

Model Based Safety Analysis and Verification of Cyber-Physical Systems

by

Ayan Banerjee

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

Approved November 2012 by the
Graduate Supervisory Committee:

Sandeep K.S. Gupta, Chair
Radha Poovendran
Georgios Fainekos
Ross Maciejewski

ARIZONA STATE UNIVERSITY

December 2012

ABSTRACT

Critical infrastructures in healthcare, power systems, and web services, incorporate cyber-physical systems (CPSes), where the software controlled computing systems interact with the physical environment through actuation and monitoring. Ensuring software safety in CPSes, to avoid hazards to property and human life as a result of un-controlled interactions, is essential and challenging. The principal hurdle in this regard is the characterization of the context driven interactions between software and the physical environment (cyber-physical interactions), which introduce multi-dimensional dynamics in space and time, complex non-linearities, and non-trivial aggregation of interaction in case of networked operations.

Traditionally, CPS software is tested for safety either through experimental trials, which can be expensive, incomprehensive, and hazardous, or through static analysis of code, which ignore the cyber-physical interactions. This thesis considers model based engineering, a paradigm widely used in different disciplines of engineering, for safety verification of CPS software and contributes to three fundamental phases: a) modeling, building abstractions or models that characterize cyber-physical interactions in a mathematical framework, b) analysis, reasoning about safety based on properties of the model, and c) synthesis, implementing models on standard testbeds for performing preliminary experimental trials.

In this regard, CPS modeling techniques are proposed that can accurately capture the context driven spatio-temporal aggregate cyber-physical interactions. Different levels of abstractions are considered, which result in high level architectural models, or more detailed formal behavioral models of CPSes. The outcomes include, a well defined architectural specification framework called *CPS-DAS* and a novel spatio-temporal formal model called Spatio-Temporal Hybrid Automata (STHA) for CPSes.

Model analysis techniques are proposed for the CPS models, which can simulate the effects of dynamic context changes on non-linear spatio-temporal cyber-physical interactions, and characterize aggregate effects. The outcomes include tractable algorithms for simulation analysis and for theoretically proving safety properties of CPS software.

Lastly a software synthesis technique is proposed that can automatically convert high level architectural models of CPSes in the healthcare domain into implementations in high level programming languages. The outcome is a tool called *Health-Dev* that can synthesize software implementations of CPS models in healthcare for experimental verification of safety properties.

DEDICATION

To my Dadu (Grandfather) and Thakuma (Grandmother) for providing me with the inspiration to be a scientist. To my parents for all their sacrifices. To my brother for being ever so charming. To my wife for her endless support.

ACKNOWLEDGEMENTS

I would like to thank my advisor Dr. Sandeep K.S. Gupta for his guidance. Throughout my PhD, he has continuously instilled in me an urge to perform ground breaking research. His grooming has helped me pass the stern tests of failure and emerge successful.

I am immensely grateful to my seniors Dr. Krishna Kumar Venkatasubramanian (Krishnaji) and Dr. Tridib Mukherjee (Tridibda) for their precious advice throughout my doctoral studies. With Krishnaji, I share my first publication, which was a landmark experience in my graduate life. Working late nights with Tridibda was so enjoyable even under immense pressure of deadlines. Special thanks to Dr. Georgios Vasamopoulos for his feedback and guidance on research and introducing me to \LaTeX , which has helped me throughout my PhD career. I would also like to thank all my lab-mates, Sailesh, Sunit, Zahra, Priyanka, Joseph, Robin, and many others with whom I have shared such memorable times at ASU. I am proud to be a member of the IMPACT lab at ASU, which has produced several outstanding PhDs over the years. I also thank Science Foundation of Arizona, National Science Foundation grants CNS # 0831544 and IIS # 1116385, and Intel for supporting me financially during my doctoral studies. Special thanks to Food and Drugs Administration for providing experimental data, case studies and an internship opportunity. I also thank my committee members Dr. Poovendran, Dr. Fainekos, and Dr. Maciejewski for providing useful comments on my research.

I dedicate all my success to my grandparents and parents who have instilled in me the constant quest for knowledge. They have shielded me from all the responsibilities allowing me to concentrate on my studies. In my wife, I found a strong support at difficult times and near important milestones in my graduate life. I thank her for being such a patient and understanding partner.

TABLE OF CONTENTS

	Page
TABLE OF CONTENTS	v
LIST OF TABLES	xi
LIST OF FIGURES	xii
LIST OF SYMBOLS	xix
CHAPTERS	
1 INTRODUCTION	1
1.1 Safety in CPSes	1
1.2 Approaches for CPS Software Safety Assurance	4
1.3 Model Based Software Analysis and Verification	6
1.4 Related Work	7
1.5 Challenges of MBSV for CPSes	9
1.6 Thesis contributions	12
Modeling Stage	13
Analysis Stage	14
Synthesis	15
2 CPS DOMAINS AND SAFETY ISSUES	17
2.1 Safety in CPS	19
2.2 Ayushman Pervasive Health Monitoring System (PHMS)	24
Home Context	24
Roaming Context	27
Hospital Context	29
Context changes	29
2.3 Artificial Pancreas	30
Discrete Control Algorithm	31

Chapter	Page
Brake Mode	32
Correction Bolus Mode	33
Meal Bolus Supervision Mode	33
Blood Glucose Predictor Model	33
2.4 Data Center Cooling Control	36
Equipment Safety	38
Job and Machine Environment	38
CRAC Behavior	40
CRAC Power Consumption	41
Inter-dependency of Cooling and Job Management	41
HTS Algorithm Design	44
Server Ranking	44
Job Placement	46
Dynamic Thermostat Setting	46
3 BACKGROUND ON MODEL BASED ENGINEERING	48
3.1 Modeling Phase	48
Architectural model	51
Behavioral model	52
Mathematical equations	52
Transfer function model	53
Computational model	54
3.2 Analysis Phase	59
Simulation Analysis	59
Reachability Analysis	60

Chapter	Page
3.3 Synthesis Phase	62
4 ARCHITECTURAL MODELING FOR FAST SIMULATIONS	63
4.1 Motivation and Related Works	63
4.2 CPS-DAS: Analysis and Design of CPSes	65
CPS-DAS Modeling Framework	65
Requirements Modeling	66
Abstract Modeling of CPSs	66
Analysis Parameter Modeling	72
CPS-DAS Analyzer	72
Implementation	74
Model Specification	75
Analysis framework	77
4.3 Case Study for Design and Analysis with CPS-DAS	78
Safety Analysis of a Single Wearable Medical Device	79
CPS-DAS Inputs	80
CPS-DAS Model	81
Safety Verification	82
Safety Analysis of Network of Devices	83
CPS-DAS Inputs	84
CPS-DAS Model	84
Safety Analysis	85
Thermal Safety of Servers in a Data center	86
CPS-DAS modeling of data centers	86
Safety analysis of data centers	88

Chapter	Page
5 SAFETY ANALYSIS UNDER DYNAMIC CONTEXTS	90
5.1 Dynamic Contexts	91
Representation of context	92
5.2 Effects of Context changes	93
5.3 Related Works	97
5.4 Specification Phase	101
5.5 Profiling Phase	106
Power Profiling	106
Thermal Profiling	108
5.6 Modeling Phase	109
Power model of PHMS	109
Thermal model of PHMS	110
Model of Infusion Control	111
5.7 PHMS Analysis	112
Effect of context change on medical control systems	114
Context driven safety violation:	115
6 FORMAL MODELING AND ANALYSIS OF CPSES	117
6.1 Motivation and Related Works	119
6.2 Preliminaries and Overview of Approach	123
Overview of Approach	125
Safety in CPSES	126
6.3 Linear 1-D Space Spatio-Temporal Hybrid Automata	128
6.4 L1STHA execution model	133
6.5 Defining the ϵ reach set of a DTL1STHA	137

Chapter	Page
Solving the PDE	138
Notations	140
6.6 Time and space bounded epsilon reach set of a DTL1STHA with a single discrete location	141
6.7 ϵ reach set for a set of initial configurations	145
6.8 Determining exit condition from an invariant set	147
6.9 Algorithm for bounded ϵ reach set of a single location DTL1STHA	148
6.10 Algorithm for computing ϵ reach set for multiple location DTL1STHA	153
DTL1STHA bounded time and space reachability analysis algorithm	155
6.11 Formal Analysis Case-Studies	157
BSN Safety Verification	159
STHA model for BSN Safety verification	160
Formal BSN Safety Verification	160
Drug infusion using infusion pumps	162
Analgesic Infusion Pump	162
Infusion Pump Formal Model	163
Formal Infusion Pump Safety Verification	164
7 TACKLING NON-LINEARITIES IN CPS SAFETY ANALYSIS	167
7.1 Non-linear dynamics in Artificial Pancreas	167
7.2 Modeling Artificial Pancreas with Hybrid Automata	168
Linear Hybrid Automata	169
Hybrid Modeling of Artificial Pancreas	171
Linearization of Bergman Minimal Model	171
Artificial Pancreas Hybrid Automata Model	172

Chapter	Page
7.3 Patient Safety Analysis	173
Reachability Analysis	173
Error Bounds on the Linear Approximation	174
8 SYNTHESIS OF CPS MODELS	178
8.1 Motivation and Related Works	178
8.2 Health-Dev: Model Based Synthesis of BSNs	179
Specification Module	179
Parser Module	182
Code Generation Module	184
9 CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS	186
9.1 Conclusions	186
9.2 Immediate extensions	186
Integrated Tool for CPS Safety Analysis, Verification, and Design	187
Synthesizing STHA Models in Hardware	188
9.3 Future research directions	190
Modeling Stage	190
Analysis Stage	193
Synthesis Stage	193
BIBLIOGRAPHY	195
BIOGRAPHICAL SKETCH	210

LIST OF TABLES

Table	Page
2.1 Definition of different safety categories in a CPS is listed. The safety categories are further classified as CPS specific, non-CPS or applicable to both	24
2.2 Power available from different scavenging sources in the environment and the human body. The scavenging hours indicated are gross estimates for the case study based on the study by Paradiso [1].	28
2.3 Scalar Symbols and Definitions.	39
4.1 Skin temperatures after eight hours of pulse oximeter operation at different device temperatures (Burn threshold 39 °C)	83
4.2 Timing and power consumption values used for the Ayushman workload and the experimental methodology	85
4.3 Tissue temperature rise for different leadership sequences (Burn threshold 39 °C)	86
5.1 Classification of existing work on model based safety analysis	98
5.2 Power profiling of sensors (TelosB, iMote, BSN node v3, shimmer)	107
5.3 Atom Power Usage for PKA computation in Ayushman	107
5.4 Thermal Profile Data for different frequency throttling modes of the Intel Atom N270 processor	108
6.1 Notations for reachable states and over approximations	141
6.2 Safety Verification Results (HA = Hybrid Automata)	160
8.1 Specifiable properties in <i>Health-Dev</i>	182

LIST OF FIGURES

Figure	Page
1.1 Cyber-physical systems, with tight coupling between computing and physical environment through interaction parameters. Solid lines indicate effect of computing operations on interaction parameters, dashed lines indicate effect of physical operations on interaction parameters, and thick solid lines indicate cyber-physical interactions.	2
1.2 Safety assurance approaches for CPS software. The principal contribution of this thesis is facilitating cyber-physical interaction modeling and verification in MBSV for CPSes.	3
1.3 Typical deployment of two representative CPSes showing the distributed nature of the computing nodes, intentional and un-intentional interactions with the physical environment, and the aggregate effect of these interactions because of concurrent operations in the distributed nodes. .	11
1.4 Stages of model based safety verification of CPS software and thesis contributions. The MBSV consists of three stages modeling, analysis and synthesis. The thesis has contributions in each of the three stages. .	13
2.1 Ayushman workload with varying computation and communication requirements. Computation includes sensing and signal processing for physiological value based security. The sensor processor is duty cycled and the frequency throttled while the radio is kept off during the computation phase.	25
2.2 Generative Model based Resource Efficient Monitoring. The sensor and the base station uses a model to reduce communication. The sensor only transmits model parameters to the base station which in turn uses the model to reconstruct the data.	26

Figure	Page
2.3 Finite State Automata representation of contexts and context changes, <i>ContextFSM</i> , for the Ayushman PHMS. The transitions are governed by random models of contexts such as mobility or occurrence of physiological conditions such as epilepsy.	30
2.4 Generic Architecture of Artificial Pancreas, which consist of an infusion pump, a glucose monitor and a remote controller. The controller receives feedback from the glucose meter and computes the future infusion rate using a model predictive control algorithm. It transfers the infusion rate to the infusion pump. Communication between the pump, controller, and the glucose meter takes place through the wireless channel.	31
2.5 Anesthesia pump simulation with no bolus and at different levels of bolus showing overshoot, settling time, stable, and unstable behavior. Bolus administration marked by red lines.	36
2.6 Heat transfer mechanisms in data center. The cool air from the CRAC is non-uniformly distributed among the servers. The heated air is recirculated throughout the data center and finally reaches the CRAC where it is again cooled.	37
2.7 Variations in the CRAC input and output temperature based on actual sensor measurements in the ASU HPC data center. The difference in these temperatures indicates the two operational modes of the CRAC.	42

Figure	Page
2.8 Architecture and work-flow of HTS. Each server is assigned a high CRAC threshold temperature that it requires to avoid redlining. The workload manager prefers servers with high CRAC thresholds and sets the CRAC threshold temperature to the lowest high threshold temperature among the chosen servers.	45
3.1 System is a combination of interacting components with inputs and outputs. Any parameter affecting the operation of the system and is outside the system boundary is an input. Any parameter observable from outside the system boundary is an output.	48
3.2 Visual model of Pulse Oximeter with components, subcomponents, and connections indicating flow of data in between them.	50
3.3 Specification of the architectural model of Pulse Oximeter in Architectural Analysis and Design Language (AADL).	51
3.4 R-C Circuit system where the voltage V_{in} is the input and voltage across the resistor V_{out} is the output.	54
3.5 Hybrid Automata model of a cooling control system with two states each having different dynamics and state transitions governed by constraints on the inlet temperature.	58
3.6 Simulation of a bouncing ball showing the height of the ball with respect to time for different values of the coefficient of restitution parameter, γ	60
3.7 Reachability analysis of the CRAC hybrid automata in Example 9, which shows the values that T_{in} and T_{out} can possibly assume at any arbitrary time.	61

Figure	Page
4.1 CPS modeling requirements and mapping of related works based on the requirements that they address with highlighting of CPS-DAS contributions	64
4.2 CPS-DAS tool architecture consisting of two parts: CPS-DAS modeling framework, which is used to specify a CPS and CPS-DAS Analyzer, which analyzes a CPS model.	65
4.3 BSN as a Global Cyber-Physical System (CPS) with several Local CPSes each having region of interest and a region of impact. Overlapping of ROI _n and ROI _m indicate cyber-physical interactions.	67
4.4 Hierarchical view of the generic constructs of a GCPS which is used to specify a CPS high level architectural model.	69
4.5 Global Interactions between two local CPSes. Intersection between the ROI _n of two LCPS indicate intended interactions while that between ROI _m s of two LCPS indicate unintended interactions.	71
4.6 Pseudocode for CPS-DAS Analyzer, which first parses the GCPS model, computes interactions within an LCPS, and then computes global interactions between LCPSes. Finally, the system properties obtained from the computation is checked with requirements for safety verification. . . .	73
4.7 AADL specification of the GCPS model (a more detailed model can be found in the publication [2])	76
4.8 Implementation logic for Peak Detection (a more detailed discussion can be found in the publication [3])	80
4.9 Summary of case studies showing CPS-DAS usage for safety evaluations that are further discussed in detail in Section 4.3	81

Figure	Page
4.10 Thermal map of fingertip skin for 8 hrs of pulse oximeter operation at 44 °C temperature	84
4.11 GCPS specification of Data center using the AADL language. The specification is for a data center with 50 chassis however only a snippet of the entire specification is shown in the thesis for ease of readability.	87
4.12 Maximum chassis inlet temperature for 80 % utilization under linear cooling model for different energy management policies. Only CM, JPM, and JPCM as representative of non-coordinated cooling, no cooling management, coordinated cooling management, respectively. JPCM allows highest maximum inlet temperatures temperatures (without violating the redline) among all the policies.	89
5.1 A computer controlled wearable infusion pump system.	94
5.2 Random way point and Levy walk mobility models used as different models for the same context. Levi walk model captures the human nature of restricting their movements to specific regions.	96
5.3 a) Drug concentration have different overshoots for different mobility patterns, b) drug concentration profile may depend on the sequence of context changes	97
5.4 Approach for analysis of SMCS under dynamic context changes.	100
5.5 PHMS analysis methodology is shown in the Figure. The context sensors post events in the event queue, the events cause state transitions in the <i>ContextFSM</i> . Each state has a different configuration of PHMS, which are analyzed using the analysis engine.	110

Figure	Page
5.6 Analysis of infusion pump safety under dynamic context changes. The levy walk model predicts higher concentration of insulin than the random walk model, the difference increases with higher probabilities of outdoor excursions.	114
5.7 Skin thermal map for two modes of smart phone operation. Typically temperatures are higher when the smart phone is executing the epilepsy detection algorithm.	116
6.1 Thermal map of human skin for two sensors placed at locations 5 and 15. The sensors were sensing at 60 Hz and were transmitting the sensed value to the base station using ZigBee radio. This workload is typical of a pulse oximeter sensor. The power consumption values are measured from a Smithsoem pulse oximter while the temperature rise is governed by the Penne’s bioheat equation [4]	120
6.2 Assumed system model for Cyber-Physical Systems.	123
6.3 Computation of image from an initial state. A boundary of the initial state is assumed and the dynamic equations are evaluated on the vertices of the boundary. The reachable state at the next time or space step is the convex hull of the images of the vertices.	125
6.4 Conceptual illustration of L1STHA with two modes l_1 and l_2 and two continuous variables v_1 and v_2	127
6.5 The L1STHA model of single channel infusion pump.	130
6.6 The normal and aggregate effect modes in the L1STHA model of multi-channel infusion pumps.	132

Figure	Page
6.7 Example execution of the L1STHA model of a multi-channel infusion pump, for 1000 seconds and over a 50 mm spatial region.	132
6.8 Figure shows STHA model of the infusion pump. PKA model denotes Eqn 6.47. The state transitions are spatio-temporal in nature.	163
6.9 Safe and unsafe initial configurations of the infusion pump. The shaded region includes the initial configurations of the infusion pump that result in drug concentration above prescribed safe values at any point in time and space.	165
7.1 Hybrid automata modeling of artificial pancreas with non-linear Bergmann minimal model governing the continuous dynamics.	171
7.2 Reachable states for the linearized artificial pancreas hybrid automata model.	173
7.3 Variation of linearization error with respect to the discretization time interval.	177
8.1 Architecture of the <i>Health-Dev</i> which has three modules: specification, parser, and code generator.	180
8.2 AADL specification of a sensor and smart phone and their TinyOS and Java codes	183
9.1 Resistor grid and diffusor network synthesis of second order differential.	189
9.2 Probabilistic timed automata composed with mobility models, STHA models, and scavenging source models.	192

LIST OF SYMBOLS

Symbol	Definition	Page
Chapter 2		
<i>C</i>	Set of computing properties	17
<i>P</i>	Set of physical properties	17
<i>I</i>	Set of interaction parameters	17
<i>G</i>	Mapping from computing properties to interaction parameters	17
<i>t</i>	current time	17
<i>H</i>	Mapping from physical properties to interaction parameters	18
<i>x</i>	A point on the x - spatial coordinate axis	18
<i>y</i>	A point on the y - spatial coordinate axis	18
<i>z</i>	A point on the z - spatial coordinate axis	18
<i>K</i>	Cyber-physical interactions	18
<i>R(t)</i>	Risk parameter in artificial pancreas	32
τ	time increment interval	32
<i>BG(t)</i>	Predicted blood glucose concentration	32
<i>BG_{thresh}^{pres}</i>	Prescribed blood glucose concentration	32
<i>BG_{thresh}^{high}</i>	High blood glucose concentration threshold	33
<i>k₁</i>	constant for risk calculation	32
<i>k₂</i>	constant for risk calculation	32
<i>k₃</i>	constant for risk calculation	32
<i>k₄</i>	constant for risk calculation	32
<i>J_{attn}</i>	attenuated insulin delivery rate	32
<i>J_{basal}</i>	basal insulin delivery rate	32
<i>k</i>	patient specific parameter for infusion	32

Symbol	Definition	Page
θ	patient specific parameter for infusion	33
J_{corr}	corrected infusion rate	33
J_{bolus}	bolus infusion rate	33
θ_{TDI}	total insulin intake	33
MS	meal size	33
$X(t)$	remote compartment insulin action	34
$I_{sc1}(t)$	interstitial insulin concentration in the first compartment	34
$I_{sc2}(t)$	interstitial insulin concentration in the second compartment	34
$I_p(t)$	plasma insulin concentration in the first compartment	34
$BG_{sc}(t)$	interstitial glucose concentration	34
Q_1	stored glucose level in first compartment	34
Q_2	stored glucose level in second compartment	34
δ	difference from reference values	34
$u_{in}(t)$	injected insulin at time t	34
$\beta(t)$	meal intake of the patient	34
A	predictor parameter	34
B	predictor parameter	34
E	predictor parameter	34
n	total number of chassis	39
h	inter event interval or event period	39
c_k^{tot}	the number of servers (blades) job k requires	39
r_{ac}	thermal capacity of air flowing out of the CRAC per unit time	39
r_{room}	thermal capacity of air in the data center room	39
f_{ii}	cold supply air fraction going from CRAC to chassis i	39

Symbol	Definition	Page
d_{ij}	heat recirculation coefficient from chassis i to j	39
m	mode of operation of CRAC, $m \in \{high, low\}$	39
t_{sw}	time taken by the CRAC to switch from one mode to the other	39
ω	idle chassis power consumption	39
α	power consumption of a chassis per unit of utilization	39
u	chassis utilization	39
N_h	total number of jobs in event period h	39
$T^{sup}(t)$	air temperature as supplied from the CRAC at time t	39
$T^{sen}(t)$	air temperature at the input of the CRAC at time t	39
T^{red}	manufacturer's redline inlet temperature	39
T_{high}^{th}	high thermostat setting for the CRAC	39
T_{low}^{th}	low thermostat setting for the CRAC	39
$(T_{high}^{th})^{max}$	upper bound on high thermostat setting for CRAC	39
$(T_{high}^{th})_i$	CRAC high thermostat setting requirement for server i	39
ΔT^{th}	temperature difference between the CRAC <i>high</i> and <i>low</i> thermostat settings	39
P_{ex}^m	power extracted by the CRAC in mode m	39
P_j^{full}	power dissipation of chassis j at 100 % percent utilization.	39
P_h^{comp}	total computing power at inter-event period h	39
$P^{AC}(t)$	power consumption of CRAC at time t	39
E_y	energy consumption for algorithm y	39
E_y^m	energy consumption of CRAC in mode m for algorithm y	39
Chapter 3		
h	height of the ball	52

Symbol	Definition	Page
g	acceleration due to gravity	52
γ	coefficient of restitution	53
s	laplace transform variable	53
V_{in}	input voltage	53
S	finite state automata	55
X	set of states in an FSM	55
X_0	set of initial states	55
U	Set of inputs to an FSM	55
Tr	transition set mapping	55
u	an input in the set U	55
x_1	a state in the set X	55
Y	set of possible outputs	55
H	mapping from state to output	55
y_0	an output in the set Y	55
HA	symbol for hybrid automata	57
\mathbb{L}	set of modes for a hybrid automata	57
\mathbb{L}_x	set of initial modes	57
\mathbb{R}	set of real numbers	57
Inv	invariant set	57
l_i	a mode in the set \mathbb{L}	57
ϕ	null set	57
Inv^o	interior of the invariant set	57
δInv	boundary of the invariant set	57
F	mapping from a mode to a function in a compact real set	57

Symbol	Definition	Page
n	number of continuous variables	57
Chapter 4 & Chapter 5		
n	number of nodes in a body sensor network	74
P_c	maximum power of a sensor node with radio on	78
P_{proc}	processing power with radio off	78
P_{radio}	power consumption by the radio	78
d	number of sleep cycles	78
t_s	time taken in the sensing phase	78
t_{TX}	time spent in transmitting packets	78
T_{PKA}	time spent in performing physiological value based key agreement ..	79
P_{PKA}	power consumption of the sensor during physiological value based key agreement execution	79
ρ	density of tissue	82
C_p	specific heat of tissue	82
K	thermal conductance of tissue	82
T	temperature of the skin	82
b	blood perfusion constant	82
T_b	temperature of blood	82
SAR	specific absorption rate of the skin	82
Chapter 6		
V	continuous states space	123
n	dimension of continuous state space	123
\mathbb{R}	set of real numbers	123
J	set of cells composing the continuous state space	124

Symbol	Definition	Page
J_i	a cell in the set J	124
J_j^\square	the interior of the cell J_j	124
J_j^\square	boundary of the cell J_j	124
<i>L1STHA</i>	Linear 1-D space Spatio Temporal Hybrid Automata	125
\mathbb{L}	set of locations	128
l_i	a particular location in the set \mathbb{L}	128
<i>Inv</i>	invariant set mapping from a location to a set of cells	128
ϕ	null set	128
A	$n \times n$ matrix of real numbers	128
B	$n \times n$ matrix of real numbers	128
C	$n \times n$ matrix of real numbers	128
u	$n \times 1$ vector of real numbers	128
t	temporal point	128
s	spatial region	128
$\eta(t, x)$	trajectory at a give time t and a given spatial point x	129
τ	τ time interval	135
x	a spatial point	129
$\eta _x.dur$	duration of a trajectory at a given space point x	129
$\eta ^t.range$	range of a trajectory at a given time point	129
h_t	time discretization interval h_t	141
h_x	space discretization interval h_x	141
s_l	spatial region for a given location l	134
T	time bound for reachability analysis	133
S	space bound for reachability analysis	133

Symbol	Definition	Page
α_t	temporal execution	134
x_t	spatial point at which the temporal execution is considered	136
$Re(\cdot)$	reset conditions for a temporal execution	170
$\alpha_t.dur$	duration of a temporal execution	134
$\eta_k _x$	trajectory of the kth location at a given space point x_t where the temporal execution is considered	134
α_s	spatial execution	135
x_0	boundary of the spatial region corresponding to location l_0	134
$\alpha_s.range$	range of the spatial execution	135
$\eta_k ^t$	the spatial trajectory at a location l_k for a given time τ_s	135
τ'	time at which a temporal discrete transition occurs	171
x'	space point at which a spatial discrete transition occurs	136
ϵ	approximation error for reach set determination	137
M	epsilon reach set	138
$d_H(P, Q)$	hausdorff distance between two sets P and Q	138
I_b	initial condition for the partial differential equation	138
I_0	boundary condition for the partial differential equation	139
ψ	laplace paramter	139
$D_{t,x}(P)$	the state reachable from the initial polygon P	141
$D_{t,x}(P, \gamma)$	γ approximation of the state reachable from the initial polygon P ..	141
$R_{TS}(V_0)$	the reach set at a give temporal and spatial bound starting from an initial configuration V_0	145
H	a constant parameter	146

Chapter 1

INTRODUCTION

Increasingly embedded computing systems in smart and context-aware mission-critical infrastructures are exhibiting tight coupling with their physical environment (Figure 1.1). Examples include: (1) physiological sensors deployed on human body that continuously monitor health and enable fast detection of medical emergencies and subsequent delivery of therapies [5–7], (2) smart buildings that detect absence of occupants and shut down the cooling unit to save energy [8], (3) data-centers or cloud computing infrastructures that use solar energy for cooling purposes [9], and (4) unmanned aerial vehicles (UAVs), that use an image of a terrain to perform surveillance [10]. In such systems either, a) context information from the physical environment is used by the software to improve performance or energy efficiency of the computing units (as in the case of sensors scavenging energy from the human body), or b) feedback from the dynamically changing physical environment is used for automated control of the environment itself (as in the case of artificial pancreas [11], where glucose meter readings from a patient is used to control blood glucose levels). Systems in which the software controls the physical environment or uses context information from the environment for computing operations are called *Cyber-Physical Systems* (CPSes) (Figure 1.1).

1.1 Safety in CPSes

The context driven tight-coupling between the cyber and the physical in CPSes, though advantageous, is subject to new forms of risks that have not been considered adequately in the traditional computing domain. These new types of risks include the cyber element adversely affecting the physical environment (*e.g.*, delivery

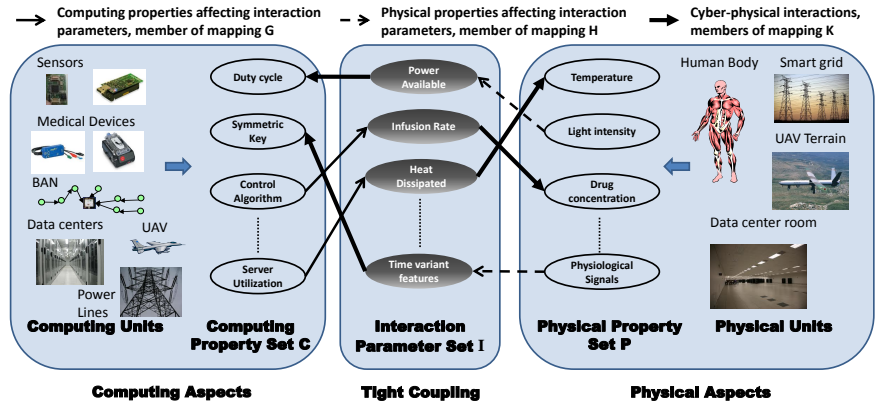


Figure 1.1: Cyber-physical systems, with tight coupling between computing and physical environment through interaction parameters. Solid lines indicate effect of computing operations on interaction parameters, dashed lines indicate effect of physical operations on interaction parameters, and thick solid lines indicate cyber-physical interactions.

of excessive medication) or vice versa (e.g., throttling of servers in a data center due to overheating and lack of cooling). Indeed according to a report published by Networking and Information Technology Research and Development (NITRD) program [12], CPSes are by definition safety critical in nature. Since risks in CPSes may have direct hazardous consequences on the physical environment, CPSes have to be analyzed and guaranteed for safety even before deployment.

CPSes are complex systems with many interacting components exhibiting aggregate behavior [13,14], i.e. properties that are only observed when components operate in coalition and not stand alone. Hence, safety violations may occur due to faults in different components of the CPS including the computing hardware, software, communication medium, physical environment, and the interaction between different components, which often cause unwanted aggregate effects. A classification of CPS safety violations with respect to sources of faults is discussed in Section 2.1 of Chapter 2. Although safety assurance research has an elaborate history dating back to 1970s when the US army declared their verification and validation pro-

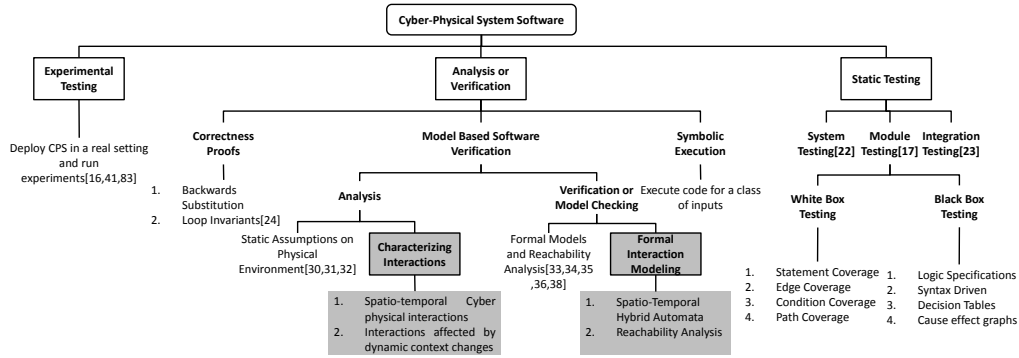


Figure 1.2: Safety assurance approaches for CPS software. The principal contribution of this thesis is facilitating cyber-physical interaction modeling and verification in MBSV for CPSEs.

gram for the safety of their Anti-Ballistic Missile [15], only recently there have been several cases of failures in CPS operation that have been attributed to software faults while interacting with the physical environment. The flight control software of Airbus A320 have been reported to behave erroneously when flying strips are altered (<http://catless.ncl.ac.uk/Risks/8.77.html#subj6>). The GRASEBY infusion pumps have been recalled after cases of potentially fatal over-infusions [16] due to software failures, which were not detected on subsequent testing of the software in isolation.

Although there has been several efforts to verify the software safety of medical devices including Food and Drug Administration’s (FDA) recent endeavors at static testing [17], most of them consider software in isolation and ignore the cyber-physical interactions. Similarly safety research in other CPS domains such as UAVs limit to static testing of the software [18–20].

This thesis focuses on the complications introduced by cyber-physical interactions in assuring safety of the physical environment from hazards caused by faulty operation of CPS software.

1.2 Approaches for CPS Software Safety Assurance

Over the years several approaches for verification of software of a system [21] has been undertaken as shown in Figure 1.2, which have been also applied to CPS software safety but with little success. CPS software often undergoes post-manufacture safety analysis through experimentations on selected test cases in controlled environments. Although experimental safety evaluation is incomprehensive due to limited test cases, regulatory agencies such as FDA depend on such evidences of safety for approving CPSEs in healthcare (medical devices). The inaccuracy in experimental safety evaluation is evident from the large number of recalls faced by several approved and marketed medical device. Reported hazards for such recalls include serious fatalities caused by malfunctioning of surrogate devices or life support systems such as infusion pump control systems. A case in point is the sudden unpredicted over-infusion of drug by the SMITHS MEDICAL Graseby 3300 Automated injection system recently listed in the Maude database [16]. The device was recalled and it underwent the same set of tests and no hazards were reported. Thus, experimental evaluation although necessary, is not sufficient to prove safety.

Static testing of software is another approach that is widely used in evaluating the correctness of code [17, 22, 23]. In this approach, each software component of CPS is first tested in module testing, then the integration of different components are tested for faults, then the CPS is tested as a whole system. Each module can be tested in two ways: a) white box testing, where the internal code of a module is analyzed by testing the execution of every statement (statement coverage), every branch of execution (branch testing), every condition (condition testing), and every possible path taken by the input during the code execution (path testing), and b) black box testing, where the specification of the code as an algorithm is used to

derive logic specifications, decision tables, test syntax, and cause effect graphs for analyzing the correctness. This approach is extensively being pursued by the regulatory agencies such as FDA. FDA has recently suggested the use of static testing tools such as CodeSonar to medical device manufacturers for proving safety of their software [17]. Note that often such testing are infeasible, e.g., the number of paths taken by an input may be infinite as in the case of while loops and hence traversing every possible path may not be feasible. Thus, such testings are inherently limited.

While experimental and static testing work on individual test cases or configurations of the system, analysis operates on a set of test cases at a time and hence are typically faster and more comprehensive than the testing process [21]. Once the implementation of the software is available correctness proofs using loop invariants or backward substitution techniques and symbolic execution of the code can be performed. Correctness proofs are typically complex and obtaining loop invariant is not feasible for several software implementations. Although researchers have devised automated loop invariant deduction techniques [24], they are mostly for synthesized code and impractical for manual implementations. Symbolic execution of the code involve classifying inputs based on their effects on faults in the software. A subset of inputs are then derived, which covers all the faults in the software. Representative inputs from the subset are then used to execute the code and check for faults. Such combinatorial analysis technique is prevalent in VLSI system design [25] and has been recently suggested for software analysis by National Institute of Standards and Technology (NIST) [26].

All the abovementioned techniques however, consider the software of CPSes in isolation and ignore the cyber-physical interactions. Further, sole reliance on these testing techniques for safety assurance is not cost effective, since they necessitate the presence of CPS software implementations, which are typically expen-

sive. Any safety hazard detected after implementation may lead to costly redesigning and re-manufacturing. Finally, hazards in case of faults during experimentation may cause irrecoverable loss of property and life. Hence, CPS software safety has to be analyzed before implementation at an early design stage. An approach that is widely used in this regard is model based software analysis and verification (Figure 1.2).

1.3 Model Based Software Analysis and Verification

The notion of model based software analysis and verification fall under the generic paradigm of model based engineering (MBE). MBE is a methodology for reasoning about the properties of a system using simplifying abstractions. Such reasonings aide in the development of a system by eliminating flaws and checking adherence to preferences at an early design stage. MBE has been used extensively to develop, manage, and verify, different components of a system. Some of the common usages of MBE include: a) developing system as a whole, model based systems engineering (MBSysE) [27], b) managing processes in a system, model based process engineering (MBPE) [28], and c) developing the software of a system, model based software engineering (MBSE) [29].

MBSE involves stages representative of the life cycle of software [29]. It includes: a) requirements analysis - when constraints such as safety, and preferences such as development platform for the software implementation are gathered and analyzed, b) development - when the software is implemented in the chosen platform, c) verification - when the operation of the software is checked for conformity to the constraints and preferences set forth in the requirements analysis stage, d) deployment - when the implemented and tested software is customized and installed on real systems, and e) maintenance - when the software is modified or enhanced to be compatible with new systems or to cope with newly occurring faults. Model based

safety analysis and verification (MBSV) stage of MBSE, is applicable to the problem of software safety verification of CPSes and have recently received considerable focus [18–20, 30, 31].

MBSV depends on an abstract representation of a software design in a mathematical form or as an algorithm. Any such abstract representation is called a *model*. The requirements of the system such as safety, is expressed as constraints on the system properties, which are variables in the model. Mathematical models can then be used for theoretical analysis (verification) while algorithmic models can be used in simulations (analysis) for checking compliance with the requirements. Once the models are analyzed or verified against the safety requirements, they are converted into implementations, which are again verified through experimentation. Thus, MBSV provides early feedback that can be used to eliminate flawed design before implementation. This quickens the process of development, reduces the cost, and potentially guarantees safety.

This thesis considers model based software analysis and verification for CPSes in presence of context driven cyber-physical interactions. First, the challenges to modeling CPS software are investigated by studying several examples in two specific domains - healthcare and cloud computing. Secondly, a number of safety analysis and verification solutions are proposed.

1.4 Related Work

MBSV has been widely used for analyzing and verifying the software of CPSes. In the medical device domain, FDA is performing collaborative research to apply MBSV theories for comprehensively proving safety properties of medical devices before marketing. Several efforts [30, 31] have used MBSV in this regard. Also in other CPS domains such as UAVs, there has been considerable work [18–20] in guaranteeing software safety using MBSV. However, such efforts isolate the device

software and physical dynamics with which it interacts through static assumptions on the properties of the environment. Such isolated analysis may be valid for evaluating harmful side effects of CPS operation on the physical environment. But for CPSes where the computing unit incorporates information from the environment in its decision making (e.g., artificial pancreas), such isolated analysis fail to provide accurate safety evaluations. To this effect, there has been a considerable push towards hybrid approaches for modeling and analysis of CPSes where the discrete time operation of the software and the continuous dynamics of the cyber-physical interactions can be modeled in a single framework. For example, for smart infusion pump control systems simplified hybrid models are used for verifying drug overdose [32]. Another hybrid modeling technique with timed automata has been used to verify pulse oximeter and infusion pump feedback control systems for administering analgesic [33]. Hybrid modeling has been of great interest in the UAV CPS domain where the cooperation of the computing logic and the aerodynamics are modeled and verified [34, 35]. Several work in the more generic context of CPS has been proposed, where the authors have developed new hybrid models such as O-minimal models [36], or applying control theory to CPS safety assessment [37] or online verification techniques [38], or incorporating real time guarantees in CPS design [39]. However, such research efforts either attempt to use the existing modeling construct such as hybrid automata or timed automata in the new domain of CPS or consider the CPS operations with only temporal dynamics. In doing so simplifying assumptions to the cyber-physical interactions are made, which introduces inaccuracies in their safety conclusions.

These assumptions ignore at least one of the five salient properties of CPSes identified in this thesis. These properties, discussed in the following section, necessitate new modeling and analysis techniques. Solutions to a subset of these require-

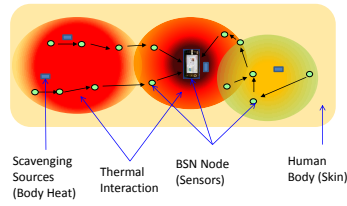
ments are provided in this thesis, while others are open problems worthy of doctoral research. This need for new modeling and analysis techniques for CPS has been recognized by the National Science Foundation (NSF), which has invested billions of dollars on CPS modeling and analysis. The research efforts in this thesis has been largely funded by three such funding sources.

1.5 Challenges of MBSV for CPSes

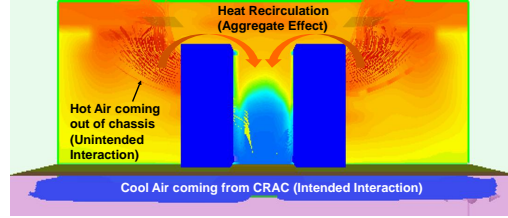
Following are the five salient properties of CPSes:

- **Hybrid nature:** To achieve safety in a CPS, the computing unit needs to extract diverse types of information, related to say thermal, mechanical, and electrical properties of the environment. Design and analysis of CPS thus requires in-depth understanding of the characteristics of these information and their effects on the computing operation. As Willems has aptly pointed out [40], systems researchers should incorporate detailed behavioral characterizations of the physical environment in the theories and techniques of computer science. Lack of such considerations result in serious violations of safety properties. For example, in case of a pulse oximeter in a Body Sensor Network (BSN), if the control of the sampling frequency is not aware of the temperature rise on human skin, severe burn hazards can occur [41]. This stresses the need for a unified inter-disciplinary approach towards CPS design for achieving safety, that combines theories from computer science with other disciplines of science and engineering. Computing systems typically have discrete operations with several states or modes of operation while physical environment is continuous in nature modeled using differential equations. Hence, an unified approach will necessitate hybrid models that can capture both discrete and continuous operations in a single construct.

- **Aggregate effects:** Many CPSes, such as BSNs and data centers, are further distributed in nature, i.e. they comprise of more than one computing entities (henceforth referred as the computing nodes) distributed across the environment. Figure 1.3 shows the deployment of the aforementioned two representative CPSes. As shown in the figure, the heating effects, caused by the operations in the sensors and the computing servers in BSNs and data centers, respectively, are the unintended interactions. The distributed nodes often perform concurrent operations; thus causing aggregation of the detrimental impact on the environment from multiple nodes. For example, in data centers, the recirculation of accumulated heat (referred as heat recirculation in Figure 1.3b) from multiple servers can cause higher increase in the operating temperature than the heat from a single server. Similarly, in BSNs, concurrent operations in more than one sensors can have accumulated heat effect on certain parts of the body depending on the deployment of the sensors (Figure 1.3a). Thus, CPSes may have aggregate energy interactions, which exist only in presence of a network. Any modeling solution of CPSes should be capable of characterizing aggregate effects in case of networked operation.
- **Spatio-temporal interaction:** The interactions in a CPS, are spatio-temporal in nature. For example, in case of infusion pumps in a BSN, that inject a given drug (*e.g.*, insulin) into the human body, the effect of the drug on the physiological parameters (*e.g.*, blood glucose concentration) vary over space as well as time. The effect is maximum at the sight of infusion and decreases at locations away from the sight being affected by the blood perfusion. Capturing spatio-temporal effects requires multiple independent variables that determine the evolution of the model. Researchers have extensively studied models that evolve over a single independent variable, most commonly



(a) BSN with on body physiological sensors (EKG, PPG and SPO2).



(b) Data Center with two rows of racks each containing several thousands of servers.

Figure 1.3: Typical deployment of two representative CPSes showing the distributed nature of the computing nodes, intentional and un-intentional interactions with the physical environment, and the aggregate effect of these interactions because of concurrent operations in the distributed nodes.

time [18–20, 30, 31, 34, 35, 42, 43]. However, there exist limited efforts to capture model evolution over both space and time.

Most of the current research in spatio-temporal modeling concentrate on discretization of space and time, which introduces error in estimation [44–47]. For CPS modeling such discretization errors can lead to wrong safety conclusions. Further, fine grain discretization of space and time leads to explosion of variables to the point of becoming intractable. Hence, analysis of CPS models should consider a continuous evaluation of spatio-temporal dynamics.

- **Non-linear interaction:** Non-linearities are inherent in the physical environment, e.g., the infused drug concentration variation over time inside human body follows an error function curve. Hence, CPS models can be non-linear in nature. Non-linearities are embodied in many different forms in a model including delays and multiplicative terms. Theoretical analysis techniques only exist for specific types of non-linearities which may not apply to a large class of CPSes.

- **Dynamic context changes:** The physical system in a CPS is constantly undergoing change in context, *e.g.*, weather changes, change in temperature, change in physiological condition. Mobility, for example, is an primary cause of such context changes in the system. These changes affect the operation of the computing units in the CPS, for example, movement from indoor to an outdoor environment changes the packet delivery rate (PDR) of the wireless medium, which can trigger a radio power management algorithm in the sensors of a BSN. Safety analysis of CPSes should consider such dynamic changes in the environment as a part of the model.

Hence, MBSV for CPSes needs to consider the five above-mentioned characteristics of CPSes. Current techniques and modeling abstractions lack in several aspects to be applicable to CPSes. A Federal Networking and Information Technology Research & Development (NITRD) report has indeed identified the importance of new design abstractions for CPS [48]. The research in this doctoral study investigates the available modeling and analysis approaches for CPSes and evaluates their capabilities to capture the five salient properties of CPSes. To this effect, new modeling abstractions and analysis techniques are proposed and evaluated by applying them to two CPS domains - healthcare and cloud computing infrastructure. A brief overview of contributions to MBSV for CPSes is discussed in the following subsection.

1.6 Thesis contributions

The thesis focuses on model based safety analysis and verification of CPS software in presence of dynamic context driven cyber-physical interactions. Any MBSV approach toward CPSes should be able to capture the five salient properties - a) hybrid operation, b) spatio-temporal evolution of system dynamics due to cyber-physical interaction, c) aggregate effects due to concurrent operation of networked systems, d)

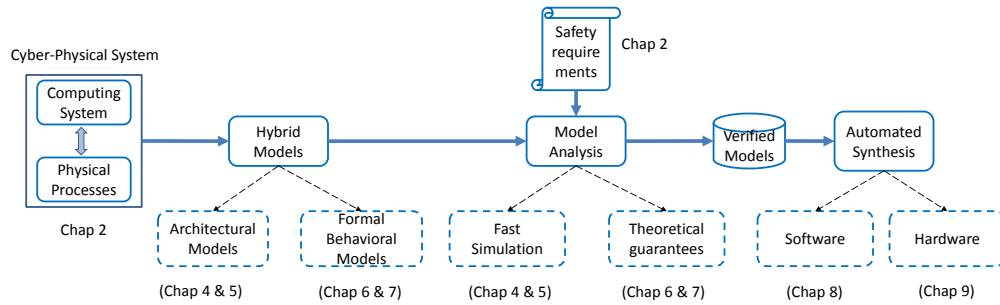


Figure 1.4: Stages of model based safety verification of CPS software and thesis contributions. The MBSV consists of three stages modeling, analysis and synthesis. The thesis has contributions in each of the three stages.

non-linear dynamics, and c) dynamic context changes due to random environment and human mobility. This section discusses three principal stages of MBSV and the contributions of this thesis in each of the stages. Three principal stages of MBSV, as shown in the Figure 1.4, are - a) Modeling, b) Analysis, and c) Synthesis.

Modeling Stage

MBSV depends on models, which are abstract representation of the system components and processes. Models can be of various kinds but can be principally classified into two different types - a) Architectural models and b) Behavioral models.

Architectural models represent a system as a connected graph components or subsystems. Each subsystem may have properties associated with them. A connection between two component may denote flow of information or a subcomponent relationship. Architectural models are useful for easy specification and fast simulation of test cases.

Behavioral models represent the internal process of a system using different types of abstractions. It can be transfer function representation, which views the system as a black box and gives the variation of output for a given input as a mathematical function or a formal model that represents the system behavior using states and transitions between them.

Contributions: To capture the five salient properties of CPSes in models the thesis makes the following contributions -

- An architectural modeling solution BAND-AiDe [2] for BSNs or CPS-DAS for CPSes in general is proposed that can capture linear or non-linear aggregate spatio-temporal physical dynamics as well as computational aspects of CPSes.
- The CPS-DAS is then extended with capability to model cyber-physical interactions under dynamic contexts [49].
- On the behavioral modeling front, a novel hybrid model, spatio-temporal hybrid automata (STHA) is proposed that can model linear aggregate spatio-temporal effects of cyber-physical interaction as well as the discrete computational operations of a CPS in a single construct.
- For non-linear CPSes having only temporal dynamics, an error bounded linear approximation technique is proposed to convert the non-linear system into linear one dimensional hybrid automata.

Analysis Stage

The models in MBSV are used to analyze system parameters and verify safety requirements before implementation or deployment. Model analysis can be broadly divided into two categories - a) simulation of test cases, and b) reachability analysis.

Simulation analysis is necessary for fast testing of given configurations of the CPS. This is typically helpful to test worst case operating conditions or for estimating the amount of resource required for a given operation.

Reachability analysis is more comprehensive in the sense that it takes a set of configurations of the CPS model and outputs the possible values that the

system parameters can reach at any arbitrary time. Reachability analysis is helpful for providing guarantees on system behavior.

Contributions: The thesis explores both simulation and reachability analysis solutions for analyzing safety of CPSes. Following are the contributions -

- The architectural models are used for simulation through a CPS-DAS analyzer proposed in the thesis.
- Reachability analysis is performed on the formal models discussed in this thesis to provide safety guarantees on the CPS models. CPS models have non-linear and spatio-temporal dynamics for which there are limited reachability analysis techniques. In this thesis, two real world case studies are considered - i) artificial pancreas, where the cyber-physical interactions are non-linear in nature but not spatio-temporal, and ii) infusion pump drug diffusion or pulse oximeter thermal effects, where the dynamics is linear but spatio-temporal. The non-linear case is handled using suitable linearization techniques and then applying existing reachability analysis techniques for linear hybrid models. For STHA models, a novel reachability analysis algorithm is proposed.

Synthesis

The output of the analysis stage is a model which meets the safety requirements. The synthesis stage takes the architectural or behavioral model and converts it into an implementation in a standard testbed. The synthesis can be in hardware or software. Automated hardware synthesis necessitates a standardized prototype testbed such as a field programmable gate array (FPGA) or field programmable analog array (FPAA). Software synthesis on the other hand necessitates a code generator.

Contributions: This thesis discusses an automated code generator from high level specification, *Health-Dev* [50], in the healthcare CPS domain for implementing wire-

less health systems such as body sensor networks controlled by a smart phone. Ongoing extension of this thesis includes emulation of cyber-physical interactions using hardware. In this regard, field programmable analog arrays (FPAA) are used for synthesizing differential equations in hardware that represent the continuous dynamics. Such synthesis is helpful in fast estimation of the cyber-physical interactions, which can be used to profile CPS properties to be used in simulation or reachability analysis. It can also serve as a physical system emulator using which the CPS software can be tested for interaction safety.

In the next few chapters, the thesis gives a detailed description of its contributions with examples in different CPS domains, analyzes the usefulness and drawbacks of the proposed solutions, and also discusses open research problems worthy of PhD level research on CPS safety verification. Let us begin with a generic definition of CPS, some example CPSes, and associated safety issues.

Chapter 2

CPS DOMAINS AND SAFETY ISSUES

A CPS, as shown in Figure 1.1, consists of embedded computing units, which tightly interact with their physical environment to provide critical functionalities such as early detection of health problems, securing sensitive data, and enabling long term uninterrupted operation. The computing units of a CPS can be characterized by a set of quantitative properties, C . These properties are related to the computing operation and are functions of the type of application executed. For example, members C can be the utilization of a server in a data center, the initial concentration input to a drug infusion control algorithm [51], the duty cycle of a sensor, or a 128 bit key for encryption during communication. The physical environment in a CPS can be similarly characterized by a set of quantitative properties, P . Examples of physical properties include time varying physiological and environmental signals such as, temperature, humidity, and amount of sunlight.

In a CPS, the properties in C are closely related to those in P through physical processes that cause variation of the properties in the physical environment. Such physical processes can be characterized by a set of *interaction parameters*, I . The interaction parameters can be associated with both the computing and physical properties in a CPS. Typical examples of interaction parameters include heat transferred from the servers in the data center to the ambient air, amount of energy harvested from the environment, or frequency domain features of physiological signals. Both the physical and computing properties affect the interaction parameters. The computing properties are time varying. Hence the mapping between the sets C and I can be represented by $G : C \times t \rightarrow I$ (thin solid arrows in Figure 1.1), where t is

time. The properties of the physical environment vary over both space and time. For example, temperature varies from place to place in a data center and the intensity of sunlight is low under shade and also has diurnal variation. Hence the mapping from the physical properties to the interaction parameters, $\mathbf{H} : \mathbf{P} \times t \times \{x, y, z\} \rightarrow \mathbf{I}$, is spatio-temporal in nature (dashed arrows in Figure 1.1), where $\{x, y, z\}$ represents a point in the coordinate space.

In practice mappings in \mathbf{G} either have to be determined by performing profiling experiments, *e.g.*, utilization to power curves for a server in the data center [52], or can be a result of the execution of an algorithm. On the other hand, the mappings in \mathbf{H} can be determined either by building of models of the physical processes, *e.g.*, electro-mechanical models of energy obtained from piezoelectric devices [53], or can be obtained through signal processing, *e.g.*, extracting security keys from physiological signals [54]. The interaction parameters define cyber-physical interactions as follows:

Definition 1 *A cyber-physical interaction is an inverse mapping \mathbf{K} from a subset of \mathbf{I} to a subset of \mathbf{P} or \mathbf{C} .*

Example 1 *Pulse-oximeter thermal effects: In case of a fingertip pulse oximeter operation [41], the sampling frequency (\mathbf{C}) affects the amount of heat dissipated (\mathbf{I}). Heat dissipated as a function of frequency (\mathbf{G}) can be obtained through power profiling of the pulse oximeter. The effect of the dissipated heat on the temperature rise of the human body (\mathbf{P}) is characterized by the Penne's bioheat equation [4]. Such a mapping is an example of \mathbf{K} . The specific heat and skin conductance of the human body also affect the temperature rise through mapping \mathbf{H} , which has to be experimentally characterized. □*

Example 2 *Drug infusion: Infusion pumps operate in close loop with physiological sensors such as glucose meter or SPO2 sensor to control drug infusion. The infusion rate (**C**) affect the drug concentration in the blood (**I**) through the diffusion process. The diffusion process (**G**) can be characterized by the pharmacokinetic model [55]. The drug concentration then affects physical properties (**P**) such as blood oxygen level, unconsciousness, or cell death rate in case of chemotherapy through physiological processes (**K**) such as change in action potential. A control algorithm in the infusion pump takes the physical properties as input and adjusts the infusion levels so as to achieve the desired physiological effects while avoiding hazards such as respiratory distress [55].* ..

