(n^2)$. Flow extraction has a complexity of $O(kdim^2)$, $dim$ is the largest order of monomial used in the multivariate polynomial regression model (Agrawal *et al.* (2014)). For the guard mining, for every transition we would have on an average $k/|T|$ segments or in the worst case $k$ segments. The guard observation matrix will then be of size $m \times k$ in the worst case. The guard mining has an observation matrix of worst case size $m \times k$, hence has a complexity of $O(max(m,k)^3)$. Hence the overall computational complexity of the N-HyMn algorithm is: $O(\frac{nk}{w}(O(w^2 + 2wB/e)) + n^2 + max(n, k/|T|)^3 + kdim^2 + k/|T| + max(m,k)min(m,k)^{1.4} + mk + max(m,k)^3)$. If we consider that $k >> m$, then the overall complexity of the N-HyMN reduces to $O(kn^2)$.

# Chapter 8

# N-HYMN: EXPERIMENTS

Through our collaboration with Mayo Clinic, we collected data from the operation of Medtronic Minimed 670G for 60 Type 1 diabetic subjects over the period of three months. In this section, we experimentally investigate the performance of N-HyMn using operational I/O time series traces in two scenarios:

**Model-Agnostic:** No reference specifications model is available. In this case, the learned HA is the most approximative specifications model of operation of the Minimed 670G.

**Model-Aware:** A certified reference specifications model of Medtronic Minimed 670G presented in Appendix A. In this case, we implement the reference specifications model to generate traces for internal input variables using operational I/O time series data and the certified reference specifications model. Our aim is to detect discrepencies between the learned operational model and the approved reference specifications model of Medtronic Minimed 670G, as depicted in Fig. 8.1.

## 8.1 Model-Agnostic Learning Scenario

The control structure of the Medtronic Minimed 670G artificial pancreas system is composed of control loops like the one shown in Fig. B.1. In general, a controller provides control actions in the form of insulin infusion delivery rate $I_D(t)$ to control blood glucose level in the human body and to enforce the behavior of the controlled process. The control algorithm determines the control actions to perform while the process model is utilized to make these control decisions. We collect input traces (CGM readings $S_G(t)$ and meal carbs announcement $CHO(t)$) and output traces (basal infusion rates $I_{Ba}(t)$ and bolus infusion rates $I_{Bo}(t)$) since these are the only observed time series traces we can collect from the

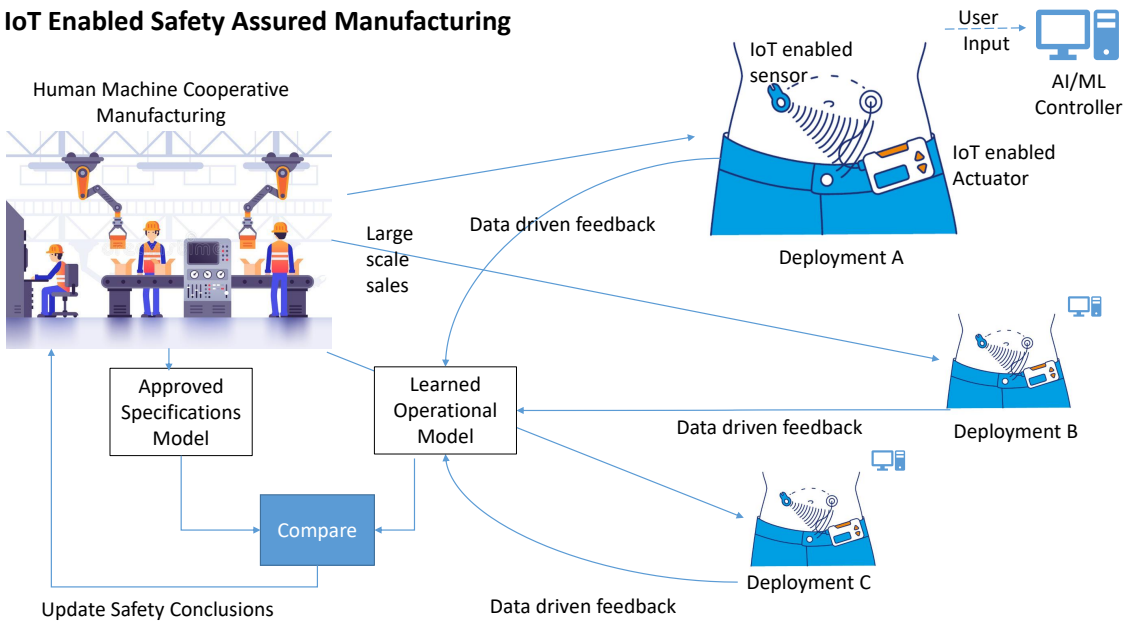


**Figure 8.1:** IoT Enabled Manufacturing of Industry 4.0 Applications.

operation of the Medtronic Minimed 670G system in the field. We apply N-HyMn to the collected I/O time series traces. The I/O segmentation step of N-HyMn shows that there is a potential mode change 10 minutes to 30 minutes following a bolus injection, as shown in Fig. 8.2. We use N-HyMn to learn six HA representations with a number of control modes $\delta \in [2, 7]$.

The reset condition operation learns linear functions connecting each output variable with input variables for every cluster of unique control mode change. The only observed operational I/O traces the Medtronic Minimed 670G artificial pancreas system are $\{S_G(n)$, $CHO(n)$, $I_{Ba}(n)$ , and $I_{Bo}(n)$. However, using limited preliminary knowledge about CLAP, we assume that this control system may not only depend on the observed input signals $S_G(t)$, $CHO(t)$ but also on other traces that can be easily deduced from the operational I/O traces. Thus, we set the eight following variable traces $\{S_G(t)$, $CHO(t)$, $S_G(t-1)$, $I_D(t-1)$, $I_D(t-2)$, $I_D(t-3)$, $I_D(t-4)$, $\frac{dSG}{dt}(n)\}$ as input variables where $I_D = I_{Ba} + I_{Bo}$ and $I_D(t)$ as output variable. Let's consider the control mode transition

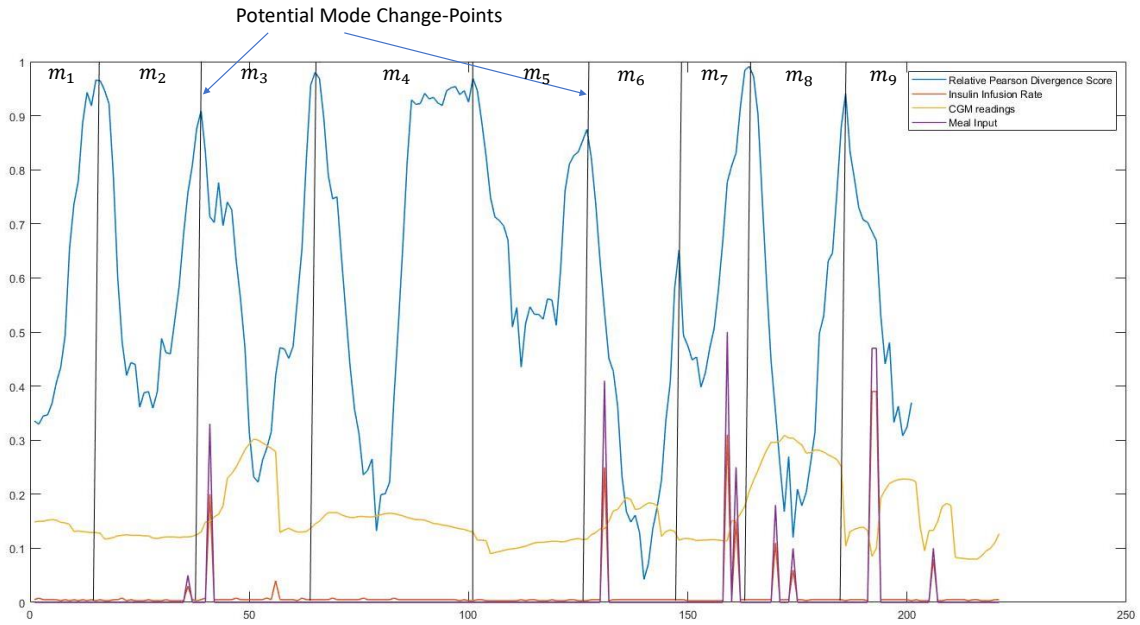


**Figure 8.2:** Pearson Divergence Score Trace of an I/O Operational Trace of CLAP.

from $m_1$ to $m_2$ of $HA_3$ with four control modes as shown in Fig. 8.3 and the output $I_{Bo}$ which can be linearly dependent on [1,8] input variables. For ease of explanation, we restrict the input variables to $\{S_G(t), CHO(t)\}$. We use observed traces for the first seven days of simulation as learning traces while the remaining nine days of simulation are used for accuracy testing. The columns of the guard condition observation matrix $\mathcal{O}_{\mathcal{G}_{m_1,m_2}}$ represent $CHO(t-2)$, $CHO(t-1)$, $S_G(t-2)$, and $S_G(t-1)$ respectively. We limit the input variables to $CHO$ and $SG$ for ease of explanation.

