TOWARD DEPENDABLE CYBER-PHYSICAL SYSTEMS: A STUDY ON DESIGN PATTERN AND EVALUATION METHODOLOGY

TAN FENG

Ph.D

The Hong Kong Polytechnic University

2016

The Hong Kong Polytechnic University

Department of Computing

Toward Dependable Cyber-Physical Systems: A Study on Design Pattern and Evaluation Methodology

TAN Feng

A thesis submitted in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

December 2015

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

..... (Signed)

TAN Feng (Name of student)

Abstract

People live in and interact with the physical world. As technology advances, embedded devices with sensing, computation, actuation, and networking capabilities are drastically changing our way to interact with the physical world. This introduces a new type of systems, namely, Cyber-Physical Systems (CPS). CPS tightly integrates discrete computing and continuous-time physical-world entities. It reshapes the way that humans interact with the physical world, and thus is believed to have deep social and economical impacts.

As many CPS applications are mission/life critical, dependability is a top concern. To build dependable CPS, various fault prevention, fault tolerance, and fault removal measures are needed in the new context of CPS.

In this thesis, we address several challenging issues on building dependable CPS.

First, we propose a fault prevention solution to guarantee Proper-Temporal-Embedding (PTE) safety rules in wireless CPS. The proposed solution exploits the leasing design philosophy to tolerate arbitrary wireless communication failures, and support real-time temporal constraints. The proposed solution is validated by two case studies: one on medical CPS and the other on control CPS. The performance of our solution is also compared to a polling based solution. Simulation results show that our proposed so-

lution achieves better user experience when wireless channel is benign or moderately adverse, and better resource usage in all scenarios.

Second, we propose a cross-domain noise profiling framework for control CPS. The proposed framework plays a key role in control CPS dependability evaluation, an essential tool to CPS fault tolerance and fault removal. Key elements of this framework include a hybrid automata reachability based dependability metric, and a Lyapunov stability theory based benchmark shrinking strategy. Case studies are carried out to validate the proposed framework and showcase its usage.

Publications

Journal Papers

1. **Feng Tan**, Liansheng Liu, Stefan Winter, Qixin Wang, Neeraj Suri, Lei Bu, Yu Peng, Xue Liu, Xiyuan Peng, "Profiling Cross-Domain Noise for Gray Box Two-Level Control CPS", submitted to ACM Transactions on Cyber-Physical Systems, under review.

2. **Feng Tan**, Yufei Wang, Qixin Wang, Lei Bu, Neeraj Suri, "A Lease based Hybrid Design Pattern for Proper-Temporal-Embedding of Wireless CPS Interlocking," in IEEE Transactions on Parallel and Distributed Systems, 26(10):2630-2642, Oct, 2015.

3. Tao Li, **Feng Tan**, Qixin Wang, Lei Bu, Jian-Nong Cao, Xue Liu, "From Offline toward Real Time: A Hybrid Systems Model Checking and CPS Codesign Approach for Medical Device Plug-and-Play Collaborations," in IEEE Transactions on Parallel and Distributed Systems, 25(3), March, 2014. pp. 642-652.

Conference/Workshop Papers

1. **Feng Tan**, Liansheng Liu, Stefan Winter, Qixin Wang, Neeraj Suri, Lei Bu, Yu Peng, Xue Liu, Xiyuan Peng, "WiP Abstract: A Framework on Profiling Cross-Domain Noise Propagation in Control CPS," in ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS'14) Work-in-Progress Session, Berlin, Germany, April, 2014.

2. **Feng Tan**, Yufei Wang, Qixin Wang, Lei Bu, Rong Zheng, and Neeraj Suri, "Guaranteeing Proper-Temporal-Embedding Safety Rules in Wireless CPS: A Hybrid Formal Modeling Approach," in Proc. of the 43rd IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), 2013.

3. Tao Li, **Feng Tan**, Qixin Wang, Lei Bu, Jian-nong Cao, Xue Liu, "From Offline toward Real-Time: A Hybrid Systems Model Checking and CPS Co-Design Approach for Medical Device Plug-and-Play (MDPnP)," in Proc. of ICCPS'12, April, 2012. pp.13-22.

4. Tao Li, Qixin Wang, **Feng Tan**, Lei Bu, Jian-nong Cao, Xue Liu, Yufei Wang, and Rong Zheng, "From Offline Long-Run to Online Short-Run: Exploring a New Approach of Hybrid Systems Model Checking for MDPnP," in Joint Workshop on High Confidence Medical Devices, Software, and Systems and Medical Device Plug-and-Play Interoperability (HCMDSS/MDPnP'11), Chicago, IL, April 11, 2011.