Broadly, cyber-physical interactions can be of two types: a) *intended interactions*, which refer to the usage of information from the physical environment for performing useful computing operations (Example 2) and b) *unintended interactions*, which refer to the side effects of operation of the computing units on the physical environment (Example 1). Further, in case of networked CPSes, there are often combined effects of the individual interactions called *aggregate effects*, as observed in multi-channel drug infusion. A case in point is the increased death rates of cancer cell when α monoclonal antibody and mathotrexate drugs are infused simultaneously rather than when infused sequentially [56]. According to a recent report by NITRD [12], CPSes are safety critical infrastructures. Safety hazards can occur due to failures in different components of CPSes. In the following section, let us consider some of the major types of safety issues occurring in a CPS.

2.1 Safety in CPS

Safety is a property of any system by virtue of which it can be guaranteed that there will be no harm to the infrastructure and to the physical environment during normal

or faulty operation of the system. In the literature, researchers have concentrated on different components of a system and have defined safety in specific contexts of the computing hardware, network, and software. One of the unique features of a Cyber-Physical System (CPS) is the interaction of the computing unit with the physical environment. Hence, for CPSes the safety concerns are related to the interaction between the computing device and the physical environment.

The most generic definition of safety for a CPS can be found in the ISO 60601 standard for safety of medical electrical equipment. ISO 60601 defines safety as the avoidance of hazards due to the operation of a medical device under normal or single fault condition [57]. This definition can also be applied to a CPS in general non-medical domains by broadening the scope of hazards considered, including faulty operation of the computing unit, radiation leaks, thermal effects, biocompatibility issues, software failures, mechanical, and electrical hazards. Hence, safety for CPS has to be considered in both CPS (hazard due to interaction) and non-CPS (hazard due to failure in computing devices) sense. In the literature, several research endeavors have considered different safety aspects of a CPS, which are listed below.

Scenario safety: It considers the safety of the CPS and its environment from a the high level decision making perspective. It considers how the CPS handles random hazardous events occurring in the environment potentially causing harm to life and infrastructure if not mitigated called criticalities. Ensuring scenario safety guarantees that under a given number of critical scenarios the CPS will mitigate all of them and return to a normal state, where there are no more critical events. This notion of scenario safety can be applied to any system (CPS or non-CPS) and involves identifying critical situations of a system, developing methodologies to handle critical conditions, and a framework to manage multiple criticalities occurring at the same

time. Example research in this regard includes the criticality response planning, evaluation, and actuation [58] [59] framework developed at the IMPACT Lab, ASU.

Network safety: CPS generally involves a network of computing units communicating with each other through wireless or wired channels to achieve mission critical and smart operations. Network channels have problems of contention when the number of communicating devices increase. Further, the wireless channel is prone to errors such as bit errors, burst errors, and multi-path fading errors. Under this circumstances the network safety ensures that information transferred from one device to the other is not corrupted, reaches within a given amount of time, and is not lost due to errors in the channel. Ensuring such network safety involve an in depth analysis of the characteristics of the communication channel and the operation of the computing device under loss of information due to network errors. This form of safety is also not limited to a CPS and is applicable to non-CPSes also since it does not require analyzing the interaction between the computing device and physical environment. Many medical responses would involve network of medical devices and it is important to analyze the impact of network delays and packet drops on critical care operations. Such evaluation of network safety in medical device networks has been performed by Gehlot in Villanova University [60]. However, in recent years, there has been an increased interest in using the human body as the communication channel in several medical applications, where in-vivo sensors are deployed [61]. In such cases, the network safety will involve characterizing the human body as the communication channel. However, such analysis is also not CPS specific, as it only characterizes the properties of the human body and does not analyze the interaction.

Software safety: This is a broad area of research and is related to the operation of the software of the CPS computing devices. It is applicable for both CPS and non-CPS systems. For non-CPS systems several notions of software safety exists:

1. *Code safety*, which considers safety from coding errors such as infinite while loops, unreachable conditions etc [62].
2. *Control system safety*, which considers safety from undershoot, overshoot, instability and long settling times (investigated as part of the design of infusion pumps [63]).
3. *Interaction safety*: For CPS specific software safety a new domain of *interaction safety* has been defined in this thesis, which considers the cooperation of the software of a CPS with the dynamic physical environment. Traditionally, researchers have focused on bypassing this cooperation characterization and transforming the safety assurance problem into a well understood problem in computer science such as formal model reachability analysis. In this regard, several static assumptions on the physical environment has been considered, which abstract out the dynamic nature of the physical environment. For example, in works such as [30, 31], infusion pump software has been modeled using a timed automata. The diffusion process is simplified so that the drug concentration in the blood is incremented by the infusion rate instantaneously. The problem of safety assurance is consequently reduced to developing bug free software or a control system analysis problem, which are essentially the non-CPS approaches discussed before. Such simplified notion of safety, however, may not entirely capture the hazards resulting from the dynamic cyber-physical interactions. For example, infusion pumps for chemotherapy require characterization of the spatial extent to which the drug diffuses. In case of pumps used for anesthesia [55], the safety analysis requires the time taken for the drug to reach a particular concentration. Hence in order to guarantee safety of CPS software it is necessary to accurately characterize the spatio

temporal dynamics of the physical environment and its tight coupling with the computing units. In essence more focus is needed on the *interaction safety*. Interaction safety hazards can occur due to different kinds of cyber-physical interactions:

- *Interaction between two computing units:* Cyber-physical interactions of two computing units in different CPSes may affect each other's operation in hazardous ways. Recently headphones are reported to interfere with pacemakers of heart patients (<http://www.medicaldevicesafety.org/>). The electromagnetic interaction of the headphone with the patient's body gets coupled with the electromagnetism induced by the pacemaker on the patient's heart and deactivates it.
- *Interactions from computing units to the physical environment:* Cyber-physical interaction between the computing units and the physical environment may have harmful effects on the physical environment.
- *Interaction from the physical environment to the computing units:* The operation of the physical environment may impose hindrance to the operation of the computing unit. For example, tissue growth around the implanted sensors can hamper sensing and communication capabilities.

Addressing interactions safety is a challenging task. Principally, it requires exact understanding of the physical processes of the environment and the properties of the computing unit that affect the physical processes. The Table 2.1 summarizes the different definitions of safety and also identifies, CPS specific ones. In this thesis, special focus is given on interaction safety of CPSes.

This thesis considers interaction safety in CPSes and provides MBE solutions to safety analysis and verification. Before going into the details of the solutions

Table 2.1: Definition of different safety categories in a CPS is listed. The safety categories are further classified as CPS specific, non-CPS or applicable to both

Definition of Safety	CPS specific	non-CPS	both
Scenario Safety		✓	
Network Safety		✓	
Code Safety			✓
Control Safety			✓
Interaction Safety	✓		

let us first discuss in detail some CPS examples that will be considered as case studies for illustrating the proposed solutions.

2.2 Ayushman Pervasive Health Monitoring System (PHMS)

Ayushman is a smart health infrastructure developed in the IMPACT Lab for privacy ensured continuous health monitoring of ambulatory individuals. The Ayushman has a multi-tier architecture enabling management of sensors, secure storage and dissemination of data, access control of user health history, query processing, service discovery and context processing. At the core of the Ayushman PHMS is a BSN consisting of a number of physiological as well as environmental sensors such as photoplethysmogram and humidity sensors and a smart phone serving as the computation and communication hub. In Ayushman we consider three different contexts which vary in their hardware software configurations, communication protocols, power management techniques, and energy sources.

Home Context

In this context, the user is being monitored in a home environment.

Requirements: The principal requirements from the PHMS are: 1) thermally safe operation of wearable device, 2) low energy consumption so as to prolong battery life, and 3) early detection of health emergency such as arrhythmia.

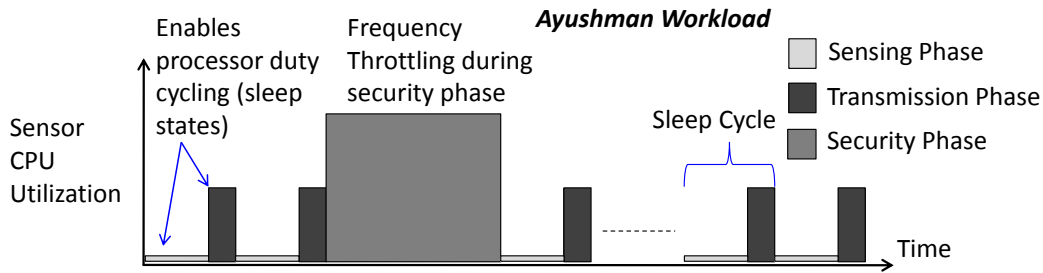


Figure 2.1: Ayushman workload with varying computation and communication requirements. Computation includes sensing and signal processing for physiological value based security. The sensor processor is duty cycled and the frequency throttled while the radio is kept off during the computation phase.

Hardware Configuration: The PHMS consists of physiological sensors such as electrocardiogram (ECG), photoplethysmogram (PPG), and galvanic skin resistance (GSR) sensors, and environmental sensors such as temperature and humidity. The Shimmer sensors (www.shimmer-research.info) is used for ECG and GSR, and the TelosB sensors used for environmental signals are commercially available programmable sensor platforms, which have a micro-controller, such as MSP430 (xbow.com) for computation and a radio for communication, such as Chipcon (www-inst.eecs.berkeley.edu/~cs150/Documents/CC2420.pdf) or Bluetooth radio. The PPG sensor is medical grade and is not programmable. The smart phone is assumed to have an Atom processor interfaced with a Bluetooth radio for communication purposes. The Intel Atom processor provides six sleep stages of which the deep sleep state (C6) consumes the least amount of power. It also supports eight throttling modes in which the processor operating frequency can be modulated. The sleep states and the throttling modes are controlled through operating system Advanced Configuration and Power Interface (ACPI).

Software Configuration: In Ayushman (as shown in the Figure 2.1) the sensors in the PHMS sense physiological data for t_s seconds and store them in local memory. After

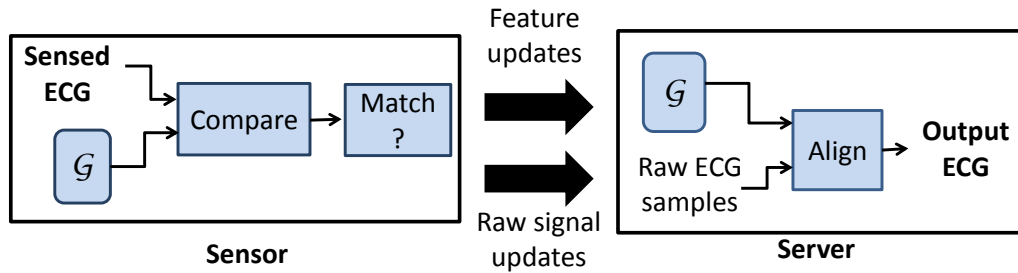


Figure 2.2: Generative Model based Resource Efficient Monitoring. The sensor and the base station uses a model to reduce communication. The sensor only transmits model parameters to the base station which in turn uses the model to reconstruct the data.

t_s seconds they transfer the data to the base station in a single burst. Every communication is secured by encryption with a secret key, which is established between each pair of PHMS devices. Key agreement between two sensors is performed following the Physiological value based Key Agreement (PKA) [64] protocol. In this protocol the two participating sensors sense the same physiological value (Photo plethysmogram signals [64]) and perform complex signal processing including FFT computation, peak detection, polynomial evaluation and quantization to derive features. The features are used to hide a secret key using the fuzzy vault paradigm and is exchanged between the two sensors (known as Vault transfer). This secret key is used for further communications between the two sensors. To maintain key freshness PKA execution is required every 24 hrs.

Energy Source: Since the user is at home, we assume that all the PHMS devices are charged using batteries. While at home this should be easy and noninvasive.

Radio Communication Protocol: In the home context, the sensors can communicate with the smart phone using either ZigBee or Bluetooth. The smart phone communicates with a laptop using the wifi. To save communication energy and enable long term monitoring with prolonged battery life, the sensors use model based data transmission paradigm [65]. In such a technique as shown in Figure 2.2, the sensor

compares the sensed signal with a model. If it matches it does not send any data, in which case the base station uses the model to regenerate the signal. If it does not match the sensor sends the sample by sample data which is used by the base station to learn a new model. Employing this technique an energy savings of 42:1 for ECG [65] and 300:1 for PPG [66] can be obtained. The model based communication technique can be applied here since the physiological signals have a definite structure.

Human Body: The safety of the PHMS depends on the thermal effects of the individual nodes on the human body. The physical properties of the human body are obtained from previous literature (<http://niremf.ifac.cnr.it/tissprop/>) where the authors had done experimental measurements in clinical trials. The physical phenomenon governing the transfer of heat from the PHMS nodes to the human body is given by the mathematical formulation by Penne [4].

Roaming Context

In this context the user is either in travel or is out for a jog doing exercise.

Requirements: The principal requirements are: 1) reliable data communication under wireless errors and low packet delivery ratio (PDR), which is typical of outdoor environments [67], and 2) batteryless operation since recharging is not possible when outdoor.

Hardware Configuration: Same as the Home context.

Software Configuration: The application workload is the same as in Home context. However, since there scarcity of energy source several power management techniques are applied in software: 1) employing radio sleep scheduling techniques, 2) enabling processor duty cycling in individual PHMS nodes, and 3) enabling processor frequency scaling in PHMS nodes. The workload for Ayushman consists of intermittent high computation and communication phases followed by longer periods

Table 2.2: Power available from different scavenging sources in the environment and the human body. The scavenging hours indicated are gross estimates for the case study based on the study by Paradiso [1].

Scavenging Source	Available Power (W)	Scavenge Time (Hrs)
Body Heat	0.1 - 0.15	24
Ambulation	1.5	2
Respiration	0.42	6
Sun Light	0.1	3

of inactivity. In periods of low computation and communication, when the sensors are sensing data (Figure 2.1 short and wide block), the processor can be put to a low power sleep state and the radio can be shutdown. However, during data transmission phases (Figure 2.1 tall and narrow block) both the processor and the radio has to operate simultaneously. In phases where only computation is required, as in the security phase (Figure 2.1 tall and wide block), the radio can be shutdown while the processor frequency can be modulated to control the energy consumption.

Energy Source: We consider four sources of scavenging energy based on their ease of use for the host: 1) body heat, 2) respiration, 3) ambulation and 4) sun light. The authors in [1] provide energy models of scavenging sources which we use in our case-study. Table 2.2 gives the available power from the scavenging sources and also the time duration for which they can perform the scavenging operation.

Radio Communication Protocol: The basic radio protocols used are the same for the Home context. However, given the low PDR, high wireless channel errors are possible. Hence, techniques such as retransmissions and dynamic power management are used which reduces the energy savings of the model based data communication protocol.

Hospital Context

In this context, a medical emergency has occurred and the user has to be admitted to a hospital.

Requirements: The principal requirements are: 1) high fidelity data sampled at recommended sampling rates, and 2) hazard free operation of medical device control systems such as infusion pumps.

Hardware Configuration: Medical grade devices and control systems are used instead of commercially available sensor platforms. An example is the infusion pump drug control system, where an infusion pump diffuses analgesic into the user. The pump receives feedback on the level of consciousness of the user from the blood oxygen level through a pulse oximeter sensor. The control algorithm of the infusion pump maintains a safe level of analgesic concentration in the blood so that it does not cause respiratory distress in the user [55].

Software Configuration: The infusion pump has a specialized control algorithm programmed in it, which continuously sends state information and receives control inputs from the server.

Energy Source: Energy is obtained either from the batteries or directly from the mains.

Radio Communication Protocol: Bluetooth, Wifi, and ZigBee are most common protocols. Often the devices are wired directly to the server.

Context changes

Context changes occur due to random events triggered by: 1) user mobility, modeled using mobility models such as random way point and Markovian model, 2) emergency events such as detection of fall, epileptic seizure, arrhythmia, and 3) user settings, such as activation and deactivation of the PHMS. The different contexts

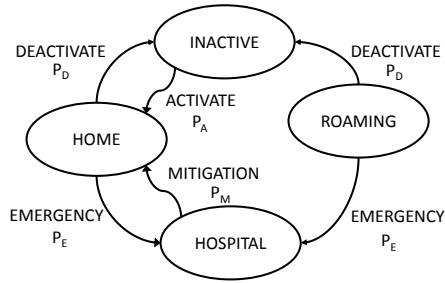


Figure 2.3: Finite State Automata representation of contexts and context changes, *ContextFSM*, for the Ayushman PHMS. The transitions are governed by random models of contexts such as mobility or occurrence of physiological conditions such as epilepsy.

can be represented as states in the *ContextFSM* and the events can cause transition from one state to the other. The events can be generated using random number generators or using the mobility models as generative functions. The *ContextFSM* for the Ayushman PHMS is shown in Figure 2.3.

2.3 Artificial Pancreas

Artificial pancreas are wireless control systems, which infuse insulin to control the blood glucose level of a type 1 diabetes patient and in the process obtain feedback using glucose sensors. On careful review of several device design submissions, we derive a generic architecture of artificial pancreas as shown in Figure 2.4. Artificial pancreas consist of an infusion pump, a glucose meter, and a remote base station operating as a closed loop control system. The remote base station runs a discrete control algorithm that takes the glucose meter reading as feedback and the required blood glucose level as the set point and computes the future infusion rate. This future infusion rate information is transferred to the insulin pump over the wireless channel, which then administers insulin at the requested rate. The glucose meter senses the blood glucose level at specified sampling rates and sends it back to the base station using the wireless channel. In the following subsections we describe

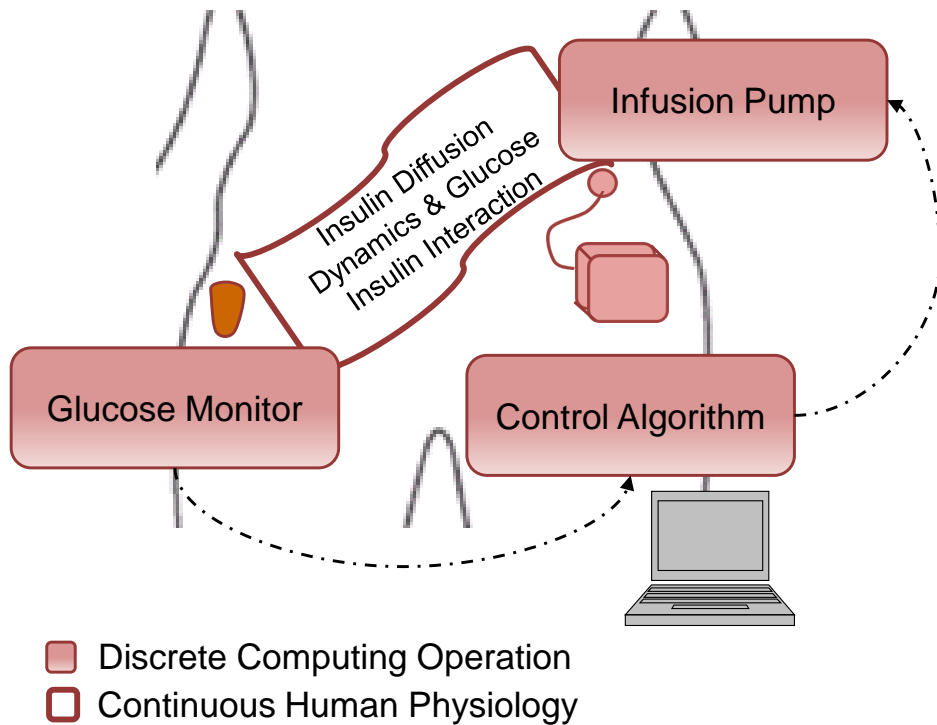


Figure 2.4: Generic Architecture of Artificial Pancreas, which consist of an infusion pump, a glucose monitor and a remote controller. The controller receives feedback from the glucose meter and computes the future infusion rate using a model predictive control algorithm. It transfers the infusion rate to the infusion pump. Communication between the pump, controller, and the glucose meter takes place through the wireless channel.

the most commonly used control algorithms and models of glucose meter feedback for artificial pancreas.

Discrete Control Algorithm

An artificial pancreas control algorithm is mostly inactive and relies on the patient's own insulin management characteristics such as basal insulin rates and activities causing carbohydrate intake or burn. However, it continuously monitors or predicts the blood glucose level based on insulin injection and glucose intake or burn history, and takes action only when the blood glucose level reaches dangerous levels

such as hypo- or hyper- glycemia. The control algorithm has three active modes of operation: i) Brake, ii) Correction bolus, iii) Meal bolus supervision modes.

Brake Mode

In the brake mode the control algorithm first estimates the risk of hypo-glycemia of the patient, $R(t + \tau)$, at a future time $t + \tau$ from the present time t . Based on this risk parameter, the braking module attenuates the basal insulin infusion rate. Three risk conditions are defined in most artificial pancreas configuration: a) **no risk** stage, when the predicted blood glucose level at time $t + \tau$, $BG(t + \tau)$, is greater than a prescribed threshold BG_{thresh}^{pres} , b) **low risk** stage with slow attenuation of basal insulin rate, when $BG_{thresh}^{low} \leq BG(t + \tau) \leq BG_{thresh}^{pres}$, where BG_{thresh}^{low} is a lower bound on the glucose concentration and c) **high risk** stage with large attenuation of insulin infusion rate, when $BG(t + \tau) < BG_{thresh}^{low}$. The risk parameter for these three ranges is given by Equation 2.1:

$$\begin{aligned}
 R(t + \tau) &= 0, \text{ no risk} & (2.1) \\
 R(t + \tau) &= k_1(k_2(\ln(BG(t + \tau)))^{k_3} - k_4)^2, \text{ low risk} \\
 R(t + \tau) &= 100, \text{ high risk}
 \end{aligned}$$

where, k_1 , k_2 , k_3 , and k_4 are parameters of the artificial pancreas system. According to these risks the basal insulin delivery rate is attenuated using the Equation 2.2:

$$J_{atm}(t) = \frac{J_{basal}}{1 + kR(t + \tau)}, \quad (2.2)$$

where $J_{atm}(t)$ is the attenuated insulin delivery rate, J_{basal} is the basal infusion rate and k is a patient specific parameter. Note that the braking mode and the subsequent modes need a glucose concentration predictor, which will be discussed in later sections.

Correction Bolus Mode

Correction bolus mode is only active when four conditions are met: a) the braking mode is inactive, b) there has not been a meal bolus for the last 2 hours, c) it has been one hour since the last correction bolus, and d) the predicted blood glucose level one hour hence is greater than a high threshold value of BG_{thresh}^{high} . If all the above mentioned conditions are met, then a correction bolus as given by Equation 2.3 is administered.

$$J_{corr} = \frac{BG(t + 60|t) - BG_{thresh}^{high}}{2\theta}, \quad (2.3)$$

where θ is also a patient specific parameter and J_{corr} is the corrected infusion rate.

Meal Bolus Supervision Mode

The meal bolus supervision mode is activated by the patient at meal times. This bolus is to ensure that the patient's blood glucose level does not exceed dangerous levels during a meal intake. The bolus is computed by the control algorithm using Equation 2.4.

$$J_{bolus} = \frac{0.4\theta_{TDI}}{2MS}, \quad (2.4)$$

where J_{bolus} is the bolus infusion rate, θ_{TDI} is the total insulin intake required by the patient and MS is the meal size. The meal size is categorized into three levels: heavy ($MS = 2.5$), medium ($MS = 3$), and small ($MS = 5$) meals.

Blood Glucose Predictor Model

In each of the supervisory modes the control algorithm requires a predictor to estimate future blood glucose levels. The most commonly used predictor is a state space model [68] proposed by Chase. Several variants of this model exists in literature and are used extensively in proposed artificial pancreas model. The insulin diffusion process is represented using a three compartmental model. In this model,

it is assumed that the injected insulin diffuses into the blood through three different membranes. There are two gut compartments where glucose is stored and the insulin is used up in the resulting interaction. Further, the insulin also diffuses into a remote compartment due to the continuous flow of the blood. The insulin concentration and stored glucose volumes at the different compartments are represented as the state parameters of the model and a state vector $S(t)$ is formed as shown in Equation 2.5.

$$S(t) = [\delta BG(t), \delta X(t), \delta I_{sc1}(t), \delta I_{sc2}(t), \delta I_p(t), \delta BG_{sc}(t), \delta Q_1(t), \delta Q_2(t)]^T, \quad (2.5)$$

where $BG(t)$ is the blood glucose concentration in mg/dl , $X(t)$ is the remote compartment insulin action in min^{-1} , $I_{sc1}(t)$ and $I_{sc2}(t)$ are the interstitial insulin concentration in the first and second compartments, respectively in mU , $I_p(t)$ is the plasma insulin concentration in mU , $BG_{sc}(t)$ is the interstitial glucose concentration in mg/dl , and $Q_1(t)$ and $Q_2(t)$ are the stored glucose levels in the first two compartments in mg . δ indicates the difference from some reference values.

Given this state vector, a discrete time state predictor is employed, which expresses the current state of the insulin glucose interaction in terms of the state at a previous time, the injected insulin $u_{in}(t)$, and the meal intake of the patient $\beta(t)$. The discrete state predictor is implemented as a Kalman filter with the state space equation given in Equation 2.6.

$$S(t) = AS(t-1) + Bu_{in}(t-1) + E\beta(t-1), \quad (2.6)$$

where, A , B , and E are the parameters for the predictor and can be determined from insulin-glucose interaction profiles obtained using T1DM simulators [69]. In many of the submissions for artificial pancreas the daily meal intake by the patient $\beta(t)$ is modeled using a zero mean Gaussian white noise process. The first element in the

state vector $S(t)$ is the predicted blood glucose level, based on which the control algorithm decides on the future infusion rate.

The discrete control algorithm decides on an infusion rate at a chosen sampling period. The effect of that insulin infusion is measured by a glucose meter and is fed back to the control algorithm, which it uses to build the state vector history. But in a formal model this glucose meter has to be represented using some mathematical formulation. In this thesis, we discuss a well studied mathematical model of the infusion diffusion process that we use to represent the glucose meter in the closed loop operation of the infusion pump.

Let us assume that the infusion pump has a safety verified software. According to the infusion pump safety criteria as listed in the generic infusion pump project [62], the infusion rate should not exceed $p\%$ of a preset value. However, this safety criteria does not correspond directly to patient safety. Further, when the medical device operates in a control loop with the human body as a feedback the drug concentration in the blood can become unstable and reach unacceptably high values even if the infusion rate remains within $p\%$ of the preset value, hence hampering patient safety. Simulations on a Simulink model of the infusion pump confirms such patient safety violations. The parameters for the pump and the pharmacokinetic model were obtained from a case study on anesthesia infusion [55]. Figure 2.5(a) shows the infusion rate and the drug concentration in the blood over time for anesthesia infusion. From Figure 2.5(a) it can be seen that the infusion rate over time has significant overshoot and undershoot before it reaches a stable point. Further, significant amount of time, settling time, is required for the drug concentration in blood to reach the reference level set by the operator. A large settling time may lead to delay in the therapeutic effect of the drug. In case of mission critical operations such as drug infusion to prevent cardiac arrest [70], a short settling time

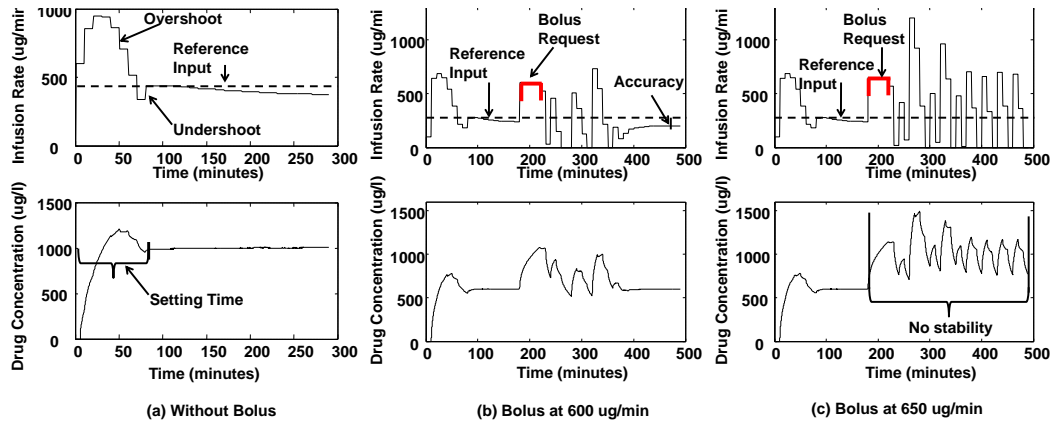


Figure 2.5: Anesthesia pump simulation with no bolus and at different levels of bolus showing overshoot, settling time, stable, and unstable behavior. Bolus administration marked by red lines.

is essential to provide therapeutic effects within a window of opportunity. Figures 2.5(b) and (c) show the case when bolus is requested by a patient. From figure 2.5(b) it can be seen that after bolus administration the pump dynamics becomes unstable. However, after some time the control actions brings the infusion rate and the drug concentration to a stable value. Figure 2.5(c) shows the same pump but now with a higher bolus rate (50 ul/min higher) causes the system to be unstable. Due to this unstable behavior the infusion rate can go above the limits and the safety criteria can be violated. Further, in case of an unstable operation the reference level of drug concentration is never reached. This causes over infusion of drugs affecting the accuracy of the pump to maintain the desired drug level. Thus, violation of the overshoot, undershoot, stability, settling time and accuracy properties can lead to patient safety hazards even though the software of the pump is certified safe.

2.4 Data Center Cooling Control

Contemporary HPC data centers use raised floors and lowered ceilings for cooling air circulation, with the computing equipment organized in rows of 42U racks arranged in an aisle-based layout, with alternating cold aisles and hot aisles (Fig-

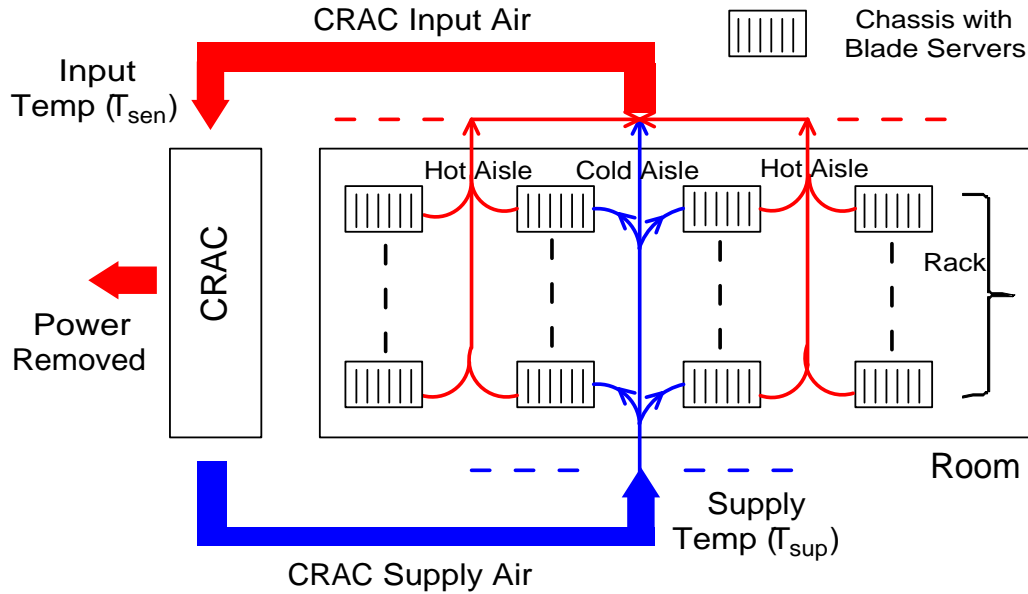


Figure 2.6: Heat transfer mechanisms in data center. The cool air from the CRAC is non-uniformly distributed among the servers. The heated air is recirculated throughout the data center and finally reaches the CRAC where it is again cooled.

ure 2.6). The computing equipment is usually in blade-server form, organized in 7U chassis. Often, in data centers, server racks are provided with chiller doors, which cool down the hot air coming out of the blade servers before it enters the data center room [71].

The cooling of the data center room is done by the CRAC, a.k.a the *heating and ventilation air conditioner* (HVAC). They supply cool air into the data center through the raised floor vents. The cool air flows through the chassis inlet and gets heated up by convection from the computing equipments and hot air comes out of the chassis outlet. The hot air goes to the inlet of the CRAC which cools it down. However, depending on the design of the data center, parts of the hot air may recirculate within the room affecting the thermal map at various positions including the inlet of the CRAC and the chassis.

Equipment Safety

The CRAC has to supply cold air so that the inlet temperature of each chassis does not exceed the redline temperature (T^{red}), which otherwise would lead to throttling of the operation of the computing unit—an undesirable phenomenon with respect to HPC job performance (throughput and turnaround time). For the safe operation of each chassis, the inlet temperature should be below the redline temperature. Thus,

$$\begin{aligned} & \text{chassis inlet temp} \leq \text{redline temp} \\ \implies & \left(\text{cold supply temp from CRAC} + \text{temp increase by recirculated heat} \right) \leq T_{red} \\ \implies & \mathbf{FT}^{sup}(t) + \mathbf{DP}(t) \leq T_{red}, \end{aligned} \quad (2.7)$$

where T^{red} is an n dimensional vector $\{T_i^{red}\}_n$, T_i^{red} is the redline temperature for chassis i , n is the total number of chassis in the data center, \mathbf{D} is an $n \times n$ matrix derived from the recirculation among the n chassis [72], and \mathbf{F} is an $n \times n$ diagonal matrix where each diagonal element, f_{ii} ($1 \leq i \leq n$), is derived from the amount of cold supply air going to the inlet of chassis i . Note that in absence of recirculation \mathbf{D} becomes a matrix populated with all zeroes [72] and \mathbf{F} becomes an identity matrix; thus the inlet at each chassis is same as the supply temperature. Table 2.3 lists the scalar symbols used.

Job and Machine Environment

The state-of-the-art in commercially available data center management software follows a conventional job queuing and issuing paradigm that focuses on optimizing *performance policy metrics*, those usually being *throughput* and *turn-around time*. The user front-end of a data center is the *submission* interface, i.e. the interface of the *scheduler*, which decides when and where (i.e. what servers) the jobs to be run at. A *job submission* usually provides: a) the executable, b) the input data, c) the

Table 2.3: Scalar Symbols and Definitions.

Symbol	Definition
n	total number of chassis
h	inter event interval or event period
c_k^{tot}	the number of servers (blades) job k requires
r_{ac}	thermal capacity of air flowing out of the CRAC per unit time
r_{room}	thermal capacity of air in the data center room
f_{ii}	cold supply air fraction going from CRAC to chassis i
d_{ij}	heat recirculation coefficient from chassis i to j
m	mode of operation of CRAC, $m \in \{high, low\}$
t_{sw}	time taken by the CRAC to switch from one mode to the other
ω	idle chassis power consumption
α	power consumption of a chassis per unit of utilization
u	chassis utilization
N_h	total number of jobs in event period h
$T^{sup}(t)$	air temperature as supplied from the CRAC at time t
$T^{sen}(t)$	air temperature at the input of the CRAC at time t
T^{red}	manufacturer's redline inlet temperature
T_{high}^{th}	high thermostat setting for the CRAC
T_{low}^{th}	low thermostat setting for the CRAC
$(T_{high}^{th})^{max}$	upper bound on high thermostat setting for CRAC
$(T_{high}^{th})_i$	CRAC high thermostat setting requirement for server i
ΔT^{th}	temperature difference between the CRAC <i>high</i> and <i>low</i> thermostat settings
P_{ex}^m	power extracted by the CRAC in mode m
P_j^{full}	power dissipation of chassis j at 100 % percent utilization. For any variable z , z^{full} denotes the value of z at 100 % utilization and z^{empty} denotes that at empty data center.
P_h^{comp}	total computing power at inter-event period h
$P^{AC}(t)$	power consumption of CRAC at time t
E_y	energy consumption for algorithm y
E_y^m	energy consumption of CRAC in mode m for algorithm y

number of servers it requires and the *estimated runtime*, and d) other constraints such as a priority, and specific *computing node* preferences. A *computing node* is a chassis containing multiple blade servers. The job run-times are normally overestimated by the users [73]. We consider user estimated job turnaround times as the jobs' **deadlines**. The scheduler aborts the jobs that do not complete by the deadline. Thus, a scheduling algorithm has to ensure meeting of the job deadlines to avoid job abortion. There are two types of decision making for job scheduling: i) *temporal* (i.e. when to start the execution of the jobs), which directly impacts the job throughput

and turnaround times; and ii) *spatial* (i.e. where to execute the jobs), which can also impact the job throughput and turnaround times if jobs are assigned to servers with low computing capabilities (e.g., processor speed). *To ensure no degradation in throughput and turnaround time, this thesis focuses only on energy-aware spatial scheduling decision making among the servers requested by the users.*

Further, an event based decision making for job scheduling is considered. An *event* comprises of arrival of new jobs (*job arrival*), beginning of job execution (*job start*) and end of job execution (*job completion*). *Inter event interval*, also referred to as **event period** (denoted by the symbol h), is the time between two consecutive job start and completion events. Computing power in a data center changes when a job starts or ends execution on a machine. Therefore, the computing power in any inter-event interval is constant over time. The following sections will describe the behavior of the CRAC unit in the data center and the inter-dependency of the cooling behavior with the computing power consumption.

CRAC Behavior

The CRAC can have many different modes of operation. For simplicity, in this thesis, we assume two operational modes viz. *high* and *low* modes. During its operation the CRAC oscillates between the *high* and *low* modes. In each mode, the CRAC extracts a constant amount of power P_{ex}^{high} and P_{ex}^{low} , respectively. Figure 2.7 shows the variations in the CRAC input (i.e. ceiling temperature) and output (i.e. supply temperature) over time from temperature sensor measurements in the ASU HPC data centers. The difference between these two temperatures (that determines the power extracted by the CRAC unit) clearly shows two distinct values indicating two operational modes.

A *mode epoch* is the time duration for which the CRAC operates in a particular mode. In a CRAC mode, the input temperature linearly varies with time. The line

gradient depends on: i) the difference of power generated in the data center, P_h^{comp} , and power extracted by the CRAC [74]; and ii) the thermal capacity of air in the data center room. Since the extracted power is different for different CRAC modes, the temperature rise depends on which mode the CRAC is in. The temperature of the supply air from the CRAC linearly varies with the input temperature based on the power extracted by the CRAC.

CRAC Power Consumption

The CRAC power consumption depends on the CRAC power mode (i.e. the power extracted by the CRAC) and Coefficient Of Performance (COP) of the CRAC. The COP of the CRAC to supply air at temperature $T^{sup}(t)$ at an instance t is normally given by $COP(T^{sup}(t)) = \frac{T^{sup}(t)}{T^{sen}(t) - T^{sup}(t)}$, where $T^{sen}(t)$ is the CRAC input temperature (Figure 2.6) at the instance t [75]. The above assumption on the COP of the CRAC unit enables the computation of the energy dissipated by a cooling unit in the operating mode m . The COP is given by $\frac{r_{ac} T^{sup}(t)}{P_{ex}^m}$. The power consumption to run the CRAC at time t is given by:

$$P^{AC}(t) = \frac{P_{ex}^m}{COP(T^{sup}(t))} = \frac{(P_{ex}^m)^2}{r_{ac} T^{sup}(t)}. \quad (2.8)$$

Any technique to reduce the CRAC power consumption has to operate in lower modes, reducing the P_{ex}^m , and increase the $T^{sup}(t)$ as far as possible.

Inter-dependency of Cooling and Job Management

As the data center utilization increases, power consumption at the chassis increases; requiring lower supply temperature to meet the redline (Equation 2.7) [73]. The supply temperature is maximum for 0% utilization and minimum for 100% utilization. Generally, however, if there is heat recirculation, the heat input to the chassis increases; thus requiring lower $T^{sup}(t)$ to keep the temperature within the redline temperature [73]. Therefore, it is important to predict the maximum supply tem-

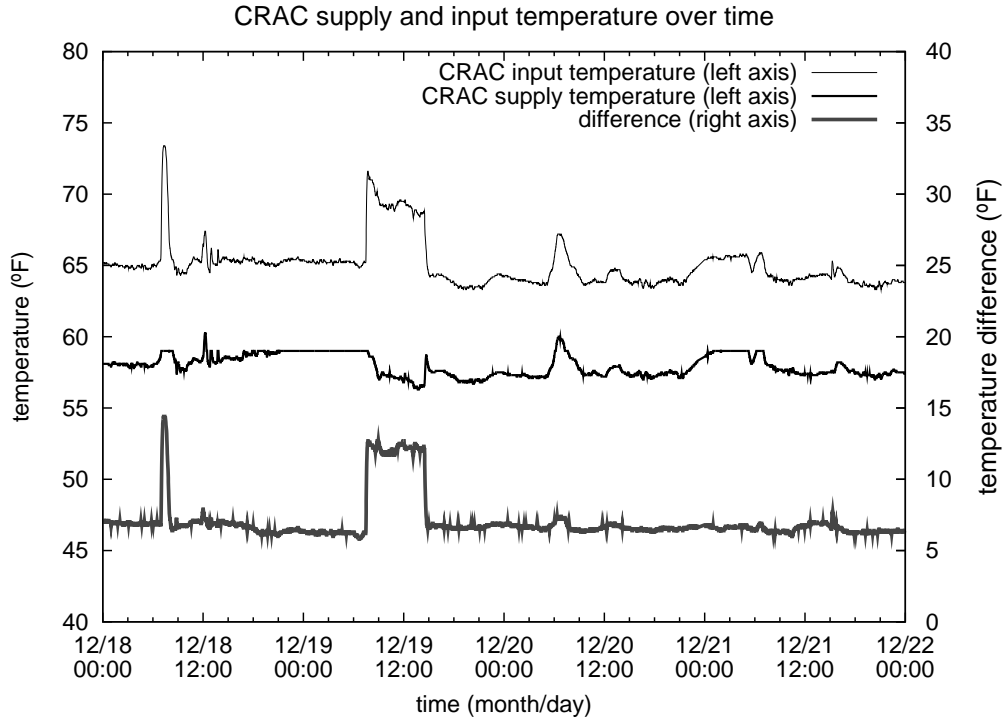


Figure 2.7: Variations in the CRAC input and output temperature based on actual sensor measurements in the ASU HPC data center. The difference in these temperatures indicates the two operational modes of the CRAC.

perature from the CRAC. In a particular CRAC mode, the supply temperature also changes linearly at the same rate as the CRAC input temperature. It should be noted that the maximum temperature can be reached when the power extraction from the CRAC is low, i.e. when it is operating in the low mode.

The CRAC switches mode when the CRAC input temperature reaches the thermostat set temperatures. When the input temperature goes below a low thermostat set temperature, T_{low}^{th} , the CRAC mode is changed from the *high* to *low*. Similarly, when $T^{sen}(t)$ reaches the high thermostat set temperature T_{high}^{th} , the CRAC mode is changed from *low* to *high*. The CRAC does not change modes instantaneously. After the $T^{sen}(t)$ crosses a set temperature the CRAC takes t_{sw} amount of time to change mode. The maximum input temperature to the CRAC and hence the

maximum supply temperature, T_{max}^{sup} , is therefore dependent on the high thermostat set temperatures and any temperature increase during the switching time, t_{sw} , as follows:

$$\begin{aligned}
& \text{max supply temp} = \text{max CRAC input temp} - \text{temp reduction due to CRAC cooling in low mode} \\
\Rightarrow T_{max}^{sup} &= \text{max CRAC input temp} - P_{ex}^{low} / r_{ac} \\
\Rightarrow T_{max}^{sup} &= \left(\text{high thermostat temp} + \text{input temp increase at low CRAC mode for } t_{sw} \right) - P_{ex}^{low} / r_{ac} \\
\Rightarrow T_{max}^{sup} &= T_{high}^{th} + \frac{P_h^{comp} - P_{ex}^{low}}{r_{room}} t_{sw} - \frac{P_{ex}^{low}}{r_{ac}}, \tag{2.9}
\end{aligned}$$

where r_{ac} is the thermal capacity of the air flowing out of the CRAC per unit time. It can be concluded from Equation 2.9 that the maximum supply temperature from the CRAC depends on the high thermostat settings and the computing power in the data center. Hence, job management (i.e. scheduling and placement) and server power management, both of which determine the computing power consumption, in conjunction with the CRAC management (i.e. dynamically updating the thermostat settings) need to be performed in such a way that for the maximum supply temperature, the redline temperature is not violated (Equation 2.7).

A programmable thermostat is typically available in data centers where the set temperatures can be dynamically changed. However, the CRAC maintains a constant difference, ΔT^{th} , between the *high* and *low* thermostat settings¹ (i.e. $\Delta T^{th} = T_{high}^{th} - T_{low}^{th}$). Depending on this difference, the minimum possible supply temperature, T_{min}^{sup} , from the CRAC can be determined (in the same way as T_{min}^{sup} in Equation 2.9) when the input temperature reaches the low thermostat set point and the CRAC operates at a high mode (i.e. the power extraction by the CRAC is higher):

¹In the rest of the thesis, changing the high thermostat set temperature refers to changing both the set temperatures.

$$\begin{aligned}
T_{min}^{sup} &= T_{low}^{th} + \frac{P_h^{comp} - P_{ex}^{high}}{r_{room}} t_{sw} - \frac{P_{ex}^{high}}{r_{ac}} \\
\Rightarrow T_{min}^{sup} &= T_{high}^{th} - \Delta T^{th} + \frac{P_h^{comp} - P_{ex}^{high}}{r_{room}} t_{sw} - \frac{P_{ex}^{high}}{r_{ac}}.
\end{aligned} \tag{2.10}$$

Equation 2.10 shows how an increase in the high thermostat set temperature can increase the supply temperature; thus potentially reducing the CRAC power consumption (from Equation 2.8). The design of cooling-aware spatial job scheduling algorithm, HTS, allows higher thermostat set temperatures.

HTS Algorithm Design

This is a cyber-physically oriented energy management algorithm developed in this thesis. The principal intuition behind the algorithm is to: i) *statically rank the servers* from best to worst (according to the potential load, i.e. the required thermostat setting, incurred on the CRAC); ii) *place (i.e. assign)* temporally scheduled jobs to the best ranked servers; and iii) dynamically set the thermostat to the required value. Figure 2.8 presents the intuitive operational flow showing the aforementioned four operations and their inter-dependencies. The following subsections describe these three operations.

Server Ranking

The ranking of the servers is necessary for the job placement to assign jobs based on the server ranks. To counter the energy inefficiencies because of the dependency on the CRAC thermostat setting, the HTS algorithm ranks the servers based on their requirement on CRAC thermostat to keep the inlet temperature within the redline temperature. The servers are ranked from highest to lowest thermostat temperature requirement. The jobs are then placed according to the server ranking; thus allowing the CRAC thermostat setting to be increased.

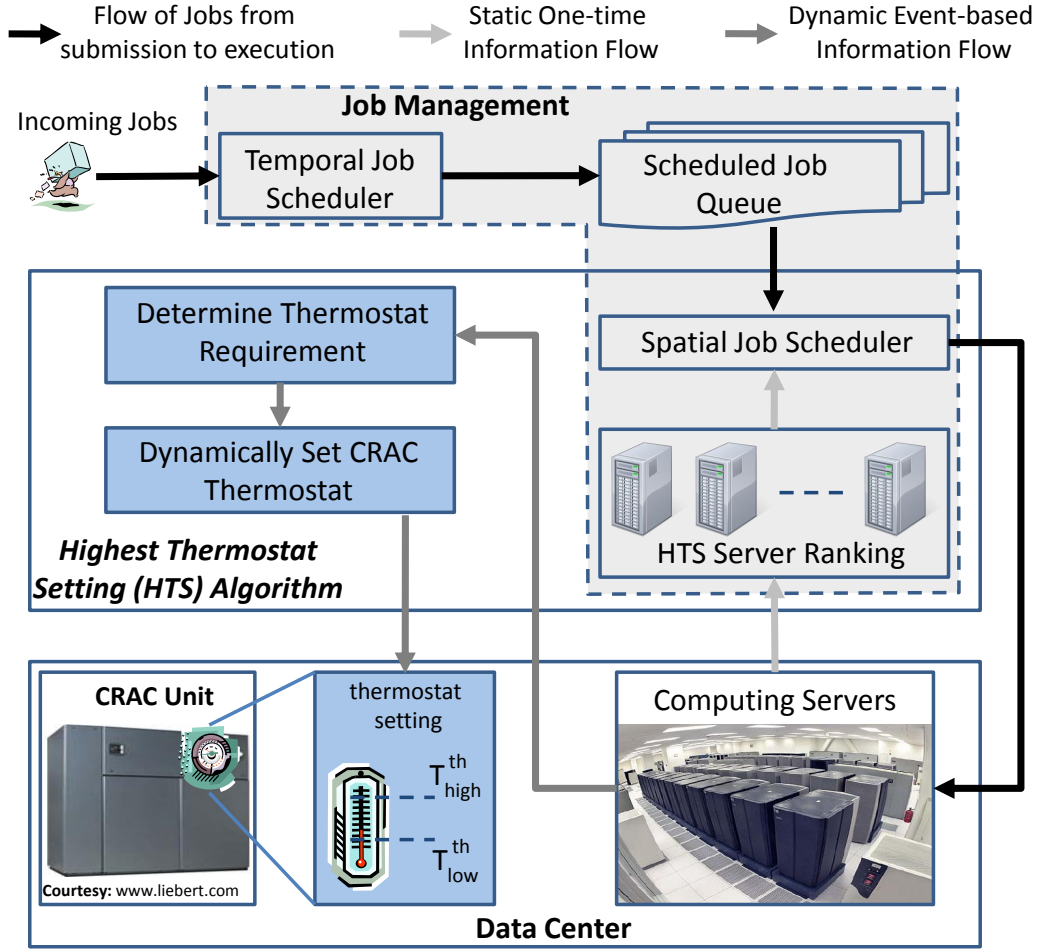


Figure 2.8: Architecture and work-flow of HTS. Each server is assigned a high CRAC threshold temperature that it requires to avoid redlining. The workload manager prefers servers with high CRAC thresholds and sets the CRAC threshold temperature to the lowest high threshold temperature among the chosen servers.

The power consumption of the servers impact the upper bound on the CRAC thermostat settings (Equation 2.12). Finding the optimal placement is NP-complete and may require hours of operation [73]. As such HTS performs a static ranking of the servers based on full utilization of the data center, which yields the thermostat setting requirement for a server i as follows:

$$(T_{high}^i)^{th} = \frac{T^{red} - \sum_j d_{ij} P_j^{full}}{\sum_j f_{ij}} + \frac{P_{ex}^{low}}{r_{ac}} - \frac{(P_h^{comp})^{full} - P_{ex}^{low}}{r_{room}} t_{sw}, \quad (2.11)$$

where P_j^{full} is the power consumption of chassis j at 100% utilization. The servers

are then statically ranked in the decreasing order of $(T_{high}^{th})_i$. The server ranking, is an one-time initialization process (performed in procedure *Initialization* in Algorithm 2.1). The ranks are represented by ranking vector \mathbf{R} .

Job Placement

As shown in Figure 2.8, the Spatial Job Scheduler takes the jobs from the scheduled job queue² and places them to the servers based on their ranks (procedure *HTS* in Algorithm 2.1). Such rank-based job placement can be easily incorporated in the current job management softwares. For example, the widely used Moab job management software allows setting up server priorities in the software's configuration [76]. The server ranking presented in the previous section can be used to prioritize the servers in Moab. The priority-based job assignment can then be enabled for job placement.

Dynamic Thermostat Setting

After placing the jobs, EDF-HTS sets the thermostat setting to the highest possible value given by Equation 2.12. As shown in Figure 2.8, first the required thermostat set temperature is determined followed by actually setting the thermostat to the required value. Equation 2.12 is used to determine the required thermostat setting after the job placement is performed. Unlike the server ranking in Section 2.4 (where the data center was assumed to be fully utilized), the thermostat requirement is computed based on the actual server utilization after job placement.

With the CPS perspective in mind let us now focus on the paradigm of model based engineering to better understand how such a technique can be useful for CPS safety analysis and verification.

²Here the assumption is that the jobs are already temporally scheduled, i.e. the jobs' start times are decided by some temporal scheduling algorithm.

Algorithm 2.1: HTS integrated in the spatio-temporal job scheduling

procedure INITIALIZATION()

Group nodes with respect to power specifications.
Sort groups with respect to computing efficiency
(i.e. MIPS/watt).
Perform server rankings, \mathbf{R} , according to the
requirement of thermostat set temperature to
meet the redline for 100% utilization (Equation 2.11).

end procedure

procedure HTS()

Place job to available node(s) with the lowest rank in \mathbf{R} .
Determine the power distribution vector, \mathbf{P}_h .
Set the CRAC thermostat using SETTHERMOSTAT(\mathbf{P}_h).

end procedure

procedure SETTHERMOSTAT(\mathbf{P}_h)

Set high thermostat setting (T_{th}^{high}) as

$$F^{-1}T_{red} - \left[\frac{P_h^{comp} - P_{ex}^{low}}{r_{room}} t_{sw} - \frac{P_{ex}^{low}}{r_{ac}} \right] - F^{-1}D\mathbf{P}_h \quad (2.12)$$

end procedure

procedure UPONJOBCOMPLETION()

Dispatch the next job in this group's queue using HTS ().
Determine the power distribution vector, \mathbf{P}_h .
Remove the job from the queue.
Set the CRAC thermostat using SETTHERMOSTAT(\mathbf{P}_h).

end procedure

procedure UPONJOBARRIVAL()

if job comes with node restrictions **then**

Insert the job in the queue of the specified node group based on a scheduling policy (e.g. FCFS or EDF).

else

Insert the job in the most energy-efficient group queue based on a scheduling policy (e.g. FCFS or EDF).

end if

for each node group, from the most to least efficient node **do**

if job's finish estimation > deadline **then**

(1) Insert the job in an "opening" having enough free servers for enough time. Continue with next job.