$$\mathcal{O}_{\mathcal{G}_{m_1,m_2}} = \begin{pmatrix} 0 & 3.33 & 104.74 & 104.69 \\ 0 & 2.66 & 169.30 & 169.36 \\ 0 & 3.33 & 108.86 & 108.84 \\ 0 & 3.33 & 101.97 & 101.93 \\ 0 & 3.33 & 102.59 & 102.56 \\ 0 & 3.33 & 83.79 & 83.81 \\ 0 & 3 & 122.75 & 122.67 \end{pmatrix}$$

Table 8.1 shows the CVMAE of the learned reset condition $I_{Bo}$ for the control mode transition $m_1$ to $m_2$ of $HA_3$ as a linear function of the output variable $I_{Bo}$ and one input variable

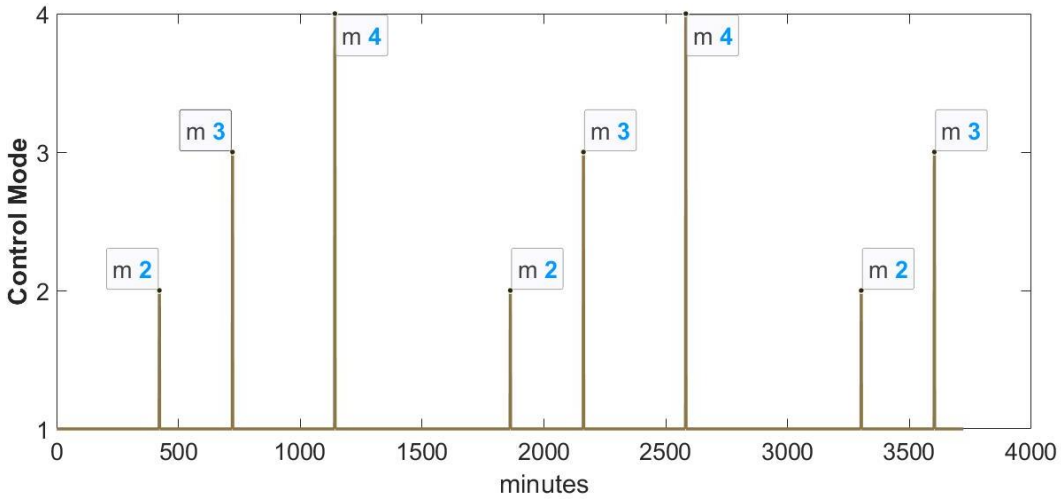


**Figure 8.3:** Density Based Clustering of Unique Control Modes.

| Variables | $S_G(n)$ | $CHO(n)$ |
|---|---|---|
| $S_G(n)$ | $0.1461(U/h)$ | $8.3466^{-16}(U/h)$ |
| $CHO(n)$ | $8.3466^{-16}(U/h)$ | $7.3282^{-16}(U/h)$ |

**Table 8.1:** Cross Validation Mean Absolute Error (CVMAE) of Possible Reset Conditions $I_{Bo}$ from $m_1$ to $m_2$ in $HA_3$.

($S_G(n)$ or $CHO(n)$) or two input variables ($S_G(n)$ and $CHO(n)$). $I_{Bo} = f(CHO)$ represents the most accurate reset condition equation of the control mode transition from mode $m_1$ to mode $m_2$ of the hybrid automata representation $HA_3$ with a CVMAE of $8.3466^{-16}$ Unit/hour (U/h), as shown in Table 8.1.

N-HyMn infers the following guard condition for the control mode transition $m_1$ to $m_2$ in

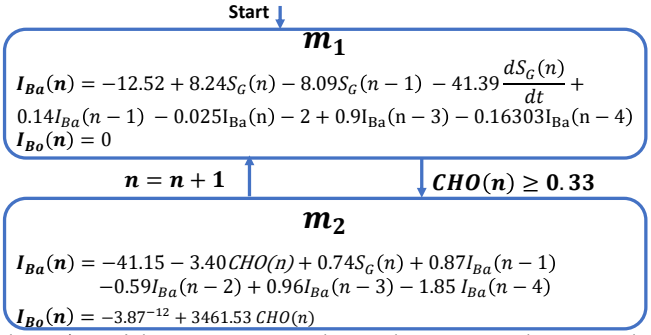


**Figure 8.4:** Partial $HA_1$ with Two Control Modes $m_1$ and $m_2$ and Output Variables $I_{Ba}$ and $I_{Bo}$.

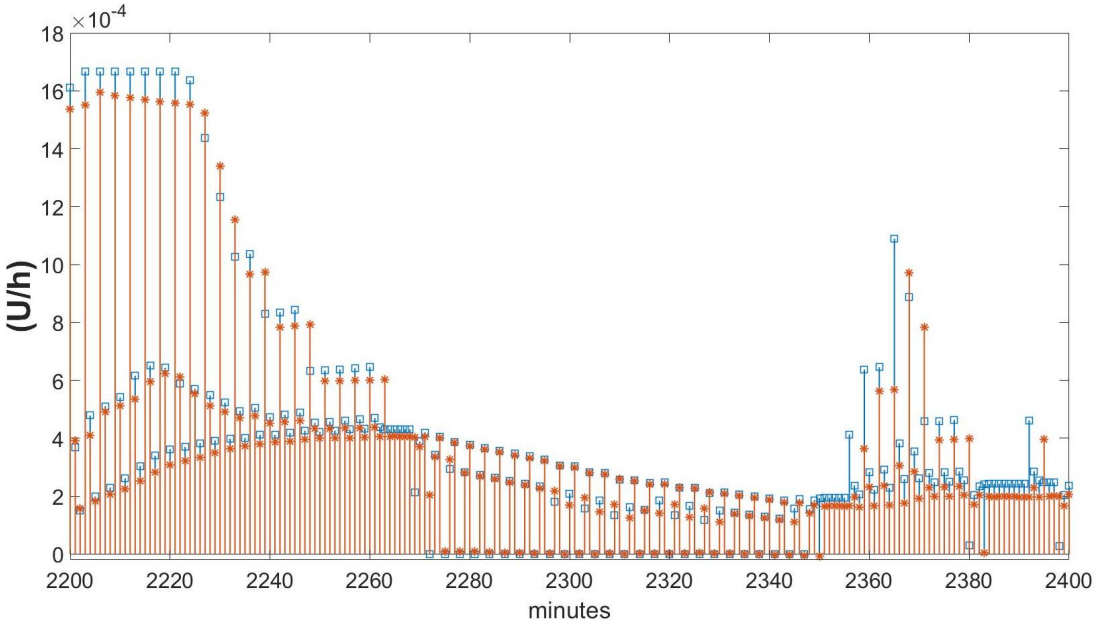


**Figure 8.5:** Comparison between Learned $I_{Ba}$ (Star) and Observed $I_{Ba}$ (Square) in Control Mode $m_1$ of $HA_1$.

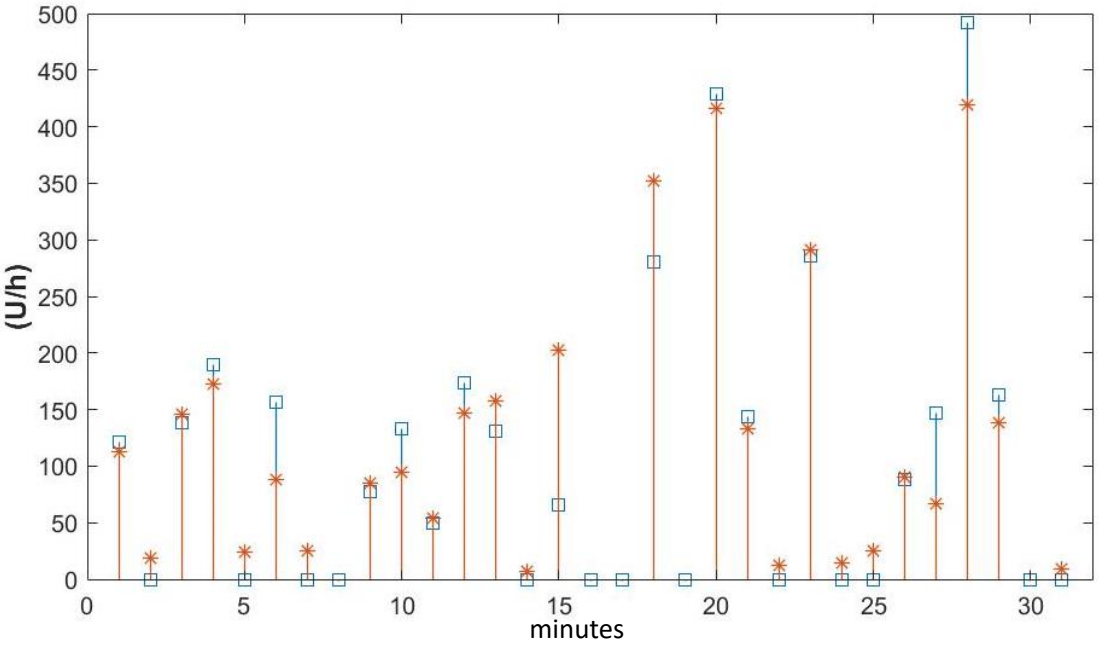


**Figure 8.6:** Comparison between Learned $I_{Ba}$ (Star) and Observed $I_{Ba}$ (Square) in Control Mode $m_2$ of $HA_1$.