Acknowledgements

First and foremost, I would like to give my deepest gratitude to my chief supervisor, Dr. Qixin Wang, who leads me into the exciting new research field: Cyber-Physical Systems. As a supervisor, he has not only provided continuous support and guidance for my Ph.D study, but also sets an ideal role model of good supervisor. His vision, knowledge, rigorous thinking, and diligence have generated great influence on my way of studying and doing research. Undoubtedly, what I have learnt from Dr. Qixin Wang will lay the foundations for my future career.

Also, I would like to thank Dr. Lei Bu, Dr. Rong Zheng and Prof. Neeraj Suri. My research work in this thesis has benefited a lot from their invaluable comments and suggestions.

Besides, I would like to thank my friends and group members, such as Mr. Yufei Wang, Mr. Jeppe Rishede Thomsen, Mr. Shuhang Gu, Mr. Yanxing Hu, Mr. Jiwei Li, Mr. Shang Gao, Mr. Enyan Huang, Mr. Zhe Peng, Mr. Renhai Chen, Mr. Pengwei Hu, Dr. Zhian He, Mr. Hu Xiao, Mr. Zhijian He, Mr.Yanming Chen etc. I am fortunate to have worked with you and learnt a lot from you. I really appreciate your time and you form a warm family for me in Hong Kong.

Last but not least, I would like to express my heart-felt gratitude to my parents,

my parents-in-law, and my wife, Xue Mei. It is your love, understanding, support and patience that encouraged me to overcome the difficulties, and gave me strength to walk through the journey of Ph.D. study.

Contents

List of Figures

Chapter 1

Introduction

In our daily lives, we live in and interact with the physical world. As technology advances, the physical environment is now converged with cyber systems of sensing, computation, actuation, and communication capabilities. Such convergence will transform our way of involvement with the physical world. For example, buildings and transportations are well-known for their high energy costs; these costs can now be restrained in smart buildings/transportation, where physical buildings/transportation are converged with smart cyber systems [S$^+$08]. However, such convergence requires examination of interplay between cyber components (e.g. sensing, computation, actuation, and communications devices) and physical ones (e.g. cooling and heating equipments, vehicles etc.).

Systems with converged cyber and physical components are called *Cyber-Physical Systems* (CPS). As defined in [RLSS10], "*Cyber-Physical Systems are physical and engineered systems whose operations are monitored, coordinated, controlled, and integrated by a computing and communication core.*" By converging the physical world

with the cyber world, CPS shall revolutionize not only computer science and technology, but also our society, just as Internet did by converging individual computers.

CPS has broad applications in medicine, control, power grid, manufacturing, transportation etc. For example, surgical robots, fly-by-wire airplanes, electric vehicles are all typical examples of CPS. Many of these CPSs are safety/mission critical, hence dependability is of top concern.

To guarantee dependability, there are three main approaches:

- **Fault Prevention**: making the system fault free by design.
- **Fault Tolerance**: designing the system so that even if faults happen in runtime, the system can tolerate them and continue to function correctly.
- **Fault Removal**: testing/debugging the system during development time, and/or fixing the system during runtime.

However, in the CPS context, fault prevention, tolerance, and removal measures differ from those for pure software systems, or pure physical systems. Particularly, we need to address the following challenges:

1. In CPS, the coupling between cyber and physical components invalidates many assumptions for pure cyber or pure physical systems, forcing us to rethink/redesign corresponding fault prevention/tolerance/removal measures. Specifically,

1.1 Physical components invalidates many assumptions and measures for pure cyber systems. For example, in pure cyber systems, check-pointing and rollback are feasible and widely used. However, many physical components

cannot be check-pointed or rolled back: we cannot kill a patient and then "roll back." Another example, when coordinating pure cyber components, only logical time matters. But when coordinating physical components, real-time (i.e. the exact durations or deadlines) also matters: if we pause a patient's breath for 30 seconds, the patient will not die; but if we pause for 30 hours, the patient dies for sure.

1.2 Cyber components invalidates many assumptions and measures for pure physical systems. Nowadays cyber components can be extremely complex and nonlinear, involving millions of lines of source code. Due to the combinatorial explosion of complexity, they are hard to model and analyze. This will invalidate many conventional modeling/analysis measures for pure physical systems, such as those based on closed-form formulae and/or automata.