(2) If Step 1 fails, push-fit the job at an earlier "time" if shifting jobs still make the deadline.

Continue with next job.

(3) If Step 2 fails, add the job to next group's queue.

end if

if required nodes in this group are idle **then**

Dispatch the job in this group's queue using HTS ().

Remove the job from the queue.

end if

end for

end procedure

BACKGROUND ON MODEL BASED ENGINEERING

This chapter gives a brief background of model based engineering and the different types of models and analysis that can be carried out on models with examples. An overview of the model based engineering methodology is provided in Section 1.6 of Chapter 1. MBE has three phases - a) modeling, b) analysis, and c) synthesis phase and we will provide a background on each of the phases in the following sections.

3.1 Modeling Phase

The basic component of MBE is a model. However, before defining a model let us first consider the definition of a system for which we are building the model.

Definition 2 System: *A system is a collection of interacting components each having a given behavior and the system exhibits an aggregate behavior expressed as a*

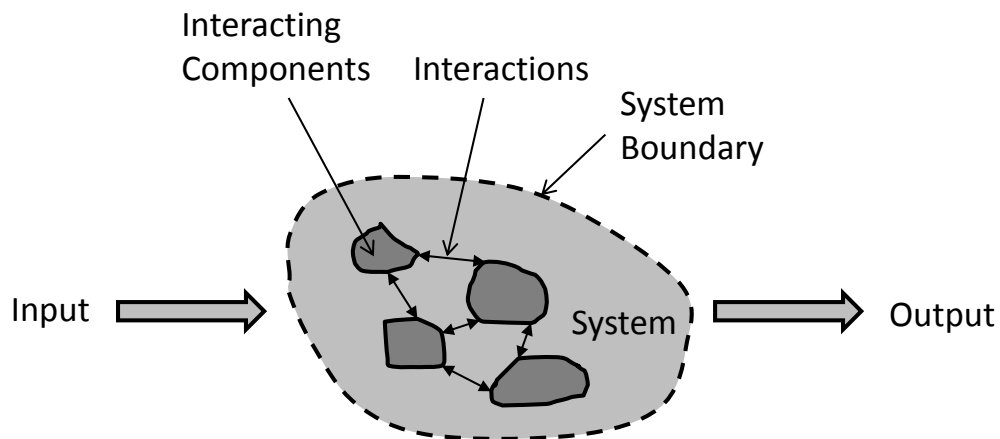


Figure 3.1: System is a combination of interacting components with inputs and outputs. Any parameter affecting the operation of the system and is outside the system boundary is an input. Any parameter observable from outside the system boundary is an output.

combination of the individual component behavior. A system has a system boundary. Any variable with a source outside the boundary affecting the system behavior is an input to the system. Any observable property of the system is the output of the system.

Figure 3.1 shows a system with interacting components. A system boundary is used to identify the components within the system and the inputs and outputs. Any variation with source outside the system boundary affecting components of the system is denoted as input. Any observable system property outside the system boundary is called output. Given this definition we consider the definition of a model.

Definition 3 Model: *A model is a simplified representation of a real system.*

This definition includes a large number of possibilities that model can mean. A model thus can mean a mathematical representation using equations, or a physical model, which can be a mock up of the real system using off-the-shelf components, or a visual model, including drawings and plans, or textual models, such as English text specification. However, any representation of a real system is not useful unless certain properties or requirements are satisfied. These requirements are related to how close the model is to the real system and whether the model satisfies constraints set by the designer. If a model is close to the real system to the satisfaction of the designer then the model is called a **validated** model. If the model satisfies constraints set by the designer then the model is called a **verified** model. A validated and verified model, i.e. a model built to the satisfaction of the requirements, represents the real system and shows a conceptual implementation that can satisfy constraints. The requirements on the model that validates it are called **functional requirements**, while the constraints that a verified model satisfies are called **non-functional requirements**.

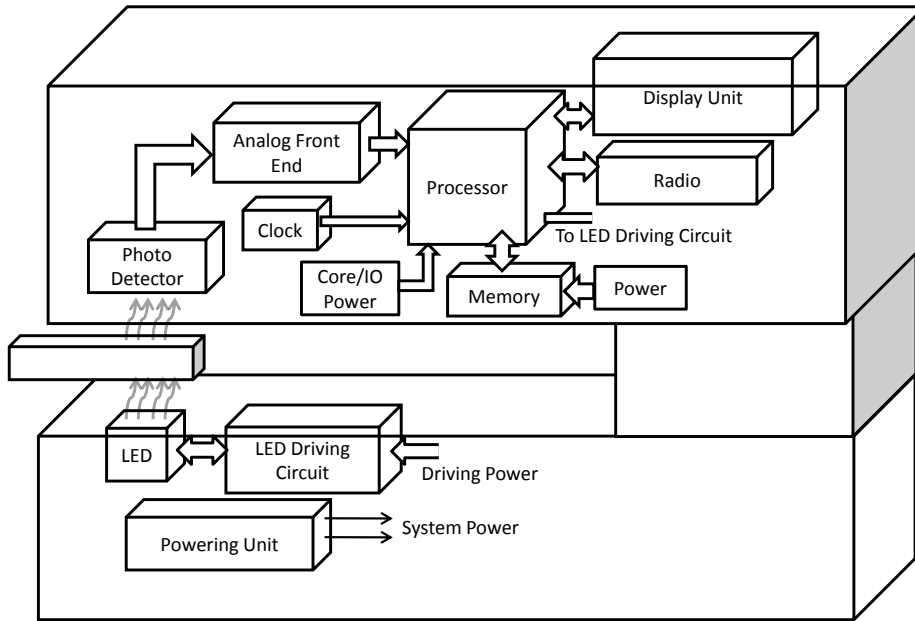


Figure 3.2: Visual model of Pulse Oximeter with components, subcomponents, and connections indicating flow of data in between them.

Example 3 Model of a pulse oximeter: *A pulse oximeter is a medical device that is installed on the index finger or on the ear lobe of a person, and it measures the blood oxygen level and pulse rate by passing light through the ear lobe or index finger. A visual model of the pulse-oximeter is shown in the Figure 3.2. The heat dissipation in the pulse-oximeter due to transfer of light through the human body can be modeled using a mathematical equation, Penne's bioheat equation [4]. If the temperature rise predicted by the model matches that from the real system then the model is validated. If the temperature rise predicted by the model is within limits such that it won't burn the skin, a safety requirement, then it is verified for safety.□□*

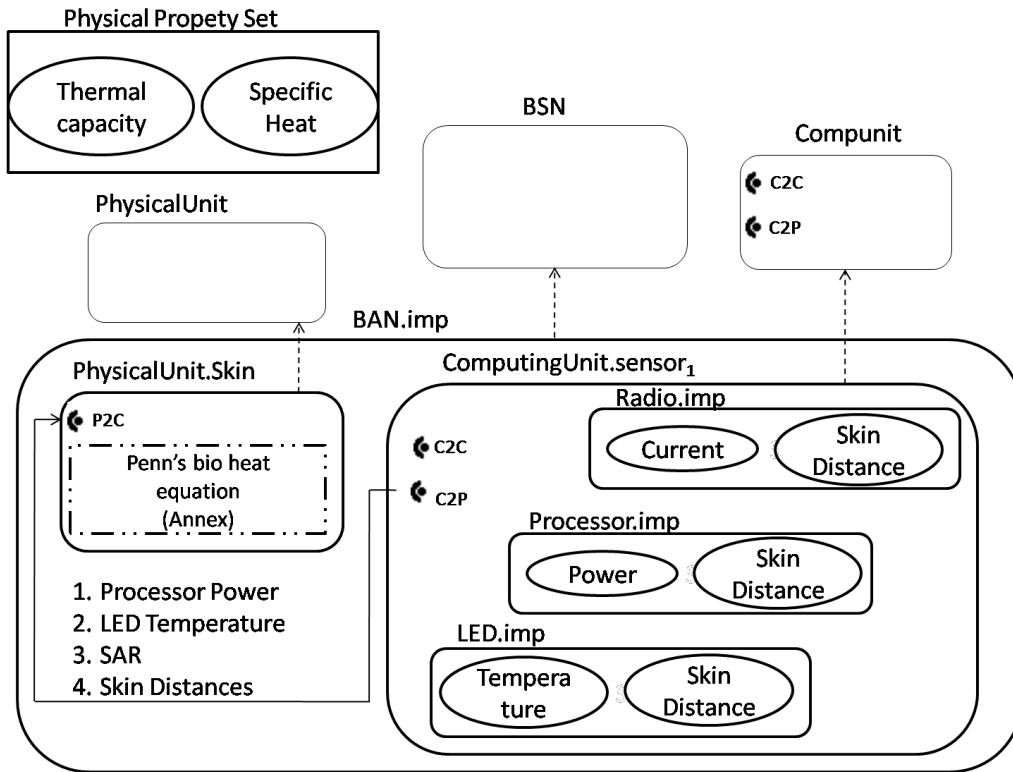


Figure 3.3: Specification of the architectural model of Pulse Oximeter in Architectural Analysis and Design Language (AADL).

Models can be classified into two basic types - architectural models and behavioral models.

Architectural model

Architectural model is a representation of a system using components and subcomponents and connections between them. The connections represent subcomponent relationship and data flow between components. Unified modeling language (UML) and system modeling language (SysML) are means for architectural modeling of systems. An architectural modeling language extensively used in this thesis is the Architecture Analysis and Design Language (AADL) and Example 4 shows one such case.

Example 4 Pulse oximeter model: Figure 3.3 shows the AADL model of the pulse oximeter. The AADL model shows that a pulse oximeter has a computing unit, which is attached to the human body part (skin). Within the computing unit component there is a radio, processor, and LED subcomponents. There is also a subcomponent for the human skin which is connected to the computing component. The connection represents heat transfer from the computing unit to the human skin. □□

Architectural models thus represent the what components the system is made of. They do not explain how the system behaves.

Behavioral model

The system behavior is described by a behavioral model. A behavioral model is a representation of the processes inside a system that causes observable outputs for given inputs. Often it is a mapping of the input to an observable output through mathematical equations, computational models, or transfer functions. There are several types of behavioral models as discussed below -

Mathematical equations

Mathematical equations represent the output of a system as a function of its inputs.

Example 5 Bouncing ball: Consider the system with a ball and a floor. The system consists of the ball and gravity, while the floor is outside the system boundary. The ball is bouncing on the floor. The behavior of bouncing is represented by kinematic equations -

$$\begin{aligned} \ddot{h} &= -g, \text{ if } \dot{h} \geq 0 \\ \dot{h} &= -\gamma\dot{h}, \text{ at } h = 0, \end{aligned} \tag{3.1}$$

where, h is the height of the ball from the ground and is an observable parameter and hence is the output of the system, g is the acceleration due to gravity and is a

system property, and γ is a damping factor and is a property of the floor and hence is an input to the system.□□

Mathematical models thus express the system behavior with respect to an independent variable for different inputs. Mathematical models of systems can become abstract and may seem removed from the real system due to several assumptions. Hence, it is of utmost importance to validate a mathematical model such that it corresponds to the real system.

Transfer function model

A transfer function model views the system as a black box and expressed the output as a function of the input. The transfer function model abstracts the system behavior in a single reversible function. Thus it is very easy to obtain the output for a given input and also the input for an observed output by just inverting the transfer function. Thus instead of representing the system behavior with respect to an independent variable the transfer function expresses the nature of system behavior for different inputs. Thus for a given input the transfer function has to be solved or transformed to obtain variation with respect to independent variables such as time or space.

Example 6 R-C circuit: Consider an R-C circuit as shown in Figure 3.4 where the input is a voltage source and the output is the voltage across the resistor. The transfer function is obtained by applying Kirchoff's law in the Laplace domain as given in Equation 3.2.

$$V_{out}(s) = \frac{V_{in}R}{R + \frac{1}{sC}} \quad (3.2)$$

Here s is the laplace transform parameter and an inverse transform is needed for a given value of the input voltage V_{in} to obtain the time variation of the output voltage.□□

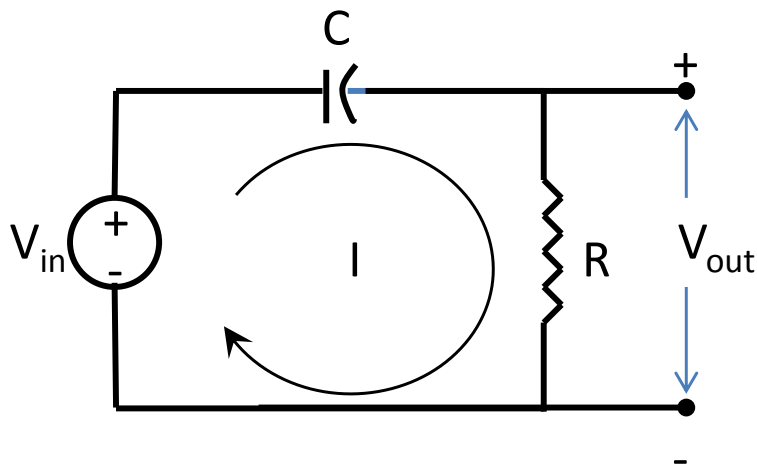


Figure 3.4: R-C Circuit system where the voltage V_{in} is the input and voltage across the resistor V_{out} is the output.

Computational model

Computational models express the behavior of a system using a finite number of well defined steps. Possible computational models are algorithms, or constructs having a well defined execution method such as Turing machines. The usefulness of such models is that the behavior of the system can be classified into types and theorems on the properties of the model can be proven.

Algorithmic model: An algorithmic model is a representation of the behavior of a system as an algorithm or a finite number of well defined steps.

Example 7 Consider a binary search engine that searches a given element from a sorted array. The behavior of this search engine can be expressed in the form of an algorithm as shown in Algorithm 3.1 Note that the algorithm finished in a finite number of steps when *CurrentIndex* becomes 0. None of the steps are non-deterministic and do not have ambiguous implications. □

Algorithm 3.1: Index = BinSearch(Sorted Array A, Number N)

```
1: CurrentIndex = length of A/2;
2: CurrentArray = A;
3: if CurrentIndex == 0 then
4:   Number N not found
5: end if
6: if (CurrentArray(CurrentIndex) = N) then
7:   Index = N;
8:   return(Index);
9: else if (CurrentArray(CurrentIndex) > N) then
10:  Index = BinSearch(A(0 - CurrentIndex - 1),N);
11: else
12:  Index = BinSearch(A(CurrentIndex + 1, length of A),N);
13: end if
14: return(Index)
```

Algorithmic models can be used to simulate a system using software and can also be used to prove theorems on the response time of the system.

Finite state automata: A finite state automata is a computational construct with a definite execution process and can be used to represent discrete time behavior of a system with random events causing changes in the behavior of the system.

Definition 4 A finite state automata can be defined as a tuple $S = \{X, X_0, U, Tr, Y, H\}$ where,

1. X is a set of states or as we will call them in this thesis, modes,
2. $X_0 \subseteq X$ is a set of initial modes,
3. U is a set of inputs,
4. $Tr \subseteq X \times U \times X$ is a transition relation such that if $(x_0, u, x_1) \in Tr$ then given a mode $x_0 \in X$ on input $u \in U$ the FSM S goes to mode $x_1 \in X$.
5. Y is a set of possible outputs
6. $H : X \rightarrow Y$ is an output mapping of a mode into a possible output $y_0 \in Y$

Example 8 Password validity checker: Consider a system that verifies whether the password entered by an user is a valid one or not according to the rule that there has to be atleast one integer in the password. The finite state automata representation of this system can have two modes $X = \{x_1, x_2\}$. The initial mode $X_0 = x_1$. The set of inputs can be $U = \{(a, b, c, \dots, z, A, B, C, \dots, -Z, 0, 1, 2, 3, \dots, 9)\}$. The transition set can be $Tr = \{(x_1, (a, b, c, \dots, z, A, B, C, \dots, -Z), x_1), (x_1, (0, 1, 2, \dots, 9), x_2), (x_2, U, x_2)\}$. The set of outputs $Y = \{Valid, Invalid\}$ and the mapping H is such that $H(x_1) = Invalid$ and $H(x_2) = Valid$. Thus this FSM describes the operation of the password validity checker and it will only consider a password valid if it has atleast one integer in it. □

There are several types of FSM. If the output map of the FSM has only *yes* and *no* values then the FSM is called a Deterministic Finite Automata (DFA). If a DFA is allowed to have a stack as a memory then it is called a Push Down Automata (PDA). If the DFA now has an infinite tape as memory where it can write and can read at any possible location without erasing any other location on the tape then it is called a Turing machine. A Turing machine is the model of our current computing systems.

Hybrid Automata: A hybrid automata model is a mix of a finite state automata and a mathematical equation model. Such models are useful for representing systems which have an underlying dynamic behavior which causes changes in modes. Further, with mode changes the nature of dynamic behavior also changes.

Definition 5 A hybrid automata is a tuple $HA = \{\mathbb{L}, \mathbb{L}_\varphi, Inv, F\}$, where

1. \mathbb{L} is a set of modes of the HA. A state of an HA is an element of the state space $\mathbb{L} \times \mathbb{R}^n$, where n is the number of continuous variables in the system.
2. \mathbb{L}_φ is a set of initial modes.
3. $Inv : \mathbb{L} \rightarrow \mathbb{R}^n$ is a mapping from the set of locations \mathbb{L} to a subset in real space. The invariant set for a location has the following properties -
 - $Inv(l_i)$ for a given mode l_i is a compact set.
 - for any two locations l_i and l_j , $Inv(l_i)^\circ \cap Inv(l_j)^\circ = \phi$, where $Inv(l_i)^\circ$ is the interior of the set $Inv(l_i)$, which does not include the boundary $\delta Inv(l_i)$ of the set and $Inv(l_i)^\circ \cup \delta Inv(l_i) = Inv(l_i)$.
 - $\bigcup_{l_i \in \mathbb{L}} Inv(l_i) = \mathbb{R}^n$
4. $F : \mathbb{L} \rightarrow \mathbb{R}^n$, is a mapping from \mathbb{L} to a set of functions on the compact set \mathbb{R}^n .

Thus for each mode there is a function governing the time variation of each of the n continuous variables.

Example 9 Cooling control system: Consider a cooling control system (CRAC), which receives hot air as input and provides cold air as output. The CRAC operates in two modes $\{m_1, m_2\}$ and in each mode extracts a constant amount of power $P_{ac}(m_i)$ from the incoming air at temperature T_{in} . The outgoing cold air at temperature T_{out} of the CRAC gets heated by heat sources in a room generating power P_{gen} . The CRAC changes mode based on a threshold on the inlet temperature T_{thresh} . Whenever the inlet temperature becomes higher than a high threshold value, the CRAC switches from mode m_1 to a mode m_2 , and when the CRAC inlet temperature

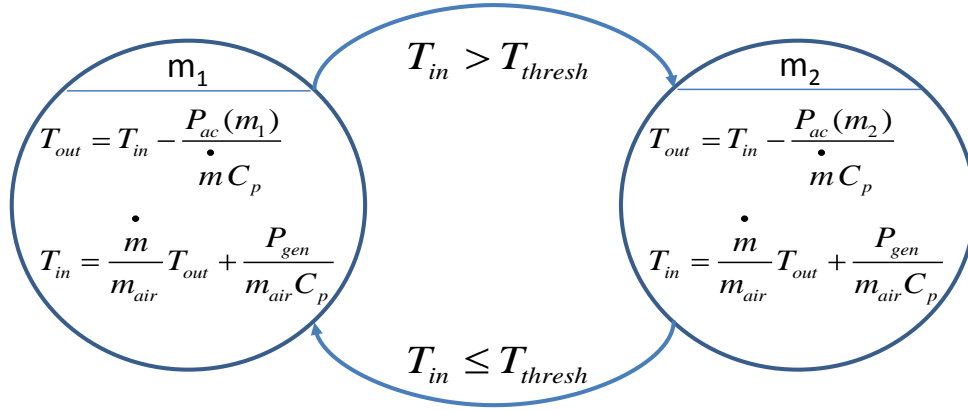


Figure 3.5: Hybrid Automata model of a cooling control system with two states each having different dynamics and state transitions governed by constraints on the inlet temperature.

becomes lower than the high threshold value, it changes back to mode m_1 . The dynamic equations governing the inlet and outlet temperatures of the CRAC at a given mode m_i are given by Equation 3.3.

$$T_{out} = T_{in} - \frac{P_{ac}(m_i)}{\dot{m}C_p} \quad (3.3)$$

$$\dot{T}_{in} = \frac{\dot{m}}{m_{air}}T_{out} + \frac{P_{gen}}{m_{air}C_p},$$

where, \dot{m} is the mass flow rate from the CRAC, m_{air} is the mass of air inside the room, and C_p is the specific heat of air. Given this CRAC system its hybrid model is as shown in the Figure 3.5. The hybrid model of CRAC consists of two modes $\mathbb{L} = \{m_1, m_2\}$ and the initial mode can be m_1 . The invariant set can consist of $Inv = \{Inv(m_1), Inv(m_2)\}$, where $Inv(m_1) = [0, T_{thresh}]$ and $Inv(m_2) = [T_{thresh}, \infty]$. The F function can be the equations 3.3 where for each mode the power drawn by the CRAC $P_{ac}(m_i)$ changes. \square

There are several variants of hybrid automata two of which is most commonly used - timed automata and a linear hybrid automata. A timed automata assumes that

the dynamic equations can only be of the form $\dot{i} = constant$, and hence there is a notion of clock. A linear hybrid automata assumes that the dynamic equations can only be of the form of linear first order differential equations. Given this overview on modeling techniques we now discuss how these models can be used for analyzing a system in the next section.


3.2 Analysis Phase

Given any model analysis is performed for two objectives - a) validation, to prove the closeness to the real system that the model represents, and b) verification, to prove that the model satisfies constraints set forth for the system. Validation is the first objective with which a model is analyzed and once closeness to the real system is proven then verification for satisfaction of non-functional requirements are performed.

Given the objectives, analysis of models can be broadly classified into two types - a) simulation analysis, and b) reachability analysis.

Simulation Analysis

Simulation analysis considers a given set of configurations of a system and a time bound. For each configuration the simulation analysis executes the system for time t and outputs the values of the observable outputs.

Example 10 *Consider the bouncing ball model in Example 5. The set of configurations can include configurations with different values of the damping factor γ . The equation 3.1 is solved for different values of γ for a time period of 100 seconds and the results are shown in the Figure 3.6. *

Thus a simulation analysis can be done on a limited number of case studies for a limited amount of time. With simulation analysis no guarantees on the system

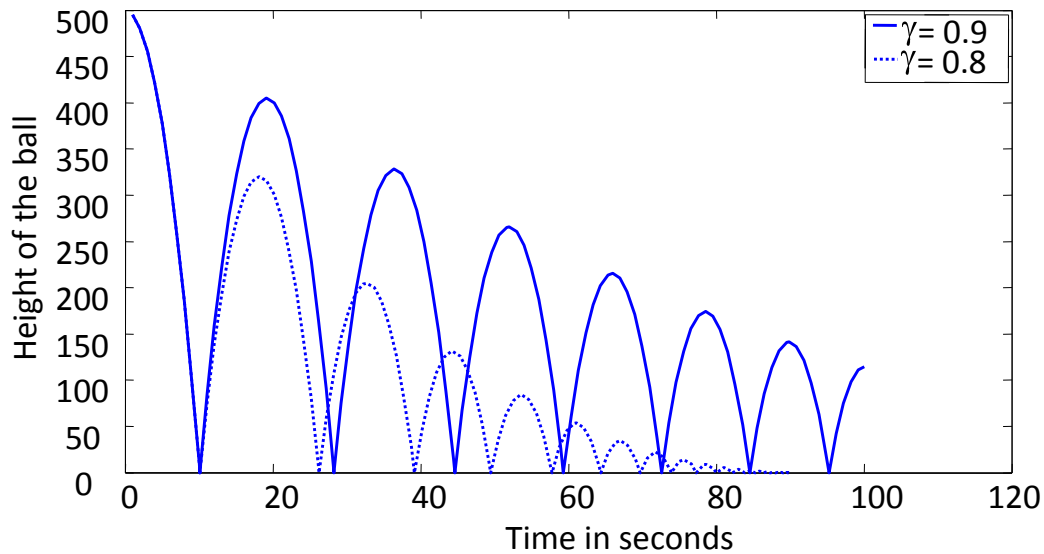


Figure 3.6: Simulation of a bouncing ball showing the height of the ball with respect to time for different values of the coefficient of restitution parameter, γ .

behavior can be established but for a given time and a given configuration actual values of the system parameters can be estimated.

Reachability Analysis

A reachability analysis takes a neighborhood of a given configuration of the system and outputs an over estimation of all possible values that the system parameters can take. The set of all possible values that the system parameters can take is called the reach set. The reach set computation is intensive and requires closed form equations representing the system behavior. As an example we take the hybrid automata model of the CRAC and perform reachability analysis on it using an open source reachability analysis tool Phaver [77].

Example 11 Consider the hybrid automata in Example 9. We will perform the reachability analysis of this model starting from a neighborhood of the initial state $335 \leq T_{in} \leq 350$ and $305 \leq T_{out} \leq 320$. Figure 3.7 shows the reach set starting

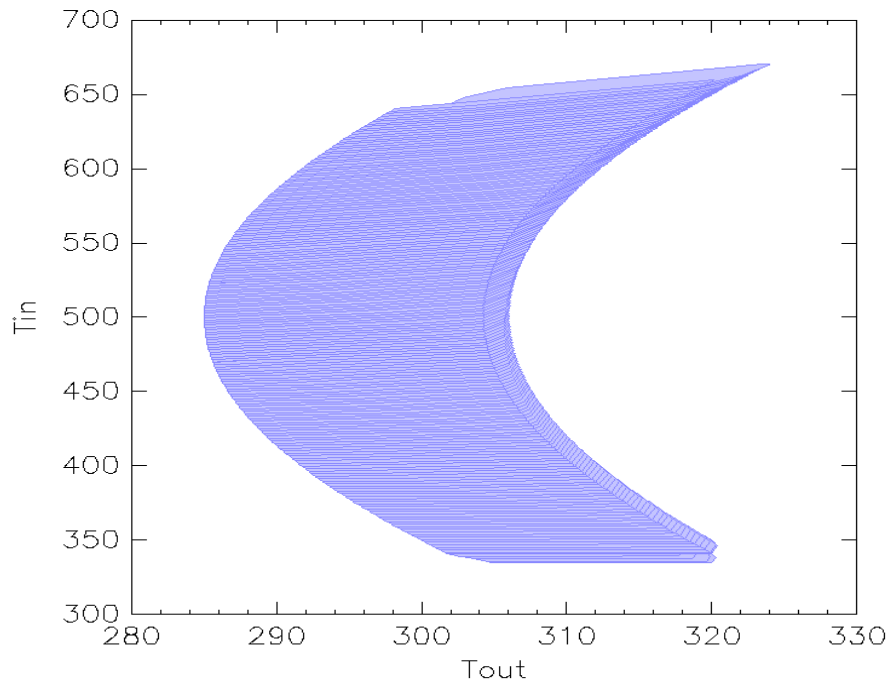


Figure 3.7: Reachability analysis of the CRAC hybrid automata in Example 9, which shows the values that T_{in} and T_{out} can possibly assume at any arbitrary time.

from the defined neighborhood. This figure shows that given any time the values of T_{in} and T_{out} for the CRAC will always remain in the shaded region. The generated power in the room is taken to be 200 W. In the highest power setting the CRAC extracts 75 W while in the lowest power state the CRAC extracts 50 W. The results show that the T_{in} increases to high values even if the T_{out} is within acceptable ranges. This is because the amount of power extracted by the CRAC is less than the amount generated in the room and hence the room gets heated up. Thus, using such arguments some values of the system parameters can be deemed unsafe. Using the reachability analysis, existence of such unsafe states can be determined. Such results can be used to conclusively prove whether a system is safe or unsafe.

□

Unlike a simulation, a reachability analysis can provide guarantees on the system properties for arbitrary time, however, it over estimates the system parameters. Hence, one does not get an exact estimate of the system parameters at a given time.

3.3 Synthesis Phase

Synthesis is the process of generating either the software implementation or the hardware emulation of a system from its model in a generic programming language or hardware platform, respectively. The software generated may often be used directly in the system. While the hardware emulation can be used for experimental verification of the system. Software synthesis generally requires a code generator that takes a model as input and converts into code in some generic programming framework such as C,C++, embedded C and so on. Hardware emulation takes a specification of a model using high level hardware design language (HDL) such as VHDL, PSpice, Verilog and System C.

Synthesis cannot be assumed as an optimized implementation of the system. But it is an implementation that can be used for experimentation in order to check requirements satisfied by the envisioned system.

Chapter 4

ARCHITECTURAL MODELING FOR FAST SIMULATIONS

Architectural models present an intuitive and fast way of representing CPSes. Such models represent the CPS as a set of connected components and subcomponents. Each component or subcomponent has properties associated with them. The connections represent data communication between the components or a subcomponent relationship. Architectural models are useful for fast simulation of a given test case. Generally they can be used to test the worst case operation of the CPS. The main contribution of this thesis in this front is the development of the CPS Design Analysis and Simulation CPS-DAS tool that provides a systematic intuitive specification format for CPSes and fast light weight analysis algorithm for various types of safety evaluations.

4.1 Motivation and Related Works

Figure 4.1 depicts the modeling requirements of CPSes, maps the related work to address the specific modeling requirements, and puts the contributions of CPS-DAS into perspective. There are three basic modeling requirements for CPSes: i) cyber entities, i.e. the application software (*e.g.*, health monitoring software) and networked computing units (*e.g.*, embedded medical devices), ii) physical environment, i.e. the human body, and iii) interactions between the physical environment and the cyber entities (both intentional and un-intentional). Intentional interaction modeling would involve sensor and actuator behavior modeling. Un-intentional interaction modeling require modeling the physical processes and the physical behavior of the cyber entities.

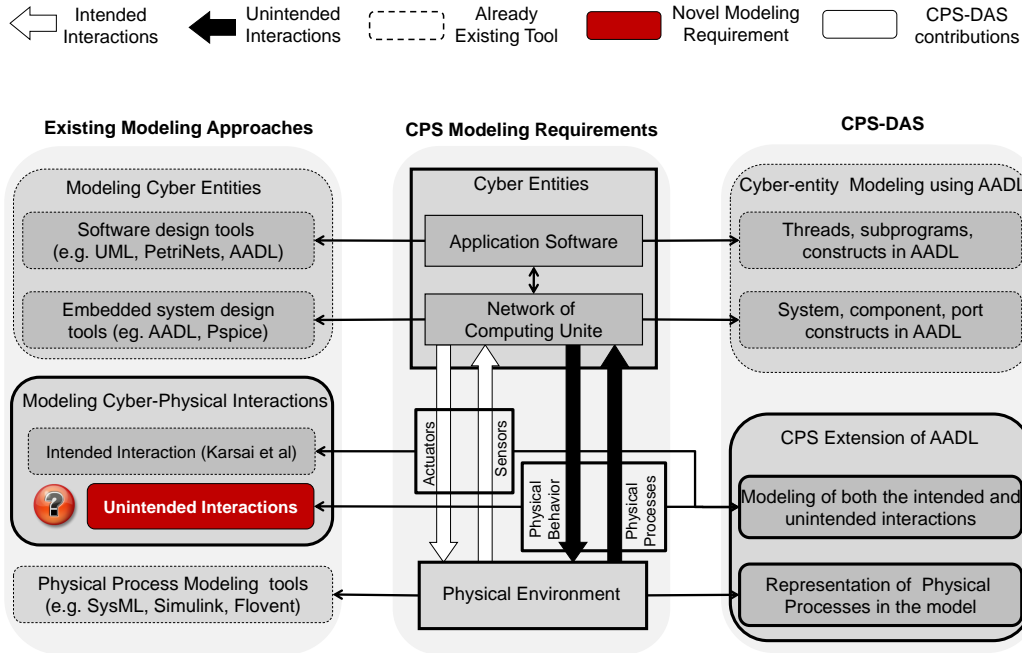


Figure 4.1: CPS modeling requirements and mapping of related works based on the requirements that they address with highlighting of CPS-DAS contributions

A large number of tools are available that model and analyze hardware of computing systems such as Pspice [78] and AADL (<http://www.aadl.info/>), and application software such as UML (<http://www.uml.org/>) and Petrinets [79]. The model based approach is also used to study the behavior of physical systems through tools such as SysML (<http://www.sysml.org/>), Simulink (<http://www.mathworks.com/>), and Flovent (<http://www.mentor.com/>). However, none of these tools can model the cyber-physical interactions. A generic framework to perform model based engineering of CPSes has been proposed in [80]. However, the framework only considers the intentional interactions for the proper functioning of the CPSes. The proposed CPS-DAS tool provides a uniform model specification and analysis platform that addresses all the three salient modeling requirements of CPSes.

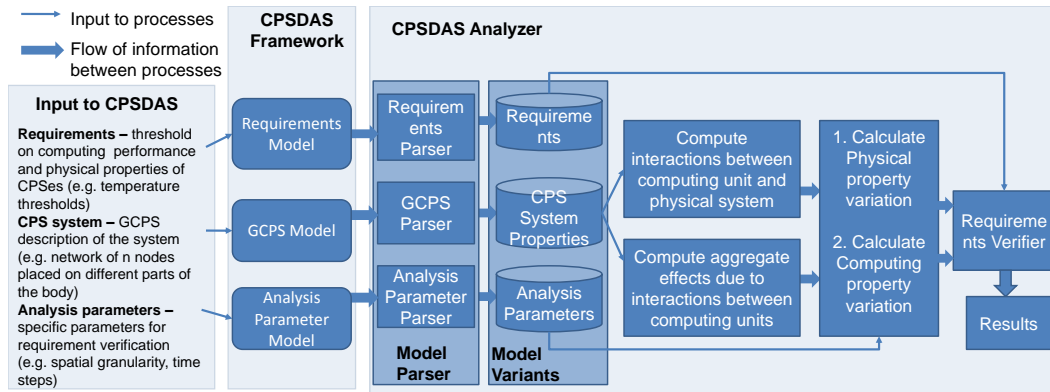


Figure 4.2: CPS-DAS tool architecture consisting of two parts: CPS-DAS modeling framework, which is used to specify a CPS and CPS-DAS Analyzer, which analyzes a CPS model.

4.2 CPS-DAS: Analysis and Design of CPSes

This section presents the CPS-DAS tool. Figure 4.2 shows the architecture of the tool and depicts how it enables the execution of the different phases in MBE. As shown in the figure, CPS-DAS consists of a model development framework, called the *CPS-DAS Modeling Framework* (Section 4.2), and a model analysis engine, called the *CPS-DAS Model Analyzer* (Section 4.2).

CPS-DAS Modeling Framework

The modeling framework helps in the model development phase of MBE. There are three inputs to the framework:

1. **CPS Requirements:** These are usually a set of limits or thresholds on the system parameters. For example, an upper limit on the maximum body temperature (system parameter) ensures safety.
2. **CPS System:** This includes CPS deployment information such as the number, types, spatial distribution, and communication topology of the nodes.

3. **Analysis Parameters:** These are specific parameters for the model analysis. Example of such parameters include time steps, spatial granularity of differential equation solvers (*e.g.*, Finite Difference Time Domain (FDTD) [81]), and functions determining the aggregate effects (*e.g.*, summation function to combine heat effects from multiple computing units).

Given these inputs, the following subsections describe modeling methodologies in the CPS-DAS modeling framework.

Requirements Modeling

The thresholds on the system parameters, which characterize the CPS requirements, can be modeled in CPS-DAS by providing a set of **constants**. These constants will be used in the requirements verification phase to check if the CPS behavior is suitably constrained.

Abstract Modeling of CPSs

CPS-DAS envisions a CPS as a set of computing (*i.e.* cyber) components (medical devices or servers or UAVs) and a physical environment (human body or recirculated air or UAV terrain). Figure 4.3 shows a typical deployment of the BSN CPS with both implanted and wearable (*i.e.* on the skin) sensor nodes. Both the intentional and un-intentional interactions are further depicted in the figure. Figure 4.4 shows all the modeling constructs in a hierarchical form. Following are the modeling constructs:

Global CPS (GCPS): A CPS is considered as a GCPS (Figure 4.3), *i.e.* a collection of distributed and networked cyber-physical subsystems. Each of these subsystems consists of a single worker node. The GCPS construct can also correspond to the portion of the physical environment observed for analysis, *e.g.*, the portion of human body monitored by the BSN.

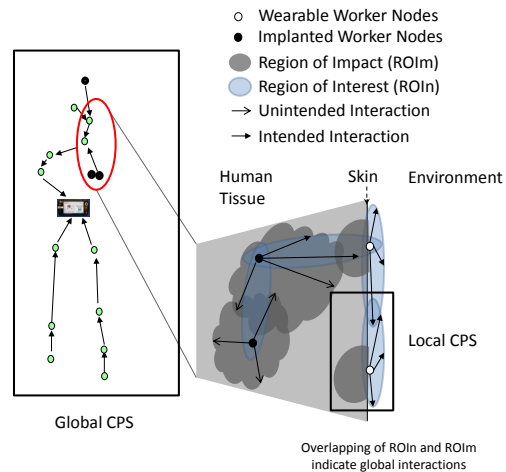


Figure 4.3: BSN as a Global Cyber-Physical System (CPS) with several Local CPSes each having region of interest and a region of impact. Overlapping of ROI_n and ROI_m indicate cyber-physical interactions.

Local CPS (LCPS): Each individual subsystem in a GCPS is referred to as an LCPS. The LCPS construct models each computing unit in a CPS as a single CPS. This enables modeling and analysis of the interactions between each individual node and the physical environment in a modular fashion. Each LCPS consists of three types of constructs:

- *Computing unit:* This construct corresponds to the computing units capable of sensing, computation, and communication. The construct facilitates the modeling of both computing and physical behavior of the computing units through the following two properties:
 - *Computing property:* This property characterizes the computing behavior, *e.g.*, the processor speed of a worker node or the amount of memory space available. The computing properties depend on the hardware configuration of the computing unit and the design of the application software. The hardware and the application software should be designed

such that CPS operations are accurate and have low latency. These design requirements specific to BSNs have been addressed in [3]. Trade-offs between these design requirements with safety are demonstrated in the specific case studies on BSNs(Section 4.3).

- *Physical property*: This property characterizes the physical behavior, *e.g.*, the power dissipation, of the computing unit. These properties often depend on the computing behavior of the nodes and can cause un-intentional interactions. For example, the processor speed determines the power dissipation of the sensor nodes in a BSN. This in turn causes un-intentional temperature rise of the human body.
- *Physical unit*: This construct is used to model the portion of the physical environment with which the computing unit interacts. These interactions usually affect the system parameters, for which the thresholds are provided in CPS requirements. For example, the heat dissipation (un-intentional interaction) from the nodes affects the body temperature (system parameter). The modeling paradigm of CPS-DAS hypothesizes that **any interaction of the computing unit with the physical world will take place within a bounded region**. Therefore, all interactions are limited within a region defined in the LCPS. To this effect, two types of regions for intentional and un-intentional interactions are specified in the physical unit. These regions are described below:
 - *Region-Of-Interest (ROIn)*: This construct facilitates the modeling of the intentional interactions, *e.g.*, the sensing and the wireless communication region. A ROIn has two attributes:
 - * **Monitored parameter**: This construct models the system parameters that are affected by the intentional interactions. For example,

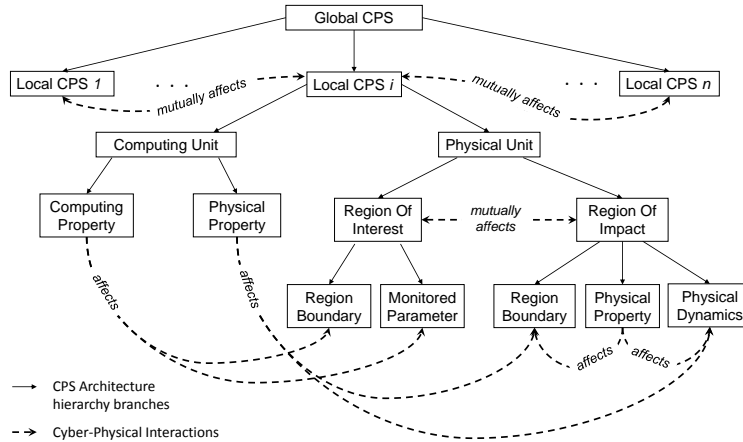


Figure 4.4: Hierarchical view of the generic constructs of a GCPS which is used to specify a CPS high level architectural model.

the physiological signal sensed by a sensor node can be a monitored parameter, which is affected by the nodes' sensing capabilities (electro-magnetic interaction [<http://www.quasarusa.com/>]).

- * **Region Boundary:** This attribute represents the limits of the bounded region, within which the intentional interactions are confined. The region boundary depends on the variation of the monitored parameters. For example, when the sensed signal is the monitored parameter, the sensing range of the sensor node becomes the region boundary.
- *Region of impact (ROI_m):* This construct is used to model the unintentional interactions. The ROI_m consists of three attributes:
 - * **Physical Property:** This attribute characterizes the properties of the physical system such as blood glucose level, tissue temperature etc. These properties depend on the location of the node, physiological conditions, and environmental factors such as temperature, and pressure (<http://urwhatueat.org/carb5.html>).

- * **Physical Dynamics:** This construct enables modeling the physical processes. These processes are normally expressed in terms of partial differential equations. For example, the process of body temperature variation is governed by the Penne's bioheat equation [4].
 - * **Region Boundary:** This is similar to region boundary in ROI_n. However, the region boundary of ROI_m depends on the physical properties and dynamics. Since the physical dynamics are governed by differential equations, the region boundary can be specified by boundary conditions on the equations. These conditions are generally limits on the physical properties outside the ROI_m. For example, in case of temperature rise in human body, we can assume that the body temperature is 37 °C outside the ROI_m. We can then employ this boundary condition to the associated differential equation [4] to obtain the region boundary.
- *Local Interactions:* The local interactions are cyber-physical interactions between the computing unit and the physical unit within an LCPS (denoted by the dashed lines in Figure 4.4). The intentional and un-intentional interactions are modeled using the following constructs, respectively:
 - **Intended interactions:** These are modeled as transfer of information between the computing unit and the ROI_n. For example, the sensing of physiological signals from the human body by a node can be modeled by this construct.
 - **Unintended Interactions:** These interactions as modeled as transfer of energy between the computing units and the ROI_m. For example, heat transfer from the computing unit can be modeled by this construct.

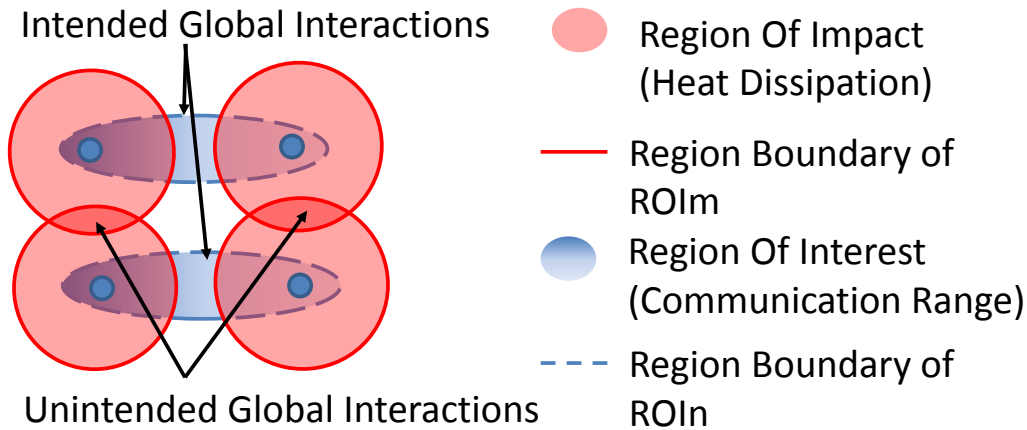


Figure 4.5: Global Interactions between two local CPSes. Intersection between the ROI_n of two LCPS indicate intended interactions while that between ROI_m of two LCPS indicate unintended interactions.

Note that an interaction is also defined between the ROI_n and the ROI_m of an LCPS. These are used for representing certain special cases, where there are inter-dependencies between the ROI_n and ROI_m.

Interactions among the LCPSs: In the GCPS model, interactions between different LCPSs can also take place. For example, the wireless communication between two sensor nodes in a BSN is a form of information transfer between two different LCPSs. This phenomenon can be modeled by introducing the concept of interconnections between the LCPSs. These interconnections are called *global interactions* (as depicted by the two way dashed arrows in Figure 4.4). Global interactions between two LCPSs normally occur whenever there is an overlap in the ROI_n or the ROI_m of the two (as shown in Figure 4.5). This notion of global interaction facilitates the modeling of a network of nodes in a CPS (through overlapping ROI_ns) or the cumulative thermal effect of the nodes on a particular area of the physical environment (overlapping ROI_ms). Global interactions can be intended or unintended due to overlapping of ROI_ns or ROI_ms of the interacting LCPSs, respectively.

Analysis Parameter Modeling

Analysis of a model generally involves specific methodology to solve equations that govern the physical dynamics. The solution techniques often require a configuration, i.e. assigning values to certain specific parameters determining the quality of the solutions. For example, solving the Pennes' equation will require the FDTD time and space discretization approach. The granularity of such discretization can be specified in the model as sets of *constants*. Further, the aggregate functions can be specified as a set of equations combining the physical dynamics of different LCPSs.

CPS-DAS Analyzer

The CPS-DAS analyzer determines the CPS behavior from the GCPS model. The behavior is then verified against the requirements given as input to the CPS-DAS modeling framework. Figure 4.2 shows the work flow of the CPS-DAS analyzer.

The models from the CPS-DAS modeling framework are first parsed into a suitable format by a **Model Parser**. There are three different components of the parser: 1) requirements parser, 2) GCPS parser, and 3) analysis parameter parser. Each of these components correspond to the three modeling aspects of the modeling framework (Section 4.2). The output of the *Requirements Parser* is a set of constants indicating the thresholds on different system parameters. *Analysis Parser* provides parameters (a set of constants) related to the analysis methodology (e.g., configuration for equation solvers, time step and spatial granularity). The *GCPS Parser* extracts the hierarchical organization of the GCPS model in a structure using which interactions and aggregate effects can be computed. The outputs of the parsers are called the *Model Variants*.

The pseudocode of the analyzer is shown in Figure 4.6. The *EvaluatePhysicalProperty* and *EvaluateMonitoredParameter* routines evaluate the variations of

Process: CPSDAS Analyzer (CPSModel)

Model Parser /* Parse model and extract the Model Variants */

```
GCPS = Model Parser(CPSModel.GCPSModel);  
[Performance Thresholds] = Requirements parser(CPSModel.RequirementsModel);  
[Analysis Parameters] = Analysis Parameter parser(CPSModel.AnalysisParametersModel);
```

Interactions between computing unit and physical system

```
for each i from 1 ... n  
    GCPS.LCPS(i).ROIm.Physical Property = EvaluatePhysicalProperty(GCPS.LCPS(i).ROIm, AnalysisParameters)  
for each i from 1 ... n  
    GCPS.LCPS(i).ROIn.Monitored Parameter = EvaluateMonitoredParameter(GCPS.LCPS(i).ROIn, AnalysisParameters)
```

Interaction between different computing units

```
for each i from 1 ... n  
    for each j from 1 ... n  
        if GCPS.LCPS(i).ROIm.RB overlap with GCPS.LCPS(j).ROIm.RB  
            IntFm(i,j) = 1;  
        else  
            IntFm(i,j) = 0;  
        if GCPS.LCPS(i).ROIn.RB overlap with GCPS.LCPS(j).ROIn.RB  
            IntFn(i,j) = 1  
        else  
            IntFn(i,j) = 0;  
    for each IntFm(i,j) == 1  
        GCPS.LCPS(i).ROIm.Physical Property = Compute the aggregate effect in the intersecting region;  
        GCPS.LCPS(j).ROIm.Physical Property = Compute the aggregate effect in the intersecting region;  
    for each IntFn(i,j) == 1  
        GCPS.LCPS(i).ROIn.Monitored Parameter = Compute the aggregate effect in the intersecting region;  
        GCPS.LCPS(j).ROIn.Monitored Parameter = Compute the aggregate effect in the intersecting region;
```

Requirements Verification

```
for each Param.time step  
    for each Param.space step  
        Compare Monitored Parameter with performance thresholds  
Return Verification Results
```

Figure 4.6: Pseudocode for CPS-DAS Analyzer, which first parses the GCPS model, computes interactions within an LCPS, and then computes global interactions between LCPSes. Finally, the system properties obtained from the computation is checked with requirements for safety verification.

the physical properties and monitored parameters within the Region Boundaries (denoted by RB) of the ROIm and ROIn of each LCPS, respectively. These evaluations essentially compute the local interactions within an LCPS. Note that the representation of the Region Boundary by the notation GCPS.LCPS(i).ROIm.RB reflects the hierarchical nature of the constructs (Figure 4.4). Similar notations for the other constructs are also used.

The next process is to identify and compute the interactions between different computing units. This process involves the computation of the global interactions. In the pseudocode, this is achieved by iterating through all n^2 possible pairs

of LCPSs (where n is the number of nodes in the BSN) and checking whether the RBs of LCPS pairs overlap. These overlapping regions are tracked by the parameters, IntFm and IntFn, for ROI_m and ROI_n, respectively. The aggregate variation of the physical properties and the monitored parameters are then evaluated for these regions of intersections. To this effect, the aggregation functions provided in the analysis parameter model are used.

After computing the interactions and aggregate effects, the analyzer checks the monitored parameters and the physical properties against the requirements. This process is called the requirements verification process.

Implementation

Given the modeling framework and the analysis work flow in the previous subsections, this subsection presents the current implementation of CPS-DAS. CPS-DAS is currently implemented for BSNs and is known as BAND-AiDe [2]. However, the same implementation can be used to specify any CPS. The modeling framework of CPS-DAS is developed using the Abstract Architecture Description Language (AADL). AADL is an industry standard language to model real time embedded systems and is suitable for implementing CPS-DAS due to the following reasons:

- AADL specifications are hierarchical in nature, which helps in specifying the GCPS model.
- AADL has dedicated construct to model hardware and software of embedded computing devices. This will be helpful in modeling the behavior of the computing units in the CPS.
- AADL has been used to model CPSes such as wireless sensor networks [82], UAVs (aadl.info).

- AADL provides facilities for language extension through development of annexes. These facilities can be used to incorporate the generic constructs of the GCPS model in AADL.

Model Specification

The GCPS in the GCPS model (Figure 4.4) can be represented using the *system* construct in AADL (as shown in Figure 4.7) and is named **Global CPS (GCPS)**. In the GCPS, the monitored region of the human body is specified as a grid. The origin and the units of the grid are specified using the *properties* construct. The constants for the requirements specification and the analysis parameters can be provided in the GCPS specification using separate *property sets*, which are AADL constructs to specify system attributes. The **GCPS** consists of several *subcomponents* called **Local CPS (LCPS)**. These are also modeled using the *system* construct. Each **LCPS** consists of three *subcomponents*: 1) **Computing Unit**, which models a single worker node, 2) **Region of Interest**, which models the ROI_n, and 3) **Region of Impact**, which models the ROI_m. The *system* construct is used to model these subcomponents. The **Computing Unit** has two *property sets*, **Computing Property Set** and **Physical Property Set**, to model its cyber and physical behavior, respectively. The **Computing unit** can be modeled in a conventional way, containing *subcomponents* such as processor, memory, radio, and bus [82].

The location of the computing unit is specified using the **Location property set**. This property set gives the coordinates of the computing unit with respect to the origin and grid units defined in the GCPS. Within an ROI_m, the position of the computing unit is considered to be the origin. The ROI_m consists of properties from the **Physical Property Set**. The dynamics of the physical system which involves specification of complex analytical expressions is specified with the help of the **CP-**

Declaration – GlobalCPS (GCPS) <i>properties</i> Control Volume Specification • (x, y, z) coordinates • Grid Units	Declaration - Computing Unit <i>features</i> port group Cyber2ROIn port group Cyber2ROIm <i>properties</i> Computing Property Set Physical Property Set	LEGENDS: 1. Port group are a collection of connections that model the cyber-physical interactions that model the cyber-physical interactions a) Cyber2ROIn – Computing to Region of Interest Interaction b) Cyber2ROIm – Computing to Region of Impact Interaction c) ROIn2Cyber – reverse of Cyber2ROIn d) ROIm2Cyber – reverse of Cyber2ROIm e) LCPSROIn – interaction between LCPSs' ROIns f) LCPSROIm – interaction between LCPSs' ROIm 2. Property Sets a) Computing Property Set – Computing behavior b) Physical Property Set – Physical behavior
Implementation - GlobalCPS <i>subcomponents</i> LCPS 1, LCPS 2,, LCPS n <i>annex</i> Aggregation Equations <i>connections</i> Connection between LCPS via port groups	Implementation - Computing Unit <i>subcomponents</i> Subcomponents instantiation <i>properties</i> Specific values <i>annex</i> Customized functionalities <i>connections</i> Connection between components via ports and devices	
Declaration – LocalCPS (LCPS) <i>features</i> port group LCPSROIn port group LCPSROIm	Declaration – Region Of Interest <i>features</i> port group ROIn2Cyber	
Implementation - LocalCPS <i>subcomponents</i> Computing Unit Region of Interest Region of Impact <i>connections</i> Connection between Computing Unit and Region Of Interest Connection between Computing Unit and Region Of Impact	Implementation – Region Of Interest <i>properties</i> Location ... <i>annex</i> Customized functionalities	
		Declaration – Region Of Impact <i>features</i> port group ROIm2Cyber Implementation – Region Of Impact <i>properties</i> Location Physical Property Set <i>annex</i> BAN-CPSAnnex Specify boundary condition equations Specify equations for physical process

Figure 4.7: AADL specification of the GCPS model (a more detailed model can be found in the publication [2])

SAnnex. This *annex* extends AADL with customized constructs for specification of partial differential equations. The **Region Boundary** of the ROIm is represented by boundary conditions on the properties of physical unit. Similar approach is taken to implement the ROIn of the LCPS.