$HA_3$: $\mathcal{G}_{m_1,m_2} = CHO(n) \geq 2.66$ . Fig. 8.4 depicts the partially learned hybrid automaton $HA_1$ with control modes $m_1$ and $m_2$, guard conditions, and control output variables $I_{Ba}$ and $I_{Bo}$. Fig. 8.5 shows a 4 hour comparison from the second day of testing traces between the run test results of the learned equation $I_{Ba}$ of control mode $m_1$ in $HA_1$ and the actual $I_{Ba}$ observed trace while Fig. 8.6 shows the same comparison for 252 hours of data in control mode $m_2$. In order to identify the most accurate HA from the 6 learned HA representations, we run the different learned HA representations in C2E2 (Duggirala *et al.* (2015b)) and compare their run test results to operational I/O data of the testing set. We perform this comparison by calculating the RMSE between the output of each HA to the observed output traces. The user defines the RMSE error bound according to their application. In our experiment, we set the error bound to 2%. All six candidate HA representations satisfy the RMSE error bound. The HA candidate model with two control modes $HA_1$ represents the most accurate observed operational model of the Medtronic Minimed 670G. The RMSE error using $HA_1$ for five test days is reported in Table 8.2. The mean RMSE is $7.763^{-5}$ with stdev. $2.25^{-6}$.

| Test Day | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| RMSE (U/h) | $0.79^{-4}$ | $0.75^{-4}$ | $0.8^{-4}$ | $0.75^{-4}$ | $0.77^{-4}$ |

**Table 8.2:** $HA_1$ RMSE Per Day.

## 8.2 Model-Aware Learning Scenario

We apply N-HyMn to historical operational data collected from the usage of Medtronic Minimed 670G insulin infusion pump. We use UVA/Padova simulator to implement the agent's control structure of the Medtronic Minimed 670G, presented in Appendix A, to generate traces for internal variables using operational I/O traces collected from the usage of Medtronic Minimed 670G insulin infusion pump and the approved operational model of

the Basal Auto mode of Medtronic Minimed 670G shown in Fig. A.1. N-HyMn mines a HA model that does not match every I/O operational data result. At this point, we realized that N-HyMn was able to match the initial I/O operational traces that use the initial static PID+IFB model, but it fails at estimating the remaing I/O time series data. This can be a results of an adaptation of the parameters of the PID+IFB controller that occurs periodically after a specific time window. However, the FDA certification documents do not mention the self-tuning component, which we later found in a patent application by the manufacturer (Patent No.: US 8,777,924 B2 Jul. 15, 2014). As reported in the patent application, the PID controller possesses a self-tuning model that updates the controller gain parameters using information collected from the CGM sensor data. If a pre-selected condition is satisfied within a predefined time window, the controller gains ($K_P$, $K_I$, and $K_D$) are increased by a factor (1+$\alpha$) at the end of each time window, where $\alpha$ is the gain update value. The predefined time window size and gain variation update model vary from one embodiment to another and a very limited release of details of the inner workings of the self-tuning component was provided which makes it extremely challenging to extract the specification for the tuning mechanism from I/O traces. As an initial input, we provide N-HyMn with an array of different values of $\alpha$ ranging from -5 to 15 to correctly estimate I/O operational time series data. Fig. 8.7 shows the gain variation update ($\alpha$) for one T1D Minimed 670G user over a period of 200 min that allowed N-HyMn to correctly learn the exact operational I/O values. However, these results need further investigation and give us an insight into our future work. Poor documentation about the control logic of the control system is one of the compounding factors leading to fatal accidents since it prevents practitioners (e.g Boeing 737 Max 8 pilots) from identifying the problem early and stopping the occurrence of catastrophic accidents (e.g MCAS). For learning the glucose subsystem model of the human model (dynamics evolution equations) used by the agent to predict blood glucose value, we assume the availability of plasma glucose concentration $B_G(t)$, plasma insulin

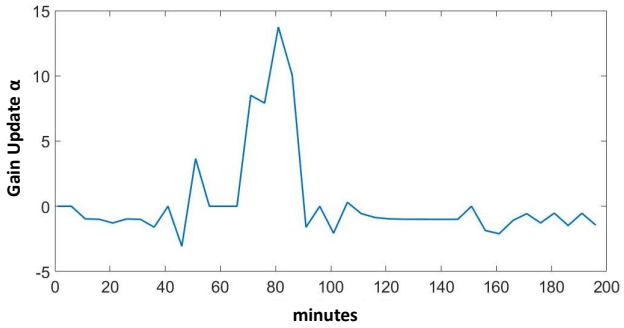


**Figure 8.7:** Controller Gain Parameter Variation Updated Values $\alpha$ for a T1D Minimed 670G Subject.

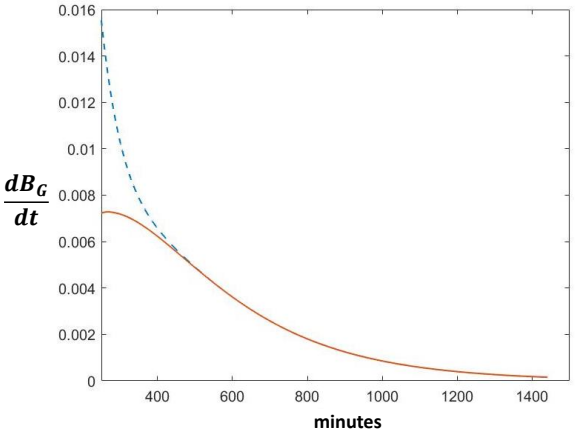


**Figure 8.8:** Learned Rate of Change of $B_G(t)$ (Dashed) and Observed $\frac{dB_G}{dt}(t)$ (Solid) in Control Mode $m_1$ and $m_2$ of $HA_1$.

concentration $I(t)$, and interstitial insulin concentration $I_{EFF}(t)$, that we generated using the UVA/Padova simulator (which uses the same glucose subsystem model). We collect these traces from the UVA/Padova T1D simulation test results and use them to learn the Bergman minimal model (glucose subsystem model) of the simulated T1D subject Bergman *et al.* (1979) . For each control mode, N-HyMn infers the following non-linear ordinary differential equations for the two control mode $m_1$ and $m_2$ (Equation (8.1)).

$$\begin{aligned}\frac{\mathrm{dB_G}}{\mathrm{d}t} &= 0.247X(t) - 1.367^{-5}B_G - 0.001B_G(t)X(t) - 7.31^{-8}B_G(t)^2 + 0.01 \\ \frac{\mathrm{dI_{EFF}}}{\mathrm{d}t} &= -0.015I(t) - 1.09^{-8}B_G(t) - 8.774^{-6}B_G(t)I(t) + 1.596^{-11}B_G(t)^2 + 1.863^{-6}\end{aligned} \tag{8.1}$$

As shown in Fig. 8.8, the learned glucose model would underestimate glucose values for $t \leq 420$ minutes since it does not take into account the counter-regulation response to hypoglycemia ($B_G < G_b$) Man *et al.* (2014).

# Chapter 9

# RELATED WORK

Model synthesis and mining has been a topic of significant interest. The existing works on this topic can be studied under the following five categories:

## 9.1 Timed Dynamical Model Mining

Works of Sethia et al. Jin *et al.* (2015), Fainekos et al. Hoxha *et al.* (2018) and others (Prabhakar *et al.* (2018); Narayan *et al.* (2018); Nenzi *et al.* (2018)) have considered mining temporal constraints for Signal temporal logics (STLs) and Metric temporal logic (MTLs) from simulation traces of a CPS. Fainekos et al. also show a method for extracting unsafe parameter range of MTL models of CP control systems Hoxha *et al.* (2018). There are two drawbacks of the proposed solutions: a) STLs and MTLs in general can answer questions related to temporal alignment of events. However, they cannot be used to model non-linear temporal dynamics. b) The recent works in this domain focuses on simulation traces of CPS. As such a significant advantage is full observability of the system. In real life deployments the system is often under-determined. For example, although the PID controller of the Minimed 670G system uses five different parameters to compute the actuation output only one is observable Steil *et al.* (2011). Bortolussi et al. Nenzi *et al.* (2018) proposes a technique to recover STL specifications using real life noisy data, but their system still is fully observable.