2. As CPS are inter-disciplinary by definition, good solutions for CPS naturally demand careful consideration and ingenious exploitation of cross-domain knowledge and constraints.

To fully address CPS fault prevention, tolerance, and removal challenges is too big a task for a single thesis to cover. As an initial attempt, we made the following contributions on several key issues.

In Chapter 2, we propose a leasing based design pattern for wireless CPS. Following this design pattern, Proper-Temporal-Embedding (PTE) safety rules (a safety rule involves real-time temporal logic and has broad applications) are guaranteed by design, regardless of arbitrary wireless communications failures in runtime. In this sense, the proposed solution can be regarded as a fault prevention measure. The content of

Chapter 2 (and corresponding appendices in Chapter 5) is published in the following IEEE/IFIP papers:

- Copyright ©2013 IEEE. Reprinted, with permission, from Feng Tan, Yufei Wang, Qixin Wang, Lei Bu, Rong Zheng, and Neeraj Suri, "Guaranteeing Proper-Temporal-Embedding Safety Rules in Wireless CPS: A Hybrid Formal Modeling Approach", in Proc. of the 43rd IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), 24-27th June, 2013.

- Copyright ©2015 IEEE. Reprinted, with permission, from Feng Tan, Yufei Wang, Qixin Wang, Lei Bu, and Neeraj Suri, "A Lease based Hybrid Design Pattern for Proper-Temporal-Embedding of Wireless CPS Interlocking", in IEEE Transactions on Parallel and Distributed Systems (TPDS), vol.26, no.10, pp.2630-2642, Oct 2015.

In Chapter 3, we introduce a cross-domain noise profiling framework for control CPS. This framework exploits hybrid automata reachability theories to measure the dependability of a control CPS. To deal with the high complexity of cyber components, the framework models the cyber subsystem as gray box, and proposes a benchmark shrinking strategy based on Lyapunov stability control theories. The proposed dependability metric and benchmarking strategy are key tools for CPS fault tolerance and fault removal: the dependability metric quantifies a CPS's fault tolerance level; while the benchmarking strategy facilitates the testing and debugging of a CPS. The content of Chapter 3 (and corresponding appendices in Chapter 5) is published in the following IEEE/ACM work-in-progress paper:

- Copyright ©2014 IEEE. Reprinted, with permission, from Feng Tan, Liansheng

Liu, Stefan Winter, Qixin Wang, Neeraj Suri, Lei Bu, Yu Peng, Xue Liu, and Xiyuan Peng, "WiP abstract: A Framework on Profiling Cross-Domain Noise Propagation in Control CPS", in Proc. of ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS) Work-in-Progress Session, Berlin, Germany, April, 2014.

Also, the content of Chapter 3 (and corresponding appendices in Chapter 5) is currently under review for journal publication.

Finally, in Chapter 4, we conclude the thesis and discuss possible future work.

Chapter 2

A Lease Based Hybrid Design Pattern for Wireless CPS Interlocking

2.1 Introduction

Cyber-Physical Systems (CPS) involve the integration of computation and physical process. Many CPS applications are safety-critical, which demand the safety property guarantee. In this chapter, we focus on the safety property in dependability attribute. Specifically, one important set of such safety rules: Proper-Temporal-Embedding (PTE).

Specifically, consider a distributed CPS system where each entity has an abstract "safe" state and an abstract "risky" state. During idle time, all entities dwell in their safe states. However, to accomplish a collective task, a distributed procedure must be

carried out: relevant entities must enter respective risky states in a fixed order and with certain required temporal spacing; and then (after the intended task is done) exit to the respective safe states in exactly the reverse order, and with certain required temporal spacing. Furthermore, each entity's continuous dwelling time (i.e. the duration that it continuously stays in the state) in its "risky" state must be upper bounded by a constant. The safety rules encompassing these discrete ordering and continuous-time temporal conditions define a temporal interlocking pattern, and is termed as *Proper-Temporal-Embedding (PTE) safety rules*.