The global and local interactions in the GCPS model are specified with the help of port group AADL construct. The port group construct is an assembly of different types of ports. These ports, along with their interconnection using the connections construct, models information transfer among the subcomponents. The **Computing Unit** has two *port groups*: 1) **CyberToROIn**, which specifies the *intended interaction* of the computing unit with the ROIn and 2) **CyberToROIm**, which specifies the unintended interaction of the computing unit with the ROIm. The global interactions are modeled using two types of *port groups*, **LCPSROIn** and **LCPSROIm**, in

each LCPS. These *port groups* signify interactions between a computing unit of an LCPS and the ROIn and ROIm of some other LCPS, respectively.

Analysis framework

The analysis framework is developed on the OSATE platform, which supports AADL model specification and analysis through java plug-ins in eclipse. The analysis framework uses the parsers provided by the OSATE platform to parse the AADL constructs. A new parser was developed for the customized differential equation constructs in the **CPSAnnex**. In GCPS Annex specific constructs were declared for denoting differential equations. The construct *DeltnXY* represents the mathematical operator $\frac{\delta^n X}{\delta^n Y}$. The differential equations were then parsed using the following context free grammar:

```
Operator =  
Variable * DeltnXY * Operator + Variable * Operator |  
    Variable * DeltnXY * Operator - Variable * Operator  
    | null  
DeltnXY = Delt1XY | Delt2XY .....  
Variable = any string | null
```

The parsed differential equations are represented in the form of a parse tree. In the Analysis plug-in, the parse tree representation of the equations was converted into a mathematical form. Currently, the GCPS Annex converts the parsed equations into FDTD form which can be solved using any FDTD solver. To this effect, the analysis plug-in is integrated with domain specific tools such as Matlab. The analysis results were provided in a graphical format through Matlab.

4.3 Case Study for Design and Analysis with CPS-DAS

This section shows the usage of the CPS-DAS tool to model and analyze BSN CPS. To this effect, the following case studies are considered: i) a BSN with wearable and implanted sensor nodes, and ii) a secure health monitoring application, Ayushman [5]. The sensor nodes are assumed to be capable of:

- sensing temperature, humidity, sound, and physiological signals (e.g., Photo-Plethysmogram (PPG));
- data communication through wireless radio; and
- communication security through Physiological value based Key Agreement (PKA) [64].

Figure 2.1 shows the Ayushman workload highlighting the three operations. As shown in Figure 2.1, the operations in the worker nodes have different CPU utilization profile. These utilization profiles determine the nodes power consumption. Hence, proper duty-cycling and communication scheduling can be employed to reduce the nodes' power consumption. The nodes are assumed to be always expending the maximum power, P_c , if duty-cycling is not employed. On the other hand, when duty-cycling is employed, then the nodes do not always consume maximum power. In such a case, the power consumption of the sensor node when radio is turned off is given by P_{proc} . During the communication operation, the processor consumes P_{proc} amount of power. The radio transmitter will also be active during this operation (P_{radio} being its power consumption). In a single day, there will be d number of sense and transmit periods (sleep cycles) for each sensor node in the BSN, with a duration of $(t_s + t_{Tx})$ seconds each, where t_s and t_{Tx} denote the time required to perform the sensing and communication operations, respectively..

The symbol t_{PKA} denotes the duration to execute the PKA. The total number of nodes is assumed to be n , and the nodes execute pairwise PKA (once everyday) to ensure freshness of keys. In PKA, frequency domain features are derived from the sensed physiological signals and are used to facilitate key agreement between two sensors. The feature extraction process involves several complex computations such as Fast Fourier Transform (FFT) and peak detection. The node power consumption during the PKA operation is comprised of the power required for the aforementioned computation, P_{PKA} , and the power required to keep the radio on, i.e. P_{radio} . In general, duty-cycling is not considered for safety verification to analyze the thermal effects of worst-case node power characteristics.

To obtain the model parameters for CPS-DAS through experimental evaluations, Ayushman was implemented in a BSN consisting of TelosB motes interfaced with temperature, humidity, and physiological sensors. The implementations were performed to meet the design requirements of accuracy and low latency. Figure 4.8 shows the implementation strategy of the peak detection stage of the PKA protocol¹. The usage of the CPS-DAS tool is demonstrated through safety verifications for: i) a *single medical device*, and ii) a *network of sensor nodes*, as summarized in Figure 4.9 and described in the following subsections.

Safety Analysis of a Single Wearable Medical Device

This subsection presents the safety verification for a single high-power wearable medical device. To this effect, the medical device considered is a TelosB mote interfaced with a Smith fingertip pulse oximeter (PPG) device (<http://www.smithsoem.com/>) deployed on the index finger. The pulse oximeter probe which is in direct contact with the human finger passes light at a particular intensity through the finger

¹The threshold indicated in the figure is specific to the peak detection [3] process and is not the same as the requirements thresholds discussed in this thesis

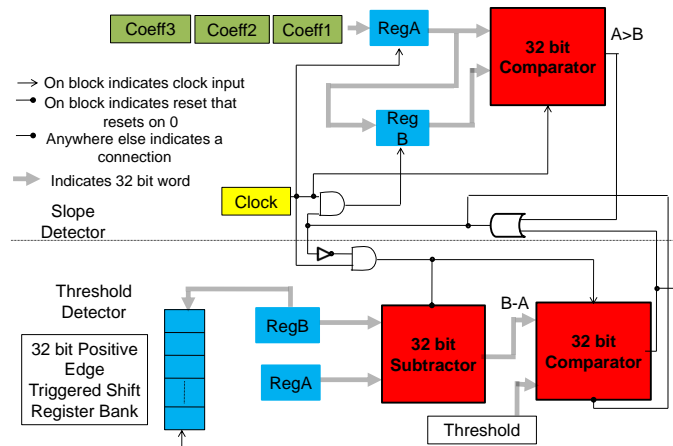


Figure 4.8: Implementation logic for Peak Detection (a more detailed discussion can be found in the publication [3])

during its operation. The pulse rates are then derived from the modulations in the sensed light intensity. Power is consumed by the device to execute the Ayushman workload, i.e. sensing, transmission, and PKA execution, as shown in Figure 2.1.

CPS-DAS Inputs

The safety requirement input to CPS-DAS is the upper threshold on the skin temperature, which is 39 °C as obtained from the Henriques Moritz equation [83]. Pennes' bio-heat equation is used to characterize the thermo-dynamic processes. The thermo-dynamics of the finger comprises of heat transfer mechanisms such as thermal conduction, convection, radiation, and electro-magnetic absorption. The thermal safety analysis parameters include the time steps and space granularity (e.g., grid size, where a grid is the smallest volume of the human body within which the temperature is assumed to be constant) of the solvers for equations (e.g., Pennes' equation) describing the thermo-dynamic processes.

Verification	Input	Model	Analysis
Safety and sustainability of a single wearable medical device (Pulse Oximeter)	<p>Requirements : <u>Safety:</u> Threshold Temperature T_{th} of the human body which should not be exceeded [Henriques and R. 1947].</p> <p>BAN system: Single sensor node consisting of a TelosB mote interfaced with the fingertip Pulse Oximeter device deployed on the index finger. Scavenging sources are considered on the human body which charge the nodes through inductive charge transfer.</p> <p>Analysis Parameter : Differential equation solver FDTD parameters, (time step and space steps)</p>	<p>Requirements Model Requirement Verification Parameter T_{th}</p> <p>BAN Model GCPS - 1 LCPS. LCPS - Computing Unit (CU) + ROIm CU – subcomponents + properties subcomponents – LED, Radio, Processor, Scavenging source properties – Power dissipation, Temperature ROIm – Physical Property + Physical Dynamics + Region Boundary Physical Property – Temperature Physical Dynamics – Penne’s bio heat equation Region Boundary – 30 mm by 30 mm square region ROIn – Charging range of the scavenging source</p> <p>Analysis Parameter Time Step and Space granularity parameters of FDTD solver</p>	<p>Safety: Thermal map of the fingertip skin for continuous operation of the pulse oximeter is tested against safety threshold set by the requirements.</p>
Safety evaluation of Network of Devices communicating securely	<p>Requirements : <u>Safety:</u> Lower bound on the tissue temperature for the optimal cluster head selection sequence.</p> <p>BAN system: Network of worker nodes implanted within the human tissue. The worker nodes are interfaced with EKG sensors. Communication protocol is cluster based [Heinzelman et al 2000], where several nodes form a cluster with a nominated cluster head. The cluster head collects data from all the cluster members and sends them to the wearable worker node. A cluster head selection sequence is specified, which denotes the order in which cluster heads are selected during the data transfer process. The Ayushman health monitoring application is executed on each node. The radio of each worker node can be turned off during different stages of their operation. Scavenging sources charging the nodes via inductive charge transfer are assumed.</p> <p>Analysis Parameter : Cluster head selection sequence, differential equation solver FDTD parameters, (time step and space steps) Sensing Rate, data transmission rate, rate of PKA execution, time interval of observation.</p>	<p>Requirements Model Upper threshold T_{th} on the maximum temperature rise of the human tissue</p> <p>BAN Model GCPS – n LCPS LCPS – Computing Unit (CU) + ROIn + ROIm CU –properties properties – Power Dissipation ROIn – Monitored Property + Region Boundary Monitored Property – Radio signal strength Region Boundary – Communication Range (intersection of ROIn indicate connectivity) ROIm – Physical Property + Physical Dynamics + Region Boundary Physical Property – Temperature Physical Dynamics – Penne’s Equation Region Boundary – preset value (intersection of ROIm require evaluation of cumulative effect)</p> <p>LCPS – Scavenging Source + ROIn CU – properties properties - Available power for scavenging source ROIn – A worker node receives power from a scavenging source if it falls inside the ROIn.</p> <p>Analysis Parameter Cluster head selection sequence Time Step and Space granularity parameters of FDTD solver Data transmission time t_{TX}, Sensing time t_s and PKA execution time t_{PKA}</p>	<p>Safety: Each cluster head selection sequence is evaluated to compute the thermal map of the tissue. Different cluster head selection sequences are compared based on the maximum temperature in the control volume.</p>

Figure 4.9: Summary of case studies showing CPS-DAS usage for safety evaluations that are further discussed in detail in Section 4.3

CPS-DAS Model

The requirement and analysis parameter modeling involves representation of the skin temperature threshold, and the time steps and grid sizes (to solve Pennes’ equation) as constants. The GCPS is represented as a system with a single LCPS. The medical device is modeled as the computing unit, which has several subcomponents such as LED array, radio, and processor. These subcomponents are the generic components of the pulse oximeter device. Each subcomponent has a set of

physical properties (such as power dissipation, operating temperature) that model their thermal characteristics. The ROI_m models the thermodynamics of the human skin using the Pennes' bioheat equation as follows:

$$\rho C_p \frac{dT}{dt} = K \nabla^2 T - b(T - T_b) + \rho \text{SAR} + P_c, \quad (4.1)$$

where ρ is the mass density, C_p is the specific heat, K is the thermal conductance, T is the temperature of the skin, P_c is the power generated by any heat source (i.e. the device), b is the blood perfusion constant, T_b is the blood temperature, and SAR is the specific absorption rate of the skin (i.e. the amount of electromagnetic radiation absorbed by unit volume of the skin).

The region boundary of the ROI_n is preset to a 30mm × 30mm square region on the fingertip skin, which is the size of the pulse-oximeter probe (<http://www.smithsoem.com/>) that is in direct contact with the fingertip. The ROI_n also models the range, up to which a scavenging source can charge a sensor node. To this effect, only body heat and sunlight are considered as the possible scavenging sources at the fingertip. The operating temperature of the pulse-oximeter probe is assumed to be constant [41]. This is because the Ayushman workload (Figure 2.1) is highly repetitive, which causes the probe temperature to reach a steady state quickly. Given the inputs and the CPS-DAS model, the CPS-DAS analyzer performs the safety analysis (described in Sections 4.3) based on the work-flow in Figures 4.2 and 4.6.

Safety Verification

CPS-DAS analyzer computes the temperature at different points on the skin by solving the Pennes' equation using the FDTD approach.

Table 4.1 shows the maximum skin temperature reached during the eight hours of operation of the pulse oximeter at different device temperatures. A sample

Table 4.1: Skin temperatures after eight hours of pulse oximeter operation at different device temperatures (Burn threshold 39 °C)

Device Temperature	Maximum Skin Temperature
43.0 °C	38.2 °C
43.5 °C	38.5 °C
44.0 °C	39.2 °C
44.5 °C	39.4 °C
45.0 °C	39.7 °C

thermal map of the skin for a pulse oximeter operating temperature of 44 °C is shown in Figure 4.10. It can be verified that the maximum temperature reached after eight hours of operation is 39.2 °C, which violates the thermal safety requirement. An experimental study performed on pulse oximeter thermal safety [41] shows that blisters are observed in human skin when a pulse oximeter is operated continuously for eight hours at a device temperature of 44 °C. Our results concur with these experimental observations and are hence verified.

Safety Analysis of Network of Devices

This section presents the safety verification for a network of sensor nodes in a BSN. To this effect, we consider low-power devices (as sensor nodes), *e.g.*, EKG sensors, interfaced with the TelosB motes. High power nodes are not considered since it has already been verified (in Section 4.3) that even a single such node can cause safety violations. Cluster based multi-hop communication protocol is used [84] for the network. In this protocol, the sensor nodes in the BSN form a cluster. Nodes in a cluster nominate a leader node, a.k.a. the cluster head. All the non-leader nodes transmit their sensed data to the cluster head. The cluster head then forwards these data to the base station, and hence has a considerably high communication workload. The absorption of electromagnetic radiation (because of communication) can cause heat dissipation, which in turn may lead to significant thermal effects in the surrounding human tissue [81].

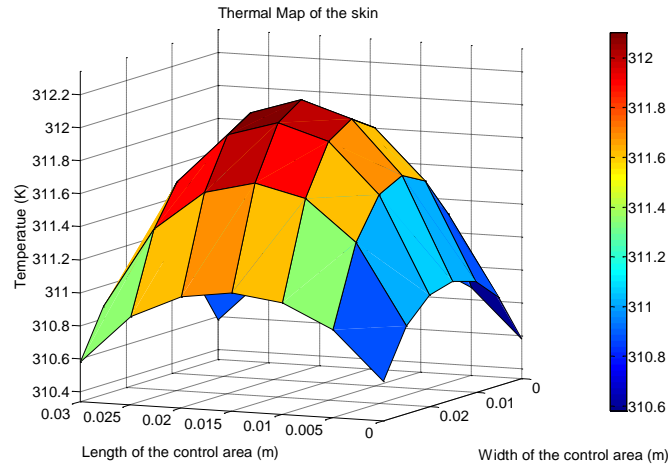


Figure 4.10: Thermal map of fingertip skin for 8 hrs of pulse oximeter operation at 44 °C temperature

CPS-DAS Inputs

Similar to the single device verification in Section 4.3, the safety requirement input for verifying the network of nodes is the upper threshold on the skin temperature, i.e. 39 °C. The exact values use for the analysis are shown in the Table 4.2. Analysis parameters corresponding to the Pennes' equation are same as that in Section 4.3. The control volume was divided into 30×30 cells where the cell size was set to 0.005m. The aggregation function for the heat effect from multiple contributing nodes is provided as follows: i) summation of the SAR values from the contributing nodes; ii) summation of the power generated from the contributing nodes; and iii) apply the summed SAR and generated power values in the Pennes' equation (i.e. Equation 4.1). In addition, the leadership sequence is input as an analysis parameter.

CPS-DAS Model

GCPS consists of multiple LCPSs. The LCPS modeling for each sensor node is same as in Section 4.3. In addition, the ROI of each LCPS models the communica-

Table 4.2: Timing and power consumption values used for the Ayushman workload and the experimental methodology

Parameter	Value	Methodology
t_s	5 s	Ayushman workload characteristics
t_{Tx}	0.39 s	Time taken to transfer five seconds of 32 bit data values sampled at a rate of 60 per second. Transfer rate is 24 Kbps which is standard for Chipcon radio [85]
t_{PKA}	16.36 s	Measured time in TelosB motes [85]
P_{radio}	56 mW	Measurements from the Chipcon radio in TelosB motes [85]
P_{PKA}	3 mW	Measured power of the TelosB mote while executing PKA [85]
P_{proc}	0.01 mW	Idle power of the TelosB mote [85]

tion range of the node. The intersections between the ROInS reflect the connectivity among the corresponding worker nodes. The leadership sequence is represented as a string of constant integers. Scavenging nodes are further modeled as LCPSs, where the ROIn represents the inductive charging range. Intersection of the ROIn of a worker node and a scavenging source means that the worker node can extract power from that source.

Safety Analysis

To evaluate the tissue temperature distribution, Pennes' bio-heat equation is solved using the FDTD solvers (as in Section 4.3). This aggregate effect was calculated by summing up the heat contribution of each node to a particular grid, as indicated in Section 4.3. The power consumption of leader sensor node executing the Ayushman application was 60mW [85] while that of a non leader sensor node was 12 mW. This power consumption was experimentally measured for TelosB motes at 0 dB and -7 dB radio attenuation, respectively. For a sample BSN cluster of 10 sensor nodes, the leaders were changed every second. Each leadership sequence was operated once for a short period of 1000 seconds and then for a long duration of two days. Our results for 1000 second of exposure matches with the observations in [81]. Table 4.3

Table 4.3: Tissue temperature rise for different leadership sequences (Burn threshold 39 °C)

Leadership Sequence	Maximum Tissue Temperature	
	1000 Sec Expo- sure	2 Days Exposure
(5 2 8 6 1 7 3 4 10 9)	37.1145 °C	37.1632 °C
(5 7 4 1 6 10 8 2 9 3)	37.1130 °C	37.1614 °C
(1 6 9 10 2 7 5 4 3 8)	37.1124 °C	37.1757 °C
(5 7 1 9 10 8 4 2 6 3)	37.1130 °C	37.1585 °C

shows maximum tissue temperatures for different leadership sequences at different exposure times. The maximum temperature is below the safety threshold of 39 °C for all these cases. It can be seen from the table that for a short duration of exposure, different leadership sequence does not have much effect on the tissue temperature rise. However, for longer durations, the temperature rise is significant (around 0.02 °C). This shows that for long term operation of implanted network of medical devices the leadership sequence plays an important role in thermal safety.

Apart from BSN CPS the CPS-DAS tool is also used on data center CPS as shown in the following case study.

Thermal Safety of Servers in a Data center

In this section, the usage of CPS-DAS for analyzing safety of servers from redlining is shown. The recirculation of heat has to be modeled using the GCPS constructs.

CPS-DAS modeling of data centers

Figure 4.11, represents the data center as a GCPS. Each server in the data center is an LCPS. The GCPS system has 50 port groups that specify the recirculation of air from the room to the input of each server (lines 3-4). Each LCPS has a port group that specifies the transfer of heat from the computing units to the environment (line 8). The heat recirculation matrix is implemented as a data component (lines 10 - 13) and each entry in the recirculation matrix is implemented as ports such that all


```

1. system GCPS
2. features
3.   P2C01: port group RecirculationPG;
4.   P2C02: port group RecirculationPG;
5. end GCPS;

6. system LCPS
7. features
8.   C2P: port group RecirculationPG;
9. end LCPS;

10. data implementation RecirculationMatrix.imp
11. subcomponents
12.   A: data behavior::integer;
13. end RecirculationMatrix.imp;

14. port group RecirculationPG
15. features
16.   Matrix1: in out data port RecirculationMatrix.imp;
17.   Matrix2: in out data port RecirculationMatrix.imp;
18. end RecirculationPG;

19. system Chassis
20. properties
21.   PhysicalPropertySet::IdlePowerDissipation => 0.0 W;
22.   PhysicalPropertySet::PowerDissipationPerUtilization => 0.0 W;
23. end Chassis;

24. system BladeServer
25. properties
26.   ComputingPropertySet::TypeOfComputingUnit => "";
27.   ComputingPropertySet::NumberOfServers => 0;
28. end BladeServer;

29. system implementation ImpactingParam.imp
30. subcomponents
31.   Tin: data behavior::float;
32. end ImpactingParam.imp;

33. system implementation ImpactedParam.imp
34. subcomponents
35.   Tout: data behavior::float;
36. end ImpactedParam.imp;

37. system implementation RegionOfImpact.imp
38. subcomponents
39.   ImplngParam: system ImpactingParam.imp;
40.   ImpEdParam: system ImpactedParam.imp;
41. end RegionOfImpact.imp;

42. system implementation BladeServer.imp01
43. properties
44.   ComputingPropertySet::TypeOfComputingUnit =>
"1955";
45.   ComputingPropertySet::NumberOfServers => 4;
46. end BladeServer.imp01;

47. system implementation LCPSManagement.imp
48. subcomponents
49.   PlacementAlgorithm: process HTSALgo.impl;
50. end LCPSManagement.imp;

51. system implementation PhysicalUnit.Air
52. properties
53.   PhysicalPropertySet::Cp => 1005.0 JpkgpK;
54.   PhysicalPropertySet::Volume => 290.304 m3;
55.   PhysicalPropertySet::Density => 1.19 kgpm3;
56. end PhysicalUnit.Air;

57. system implementation Chassis.imp01
58. subcomponents
59.   Blade01: system BladeServer.imp01;
60.   Blade02: system BladeServer.imp01;
61.   Utilization: data behavior::integer;
62.   Power: data behavior::integer;
63.   Idle: data behavior::integer;
64.   Unit: data behavior::integer;
65. properties
66.   PhysicalPropertySet::IdlePowerDissipation => 2420.0 W;
67.   PhysicalPropertySet::PowerDissipationPerUtilization => 175.0 W;
68.   PhysicalPropertySet::FlowRate => 0.2454147 m3ps;
69. annex CPSAnnex {**
70. states
71.   s0 : initial complete state;
72. transitions
73.   s0 -[ ]-> s0 {Power := Idle + Unit * Utilization; };
74. **};
75. end Chassis.imp01;

76. system implementation LCPS.imp01
77. subcomponents
78.   ComputingUnit: system Chassis.imp01;
79.   ROIIm: system RegionOfImpact.imp;
80. annex behavior_specification {**
81. states
82.   s0 : initial complete state;
83. transitions
84.   s0 -[ ]-> s0 {
85.     C2P.Matrix1 := a1;
86.     C2P.Matrix2 := a2;
87. };
88. **};
89. end LCPS.imp01;

90. system implementation ACUnit.imp
91. Model: "Linear Transient Model"
92. end ACUnit.imp;

93. system implementation GCPS.imp
94. subcomponents
95.   LCPSManager: system LCPSManagement.imp;
96.   LCPS01: system LCPS.imp01;
97.   LCPS02: system LCPS.imp02;
98.   PhysUnit: system PhysicalUnit.Air;
99.   LCPSAC: system LCPS.ACUnit;
100.   Tout: data behavior::integer;
101.   Tin: data behavior::integer;
102.   Recir: data behavior::integer;
103.   Power: data behavior::integer;
104. connections
105.   Recirc01: port group LCPS01.C2P -> P2C01;
106.   Recirc02: port group LCPS02.C2P -> P2C02;
107. annex CPSAnnex {**
108. states
109.   s0 : initial complete state;
110. transitions
111.   s0 -[ ]-> s0 {Tout := Tin + Recir * Power; };
112. **};
113. end GCPS.imp;

```

Figure 4.11: GCPS specification of Data center using the AADL language. The specification is for a data center with 50 chassis however only a snippet of the entire specification is shown in the thesis for ease of readability.

the components can access them. An LCPS consists of a chassis which consists of 10 blade servers. Each blade server has computing properties such as type of servers or number of cores and physical properties such as heat dissipation (lines 19-28). The region of impact is discretized into points of interest, which are the inlets of the servers. The LCPS has recirculated heat values for all other LCPSes. Such matrices are specified using the *behavior_annex* (lines 80 - 88). The GCPS consists of several LCPSes. The port group of each LCPS is connected to appropriate port groups of all other LCPSes to feed in recirculated heat values. The dynamic equations governing the temperature rise in the data center are specified using the *CPSAnnex* (lines 69-74 and 107-112).

Safety analysis of data centers

Several management strategies for the data center were simulated using this GCPS modeling. A comprehensive discussion of the algorithms and the results of the comparison are available in the publication by Mukherjee [86]. The algorithms simulated using the GCPS modeling are: i) Cooling Management (CM) - which places a job in the first available server with an arbitrary ordering and then sets the thermostat such that no server crosses redline according to the conditions discussed in Chapter 2; ii) Job and Power Management (JPM), which assumes an Earliest Deadline First (EDF) temporal schedule with a Least Recirculated Heat (LRH) placement algorithm; and iii) Job, Power and Cooling Management (JPCM), which assumes an Earliest Deadline First (EDF) temporal schedule with a Highest Thermostat Setting (HTS) placement algorithm. Figure 4.12 shows that for any of the policies the peak temperature for any server in the data center never exceeds the redline value of 40 °C.

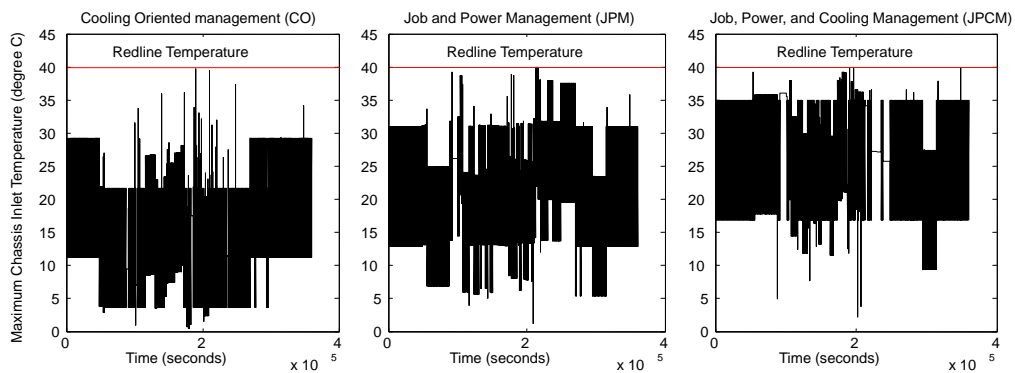


Figure 4.12: Maximum chassis inlet temperature for 80 % utilization under linear cooling model for different energy management policies. Only CM, JPM, and JPCM as representative of non-coordinated cooling, no cooling management, coordinated cooling management, respectively. JPCM allows highest maximum inlet temperatures temperatures (without violating the redline) among all the policies.

Chapter 5

SAFETY ANALYSIS UNDER DYNAMIC CONTEXTS

The operation of a CPS is characterized by dynamic changes in the environment e.g., change in user location or the weather changes affecting energy scavenging or sudden change in workload in a data center. Such changes in context affect the operation of every subcomponent of a CPS. To illustrate this claim let us consider the example of the Ayushman PHMS [5] discussed in Chapter 2 developed at the IMPACT Lab at ASU.

In case of the network controlled infusion pump in the Ayushman PHMS, the controller sends control inputs to infusion pump over the wireless channel to maintain the analgesic drug concentration to a particular level. The controller gets feedback from a pulse oximeter on the human body on the blood oxygen level. The pump should stop infusing immediately when the blood oxygen level falls below a certain level to prevent respiratory distress [55]. Since the wireless channel is prone to errors, the packets containing control information can get corrupted or dropped at random. If control informations do not reach the infusion pump the pump maintains the previous control information for a preset time and then shuts down. Further, if the controller does not obtain an accurate estimation of the blood oxygen level it can cause unstable or oscillatory infusion rates, which is harmful for the user. To this effect, the communication protocol is modified to consider dynamic power control for avoiding packet drop. The design of the technique considered a home environment and modulates radio power to increase the packet delivery rate (PDR) of the medium from 0.6 to an acceptable level of 0.8. However, if the user now moves from home to an outdoor open space such as the balcony, the PDR of the

medium drops drastically [67], and the dynamic power control may not increase the PDR to the acceptable level. If this happens too frequently, a property governed by the user's mobility model, a number of control packets may be dropped, which can lead to overshoot, undershoot and oscillations in the analgesic concentration in the blood as result causing pathological conditions such as respiratory distress. Hence, in such a scenario the user mobility pattern may be unsafe for his health! The operation of a PHMS involves such dynamic context driven interaction between the embedded computing device and the environment including human physiology.

Thus dynamic context changes may affect the safety of a CPS. Hence it is essential to consider the modeling of dynamic contexts for analyzing CPS safety. This chapter focuses on extending the CPS-DAS tool with capabilities to model and analyze dynamic contexts. The running example that is used in this chapter is the Ayushman PHMS.

5.1 Dynamic Contexts

Formally a context is defined as a triple $\{G, M, I\}$ such that G is a set of objects, M is a set of attributes, and I is a bipartite graph mapping between the sets of objects G and attributes M . As an example, let us consider that the user of a PHMS is at home. The set of objects can include the wireless channel, the devices in the PHMS such as infusion pumps, glucose meter, thermometer, or blood pressure sensor, and the human physiological parameters. The attribute set may consist of the packet drop ratio and electromagnetic properties of the wireless channel, sampling frequency of the sensors, and mathematical models of the physiological processes.

A context change can be represented by a change in the object set G , attribute set M , and bipartite mapping I . This change can occur due to several reasons and are mostly random in nature. In this work, three causes of context changes are considered:

- a **Mobility:** An user of a PHMS is always in a state of motion in her day to day life. This leads to change in environmental properties such as temperature and humidity, or wireless channel properties such as the packet delivery ratio (PDR) of indoor and outdoor environments. This change is exhibited by a change in the mapping I where the same set of objects, humidity, temperature, PDR are mapped to different values.
- b **Physiology:** Random physiological events such as epileptic seizure can cause changes in the PHMS or the operation of the PHMS. It can introduce new medical devices such as an EEG sensor, or it can cause execution of a new algorithm for analyzing epilepsy. This is exhibited through a change in the object set G due to the introduction of the new sensor and subsequently a change in the set I .
- c **User activities:** Random user activities such as exercise or food intake can cause changes in the PHMS. For example, during exercise the energy scavenged from the scavenging source may be sufficient for sustaining the operation of the medical devices. This is exhibited in a change in the attribute set M of the PHMS, in specific the scavenged energy attribute of the source.

The causes of the context changes are random in nature and hence are to be modeled using random processes. For example, the mobility of human users are generally modeled using random processes such as Random way point model or the Levy walk model.

Representation of context

Mathematically contexts can be represented as finite state machines, called the *ContextFSM*. The object set G , the attribute set M , and a unique mapping I forms a state in the *ContextFSM*. Any change in G , M , or I causes a state transition. The state transitions are governed by the context changing events. The events can be

represented by a random process that outputs a random sequence of 1s and 0s, where 1s indicate a state transition. The parameters of the random process are dependent on the user's mobility, physiology and activity patterns. As discussed further in the text these random models are not considered for analyzing the safety of the PHMS in current literature. This thesis provides a methodology to consider the random context changes in the analysis in a tractable manner.

5.2 Effects of Context changes

Mobility is a basic human nature. Due to this mobility of human beings, its environment changes continuously. Associated with this are several other contextual factors such as physiological condition, mood, and time of the day. Each context affects the human body parameters in different ways. For example, during hot and humid summers the body sweats leading to a lower average skin temperature, or when a person is excited the skin conductance increases. These parameters affect the way a medical device interacts with the human. As an example, consider a wearable computer controlled infusion pump on a patient as shown in the Figure 5.1.

Infusion Pump Control System: The infusion pump is a medical electrical equipment that obtains commands from a remote computer or a smart phone over the wireless channel and accordingly injects a dose of drug such as insulin or anaesthetic into the human body. The controller obtains feedback from the human body using sensors such as a glucometer and according to a control algorithm computes the future infusion rate and sends it to the pump. In the literature, the feedback has been modeled by pharmacokinetic diffusion equations, which take the infusion rate as input and outputs the drug concentration in the blood. The controller then follows a predetermined algorithm to calculate the infusion rate so that the drug concentration reaches a prescribed value without overshooting a safety limit. Algorithm

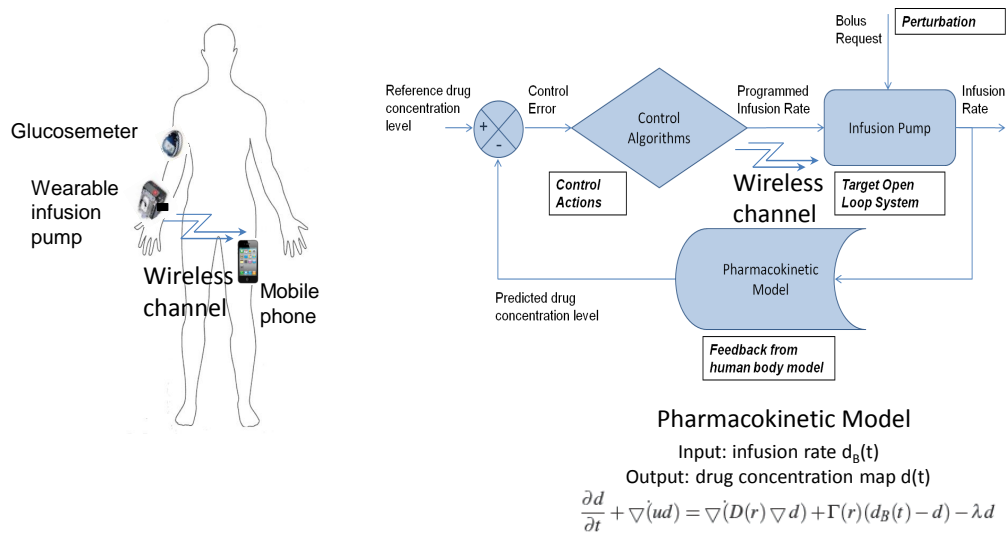


Figure 5.1: A computer controlled wearable infusion pump system.

Algorithm 5.1: Discrete Time Infusion Control (Desired Drug Level (C_{p_d}), Initial Infusion (x_0), Increment Step (δx), Time Step (δt))

- 1: NoOfSteps = Total Time / δt
- 2: **for** i **do** = 1 to NoOfSteps
- 3: Infusion rate $x_i = x_{i-1}$
- 4: Predicted Present Plasma Drug Level C_{p_p} = Simulation of PKA with infusion rate x_i
- 5: Increment Infusion Rate $x_{iemp} = x_i + \delta x$
- 6: Predict Future Plasma Drug Level C_{p_f} = Simulation of PKA with infusion rate x_{iemp}
- 7: Assume linearity of the PKA model, slope = $\frac{C_{p_f} - C_{p_p}}{x_{iemp} - x_i}$
- 8: y - intercept $C_{p_0} = C_{p_p} - \text{slope} \times x_i$
- 9: **if then** $C_{p_0} \geq C_{p_d}$
- 10: New Infusion Rate $x_{i+1} = \frac{C_{p_0} - C_{p_d}}{\text{slope}}$
- 11: **else**
- 12: New Infusion Rate $x_{i+1} = x_{iemp}$
- 13: **end if**
- 14: **end for**

5.1 [55] is one such algorithm that the infusion controller follows to derive the future infusion rate from the past history of drug concentration in the blood. The controller in this case increments the infusion rate by a small amount δx and estimates the drug concentration using the pharmacokinetic model. If the estimated drug concentration is greater than the desired level, the controller reduces infusion rate by linearly

approximating the model such that the estimated drug concentration remains below the desired level.

An important factor in this infusion pump device is the transfer of information from the controller to the pump through the wireless channel. Wireless channels are prone to errors leading to loss of control information. When the pump fails to receive a control information the pump maintains the infusion rate obtained in the last successfully received information. Mobility affects the wireless channel characteristics leading to time varying packet delivery ratio (PDR), which may affect the drug concentration due to faulty infusion. These effects further, vary with different mobility patterns. Hence to characterize these effects different models of mobility have to be studied.

Models of Mobility: Over the years several models of mobility for human has been studied including the random walk and Brownian motion models. The most popularly used mobility model is the random walk model. However, recently, the Levy walk mobility model was found to fit the average human mobility the best [87]. A mobility model consists of three parameters: a) flight length, which is the distance traveled, b) flight direction, direction of the movement, and c) pause time, time for which the person stays at a particular position. The random walk mobility model assumes that the probability that the flight length is greater than a certain value follows a gaussian distribution. This says that it is less probable for a person to move further away from a given spatial location. However, a recent research has shown that the flight length follows a power law distribution in specific levy distribution. This comes from the ever inquisitive nature of human being, which compels her to explore remote regions. Instances of the two mobility models are shown in Figure 5.2, where in random walk the shorter flight lengths are more frequent, while levy walk model has more frequent longer flight lengths.

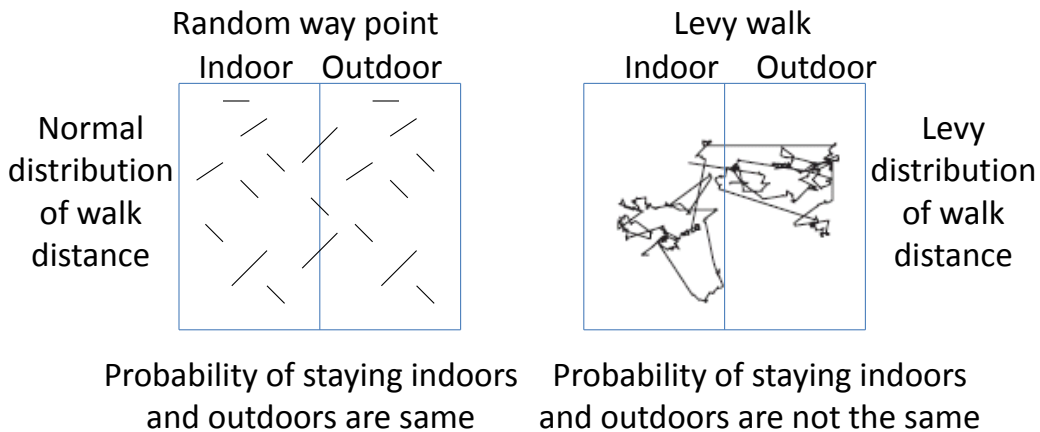


Figure 5.2: Random way point and Levy walk mobility models used as different models for the same context. Levi walk model captures the human nature of restricting their movements to specific regions.

Effect of mobility patterns on infusion pump operation: The infusion pump control system was simulated under different mobility patterns of the user. Two different channel properties were considered: a) indoor, with a PDR of 0.8 and b) outdoor, with a PDR of 0.4 as suggested in [88]. A stretch of 20 feet was considered with a door separating indoor and outdoor environment at the 10 feet mark and computed the sequence of indoor and outdoor movements for random and Levy walk models. Packet drops were simulated using the Ricean fading model and the average case drug concentration for 1000 runs for both the mobility models is shown in Figure 5.3a). The figure shows that since the Levy walk mobility pattern has more frequent outdoor visits, it causes more loss of control information and hence causes drug overshoots due to faulty infusion. On the other hand random walk has shorter excursions leading less frequent change of environment and hence less overshoot. Further, apart from the frequency of outdoor visits, different sequences of indoor to outdoor transitions also affect the drug concentration as shown in Figure 5.3b for Levy walk model.

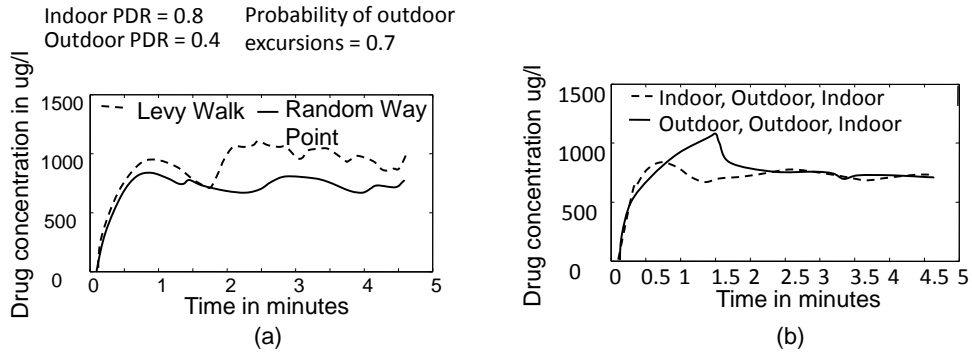


Figure 5.3: a) Drug concentration have different overshoots for different mobility patterns, b) drug concentration profile may depend on the sequence of context changes

This establishes the importance of considering dynamic context changes in analyzing PHMSes. Before going into the details of our analysis methodology, let us first look into the literature for similar research endeavors and distinguish our approach from others.

5.3 Related Works

Model based engineering is a common development paradigm in the medical device domain. But only recently there have been efforts to model and analyze the human body along with the medical device as a closed loop system. This advancement is mainly due to the classification of medical devices as mission critical cyber physical systems (CPSes), where computing systems operate in close loop with their physical environment to accomplish mission critical objectives, such as drug infusion control, precise chemotherapy. Hence, in the literature there are several tools, frameworks, and methodologies, that are available for medical device design or CPS design in general. These works are classified into five categories as shown in the Table 5.1.

Hardware or Software Modeling: Several tools exists to model, analyze and simulate the hardware and software of medical devices. Some of them model the hardware and electric circuitry of the medical device while others model the system soft-

Table 5.1: Classification of existing work on model based safety analysis

Class	Approaches	Device Modeling	Physical Modeling	Interaction Modeling	Domain	Dynamic Contexts
Hardware/Software Modeling	PSpice [78], VHDL and Verilog [89], AADL(http://www.aadl.info/), UML [90], ANDES [82], Hardware Testbeds [91,92]	✓	✗	✗	Multiple	✗
Physical Modeling	Matlab and Simulink [93], SysML [94], Flovent [95], TrueTime [96], Modelica [97], Theoretical modeling of human body parts such as heart models [98], diffusion models [99]	✗	✓	✗	Multiple	✗
Interfacing tools	BAND-AiDe [2] (AADL + Matlab), LabView + Ptolemy [100], Matlab + DeterLab [101], Multi-domain modeling using architectural views [102]	✓	✓	Limited capability	Mostly Single physical domain with the exception of Ptolemy [100] and multi-domain modeling [102]	✗
Integrated Modeling	Modelica [103], BAND-AiDe [2], Hybrid Quartz [104]	✓	✓	✓	Multiple domain but often limited in representing computing systems	✗
Formal Modeling	Timed Automata [33,98,105], Petrinets [106], Hybrid Systems [77,97,100]	✓	✓	✓	Multiple	✗

ware. The main drawbacks of these tools are that they totally ignore the modeling of the human body and the cyber-physical interactions. Further, they do not support context modeling and can only do simulations for a given fixed configuration.

Physical Modeling: Models of the human physiological processes and organs has been considerably researched in the past with efficient solutions. However, existing tools do not support modeling the cooperation of the computing and physical environment.

Interfacing tools: To model the interaction, researchers have proposed interfacing tools from computing and physical domains. The interaction between these tools are typically coarse grain where, simulation results from one tool is used by the other to perform its own simulations. Such discretization of inter-communication between tools lead to errors in estimation. Further, with random contexts it is difficult to predict context changes, which can occur in the middle of a simulation. Hence, such deterministic scheduling of communication may cause difficulties in modeling contexts. Event based communication between tools on the other hand causes an increase in simulation time during frequent context changes. Hence, for requirements analysis of PHMSes a single tool is necessary for simulating computing functions, physical processes, and random contexts.

Integrated modeling: To simulate the computing and physical processes of a PHMS researchers have proposed integrated modeling tools, where both the aspects are modeled and analyzed in a single framework. However, such tools are typically used on specific PHMS test cases, and generally involve extreme cases of operation. To model effects of context changes using such tools, a large number of test cases will have to be simulated in sequence. Further, since context changes involve changes in PHMS configurations, automation of simulation may not be possible leading to inaccuracies and increased time. Hence, tools are required that

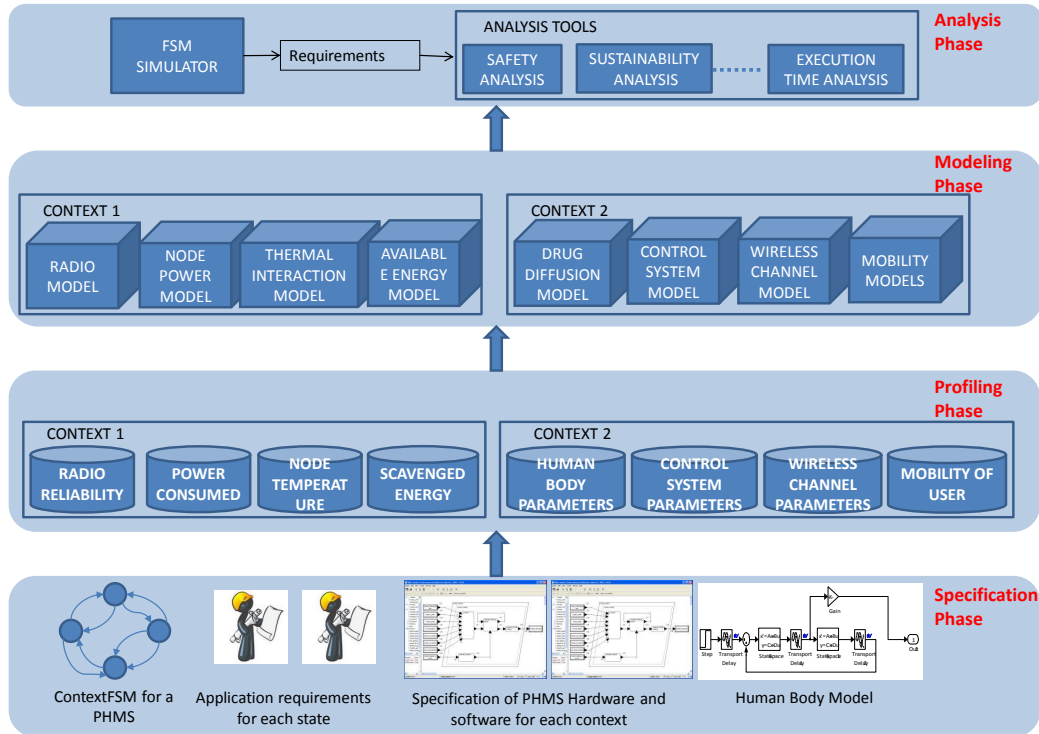


Figure 5.4: Approach for analysis of SMCS under dynamic context changes.

inherently support context modeling and dynamic changes in PHMS models.

Formal modeling: Formal models are used extensively for mathematically characterizing the computing and physical processes. Hybrid systems are used to specify PHMSes in a single mathematical framework. However, most attempts have been restricted to modeling stand alone medical devices. Further, there is no theory for analyzing random contexts in a formal framework. This work proposes an approach to formally specify contexts using a *ContextFSM* and describes a simulation methodology for random context changes. The different stages of the model based analysis as shown in the Figure 5.4 are explained next using the Ayushman PHMS discussed in Chapter 2 as the example.

5.4 Specification Phase

The specification of a PHMS is done using the industry standard AADL language (www.aadl.info). AADL is a hierarchical model specification tool that provides constructs dedicated to modeling embedded software and hardware. However, AADL inherently does not support specification of context and context transitions and physical dynamics of the human body. The behavioral annex is used to specify context as states and context changes as a finite state automata, *ContextFSM*. Further, AADL is extended to incorporate specification of complex physical processes as a series of differential equations through the development of annex (language extensions). The AADL specifications shown in this thesis denotes the AADL specific constructs in bold.

A PHMS specification has five subcomponents:

PHMS specification: It has three different contexts and their dynamic transition logic, AADL Spec 1 (*PervasiveHealthMonitoringSystem*). For each context, there is a different hardware and software implementation indicated by the subcomponents in the AADL Spec 1, and are specified using the **system implementation** construct. Each context is specified using the **mode** construct, and the context transitions are specified as shown in the AADL Spec 1. The transitions take place on occurrence of events, which are specified in the **features** section of the specification. The events are generated from context sensors specified using the *ContextSensor* **system** component. In this component, mobility models can be specified, which can be later used to generate random events that cause state transitions. In this particular example, there are *four* states - home, roaming, hospital, and inactive, and six events - RoamingActive, AtHome, Emergency, Mitigate, Activate and DeActivate. In each context, the PHMS consists of context sensor nodes, energy source, the hu-

AADL Spec 1: PHMS context specification

```
system PervasiveHealthMonitoringSystem
  features
  RoamingActive: in out event port;
  AtHome: in out event port;
  DeActivate: in out event port;
  Activate: in out event port;
  Emergency: in out event port;
  Mitigation: in out event port;
end PervasiveHealthMonitoringSystem;

system ContextSensor
  subcomponents
  process RandomWayPointModel
end ContextSensor;

system implementation PervasiveHealthMonitoringSystem.imp
  subcomponents
  P0: system PervasiveHealthMonitoringSystem;
  P1: system PHMS.HomeContext in modes Home;
  P2: system PHMS.OutdoorContext in modes Outdoor;
  P3: system PHMS.HospitalContext in modes Hospital;
  R1: system ContextSensor.RoamingActive;
  H1: system ContextSensor.AtHome;
  E1: system ContextSensor.Emergency;
  M1: system ContextSensor.Mitigation;
  UA: system UserInput.Activate;
  UA: system UserInput.DeActivate;
  connections
  event port R1.SensorOutput → P0.RoamingActive;
  event port H1.SensorOutput → P0.AtHome;
  event port E1.SensorOutput → P0.Emergency;
  event port M1.SensorOutput → P0.Mitigation;
  event port UA.UserEvent → P0.Activate;
  event port UD.SensorOutput → P0.DeActive;
  modes
  Home: initial mode;
  Roaming: mode;
  Inactive: mode;
  Hospital: mode;
  Home: -[ P0.RoamingActive ] → Roaming;
  Roaming: -[ P0.AtHome ] → Home;
  Home: -[ P0.DeActivate ] → Inactive;
  Roaming: -[ P0.DeActivate ] → Inactive;
  Inactive: -[ P0.Activate ] → Home;
  Home: -[ P0.Emergency ] → Hospital;
  Roaming: -[ P0.Emergency ] → Hospital;
  Hospital: -[ P0.Mitigate ] → Home;
end PervasiveHealthMonitoringSystem.imp;
```


man body specification, and specification of the coordination between the devices and the human body.

Context sensor node: The context sensor node specification, AADL Spec 2 (*PHMSnode.ContextSensorNode1*), has the processor, application algorithm, and the radio. The processor (*Processor.Atom*) power for different sleep states and the idle power consumption at various operating frequencies are obtained from the experiments performed in the profiling phase. The busy power consumption of the processor for different operating frequencies are dependent on the application threads, modeled in the *Application.Ayushman* subcomponent. The model of the application workload consists of three main threads: 1) Sensing, 2) Data Transmission and 3) PKA Execution, each of which models the power consumption and thread execution time for different frequency of operation of the processor. The *Radio.CC2420* subcomponent models the power consumption of the radio for three different modes of operation: 1) Radio transmission, 2) Reception and 3) Radio turned off. The model based communication algorithm can be specified as a state machine using the **mode** construct, where there will be three modes: 1) model match, when the model matching thread will be executed, 2) feature update, when the thread will compute features from the signal and send it back to the base station, and 3) raw signal update, when the sensor will just blindly send sample by sample data. For each mode the power consumption and data sent properties will capture the compression obtained for each mode. The wireless channel properties for the radio can be specified using the properties construct, which can change for different modes.

Energy scavenging source: The total scavenged energy can be modeled as a property of the system implementation,

Human body specification: The modeling of the human body is complex and AADL is not geared towards it. Property constructs can be used to specify the physical

properties (thermal conductance, specific heat, blood perfusion as shown in AADL Spec 3). However, the complex physical dynamics of human body that controls its thermal behavior requires specification of differential equations in AADL which it does not support. In this regard, an annex (*CPS_annex*) was developed which extends AADL to incorporate specification of differential (both partial and total) equations in the model itself. Specific constructs for denoting differentials were developed in the annex. The annex implementation is described in detail in Section 5.7.

Specification of coordinated operation: The coordinated operation results in changes in the complex physical processes with events occurring in the computing domain. In the infusion pump example, the diffusion of drug is governed by the pharmacokinetic (PCK) process [55], which can be modeled as a set of differential equations.

However, the equations change with the change in state of the controller. The controller algorithm takes the drug concentration predicted by the PCK process as input and varies the infusion rate to keep the drug concentration at a given level. Such an algorithm can be represented using a state machine, which captures both the computing and physical behavior of the infusion pump. A hybrid automata can be used in this regard for to capture the continuous physical dynamics in each state. However, the **mode** construct cannot be used since there is no provision to specify equations for a given state and transitions cannot depend on the variation of a system variable. Instead a combination of the *behavior_annex* and the *CPS_annex* is used in AADL to specify the control algorithm as a hybrid automata, AADL Spec 4 (*NetworkControlledDevice.InfusionPump*). As shown in the specification, the partial differential equations can be specified using *CPS_annex*, *PDE1* and *PDE2* and associate them with states *s1* and *s2* in the *behavior_annex*. Further, events in *behavior_annex* can occur when a variable crosses threshold (*Overshoot* event).