## 9.2 Hybrid Model Synthesis

Several previous work have proposed algorithms and frameworks for mining or synthesizing a hybrid automata. Lyde and Might presented an approach for synthesizing hybrid automata from control code of cyber-physical systems, which is then applied in model-checking safety verificationLyde and Might (2013). Minopoli and Frehse developed a tool for translating a Simulink model into a formal verification model -hybid automaton- that is used for reachability analysis safety verification Minopoli and Frehse (2016b). However, our work differs in that we infer a hybrid automaton from input/output operational traces.

## 9.3 Hybrid Model Mining

Medhat et al. proposed a framework for mining Mealy automata from black-box systems using only execution traces. Their framework is limited to systems that 1) exhibit input changes in the form of step functions and these changes are assumed to have an instantaneous effect in the output trace and 2) guards conditions are time-based, which is often not observed in practice Medhat *et al.* (2015). Balakrishnan et al. presented an algorithm to determine a maximum-likelihood hybrid system model using only continuous output of the system Balakrishnan *et al.* (2004), but this work assumes that guard conditions are independent of the continuous state variables which limits the class of hybrid automata that can be inferred using the proposed technique. Blackmore et al. extended Balakrishnan et al. work by including autonomous mode transitions which are conditioned on the continuous state, but their approach assumes that the guard conditions are given Blackmore *et al.* (2007). Ly et al. presented a high computational complexity multi-modal symbolic regression algorithm to infer non-linear symbolic expressions that model the behavior of a dynamical system from unlabeled time series data Ly and Lipson (2012). Unlike N-HyMn, the learning

algorithm infers non-linear dynamics evolution of a dynamical system with no closed-loop control feedback and requires the number of modes to be fixed a priori. Moreover, the behavior of the system is defined as a strict input/output relationship, as opposed to N-HyMn where behaviors are represented by differential equations. In addition, some of the related approaches require a priori knowledge of number of discrete modes Santana *et al.* (2015) Ly and Lipson (2012), as opposed to N-HyMn. Niggemann et al. share identical motivation for the automated learning of hybrid system's behavioral model through HyBULTA and application of the learned model to detect anomalies in the overall system behavior (Niggemann *et al.* (2014), Niggemann and Lohweg (2015)). On the other hand, HyBUTLA Niggemann *et al.* (2012) infers hybrid timed probabilistic automata while N-HyMn relaxes this timing constraint which allows N-HyMn to infer hybrid automata models for a larger class of hybrid systems. Summerville et al. and Soto et al. propose distinct methodologies that synthesize linear hybrid automata from observed run-time behavior of control systems (Summerville *et al.* (2017), Soto *et al.* (2019)). However, N-HyMn differs in the fact that their work is limited to systems where the derivatives of the continuous state variables are constant and guards are simple rectangular conditions over the system variables, which is not always observed in practice. Soto et al. proposed membership-based synthesis algorithm takes as input piece-wise linear (PWL) function that approximates time series data, whereas N-HyMn uses directly time series traces as input. Thus, membership-based synthesis method may not be applicable for non-linear systems, since PWL approximation introduces a tradeoff between accuracy and tractability. CHARDA requires an exhaustive construction of all possible models with a condition that a likelihood function is available for a given template model. In addition, CHARDA's segmentation and mode clustering approach is based on a principled penalty function for model complexity. Thus, two distinct mode can be merged into a single mode if the latter is less complex. This cannot be applicable in situations where learning the exact system model is crucial.

In this work, our goal is to learn the exact initial specifications model for model's conformance verification purposes Summerville *et al.* (2017).

## 9.4 Conformance Testing

N-HyMn shares similar motivation of the verification of the conformance between a running CPS and the formal specifications of its required behavior, which is referred to as *conformance testing* (Woehrle *et al.* (2012), Abbas (2015)). Woehrle et al. presented a conformance testing method that relies on mapping the specifications of the system and its implementation generated traces to timed automata and verifying whether each generated implementation trace is included in the traces of the specifications timed automaton. However, their approach is solely limited to the class of timed automata. As opposed to this conformance notion, other works define conformance testing as a closeness measure between an implementation and the specifications model, whose computation solely relies on system traces (Abbas (2015), Araujo *et al.* (2018)). However, even for simple linear systems, providing guarantees about the conformance degree remains a challenge.

# Chapter 10

## VERIFICATION AI-ENABLED CPS WITH LEARNING AGENT

A-enabled CPS with learning agents adjust their behavior in response to the changing physical system in order to achieve improved control. This significantly increases the complexity of model checking verification and reachability analysis techniques. For formally analyzing learning agents, we explore co-simulation of self-adaptive predictive (SAP) controllers and propose a novel co-simulation platform that can be used to analyze the effectiveness of verification and reachability analysis techniques developed for SAP controllers. SAP control is a promising approach to regulate Cyber-Physical Systems (CPS) with changing conditions by adjusting the control parameters. In the medical domain, self-adaptive control theory has gained increasing interest where emerging innovative medical devices adopt it to deliver more accurate, personalized treatment to patients (Turksoy and Cinar (2014); Hovorka *et al.* (2004)). For example, recent artificial pancreas (AP) control systems adjust insulin administration based on prediction over patients' blood glucose levels, where self-adaptation mechanisms optimize control parameters based on feedback from patients to account for the ever-changing characteristics of their glycemic regulatory system (Eren-Oruklu *et al.* (2008)). Simulation-based modeling tools, such as Matlab/Simulink are often used to model and evaluate the design of medical devices with self-adaptive predictive control.

In SAP, the controller responds not only to the dynamics of the physical system but also to the subtle changes in the dynamics over time. This introduces time variance in the models used for analysis and design of SAP controllers. Typically models deal with time variance of the parameters describing the physical system and a common method to model is through a system of differential equations involving the parameters.

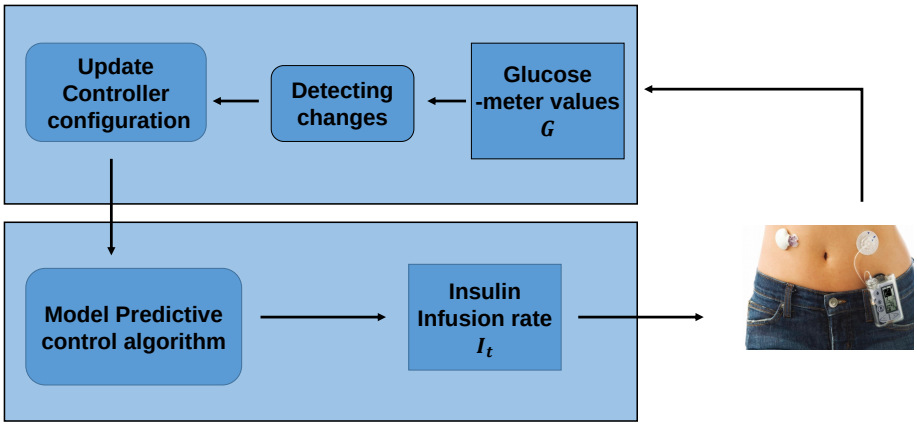


**Figure 10.1:** Artificial Pancreas: Self Adaptive Predictive Control System.

Formal safety verification of SAP controllers lies in verifying whether a certain unsafe set can be reached from a set of initial states. This verification is typically performed through a hybrid analysis of the co-variation of the inputs and outputs of the controller following a discrete control strategy and the time variation of the physical system parameters. As such if the physical model is time invariant, the verification problem is often intractable (Moon *et al.* (1998) ,Ravi and Somenzi (1995)). Techniques such as reachability analysis for the time invariant case cannot provide exact solutions and instead approximations are used (Chutinan and Krogh (2003)). *The time variance of the physical system models in SAP is an added complexity which further exacerbates the problem.* There has been very limited work on verification of SAP controllers assuming time variance of the physical models. Even the simpler problem of co-simulation of SAP controllers and physical system has not been studied in extensive detail.

**Example of Self-Adaptive Predictive Control Systems:** Artificial Pancreas (AP) systems are safety critical cyber-physical systems and are used for automated control of blood glucose level for Type1 diabetic patients. The aim is to maintain the prescribed level of blood glucose, and avoid hypoglycemic and hyperglycemic events. These dangerous events hap-

pen as a result of an inaccurate infusion rate of insulin $I_t$, e.g. if the glucose concentration G goes above 180mg/dl, it can lead to hyperglycemia while low glucose level i.e. below 50mg/dl can cause hypoglycemia. Self-adaptive predictive AP, shown in Figure 10.1, consists of a sensor that measures patient's glucose concentration and predictive control algorithm which estimates the value of the patient's blood glucose concentration and computes the insulin infusion rate to maintain until the next time step. Different conditions including meal consumption and physical activity can cause tremendous change in the parameters of the predictive model describing blood glucose and insulin interaction. This model is nonlinear in nature and is used by the controller to predict the value of blood glucose 30 minutes ahead in time and outputs the right amount of insulin infusion rate $I_t$ for the infusion pump to maintain until the next time step. Therefore, adjusting controller parameters in response to disturbances or systemic changes is a promising approach to regulate AP and to achieve improved control Hovorka *et al.* (2004).