The scenario can be further illustrated by Fig. 2.1, a classical medical CPS of laser tracheotomy [L$^+$12]. The oxygen ventilator has the "safe" ventilating state, and the "risky" pause state; the laser-scalpel has the "safe" shutoff state, and the "risky" emission state. In order to emit the laser, the oxygen ventilator must first enter the pause state, and only then can the laser-scalpel enter the emission state. Otherwise, the laser emission can trigger fire on the oxygen ventilated trachea of the patient. Conversely, the laser-scalpel must first exit the emission state, and then the ventilator can exit the pause state. Thirdly, certain minimal temporal spacing must be maintained during enter/exit of "risky" states, as shown by $t1$ and $t2$ in Fig. 2.1 (e.g., $t1$ means that only after the oxygen ventilator has paused for $t1$ can laser start emission, otherwise the patient's trachea may still have high enough oxygen concentration to catch fire; note this "pause $t1$ before laser emission" approach is chosen in real practice because hard real-time and error-free trachea oxygen level sensing is impractical). Fourthly, the continuous dwelling time, as shown by $t3$ and $t4$ in Fig. 2.1, must each be upper bounded by a constant (e.g., the ventilator pause duration $t3$ must be upper bounded, for otherwise the patient may suffocate to death). Modeling these sequenced CPS operations constitute *design patterns*.

However, CPS environment often entails wireless-connected sensing, control and

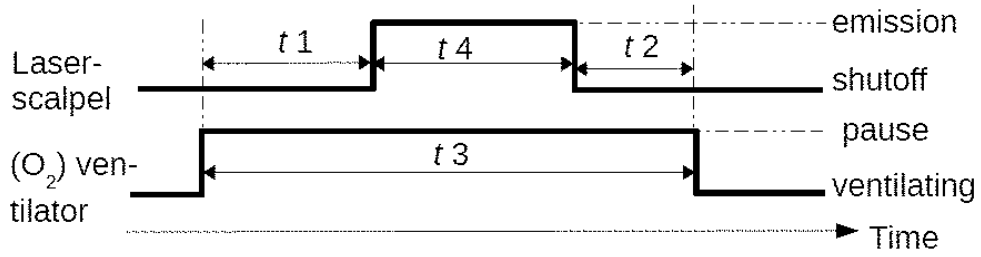


Figure 2.1. Proper-Temporal-Embedding Example

computing entities, guaranteeing PTE safety rules necessitates consideration of unreliable wireless communication. To this end, we utilize and adapt the established design pattern of "*leasing*" [G$^+$89, T$^+$97, A$^+$05, K$^+$08, B$^+$07, A$^+$10], to ensure auto-reset of distributed entities under communication faults. The basic idea is that each entity's dwelling duration in risky state is "*lease*" based (aka *leasing* based). A lease is a timer, which takes effect when the entity enters the risky state. When the lease expires, the entity exits the risky state, no matter if it receives exit command from another entity or not.

Lease based design pattern has been widely adopted in distributed computer systems, particularly distributed storage and database systems. We find it can also be applied to cyber-physical systems, where discrete and continuous states intermingle. Compared to the many existing leasing based designs in computer systems, the wireless CPS leasing based design faces the following paradigm shifts.

First, leasing based designs in computer (i.e. cyber) systems are often integrated with distributed check-point and roll-back [G$^+$89, T$^+$97, A$^+$05, K$^+$08]. However, in CPS, computers often have little control over the physical world states: these states cannot be check-pointed or rolled-back. For example, we cannot revive a killed patient; nor can we recover a piece of burnt wood.

Second, in addition to logic-time, continuous-time durations (e.g. the maximal dwelling duration and safeguard interval in PTE safety rules) matter.

Considering the above paradigm shifts, our leasing based design pattern shall not use check-point or roll-back. Instead, its safety is guaranteed by properly configuring continuous-time temporal parameters.

These heuristics are systematically developed into a lease based design pattern for wireless CPS PTE safety guarantee in this work. Specifically, we make the following contributions:

1. We formalize a temporal interlocking/mutual-exclusion pattern (i.e. PTE safety rules) for CPS physical component interactions.

2. We propose a rigorous leasing based design pattern for wireless CPS; and identify a set of closed-form constraints on software (i.e. cyber) configuration parameters. We prove that as long as these constraints are satisfied, the design pattern guarantees PTE safety rules under arbitrary packet losses over wireless.

3. We propose utilizing *hybrid modeling* [A$^+$93, H$^+$95, A$^+$96] to describe and analyze CPS design patterns. Hybrid modeling is a formal technique to describe/analyze both the discrete and continuous dynamics of a system, hence it is suitable for CPS. Recently, hybrid modeling has gained popularity for CPS, though to our best knowledge, it is mostly used for verification and we are the first to apply it to CPS design pattern research.

4. We propose a formal methodology to refine the design pattern hybrid automata into specific wireless CPS designs. This methodology can effectively isolate physical world parameters (which are much harder to control, compared to the

software/cyber parameters) from affecting the PTE safety of the resultant specific wireless CPS designs.