AADL Spec 2: PHMS sensor node specification

```
system implementation PHMSnode.ContextSensorNode1
  subcomponents
    S1: system Processor.Atom;
    S2: system Radio.CC2420;
    A1: process Application.Ayushman;
    ...
end PHMSnode.sensorNode1;

system implementation Processor.Atom
  modes
    SleepState1 : mode;
    Frequency1 : mode;
    ...
  properties
    IdlePower => 0.160 W in modes SleepState1;
    Temperature => 43 °C in modes Frequency1 ;
    ...
end Processor.Atom

system implementation Radio.CC2420
  modes
    RadioOnTx : mode;
    ...
  properties
    Power => 0.058 W in modes RadioOnTx;
    ...
end Radio.CC2420

process implementation Application.Ayushman
  subcomponents
    T1: thread Sensing;
    T2: thread PKA;
    T2: thread Transmission;
end Application.Ayushman

thread implementation Sensing
  modes
    Frequency1 : mode;
    ...
  properties
    Power => 0.030 W in modes Frequency1;
    ExecutionTime => 5 s in modes Frequency1;
    ...
end Sensing
```

AADL Spec 3: Human Body Specification

```
system implementation HumanBody.skin
properties
  SpecificHeat => 1.6 J/(Kg.K);...
annex
  Dell<Temperature><Time> = K(Pdel2<Temperature><x>+
  Pdel2<Temperature><y> + Pdel2<Temperature><z>) + ...
end HumanBody.skin;
```

AADL Spec 4: Infusion Control Algorithm Specification

```
system implementation NetworkControlledDevice.InfusionPump
properties
  DiffusionCoefficient (D) => 1.6 J/(Kg.K);
  ...
annex CPS_annex {**
  PDE1: Dell<DrugConc><Time> =
  D(Pdel2<DrugConc><x>+Pdel2<DrugConc><y>) + InfusionRate + ...
  PDE2: Dell<DrugConc2><Time> = D(Pdel2<DrugConc2><x>
  +Pdel2<DrugConc2><y>) + ...
**}
annex behavior_specification {**
states
  s0: initial state;
  s1, s2, s3: state;
transitions
  s0: -[ Bolus ?] → s1 {PDE1;};
  s1: -[ OverShoot ? {PDE1;}] → s2 {PDE2;}; **}
end NetworkControlledDevice.InfusionPump;
```

5.5 Profiling Phase

The available energy profiles of the scavenging sources are already obtained from [1] while the profiling of human body for its thermal properties are obtained from previous literature. Thus, the power and thermal profiles of the PHMS node are derived under the Ayushman workload.

Power Profiling

For power measurement of the Atom processor the Mobile Intel 945 GSE (GMCH) chipset is used. The power measurement setup provides the board power consumption, which includes the CPU power as well as power for driving the chipset

Table 5.2: Power profiling of sensors (TelosB, iMote, BSN node v3, shimmer)

Tasks	Consumed Power (mW)	Execution Time (ms)
Statistics operation (mean, standard deviation)	5, 162, 6.7, 6.73	230, 220, 207, 200
Fast Fourier Transform	5.1, 162, 6.5, 6.66	435, 102, 425, 415
Peak Detection	5.6, 156.6, 6.8, 6.6	100, 160, 90, 88

Table 5.3: Atom Power Usage for PKA computation in Ayushman

Percent Throttling	Power Consumption W
0	0.191
13	0.1864
25, 37	0.17
50, 62, 75, 87	0.167

and other associated components. In order to isolate the CPU power consumption during Ayushman execution, the idle power of the board is first measured for each throttling mode by allowing the CPU to run idle for three minutes (for stabilizing the watt meter). Then the Ayushman workload is executed to measure the average platform power. The difference between the two power values gives the power consumed by the processor during the execution of the workload, which is shown in Table 5.3 for different throttling modes.

The power consumption of the TelosB motes were experimentally obtained by running the BSNBench benchmarking suite [107]. The benchmarking suite has specific tasks for obtaining power consumption due to computation, sensing, and communication. The sensing and computation power consumption is listed in Table 5.2 for benchmark signal processing applications such as Fourier transform, and peak detection. The power consumption of the Chipcon radio was measured

Table 5.4: Thermal Profile Data for different frequency throttling modes of the Intel Atom N270 processor

Operating mode (Percent Throttling)	Atom Processor Operating Temperature °C	Maximum Skin Temperature °C after 24 Hr of operation
0, 13, 25	43	39.4365
37, 50	42	39.3325
62, 75	41	39.2295
87	39	39.0264

during transmitting packets at a bit rate of 250 kbps, standard for a sensor node (www.xbow.com). The current consumption of the CC2420 radio used in the PHMS was measured to be 18.41 mA during transmission and 19.20 mA during reception. Considering the operating voltage to be 3 V the maximum power consumption of the radio is 58 mW.

Thermal Profiling

The thermal effects of the TelosB sensors are negligible and hence are not considered in the analysis. However, the Intel atom based smart phone dissipates a greater amount of power and hence can cause thermal harm to the human skin. The temperature of the Intel Atom processor is measured using the on board thermal sensor of the Mobile Intel 945 GSE development platform (GMCH). The N270 processor has dedicated MSR registers (<http://download.intel.com/design/processor/datashts/320032.pdf>), which stores the digital thermometer reading. The Atom processor was first kept at sleep state (C6) to allow for the core temperature to stabilize to a low value. Then the processor was brought back to normal state and the Ayushman application was run at different throttling modes of the processor. Table 5.4 shows the peak operating temperature attained by the Atom processor during the execution of the Ayushman application at the different processor throttling modes.

5.6 Modeling Phase

From the AADL specification and using the profiling results, abstract models of the different components of the PHMS can be built as discussed in this section.

Power model of PHMS

Power consumption of hardware components (processor, radio) for execution of different threads (*Thread_power*) and their timings (*Thread_Time*) are extracted from the AADL meta-data. It is contended that during the period of sensing $t_s = 5$ secs, the micro controller is in idle state, where it consumes P_{idle} amount of power (≈ 1 mW in TelosB motes). For a PHMS with n nodes the sensing process can be performed in parallel by all sensors. After each sensing period the sensed data is transferred to the base station. During this transmission period t_{Tx} the processor is in idle state, consuming P_{idle} amount of power (approximately for 0.39 secs to transmit five seconds of 32 bit data values 60 Hz sampling rate and a transfer rate of 24 Kbps [85]). The radio transmitter will also be active during this period ($P_{radio} \approx 58$ mW being its power consumption). In a 24 hr period there will be x number of sense and transmit periods (sleep cycles) for each sensor in the PHMS, with a duration of $(t_s + t_{Tx})$ secs each. Further, in a single day of operation of Ayushman the PHMS nodes under go pairwise PKA execution to maintain the freshness of the encryption key among two nodes. During this execution of PKA the processor should be in active state consuming P_{PKA} amount of power for the duration of execution of the PKA algorithm t_{PKA} . The value of P_{PKA} is around 10 mW and t_{PKA} is around 1 sec as obtained from actual measurements averaged over all the commercially available platforms. Further, during the transfer of the vault ($t_{Vault} = 6.75$ secs [85]), the radio is active. Thus total energy consumption is given by Equation 5.1.

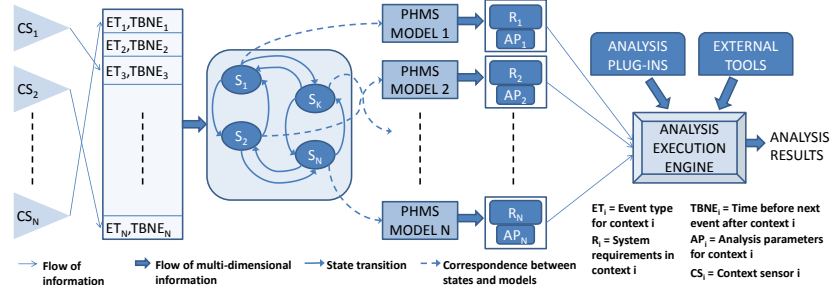


Figure 5.5: PHMS analysis methodology is shown in the Figure. The context sensors post events in the event queue, the events cause state transitions in the *ContextFSM*. Each state has a different configuration of PHMS, which are analyzed using the analysis engine.

Total PHMS Energy = Sensing Energy + Data Transmission Energy + PKA computation energy
 + vault transfer energy

$$\begin{aligned}
 \Rightarrow E_{PHMS} = & nx(t_s)P_{idle} + nxt_{Tx}(P_{radio} + P_{idle}) + t_{PKA}P_{PKA}\left(\frac{n(n-1)}{2}\right) \\
 & + t_{Vault}(P_{idle} + P_{radio})\left(\frac{n(n-1)}{2}\right) \quad (5.1)
 \end{aligned}$$

x is the number of sleep cycle to be sustained in 24 hrs.

Thermal model of PHMS

The operating temperature (*Operating_Temperature*) of the PHMS node, power consumption (*Component_Power*) of each component, human skin properties (*Skin_Properties*) and the distances of the components from the skin (*Skin_Distance*) are obtained from the metadata information after parsing the AADL specification. These parameters are used in the estimation of the temperature rise in the human body parts, which is governed by the following physical phenomenon:

- 1) radiative heat transfer from the Atom processor which depends on the operating temperature of the processor (modeled using the Stefan's Law),
- 2) conductive heat transfer from the processor,
- 3) electro-magnetic radiation absorption by the

body part (modeled by calculating the Specific Absorption Rate (SAR) [81]) and 4) convective heat extraction by blood. These physical processes are combined in a partial differential equation known as Penne's equation which gives the temperature variation of the human body part over space and time (Equation 5.2).

$$\rho C_p \frac{dT}{dt} = K \nabla^2 T - b(T - T_b) + \rho \text{SAR} + P_c \quad (5.2)$$

where ρ is the mass density, C_p is the specific heat, K is the thermal conductance and T is the temperature of the body part, P_c is the power generated by the processor, b is the blood perfusion constant, T_b is the blood temperature, and SAR is the specific absorption rate of the body part for electromagnetic radiation.

Model of Infusion Control

The control algorithm of the pump considers an initial infusion rate of x_0 . The control algorithm discretizes time and queries the pharmacokinetic model after each discretized step δt for an estimation of the drug concentration. It then either increases the infusion rate by δx or decreases it according to a linear approximation of the diffusion process. The initial infusion rate, time discretization step and the infusion increment step are the variables of the control system which can be tuned to obtain different behaviors. Further, the infusion pump can get random Bolus requests (a step rise in infusion rate). The magnitude of the bolus is also a variable of the system. The pharmacokinetic model expresses the drug diffusion process as a set of multi-variable linear differential equations in state space form (Equation 5.3).

$$\dot{y}_1 = A_1 y_1 + B_1 z_2 + E_1 u(t - T_i), z_1 = C_1 y_1(t - T_p) \quad (5.3)$$

$$\dot{y}_2 = A_2 y_2 + B_2 z_1, z_2 = C_2 y_2(t - T_r)$$

Here y_1 and y_2 are the state space variables of the equation. $A_1, A_2, B_1, B_2, C_1, C_2$ and E_1 are constants. z_1 is the drug concentration in the blood while z_2 is the drug

concentration of the tissue. The initial infusion rate $u = x_0$ is the input to the model and the output is the drug concentration in the blood. The differential equations in the model are time-variant. This is because they consider time delays related to the infusion input (T_i), cardio-pulmonary transport delay T_p and the arterial, capillary and venous transport delays T_r . The time-variant nature of the physical process comes from the consideration of the transport delays.

5.7 PHMS Analysis

The AADL specification of the PHMS is first parsed to hierarchical XML based metadata. This XML metadata is now used to extract information for analysis. The methodology for analyzing a PHMS for requirements verification under the dynamically changing environment is shown in Figure 5.5. The first step in the analysis procedure is to generate context transition test cases. In this step, a random sequence of events are generated according to context models such as mobility models, arrhythmia occurrence probability, and bolus request frequency. These events are classified into event types (ET_i) and are appended with an estimate of the time before next event ($TBNE_i$) and arranged into an event queue. The *ContextFSM* is then simulated starting from the initial state in accordance with the events. In each state, the context specific PHMS is parsed to obtain the requirements and analysis parameters. Depending upon the requirements different analysis plug-ins are employed to perform the simulation of the PHMS model. Further, domain specific tools such as Matlab can also be used to analyze the PHMS model. The execution of the appropriate plug-in for the correct analysis parameters and checking the compliance with the requirements is performed by the analysis execution unit (Figure 5.5). The output of the analysis execution engine is the compliance results. This analysis methodology is developed in the OSATE simulation environment, a JAVA based AADL specific software development platform.

To specify the human body dynamics an AADL annex called *CPS_annex* was developed as discussed in Section 5.4. The *CPS_annex* parses AADL model parameters according to a specified grammar, which can be used to specify complex equations including partial differential equations in the AADL model itself. Specific constructs were declared for denoting differential operators and the following grammar is proposed which can be used to parse the specified equation in a suitable format.

```

Expression := SubExpression { (+ | -)
                        SubExpression}*
SubExpression := Term { (* | /)Term}*
Term := DerivativeExpression | PortIdentifier
      | AADL Property
DerivativeExpression :=
Del<DerivativeOrder><DependentVariable>
      <IndependentVariable> |
Pdel<DerivativeOrder><DependentVariable>
      {<IndependentVariable>}+

```

A differential equation is denoted by an *Expression*. *Expressions* consist of one or more *SubExpressions* which are combined either by addition (+) or subtraction (-) (**SubExpression*** denotes zero or more *SubExpressions*). Each *SubExpression* further can consist of one or more *Terms* which are combined by division (/) or multiplication (*) operations. Each term can either be a derivative operator (denoted by the term *DerivativeExpression*) or a constant value obtained from a port communication (*PortIdentifier*) or an AADL property. Two types of derivatives are considered in this grammar: 1) absolute derivatives with respect to a single inde-

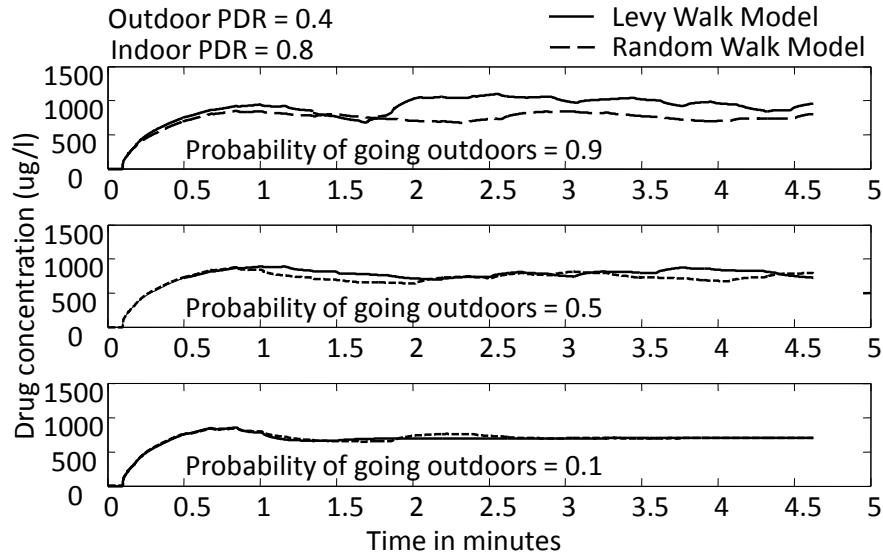


Figure 5.6: Analysis of infusion pump safety under dynamic context changes. The levy walk model predicts higher concentration of insulin than the random walk model, the difference increases with higher probabilities of outdoor excursions.

pendent variable and 2) partial derivatives with respect to multiple variables. For representing the absolute derivatives the *Del* operator is used which has three parts to its specification: 1) derivative order, 2) dependent variable and 3) a single independent variable. Partial derivatives are represented by the *Pdel* operator which has the same structure as *Del* but can have multiple independent variables. This grammar allows ordinary and partial differential equation specification in AADL model itself.

The usage of the analysis framework to analyze the dynamic context driven interactions between the PHMS devices and the human body is considered in the next section.

Effect of context change on medical control systems

The usage of our analysis framework is shown considering the infusion pump example. The PHMS is in a hospital context. However, the patient wants to move around in the hospital and goes to the balcony to enjoy the view outside. This will

trigger a context change in the PHMS and the system state will transit from hospital to outdoor. In such a scenario, specifically the wireless channel properties will change resulting in a different packet delivery ratio (PDR) for the radio communication. Since the infusion pump is controlled through the wireless channel by the controller, change in the PDR may cause a drop in communication quality between the controller and the pump. Low PDR may lead to packet loss from the controller to the infusion pump. This may cause delay or loss of control inputs to the pump. In the analysis framework, two different mobility models, random and Levy walk [87], were used to simulate the context change. The hospital region was divided into two parts: indoor (PDR = 0.8) and outdoor (PDR = 0.4). The contexts were simulated for 10 cases with probability of outdoor visits varying from 0.1 to 0.9. For each sequence of control inputs the control algorithm and the pharmacokinetic model were simulated in coordination. The results of the simulation is shown in Figure 5.6. The results show that a random mobility pattern is less harmful (causes lower overshoots in the average case) than a Levy walk pattern. This is because in the Levy walk pattern the patient is more inclined towards an outdoor visit. However, in random walk pattern the outdoor visits are more uniformly distributed. Such complex simulation of dynamic context changes and its effect on the medical device and human body coordination cannot be performed in contemporary simulation tools and is only facilitated by our methodology.

Context driven safety violation:

This example considers the thermal impact of the Intel Atom based smart phone on the human body. Two contexts are considered: 1) the smart phone is idle, and 2) when it receives data from the ECG sensor and starts online processing of the signal to detect onset of epilepsy [108]. The epilepsy detection algorithm involves the computation of Fast Fourier Transform (FFT) of the signal and peak detection. Once

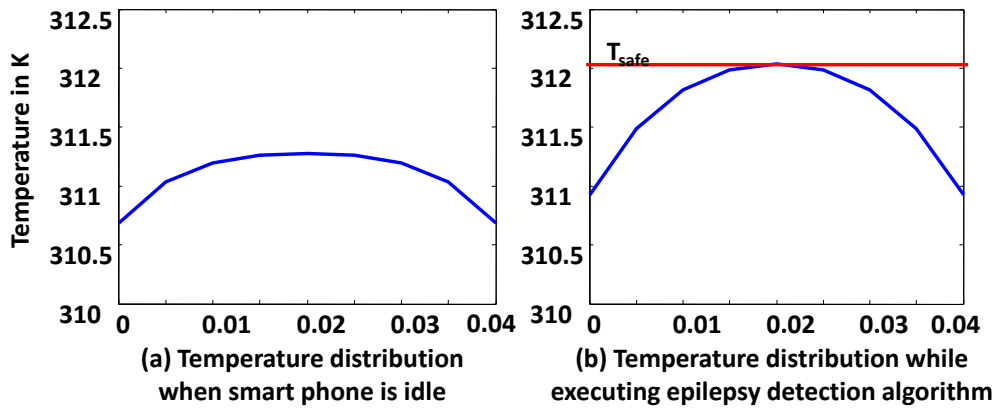


Figure 5.7: Skin thermal map for two modes of smart phone operation. Typically temperatures are higher when the smart phone is executing the epilepsy detection algorithm.

the smart phone transits to the epilepsy detection context it stays in that context for 1000 seconds. The thermal map for the operation of the smart phone is shown in Figure 5.7. The results show that when the smart phone was in the idle state, it was thermally safe. However, continuous execution of the epilepsy detection algorithm is not safe since after 1000 seconds of operation the skin temperature reaches safety limits of 312 K.

FORMAL MODELING AND ANALYSIS OF CPSES

The previous chapters talked about architectural modeling of the CPS and handling dynamic contexts. With architectural modeling, analysis is limited to only simulations. Theoretical guarantees on the safety of CPS can only be obtained with a more formal representation of CPS with semantics that represent the evolution of the cyber-physical interaction between the computing unit and the physical system over both space and time. This chapter, first reviews existing formal modeling techniques for applicability to CPSES. It then discusses a novel formal model, Spatio-Temporal Hybrid Automata (STHA) and develops its approximate reachability analysis technique. The reachability analysis outputs a comprehensive set of states, or possible values of the system parameters, that can occur within a given time and space bound.

Formal models for CPSES should thus capture four salient features: 1) discrete time behavior of the computing nodes; 2) continuous dynamics of the physical environment; 3) spatio-temporal variation of the continuous physical parameters; and 4) aggregate effects because of concurrent operations of networked computing nodes.

Traditionally, hybrid automata [42, 43] are used for capturing both discrete and continuous behavior of a system. However, current tools to model a hybrid automata [77] only consider one dimensional variation of parameters, generally over time, and are hence insufficient for modeling the spatio-temporal variation of physical parameters in CPSES. Safety analysis of hybrid automata generally involve reachability study of the state trajectories to analyze whether the designated sets of unsafe

states are reached with progression in time. However, for CPSes reachability study should evaluate state trajectories in both time and space. To this effect, researchers have considered spatial network of hybrid automata to capture the spatial propagation of physical parameters across the human body [44–47].

Spatial networks of hybrid automata discretize the space as a grid and statically allocate hybrid systems at specific points. Thus, they depend on fixed spatial boundaries to setup the network. However, most of physical dynamics in human body are *Free Boundary Problems* [109] where the boundary conditions in space causes the spatial extent of interactions to expand. For example, the drug concentration due to infusion at a given time is maximum at the site of infusion and gradually decreases as we move away from the site. A point in space, which has a negligible concentration at time t_1 , can have a considerable concentration at a later time t_2 . Thus, a space point, which was not considered in the analysis since it was outside the boundary, will be ignored by a spatial network of hybrid automata, even if at a later time it has considerable concentration. Thus, they also fail to accurately model the aggregate effects due to concurrent operations in networked computing units.

Modeling aggregate effects in CPSes, requires new dynamic equations in the formal model. For example, the coefficients of the equation governing the temperature rise in the human tissue changes when there are multiple heat sources (i.e. the sensor nodes) [2, 81]. Thus, a single formal specification is necessary for the entire network where aggregate effects can be expressed by incorporating new dynamic equations. Hilbertean transforms [110] have also been used to specify CPSes formally. However, to the best of our knowledge, no formal verification of safety has been performed.

Computationally tractable reachability analysis of a hybrid automata is only possible to date for affine dynamics in one dimension [42, 43], since there exists

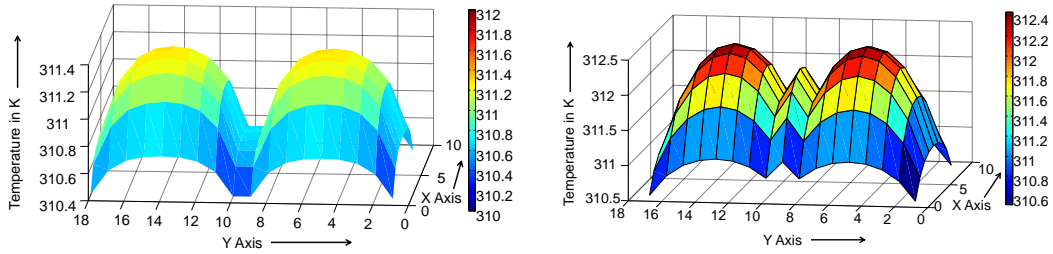
closed form solutions of the differential equations. For the most commonly occurring form of spatio-temporal dynamics, linear second order partial differential equations (PDEs) as observed in Penne's bioheat equation [4] or infusion pump diffusion dynamics [111], there exists no closed form solution even for a single spatial dimension. Hence, traditional method for reachability analysis with zonotopes [112] do not apply. For temporal dynamics whose solution cannot be computed with infinite precision, a time bounded reachability analysis is proposed in [113]. This thesis takes an approach similar to [113] and proposes *a novel bounded time and space reachability analysis for spatio-temporal hybrid automata, where the interactions are represented as linear second order PDEs with one space dimension*. The thesis makes the following fundamental contributions:

1. Defines a *Linear 1-space dimension Spatio-Temporal Hybrid Automata (L1STHA)* capturing the aggregate spatio-temporal dynamics of cyber-physical interactions in CPSes;
2. Develops a *bounded time and space reachability analysis of L1STHA models* enabling safety verification of CPSes based on their spatio-temporal behavior.
3. Applies the reachability analysis technique to medical CPSes specifically infusion pumps for safety analysis.

In the next section, we discuss some preliminary ideas, an overview of our approach and the infusion pump example that is used in this thesis.

6.1 Motivation and Related Works

The principal source of complexity in modeling a CPS is the continuous interaction of the computing system with the physical environment leading to spatio-temporal variation of the system properties. Figures 6.1 a and b, show the thermal map of the



(a) Thermal map of human skin after 480 seconds of operation. (b) Thermal map of human skin after 18720 seconds of operation.

Figure 6.1: Thermal map of human skin for two sensors placed at locations 5 and 15. The sensors were sensing at 60 Hz and were transmitting the sensed value to the base station using ZigBee radio. This workload is typical of a pulse oximeter sensor. The power consumption values are measured from a Smithsoem pulse oximter while the temperature rise is governed by the Penne’s bioheat equation [4]

human skin governed by the Pennes bioheat equation due to heat dissipated from two sensors places close to each other. It can be observed that the temperature rise of the human skin varies over space, indicated by the different intensities of the temperature rise at different spatial locations, and time, indicated by the change in the thermal map between part a and b of the Figure 6.1.

Incapability of traditional hybrid automata: As discussed in the Chapter 1, a hybrid automata can effectively capture both continuous and discrete time behavior of systems [42, 43]. In such a system, the discrete time behavior is modeled as discrete states. Associated with each state is a differential equation that governs the continuous dynamics based on which the state variables (usually the physical parameters) change over time. Transitions among the states occur with occurrences of events normally caused by the continuous dynamics, e.g., state variable values crossing thresholds. These systems can only express continuous dynamics in a single (time) dimension. However, for a CPS the continuous dynamics varies over four dimensions: three dimensions of space and one dimension of time. Hence, single dimensional hybrid automata cannot directly model a CPS behavior.

Drawbacks of spatial networks of hybrid automata: Recent efforts have attempted to discretize the spatial dimensions in a CPS model and have used spatial network of hybrid automata to capture the spatial propagation of physical parameters across the environment [44–47]. However, such efforts have two important drawbacks. Firstly, the discretization of the spatial dimension leads to errors in the modeling phase. The solution of the differential equations for analyzing a hybrid automata involves discretization of the time dimension. If the spatial dimensions are also discretized then the errors due to quantizations are likely to be amplified during the analysis of the hybrid automata.

Secondly, such spatial networks of hybrid automata may not capture the aggregate effects of cyber physical interactions in a CPS. As seen in Figure 6.1 (a), the temperature rise for both the sensors had died down significantly at the spatial locations 9 to 10. However as time progressed the temperature at those locations increased significantly and became comparable to the temperatures at locations 5 and 15 (the locations where sensors were placed). This happened due to aggregate effects of the interaction between the two sensors, which caused a change in the continuous dynamics at the spatial locations 9 and 10. Hence, it can be seen that due to aggregate effects the continuous dynamics in certain spatial regions may change. If a spatial network of hybrid automata were used in such a scenario, the continuous dynamics would have been fixed over time at all spatial locations. Hence, it may fail to capture such aggregate effects.

Model composition and aggregate effects: Hilbertean transforms [110] have also been used to specify CPSs formally. However, no formal verification of safety has been performed on them. Further, they do not consider the aggregate effects of cyber-physical interactions due to network of computing units. Composition of multiple hybrid automata has been studied to synthesize the global behavior of systems

from individual sub-components [114]. Such composition can be useful to capture the effects of the concurrent operations in multiple computing nodes. However, the aggregate effect of the multiple computing nodes may introduce new dynamic equations governing the variation of the system parameters. Therefore, composition of the models has to allow changes in the equations for physical dynamics to capture the aggregate effects which is not supported in these models.

This research proposed to overcome these challenges with a novel form of hybrid automata called Spatio Temporal Hybrid Automata (STHA), which can capture spatio-temporal dynamics in a CPS. Further, a composition relation for STHA is also defined that can capture the aggregate effects of cyber-physical interactions in presence of a network of sensors.

Analysis of CPS models: Safety verification in hybrid automata normally involve reachability study of the state trajectories to analyze if designated sets of unsafe states are reached with progress in time. However, the continuous dynamics of the physical parameters, caused by the cyber-physical interactions, is normally driven by dynamic equations in four-dimensional spatio-temporal domain [4]. As expected, most of the PDE models will not admit analytical solutions. Hence, it is often analytically not possible to bound the system parameters governed by these equations. Therefore, this research focuses on trying to prove bounded time safety guarantees. Towards that goal, existing reachability analysis techniques [115–125] have to be adopted and modified and applied to STHA. The reachability analysis for the one dimensional hybrid automata is also an open area of research given the complexity of solutions of PDEs and only for very specific classes of equations such as linear differential equations, techniques to perform accurate reachability analysis exist. This research also proposes the development of a simulation technique for STHA that can provide time bounded reachability analysis.

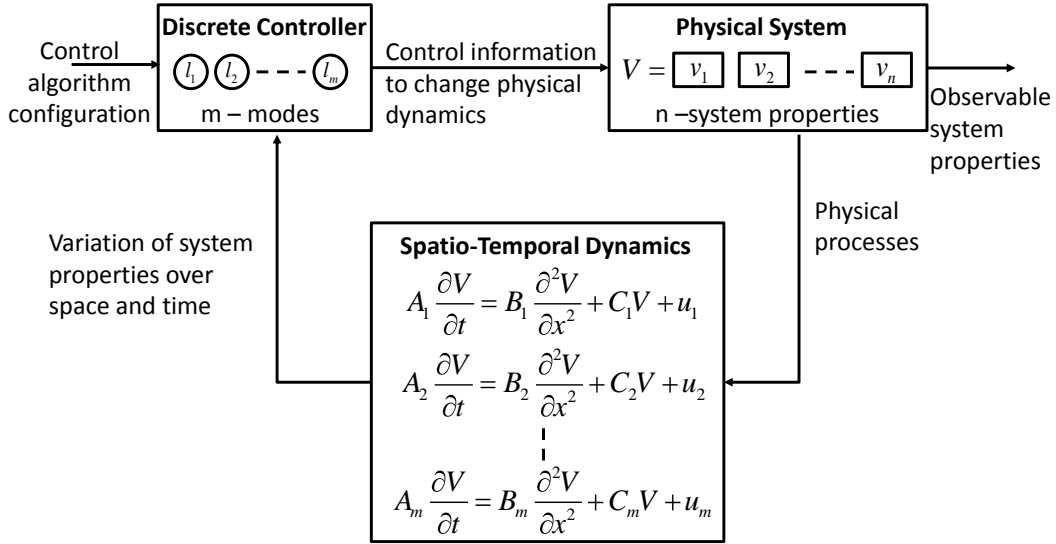


Figure 6.2: Assumed system model for Cyber-Physical Systems.

6.2 Preliminaries and Overview of Approach

Our system model of CPSes consists of a computing unit as a controller of the physical environment (Figure 6.2). The controller takes feedback from the physical environment and makes a decision on the control information, which is sent back to the physical environment. The control algorithm has m modes of operation, (l_1, l_2, \dots, l_m) , each with a different control policy. The physical system can be represented using n continuous variables or *system properties*, which vary according to a second order PDE. In this thesis, we restrict these PDEs to one space dimension x .

There are three notions of state in our model: a) discrete state, b) continuous state, and c) state. A *discrete state* represents the computing modes in a CPS and is similar to the notion of state in a Finite State Automata. Each point in the *continuous state space* (a subset V of \mathbb{R}^n) is a value assumed by the continuous variables of a system and is called a *continuous state*. A *state* is a tuple (l, v) , consisting of a discrete state or mode, l and a continuous state v .

In this theory, we assume that the continuous state space is compact. That is each subset J of the continuous state space has a continuous interior denoted by J° , an open set (e.g., $0 < x < 10$), and a boundary denoted as J^\square given by boundary conditions (e.g. $x = 0, x = 10$). The set $J = J^\circ \cup J^\square$ (e.g. $0 \leq x \leq 10$). The entire continuous state space V , is partitioned into a collection of polyhedral subsets $J = \{J_1, J_2, \dots\}$ such that: a) $\bigcup_{J_i \in J} J_i = V$, and b) $J_i^\square \cap J_j^\square = \emptyset$ for $i \neq j$. Each polyhedron subset is called a *cell* and is a non-empty set of real numbers. Further, two cells are called *adjacent* if $J_i^\square \cap J_j^\square = n - 1$. Two cells are *connected* if there is a sequence of adjacent cells between the two.

Hausdorff distance, $d_H(P, Q)$, between two sets P and Q is the maximum Euclidean distance of any point in P to its corresponding closest point in Q [126].

The main advantage of considering Hausdorff distance is that a δ neighborhood of a point P in \mathbb{R}^n , denoted by $B_\delta(P)$, computed using Hausdorff distance, is a hypercube of side length 2δ . The set of *vertices* of a neighborhood $B_\delta(P)$ is denoted by $Vert(B_\delta(P))$. This is illustrated in Figure 6.3, where the δ Hausdorff neighborhood of a point P in two dimension is a square of side length 2δ . This immensely simplifies the computation of convex hull of a set of points required for reachability analysis [127].

Unit vectors are used to denote direction. A unit vector \hat{n} from a set P to Q is a vector of unit length along the Hausdorff distance from the set P to Q . The dot product of the temporal or spatial variation of any variable v with the unit vector \hat{n} denoted by $\frac{\partial v}{\partial t} \odot \hat{n}$ or $\frac{\partial^2 v}{\partial x^2} \odot \hat{n}$, is the amount of variation in the direction from set P to Q . This notion is useful for capturing the transition between L1STHA modes.

The *image* of any continuous state v , over time t and space x , denoted by $D_{t,x}(v)$ is the computation of the solution of the linear 1-space dimensional second order PDE, for a time t and space x , with v as the initial condition.

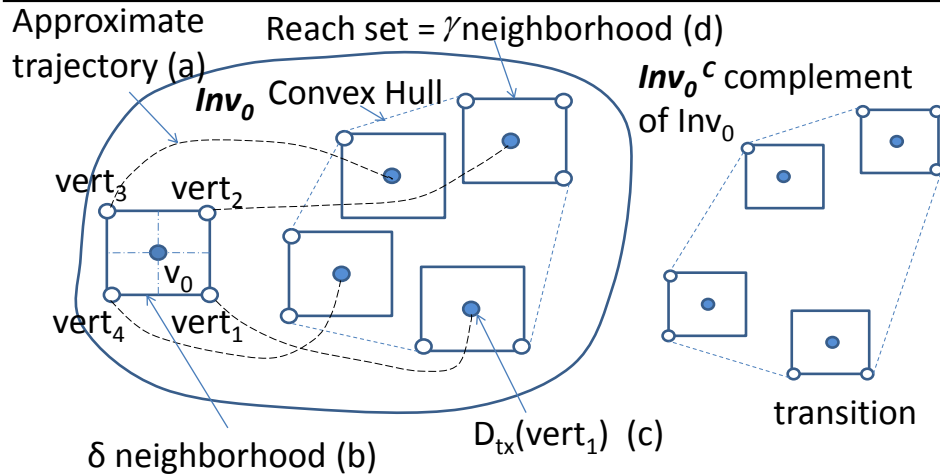
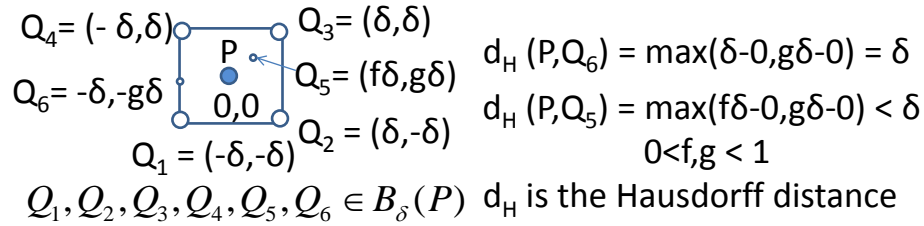


Figure 6.3: Computation of image from an initial state. A boundary of the initial state is assumed and the dynamic equations are evaluated on the vertices of the boundary. The reachable state at the next time or space step is the convex hull of the images of the vertices.

Overview of Approach

The STHA expresses the variation of system parameters of a CPS according to a control logic specified using the discrete states or modes. In our approach, we first define the L1STHA, and then define its execution logic, which formalizes how the L1STHA operates in space and time. The execution is governed by the linear 2nd order PDE.

The reachability analysis of L1STHA is the method of approximating the continuous states that can occur during the execution of the L1STHA in space and time from a given initial state (defined later), or in other words estimating the *reach set*. This involves computing the image of an initial state using the solution of the PDE

(Figure 6.3). To this effect, we first consider a L1STHA with only a single mode and develop the algorithm to estimate its reach set (Section 6.9).

The reach set of a single mode L1STHA is computed using the following steps (marked in Figure 6.3): a) we first find a suitable discretization of time and space that guarantees that the error in computing the image of the PDE is within the desired accuracy ϵ , b) we then consider a δ neighborhood of the initial state v_0 , $B_\delta(v_0)$, c) the image of the vertices of the neighborhood, $Vert(B_\delta(v_0))$, is then computed using the PDE for a given time and space, d) the γ neighborhood of each point in the image is considered, resulting in a new set of vertices, d) the convex hull [127], of the new set of vertices gives the γ approximation of the reach set. As shown later in the formulation the γ and δ are related linearly to ϵ . The same algorithm is applicable for estimating the reach set of multi-mode L1STHA (Section 6.10). The only difference is that in the process of computing the images of vertices the reachability analysis algorithm keeps track of the transitions made by the L1STHA into different modes. For safety analysis, a subset of continuous states is designated as *unsafe*. If the computed reach set intersects the unsafe set, the CPS is deemed unsafe.

Safety in CPSes

Safety of CPSes in the medical domain is defined as the avoidance of unwanted hazards to the human body due to the cyber-physical interactions. Such a definition is generic and is applicable to CPSes in general. Hazards can be of several types as listed in ISO 60601 standard for medical electrical equipments. In this thesis, we consider physiological hazards due to drug overdose in infusion pumps.

Example 12 *Drug infusion:* *Infusion pumps operate in a close loop with a networked controller to keep the drug concentration in the human blood within recom-*

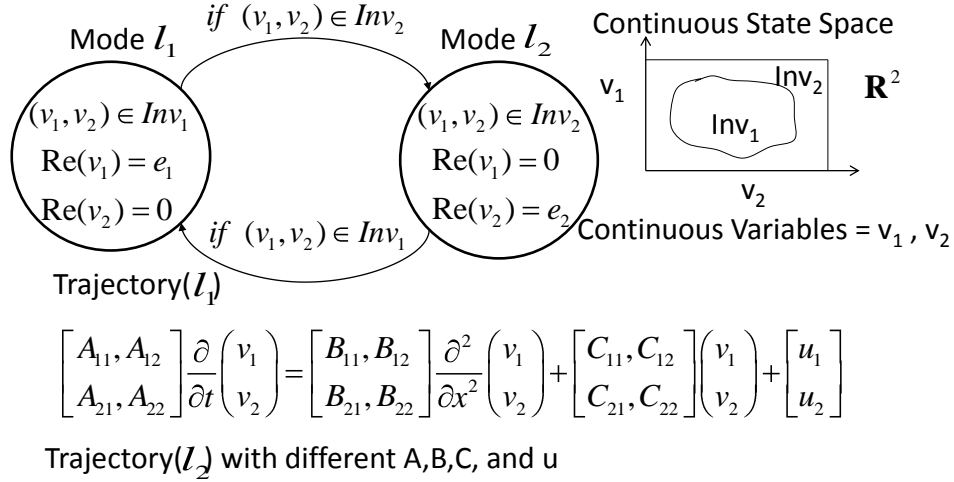


Figure 6.4: Conceptual illustration of L1STHA with two modes l_1 and l_2 and two continuous variables v_1 and v_2

mended limits. The infusion pump has three modes: a) basal, where infusion rate is I_0 , b) braking, where infusion rate is a fraction f of I_0 , and c) correction bolus, where infusion rate is incremented by I_b . Diffusion dynamics of the drug is spatio-temporal in nature and can be modeled using multi-dimensional PDE Equation 6.1 [111].

$$\frac{\partial d}{\partial t} = \nabla(D \nabla d) + \Gamma(d_B(t) - d) - \lambda d, \quad (6.1)$$

where $d(x, t)$ is the tissue drug concentration at time t and distance x from the infusion site, D is the diffusion coefficient of the blood, Γ is the blood to tissue drug transfer coefficient, and $d_B(t)$ is the prescribed infusion rate at time t , and λ is the drug decay coefficient. A control algorithm in the infusion pump samples Equation 6.1 and adjusts the infusion levels so as to achieve the desired physiological effects while avoiding hazards such as hyperglycemia. \diamond

6.3 Linear 1-D Space Spatio-Temporal Hybrid Automata

The L1STHA expresses the variation of systems properties according to a discrete control algorithm of the computing units in the CPS. A L1STHA is described as:

Definition 6 *L1STHA*: It is a tuple $\{\mathbb{L}, Inv, A, B, C, u, Re\}$,

- \mathbb{L} is a set of m discrete states or modes $\{l_1, l_2, \dots, l_m\}$.
- $Inv : \mathbb{L} \rightarrow 2^J$ is the invariant set, which maps each discrete state to a set of cells such that:
 - for each $l \in \mathbb{L}$ the cells in $Inv(l)$ are all connected,
 - for any two different modes $\{l_i, l_j\} \in \mathbb{L}$, $Inv^\square(l_i) \cap Inv^\square(l_j) = \phi$,
 - $\bigcup_{i \in \{1..m\}} Inv(l_i) = V$,
- $A : \mathbb{L} \rightarrow \mathbb{R}^n \times \mathbb{R}^n$, maps a mode to an $n \times n$ real valued matrix,
- $B : \mathbb{L} \rightarrow \mathbb{R}^n \times \mathbb{R}^n$, maps a mode to an $n \times n$ real valued matrix,
- $C : \mathbb{L} \rightarrow \mathbb{R}^n \times \mathbb{R}^n$, maps a mode to an $n \times n$ real valued matrix,
- $u : \mathbb{L} \rightarrow \mathbb{R}^n$, maps a mode to an $n \times 1$ real valued vector,
- $Re(\cdot) : \mathbb{L} \times V \rightarrow \mathbb{R}^n$ is a reset function that sets initial conditions of the variables in V at each mode $l_i \in \mathbb{L}$. \diamond

Associated with the definition of L1STHA is the definition of its trajectory that relates the variables A, B, C , and u .

Definition 7 *Trajectory*: The trajectory of an L1STHA with n continuous dimensions for time $t \in \mathbb{R}$ and within a region $s \subset \mathbb{R}$ at a discrete mode $l_i \in \mathbb{L}$ is defined as the 1-D space spatio-temporal mapping $\eta : [0, T] \times S \rightarrow \mathbb{R}^n$ such that:

- $\eta(t, x)$ for $t \in [0, T]$ and $x \in S$ follows the PDE:

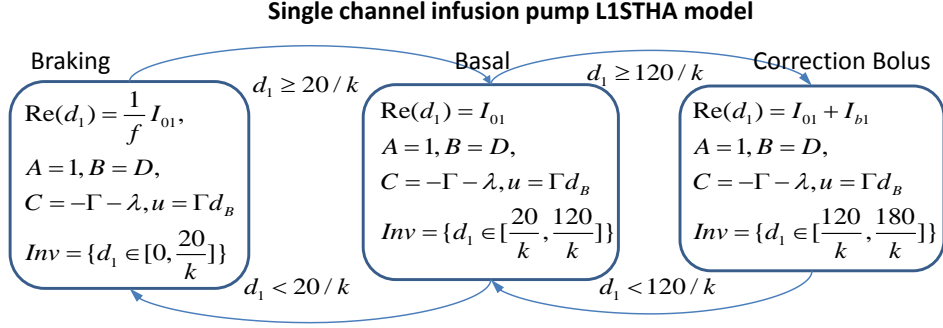
$$A_i \frac{\partial \eta(t, x)}{\partial t} = B_i \frac{\partial^2 \eta(t, x)}{\partial x^2} + C_i \eta(t, x) + u_i, \quad (6.2)$$

where $A_i, B_i, C_i,$ and u_i are for a mode l_i .

- and $\eta(t, x) \in \text{Inv}_i \forall t \in [0, T]$ and $x \in S$. \diamond

The duration of the trajectory at a given spatial coordinate x is denoted by $\eta|_x.dur$ and its spatial range at a given time t is denoted by $\eta^t.range$. A trajectory ends if at any space or time point the continuous variables cross invariant set boundaries. Figure 6.4 shows a conceptual view of a L1STHA. It has two modes l_1 and l_2 . Each mode is associated with an invariant set, a subset in the real space \mathbb{R}^2 . The invariants are used to determine transitions between modes (discussed later). Each mode has a reset function (assigning of constant values (e_1, e_2)), which reflects the effect of control operation on the system variables "whenever" or "wherever" the L1STHA first enters a mode. In each mode the trajectory is a linear 1-space dimensional PDE, with different values of $A, B, C,$ and u . L1STHA is applied on Example 12 as follows:

Example 13 Infusion Pump: *The L1STHA model for the infusion pump is shown in Figure 6.5. The L1STHA model has three modes in the set \mathbb{L} - a) correction bolus mode, b) braking mode, and c) basal infusion mode. The L1STHA is initially at the basal infusion mode l_0 . The continuous variable for the L1STHA model of infusion pump is the blood glucose concentration in the blood, which varies over space and time. The L1STHA is in: a) the basal infusion mode if the blood glucose concentration is within 20 mg/dl and 120 mg/dl with infusion rate I_0 , b) the correction bolus mode l_1 , if the blood glucose concentration is greater than 120 mg/dl with infusion rate $(I_b + I_0)$ and c) the braking mode l_2 , if the blood glucose*



$V = \{d_1\}$, $\mathbf{L} = \{\text{Braking, Basal, Correction Bolus}\}$, I_{01} = infusion increment for drug 1, I_b = Basal infusion rate of drug 1.

Figure 6.5: The L1STHA model of single channel infusion pump.

concentration is below 20 mg/dl with infusion rate I_0/f . Thus, the Inv set consists of the mappings $\{Inv(l_0) = ([20/k, 120/k])\}$, $\{Inv(l_1) = ([120/k, \infty])\}$, and $\{Inv(l_2) = ([0, 20/k])\}$. Here k is a constant factor that converts blood glucose concentration to insulin concentration. Such linear relationship is suggested by the Bergman Minimal Model [99]. It can be seen that the invariant mapping Inv satisfies the conditions in Definition 6. In each mode, the reset function Re represents the decision of the control algorithm to increase or decrease the infusion rate whenever the L1STHA enters the mode. The A, B, C , and u values can be derived from the diffusion Equation 6.1 and the trajectory Definition 7 as $A = 1$, $B = D$, $C = -\Gamma - \lambda$, and $u = \Gamma d_b(t)$. \diamond

With this definition of L1STHA we can also characterize aggregate effects as shown in the following example.

Example 14 Multi-channel Infusion with aggregate effects: Infusion pumps used in chemotherapy [111], often have multiple channels of infusion leading to aggregate effects of drugs. If we consider a region of the body at a fairly large distance from the site of infusion of a drug, the concentration decreases to negligible amounts (below a low threshold) at a given time according to Equation 6.1. How-

ever, over time the concentration at that region may increase to such an extent that it cannot be ignored. In case of a two channel infusion this phenomenon can happen for both the drugs at a given region. In that case the effective concentration of drug is a non-trivial combination of the dynamics of the individual drugs. Specifically, the drug concentration also follows Equation 6.1 but with modified parameters and conditions. To capture this aggregate effects, we first consider L1STHA models similar to Example 13, for the individual drugs with concentration d_1 and d_2 . The L1STHA of the mutli-channel pump has a mode set which is the Cartesian product of the mode sets of the individual L1STHA models. If we consider that the low threshold is $20/k$ mg/dl, then aggregate effects can only occur when $(d_1, d_2) \in ([20, \infty], [20, \infty])$ i.e., when the individual L1STHA models are either in basal or correction bolus modes. Hence of the nine possible modes that can occur due to the Cartesian product only four are aggregate effect modes (Figure 6.6). The aggregate effect can be modeled by introducing a new variable d_3 to all the modes. In modes without aggregate effects $d_3 = 0$, while in modes with aggregate effects d_3 follows a new PDE with new parameters D_3, Γ_3, λ_3 , and d_{B3} as suggested by [111]. The transition to the aggregate effect modes can only occur if both $d_1 > 20/k$ and $d_2 > 20/k$. The equation expressing the aggregate effect has to be specified to the L1STHA, its execution will determine when and where the aggregate effect occurs and with what intensity. Note that this condition is spatio-temporal in nature and unlike spatial networks of hybrid automata, impose no restriction on the space or time at which aggregate effects can occur. \diamond

The analysis of L1STHA models requires solution to the differential equations, which depend on initial and boundary conditions and govern mode transitions.

Multi channel Infusion pump formal model

Correction Bolus 1 + Braking 2

Aggregate Correction Bolus 1 + Correction Bolus 2

$$\text{Re}(d_1) = I_{01} + I_{b1}$$

$$\text{Re}(d_2) = \frac{1}{f} I_{02}, \quad \text{Re}(d_3) = 0$$

$$A = \begin{bmatrix} 1, 0, 0 \\ 0, 1, 0 \\ 0, 0, 0 \end{bmatrix}, \quad B = \begin{bmatrix} D_1, 0, 0 \\ 0, D_2, 0 \\ 0, 0, 0 \end{bmatrix},$$

$$C = \begin{bmatrix} -\Gamma_1 - \lambda_1, 0, 0 \\ 0, -\Gamma_2 - \lambda_2, 0 \\ 0, 0, 1 \end{bmatrix}, \quad u = \begin{bmatrix} \Gamma_1 d_{B1} \\ \Gamma_2 d_{B2} \\ 0 \end{bmatrix}$$

$$\text{Inv} = \{d_1 \in [\frac{120}{k}, \frac{180}{k}], d_2 \in [0, \frac{20}{k}]\}$$

$$\text{Re}(d_1) = I_{01} + I_{b1}$$

$$\text{Re}(d_2) = I_{02} + I_{b2}$$

$$\text{Re}(d_3) = \max(d_1(t, x_b), d_2(t, x_b))$$

$$A = \begin{bmatrix} 1, 0, 0 \\ 0, 1, 0 \\ 0, 0, 1 \end{bmatrix}, \quad B = \begin{bmatrix} D_1, 0, 0 \\ 0, D_2, 0 \\ 0, 0, D_3 \end{bmatrix},$$

$$C = \begin{bmatrix} -\Gamma_1 - \lambda_1, 0, 0 \\ 0, -\Gamma_2 - \lambda_2, 0 \\ 0, 0, -\Gamma_3 - \lambda_3 \end{bmatrix}, \quad u = \begin{bmatrix} \Gamma_1 d_{B1} \\ \Gamma_2 d_{B2} \\ \Gamma_3 d_{B3} \end{bmatrix}$$

$$\text{Inv} = \{d_1 \in [\frac{120}{k}, \frac{180}{k}], d_2 \in [\frac{120}{k}, \frac{180}{k}]\}$$

$$L = \{\text{Braking1, Basal1, Correction Bolus1}\} \times \{\text{Braking2, Basal2, Correction Bolus2}\}$$

$$V = \{d_1, d_2, d_3\} \quad \text{Nine states out of which four are aggregate effect states}$$

$$L_{aggr} = \{\text{Correction Bolus 1 + Correction Bolus2, Correction Bolus 1 + Basal 2,}$$

$$\text{Basal 1 + Correction Bolus 2, Basal 1 + Basal 2}\}$$

Figure 6.6: The normal and aggregate effect modes in the L1STHA model of multi-channel infusion pumps.

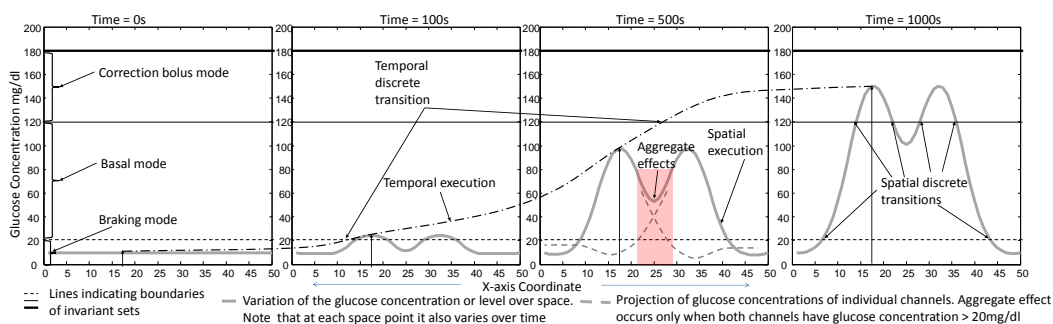


Figure 6.7: Example execution of the L1STHA model of a multi-channel infusion pump, for 1000 seconds and over a 50 mm spatial region.

6.4 L1STHA execution model

The definition of L1STHA is not complete and useful unless we define an execution and a discrete transition. Figure 6.7 shows a simulation of the multi-channel infusion pump L1STHA following the execution model to be discussed in this section. The figure shows three modes of operation of the infusion pump: basal, correction and braking. The plots show the trajectory, solution of Equation 6.1, with respect to space at given times (0s, 100s, 500s, and 1000s). It is to be noted that we define the L1STHA execution model with a *time bound* T and a *space bound* S .

Definition 8 *Initial State:* The initial state of a L1STHA at a given spatial coordinate $x \in S$, is a continuous state $v_{0,x} \in Inv_0$ such that $\eta(0, x) = v_0$ \diamond

As shown in the graph with time = 0s in Figure 6.7, initial state at the space point $x = 15\text{mm}$ is the value of the glucose concentration, which is 20 mg/dl.

Definition 9 *Initial Configuration:* The initial configuration of a L1STHA is the function $\eta(0, \cdot) : x \rightarrow \mathbb{R}^n$, & $x \in S$. \diamond

The initial configuration is the spatial variation of the glucose concentration at time $t = 0\text{s}$ as shown by the thick gray line in Figure 6.7 at time = 0s.