## 10.1 Related Work

Model checking is one of the techniques used to ensure the correctness of the system by exploring all the possible environment states and ensuring that the system behaves as required in every state. However, the system model employed is not an accurate representation for time-invariant systems (Jacklin *et al.* (2004)). On the other hand, reachability analysis over hybrid automata provides a higher level of safety verification and has been extensively studied in the literature for time-invariant systems (Frehse (2015)). However, exact computation of reachable sets is still considered a difficult task and becomes even more complicated for time-varying systems Althoff *et al.* (2011). Therefore, union of short-term simulations on a set of initial conditions has been proposed as an approach to compute overapproximation of reachable sets for time-varying systems (Althoff *et al.* (2011)).
Iftikhar and Weyns have proposed an approach to validate behavioral properties of de-

centralized self-adaptive systems (Iftikhar and Weyns (2012)). This approach focuses on checking that the implementation of the system behaves complying with the model. The self-adaptive system is modeled with timed-automata and required properties are specified using timed-computation tree logic. The model is then verified using Uppaal (Larsen *et al.* (1997)). Another formal verification approach of adaptive real-time systems to verify tasks schedulability has been proposed by Hatvani (2014). Hatvani uses adaptive tasks automata to model adaptive real-time systems and introduces schedulability predicates as part of the adaptive task automata to define the schedulibility of a task. Tasks can be described in the model as long as their behavior can be modeled using task automata. The main contribution of the authors lies in defining decidability to prevent missed task deadlines when adjustments to the altered environmental conditions are performed.

The following are the main assumptions of the previously discussed approaches: 1- adaptation scenarios have to be predefined, 2- an environment model should be available since it specifies the failure events that have to be tested, and 3- proper test selection must be defined since exhaustive testing of systems is not feasible. None of the discussed approaches can be utilized to model and analyze SAP control systems since adaptation scenarios can not be predefined for SAP systems where configuration functions are linear combination between the parameters of the predictive model and the changing conditions of the environment. In addition, an environmental model with changing characteristics is not available for SAP control systems. Similarly, *Tan* has presented a model-based framework for developping self-adaptive systems (Tan (2006)). Tan introduced a configuration language to specify reconfiguration requirements and events triggering the reconfiguration are specified in temporal logic while the system behavior is depicted in the hybrid automata model of the system. However, the reconfiguration mechanism is limited to a constant function which can not be applied to predictive self-adaptive control system, where the configuration function is a linear combination between the parameters of the predictive model and

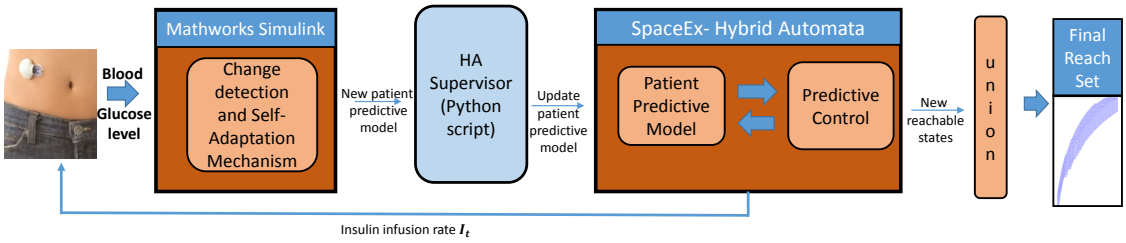


**Figure 10.2:** SAP Co-Simulation Framework. Mathworks and SpaceEc Executing Simultaneously

the changing conditions of the environment.

The proposed framework aims at designing and formally verifying self-adaptive predictive (SAP) control systems using co-simulation and reachability analysis. This co-simulation framework represents the first step towards developing a complete verification methodology for SAP controllers. It represents a time synchronized simulation of the SAP controller discrete decision making, physical model update method, and physical system evolution.

## 10.2 Proposed Approach: Co-Simulation Framework

The proposed approach depicted in Figure 10.2 is an alternative modeling technique for devices with self-adaptive predictive control. For ease of understanding, we present the SAP co-simulation framework for the artificial pancreas self-adaptive predictive system presented in Figure 10.1. The following represent the main steps of the co-simulation framework depicted in Figure 10.2:

- A patient predictive model is used to estimate the value of blood glucose 30 minutes ahead in time and computes the insulin infusion rate to maintain until the next time step. This model is represented by nonlinear equations 10.1, 10.2 and 10.3.

$\dot{X}$ represents the rate of the variation in the interstitial insulin concentration, $\dot{G}$ is the rate of change of blood glucose concentration for the infused insulin concentration $X$, and $\dot{I}$ is the variation in plasma insulin concentration (Andersen and Højbjerre (2002)).

$$\dot{X} = -k_2 X(t) + k_3(I(t) - I_b), \tag{10.1}$$

$$\dot{G} = -X(t)G(t) + k_1(G_b - G(t)), \tag{10.2}$$

$$\dot{I} = -k_4 I(t) + k_5(G(t) - k_6)t. \tag{10.3}$$

This model contains parameters $k_1, ..., k_6$ that are likely to change and need to be adapted for accuracy purposes. Some conditions including meal comsumption, exercise, and emotional changes can be the cause of these changes Turksoy and Cinar (2014). We first derive an approximate linear system that matches closely with the real AP system (Lamrani *et al.* (2018)).

- The change detection and self-adaptation mechanism detects changes in the behavior of the human body using recent blood glucose measurements. These changes physically correspond to significant change in glucose levels Eren-Oruklu *et al.* (2008). The change detection method compares the expected value of the model parameters and the vector of unbiased parameter estimates computed. It then adapts the predictive model accordingly by re-estimating the changing parameters of the model using the more recent data only Lamrani *et al.* (2018).

- **The HA supervisor** is in the form of a python script and performs the following steps:

  1. Generates initial model file in SpaceEx's XML format with initial patient predictive model settings $(k_1, k_2, ..., k_n)$ .

2. Calls SpaceEx executable file to run the command line program that takes a model file in XML format and a configuration file that specifies the initial states, sampling time, and other options. The sampling time can be adaptively computed by the reachability analysis support functions or manually selected taking into consideration that a discrete transition should not occur between two consecutive sampling times. SpaceEx analyzes the system and produces an output file $O_1.txt$ containing the reachable states computed.

3. Once a change is detected, it generates a new patient predictive model XML file with new parameter settings ($k'_1, k'_2, ..., k'_n$).

4. Calls SpaceEx executable file to run the command line program with the new generated model file. SpaceEx analyzes the system and produces an output file $O_2.txt$ containing the reachable states.

5. This process continues until termination criterion is satisfied.

- The final reach set of the self-adaptive control system is a union of all reachable states obtained with all controller configurations generated at runtime. Figure 10.3 shows an example of reach set computation for the artificial pancreas self-adaptive predictive control system. At every iteration, a new controller configuration is generated and the reach set is computed accordingly. The final reach set is obtained by combining all the regions of the state space that the system has visited, as shown in Figure 10.3.

The proposed approach strives to:

- Support modeling of predictive control systems using hybrid automata, and runtime self-adaption of hybrid automata based on new configurations from other modeling tools such as Simulink.

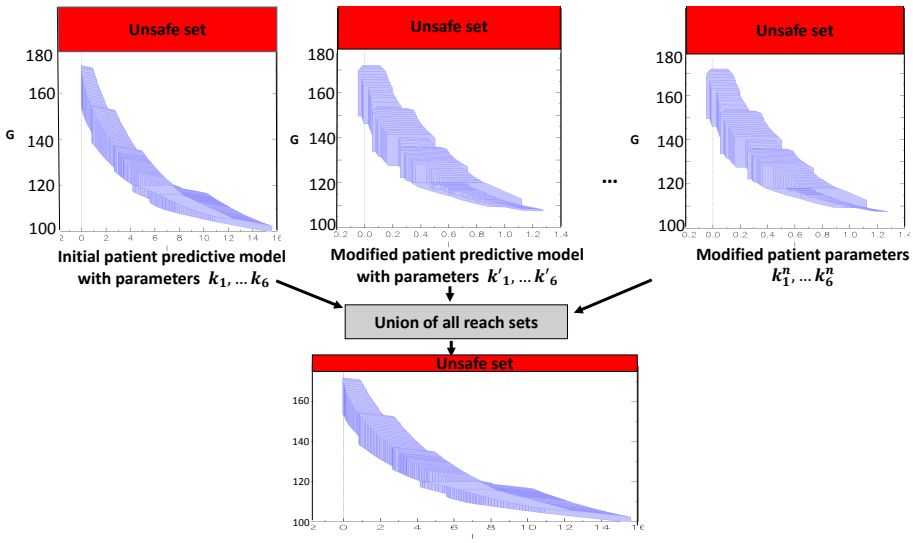


**Figure 10.3:** Reach Set of the Artificial Pancreas Self-Adaptive Predictive Control System.

- Provide an alternative modeling technique for devices with self-adaptive predictive control.
- Verify the safety of self-adaptive predictive control devices by checking whether the sets of reachable states of the system intersects with the unsafe set.