5. We conduct two case studies, respectively on wireless medical CPS and wireless control CPS, to validate our proposed approach. We also compare our approach with a polling based approach proposed by Kim et al [K+10]. The comparison results show that both approaches can guarantee PTE safety against arbitrary communication failures. In terms of resource occupation efficiency and user experience, the polling based approach performs better under severely adverse wireless medium conditions; while ours performs better under benign or moderately adverse wireless medium conditions.

The rest of this chapter is organized as follows. Section 2.2 introduces the CPS hybrid modeling background; Section 2.3 describes the requirements to guarantee PTE safety rules; Section 2.4 formally defines the leasing based design pattern, proves its guarantee of PTE safety rules, and describes how to elaborate the design pattern into specific designs. Section 2.5 and 2.6 respectively evaluate our proposed approach with emulation/experiment based case studies and simulation based comparisons. Section 2.7 discusses related work. Section 2.8 concludes this chapter.

2.2 Background, Terms and Models

2.2.1 The Hybrid Modeling Terminology

Hybrid modeling is based on hybrid automaton [A+93, H+95, A+96, GGH+03, LL09], a tool that suits CPS modeling extremely well because it can formally de-

scribe/analyze both discrete (cyber) and continuous (physical) dynamics. For example, Fig. 2.2 illustrates a hybrid automaton A'_{vent} that describes the discrete/continuous behaviors of a stand-alone ventilator (see Appendix 5.1 of Chapter 5 for a more detailed description on the ventilator's working mechanism). $H_{\text{vent}}(t)$ is the height of the ventilator piston at time t. The hybrid automaton execution initially dwells in the location of "PumpOut": the piston continuously moves downward at velocity $\dot{H}_{\text{vent}}(t) = -0.1$(m/s). When the piston hits bottom ($H_{\text{vent}} = 0$), a discrete event happens: the execution moves to location "PumpIn". Once in location "PumpIn", the piston continuously moves upward at velocity $\dot{H}_{\text{vent}}(t) = +0.1$(m/s). When the piston hits ceiling ($H_{\text{vent}} = 0.3$(m)), a discrete event happens: the execution moves to location "PumpOut" again, so on and so forth.

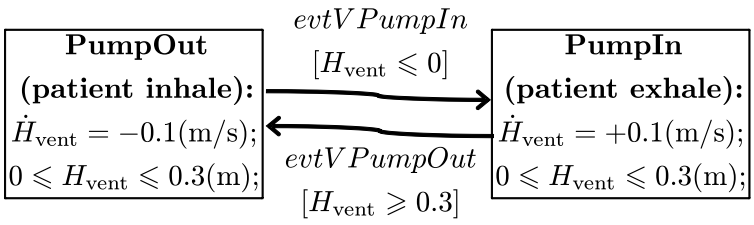


Figure 2.2. Hybrid Automaton A'_{vent} of a Stand-Alone Ventilator. $H_{\text{vent}}(t)$ is the data state variable denoting the ventilator's piston height at time t. "PumpOut" is the only initial location.

In the rest of this chapter, we reuse the notations proposed by Alur et al. [A+96] to formally describe hybrid automata. For reader's convenience, the notation list is also presented in Appendix 5.2 of Chapter 5

2.2.2 System and Fault Model

A *hybrid system* $\mathcal{H}$ is a collection of *hybrid automata* (each is called a *member hybrid automaton* of $\mathcal{H}$), which execute concurrently and coordinate with each other via event communications (i.e., the sending/receiving of synchronization labels). For simplicity, in this work, we assume no shared data state variables nor shared locations between different hybrid automata of a hybrid system. That is, data state variable names or location names are local to their respective hybrid automata[1].

A distributed sink-based wireless CPS consists of the following *entities*: a *base station* ξ_0 and N (*in this work, we require* $N \geq 2$) *remote entities* ξ_1, ξ_2, ..., ξ_N. A wireless communication link from the base station to a remote entity is called a *downlink*; and a wireless communication link from a remote entity to the base station is called an *uplink*. We assume that there is *no* direct wireless communication links between any two remote entities (such practice is desirable for wireless applications with high dependability requirements [W$^+$11, W$^+$07]).

We assume that each packet's checksum is strong enough to detect any bit error(s); a packet with bit error(s) is discarded at the receiver. Our fault model assumes that packets sent via wireless can be arbitrarily lost (not received at all, or discarded at the receiver due to checksum errors). As per PTE safety requirements, the uplink communication delays are specified and handled by the base station. For the downlink, the remote entities locally specify delays as acceptable or as lost-messages.