Definition 10 *Mode Boundary:* The mode boundary for any mode $l \in \mathbb{L}$ at a time t is the boundary of a spatial region s_l such that $\eta(t, x) \in Inv_l \forall x \in s_l$ \diamond

As time progresses the L1STHA is at different modes at different spatial regions. If we consider the time $t = 500\text{s}$, then the spatial region from 0mm to 9 mm is in braking mode, 9mm to 42mm is in basal infusion mode, and 42mm to 50mm is in braking mode. Hence, each mode has a spatial boundary at a given time called mode boundary while s_l for a mode l is the spatial region in which the L1STHA is in mode l . Note that the mode boundary shifts as time progresses.

Definition 11 *Boundary State:* The boundary state of a mode l_0 in L1STHA at a given time $t \in [0, T]$, is a continuous state $v_{t,x_0} \in Inv_0$ such that $\eta(t, x_0) = v_{t,x_0}$, where x_0 is in the mode boundary of l_0 . \diamond

The value of the glucose concentration at the mode boundary at a given time is the boundary state at that time. For $t = 500s$ the boundary state of the aggregate basal mode is with glucose concentration 78.4 mg/dl.

Definition 12 *Boundary Variation* The boundary variation of an L1STHA is a set of all functions $\eta(t \in [0, T], s_{l_0}^\square)$. An element in this set is of the form $\eta(t, x)$, where $t \in [0, T]$ and $x \in s_{l_0}^\square$.

Definition 13 *Temporal Execution:* A temporal execution α_t of a L1STHA at a given spatial coordinate $x \in S$ from an initial state $(l_0, v_{0,x}) \in \mathbb{L} \times \mathbb{R}^n$, is a concatenation of trajectories at x , $\alpha_t = \eta_0|_x \eta_1|_x \dots$ where:

- $\eta_0(0, x) = v_{0,x}$,
- $\eta_k(0, x) = Re(\eta_{k-1}(\eta_{k-1}|_x.dur, x))$,
- $\alpha_t.dur = \sum_i \eta_i|_x.dur$,

where η_k is the trajectory defined at a mode $l_k \in \mathbb{L}$, $Re(\cdot)$ is the reset function (Definition 6), and $\alpha_t.dur$ represents the duration of the temporal trajectory. $\eta_k|_x$ is the trajectory at a given coordinate x for a mode l_k . \diamond

A temporal execution is the variation of a continuous variable at a given space coordinate over time, as shown by the chain-dot lines in Figure 6.7. During the execution there can be transitions to different modes with changes in the dynamic equations. The first condition in the definition states that the execution starts from an initial state, the second condition shows the concatenation, where $\eta_{k-1}|_x.dur$ denotes the

time that the trajectory crossed the boundary of the invariant set for mode l_{k-1} at space x .

Definition 14 Spatial Execution: A spatial execution α_s of a L1STHA at a given time $t \in [0, T]$ from a boundary state $(l_0, v_{t,x_0}) \in \mathbb{L} \times \mathbb{R}^n$ is a concatenation of trajectories at time t , $\alpha_s = \eta_0|^t \eta_1|^t \dots$ where:

- $\eta_0(t, x_0) = v_{t,x_0}$,
- $\eta_k(t, x_0) = Re(\eta_{k-1}(t, \eta_{k-1}|^t.range))$,
- $\alpha_s.range = \bigcup_i \eta_i|^t.range$,

where η_k is the trajectory defined at a mode $l_k \in \mathbb{L}$, and $\alpha_s.range$ represents the range of the spatial execution. $\eta_k|^t$ is the trajectory at a given time t for a mode l_k . \diamond

A spatial execution is the variation of the continuous variables over space at a given time as shown by the thick gray lines in the four different graphs. At different times the spatial execution changes (Figure 6.7).

Definition 15 Temporal Discrete Transition: A temporal discrete transition at a given coordinate x from the mode l_i to l_j occurs at a continuous state $v(t', x)$ at a time t' , whenever $v(t', x) \in Inv_i^\square \cap Inv_j^\square$, where $v(t', x) = \lim_{t \rightarrow t'} v(t, x)$ and $v(t, x) \in Inv_i^\square$ for $t \in [t' - \tau, t']$ for some $\tau > 0$. \diamond

Temporal transition at a given space coordinate x occurs at a time t' if as the time approaches t' , the continuous state approaches the boundary of the invariant set of a mode l_i . The L1STHA transits to the state l_j if the invariant sets are connected and the continuous state approaches the intersection of the boundaries of the invariant sets of l_i and l_j . An example is shown in Figure 6.7, where the glucose concentration crosses invariant set boundaries from basal to correction bolus mode as time progresses (chain and dot line).

Definition 16 *Spatial Discrete Transition*: A spatial discrete transition at a give time t from the mode l_i to l_j occurs at a continuous state $v(t, x')$ at a spatial coordinate x' , wherever $v(t, x') \in \text{Inv}_i^\square \cap \text{Inv}_j^\square$, where $v(t, x') = \lim_{x \rightarrow x'} v(t, x)$ and $v(t, x) \in \text{Inv}_i^\square$ for $x \in [x - s, x]$ for some $s > 0$. \diamond

Spatial transition at a given time occurs similar to a temporal transition when the spatial execution crosses invariant set boundaries as shown in the leftmost graphs in Figure 6.7.

It is to be noted that in this theory we consider every transition to be deterministic and transversal. A deterministic transition means that at any time or space point the L1STHA from a given mode can only transit to a unique mode. This also guarantees that at a fixed time and space point the L1STHA is at an unique mode. Further, a transversal transition is assumed, which has the property that given the L1STHA transits from a mode l_1 to l_2 it stays at l_2 for a finite amount of time. This assumption prevents zeno behavior [128], where in a very small amount of time there are infinite transitions.

Definition 17 *Deterministic Transversal Discrete Transitions*: A discrete transition is called deterministic if from a given location l_i the continuous state $v(\tau_s, x_t)$ can only transit to one unique location l_j . A deterministic discrete transition is called transversal if there exists some $\epsilon > 0$ such that:

$$\frac{\overrightarrow{\delta v_i(t, x_t)}}{\delta t} \cdot \hat{n}_i > \epsilon \bigwedge \frac{\overrightarrow{\delta v_j(t, x_t)}}{\delta t} \cdot \hat{n}_i > \epsilon, \quad (6.3)$$

$$\frac{\overrightarrow{\delta v_i(\tau_s, x)}}{\delta x} \cdot \hat{n}_i > \epsilon \bigwedge \frac{\overrightarrow{\delta v_j(\tau_s, x)}}{\delta x} \cdot \hat{n}_i > \epsilon, \quad (6.4)$$

where v_i satisfies $A_i \frac{\delta v(\tau, x)}{\delta \tau} = B_i \frac{\delta^2 v(\tau, x)}{\delta x^2} + C_i v(\tau, x) + u_i$ and v_j satisfies $A_j \frac{\delta v(\tau, x)}{\delta \tau} = B_j \frac{\delta^2 v(\tau, x)}{\delta x^2} + C_j v(\tau, x) + u_j$, and \hat{n}_i is a outward normal unit vector to Inv_i^\square at $v(\tau_s, x_t)$.

Definition 18 *Deterministic Transversal Linear 1-D space Spatio-Temporal Hybrid Automata (DTL1STHA):* Given an L1STHA, a starting state (l_0, v_0) , a time bound T , a space region S , and a jump bound N , it is called a DTL1STHA if all the discrete transitions starting from v_0 for time T , within space region S , and for a maximum of N jumps are deterministic and transversal.

6.5 Defining the ϵ reach set of a DTL1STHA

Given these definitions of DTL1STHA and its execution and trajectory lets define ϵ reach set of a DTL1STHA. The determination of the ϵ reach set is bounded by time and space and an analysis technique similar to Kim [113] is proposed. However, Kim did it for only temporal execution of a linear first order hybrid automata. This thesis considers spatio-temporal execution of a DTL1STHA with linear second order differential equations.

Definition 19 *Reach State of a DTL1STHA:* A continuous state $V \in \mathbb{R}^n$ is reachable if it is reached by a spatial or temporal execution at some time t and spatial coordinate x .

Note that if a continuous state V is reached by a temporal execution given a spatial point x at time t , then it is also reached by a spatial execution given a time t at a spatial coordinate x .

Definition 20 *Time and Space Bounded Reach Set of DTL1STHA:* Given a time bound T and a spatial region S , the time and space bounded reach set of DTL1STHA, $R_{TS}(V_0)$, from an initial configuration $V_0 \subset Inv_0$ is the set of continuous states that can be reached by any spatial execution within the region S or temporal execution of duration T .

Definition 21 ϵ reach set of DTL1STHA: Given an $\epsilon > 0$ a set of continuous states M of a DTL1STHA starting from an initial configuration $\{l_0, V_0\} \in \mathbb{L} \times \mathbb{R}^n$ over a time duration of T and within a spatial region S , is called an epsilon reach set, if:

- $R_{TS}(V_0) \subseteq M$, and
- $d_H(R_{TS}(V_0), M) \leq \epsilon$, where $d_H(P, Q)$ is the hausdorff distance [126] between two set P and Q .

Solving the PDE

In order to compute the reachable states of the DTL1STHA, we need to have the capability to solve the partial differential Equation 6.2. The initial conditions for solving this PDE is given by -

$$\eta(0, x) = I_b \forall x \in s \quad (6.5)$$

where I_b is a constant value over time and space.

It is to be noted that this PDE has to be solved as a free boundary problem. This assumption comes from the observations in various medical device examples. Let us consider an infusion pump diffusing insulin into the human body. The insulin concentration is governed by the PDE of the same form as in Equation 6.2. The concentration spreads in space reaching different parts of the body through blood circulation over time. Hence as time progresses the effect of insulin spreads further away from the site of infusion. Further, at any given position the insulin concentration increases with time. Hence, it is not intuitive to have a constant boundary condition. Hence, we will consider this problem as a free boundary problem, where the boundary at which the minimum insulin concentration is observed moves away from the site of infusion with some velocity. However, one fixed boundary condition exists at the site of infusion. This corresponds to the constant infusion rate from the infusion needle. The constant boundary condition is given by:

$$-B \frac{\delta \eta(t, x)}{\delta x} = I_0, \quad (6.6)$$

where I_0 is a constant that can be the bolus infusion rate in case of infusion pumps. Given these initial and boundary conditions the free boundary PDE is solved in the following subsection.

From Equation 6.2, we can rewrite it as:

$$B \frac{\delta^2 \eta}{\delta x^2} - A \frac{\delta \eta}{\delta t} + C(\eta + C^{-1}u) = 0 \quad (6.7)$$

Replacing η by $\theta = \eta + C^{-1}u$ in equation 6.7 we get:

$$B \frac{\delta^2 \theta}{\delta x^2} - A \frac{\delta \theta}{\delta t} + C(\theta) = 0 \quad (6.8)$$

Taking Laplace transform on both sides of Equation 6.8 we get:

$$B \frac{\delta^2 \theta(\psi)}{\delta x^2} - (A\psi - C)\theta(\psi) = 0 \quad (6.9)$$

where ψ is the laplace parameter.

To solve the homogeneous second order linear differential equation let us assume that $\theta(\psi) = e^{mx}$. This assumption leads to the following values of m as in Equation 6.10:

$$m = \pm \sqrt{B^{-1}(A\psi - C)} \quad (6.10)$$

Since plus value indicates exponential increase in space, which is not observed, only consider negative value of m is considered.

Hence the final solution for $\theta(\psi)$ is given by:

$$\theta(\psi) = -k_1 e^{-[\sqrt{B^{-1}(A\psi - C)}]x} \quad (6.11)$$

where k_1 is a constant to be found out from the boundary condition.

Applying laplace transform to the boundary condition Equation 6.6, and putting $\theta(\psi)$ we get:

$$\theta(\psi) = \frac{B^{-1}I_0}{\psi} [B^{-1}(A\psi - C)]^{-1/2} e^{-[\sqrt{B^{-1}(A\psi-C)}]x} \quad (6.12)$$

$\theta(t)$, the temporal variation can be obtained by performing an inverse laplace transform of Equation 6.12. Applying the inverse laplace transform on the Equation 6.12 we get,

$$\theta(t) = \frac{1}{2\pi j} \int_{-j\infty}^{j\infty} \frac{B^{-1}I_0}{\psi} [B^{-1}(A\psi - C)]^{-1/2} e^{-[\sqrt{B^{-1}(A\psi-C)}]x} e^{\psi t} d\psi \quad (6.13)$$

where $j = \sqrt{-1}$.

After performing the integration we get the value of $\theta(t)$ as in Equation 6.14:

$$\begin{aligned} \theta(t) = & \frac{1}{2} I_0 B^{-1} C^{-1} [e^{-(\sqrt{B^{-1}C})x} \operatorname{erfc}\left(\frac{(\sqrt{B^{-1}C})x}{2(\sqrt{CA^{-1}t})} - (\sqrt{CA^{-1}t})\right) \\ & - e^{(\sqrt{B^{-1}C})x} \operatorname{erfc}\left(\frac{(\sqrt{B^{-1}C})x}{2(\sqrt{CA^{-1}t})} + (\sqrt{CA^{-1}t})\right)] \end{aligned} \quad (6.14)$$

Hence the value of η after solving the differential equation is given by Equation 6.15.

$$\begin{aligned} \eta(t, x) = & I_b + I_0 (\sqrt{4BC})^{-1} [e^{-\sqrt{B^{-1}C}x} \operatorname{erfc}((\sqrt{4BA^{-1}t})^{-1}x - \sqrt{CA^{-1}t}) \\ & - e^{\sqrt{B^{-1}C}x} \operatorname{erfc}((\sqrt{4BA^{-1}t})^{-1}x + \sqrt{CA^{-1}t})] \end{aligned} \quad (6.15)$$

$\eta(t, x)$ can be computed from the Equation 6.15.

Notations

The table 6.1 lists the important notations and their definitions that will be used further in the text. With these notations defined and the assumptions on exact computation, we move on to the time and space bounded epsilon reach set of a DTL1STHA with a single location or discrete state.

Table 6.1: Notations for reachable states and over approximations

Notation	Definition
$D_{t,x}(P)$	It is the set of reachable states at time t and coordinate x from a set P at time 0 and origin.
$D_{t,x}(P, \gamma)$	It is an over approximation of the set $D_{t,x}(P)$ such that i) $D_{t,x}(P) \subset D_{t,x}(P, \gamma)$ and ii) $d_H(D_{t,x}(P), D_{t,x}(P, \gamma)) \leq \gamma$ for some $\gamma > 0$.
$D_{0,0}(P, \gamma)$	It is the γ over approximation of the set P .
$B_r(p)$	It is a hypercubic neighborhood of a point p in the state space i.e. $B_r(p) = \{y \in \mathbb{R}^n : \ y - p\ \leq r\}$

6.6 Time and space bounded epsilon reach set of a DTL1STHA with a single discrete location

Let us first consider the bounded epsilon reach set computation of a DTL1STHA with a single discrete location l_0 and the associated invariant set Inv_0 . We want to find out the epsilon reach set by sampling the time and the space dimension with time intervals h_x at a spatial coordinate x and space grid h_t at a given time. Now sampling will lead to errors in state estimation. Hence, we need to find an epsilon ϵ such that from any sampled state $\eta(t, x)$ the trajectory remains within $B_\epsilon(\eta(t, x))$ before the next sampled state is considered. We now derive the values of h_x and h_t .

To ensure that the trajectory stays within a hypercube of dimension length ϵ we have to obtain h_x and h_t such that,

$$\max_{\tau \in [0, h_x]} \|\eta(t + \tau, x) - \eta(t, x)\|_\infty < \epsilon, \quad (6.16)$$

$$\max_{s \in [0, h_t]} \|\eta(t, x + h_t) - \eta(t, x)\|_\infty < \epsilon \quad (6.17)$$

We first consider the time differential of η -

$$\max_{t \in [t, t+h_x]} \|\dot{\eta}(t, x)\|_\infty = \max_{t \in [t, t+h_x]} \|A^{-1}B \frac{\delta^2 \eta(t, x)}{\delta x^2} + A^{-1}C\eta(t, x) + A^{-1}u\|_\infty \quad (6.18)$$

or,

$$\begin{aligned} \max_{t \in [t, t+h_x]} \|\dot{\eta}(t, x)\|_\infty &\leq \|A^{-1}B\|_\infty \max_{t \in [t, t+h_x]} \left\| \frac{\delta^2 \eta(t, x)}{\delta x^2} \right\|_\infty \\ &+ \|A^{-1}C\|_\infty \max_{t \in [t, t+h_x]} \|\eta(t, x)\|_\infty + \|A^{-1}u\|_\infty \end{aligned} \quad (6.19)$$

We need to express $\max_{t \in [t, t+h_x]} \left\| \frac{\delta^2 \eta(t, x)}{\delta x^2} \right\|_\infty$ and the term $\max_{t \in [t, t+h_x]} \|\eta(t, x)\|_\infty$ in terms of the location invariants so that they can be easily computed. The absolute max of the term $\max_{t \in [t, t+h_x]} \|\eta(t, x)\|_\infty$ is the maximum value that η can take while staying in the location. This means that $\max_{t \in [t, t+h_x]} \|\eta(t, x)\|_\infty = \eta^{max} = \max_{t \in T \text{ and } x \in S} Inv_0$.

To find the maximum of the double differentiation we employ Finite Difference Time Domain approximation. The double differentiation can be expressed as follows:

$$\frac{\delta^2 \eta(t, x)}{\delta x^2} = \frac{2\eta(x) - \eta(x + h_t) - \eta(x - h_t)}{h_t^2} \quad (6.20)$$

The maximum value of this term is obviously $\frac{2(\eta^{max} - \eta^{min})}{h_t^2}$, where $\eta^{min} = \min_{t \in T \text{ and } x \in S} Inv_0$. Hence we get,

$$\begin{aligned} \max_{t \in [t, t+h_x]} \|\dot{\eta}(t, x)\|_\infty &\leq \|A^{-1}B\|_\infty \frac{2(\eta^{max} - \eta^{min})}{h_t^2} + \|A^{-1}C\|_\infty \eta^{max} \\ &+ \|A^{-1}u\|_\infty \end{aligned} \quad (6.21)$$

Therefore we can derive $\max_{\tau \in [0, h_x]} \|\eta(t + \tau, x) - \eta(t, x)\|_\infty$ as follows -

$$\max_{\tau \in [0, h_x]} \|\eta(t + \tau, x) - \eta(t, x)\|_\infty \leq \int_t^{t+h_x} \max_{t \in [t, t+h_x]} \|\dot{\eta}(t, x)\|_\infty dt \quad (6.22)$$

$$\max_{\tau \in [0, h_x]} \|\eta(t + \tau, x) - \eta(t, x)\|_\infty \leq \|A^{-1}B\|_\infty \frac{2(\eta^{max} - \eta^{min})}{h_t^2} h_x + \|A^{-1}C\|_\infty \eta^{max} h_x + \|A^{-1}u\|_\infty h_x \quad (6.23)$$

Therefore to ensure that the trajectory stays within the hypercubic bound ϵ during the sampling period h_t we should have -

$$\|A^{-1}B\|_\infty \frac{2(\eta^{max} - \eta^{min})}{h_t^2} h_x + \|A^{-1}C\|_\infty \eta^{max} h_x + \|A^{-1}u\|_\infty h_x \leq \epsilon \quad (6.24)$$

Hence, we can take

$$\|A^{-1}B\|_{\infty} \frac{2(\eta^{max} - \eta^{min})}{h_t^2} h_x + \|A^{-1}C\|_{\infty} \eta^{max} h_x + \|A^{-1}u\|_{\infty} h_x = \epsilon/2 \quad (6.25)$$

Similarly we can write,

$$\max_{s \in [x, x+h_t]} \left\| \frac{\delta^2 \eta(t, x)}{\delta x^2} \right\|_{\infty} \leq \|B^{-1}A\|_{\infty} \frac{\eta^{max} - \eta^{min}}{h_x} - \|B^{-1}C\|_{\infty} \eta^{min} + \|B^{-1}u\|_{\infty} \quad (6.26)$$

Therefore,

$$\begin{aligned} \max_{s \in [0, h_t]} \|\eta(t, x+s) - \eta(t, x)\|_{\infty} &\leq \int_x^{x+h_t} \int_x^{x+h_t} \|B^{-1}A\|_{\infty} \frac{\eta^{max} - \eta^{min}}{h_x} dx dx \quad (6.27) \\ &- \int_x^{x+h_t} \int_x^{x+h_t} [\|B^{-1}C\|_{\infty} \eta^{min} + \|B^{-1}u\|_{\infty}] dx dx \end{aligned}$$

or,

$$\begin{aligned} \max_{s \in [0, h_t]} \|\eta(t, x+s) - \eta(t, x)\|_{\infty} &\leq \frac{\|B^{-1}A\|_{\infty} (\eta^{max} - \eta^{min}) h_t^2}{2h_x} \quad (6.28) \\ &- \|B^{-1}C\|_{\infty} \eta^{min} h_t^2 / 2 - \|B^{-1}u\|_{\infty} h_t^2 / 2 \end{aligned}$$

Again to ensure that the trajectory remains within the hypercube of dimension ϵ we should have -

$$\frac{\|B^{-1}A\|_{\infty} (\eta^{max} - \eta^{min}) h_t^2}{2h_x} - \|B^{-1}C\|_{\infty} \eta^{min} h_t^2 / 2 - \|B^{-1}u\|_{\infty} h_t^2 / 2 \leq \epsilon \quad (6.29)$$

Hence we can assume,

$$\frac{\|B^{-1}A\|_{\infty} (\eta^{max} - \eta^{min}) h_t^2}{2h_x} - \|B^{-1}C\|_{\infty} \eta^{min} h_t^2 / 2 - \|B^{-1}u\|_{\infty} h_t^2 / 2 = \epsilon/2 \quad (6.30)$$

The value of h_t and h_x can be found out by simultaneously solving the equations 6.25 and 6.30.

The simultaneous solution to these equations may not always be feasible, leading to inapplicability of the ftd approximation. We see that if the elements of the

C vector are positive then the spatial sampling interval is imaginary. Only when all elements in C are negative there is a feasible value of the spatial sampling interval. If the elements of C are all positive then the spatial and temporal sampling intervals can be computed from the equations 6.31 and 6.32.

$$h_x = \frac{\epsilon}{BA^{-1} \frac{\partial^2 \eta}{\partial x^2} |_{max} + CA^{-1} \eta^{max} + uA^{-1}} \quad (6.31)$$

$$h_t^2/2 = \frac{\epsilon}{BA^{-1} \frac{\partial \eta}{\partial t} |_{max} - CB^{-1} \eta^{min} - uB^{-1}} \quad (6.32)$$

Given these sampling intervals let us now define an epsilon reach set. The following definition will give us the method to compute the epsilon reach set of a one location DTL1STHA.

Lemma 6.6.1 *Given $\epsilon > 0$ a bounded ϵ reach set $R_{TS}(V_0, \epsilon)$ of a one location DTL1STHA from an initial configuration $V_0 \subset Inv_0$ from time $[0, T]$ and within space S can be determined as -*

$$R_{TS}(V_0, \epsilon) = \bigcup_{k=0}^{m-1} \bigcup_{j=0}^{r-1} B_\epsilon(\eta(kh_x, B_{jh_t}(x_0))), \quad (6.33)$$

where $m = T/h_x$, $r = \|S^{b \cdot o \cdot x \cdot d \cdot o \cdot t}\|/h_t$, x_0 is the origin of the coordinate system, and h_x and h_t are derived from the equations 6.31 and 6.32. The set has the following properties -

i $\text{Lim}_{\epsilon \rightarrow 0} R_{TS}(V_0, \epsilon) = R_{TS}(V_0)$ and

ii It contains the $\epsilon/2$ neighborhood of $R_{TS}(V_0)$ i.e,

$$\bigcup_{z \in R_{TS}(V_0)} B_{\epsilon/2}(z) \subseteq R_{TS}(V_0, \epsilon) \quad (6.34)$$

Proof: For the first property of the set $R_{TS}(V_0, \epsilon)$, we observe that, as $\epsilon \rightarrow 0$ we have $h_t \rightarrow 0$ and $h_x \rightarrow 0$. This establishes that in the limiting case there is no discretization. Hence, $\text{Lim}_{\epsilon \rightarrow 0} R_{TS}(V_0, \epsilon) = R_{TS}(V_0)$.

Further, we have already selected h_x and h_t as half of the value required to guarantee ϵ reach set. Hence, $\eta(kh_x, B_{jh_t}(x_0))$ is in $\epsilon/2$ neighborhood of $R_{TS}(V_0)$. Since, $R_{TS}(V_0, \epsilon)$ considers the ϵ neighborhood of $\eta(kh_x, B_{jh_t}(x_0))$ it already contains the $\epsilon/2$ neighborhood of any $\eta(t, s)$. Hence proved.

6.7 ϵ reach set for a set of initial configurations

In this section, we consider a set of initial configurations which are within a δ ball of the initial configuration $V_0, B_\delta(V_0)$. First it is shown that there exists a $\delta \in R^+$ such that the reach set starting from the set of initial configurations $B_\delta(V_0)$ is contained in the ϵ neighborhood of $R_{TS}(V_0)$.

Lemma 6.7.1 *Given an $\epsilon > 0$, an initial configuration V_0 of a DTL1STHA, a time interval $[0, T]$ and a spatial region S , there exists a $\delta > 0$ such that,*

$$R_{TS}(B_\delta(V_0)) \subseteq R_{TS}(V_0, \epsilon), \quad (6.35)$$

where $B_\delta(V_0)$ is a δ neighborhood of the initial configuration V_0 and $R_{TS}(B_\delta(V_0))$ is the reachable set starting from the set of initial configurations $B_\delta(V_0)$ upto a time T and within spatial region S . In particular it will be shown that, $\delta = \epsilon/2H$ for an appropriate H .

Proof: Let us consider a point p within the location boundary of the initial location l_0 . Let the initial state at the point p be v_0 . A continuous state can be reached either by traversing in time or through space following the continuous dynamics of the DTL1STHA:

$$\eta(t, x) = I_b + \frac{I_0}{\sqrt{4bc}} [e^{-\sqrt{c/b}x} \operatorname{erfc}\left(\frac{x}{\sqrt{4b/at}} - \sqrt{c/at}\right) - e^{\sqrt{c/b}x} \operatorname{erfc}\left(\frac{x}{\sqrt{4b/at}} + \sqrt{c/at}\right)] \quad (6.36)$$

If we consider two continuous states v_1 and v_2 in $R_{TS}(B_\delta(V_0))$ then the maximum difference between the two is given by -

$$\begin{aligned} \|v_1 - v_2\|_{max} &= & (6.37) \\ & \max(\|I_{b1} - I_{b2}\| + \frac{\|I_{01} - I_{02}\|}{\sqrt{4bc}} [e^{-\sqrt{c/bx}} \operatorname{erfc}(\frac{x}{\sqrt{4b/at}} - \sqrt{c/at}) \\ & - e^{\sqrt{c/bx}} \operatorname{erfc}(\frac{x}{\sqrt{4b/at}} + \sqrt{c/at})]) \\ & \leq \|I_{b1} - I_{b2}\| + 2 \frac{\|I_{01} - I_{02}\|}{\sqrt{4bc}} \end{aligned}$$

The initial configurations for v_1 and v_2 were within a δ bound of the initial configuration V_0 , $\forall x \in S_{t_0}$. Hence, $\|I_{b1} - I_{b2}\| \leq \delta$, this is obtained by putting $t = 0$ in the Equation 6.15. I_0 is the impulse bolus at time $t = 0$. Hence, we can assume that $\|I_{01} - I_{02}\| \leq \delta$. For the difference between the continuous states to be less than ϵ , we should have,

$$\delta(1 + \frac{2}{\sqrt{4bc}}) \leq \epsilon/2 \quad (6.38)$$

Hence, we get for an $H = (1 + \frac{2}{\sqrt{4bc}})$, the reachable states from a $\delta = \epsilon/2H$ boundary of the initial configuration V_0 is within an epsilon bound of the reachable states from V_0 .

Let us now focus on proving that the γ approximation of the reachable states starting from a δ ball around the initial configuration is also within ϵ approximation of the reachable states starting from V_0 .

Lemma 6.7.2 Given an $\epsilon > 0$, a one mode DTL1STHA, an initial configuration V_0 , a time interval T , and a spatial region S , there exists a $\delta > 0$ and a $\gamma > 0$ such that, $R_{TS}(B_\delta(V_0), \gamma) \subseteq R_{TS}(V_0, \epsilon)$. In particular we show that, $R_{TS}(V_0) \subseteq R_{TS}(B_{\epsilon/4H}(V_0), \epsilon/4) \subseteq R_{TS}(V_0, \epsilon)$, where $H = (1 + \frac{2}{\sqrt{4bc}})$.

Proof: Let us consider continuous states v_m and v_p in $B_\delta(V_0)$ and V_0 respectively. The continuous state reached after time t and at spatial location x starting from v_m is $\eta(t, x)|_{v_m}$. Let v_n be a continuous state in $R_{TS}(B_\delta(V_0), \gamma)$, then we get,

$$\|v_n - \eta(t, x)|_{v_p}\| = \|v_n - \eta(t, x)|_{v_m} + \eta(t, x)|_{v_m} - \eta(t, x)|_{v_p}\| \quad (6.39)$$

$$\leq \|v_n - \eta(t, x)|_{v_m}\| + \|\eta(t, x)|_{v_m} - \eta(t, x)|_{v_p}\| \quad (6.40)$$

$$\leq \gamma + H\delta \quad (6.41)$$

Since the sampling rate accounts for half of the approximation error, we can say that $v_n \in R_{TS}(V_0, 2(\gamma + H\delta))$. Therefore if we choose $\gamma = \epsilon/4$ and $\delta = \epsilon/4H$ the lemma is proved.

6.8 Determining exit condition from an invariant set

With a suitable over approximation we can determine the first time the DTL1STHA crosses the boundary of an invariant set. Let us consider that the first time that the set of reachable states crosses the continuous domain V at time $\tau_1 < T$. Then we have the following lemma:

Lemma 6.8.1 *Given a DTL1STHA, an initial state v_0 at a given space point $x_0 \in S$, if the trajectory $\eta(t, x)$ exits the domain V at time $\tau_1 < T$ and space point $x_1 \in S$, then for all small enough $\delta > 0$ there exists some h_x and h_t such that, $B_\delta(\eta(mh_x, nh_t)) \subset V^C$ for some m and n .*

Proof: Let us consider a ball of radius r around $\eta(\tau_1, x_1)$. Now since the trajectory is moving to V^C at (τ_1, x_1) then $\langle \eta(\tau_1, x_1). \hat{n}_1 \rangle > 0$, where \hat{n} is the unit normal vector at $\eta(\tau_1, x_1)$ outward of V^\square . Now since the dynamic equations of the DTL1STHA are continuous then we can say that for an appropriately small r , $\forall v \in B_r(\eta(\tau_1, x_1)) \cap V^\square$, $\langle v. \hat{n}_1 \rangle > 0$. If we consider a larger values of $\frac{\delta\eta}{\delta t}$ and $\frac{\delta^2\eta}{\delta x^2}$ then the approximation error for the reachable states will be larger which will result in a larger

boundary for the reachable states. Let us consider that, $\frac{\delta\eta}{\delta t}|_{max} = (b/a)(2\frac{\delta^2\eta}{\delta x^2} + r) + (c/a)(2\eta_{max} + r) + u/a$ and $\frac{\delta^2\eta}{\delta x^2}|_{max} = (a/b)(2\frac{\delta\eta}{\delta t} + r) - (c/b)(\eta_{min}/2 - r) - u/b$. Then if we consider $2h_x < r/\frac{\delta\eta}{\delta t}|_{max}$ and $2h_t^2 < r/\frac{\delta\eta}{\delta x}|_{max}$ then it is assured $\eta(\tau_1 + 2h_x, x_1 + 2h_t) \in V^C$. Since, the domain is compact we can always find a δ from Lemma 6.6.1, such that $B_\delta(\eta(mh_x, nh_t)) \subset V^C$.

6.9 Algorithm for bounded ϵ reach set of a single location DTL1STHA

The algorithm computes the approximate image of an initial configuration V_0 over time and outputs the reachable states based on the approximation parameter ϵ . The input to the algorithm are: i) a single location DTL1STHA, ii) an initial configuration V_0 , iii) an invariant set Inv , iii) the parameter ϵ , iv) a parameter α , v) the time bound T , and vi) the space bound S . The algorithm outputs the set of reachable states the time and spatial location at which the DTL1STHA exits the invariant set Inv and the γ and δ parameters.

The algorithm 6.1 first considers a δ boundary of the initial configuration V_0 and tries to compute the reachable states that are either within V or are in V^C . Note that if a set of reachable state intersects with both V and V^C then it means that at the given space and time the DTL1STHA is in two different locations which is not feasible. However, since the computed reachable states are an ϵ approximation such infeasible cases may occur during execution of the algorithm. Hence, if such an infeasible case occurs the algorithm updates the values of γ and δ such that the bounded epsilon reach set is within either V or V^C .

The algorithm increments time in steps of h_x . For time $t = 0$, the algorithm starts with $B_\delta(V_0)$ and computes its image for different space points starting from $x = 0$, to $x = S$ in steps of h_t . After each computation of image the algorithm checks whether the image is within V or not. If it is not within V there can be two cases: a) it is within V^C , in that case the algorithm terminates and the reachable set is

the image of $B_\delta(P_0)$ at the current time and space location, and b) it is not within V^C , in that case the algorithm returns an empty set as the reachable state, updates the parameters γ and δ by multiplying the previous values with α and restarts the computation at $t = 0$ and $x = 0$. If the image is within V for all $x \in S$, the algorithm then increments t by h_x . It then computes the image of $B_\delta(P_0)$ at $x = 0$, and $t = t + h_x$. The computation of images with respect to spatial dimension is again repeated until $t < T$.

We now discuss how the image can be computed in a simple manner with the help of the theorems and lemmas established in the previous sections. We will first discuss the methodology of image computation and then prove that the computation methodology leads to an ϵ approximation of the image.

Given an initial state the δ neighborhood can be computed by considering points at a hausdorff distance of δ . This is essentially a box around the initial state as shown in the Figure 6.3. The images are computed twice, once through time and for each time interval through space. For every vertex of the box the spatial or temporal image for the next interval is computed by simulating the Equation 6.15 with approximate values of the error functions. The next step is to create a δ neighborhood of the images of the vertices. This step results in a fresh set of vertices created from the neighborhood of the images of the vertices of the neighborhood of the initial state. The convex hull of the new set of vertices gives the ϵ reachable states after h_x time and at space location $x_0 + nh_t$ for some $n > 0$.

If the convex hull is entirely within Inv_0 then there is no transition between locations. On the other hand if the convex hull is entirely within Inv_0^C , then there is a transition to a different location. Note that the convex hull cannot intersect both Inv_0 and Inv_0^C . Intersection means that the DTL1STHA is in two locations at the same time and space, which indicates that the DTL1STHA is not deterministic.

Algorithm 6.1: $(R_{set}, t, x) = \text{CalcReach}(D\Sigma, Inv, V_0, T, S, \epsilon, \alpha)$

```

1:  $R_{set} = \phi$ ;
2: while  $R_{set} == \phi$  do
3:    $\gamma = \alpha\epsilon$ ;  $\delta = \alpha\epsilon$ ;
4:    $h_x = \frac{\gamma/2}{\frac{b}{a} \frac{\delta^2 \eta}{\delta x^2} |l_{max} + \frac{c}{a} \eta_{max} + \frac{u}{a}}$ ;
5:    $h_t = \sqrt{\frac{\gamma}{\frac{b}{a} \frac{\delta \eta}{\delta t} |l_{max} - \frac{c}{b} \eta_{min} - \frac{u}{b}}}$ ;
6:    $t = 0$ ;  $P_c = B_\delta(V_0)$ ;
7:   while  $t \leq T$  do
8:      $x = 0$ ;
9:      $D_{t,x}(P_c) = P_c$ ;
10:    while  $x \leq S$  do
11:       $x = x + h_t$ ;
12:       $D_{t,x}(P_c) = \text{Compute the image at } x \text{ from } D_{t,x-h_t}(P_c)$ ;
13:      if  $(D_{t,x}(P_c) \not\subset Inv)$  then
14:        if  $(D_{t,x}(P_c) \subset Inv^C)$  then
15:           $R_{set} = R_{set} \cup D_{t,x}(P_c)$ ;
16:          return  $(R_{set}, D_{t,x}(P_c), t, x, \gamma, \delta)$ ;
17:        else
18:           $R_{set} = \phi$ ;
19:           $(R_{set}, D_{t,x}(P_c), t, x, \gamma, \delta) = \text{CalcReach}(D\Sigma, Inv, V_0, T, S, \alpha\epsilon, \alpha)$ ;
20:        end if
21:      end if
22:    end while
23:     $t = t + h_x$ ;
24:     $D_{t,x}(P_c) = \text{Compute the image at } t \text{ from } D_{t-h_x,x}(P_c)$ ;
25:    if  $(D_{t,x}(P_c) \not\subset Inv)$  then
26:      if  $(D_{t,x}(P_c) \subset Inv^C)$  then
27:         $R_{set} = R_{set} \cup D_{t,x}(P_c)$ ;
28:        return  $(R_{set}, D_{t,x}(P_c), t, x, \gamma, \delta)$ ;
29:      else
30:         $R_{set} = \phi$ ;
31:         $(R_{set}, D_{t,x}(P_c), t, x, \gamma, \delta) = \text{CalcReach}(D\Sigma, Inv, V_0, T, S, \alpha\epsilon, \alpha)$ ;
32:      end if
33:    end if
34:  end while
35: end while

```


Formally, the algorithm first computes P_0 the δ neighborhood of the initial state $v_{x_0} = V_0(x_0)$ within the initial configuration V_0 . The set $D_{t+h_x, x}(P_0)$ or $D_{t, x+h_t}(P_0)$ is computed from the set $D_{t, x}(P_0)$ by exploiting the polyhedral structure of P_0 . The algorithm first considers the set of vertices $Vert(t, x)$ of the image $D_{t, x}(P_0)$. Then for each $vert(t, x) \in Vert(t, x)$, $vert(t + h_x, x)$ or $vert(t, x + h_t)$ using the solution of the dynamic Equation 6.15. Hence the set $Vert(t + h_x, x)$ or $Vert(t, x + h_t)$ can be computed from $Vert(t, x)$, which are essentially the images of the vertices in $Vert(t, x)$. The image $D_{t+h_x, x}(P_0)$ or $D_{t, x+h_t}(P_0)$ can then be computed from the convex hull of the set of vertices in $Vert(t + h_x, x)$ or $Vert(t, x + h_t)$ respectively. To obtain a γ approximation of the reachable states we can take γ boundaries of the new vertex set $B_\gamma(Vert(t + h_x, x))$ or $B_\gamma(Vert(t, x + h_t))$ are obtained. Since the h_t and h_x were determined such that the computation of the dynamic equations wont result in an error greater than γ we can conclude that the γ approximation of the reachable states $R_{t+h_x, x}(v_{x_0})$ or $R_{t, x+h_t}(v_{x_0})$ can be computed by taking the convex hull of the vertices of $\bigcup_{v \in Vert(t+h_x, x)} Vert(B_\gamma(v))$ or $\bigcup_{v \in Vert(t, x+h_t)} Vert(B_\gamma(v))$ respectively.

Lemma 6.9.1 *If \mathbb{H} is the convex hull of the set $\bigcup_{v \in V} Vert(B_\gamma v)$ then it is the closed γ neighborhood of the convex hull of the set V .*

Proof: Consider a point w in \mathbb{H} . Then there exists two points y_1 and y_2 in $\bigcup_{v \in V} Vert(B_\gamma v)$ such that, $w = \lambda y_1 + (1 - \lambda)y_2$ for some $\lambda > 0$. There should exist some v_1 and v_2 in V such that $\|y_1 - v_1\| \leq \gamma$ and $\|y_2 - v_2\| \leq \gamma$. Further, $w' = \lambda v_1 + (1 - \lambda)v_2$ must be in the convex hull of the set V .

Note that,

$$\|w - w'\| = \|\lambda y_1 + (1 - \lambda)y_2 - \lambda v_1 - (1 - \lambda)v_2\| \quad (6.42)$$

$$\leq \lambda \|y_1 - v_1\| + (1 - \lambda) \|y_2 - v_2\| \quad (6.43)$$

$$\leq \gamma \quad (6.44)$$

Hence, w is in the γ neighborhood of V .

Let us consider w_v in the γ neighborhood of the convex hull of the set V . Then there exists an element v_c in the convex hull of V such that, $\|w_v - v_c\| \leq \gamma$. Further, there exists v_1 and v_2 in V such that $\|v_c = \lambda v_1 + (1 - \lambda)v_2\|$. Let $\|w_c\| = \|v_c + \alpha\gamma\|$, where $\alpha \in [0, 1]$. Now we can write,

$$\|v_c + \alpha\gamma\| = \|\lambda v_1 + \alpha\lambda\gamma + (1 - \lambda)v_2 + \alpha(1 - \lambda)\gamma\| \quad (6.45)$$

$$= \|\lambda(v_1 + \alpha\gamma) + (1 - \lambda)(v_2 + \alpha\gamma)\| \quad (6.46)$$

Hence, w_c is expressed as a convex sum of two elements in the set $\bigcup_{v \in V} \text{Vert}(B_\gamma v)$. Thus, $w_c \in \mathbb{H}$.

We will now show that the algorithm 6.1 terminates in a finite number of steps and gives the $R_{TS}(B_\delta(x_0), \gamma)$ as output.

Theorem 6.9.2 *Given the input $(D\Sigma, V, V_0, T, S, \epsilon, \alpha)$ to the Algorithm 6.1, the algorithm terminates in a finite number of steps and outputs the set $R_{TS}(B_\delta(x_0), \gamma)$.*

Proof: The algorithm terminates whenever the current time is greater than T and the current space point is outside S . Since the time and space points increment in fixed steps the algorithm terminates in a finite number of steps for these conditions. However, if the ϵ reach set intersects both V and V^C then the algorithm restarts computation with reduced δ and γ . This has the potential of running into infinite loops with δ and γ reducing to arbitrarily small value.

If there exists a time $\tau_1 < T$ or a space point $x_1 \in S$ such that, the trajectory moves to V^C then from Lemma 6.8.1 there exists a $\delta_1 > 0$ such that $B_{\delta_1}(\eta(nh_x, mh_t)) \subset V^C$ for some n and m . Let $t_a = nh_x$ and $x_a = mh_t$. Now for this t_a and x_a , from Lemma 6.7.2 there exists a γ_1 such that $R_{t_a x_a}(B_{\delta_1}(V_0), \gamma_1) \subseteq B_r(\eta(t_a, x_a))$.

Now if $t_f = \min(\tau_1, T)$ and $x_f = \min(x_a, S)$, then from Lemma 6.7.2 there exists $\delta_2 > 0$ and $\gamma_2 > 0$ such that $R_{t_f x_f}(B_{\delta_2}(V_0), \gamma_2) \subseteq R_{t_f x_f}(V_0, \epsilon)$. If we assume that $\delta = \min(\delta_1, \delta_2)$ and $\gamma = \min(\gamma_1, \gamma_2)$, then there is a $\delta > 0$ and a $\gamma > 0$ such that the algorithm does terminate and outputs the ϵ reach set.

6.10 Algorithm for computing ϵ reach set for multiple location DTL1STHA

The lemma 6.8.1 states that if the trajectory exits a given domain V then there exists a suitable $\delta > 0$ such that the reachable states from the δ neighborhood of the initial states, computed using the convex hull methodology, exits the domain V . This fact along with the assumption that the transitions are all deterministic and transversal can be used to determine transitions between different locations. If a single location DTL1STHA exits a domain V at a time $\tau_1 < T$ or space $x_1 < S$ then the Algorithm 6.1 gives us an approximate time and spatial location at which the exit occurred and the ϵ reach set. This fact can be utilized to find transitions for the more general case of the DTL1STHA.

The underlying idea is to use the algorithm 6.1 to find exit condition from an invariant set Inv_0 . In the process the δ and γ parameters are changed. On transition an updated reachable set is obtained, which can intersect multiple invariant sets of different locations. For a deterministic transition the DTL1STHA can only transit to one location. We prove subsequently, that by proper adjustment of the δ and γ parameters it is possible to obtain an ϵ reach set such that the reach set intersects only a single invariant set.

The algorithm is similar to the algorithm 6.1, where it starts with the computation of images from the δ neighborhood of the initial configuration. The image computation procedure follows the same convex hull methodology. The only difference is that now when the reach set is not in the invariant set of the current location then the algorithm has to decide on the new location and the new initial configu-

ration. The algorithm adjusts the values of δ and γ , to compute the reach set on transition, the new location after transition, and the time and spatial location of transition. The new reach set is then considered as the initial state of the new location and the computation continues.

Hence for the algorithm to function properly we need to find out the time τ and the spatial location x at which a transition takes place and the value of the trajectory $\eta(\tau, x)$ at the transition point. In the next lemma we will show that, it is possible to approximate the time of transition and the value of the trajectory with a given error bound.

Lemma 6.10.1 *Given that at $\tau_1 < T$ and $x < S$ the trajectory $\eta(\tau_1, x) \in Inv_0^\square$ satisfies deterministic transversal transition conditions, there exists a $\delta > 0$ such that $B_{2\delta}(\eta(\tau_1, x)) \subset Inv_0 \cup Inv_1$ and also there exists Δ_t and Δ_x such that -*

1. $\eta(t, x) \in Inv_1^\square$ for $t \in [\tau_1, \tau_1 + \Delta_t]$ and $x \in [x, x + \Delta_x]$
2. $\bigcup_{\eta \in Tr_{0 \rightarrow 1}} R_{[\tau_1, \tau_1 + \Delta_t], [x, x + \Delta_x]}(\eta) \subset Inv_1^\square$, where, $Tr_{0 \rightarrow 1} = B_\delta(\eta(\tau_1, x)) \cap Inv_0^\square \cap Inv_1^\square$

Proof: Let us consider that $Inv_0 \cap Inv_1 \cap Inv_2 \neq \phi$. Since the transition is deterministic, if $\eta(\tau_1, x) \in Inv_0 \cap Inv_1$ then $\eta(\tau_1, x) \notin Inv_0 \cap Inv_2$. Hence, $\eta(\tau_1, x) \notin Inv_2$. Now since Inv_2 is assumed to be compact then there exists a $\delta' > 0$ such that $B_{\delta'}(\eta(\tau_1, x)) \notin Inv_2$. Hence, $B_{\delta'}(\eta(\tau_1, x)) \in Inv_0 \cup Inv_1$.

Since, $\eta(\tau_1, x)$ satisfies the deterministic transversal transition condition at Inv_0^\square , then there exists a unit normal vector \hat{n}_1 out of Inv_0^\square such that $\langle \eta(\tau_1, x), \hat{n}_1 \rangle >> 0$. If we consider $\frac{\delta\eta}{\delta t} = \max((b_0/a_0)(\frac{\delta^2\eta}{\delta x^2}) + (c_0/a_0)(\eta_{max}) + u_0/a_0, (b_1/a_1)(\frac{\delta^2\eta}{\delta x^2}) + (c_1/a_1)(\eta_{max}) + u_1/a_1)$, and $\frac{\delta^2\eta}{\delta x^2} = \max((a_0/b_0)(\frac{\delta\eta}{\delta t}) - (c_0/b_0)(\eta_{min}) - u_0/b_0, (a_1/b_1)(\frac{\delta\eta}{\delta t}) - (c_1/b_1)(\eta_{min}) - u_1/b_1)$, then there exists a $\delta'' > 0$ such that, for all $\eta(t, x) \in B_{\delta''}(\eta(\tau_1, x)) \cap Inv_0 \cap Inv_1$ $\langle \eta(\tau_1, x), \hat{n}_1 \rangle >> 0$. Now for these values of $\frac{\delta\eta}{\delta t}$ and $\frac{\delta^2\eta}{\delta x^2}$ if we select $\delta = \min(\delta', \delta'')$ and $\Delta_t = \delta/2, \Delta_x = \delta/2$ then both the claims satisfy.

The above lemma 6.10.1 gives a methodology to over-approximate $\eta(t, x)$, where a δ neighborhood of $\eta(t, x)$ is intersected with the Inv_0 and Inv_1 . Given such an appropriate $\delta > 0$, an appropriate δ_0 neighborhood of $v_0 = \eta(0, 0)$ can also be found such the reach set at time t and space location x is within a δ neighborhood of $\eta(t, x)$.

DTL1STHA bounded time and space reachability analysis algorithm

The Lemma 6.8.1, the resulting Algorithm 6.1, along with the assumption that the transitions are all deterministic and transversal can be used to augment Algorithm 6.1 to use it for multi-mode DTL1STHA. Using Algorithm 6.1 an updated reachable set can be obtained on transition, which can intersect multiple invariant sets of different modes. However, a DTL1STHA can only transit to one mode. We prove subsequently, that by proper adjustment of the δ and γ parameters it is possible to obtain an ϵ reach set using Algorithm 6.1 such that the reach set intersects only a single invariant set on transition.

In this regard, we need to find out the time τ and the spatial location x at which a transition takes place and the value of the trajectory $\eta(\tau, x)$ at the transition point. In the next lemma we show that, it is possible to approximate the time of transition and the value of the trajectory.

The lemma 6.10.1 gives a methodology to over-approximate $\eta(t, x)$ if it satisfies the deterministic transversal transition condition. It takes a similar approach as Lemma 6.8.1 to estimate the state at the time τ and space x of transition. It approximates the state $\eta(\tau, x)$ by intersecting the δ neighborhood of $\eta(t, x)$ with the Inv_0 and Inv_1 . The lemma further states that if the reach set from $B_\delta(\eta(\tau, x)) \cap Inv_0^\square \cap Inv_1^\square$ is computed for an appropriate Δ_t time interval and Δ_x space interval then the reach set lies within Inv_1 .

Given an appropriate $\delta > 0$ from Lemma 6.10.1, an appropriate δ_0 neighborhood of V_0 can also be found as guaranteed by lemma 6.8.1, which ensures that the ϵ reach set computed following the lemma 6.6.1 lies within Inv_1 .

Lemma 6.10.2 *If the reachable states during a deterministic transverse transition at time τ_1 and space location x_1 is within a δ neighborhood of $\eta(\tau_1, x_1)$, there exists a δ_0 such that $D_{\tau_1, x_1}(B_{\delta_0}(v_0)) \subset B_\delta(\eta(\tau_1, x_1))$ and $D_{\tau_1, x_1}(B_{\delta_0}(V_0)) \cap Inv_0 \cap Inv_1$ is an over-approximation of $\eta(\tau_1, x_1)$, where $D_{\tau_1, x_1}(B_{\delta_0}(v_0))$ is the reach set computed by Algorithm 6.1. \diamond*

The Lemma 6.10.2 states that if for $\delta > 0$ a transition is detected by computing the reach set using Lemma 6.6.1, as in Algorithm 6.1, a corresponding δ_0 boundary of the initial configuration can be found. The lemma 6.10.2 can then be used in arithmetic induction to prove that if the reach set for any transition k is computed following lemma 6.10.1 then a δ_0 can always be found such that Algorithm 6.1 can capture it. This allows us to use the Algorithm 6.1 with some changes for computing reach set of multi-mode DTL1STHA. The changes include a *Transition* function that can output the new mode, and the new initial configuration to be used for each transition. Further, if the *Transition* algorithm fails to find the new mode, then Lemma 6.10.2 allows it to reduce the value of δ_0 and restart the overall computation with an increased assurance of finding the new mode in the next iteration.

The reachability analysis algorithm for multi-mode DTL1STHA as shown in Algorithm 6.2, takes as input: a) the DTL1STHA $D\Sigma$, b) initial mode l_0 , c) the initial configuration V_0 , d) the time bound T , e) the space bound S , f) the approximation parameter ϵ , and g) the parameter α used to adjust δ and γ . It outputs the reach set R_{set} . The algorithm has the following simple steps:

- a)** From an initial state it starts computing the reach set following Algorithm 6.1. On computation of reach set the algorithm checks whether it is within the invariant set of the current mode (line 13 to 15).
- b)** Whenever a reach set computed is not within the invariant set, the *Transition* algorithm is called. *Transition* Algorithm 6.3 checks whether the reach set is within a unique Inv_i . If it is within an unique Inv_i the algorithm returns the reach set, the new mode, and the new initial configuration for the mode. If not then the algorithm recalculates δ and γ and returns an empty reach set.
- c)** If *Transition* returns a valid transition then the algorithm resumes its computation from the new initial configuration, new mode, current time and space point.
- d)** Else it restarts the computation at time $t = 0$ and space $x = 0$ with the new values of δ and γ .

The reachability algorithm for multi-mode L1STHA is given in the Algorithm 6.2. Now from lemma 6.10.2, we know that there always exists a δ_0 for which every transition is captured by the algorithm *Transition*, hence at some point of time or space the *Transition* function will return a non-empty reach set and the algorithm will proceed in time or space. The complexity of this algorithm is of the order of $O(\frac{T}{h_x} \frac{S}{h_t} \log_\alpha(\delta))$ since there are two loops for time and space and each can be reiterated at the max $\log_\alpha(\delta)$ times.

6.11 Formal Analysis Case-Studies

Given the STHA model and its reachability analysis algorithm, presented in the previous chapter, this chapter describes the application of the formal method in two case studies—BSN safety (Section 6.11), and infusion pump (Section 6.11)—as mentioned in Chapter 2. For each of the case studies we first discuss the CPS design requirements, the STHA formal models and then their reachability analysis.