# Chapter 11

# CONCLUSIONS AND FUTURE WORK

The operational safety verification approach we proposed is based on a data science driven algorithm N-HyMn that infers non-linear hybrid automata representation from I/O operational traces of Industry 4.0 agent-based cyber-physical systems. The operational model can be learned in two different scenarios: a) model-aware, where the operation of the CPS can be compared with the specifications given by the manufacturer to ensure that the operation of the system conforms with the safety assured design, facilitating the detection of intentional or unintentional deviations from the certified specifications. and b) model-agnostic, where in absence of a specification model, the learned hybrid automaton can be used to evaluate potential safety threats through reachability analysis. Future research can involve developing an approach that automates and optimizes I/O data collection since it is crucial that the data used for model learning is collected from different regions of interest of the operation of the AI-enabled CPS. Another important direction is the analysis of the effect of the learning error on the learned reach set in order to correctly analyze the area of intersection between the unsafe set and learned reach set (for the unsafe safety guarantee case). Finally, since many security losses overlap with safety accidents, the proposed safety verification approach can be leveraged to prevent security losses.

# REFERENCES


Abbas, H., G. Fainekos, S. Sankaranarayanan, F. Ivančić and A. Gupta, "Probabilistic temporal logic falsification of cyber-physical systems", ACM Transactions on Embedded Computing Systems (TECS) **12**, 2s, 1–30 (2013).

Abbas, H. Y., *Test-Based Falsification and Conformance Testing for Cyber-Physical Systems.* (Arizona State University, 2015).

Agrawal, A., P. D. Deshpande, A. Cecen, G. P. Basavarsu, A. N. Choudhary and S. R. Kalidindi, "Exploration of data science techniques to predict fatigue strength of steel from composition and processing parameters", Integrating Materials and Manufacturing Innovation **3**, 1, 8 (2014).

Althoff, M., C. Le Guernic and B. H. Krogh, "Reachable set computation for uncertain time-varying linear systems", in "Proceedings of the 14th international conference on Hybrid systems: computation and control", pp. 93–102 (2011).

Alur, R., C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis and S. Yovine, "The algorithmic analysis of hybrid systems", Theoretical computer science **138**, 1, 3–34 (1995).

Alur, R., C. Courcoubetis, T. A. Henzinger and P.-H. Ho, "Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems", in "Hybrid systems", pp. 209–229 (Springer, 1992).

Alves, E. E., D. Bhatt, B. Hall, K. Driscoll, A. Murugesan and J. Rushby, "Considerations in assuring safety of increasingly autonomous systems", (2018).

Aminikhanghahi, S. and D. J. Cook, "A survey of methods for time series change point detection", Knowledge and information systems **51**, 2, 339–367 (2017).

Ammann, P. and J. Offutt, *Introduction to software testing* (Cambridge University Press, 2016).

Amodei, D., C. Olah, J. Steinhardt, P. Christiano, J. Schulman and D. Mané, "Concrete problems in ai safety", arXiv preprint arXiv:1606.06565 (2016).

Andersen, K. E. and M. Højbjerre, "A bayesian approach to bergman's minimal model", Insulin **50**, 100, 200 (2002).

Araujo, H., G. Carvalho, M. Mohaqeqi, M. R. Mousavi and A. Sampaio, "Sound conformance testing for cyber-physical systems: Theory and implementation", Science of Computer Programming **162**, 35–54 (2018).

Asarin, E., O. Bournez, T. Dang and O. Maler, "Approximate reachability analysis of piecewise-linear dynamical systems", in "International Workshop on Hybrid Systems: Computation and Control", pp. 20–31 (Springer, 2000).

Balakrishnan, H., I. Hwang, J. S. Jang and C. J. Tomlin, "Inference methods for autonomous stochastic linear hybrid systems", in "International Workshop on Hybrid Systems: Computation and Control", pp. 64–79 (Springer, 2004).

Banerjee, A., Y. Zhang, P. Jones and S. Gupta, "Using formal methods to improve home-use medical device safety", Biomedical instrumentation & technology **47**, s1, 43–48 (2013).

Bergman, R. N., Y. Z. Ider, C. R. Bowden and C. Cobelli, "Quantitative estimation of insulin sensitivity.", American Journal of Physiology-Endocrinology And Metabolism **236**, 6, E667 (1979).

Blackmore, L., S. Gil, S. Chung and B. Williams, "Model learning for switching linear systems with autonomous mode transitions", in "2007 46th IEEE Conference on Decision and Control", pp. 4648–4655 (IEEE, 2007).

Bubeck, S. *et al.*, "Convex optimization: Algorithms and complexity", Foundations and Trends® in Machine Learning **8**, 3-4, 231–357 (2015).

Chen, T. Y., S. C. Cheung and S. M. Yiu, "Metamorphic testing: a new approach for generating next test cases", Tech. rep., Technical Report HKUST-CS98-01, Department of Computer Science, Hong Kong … (1998).

Chutinan, A. and B. H. Krogh, "Computational techniques for hybrid system verification", IEEE transactions on automatic control **48**, 1, 64–75 (2003).

Contag, M., G. Li, A. Pawlowski, F. Domke, K. Levchenko, T. Holz and S. Savage, "How they did it: An analysis of emission defeat devices in modern automobiles", in "2017 IEEE Symposium on Security and Privacy (SP)", pp. 231–250 (IEEE, 2017).

DSA, "Service inquiry report into the loss of watchkeeper (wk043) unmanned air vehicle over cardigan bay in west wales", Defence Safety Authority, Corporate report Report (2019).

Duggirala, P. S., S. Mitra, M. Viswanathan and M. Potok, "C2e2: A verification tool for stateflow models", in "International Conference on Tools and Algorithms for the Construction and Analysis of Systems", pp. 68–82 (Springer, 2015a).

Duggirala, P. S., S. Mitra, M. Viswanathan and M. Potok, "C2e2: A verification tool for stateflow models", in "International Conference on Tools and Algorithms for the Construction and Analysis of Systems", pp. 68–82 (Springer, 2015b).

Eren-Oruklu, M., A. Cinar, C. Colmekci and M. C. Camurdan, "Self-tuning controller for regulation of glucose levels in patients with type 1 diabetes", in "2008 American Control Conference", pp. 819–824 (IEEE, 2008).

Fan, C., B. Qi, S. Mitra and M. Viswanathan, "Dryvr: Data-driven verification and compositional reasoning for automotive systems", in "International Conference on Computer Aided Verification", pp. 441–461 (Springer, 2017).

FDA, U., "Summary of safety and effectiveness data (ssed) of the medtronic minimed 670g system. 2016", (2016).

Frehse, G., “Reachability of hybrid systems in space-time”, in “2015 International Conference on Embedded Software (EMSOFT)”, pp. 41–50 (IEEE, 2015).

Frehse, G., C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang and O. Maler, “Spaceex: Scalable verification of hybrid systems”, in “International Conference on Computer Aided Verification”, pp. 379–395 (Springer, 2011).

Girard, A., A. A. Julius and G. J. Pappas, “Approximate simulation relations for hybrid systems”, Discrete event dynamic systems **18**, 2, 163–179 (2008).

Haidar, A., “The artificial pancreas: How closed-loop control is revolutionizing diabetes”, IEEE Control Systems Magazine **36**, 5, 28–47 (2016).

Hatton, L. and A. Rutkowski, “” lessons must be learned”-but are they?”, IEEE Software **36**, 4, 91–95 (2019).

Hatvani, L., *Formal verification of adaptive real-time systems by extending task automata*, Ph.D. thesis, Mälardalen University (2014).

Haugh, B. A., D. A. Sparrow and D. M. Tate, “The status of test, evaluation, verification, and validation (tev&v) of autonomous systems”, (2018).

Henzinger, T. A., “The theory of hybrid automata”, in “Verification of digital and hybrid systems”, pp. 265–292 (Springer, 2000).

Henzinger, T. A., R. Majumdar and V. S. Prabhu, “Quantifying similarities between timed systems”, in “International Conference on Formal Modeling and Analysis of Timed Systems”, pp. 226–241 (Springer, 2005).

Hovorka, R., V. Canonico, L. J. Chassin, U. Haueter, M. Massi-Benedetti, M. O. Federici, T. R. Pieber, H. C. Schaller, L. Schaupp, T. Vering *et al.*, “Nonlinear model predictive control of glucose concentration in subjects with type 1 diabetes”, Physiological measurement **25**, 4, 905 (2004).

Hoxha, B., A. Dokhanchi and G. Fainekos, “Mining parametric temporal logic properties in model-based design for cyber-physical systems”, International Journal on Software Tools for Technology Transfer **20**, 1, 79–93 (2018).

Iftikhar, M. U. and D. Weyns, “A case study on formal verification of self-adaptive behaviors in a decentralized system”, arXiv preprint arXiv:1208.4635 (2012).

Instruments, M., “An introduction to functional safety and iec 61508”, Online]. Disponible en http://www. mtl-inst. com/images/uploads/datasheets/App _Notes/AN9025. pdf (2011).

ISO, I., “26262: Road vehicles-functional safety”, International Standard ISO/FDIS **26262** (2011).

Ivanov, R., J. Weimer, R. Alur, G. J. Pappas and I. Lee, “Verisig: verifying safety properties of hybrid systems with neural network controllers”, in “Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control”, pp. 169–178 (2019).

Jacklin, S., J. Schumann, P. Gupta, M. Lowry, J. Bosworth, E. Zavala, K. Hayhurst, C. Belcastro and C. Belcastro, "Verification, validation, and certification challenges for adaptive flight-critical control system software", in "AIAA Guidance, Navigation, and Control Conference and Exhibit", p. 5258 (2004).

Jin, X., A. Donzé, J. V. Deshmukh and S. A. Seshia, "Mining requirements from closed-loop control models", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **34**, 11, 1704–1717 (2015).

Johnston, P. and R. Harris, "The boeing 737 max saga: Lessons for software organizations", Software Quality Professional **21**, 3, 4–12 (2019).

Kim, K.-D. and P. R. Kumar, "Cyber–physical systems: A perspective at the centennial", Proceedings of the IEEE **100**, Special Centennial Issue, 1287–1308 (2012).

Kong, S., S. Gao, W. Chen and E. Clarke, "dreach: $\delta$-reachability analysis for hybrid systems", in "International Conference on TOOLS and Algorithms for the Construction and Analysis of Systems", pp. 200–205 (Springer, 2015).

Ladkin, P. B., "An overview of iec 61508 on e/e/pe functional safety", Bielefeld, Germany (2008).

Lamrani, I., A. Banerjee and S. K. Gupta, "Hymn: Mining linear hybrid automata from input output traces of cyber-physical systems", in "2018 IEEE Industrial Cyber-Physical Systems (ICPS)", pp. 264–269 (IEEE, 2018).

Larsen, K. G., P. Pettersson and W. Yi, "Uppaal in a nutshell", International journal on software tools for technology transfer **1**, 1-2, 134–152 (1997).

Lee, J., B. Bagheri and H.-A. Kao, "A cyber-physical systems architecture for industry 4.0-based manufacturing systems", Manufacturing letters **3**, 18–23 (2015).

Leveson, N., *Engineering a safer world: Systems thinking applied to safety* (MIT press, 2011).

Leveson, N. G., "Software safety: Why, what, and how", ACM Comput. Surv. **18**, 2, 125–163, URL `http://doi.acm.org/10.1145/7474.7528` (1986).

Levin, S., "Uber crash shows' catastrophic failure'of self-driving technology, experts say", The Guardian **22** (2018).

Liu, S., M. Yamada, N. Collier and M. Sugiyama, "Change-point detection in time-series data by relative density-ratio estimation", Neural Networks **43**, 72–83 (2013).

Ly, D. L. and H. Lipson, "Learning symbolic representations of hybrid dynamical systems", Journal of Machine Learning Research **13**, Dec, 3585–3618 (2012).

Lyde, S. and M. Might, "Extracting hybrid automata from control code", in "NASA Formal Methods Symposium", pp. 447–452 (Springer, 2013).

Man, C. D., F. Micheletto, D. Lv, M. Breton, B. Kovatchev and C. Cobelli, "The uva/padova type 1 diabetes simulator: new features", Journal of diabetes science and technology **8**, 1, 26–34 (2014).

Medhat, R., S. Ramesh, B. Bonakdarpour and S. Fischmeister, "A framework for mining hybrid automata from input/output traces", in "Proceedings of the 12th International Conference on Embedded Software", pp. 177–186 (IEEE Press, 2015).

Milanese, M. and G. Belforte, "Estimation theory and uncertainty intervals evaluation in presence of unknown but bounded errors: Linear families of models and estimators", IEEE Transactions on automatic control **27**, 2, 408–414 (1982).

Minopoli, S. and G. Frehse, "From simulation models to hybrid automata using urgency and relaxation", in "Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control", pp. 287–296 (ACM, 2016a).

Minopoli, S. and G. Frehse, "Sl2sx translator: from simulink to spaceex models", in "Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control", pp. 93–98 (ACM, 2016b).

Moon, I.-H., J.-Y. Jang, G. D. Hachtel, F. Somenzi, J. Yuan and C. Pixley, "Approximate reachability don't cares for ctl model checking", in "Proceedings of the 1998 IEEE/ACM international conference on Computer-aided design", pp. 351–358 (1998).

Myers, G. J., C. Sandler and T. Badgett, *The art of software testing* (John Wiley & Sons, 2011).

Narayan, A., G. Cutulenco, Y. Joshi and S. Fischmeister, "Mining timed regular specifications from system traces", ACM Trans. Embed. Comput. Syst. **17**, 2, 46:1–46:21, URL `http://doi.acm.org/10.1145/3147660` (2018).

Nenzi, L., S. Silvetti, E. Bartocci and L. Bortolussi, "A robust genetic algorithm for learning temporal specifications from data", in "International Conference on Quantitative Evaluation of Systems", pp. 323–338 (Springer, 2018).

Niggemann, O. and V. Lohweg, "On the diagnosis of cyber-physical production systems: state-of-the-art and research agenda", in "Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence", pp. 4119–4126 (AAAI Press, 2015).

Niggemann, O., B. Stein, A. Vodencarevic, A. Maier and H. K. Büning, "Learning behavior models for hybrid timed systems.", in "AAAI", vol. 2, pp. 1083–1090 (2012).

Niggemann, O., S. Windmann, S. Volgmann, A. Bunte and B. Stein, "Using learned models for the root cause analysis of cyber-physical production systems", in "Int. Workshop Principles of Diagnosis (DX)", (2014).

Prabhakar, P., R. Lal and J. Kapinski, "Automatic trace generation for signal temporal logic", in "2018 IEEE Real-Time Systems Symposium (RTSS)", pp. 208–217 (2018).

Rajkumar, R., I. Lee, L. Sha and J. Stankovic, "Cyber-physical systems: the next computing revolution", in "Design Automation Conference", pp. 731–736 (IEEE, 2010).

Ravi, K. and F. Somenzi, "High-density reachability analysis", in "Proceedings of IEEE International Conference on Computer Aided Design (ICCAD)", pp. 154–158 (IEEE, 1995).

Ruiz, J. L., J. L. Sherr, E. Cengiz, L. Carria, A. Roy, G. Voskanyan, W. V. Tamborlane and S. A. Weinzimer, "Effect of insulin feedback on closed-loop glucose control: a crossover study", Journal of diabetes science and technology **6**, 5, 1123–1130 (2012).

Russell, S. J. and P. Norvig, *Artificial intelligence: a modern approach* (Malaysia; Pearson Education Limited,, 2016).

Santana, P. H., S. Lane, E. Timmons, B. C. Williams and C. Forster, "Learning hybrid models with guarded transitions.", in "AAAI", pp. 1847–1853 (2015).

Scherer, M. U., "Regulating artificial intelligence systems: Risks, challenges, competencies, and strategies", Harv. JL & Tech. **29**, 353 (2015).

Seshia, S. A., D. Sadigh and S. S. Sastry, "Towards verified artificial intelligence", arXiv preprint arXiv:1606.08514 (2016).

Sheridan, T. B. and R. Parasuraman, "Human vs. automation in responding to failures: An expected-value analysis", in "Proceedings of the Human Factors and Ergonomics Society Annual Meeting", vol. 44, pp. 1–4 (Sage Publications Sage CA: Los Angeles, CA, 2000).

Soto, M. G., T. A. Henzinger, C. Schilling and L. Zeleznik, "Membership-based synthesis of linear hybrid automata", in "International Conference on Computer Aided Verification", pp. 297–314 (Springer, 2019).

Steil, G. M., C. C. Palerm, N. Kurtz, G. Voskanyan, A. Roy, S. Paz and F. R. Kandeel, "The effect of insulin feedback on closed loop glucose control", The Journal of Clinical Endocrinology & Metabolism **96**, 5, 1402–1408 (2011).

Summerville, A., J. Osborn and M. Mateas, "Charda: Causal hybrid automata recovery via dynamic analysis", arXiv preprint arXiv:1707.03336 (2017).

Tan, L., "Model-based self-adaptive embedded programs with temporal logic specifications", in "2006 Sixth International Conference on Quality Software (QSIC'06)", pp. 151–158 (IEEE, 2006).

Transportasi, K. N. K., "Aircraft accident investigation report", Ministry of Transportation, Indonesia, Report (2018).

Turksoy, K. and A. Cinar, "Adaptive control of artificial pancreas systems-a review", Journal of healthcare engineering **5** (2014).