Algorithm 6.2: AlgoReachSet($D\Sigma, l_0, V_0, T, S, N, \epsilon, \alpha$)

```

1:  $R_{set} = \phi; \gamma = \alpha\epsilon; \delta = \alpha\epsilon;$ 
2:  $v_x = \max_i(\frac{a_i}{b_i} \frac{\delta\eta}{\delta t} |_{max} - \frac{c_i}{b_i} \eta_{min} - \frac{u_i}{b_i});$ 
3:  $v_t = \max_i(\frac{b_i}{a_i} \frac{\delta^2\eta}{\delta x^2} |_{max} + \frac{c_i}{a_i} \eta_{max} + \frac{u_i}{a_i});$ 
4: while ( $R_{set} == \phi$ ) do
5:    $h_x = \gamma/(2v_x); h_t = \sqrt{\gamma/v_t};$ 
6:    $t = 0; P_c = B_\delta(V_0); l_c = l_0; \text{jump} = 0; V_c = V_0; \text{ValidTran} = \text{True};$ 
7:   while ( $\text{jump} < N$ ) do
8:     while ( $t < T$ ) do
9:        $x = 0; D_{t,x}(P_c) = P_c;$ 
10:      while ( $x < S$ ) do
11:         $x = x + h_x;$ 
12:         $D_{t,x}(P_c) = \text{Compute image at } x \text{ from } D_{t,x-h_x}(P_c);$ 
13:        if ( $D_{t,x-h_x}(P_c) \subset \text{Inv}(l_c)$ ) then
14:           $R_{set} = R_{set} \cup D_{t,x}(P_c);$ 
15:        else
16:           $(l_c, V_c, \text{ValidTran}, \gamma, \delta, R_{set}) = \text{Transition}(D\Sigma, l_c, V_c, l_0, V_0, \text{ValidTran}, \epsilon, \alpha, \gamma,$ 
17:             $\delta, D_{t,x}(P_c), R_{set});$ 
18:        if ( $\text{!ValidTran}$ ) then
19:          break;
20:        else
21:           $P_c = V_c; D_{t,x}(P_c) = B_\delta(V_c);$ 
22:           $\text{jump} = \text{jump} + 1;$ 
23:          if ( $\text{jump} > N$ ) then
24:            break;
25:          end if
26:        end if
27:      end if
28:    end while
29:     $t = t + h_t;$ 
30:     $D_{t,x}(P_c) = \text{Compute image at } t \text{ from } D_{t-h_t,x}(P_c);$ 
31:    if ( $D_{t,x}(P_c) \subset \text{Inv}(l_c)$ ) then
32:       $R_{set} = R_{set} \cup D_{t,x}(P_c);$ 
33:    else
34:       $(l_c, V_c, \text{ValidTran}, \gamma, \delta, R_{set}) = \text{Transition}(D\Sigma, l_c, V_c, l_0, V_0, \text{ValidTran}, \epsilon, \alpha, \gamma,$ 
35:         $\delta, D_{t,x}(P_c), R_{set});$ 
36:    if ( $\text{!ValidTran}$ ) then
37:      break;
38:    else
39:       $P_c = V_c; D_{t,x}(P_c) = B_\delta(V_c);$ 
40:       $\text{jump} = \text{jump} + 1;$ 
41:      if ( $\text{jump} > N$ ) then
42:        break;
43:      end if
44:    end if
45:  end if
46: end while
47: end while
48: end while

```


Algorithm 6.3: $(l_c, V_c, ValidTran, \gamma, \delta, R_{set}) = \text{Transition}(D\Sigma, l_c, V_c, l_0, V_0, ValidTran, \epsilon, \alpha, \gamma, \delta, D_{t,x}(P_c), R_{set})$

```

1: for (i = 1 to n) do
2:    $\Phi(i) = \text{Intersection of } D_{t,x}(P_c) \text{ and } Inv(l_i)$ ;
3: end for
4: IntersectStates = find the number of nonzero entries in  $\Phi$ ;
5: if ( $IntersectStates > 1$ ) then
6:    $l_c = l_0; V_c = V_0; \gamma = \alpha\epsilon; \delta = \alpha\epsilon; R_{set} = \phi$ ;
7:   ValidTran = False;
8: else
9:    $l_c = l_i$  such that  $\Phi(i) > 0$ ;
10:   $V_c = Inv(l_i)$ ;
11:  ValidTran = True;
12:   $R_{set} = R_{set} \cup D_{t,x}(P_c)$ ;
13: end if
14: return  $(l_c, V_c, ValidTran, \gamma, \delta, R_{set})$ 

```

BSN Safety Verification

As described in Section 6.11 of Chapter 2, thermal safety of BSN operations is generally characterized by a threshold temperature, which when exceeded causes thermal damage to human skin. The temperature rise on human skin due to heat dissipation in a single BSN node is given by the Penne's bioheat equation [4]. From Equation 5.2 it can be observed that the temperature rise varies over both time and three dimensional space; the time differential of the temperature depends on its spatial gradient. Further, the temperature rise depends on the SAR due to electromagnetic radiation from the BSN node radio. In case of network of sensors, a specific region of human skin may be affected by electromagnetic absorption from multiple sensors. The aggregate effects of electromagnetic radiation on a specific region for multiple sensors can be obtained by adding the individual SAR values. Given such spatio-temporal modeling and analysis requirements we next discuss the STHA modeling.

Table 6.2: Safety Verification Results (HA = Hybrid Automata)

Gap between sensors (m)	Max Skin Temperature Rise Predicted ($^{\circ}\text{C}$)		Verification Results		Elapsed Time (s)
	STHA	HA	STHA	HA	
	0.005	2.41	2.39	Un-Safe	
0.005	2.2	2.19	Un-Safe	Safe	1240
0.005	1.56	1.48	Safe	Safe	520
0.015	2.3	2.24	Un-Safe	Un-Safe	86400
0.015	2.1	1.98	Safe	Safe	1240
0.015	1.4	1.37	Safe	Safe	520

STHA model for BSN Safety verification

The dynamic equation $Eq1$ which governs the temperature rise in state I_1 is same as Equation 5.2. In state N_1 , the sensor power is considered to be zero and $Eq2$ represents the special case of Equation 5.2 with $P_c = 0$. $Eq3$ for the aggregate effect takes the same form of Equation 5.2, but P_c and SAR values are the addition of the power and SAR values from both the sensors. Further, the radiation term also gets added up. Given the STHA model, the formal safety verification is performed in Section 6.11 using the reachability analysis algorithm.

Formal BSN Safety Verification

For the formal verification of STHA models we use the reachability analysis algorithm. We discretize three of the four spatio-temporal dimensions and use the PHaver tool for reachability analysis on the remaining dimension. The tool accepts a hybrid model, a list of forbidden (unsafe) states and an initial set of states and outputs whether the execution of the model will transition to unsafe states. To reduce complexity of the analysis we considered only one dimensional space for this case study. Thus, in our verification we had only two dimensions: 1) time and 2) one dimensional space. We discretized time as discussed in the algorithm and evaluated the continuous dynamics of the hybrid automata in the one dimensional space.

We consider two sensors placed at different distances from each other on human arm and evaluate the thermal effects on human skin along the line connecting their centers. The Penne's equation was simplified to consider only one spatial dimension over which reachability analysis is performed. This reachability analysis is then repeated over several discrete time steps to determine safety of the BSN operation. In any time step if the STHA model reaches unsafe states the analysis is halted and the BSN design is deemed unsafe. The parameters of the Penne's equation are obtained from [81].

In order to show the effectiveness of the STHA modeling with respect to a one dimensional hybrid automata, we develop a one dimensional hybrid automata model of the BSN. It is to be noted that one advantage of STHA is its capability to handle multiple space dimensions. But since we have simplified the case study to have only one spatial dimension, we can use the one dimensional hybrid automata to model BSN safety over space at a fixed time. The hybrid automata in this case will look similar to the STHA model but $Eq3$ will be the same as Eq_1 . This is because the one dimensional hybrid automata model cannot capture the aggregate effects due to network of sensors in the BSN.

The safety threshold temperature is computed to be 39.2 °C from the Henrique Moritz thermal damage parameter [129]. Table 6.2 shows the skin temperature rise predicted by both the STHA and the hybrid model for different distances between the sensors and at different time instants. It further shows the results of the reachability analysis in one dimensional space by the STHA and hybrid automata models. It can be observed from the results that when the distance between the two sensors is 0.005 m both the STHA and hybrid automata model predicts reachability to unsafe states after 86400 seconds of operation. However, STHA model predicts that unsafe states are reached much before at the time instant of 1240 seconds. Whereas the

hybrid model predicts safe operation. Further, it can be observed that for every case the hybrid automata model underestimates the temperature rise as it ignores the aggregate effects. However, STHA predicts a higher temperature rise due to aggregate effects. Thus, the STHA model is successful in capturing the aggregate effects of networked operation in sensors, which a one dimensional hybrid automata fails to do.

Drug infusion using infusion pumps

In this section we show the usage of CPS-DAS for safety analysis of single and multi-channel infusion pumps.

Analgesic Infusion Pump

The analgesic infusion pump control algorithm obtains feedback from the human body and attempts to keep the drug concentration in the blood at a desired value C_d . The drug diffusion model takes the infusion rate as input and estimates the drug concentration in blood (C_p) using a spatio-temporal partial differential Equation 6.47.

$$\frac{\partial d}{\partial t} + \nabla(ud) = \nabla(D(r) \nabla d) + \Gamma(r)(d_B(t) - d) - \lambda d \quad (6.47)$$

where $d(r, t)$ is the tissue drug concentration at time t and distance r from the infusion site, $u(t)$ is the infusion rate, $D(r)$ is the spatially varying diffusion coefficient of the blood, $\Gamma(r)$ is the spatially varying blood to tissue drug transfer coefficient, and $d_B(t)$ is the desired drug concentration level after time t . However, in case of analgesic infusion only the drug concentration at the infusion site is considered important [55]. Hence, the spatial region in the human body is discretized into two compartments: 1) the tissue and 2) the capillary blood. The Equation 6.47 is then simplified to the pharmacokinetic Equation 5.3 (PKA model) proposed in [55], also called the two compartment model. In Equation 5.3, the first two lines represent the diffusion process in the tissue, while the next two lines represent the diffusion

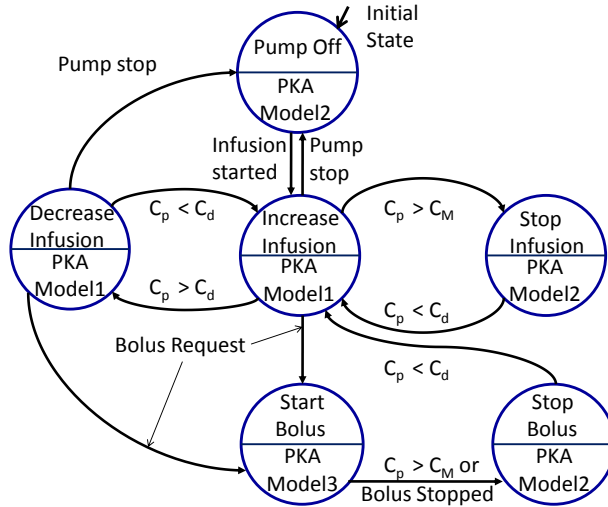


Figure 6.8: Figure shows STHA model of the infusion pump. PKA model denotes Eqn 6.47. The state transitions are spatio-temporal in nature.

process in the blood. Also the equations governing diffusion in these to spatial locations are interrelated and cannot be solved separately. This shows the strong spatial nature of variation of the drug concentration. $u(t)$ is the infusion rate of the infusion pump. The differential equations in the model are time-delayed. This is because they consider time delays related to the infusion input (T_i), cardio-pulmonary transport delay T_p and the arterial, capillary and venous transport delays T_r . The controller first considers an initial infusion rate, which is the default infusion rate provided by the care giver, and estimates the drug concentration C_p using Equation 5.3. If $C_p > C_d$ then the controller increases the infusion rate by an increment δx , else it decreases by the same amount. The controller queries the PKA Model every δt time interval.

Infusion Pump Formal Model

The formal model for the infusion pump control system (Figure 6.8) has six states. Each state is represented jointly by a discrete state in the infusion pump and a variant of the PKA model, which governs the state variables - infusion rate and drug

concentration. Initially the pump is in *Pump Off* state with zero infusion rate and waits for `Infusion Start` event from the care giver. In this state, the variation of the drug concentration follows *PKA Model 2*, which is Equation 6.47 with $d_B(t) = 0$. On the `Infusion Start` event the infusion pump model goes to the *Start Infusion* state, where the infusion rate is set to a default value x_0 . In this state, the *PKA Model 1* is used to estimate the drug concentration C_p , where $d_B(t)$ is a non zero constant. If $C_p < C_d$ then the pump transits to the *Increase Infusion* state where the infusion rate is increased by an amount δx , else if $C_p > C_d$, then the pump transits to the *Decrease Infusion* state, where the infusion rate is decreased by δx amount. These two states also use the *PKA Model 1* to evaluate drug concentration in the blood. In the *Increased Infusion* state if the drug concentration exceeds allowable maximum C_M then the model goes to *Stop Infusion* state where the infusion rate is zero and drug concentration follows *PKA Model 2*. However, if $C_d < C_p < C_M$ then the model transits to the *Decrease Infusion* state and stays there until $C_p < C_d$ again when it comes back to the *Increase Infusion* state. When in *Increase Infusion* or *Decrease Infusion* states, if a `bolus request` event occurs, then the model transits to the *Start Bolus* state. In this state, the drug concentration is governed by *PKA Model 3*, where the infusion rate in Equation 6.47 is a step input for a short duration T_b . When the bolus duration elapses or when $C_p > C_M$ in the *Start Bolus* state the model transits to the *Stop Bolus* state (infusion rate = 0). The model comes back to the *Increase Infusion* state whenever $C_p < C_d$.

Formal Infusion Pump Safety Verification

Five control parameters of the infusion pump are considered in this case study: 1) *control input delay*, the time taken to transmit the control input from the controller to the infusion pump, 2) *set point*, the drug concentration that is required to be main-

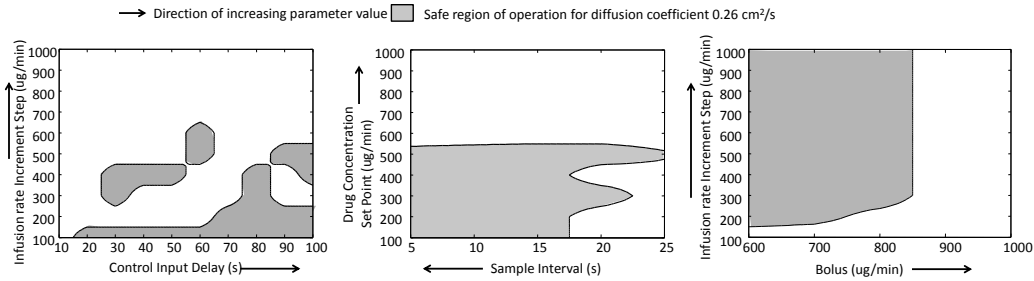


Figure 6.9: Safe and unsafe initial configurations of the infusion pump. The shaded region includes the initial configurations of the infusion pump that result in drug concentration above prescribed safe values at any point in time and space.

tained, 3) *bolus value*, the infusion rate when bolus is requested by the patient, 4) *sampling interval*, the time interval at which the controller samples the pharmacokinetic model, and 5) *infusion increment*, the maximum amount by which the controller can increment the infusion rate. A configuration of the infusion pump is a tuple with numeric values assigned to the above five parameters. In addition to these five control parameters we model the wireless channel PDR using the log normal shadow fading model. On the loss of a control information, the infusion pump maintains the infusion rate at the previous time slot. The reachability analysis as discussed in Section 6.10 was performed on this model and the safe configurations of the infusion pump were obtained as shown in Figure 6.9. In the figure, the five parameters were considered in groups of three with the other two assuming constant values (a total of 100000 simulation runs). The control input delay was varied from 10 to 100 seconds, the infusion increment step from 100 to 1000 ug/min, the set point from 100 to 1000 ug/l, the bolus from 600 to 1000 ug/min, and sampling interval from 5 to 25 s as recommended in the Graseby 3300 technical manual [130]. In each of the figure, the unshaded regions are the unsafe states for the individual with diffusion coefficient $0.26 \text{ cm}^2/\text{sec}$, where the drug concentration exceeded 1300 ug/min. The shaded regions bounded by the bold lines are safe for the assumed values of the control

parameters. The infusion pump failure example described in Chapter 1, can occur due to two reasons: a) choice of infusion pump parameters in the unsafe region, and b) the channel PDR falling in an unsafe region. Both suggest that during the testing phase of the infusion pump, the environmental context driven cyber-physical interaction between the pump and the human body were not considered. Further, the region bounded by the dashed lines show the unsafe configurations for a different individual with a different diffusion coefficient. The overlapping of these two regions indicates that the safe configurations of the pump for one individual is unsafe for the other. These results verify our hypotheses set forth in Chapter 1: a) the consideration of cyber-physical interactions between the human and the BSN is essential, and b) the characterization of interactions has to account for variance across population.

Chapter 7

TACKLING NON-LINEARITIES IN CPS SAFETY ANALYSIS

Nonlinearities are inherent in a CPS since it involves a close interaction of the computing system with the physical environment. Non-linearities impose hurdles in reachability analysis of CPS models. Over the years researchers have concentrated in building the theory of reachability analysis of linear time invariant systems. However, it is typically hard to tackle non-linearities. The principal hurdle is that non-linearities can arise in many different forms in the model. It can be of the form of a multiplicative term of two system parameters if there is internal feedback in the physical system. It can be a power term for example in case of StefanŠs law [131] of radiation. Given a type of non-linearity there may not exist a closed form solution to the input-output relation of the system. Thus there may not be a single reachability analysis algorithm that can handle all non-linearities. This chapter considers one type of nonlinearity found in artificial pancreas and shows how it can be tackled using appropriate linearization and consequent error analysis.

7.1 Non-linear dynamics in Artificial Pancreas

Several models of glucose-insulin interaction have been proposed [132, 133], however, most frequently used is the Bergman's Minimal Model (BMM) [99]. The BMM represents the insulin action process as a set of three differential equations. The BMM provides a relation between the interstitial insulin concentration $X(t)$, blood glucose concentration $G(t)$, and the plasma insulin concentration $I_p(t)$ as given in Equation 7.1 - 7.3.

$$\frac{dX(t)}{dt} = -k_2.X(t) + k_3.(I(t) - I_b), \quad (7.1)$$

$$\frac{dG(t)}{dt} = -X(t).G(t) + k_1.(G_b - G(t)) \quad (7.2)$$

$$\frac{dI(t)}{dt} = -k_4.I(t) + k_5.(G(t) - k_6)^+.t, \quad (7.3)$$

where k_1, k_2, k_3, k_4, k_5 , and k_6 are constants to be determined for a specific patient. I_b is the basal plasma insulin concentration and G_b is the basal glucose concentration in the blood. Further, $G(0) = G_0$, $X(0) = 0$, and $I(0) = I_0$.

The BMM is non-linear in two of the differential equations. The non-linear effect in Equation 7.2 is due to the interaction of the interstitial insulin with the glucose in the blood. The non-linearity in the Equation 7.3 is more complex in the sense that it has a sharp change in slope at time t when $G(t) = h$. These non-linearities make it hard to model using currently available formal analysis tools, which can only perform analysis on linear systems.

7.2 Modeling Artificial Pancreas with Hybrid Automata

Hybrid automata are most suited to model the discrete control algorithms and the continuous human physiological processes in a single formal definition. The hybrid systems have finite number of states to represent operational modes of the discrete control algorithms. Further, each state has an associated set of differential equations, which governs the variation of the system properties in the state. The automata transits from one state to another when the system properties cross thresholds. This model fits the artificial pancreas system perfectly, since the control algorithm also switches states depending upon when the glucose concentration crosses pre-specified thresholds. A Linear Hybrid Automata (LHA), however, assumes that the differential equations are all linear in nature. Theoretical analysis of hybrid automata is complex and often infeasible with the exception of LHAs. A

number of software tools such as PHaver [77], Ptolemy [134], are readily available for performing the reachability analysis of an LHA. Hence, in this thesis we discuss the modeling of artificial pancreas using LHAs. But before going into the details of modeling and analysis, we first formally define an LHA.

Linear Hybrid Automata

An LHA assumes that any system variable η_i is governed by a differential equation of the form of Equation 7.4.

$$\frac{d\eta_i(t)}{dt} = A\eta_i(t) + u, \quad (7.4)$$

where, A and u are constants over time. Formally, an LHA is defined as -

Definition 22 *Linear Hybrid Automata* An n -dimensional Linear Hybrid Automata is a tuple $\{\mathbb{L}, \mathbb{I}, \mathbb{A}, \mathbb{C}\}$ where -

- \mathbb{L} is a set of locations or discrete states of the LHA. A state of an LHA is an element of the set space $\mathbb{L} \times \mathbb{R}^n$.
- $Inv : \mathbb{L} \rightarrow 2^{\mathbb{C}}$ is a mapping from the set of locations \mathbb{L} to a set of cells \mathbb{C} and is called the invariant set for a given location. The set \mathbb{C} consists of cells $C_i \subset \mathbb{R}^n$, such that for any two cells $C_i \cap C_j = \phi$ and $\bigcup_{i \in \mathbb{L}} C_i = \mathbb{R}^n$. The invariant set for a location has the following properties -
 - all the cells within $Inv(l_i)$ for a given location l_i are connected. Two cells are connected when there is a path of adjacent cells in between them. Two cells are adjacent if $\delta C_m \cap \delta C_n \neq \phi$.
 - for any two locations l_i and l_j , $Inv(l_i)^o \cap Inv(l_j)^o = \phi$, where $Inv(l_i)^o$ is the interior of the set $Inv(l_i)$, which does not include the boundary $\delta Inv(l_i)$ of the set and $Inv(l_i)^o \cup \delta Inv(l_i) = Inv(l_i)$.
- $\bigcup_{l_i \in \mathbb{L}} Inv_l = \mathbb{R}^n$

- $A : \mathbb{L} \rightarrow R^{n \times n}$ is a function that maps each location to an $n \times n$ real valued matrix.
- $u : \mathbb{L} \rightarrow R^n$ is a function that maps each location to an n -dimensional real valued vector.

The definition 22 only defines the components of an LHA. But for an LHA to be executed we need to define three entities: a) trajectory at a given location, b) execution over a given time, and c) discrete state transition.

Definition 23 *Trajectory*: The trajectory of an LHA for a location l_i , for a duration $t > 0 \in \mathbb{R}$ is the map $\eta : [0, t] \rightarrow \mathbb{R}^n$ such that -

1. $\eta(\tau)$ satisfies Equation 7.4 $\forall \tau \in [0, t]$ and
2. $\eta(\tau) \in \text{Inv}(l_i) \forall \tau \in [0, t]$

We denote the duration of a trajectory as $\eta.dur$.

Definition 24 *Execution*: The execution α of an LHA from an initial state $(l_0, v_0) \in \mathbb{L} \times \mathbb{R}^n$ for a duration t is a concatenation of finite or infinite number of trajectories $\eta_0 \eta_1 \eta_2 \dots$ such that -

- $\eta_0(0) = v_0$,
- $\eta_{k+1}(0) = \text{Re}(\eta_k(\eta_k.dur))$ for $k \geq 1$, and
- $\alpha.dur = \sum_{\forall k} \eta_k.dur$,

where $\alpha.dur$ is the duration of the execution. $\text{Re}(\cdot)$ is a reset function, which allows resetting the values of the system variables after an execution.

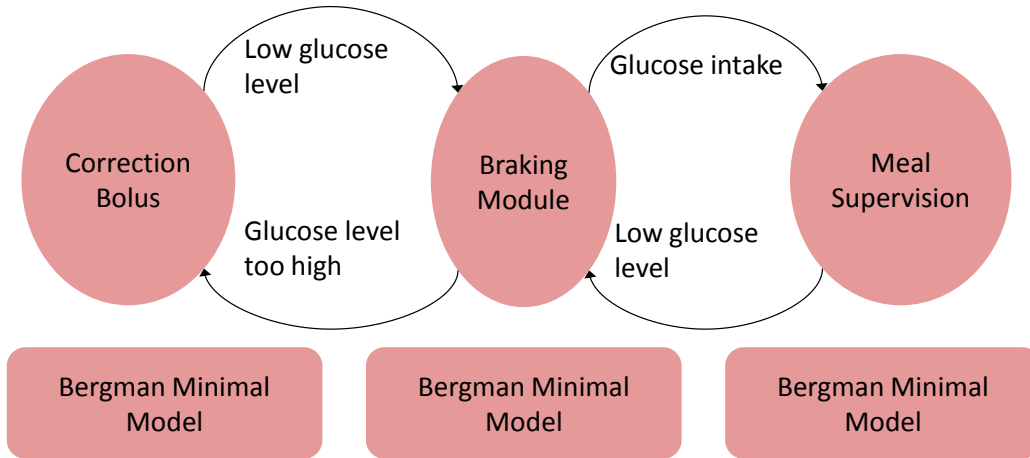


Figure 7.1: Hybrid automata modeling of artificial pancreas with non-linear Bergmann minimal model governing the continuous dynamics.

Definition 25 *Discrete state transition:* For two locations or discrete states $(l_i, l_j) \in \mathbb{L}$ a discrete transition takes place from l_i to l_j at a state $(l_i, v(\tau)) \in \mathbb{L} \times \mathbb{R}^n$ and at time τ if $v(\tau) \in \text{Inv}(l_i) \cap \text{Inv}(l_j)$ and $v(\tau) = \text{Lim}_{\tau' \rightarrow \tau} v(\tau')$ where $v(\tau') \in \text{Inv}(l_i)^o$ for some $\delta > 0$ such that $\tau' \in [\tau - \delta, \tau]$.

We next show how this definition of LHA is directly applicable to artificial pancreas modeling in the following section.

Hybrid Modeling of Artificial Pancreas

To apply the LHA theory to the Artificial Pancreas system, the first step is to linearize the Bergman Minimal Model (BMM).

Linearization of Bergman Minimal Model

To linearize the BMM we consider a small time interval $h > 0$ such that the changes in the interstitial insulin concentration Δx , blood glucose concentration Δg , and the plasma insulin concentration Δi are small enough to neglect the multiplicative terms $\Delta x \Delta g$ and $h \Delta g$. We then consider an initial time t_0 where we have the measured

values of $X(t_0)$, $G(t_0)$, and $I(t_0)$. We can then replace $X(t) = X(t_0) + \Delta x$ in the BMM equations 7.1 and neglect the multiplicative terms to obtain the linearized equations 7.5.

$$\begin{aligned}\frac{d\Delta X}{dh} &= -k_2\Delta x + k_3\Delta i & (7.5) \\ \frac{d\Delta g}{dh} &= -\Delta x G(t_0) - \Delta g X(t_0) - k_1\Delta g \\ \frac{d\Delta i}{dh} &= -k_4\Delta i + k_5[G(t_0) - k_6]h + k_5\Delta g t_0, \text{ if } \Delta g > k_6 - G(t_0) \\ \frac{d\Delta i}{dh} &= -k_4\Delta i, \text{ if } \Delta g \leq k_6 - G(t_0).\end{aligned}$$

Artificial Pancreas Hybrid Automata Model

The artificial pancreas hybrid automata (APHA) can consist of 12 states in a hierarchical design. At the top level, we can view the APHA as a three state hybrid automata as shown in Figure 7.1. Each of the three states exactly corresponds to the three discrete states of the control algorithm. The transition between these three states are governed by threshold crossing by the glucose concentration. Each of the three top level states can be divided into two states corresponding to the two differential equations for the plasma insulin concentration. The transition between these two states are governed by thresholds on the glucose concentration given in Equation 7.5. Further, to reduce the approximation errors for the linearization, the time h has to be small enough. Hence, given a small enough h , to apply the linearized equations for estimating states over a long time, the glucose and insulin concentration has to be reset to the current value. Resetting can only be done in an LHA when there is a state transition. Hence, each of the six states in the APHA has to be further divided into two states. The transition between these states is governed by thresholds on the time intervals. The time for which the APHA stays in a particular state is restricted to a small time h .

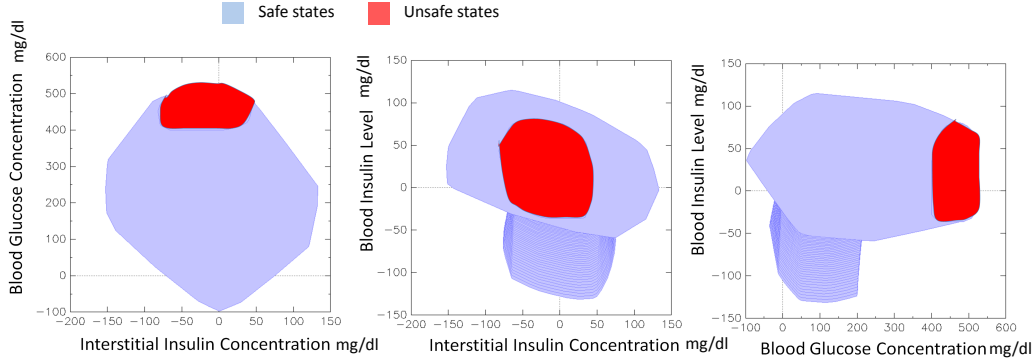


Figure 7.2: Reachable states for the linearized artificial pancreas hybrid automata model.

7.3 Patient Safety Analysis

To analyze patient safety we consider reachability analysis of the linearized hybrid model of the artificial pancreas. Linearization introduces errors in estimation. In this section, we first discuss the reachability analysis and then find an upper bound on the error of the analysis.

Reachability Analysis

Reachability analysis of an LHA from an initial set of states $\{I_0, V_0\} \in \mathbb{L} \times \mathbb{R}^n$, where $V_0 \subset \mathbb{R}^n$, involves finding the set $V^{final} \subset \mathbb{R}^n$ such that $\forall v_0 \in V_0$ and some time $t > 0$ there exists an execution α_{v_0} of duration t such that $\alpha_{v_0}(t) = v_{final}$ for some $v_{final} \in V^{final}$.

In other words, reachability analysis on an LHA results in all the continuous states that can be reached by the LHA at any time starting from a given set of initial states. Hence, when applied to artificial pancreas model it gives all possible values of the interstitial insulin, plasma insulin, and blood glucose concentration that can occur during its operation. Generally, computation of the exact reachable set is infeasible. Hence, an over-approximation of the set is obtained through theoretical

analysis. A number of LHA reachability analysis tools exist in the literature including PHaver [77], Ptolemy [134], but we show here the usage of the PHaver tool.

In the reachability analysis we considered initial states with blood glucose, plasma insulin, and interstitial insulin concentration within a range of $\{G(t), I(t), X(t)\} = \{[60, 180], [0, 100], [0, 1]\}$. The reachable states or the possible values that the parameters can take at any arbitrary time are shown in lighter shade in the Figure 7.2. If the glucose concentration crosses 180 mg/dl then it is considered unsafe which is shown with dark shades in the Figure 7.2. The analysis shows that for some configurations of the AP the glucose concentration reaches unsafe levels to cause hyper-glycemia. The specific set of initial states did not cause any hypo-glycemic cases. Given that we can perform the linearization and subsequent formal analysis we are now interested in the accuracy of the formal analysis in the following section.

Error Bounds on the Linear Approximation

The linearization of the differential equations will lead to error in analysis. It is essential to quantify this error in order to characterize the appropriateness of the linearization. In this section we first derive an upper bound on the approximation error and then argue that the error can be arbitrarily minimized with proper choice of the time interval h .

As required for the linearization, we choose an initial time t_0 and a time interval h . Assuming that Δx , Δi , and Δg are constant in the small h time interval, we integrate the equations 7.5 over the interval time. We denote $X'(t_0 + h)$, $G'(t_0 + h)$, and $I'(t_0 + h)$ as the estimations of $X(t_0 + h)$, $G(t_0 + h)$, and $I(t_0 + h)$, obtained on linear approximation of the differential equations. Then we get Equations 7.6 on integration,

$$\begin{aligned}
X'(t_0 + h) - X(t_0) &= -k_2[X'(t_0 + h) - X(t_0)]h & (7.6) \\
&+ k_3[I'(t_0 + h) - I(t_0)]h \\
G'(t_0 + h) - G(t_0) &= -[X'(t_0 + h) - X(t_0)]G(t_0)h \\
&- [G'(t_0 + h) - G(t_0)]X(t_0)h \\
&- k_1[G'(t_0 + h) - G(t_0)]h \\
I'(t_0 + h) - I(t_0) &= -k_4[I'(t_0 + h) - I(t_0)]h \\
&+ k_5[G(t_0) - k_6]h^2/2 \\
&+ k_5[G'(t_0 + h) - G(t_0)]t_0h
\end{aligned}$$

The estimations $X'(t_0 + h)$, $G'(t_0 + h)$, and $I'(t_0 + h)$ can be found by simultaneously solving the set of three equations 7.6.

Over-approximations $X^o(t_0 + h)$, $G^o(t_0 + h)$, and $I^o(t_0 + h)$ of the parameters $X(t_0 + h)$, $G(t_0 + h)$, and $I(t_0 + h)$ can be determined by over-estimating the derivatives of Equation 7.1. The over estimation is done by neglecting all the negative terms in the derivative as shown in Equation 7.7.

$$\begin{aligned}
\frac{dX^o(t)}{dt} &= k_3(I^o(t) - I_b) & (7.7) \\
\frac{dG^o(t)}{dt} &= k_1(G_b - G^o(t)) \\
\frac{dI^o(t)}{dt} &= k_5(G^o(t) - k_6)t
\end{aligned}$$

Solving the ordinary nonlinear differential equations 7.7 we can obtain an upper bound on the approximation error for estimating the plasma insulin concentration as given in Equation 7.8.

$$\begin{aligned}
I^o(t_0 + h) - I'(t_0 + h) & \quad (7.8) \\
= & (k_5 G_b - k_5 k_6)(2t_0 + h) \frac{h}{2} \\
- & k_5(G_b - G(t_0)) \left[\frac{t_0 - (t_0 + h)e^{-k_1 h}}{k_1} + \frac{1 - e^{-k_1 h}}{k_1^2} \right] \\
- & \frac{k_5(G(t_0) - k_6)h^2/2(1 + k_2 h)[1 + X(t_0)h + k_1 h]}{(1 + k_4 h)(1 + k_2 h)[1 + X(t_0)h + k_1 h] + k_3 k_5 t_0 h^3 G(t_0)}
\end{aligned}$$

Similar equations can be found for the variables $X^o(t)$ and $G^o(t)$ can be found following the similar process. For a chosen h the error in the values of $X(t)$, $G(t)$, and $I(t)$ can be computed from the Equation 7.8. Further, given an error bound the value of h can be chosen from the Equation 7.8 so that the parameter estimation error is within the desired bounds.

Figure 7.3 shows the linearization error with respect to the time discretization interval. The initial blood glucose concentration was varied from 120 mg/dl to 180 mg/dl, the interstitial insulin concentration was varied from 0 U/dl to 1 U/dl in steps of 0.1 U/dl while the insulin concentration was varied from 0 mg/dl to 100 mg/dl in steps of 1 mg/dl. The error curve can be used to obtain the linearization error for a given discretization interval or to determine a discretization interval for a given error. The different lines in the graph indicate error curves for different initial configurations of the AP. The boundary lines limit the maximum linearization error within a band and is a measure of confidence on the results of the formal safety analysis. As shown in this example a suitable $h = 6.82 \text{secs}$ can be selected from this curve so that the error is within 5%.

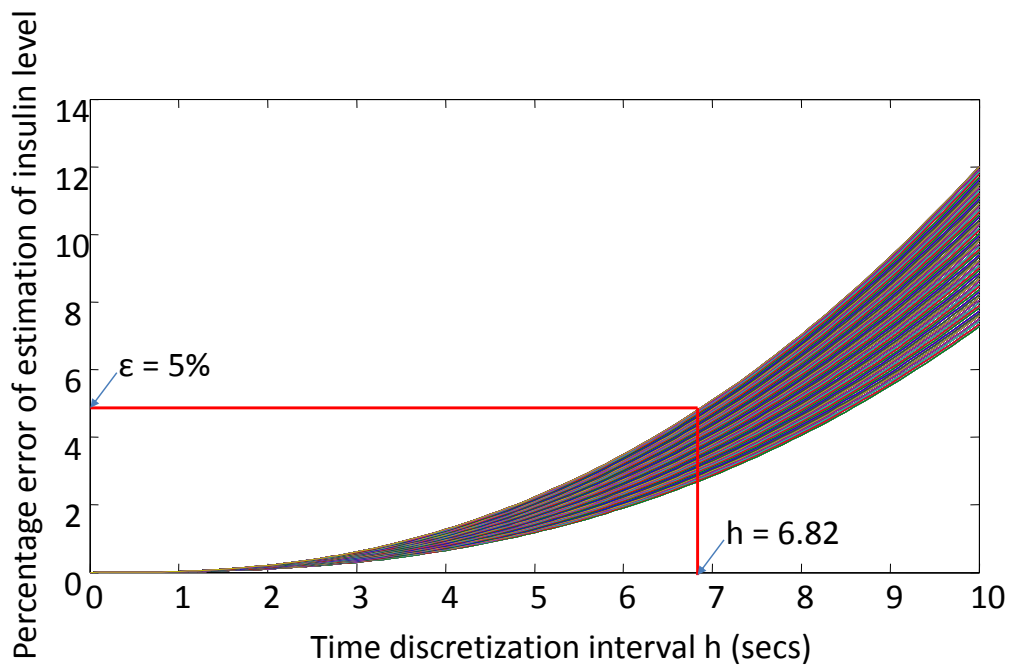


Figure 7.3: Variation of linearization error with respect to the discretization time interval.

Chapter 8

SYNTHESIS OF CPS MODELS

This chapter discusses the contributions and ongoing work on automated synthesis of CPS models.

8.1 Motivation and Related Works

Software synthesis: Automated synthesis of sensor programs has been a major focus of research. However, most of the work have concentrated in either developing APIs or TinyOS specific sensor code generation. There exists several work on API development for TinyOS to support signal processing, intelligent storage, and energy efficient communication [135–137]. However, to use these APIs the user has to write TinyOS code for the sensors, requiring domain specific knowledge. Further, the authors [136, 137] do not consider the interfacing of the sensors with smart phone, which is an important component of a BSN used in a PHMS. The mobile middleware [135] provides basic API for the smart phone to control the sensors. However, this again requires user effort to write the code.

Researchers have also focused on code generation for sensors from a high level specification such as Simulink [137], or graphical specifications [138–140]. The framework proposed in [137] comes closest to *Health-Dev* since they generate code for TinyOS based sensors and can also execute sensor algorithms. However, the supported algorithms are not related to physiological signal processing (required for a PHMS). They are execution sequences of sensor operations such as sensing, communicating, and radio duty cycling. The other works such as RaPTEX [138], Viptos [139], and TOSDev [140] consider the sensor nodes as only a sensing and communication device and does not allow physiological signal processing. Hence,

a framework that can support intuitive high level specification for both sensors and smart phones with OS and hardware independent code generation, and physiological signal processing is missing in the literature. *Health-Dev* fills this gap.

Hardware Synthesis of Hybrid Models: Several efforts have been undertaken for the hardware synthesis of physical processes involving continuous dynamics. Two very common approaches to synthesis are to either develop an Application Specific Integrated Circuit (ASIC) [141, 142] or to use Field Programmable Gate Arrays (FPGAs) [143–145]. In both the approaches there is a common theme of discretization of the continuous dynamics and approximation of the actual physical behavior. However, such discretization leads to errors in the synthesis, which are hard to characterize given the complexity of the continuous dynamics. This research proposes to use the latest reconfigurable analog signal processors (RASPs) to implement the continuous dynamics in STHA. Such a system uses analog signal processor modules to implement continuous signal characteristics and can form the building blocks of any continuous variable, which is governed by a complex differential equation.

8.2 Health-Dev: Model Based Synthesis of BSNs

Health-Dev consists of three modules: 1) specification, 2) parsing, and 3) code generation (Figure 8.1).

Specification Module

At the front end it provides the user with an interface to provide a high level specification of the BSN. In the specification module, the user provides three types of information:

Sensor specification: In *Health-Dev*, the user specifies the computation, and communication requirements for each type of sensed signal. Multiple sensors can be implemented in the same sensor platform depending upon hardware capabilities

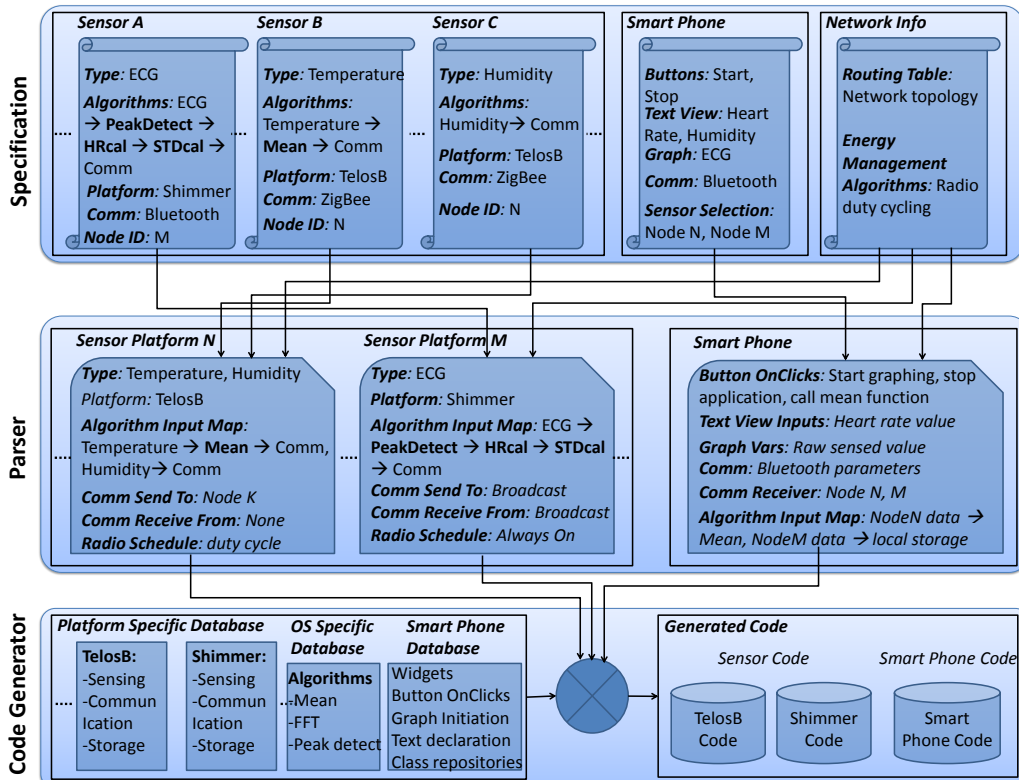


Figure 8.1: Architecture of the *Health-Dev* which has three modules: specification, parser, and code generator.

of the platform. For each sensed signal a separate specification module is maintained, which has two ports: i) input data, denoting the raw sensed data and 2) output data, denoting the data that is to be reported via the wireless network. In this module, three types of information are maintained: 1) Sensor properties: This class of information includes the type of sensor, e.g., temperature or humidity or electrocardiogram (ECG) signals, the sampling frequency of the signals, sensitivity, and the platform type. *Health-Dev* supports most commercially available sensor platforms such as TelosB (xbow.com), Shimmer (shimmer-research.info), BSN v3 (<http://vip.doc.ic.ac.uk/bsn/a1892.html>), and iMote2 (bullseye.xbow.com).

2) Sensor subcomponents: Each sensor has two classes of subcomponents related to the computation performed on the sensed value and communication of data

(Figure 8.2). For computation in a sensor, the *Health-Dev* provides algorithm sub-components. An algorithm subcomponent has an input port and an output port. This subcomponent also has a property to specify the name of the algorithm. A database of algorithms are maintained in the code generator module. If the algorithm name matches any one of the available algorithms then the database code is used in the synthesis. Otherwise, the user has to implement the algorithm in the form of a function in the platform specific programming language and include it in the database. The communication subcomponent has properties to specify the communication protocol, which can be either Bluetooth or ZigBee, multi-hop or single hop communication, and the frequency of packet transmission. This subcomponent also has four ports pertaining to the *source id* from which a packet is received, *received data*, *destination id* to which data has to be sent, and *send data*. Further, there are two additional data subcomponents in the sensor to specify its *node id*, and its one hop neighbor node ids. Sensors with the same node id are implemented in the same platform.

3) Sensor connections: To facilitate specification of algorithm execution sequence and type of information transferred to the network, *Health-Dev* provides two types of connections (Figure 8.2). Using the algorithm connections the user can specify the inputs of each algorithm, which can be either the raw signal or the output of another algorithm. For example, the raw ECG samples from the sensor can be passed through a peak detector, heart rate calculator (HRCal), and heart standard deviation calculator in sequence (Figure 8.1). The output can be transmitted to the smart phone using the communication subcomponent. The communication connections can then be used to specify the transmission and reception parameters.

Network specification: The routing information and communication energy management schemes are specified in this subcomponent. Routing information can be in-

Table 8.1: Specifiable properties in *Health-Dev*

Sensing	Computation	Communication	Phone
1. Node id	1. Physiological processing	1. Routing protocol (Bluetooth or ZigBee)	1. Buttons onclicks
2. Platform	(BSNBench, heart rate	2. destination & source address	2. text view inputs
3. Sensor type	calculator, ECG models)	3. packet contents	3. graphing variables
4. Sampling frequency	2. Execution sequence	4. radio duty cycle	4. execution sequences

cluded in a routing table, a separate file, and the filename can be specified in the routing info field. If a routing table file is provided then the user does not need to specify the destination and source id. The parser will extract those information from the routing table. If none is specified then a default scheme of flooding is employed. For communication energy management, three modes of radio operation are supported - radio is always on, or is only on during transmissions, or duty cycled.

Smart phone specification: *Health-Dev* allows the user to specify the UI of the smart phone base station, the sensor nodes in the BSN that can communicate with the base station, and the algorithms to be run in the base station on the received data. The user can specify buttons, text views, graphs in the smart phone subcomponent. In the button *OnClick* functions *Health-Dev* currently allows only starting and stopping graphs and running basic algorithms such as computation of mean, standard deviation of received data. Text view and graph view inputs are specified using connections in the smart phone subcomponents. Currently in *Health-Dev* only Android based smart phones are considered, which communicate with the sensors using the Bluetooth communication protocol (Table 8.1).

Parser Module

The parser module takes the specification of the BSN as input and generates meta models, which are used in the code generation module for actual code synthesis.

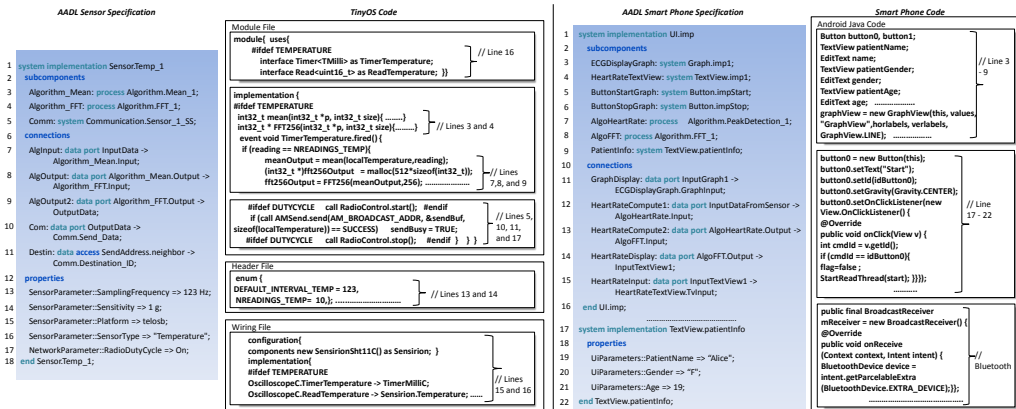


Figure 8.2: AADL specification of a sensor and smart phone and their TinyOS and Java codes

Sensor parser: The main function of this parser is to convert the platform independent specification of the sensors to a platform specific one. The parser extracts the node id and sensor platform information from each sensor specification and groups the sensors with same node id to the same sensor platform and creates a meta model for that platform or “note”. The parser then parses the sensor types and gives platform specific names to the sensors for example for TelosB platform a temperature sensor is named as `Sensirion Sht11`. For each algorithm the parser extracts the appropriate filename from the platform specific database and includes it in the meta model. It then parses the connections in the specification and for each algorithm extracts the input map. For example, the input to the mean function in sensor platform N in Figure 8.1, are the temperature readings and the number of readings. For each communication specification the parser extracts the destination node id, data to be sent, source node id, data received, and routing information, and includes them in the meta model.

Smart phone parser: The smart phone parser in *Health-Dev* is made specific for Android development and is primarily concerned with setting up the UI. The UI in Android is an XML specification with specific keywords defined in the Android Java

widget. The smart phone parser reads the UI specification and converts it to the corresponding XML specification.

Code Generation Module

The code generation module takes the parser output and uses code databases to generate code for sensor platforms and smart phone. There are three types of code databases:

Sensor platform specific database: The sensor platform program generally follows an event driven paradigm, where there are event handler functions for sensing and communication. In these event handlers, computation tasks can be posted for execution after the processing of the events. This is especially true for TinyOS, which is the most popular OS used in sensor platforms. Further, the OS has several platform independent hardware abstraction modules that can be used to handle sensing and communication hardware. These abstractions are then wired to appropriate hardware units to generate executable code. In the database, for each platform, a basic module file for handling all available type of sensing and communication hardware and their event handlers are stored. The different types of sensing and communication protocols are differentiated using preprocessing directives. Similarly a wiring file is generated with all possible hardware modules differentiated by pre-processing directives.

OS specific database: A collection of algorithms related to physiological signal processing and health data statistics analysis are kept in this data base. The algorithms include the BSNBench suite [146], a benchmark specifically designed for BSNs consisting of FFT, peak detection, and statistics operation. Further, it includes physiological signal specific algorithms such as heart rate calculator from ECG signal or photo-plethysmogram pulse width calculator, signal normalization, correlation computation.

Smart phone database: This database consists of parameterized declaration and call back codes written in Java for Buttons, Text View, and Graph Views. The Java widgets provided by Android are also kept in the smart phone database.

Chapter 9

CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

9.1 Conclusions

CPSes are increasingly becoming pervasive and are enabling critical operations in systems providing improved health care, smart-spaces, green and cost effective amenities. To enable wide acceptance of CPSes, their safe, secure and sustainable operation has to be ensured. The tight coupling between computing units and physical environment in CPS when used intelligently can assure safe, secure and sustainable systems. However, the complex nature of the interactions impose several hard challenges in CPS design. This thesis focuses on the problem of assuring safety of the physical environment under the operation of CPSes. The thesis considers the interaction of the CPS software with of dynamically changing physical environment and proposes analytical and simulation based techniques to estimate harmful effects of the interaction. The main contributions of the thesis are: a) CPS-DAS tool for simulation analysis of interaction safety, b) Spatio-Temporal Hybrid Automata for formal verification of interaction safety, c) a tractable algorithm for analyzing interaction safety under dynamic contexts, and d) *Health-Dev* tool for automatic synthesis of healthcare CPS software from models.

9.2 Immediate extensions

Research in the area of MBE of CPSes has recently received considerable attention and offers several unsolved problems. This thesis attempts to solve some of the important ones related to safety assurance of CPSes and in the process opens up new venues of research. Some of these problems are part of ongoing research which will be discussed first. This section discusses the extensions that are being

currently worked on. The extensions include the development of a tool for modeling, analysis or verification, and development of CPSes and a technique for synthesizing the STHA models of CPSes in hardware.

Integrated Tool for CPS Safety Analysis, Verification, and Design

An immediate future extension of this thesis, is the development of a tool that takes intuitive architectural specification of CPSes using the constructs discussed in Chapter 4, and enables both simulation and reachability analysis for safety verification. However, this will require an automated conversion of architectural models into formal models. Such model conversions are non-trivial and needs to carefully consider boundary conditions.

Each LCPS in the GCPS model can be represented with a STHA model. The state parameters for the GCPS model can be derived from the specification of the computing unit (discrete variables) and ROIm and ROIn (dynamic equations). State transitions will be affected by events in the computing domain or due to intersection of the ROIn or ROIm. Separate states will be used to specify cumulative effects of interactions. Transitions to those states will be governed by events generated in the spatial domain (intersection of ROIn or ROIm). However, the following problems arise with such specification:

1. **Determination of region boundaries:** The region boundaries can be derived from the physical dynamics of the human body specified in the ROIm. However, in most applications of CPSes the variation of the monitored parameter (determined by the physical dynamics) varies asymptotically in space. In that case the region boundaries are infinite. The region boundaries can be limited in extent by setting thresholds on the variation of the monitored parameter. For example, in case of thermal side-effect evaluation a low threshold

on the human tissue temperature rise can be imposed such that temperature variations below this threshold is neglected. These thresholds can be used as boundary conditions for the physical dynamics equation to compute the region boundary. An alternative to this approach is the specification of the region boundary equation in the model itself. For example, the user considers the thermal effects within a circular region centered at the medical device location. However, this may lead to inaccuracies in the estimation of the effects of the interactions.

2. Computation of aggregate effect: Aggregate effects of interactions can be evaluated by considering the overlap in the ROIs and ROIs and combining the physical dynamic equations for the two regions. For special types of region boundaries (square, circle) closed form equations for determination of overlap regions is readily available. But in real life scenarios, region boundaries can have complex shapes and the determination of overlapping regions can become mathematically rigorous.

Synthesizing STHA Models in Hardware

Hardware synthesis of STHA models is beneficial due to two important reasons:

1. Simulation of the differential equations governing the physical processes in hardware is usually orders of magnitude faster than software.
2. CPSs generally consist of several computing systems interfaced together to form a complex system. A hardware synthesis of the physical system will enable experimental testing of individual components of the CPS by interfacing them with the implemented physical systems.

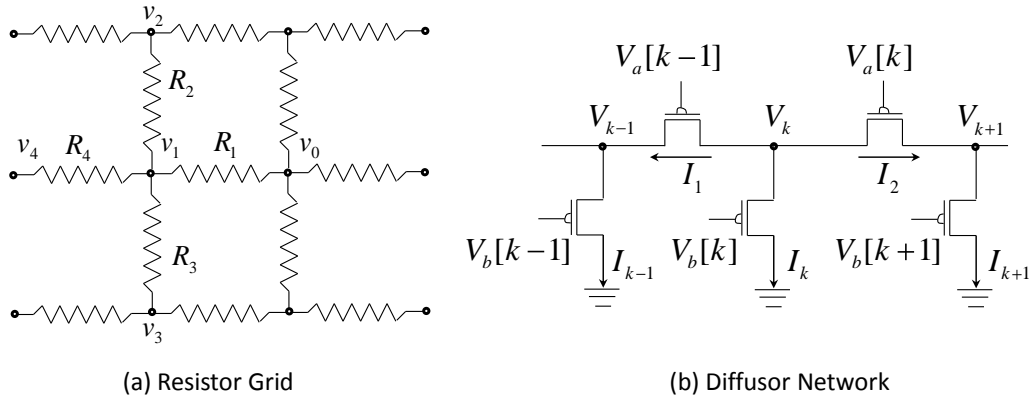


Figure 9.1: Resistor grid and diffuser network synthesis of second order differential.

Research in this area is ongoing where diffuser networks are used for the synthesis of differential equations representing the physical system. A diffuser network can be conceptually viewed as a grid of resistors, as shown in the Figure 9.1. A given 2nd order derivative $\frac{\delta^2 V}{\delta x^2}$ can be discretized using the finite difference time domain method [147] and represented as -

$$V^{m+1}(i, j) = K \times [V^m(i + 1, j) + V^m(i, j + 1) + V^m(i - 1, j) + V^m(i, j - 1)], \quad (9.1)$$

where the time is discretized into slots indexed by m , the x and y dimensions are discretized with slots indexed by i and j , respectively and K is a constant. This differential can be easily represented in the resistor grid if we consider the parameter $V^{m+1}(i, j)$ as voltages $v_0, v_1, v_2 \dots v_n$ at appropriate nodes in the resistor grid (Figure 9.1(a)). Applying Kirchoff's law to the resistor grid we get,

$$v_1 = \frac{1}{1/R_1 + 1/R_2 + 1/R_3 + 1/R_4} \times \left(\frac{v_0}{R_1} + \frac{v_2}{R_2} + \frac{v_3}{R_3} + \frac{v_4}{R_4} \right), \quad (9.2)$$

which is similar to the equation 9.1. The value of the resistors can be properly calibrated based on the constant K .

The diffuser network uses CMOS circuits operated in collector mode and can efficiently emulate the resistor grid circuit. Further, due to the floating gate nature of the CMOS devices variable capacitors can also be simulated hence enabling a more accurate representation of differentials. As shown in the Figure 9.1 (b) the input current at the node V_k is given by [148] -

$$I_{in_k} = -\frac{CV_T}{I_k} \frac{dI_k}{dt} - I_{k-1} e^{-\kappa/V_T(V_{a_{k-1}}-V_{b_{k-1}})} - I_{k+1} e^{-\kappa/V_T(V_{a_k}-V_{b_{k+1}})} + I_k \quad (9.3)$$

$$+ I_k e^{-\kappa/V_T(V_{a_k}-V_{b_k})} + I_k e^{-\kappa/V_T(V_{a_{k-1}}-V_{b_k})}$$

Here I_k s are the source currents of the CMOSs and I_{in_k} is the drain current at node V_k . κ is a constant depending on the semiconductor device and V_T is the threshold voltage, and V_a and V_b are the gate voltages of the corresponding nodes. Interestingly, the z transform of equation 9.3 has the same form as that of a second order derivative.

With this theory as background, connections of diffuser networks and resistor grids are being used to implement the differential equations representing physical dynamics in hardware.

9.3 Future research directions

This section discusses the future research directions in each of the three stages of MBSV that are considered in this thesis.

Modeling Stage

This thesis considers modeling the cyber-physical interactions using hybrid models. However, it makes important assumptions that the computing unit is the principal active system in the CPS and it responds to context changes in the physical environment. When such assumptions are lifted interesting research problems may arise.

1. Modeling Interaction with Non-Passive Physical Systems

Most of the case studies considered in the thesis, represent the physical system as a passive entity expressed by differential equations. This assumption is often not true. Consider the example where a sensor in a BSN that measures physiological values from the human body and transfers it to a the base station is implanted in the tissue. Let the ROI_n be defined as the communication range of the sensor node and the ROI_m be defined as the area of the surrounding tissue that receives thermal energy from the sensor due to its heat dissipation. Implantation often leads to growth of tissue around the computing unit resulting in a change in the ROI_m of the system [149]. However, this phenomenon leads to a change in the electromagnetic environment of the worker node thus altering, maybe reducing, its communication capabilities or affecting the ROI_n. The radio power control algorithm will then increase the transmission power to increase the communication range. Thus the radio will now heat the surrounding tissue even more effectively leading to thermal runaway. Thus growth in the physical system, in this case human body, makes it an active component. There is no known differential equation model for such effects. Hidden markov model or neural networks may be used to model such active behavior, but such models are not specifiable in CPS-DAS. Thus, modeling the interaction with computing units and an active physical system in a CPS is an open research problem.

2. Hierarchical Formal Models for Dynamic Contexts

Dynamic contexts are probabilistic in nature. A formal approach to modeling dynamic context may thus necessitate a probabilistic timed automata (PTA) [150] as shown in the Figure 9.2. For example, mobility of an user can be modeled using PTAs. Each state in the PTA represents the location context of the user. The time spent in a particular state is determined by different random distributions. Depending on the nature of the individual certain excursions will be deterministic. For example,

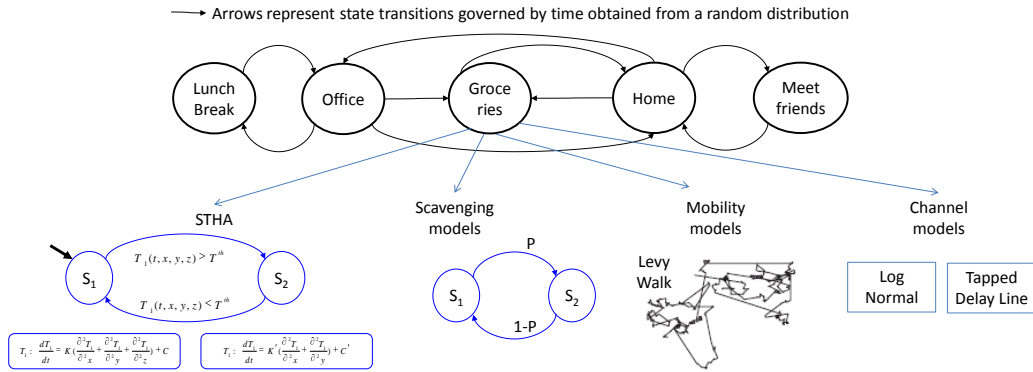


Figure 9.2: Probabilistic timed automata composed with mobility models, STHA models, and scavenging source models.

the user has to leave home at 8 am to get to his office or during lunch the user might go to his favorite place often. For states such as meet friends the location of the user may follow the Levy walk probability distribution as suggested in [87]. Hence context models are discrete while the models of physiology are continuous. A hybrid automata approach would intuitively sound feasible for modeling the cooperation. However, the state transition relations for a hybrid automata do not depend on discrete events. Instead they are governed by system parameter variation according to the continuous dynamics. In order to model the effect of dynamic contexts and cyber-physical interactions, composition relations of the context models and physical models have to be defined. In this regard, hierarchical formal models as shown in the Figure 9.2, can be investigated. In such a model, a probabilistic timed automata representation of the context will be considered. In each state, the human physiology can be represented using hybrid automata. Given the spatio-temporal nature of human physiology, a spatio-temporal hybrid automata can be used. Hence, each state in the finite state automata will have a corresponding STHA to represent the effect of dynamic contexts on the cyber-physical interaction. Such models are com-

plex and theoretical analysis is not guaranteed to be feasible. Nevertheless this is an important area of research.

Analysis Stage

3. Extending STHA Reachability Analysis to Multidimensional Space

The STHA reachability analysis presented in the thesis is only for a single dimensional space. An important extension is to convert this algorithm for multidimensional space. However, such extensions may not be a direct corollary of the one dimensional case. If the partial differential equation expressing the physical dynamics is such that the variation in each spatial dimension is independent of the other, then replicating the spatial image computation technique in Algorithm 6.2 for the other two dimensions will suffice. Otherwise, a closed form approximation similar to the Equation 6.15 for the one dimensional case will have to be derived. The discretization parameters for each dimension will have to be chosen separately such that the ϵ approximation still holds. Non-linear spatio-temporal dynamics further aggravates the problem since non-linearities may arise in several forms necessitating solutions targeting special forms of differential equations.

Synthesis Stage

4. Online verification

The analysis and verification technique proposed in this thesis are all offline and performed before implementation of the CPS. Moreover, the verification is done assuming a range of input parameters that can occur a real setting. The analysis or verification results are not valid if the input parameters are out of the assumed range. However, the physical environment is random and may have unexpected variations which may not be taken into account in the verification stage. In such a scenario, a static design of the system may not have the same safety properties as promised in

the verification stage. Hence, different inputs to the CPS has to be reverified for the new set of physical properties. Thus an online verification approach is necessary, which takes the current physical parameters of the CPS and verifies each control input before applying it to the system on-the-fly.

Online verification has received recent focus in the domain of transportation and medical CPS [38, 151]. Researchers have demonstrated online verification using linear hybrid automata in theory and simulation but not in practice. However, several fundamental research issues need to be solved before online verification can be realized in practice.

1. Is online verification of more complex models of CPS such as STHA feasible?

Online verification requires fast execution of the synthesized formal models. The complexity of the execution of STHA model depends on the desired accuracy or over estimation and can be time consuming for very accurate results.

2. What type of synthesis of formal model will make online verification feasible?

There can be two types of synthesis: a) in software, and b) in hardware. Synthesis in software can be added as a module to the existing software of the CPS, however its execution will be much slower. Synthesis in hardware has a faster execution but it can make the CPS bulky, which can be a major concern for applications such as UAV or medical devices.

BIBLIOGRAPHY

- [1] J. A. Paradiso et al, "Energy scavenging for mobile and wireless electronics," *Pervasive Computing, IEEE*, vol. 4, no. 1, pp. 18–27, Jan.-March 2005.
- [2] A. Banerjee, S. Kandula, T. Mukherjee, and S. K. S. Gupta, "Band-aide: A tool for cyber-physical oriented analysis and design of body area networks and devices," *ACM Trans. Embed. Comput. Syst.*, vol. 11, no. S2, pp. 1–29, 2012.
- [3] A. Banerjee, K. Venkatasubramanian, and S. Gupta, "Challenges of implementing cyber-physical security solutions in body area networks," in *BodyNets '09: Proceedings of International Conference on Body Area Networks*, 2009.
- [4] H. H. Pennes, "Analysis of tissue and arterial blood temperature in the resting human forearm," in *Journal of Applied Physiology*, vol. 1.1, 1948, pp. 93–122.
- [5] K. Venkatasubramanian, G. Deng, T. Mukherjee, J. Quintero, V. Annamalai, and S. K. S. Gupta, "Ayushman: A Wireless Sensor Network Based Health Monitoring Infrastructure and Testbed," in *Distributed Computing in Sensor Systems*, July 2005, pp. 406–407.
- [6] I. Korhonen, J. Parkka, and M. Van Gils, "Health monitoring in the home of the future," *Engineering in Medicine and Biology Magazine, IEEE*, vol. 22, no. 3, pp. 66–73, May-June 2003.
- [7] R. Paradiso, G. Loriga, and N. Taccini, "A wearable health care system based on knitted integrated sensors," *Information Technology in Biomedicine, IEEE Transactions on*, pp. 337–344, Sept. 2005.
- [8] V. Hartkopf, V. Loftness, A. Mahdavi, S. Lee, and J. Shankavaram, "An integrated approach to design and engineering of intelligent buildings—the intelligent workplace at carnegie mellon university," *Automation in Construction*, vol. 6, no. 5-6, pp. 401 – 415, 1997. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0926580597000198>
- [9] A. Haywood, J. Sherbeck, P. Phelan, G. Varsamopoulos, and S. Gupta, "A sustainable data center with heat-activated cooling," in *Thermal and Thermomechanical Phenomena in Electronic Systems (ITherm), 2010 12th IEEE Intersociety Conference on*, june 2010, pp. 1 –7.
- [10] T. Mukherjee and S. K. S. Gupta, "Mcma+cret: A mixed criticality management architecture for maximizing mission efficacy and tool for expediting certification of uavs," in *IEEE Workshop on Mixed Criticality: Roadmap to Evolving UAV Certification, CPSWeek'09*.

- [11] B. K. et al, "Control to range for diabetes: Functionality and modular architecture," *Journal of diabetes Science and Technology*, 2012.
- [12] T. N. I. T. Research and D. Program, "Different definition of cyber physical systems." [Online]. Available: http://www.nitrd.gov/about/blog/white_papers/16-Importance_of_Cyber-Physical_Systems.pdf
- [13] Y. Bar-Yam, *Dynamics of Complex Systems*, ser. Studies in Nonlinearity. Westview Press, jul 2003. [Online]. Available: <http://www.amazon.com/exec/obidos/ISBN=0813341213/newenglandcompleA/>
- [14] D. Harel, "Statecharts: A visual formalism for complex systems," *Sci. Comput. Program.*, vol. 8, pp. 231–274, June 1987. [Online]. Available: <http://dl.acm.org/citation.cfm?id=34884.34886>
- [15] H. of U.S. Missile Defense Efforts 1945-Present, "Anti ballistic missile," 2012. [Online]. Available: http://www.mda.mil/news/history_resources.html
- [16] FDA, "Maude database," http://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfmaude/detail.cfm?mdrfoi__id=2287727.
- [17] Food and D. Administration, "Fda uses grammatech to analyze recalled medical devices." [Online]. Available: http://www.grammatech.com/products/codesonar/GrammaTech_FDA_Profile.pdf
- [18] M. Whalen, D. Cofer, S. Miller, B. H. Krogh, and W. Storm, "Integration of formal analysis into a model-based software development process," in *Proceedings of the 12th international conference on Formal methods for industrial critical systems*, ser. FMICS'07. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 68–84. [Online]. Available: <http://dl.acm.org/citation.cfm?id=1793603.1793612>
- [19] C. Jean-Charles, B. Eric, and S. Christel, "Model based safety analysis for an unmanned aerial system," *Dependable Robots in Human Environments*, June 2010.
- [20] J. T. Luxhøj and A. Öztekin, "A regulatory-based approach to safety analysis of unmanned aircraft systems," in *Proceedings of the 8th International Conference on Engineering Psychology and Cognitive Ergonomics: Held as Part of HCI International 2009*, ser. EPCE '09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 564–573. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-02728-4_60
- [21] C. Ghezzi, M. Jazayeri, and D. Mandrioli, *Fundamentals of Software Engineering*, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2002.

- [22] Y. Zhang, P. L. Jones, and R. Jetley, "Journal od diabetes science and technology," *A Hazard Analysis for a Generic Insulin Infusion Pump*, pp. 263 – 283, March 2010.
- [23] E. N. Johnson and S. Fontaine, "Use of flight simulation to complement flight testing of low cost uavs," *American Institute of Aeronautics and Astronautics*, 2001.
- [24] S. Kong, Y. Jung, C. David, B.-Y. Wang, and K. Yi, "Automatically inferring quantified loop invariants by algorithmic learning from simple templates," in *Proceedings of the 8th Asian conference on Programming languages and systems*, ser. APLAS'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 328–343. [Online]. Available: <http://dl.acm.org/citation.cfm?id=1947873.1947904>
- [25] V.-K. Kim, T. Chen, and M. Tegetho, "Fault coverage estimation for early stage of vlsi design," in *Proceedings of the Ninth Great Lakes Symposium on VLSI*, ser. GLS '99. Washington, DC, USA: IEEE Computer Society, 1999, pp. 105–. [Online]. Available: <http://dl.acm.org/citation.cfm?id=795678.797355>
- [26] R. Kuhn, Y. Lei, and R. Kacker, "Practical combinatorial testing: Beyond pairwise," *IT Professional*, vol. 10, no. 3, pp. 19–23, may 2008. [Online]. Available: <http://dx.doi.org/10.1109/MITP.2008.54>
- [27] A. Wymore, *Model-Based Systems Engineering*, ser. Systems Engineering Series. Taylor & Francis, 1993. [Online]. Available: <http://books.google.com/books?id=CLgsYC3K2yAC>
- [28] B. Cott, R. Durham, P. Lee, and G. Sullivan, "Process model-based engineering," *Computers and Chemical Engineering*, vol. 13, no. 9, pp. 973 – 984, 1989. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/0098135489870401>
- [29] G. Bruno, *Model-based software engineering*, ser. ITCP Computer Science Series. Chapman & Hall, 1995. [Online]. Available: <http://books.google.com/books?id=L7JQAAAAMAAJ>
- [30] R. Jetley, S. P. Iyer, and P. L. Jones, "A formal methods approach to medical device review," *Computer*, vol. 39, no. 4, pp. 61–67, 2006.
- [31] D. E. Arney, R. Jetley, P. Jones, I. Lee, A. Ray, O. Sokolsky, and Y. Zhang, "Generic infusion pump hazard analysis and safety requirements version 1.0," 2009. [Online]. Available: http://repository.upenn.edu/cis_reports/893
- [32] S. Sankaranarayanan, H. Homaei, and C. Lewis, "Model-based dependability analysis of programmable drug infusion pumps," in *Proceedings of the 9th*

- international conference on Formal modeling and analysis of timed systems*, ser. FORMATS'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 317–334. [Online]. Available: <http://dl.acm.org/citation.cfm?id=2044973.2045003>
- [33] D. Arney, M. Pajic, J. M. Goldman, I. Lee, R. Mangharam, and O. Sokolsky, “Toward patient safety in closed-loop medical device systems,” in *ICCPS '10: Proceedings of the 1st ACM/IEEE International Conference on Cyber-Physical Systems*. New York, NY, USA: ACM, 2010, pp. 139–148.
- [34] A. Karimodini, H. Lin, B. Chen, and T. H. Lee, “Developments in hybrid modeling and control of unmanned aerial vehicles,” in *Control and Automation, 2009. ICCA 2009. IEEE International Conference on*, dec. 2009, pp. 228 – 233.
- [35] S. Bayraktar and G. E. Fainekos, “Hybrid modeling and experimental cooperative control of multiple unammned aerial vehicles,” Tech. Rep., 2004. [Online]. Available: http://repository.upenn.edu/cgi/viewcontent.cgi?article=1022&context=cis_reports
- [36] G. Lafferriere, G. Pappas, and S. Sastry, *O-minimal Hybrid Systems*, ser. Memorandum (University of California, Berkeley. Electronics Research Laboratory). Electronics Research Laboratory, College of Engineering, University of California, 1998. [Online]. Available: <http://books.google.com/books?id=-rWZGwAACAAJ>
- [37] H. Fawzi, P. Tabuada, and S. N. Diggavi, “Secure estimation and control for cyber-physical systems under adversarial attacks,” *CoRR*, vol. abs/1205.5073, 2012. [Online]. Available: <http://dblp.uni-trier.de/db/journals/corr/corr1205.html#abs-1205-5073>
- [38] L. Bu, Q. Wang, X. Chen, L. Wang, T. Zhang, J. Zhao, and X. Li, “Toward online hybrid systems model checking of cyber-physical systems’ time-bounded short-run behavior,” *SIGBED Rev.*, vol. 8, no. 2, pp. 7–10, jun 2011. [Online]. Available: <http://doi.acm.org/10.1145/2000367.2000368>
- [39] Y. Zhao, J. Liu, and E. A. Lee, “A programming model for time-synchronized distributed real-time systems,” in *Proceedings of the 13th IEEE Real Time and Embedded Technology and Applications Symposium*, ser. RTAS '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 259–268. [Online]. Available: <http://dx.doi.org/10.1109/RTAS.2007.5>
- [40] J. C. Willems, “The behavioral approach to open and interconnected systems,” *Control Systems Magazine*, pp. 46–99, 2007.

- [41] D. G. M. Greenhalgh et al, "Temperature threshold for burn injury: An oximeter safety study," *Journal of Burn Care and Rehabilitation*, vol. 25, no. 5, pp. 411–415, 2004.
- [42] S. Coleri, M. Ergen, and T. J. Koo, "Lifetime analysis of a sensor network with hybrid automata modelling," in *WSNA '02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*. New York, NY, USA: ACM, 2002, pp. 98–104.
- [43] L. Junsoo et al, "Modeling communication networks with hybrid systems," *Networking, IEEE/ACM Transactions on*, vol. 15, no. 3, pp. 630–643, June 2007.
- [44] A. Schafer et al, "Conceptual Modeling and Analysis of Spatio-Temporal Processes in Biomolecular Systems," in *Sixth Asia-Pacific Conference on Conceptual Modelling (APCCM 2009)*, ser. CRPIT, S. Link and M. Kirchberg, Eds., vol. 96. Wellington, New Zealand: ACS, 2009, pp. 39–48.
- [45] E. Bartocci et al, "Spatial Networks of Hybrid I/O Automata for Modeling Excitable Tissue," *Electronic Notes in Theoretical Computer Science (ENTCS)*, vol. 194, no. 3, pp. 51–67, 2008.
- [46] R. Ghosh et al, "A query-based technique for interpreting reachable sets for hybrid automaton models of protein feedback signaling," in *Proceedings of the American Control Conference*, June 2005, pp. 4417–4422.
- [47] Y. Qin et al, "Hybrid cellular automata model for railway transportation system and its implementation on GIS," in *IEEE Proceedings of Intelligent Vehicles Symposium, 2003.*, June 2003, pp. 543 – 546.
- [48] NITRD, "High-confidence medical devices: Cyber-physical systems for 21st century health care." [Online]. Available: <http://www.nitrd.gov/Publications/PublicationDetail.aspx?pubid=31>
- [49] A. Banerjee and S. Gupta, "Your mobility can be injurious to your health: Analyzing pervasive health monitoring systems under dynamic context changes," in *Pervasive Computing and Communications (PerCom), 2012 IEEE International Conference on*, march 2012, pp. 39 –47.
- [50] A. Banerjee, S. Verma, P. Bagade, and S. K. Gupta, "Health-dev: Model based development pervasive health monitoring systems," *Wearable and Implantable Body Sensor Networks, International Workshop on*, vol. 0, pp. 85–90, 2012.
- [51] J. Jacobs, "Algorithm for optimal linear model-based control with application to pharmacokinetic model-driven drug delivery," *Biomedical Engineering, IEEE Transactions on*, vol. 37, no. 1, pp. 107 –109, 1990.

- [52] G. Varsamopoulos, Z. Abbasi, and S. K. S. Gupta, "Trends and effects of energy proportionality on server provisioning in data centers," in *International Conference on High performance Computing Conference (HiPC2010)*, Dec. 2010.
- [53] C. Green, Z. Ounaies, and E. Hughesa, "Harvesting energy using a thin unimorph prestressed bender: Geometrical effects," *Journal of Intelligent Material Systems and Structures*, 2005.
- [54] K. Venkatasubramanian, A. Banerjee, and S. K. S. Gupta, "Pska: Usable and secure key agreement scheme for body area networks," *IEEE Transactions on Information Technology in Biomedicine SI Wireless Health*, vol. 14, no. 1, pp. 60–68, Jan 2010.
- [55] D. Wada and D. Ward, "The hybrid model: a new pharmacokinetic model for computer-controlled infusion pumps," *Biomedical Engineering, IEEE Transactions on*, vol. 41, no. 2, pp. 134–142, feb. 1994.
- [56] R. N. Maini, F. C. Breedveld, J. R. Kalden, J. S. Smolen, D. Davis, J. D. MacFarlane, C. Antoni, B. Leeb, M. J. Elliott, J. N. Woody, T. F. Schaible, and M. Feldmann, "Therapeutic efficacy of multiple intravenous infusions of anti-tumor necrosis factor a monoclonal antibody combined with low-dose weekly methotrexate in rheumatoid arthritis," *Arthritis and Rheumatism*, vol. 41, no. 9, pp. 1552 – 1563, 1998.
- [57] "ISO 60601 safety standard," http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm\?csnumber=45605.
- [58] K. Venkatasubramanian, T. Mukherjee, and S. K. S. Gupta, "Caac - an adaptive and proactive access control approach for emergencies for smart infrastructures," *ACM Transactions on Autonomous and Adaptive Systems Special Issue on Adaptive Security*, 2011.
- [59] T. Mukherjee, K. Venkatasubramanian, and S. K. S. Gupta, "Performance modeling of critical event management for ubiquitous computing applications," in *MSWiM '06: Proceedings of the 9th ACM international symposium on Modeling analysis and simulation of wireless and mobile systems*. New York, NY, USA: ACM, 2006, pp. 12–19.
- [60] V. Gehlot and E. B. Sloane, "Ensuring patient safety in wireless medical device networks," *Computer*, vol. 39, pp. 54–60, April 2006.
- [61] N. Cho, J. Yoo, S.-J. Song, J. Lee, S. Jeon, and H.-J. Yoo, "The human body characteristics as a signal transmission medium for intrabody communication,"

- Microwave Theory and Techniques, IEEE Transactions on*, vol. 55, no. 5, pp. 1080 –1086, may 2007.
- [62] U. of Pennsylvania, “Generic infusion pump project,” <http://rtg.cis.upenn.edu/gip.php3>.
- [63] L. Bleris and M. Kothare, “Implementation of model predictive control for glucose regulation on a general purpose microprocessor,” in *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on*, dec. 2005, pp. 5162 – 5167.
- [64] K. Venkatasubramanian et al, “Plethysmogram-based secure inter-sensor communication in body area networks,” *Military Communications Conference, 2008. MILCOM 2008. IEEE*, pp. 1–7, Nov. 2008.
- [65] S. Nabar, A. Banerjee, S. Gupta, and R. Poovendran, “Gem-rem: Generative model-driven resource efficient ecg monitoring in body sensor networks,” in *Body Sensor Networks (BSN), 2011 International Conference on*, may 2011, pp. 1 –6.
- [66] —, “Resource-efficient and reliable long term wireless monitoring of the photoplethysmographic signal,” in *Wireless Health (WH) accepted for publication, 2011 International Conference on*, October 2011, pp. 1 –6.
- [67] A. Natarajan, B. de Silva, K.-K. Yap, and M. Motani, “To hop or not to hop: Network architecture for body sensor networks,” in *Sensor, Mesh and Ad Hoc Communications and Networks, 2009. SECON '09. 6th Annual IEEE Communications Society Conference on*, june 2009, pp. 1 –9.
- [68] J. G. Chase, K. Mayntzhusen, P. D. Docherty, S. Andreassen, K. A. McAuley, T. F. Lotz, and C. E. Hann, “A three-compartment model of the c-peptide-insulin dynamic during the dist test,” *Mathematical Biosciences (15 September)*, 2010.
- [69] T. E. Group, “T1dm simulator,” 2012. [Online]. Available: <http://www.tegvirginia.com/T1DM.htm>
- [70] Y. Ozier, P. Guéret, F. Jardin, J. Farcot, J. Bourdarias, and A. Margairaz, “Two-dimensional echocardiographic demonstration of acute myocardial depression in septic shock,” *Critical Care Medicine*, vol. 12, no. 7, pp. 596 – 599, 1984.
- [71] S. R. LaPlante, N. Aubry, L. Rosa, P. Levesque, B. S. Aboumrad, D. Porter, C. Cavanaugh, and J. Johnston, “Liquid cooling of a high density computer cluster,” [online], 2006.

- [72] Q. Tang, T. Mukherjee, S. Gupta, and P. Cayton, "Sensor-based fast thermal evaluation model for energy efficient high-performance datacenters," in *Intelligent Sensing and Information Processing, 2006. ICISIP 2006. Fourth International Conference on*, 15 2006-Dec. 18 2006, pp. 203–208.
- [73] T. Mukherjee, A. Banerjee, G. Varsamopoulos, S. K. S. Gupta, and S. Rungta, "Spatio-temporal thermal-aware job scheduling to minimize energy consumption in virtualized heterogeneous data centers?" *Computer Networks*, June 2009. [Online]. Available: <http://dx.doi.org/10.1016/j.comnet.2009.06.008>
- [74] G. Varsamopoulos et al, "Energy efficiency of thermal-aware job scheduling algorithms under various cooling models," in *International Conference on Contemporary Computing IC³*, Noida, India, Aug. 2009.
- [75] M. J. Moran and H. N. Shapiro, *Fundamentals of Engineering Thermodynamics, 6th Edition*. Wiley, 2007.
- [76] "Moab grid suite of ClusterResources Inc." <http://www.clusterresources.com/>. [Online]. Available: <http://www.clusterresources.com/>
- [77] G. Frehse, "Phaver: Algorithmic verification of hybrid systems past hytech," in *HSCC*, 2005, pp. 258–273.
- [78] P. W. Tuinenga, *Spice: A Guide to Circuit Simulation and Analysis Using PSpice*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1991.
- [79] G.-M. Elena and M. José, "Argospe: Model-based software performance engineering," in *ICATPN*, 2006, pp. 401–410.
- [80] . Karsai et al, "Model-integrated development of cyber-physical systems," in *SEUS '08: Proceedings of the 6th IFIP WG 10.2 international workshop on Software Technologies for Embedded and Ubiquitous Systems*. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 46–54.
- [81] T. Qinghui et al, "Communication scheduling to minimize thermal effects of implanted biosensor networks in homogeneous tissue," *Biomedical Engineering, IEEE Transactions on*, vol. 52, no. 7, pp. 1285–1294, July 2005.
- [82] V. Prasad et al, "Andes: An analysis-based design tool for wireless sensor networks," in *Real-Time Systems Symposium, 2007. RTSS 2007. 28th IEEE International*, Dec. 2007, pp. 203–213.
- [83] F. C. J. Henriques et al, "Studies of thermal injury: I. the conduction of heat to and through skin and the temperatures attained therein. a theoretical and an experimental investigation," in *Am J Pathol.*, July. 1947, pp. 530–549.

- [84] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *HICSS '00: Proceedings of the 33rd Hawaii International Conference on System Sciences-Volume 8*. Washington, DC, USA: IEEE Computer Society, 2000, p. 8020.
- [85] K. Venkatasubramanian et al, "Green and sustainable cyber-physical security solutions for body area networks," in *BSN '09: Proceedings of the 2009 Sixth International Workshop on Wearable and Implantable Body Sensor Networks*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 240–245.
- [86] T. Mukherjee, A. Banerjee, G. Varsamopoulos, and S. K. S. Gupta, "Model-driven coordinated management of data centers," *Comput. Netw.*, vol. 54, no. 16, pp. 2869–2886, nov 2010. [Online]. Available: <http://dx.doi.org/10.1016/j.comnet.2010.08.011>
- [87] I. Rhee, M. Shin, S. Hong, K. Lee, and S. Chong, "On the levy-walk nature of human mobility," in *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, april 2008, pp. 924 –932.
- [88] A. Natarajan, B. de Silva, K.-K. Yap, and M. Motani, "To hop or not to hop: Network architecture for body sensor networks," in *Sensor, Mesh and Ad Hoc Communications and Networks, 2009. SECON '09. 6th Annual IEEE Communications Society Conference on*, June 2009, pp. 1–9.
- [89] Altera, "Vhdl," <http://www.altera.com/support/examples/vhdl/vhdl.html>.
- [90] OMG, "Unified modeling language," <http://www.uml.org/>.
- [91] C.-L. Fok, A. Petz, D. Stovall, N. Paine, C. Julien, and S. Vishwanath, "Pharos: A testbed for mobile cyber-physical systems," *Univ. of Texas at Austin, Tech. Rep. TR-ARiSE- 2011-001*, 2011.
- [92] D. Acharya, V. Kumar, and H.-J. Han, "Performance evaluation of data intensive mobile healthcare test-bed in a 4g environment," in *Proceedings of the 2nd ACM international workshop on Pervasive Wireless Healthcare*, ser. MobileHealth '12. New York, NY, USA: ACM, 2012, pp. 21–26. [Online]. Available: <http://doi.acm.org/10.1145/2248341.2248353>
- [93] Mathworks, "Matlab and simulink," <http://www.mathworks.com/>.
- [94] SysML, "Systems modeling language (sysml)," <http://www.sysml.org/>.
- [95] "Flovent." [Online]. Available: <http://www.mentor.com/products/mechanical/products/flovent>

- [96] A. Cervin and K.-E. Arzen, *Model-Based Design for Embedded Systems*. CRC Press, 2011, ch. TrueTime: Simulation Tool for Performance Analysis of Real-Time Embedded Systems, pp. 93–119.
- [97] Modelica and the Modelica Association, “Modelica,” <https://modelica.org/>.
- [98] Z. Jiang, M. Pajic, and R. Mangharam, “Model-based closed-loop testing of implantable pacemakers,” in *Proceedings of the 2011 IEEE/ACM Second International Conference on Cyber-Physical Systems*, ser. ICCPS '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 131–140. [Online]. Available: <http://dx.doi.org/10.1109/ICCPS.2011.28>
- [99] K. E. Andersen, “A bayesian approach to bergman’s minimal model,” in *in: C.M. Bishop, B.J. Frey (Eds.), Proceedings of the Ninth International Workshop on Artificial Intelligence*,, 2003.
- [100] J. C. Jensen, D. Chang, and E. A. Lee, “A model-based design methodology for cyber-physical systems,” in *Wireless Communications and Mobile Computing Conference (IWCMC), 2011 7th International*, July 2011, pp. 1666 – 1671.
- [101] W. Yan, Y. Xue, X. Li, J. Weng, T. Busch, and J. Sztipanovits, “Integrated simulation and emulation platform for cyber-physical system security experimentation,” in *Proceedings of the 1st international conference on High Confidence Networked Systems*, ser. HiCoNS '12. New York, NY, USA: ACM, 2012, pp. 81–88. [Online]. Available: <http://doi.acm.org/10.1145/2185505.2185519>
- [102] A. Bhave, B. Krogh, D. Garlan, and B. Schmerl, “Multi-domain modeling of cyber-physical systems using architectural views,” in *Proceedings of the 1st Analytic Virtual Integration of Cyber-Physical Systems Workshop*,, 30 November 2010.
- [103] D. Henriksson and H. Elmqvist, “Cyber-physical systems modeling and simulation with Modelica,” in *Proceedings of the 8th International Modelica Conference, Dresden, Germany*, 2011, pp. 502–509.
- [104] K. Bauer, “A new modelling language for cyber-physical systems,” Ph.D. dissertation, Department of Computer Science, University of Kaiserslautern, Germany, Kaiserslautern, Germany, January 2012.
- [105] D. Arney, R. Jetley, P. Jones, I. Lee, and O. Sokolsky, “Formal methods based development of a pca infusion pump reference model: Generic infusion pump

- (gip) project,” in *HCMDSS-MDPNP '07: Proceedings of the 2007 Joint Workshop on High Confidence Medical Devices, Software, and Systems and Medical Device Plug-and-Play Interoperability*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 23–33.
- [106] W. Reisig, *Petri nets: an introduction*. New York, NY, USA: Springer-Verlag New York, Inc., 1985.
- [107] S. Nabar, A. Banerjee, S. K. S. Gupta, and R. Poovendran, “Evaluation of body sensor network platforms: a design space and benchmarking analysis,” in *Wireless Health 2010*, ser. WH '10. New York, NY, USA: ACM, 2010, pp. 118–127. [Online]. Available: <http://doi.acm.org/10.1145/1921081.1921096>
- [108] H. Veld and M. Ordelman, “Context aware algorithm for epileptic seizure detection,” in *Awareness deliverables*, 2005.
- [109] C. Cortazar, M. Elgueta, and J. D. Rossi, “A nonlocal diffusion equation whose solutions develop a free boundary,” *Annales Henri Poincare*, vol. 6, pp. 269–281, 2005, 10.1007/s00023-005-0206-z. [Online]. Available: <http://dx.doi.org/10.1007/s00023-005-0206-z>
- [110] M. C. Bujorianu et al, “An integrated specification logic for cyber-physical systems,” in *ICECCS 2009*, 2009, pp. 291–300.
- [111] T. L. Jackson and H. M. Byrne, “A mathematical model to study the effects of drug resistance and vasculature on the response of solid tumors to chemotherapy,” *Mathematical Biosciences*, vol. 164, no. 1, pp. 17 – 38, 2000.
- [112] A. Girard and C. Guernic, “Zonotope/hyperplane intersection for hybrid systems reachability analysis,” in *Proceedings of the 11th international workshop on Hybrid Systems: Computation and Control*, ser. HSCC '08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 215–228. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-78929-1_16
- [113] Dr. KYOUNG-DAE KIM, “Phd dissertation middleware and control of cyber-physical systems: Temporal guarantees and hybrid system analysis,” *under Dr. P.R. Kumar*.
- [114] T. Henzinger, “The theory of hybrid automata,” *Logic in Computer Science, Symposium on*, vol. 0, p. 278, 1996.
- [115] M. Althoff, O. Stursberg, and M. Buss, “Computing reachable sets of hybrid systems using a combination of zonotopes and polytopes,” *Nonlinear Analysis: Hybrid Systems*, vol. 4, no. 2, pp. 233–249, 2010.

- [116] R. Alur, T. Dang, and F. Ivancic, "Reachability analysis of hybrid systems via predicate abstraction," in *Hybrid Systems: Computation and Control*, ser. LNCS, vol. 2289. Springer, 2002, pp. 35–48.
- [117] E. Asarin, T. Dang, and A. Girard, "Reachability analysis of nonlinear systems using conservative approximation," in *Hybrid Systems: Computation and Control*, ser. LNCS, vol. 2623. Springer, 2003, pp. 22–35.
- [118] T. Dang, "Approximate reachability computation for polynomial systems," in *Hybrid Systems: Computation and Control*, ser. LNCS, J. P. Hespanha and A. Tiwari, Eds., vol. 3927. Springer, 2006, pp. 138–152.
- [119] A. Girard, "Reachability of uncertain linear systems using zonotopes," in *Hybrid Systems: Computation and Control*, ser. LNCS, vol. 3414, 2005, pp. 291–305.
- [120] Z. Han and B. H. Krogh, "Reachability analysis of hybrid control systems using reduced-order models," in *Proceedings of the American Control Conference*, 2004, pp. 1183–1189.
- [121] A. Kurzhanski and P. Varaiya, "Ellipsoidal techniques for hybrid dynamics: the reachability problem," in *New Directions and Applications in Control Theory*, ser. LNCIS, W. P. Dayawansa, A. Lindquist, and Y. Zhou, Eds., vol. 321. Springer, 2005, pp. 193–205.
- [122] G. E. Fainekos, S. Sankaranarayanan, F. Ivančić, and A. Gupta, "Robustness of model-based simulations," in *IEEE Real-Time Systems Symposium*, 2009, to appear.
- [123] A. Chutinan and B. Krogh, "Verification of polyhedral invariant hybrid automata using polygonal flow pipe approximations," in *Hybrid Systems: Computation and Control*, ser. LNCS, vol. 1569. Springer, 1999, pp. 76–90.
- [124] A. Girard and G. J. Pappas, "Verification using simulation," in *Hybrid Systems: Computation and Control (HSCC)*, ser. LNCS, vol. 3927. Springer, 2006, pp. 272–286.
- [125] B. I. Silva and B. H. Krogh, "Formal verification of hybrid systems using Check-Mate: a case study," in *Proceedings of the American Control Conference*, vol. 3, Jun 2000, pp. 1679–1683.
- [126] N. Grégoire and M. Bouillot, "Hausdorff distance between convex polygons." [Online]. Available: <http://cgm.cs.mcgill.ca/~godfried/teaching/cg-projects/98/normand/main.html>

- [127] H. Alt, J. Blümel, and H. Wagener, "Approximation of convex polygons," in *Automata, Languages and Programming*, ser. Lecture Notes in Computer Science, M. Paterson, Ed. Springer Berlin / Heidelberg, 1990, vol. 443, pp. 703–716, 10.1007/BFb0032068. [Online]. Available: <http://dx.doi.org/10.1007/BFb0032068>
- [128] A. Lamperski and A. Ames, "On the existence of zeno behavior in hybrid systems with non-isolated zeno equilibria," in *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, dec. 2008, pp. 2776 –2781.
- [129] S. Weininger et al, "Factors to consider in a risk analysis for safe surface temperature," in *Product Safety Engineering, 2005 IEEE Symposium on*, Oct. 2005, pp. 83–91.
- [130] S. Medical, "Graseby 3300 technical manual," http://www.frankshospitalworkshop.com/equipment/documents/infusion_pumps/service_manuals/Graseby_3300_Syringe_Pump_-_Service_manual.pdf.
- [131] Stefan and Boltzman, "Stefan-boltzman law." [Online]. Available: <http://hyperphysics.phy-astr.gsu.edu/hbase/thermo/stefan.html>
- [132] E. Lehmann and T. Deutsch, "A physiological model of glucose-insulin interaction in type 1 diabetes mellitus," *Journal of Biomedical Engineering*, vol. 14, no. 3, pp. 235 – 242, 1992, <ce:title>Annual Scientific Meeting</ce:title>. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/014154259290058S>
- [133] F. Erzen, *Studies on Modeling Glucose Insulin Interaction in Human Body and Development of a Simulation Package*. Illinois Institute of Technology, 2000. [Online]. Available: <http://books.google.com/books?id=bKJyNwAACAAJ>
- [134] E. Lee, "Ptolemy project." [Online]. Available: <http://ptolemy.eecs.berkeley.edu/>
- [135] X. Chen, A. Waluyo, I. Pek, and W.-S. Yeoh, "Mobile middleware for wireless body area network," in *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*, 31 2010-sept. 4 2010, pp. 5504 –5507.
- [136] G. Fortino, A. Guerrieri, F. Bellifemine, and R. Giannantonio, "Spine2: developing bsn applications on heterogeneous sensor nodes," in *Industrial Embedded Systems, 2009. SIES '09. IEEE International Symposium on*, july 2009, pp. 128 –131.

- [137] M. Mozumdar, F. Gregoretti, L. Lavagno, L. Vanzago, and S. Olivieri, "A framework for modeling, simulation and automatic code generation of sensor network application," in *Sensor, Mesh and Ad Hoc Communications and Networks, 2008. SECON '08. 5th Annual IEEE Communications Society Conference on*, june 2008, pp. 515–522.
- [138] J. B. Lim, B. Jang, S. Yoon, M. L. Sichitiu, and A. G. Dean, "Raptex: Rapid prototyping tool for embedded communication systems," *ACM Trans. Sen. Netw.*, vol. 7, pp. 7:1–7:40, August 2010.
- [139] E. Cheong, E. A. Lee, and Y. Zhao, "Viptos: a graphical development and simulation environment for tinyos-based wireless sensor networks," in *Proceedings of the 3rd international conference on Embedded networked sensor systems*, ser. SenSys '05. New York, NY, USA: ACM, 2005, pp. 302–302.
- [140] W. P. McCartney and N. Sridhar, "Tosdev: a rapid development environment for tinyos," in *Proceedings of the 4th international conference on Embedded networked sensor systems*, ser. SenSys '06. New York, NY, USA: ACM, 2006, pp. 387–388.
- [141] M.-C. Hsiao, C.-H. Chan, V. Srinivasan, A. Ahuja, G. Erinjippurath, T. P. Zanos, G. Gholmieh, D. Song, J. D. Wills, J. LaCoss, S. Courellis, A. R. Tanguay, J. J. Granacki, V. Z. Marmarelis, and T. W. Berger, "Vlsi implementation of a nonlinear neuronal model: a neural prosthesis to restore hippocampal trisynaptic dynamics." *Conf Proc IEEE Eng Med Biol Soc*, vol. 1, pp. 4396–9, 2006. [Online]. Available: <http://www.biomedsearch.com/nih/VLSI-implementation-nonlinear-neuronal-model/17946244.html>
- [142] Y.-H. Kuo, C.-I. Kao, and J.-J. Chen, "A fuzzy neural network model and its hardware implementation," *Fuzzy Systems, IEEE Transactions on*, vol. 1, no. 3, pp. 171–183, aug 1993.
- [143] B. Bishop, T. Kelliher, and M. Irwin, "Hardware/software co-design for real-time physical modeling," in *Multimedia and Expo, 2000. ICME 2000. 2000 IEEE International Conference on*, vol. 3, 2000, pp. 1363–1366 vol.3.
- [144] J.-G. Juang and W.-K. Liu, "Hardware implementation of a hybrid intelligent controller for a twin rotor mimo," in *Innovative Computing Information and Control, 2008. ICICIC '08. 3rd International Conference on*, june 2008, p. 185.
- [145] E. Motuk, R. Woods, and S. Bilbao, "Fpga-based hardware for physical modelling sound synthesis by finite difference schemes," in *Field-Programmable Technology, 2005. Proceedings. 2005 IEEE International Conference on*, dec. 2005, pp. 103–110.

- [146] S. Nabar, A. Banerjee, S. K. S. Gupta, and R. Poovendran, "Evaluation of body sensor network platforms: a design space and benchmarking analysis," in *Wireless Health 2010*, ser. WH '10. New York, NY, USA: ACM, 2010, pp. 118–127. [Online]. Available: <http://doi.acm.org/10.1145/1921081.1921096>
- [147] A. Taflove and S. C. Hagness, *Computational Electrodynamics: The Finite-Difference Time-Domain Method, Third Edition*, 3rd ed. Artech House Publishers, jun 2005.
- [148] P. Smith and P. Hasler, "A programmable diffuser circuit based on floating-gate devices," in *Circuits and Systems, 2002. MWSCAS-2002. The 2002 45th Midwest Symposium on*, vol. 1, aug. 2002, pp. 1 – 291–4 vol.1.
- [149] D. Paul, L. Nathan, Y. Bazhang, M. Yvonne, and F. Moussy, "Study of the effects of tissue reactions on the function of implanted glucose sensors," *Journal of Biomedical Materials Research Part A*, pp. 699–706, 2007.
- [150] J. Sproston and A. Troina, "Simulation and bisimulation for probabilistic timed automata," in *FORMATS*, 2010, pp. 213–227.
- [151] L. Bu, D. Xie, X. Chen, L. Wang, and X. Li, "Demo abstract: Bachol - modeling and verification of cyber-physical systems online," in *Cyber-Physical Systems (ICCPs), 2012 IEEE/ACM Third International Conference on*, april 2012, p. 222.

BIOGRAPHICAL SKETCH

Ayan Banerjee graduated from the School of Computing, Informatics, and Decision Systems Engineering at Arizona State University. He received his Bachelor's degree in Electronics and Telecommunication Engineering from Jadavpur University, India, in 2007, and began his Ph.D. in Computer Science under Dr. Sandeep Gupta later that year. His research interests include formal modeling and analysis, sensor networks, embedded systems, pervasive health monitoring, and energy efficient cloud computing.

His research focuses on the safety of Cyber-Physical Systems (CPSes). CPSes are control systems, which require a closed loop interaction with their environment for their operation. Examples include sensor networks and data centers.

Specially two important characteristics of CPSes render the current techniques inapplicable:

- a Spatio-temporal multi-dimensional variation of the effect of interactions e.g. the drug concentration due to infusion pump operation varies over time as well as it is different at different points in the body.
- b Non linear dynamics due to feedback from different components of the system.

He has defined novel hybrid systems that can capture the spatio-temporal effects of CPS interactions and developing techniques to perform reachability analysis on them for proving safety properties. In this regard, he has proposed a SpatioTemporal Hybrid Automata in his PhD dissertation and has developed a time bounded reachability analysis technique for it. The modeling has been applied to the medical domain and used several examples such as the infusion pump and the artificial pancreas to validate my technique.

Below is a list of his publications:

Conferences:

- 1 Sunit Verma, Joseph Milazzo, Yu Xie, Priyanka Bagade, Ayan Banerjee, and Sandeep K.S. Gupta, "Model-based Wireless Health System Design Tool", In Proceedings of 3rd Conference in Wireless Health '12, San Diego, CA
- 2 A. Banerjee, S. Verma, P. Bagade, and S.K.S. Gupta ,Health-Dev: Model Based Development of Pervasive Health Monitoring Systems, The 9th International Conference on Wearable and Implanted Body Sensor Networks 2012, London UK, May 9 - 12
- 3 A. Banerjee and Sandeep K.S. Gupta ,Your Mobility can be Injurious to Your Health: Analyzing Pervasive Health Monitoring Systems under Dynamic Context Changes , IEEE International Conference on Pervasive Computing and Communication (PerCom) 2012

- 4 Ayan Banerjee, Sandeep K.S. Gupta, Georgios Fainekos, and Georgios Varsamopoulos ,Towards Modeling and Analysis of Cyber-Physical Medical Systems , Isabel 2011, Barcelona, Spain
- 5 A. Banerjee, S. Nabar, S.K.S. Gupta, and R. Poovendran ,Energy-efficient Long Term Physiological Monitoring , Wireless Health Demo 2011, San Diego, California
- 6 S. Nabar, A. Banerjee, S.K.S. Gupta, and R. Poovendran ,Resource-efficient and Reliable Long Term Wireless Monitoring of the Photoplethysmographic Signal, Wireless Health 2011 (Accepted for publication), San Diego, California
- 7 Sandeep K.S. Gupta, Rose Robin Gilbert, Ayan Banerjee, Zahra Abbasi, Tridib Mukherjee, Georgios Varsamopoulos, GDCSim - An Integrated Tool Chain for Analyzing Green Data Center Physical Design and Resource Management Techniques , International Green Computing Conference, 2011, Orlando.
- 8 S. Nabar, A. Banerjee, S.K.S. Gupta, and R. Poovendran GeM-REM: Generative Model-driven Resource-efficient ECG Monitoring in Body Sensor Networks. To appear in IEEE BSN 2011
- 9 S. Nabar, A. Banerjee, S.K.S. Gupta, and R. Poovendran Evaluation of Body Sensor Network Platforms: A Design Space and Benchmarking Analysis , Wireless Health 2010, San Diego, CA, 2010, Accepted for publication
- 10 Ayan Banerjee, Tridib Mukherjee, Georgios Varsamopoulos, and Sandeep K. S. Gupta. Cooling-Aware and Thermal-Aware Workload Placement for Green HPC Data Centers. International Conference on Green Computing Conference (IGCC), Chicago, IL, August 2010.
- 11 Georgios Varsamopoulos, Ayan Banerjee, Sandeep Gupta. Energy Efficiency of Thermal-Aware Job Scheduling Algorithms under Various Cooling Models. International Conference on Contemporary Computing (IC3), Noida , India, August 2009.
- 12 K. Venkatasubramanian, A. Banerjee, S. K. S. Gupta, Green and Sustainable Cyber Physical Security Solutions for Body Area Networks In Proc of 6th Workshop on Body Sensor Networks (BSN'09), Berkeley, CA, June 2009.
- 13 A. Banerjee, K. Venkatasubramanian, S. K. S. Gupta, Challenges of Implementing Cyber-Physical Security Solutions in Body Area Networks In Proc of International Conference on Body Area Networks (BodyNets'09), Los Angeles, CA, April 2009.
- 14 K. Venkatasubramanian, A. Banerjee, S. K. S. Gupta, Plethysmogram-based Secure Inter-Sensor Communication in Body Area Networks In Proc of IEEE Military Communications Conference, (MILCOM'08), San Diego, CA, November 2008.

- 15 K. Venkatasubramanian, A. Banerjee, S. K. S Gupta, EKG-based Key Agreement in Body Sensor Networks, In Proc. of 2nd Mission Critical Networks Workshop, IEEE Infocom Workshops, Phoenix, AZ, April 2008.
- 16 G. Varsamopoulos, S. K. S. Gupta, A. Banerjee, and K. K. Venkatasubramanian, Integrating Cyber-Physical Research and Thinking in College Education, In Proc. International Symposium on Integrating Research, Education, and Problem Solving (IREPS 2011)

Journals:

- 1 A. Banerjee, K. Venkatasubramanian, T. Mukherjee, and S.K.S. Gupta Ensuring Safety, Security and Sustainability of Mission-Critical Cyber Physical Systems, IEEE Proceedings special issue on cyber-physical systems
- 2 Ayan Banerjee, Tridib Mukherjee, Georgios Varsamopoulos, and Sandeep K. S. Gupta Integrating Cooling Awareness with Thermal Aware Workload Placement for HPC Data Centers, Elsevier Comnets Special Issue in Sustainable Computing (SUSCOM) 2011.
- 3 Sandeep K. S. Gupta, Tridib Mukherjee, Georgios Varsamopoulos, and Ayan Banerjee Research Directions in Energy-Sustainable Cyber-Physical Systems, Elsevier Comnets Special Issue in Sustainable Computing (SUSCOM) 2011 (Invited Paper).
- 4 A. Banerjee, S. Kandula, T. Mukherjee, and S.K.S. Gupta BAND-AiDe: A Tool for Cyber-Physical Oriented Analysis and Design of Body Area Networks and Devices, ACM Transactions in Embedded Computing Systems, Special Issue on Wireless Health 2010, Accepted for publication
- 5 Tridib Mukherjee, Ayan Banerjee, Georgios Varsamopoulos, and S. K. S. Gupta, Model-driven Co-ordinated Management of Data Centers. (Elsevier) Computer Networks, Special Issue on Managing Emerging Computing Environments(ComNet), accepted (2010).
- 6 T. Mukherjee, A. Banerjee, G. Varasamopoulos, and S. K. S. Gupta, Spatio-Temporal Thermal-Aware Job Scheduling to Minimize Energy Consumption in Virtualized Heterogeneous Data Centers. Elsevier Computer Networks (ComNet), Vol. 53, Issue 17, Pages 2888-2904, December, 2009.
- 7 K. Venkatasubramanian, A. Banerjee, S.K.S Gupta, PSKA: Usable and Secure Key Agreement Scheme for Body Area Networks in Transaction in IEEE Transactions on Information Technology in Biomedicine (TITB).

Book Chapters:

- 1 Zahra Abbasi, Michael Joans, Ayan Banerjee, Georgios Varsamopoulos and Sandeep K. S. Gupta, Evolutionary Green Computing Solutions for Distributed Cyber Physical Systems, Springer book on Evolutionary based Solutions for Green Computing, 2012