Woehrle, M., K. Lampka and L. Thiele, "Conformance testing for cyber-physical systems", ACM Transactions on Embedded Computing Systems (TECS) **11**, 4, 84 (2012).

Zhang, J., J. Chen, D. Hao, Y. Xiong, B. Xie, L. Zhang and H. Mei, "Search-based inference of polynomial metamorphic relations", in "Proceedings of the 29th ACM/IEEE international conference on Automated software engineering", pp. 701–712 (ACM, 2014).

# APPENDIX A

# MEDTRONIC MINIMED 670G DESCRIPTION

## A.1 Medtronic Minimed 670G Control System Specifications

The Minimed 670 G CLAP control system is responsible for delivering throughout the day basal insulin and large amounts of insulin (bolus) to cover meals or correct high glucose levels. Fig. B.1 shows the hybrid automaton model of Minimed 670G consisting of four control modes: Basal Auto, Food Correction Bolus, BG Correction Bolus, BG and Food Correction Bolus, and Suspend Before Low. The glucose subsystem of the human body represents the predictive environmental model and is governed by the non-linear ordinary differential equations of the Bergman minimal model Bergman *et al.* (1979), which describes the evolution of the interstitial insulin concentration $I_{EFF}(t)$, blood glucose concentration $B_G(t)$, and plasma insulin concentration $I_p(t)$. The external input continuous variables of the CLAP are meal carbs amount $CHO(t)$ and finger stick BG reading $B_{GF}(t)$, which are provided by the user. The internal input continuous variables of the CLAP are $\{I_{EFF}(t)$, $B_G(t)$, $PID(t)$, $IFB(t)\}$. $I_D(t)$ is the output contrinuous variable and also represents the reset condition of the CLAP. In basal mode, the closed-loop insulin delivery $I_D(t) = I_{Ba}(t)$, where $I_{Ba}(t) = f_1(P(t), I(t), D(t), I_p(t-1))$, as shown in Fig. A.1 (Ruiz *et al.* (2012)). $P(t)$, $I(t)$, and $D(t)$ denotes the proportional, integral, and derivative terms of the PID controller. $I_p(t-1)$ represents a real-time estimate of insulin concentration one time step back in time, where $t$ denotes the most recent value. Initially, the control system is in basal mode and the transition from Auto Basal to other modes is enabled when the according guard condition from the total eight guard conditions of CLAP is satisfied, as shown in Fig. B.1. In the bolus mode, $I_D(t) = I_{Ba}(t) + I_{Bo(t)}$, where $I_{Bo(t)} = f_2\ (CHO(t), BW)$, $BW$ refers to the body weight of the subject, and $f_1$ and $f_2$ are linear functions.
Note that the availability of the specifications model of the internal variables $\{I_{EFF}(t)$, $B_G(t)$, $PID(t)$, $IFB(t)\}$, described in Fig. A.1, is necessary in the specifications **Model-Aware** learning scenario. In the specifications **Model-Agnostic** learning scenario, we assume that the the specifications model of the internal variables $\{I_{EFF}(t)$, $B_G(t)$, $PID(t)$, $IFB(t)\}$ is not available. $K_P$, $K_D$, and $K_I$: PID controller gains. $g_1$: IFB parameter for subcutaneous insulin, $g_2$: IFB parameter for plasma insulin, $g_3$: IFB parameter for effective insulin. $b_i$ is an insulin delivery coefficient, $a_{ij}$ is a subcutaneous insulin pharmacokinetic constant.

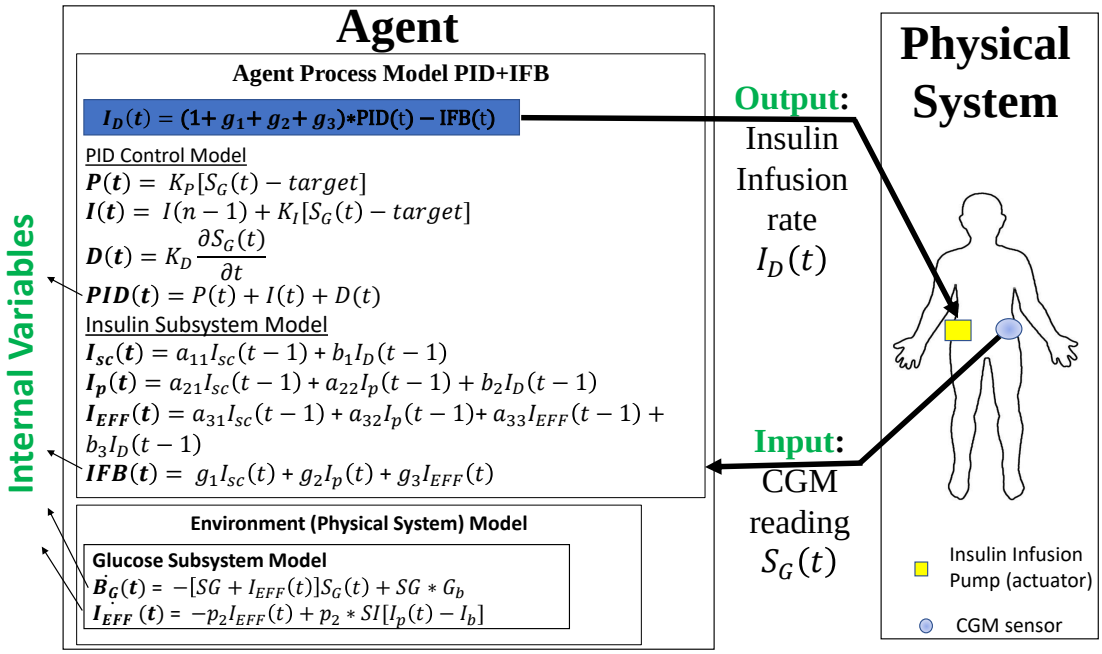


**Figure A.1:** Control Structure of the Medtronic Minimed 670G Insulin Pump System (Basal Auto Mode Specifications).

# APPENDIX B

# MEDTRONIC MINIMED 670G HYBRID AUTOMATON

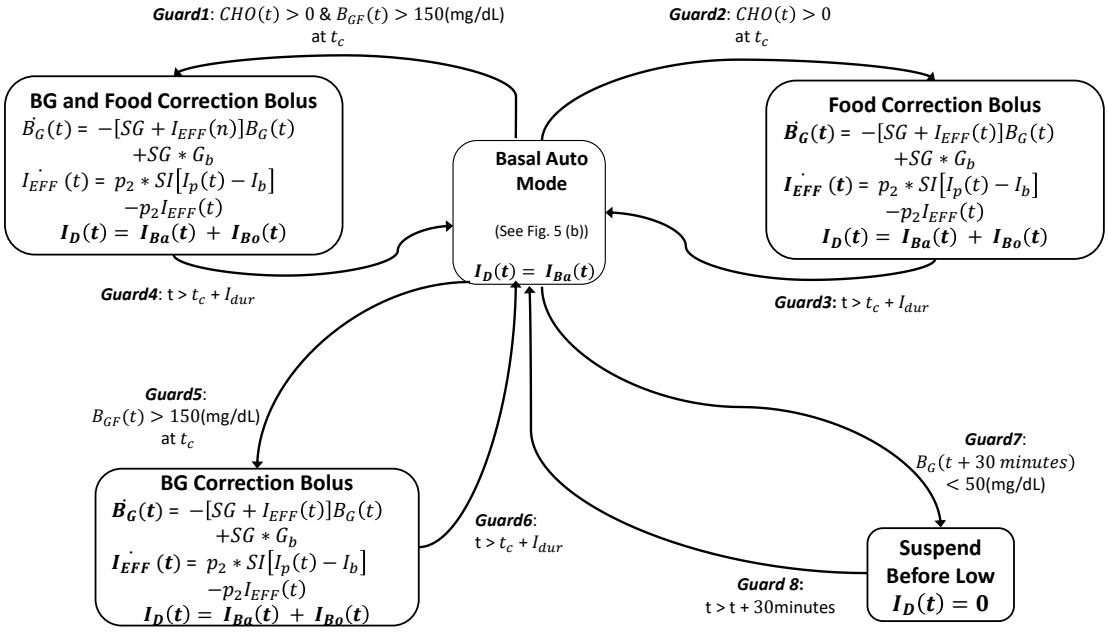


**Figure B.1:** Non-Linear Hybrid Automaton of the Artificial Pancreas Control System (Medtronic Minimed 670G).

Variable $I_{EFF}(t)$ refers to interstitial insulin, $B_G(t)$: plasma glucose ($G_b$ its basal value), $Ip(t)$: plasma insulin ($I_b$ its basal value), $I_{sc}(t)$: subcutaneous insulin, $I_{Ba}(t)$: closed-loop basal insulin delivery profile, $I_{Bo}(t)$: bolus insulin delivery profile, $I_D(n)$: total exogeneous insulin infusion, and $I_{dur}$: duration of bolus injection. SG, SI, and p2 are model parameters.