[1] To make an analogy, each hybrid automaton is like a class in Object-Oriented programming. Data state variables and locations are like class members, hence are "local" ("encapsulated") to their respective hybrid automata (classes). Interactions between hybrid automata are carried out via message (aka event) passing.

2.3 Specification of PTE Safety Rules

For the wireless CPS system and communications fault model described in Section 2.2.2, various safety requirements can be proposed. Addressing all of them is beyond the scope of this work. Instead, this work considers a representative subset of such safety requirements, i.e. the requirement to guarantee PTE safety rules. We start by defining these safety rules.

Let hybrid system $\mathcal{H} = \{A_i | \ (i = 0, 1, \ldots, N)\}$ describe a wireless CPS. The hybrid automaton A_i describes wireless CPS member entity ξ_i. The synchronization labels/functions describe the communication relationships between these hybrid automata.

We assume that for each hybrid automaton $A_i = (\vec{x}_i(t), V_i, \mathsf{inv}_i, F_i, E_i, g_i, r_i, L_i, \mathsf{syn}_i, \Phi_{0,i})$ (where $i = 1 \sim N$), V_i is partitioned into two subsets: V_i^{safe} and V_i^{risky}. We call a location v a "*safe-location*" iff $v \in V_i^{\text{safe}}$; and a "*risky-location*" iff $v \in V_i^{\text{risky}}$ (note we do not differentiate the safe/risky locations for ξ_0).

There are two types of PTE safety rules, namely:

PTE SAFETY RULE 2.1 (BOUNDED DWELLING) *Each entity ξ_i's ($i = 1 \sim N$) continuous dwelling time (i.e. continuous-stay time-span) in risky-locations is upper bounded by a constant.*

To describe the second PTE safety rule, however, we must first introduce the following definition.

DEFINITION 2.1 (PROPER-TEMPORAL-EMBEDDING PARTIAL ORDER) *We say that entity ξ_i and ξ_j has a* proper-temporal-embedding partial order $\xi_i \prec \xi_j$ *iff their respective hybrid automata A_i and A_j always satisfy the following properties:*

p1. If ξ_i dwells in safe-locations at time t (i.e. A_i's location counter $\ell_i(t) \in V_i^{\text{safe}}$), then throughout interval $[t, t + T^{\mathsf{min}}_{\text{risky}:i\to j}]$, ξ_j dwells in safe-locations, where positive constant $T^{\mathsf{min}}_{\text{risky}:i\to j}$ is the ξ_i to *ξ_j* enter-risky safeguard interval.

p2. Whenever ξ_j dwells in risky-locations, ξ_i dwells in risky-locations.

p3. If ξ_j dwells in risky-locations at time t, then throughout interval $[t, t + T^{\mathsf{min}}_{\text{safe}:j\to i}]$, ξ_i dwells in risky-locations, where positive constant $T^{\mathsf{min}}_{\text{safe}:j\to i}$ is the ξ_j to *ξ_i* exit-risky safeguard interval.

Intuitively, Property p2 implies that whenever entity ξ_j is in risky-locations, then entity ξ_i is already in risky-locations. Property p1 and p3, in addition, specify the safeguard interval requirements that ξ_i and ξ_j enter/exit respective risky-locations. Specifically, Property p1 implies that before ξ_j enters its risky-locations, ξ_i should have already been in risky-locations for at least $T^{\mathsf{min}}_{\text{risky}:i\to j}$. Property p3 implies that after ξ_j exits its risky-locations (i.e. returns to safe-locations), ξ_i must stay in risky-locations for at least $T^{\mathsf{min}}_{\text{safe}:j\to i}$.

The above intuition is illustrated by Fig. 2.1, where in laser tracheotomy, ventilator $\prec$ laser-scalpel, if we consider "pause" and "emission" are risky-locations and "ventilating" and "shutoff" are safe-locations.

With this notion of PTE partial ordering, the second PTE safety rule is defined as:

PTE SAFETY RULE 2.2 (PROPER-TEMPORAL-EMBEDDING) *The proper-temporal-embedding partial ordering between entities ξ_1, ξ_2, . . ., ξ_N forms a full ordering.*

In the following, for narrative simplicity and without loss of generality, we assume that PTE Safety Rule 2.2 implies a full ordering of

$$\xi_1 < \xi_2 < \ldots < \xi_N. \tag{2.1}$$

We call a safety rule set belongs to the category of *PTE safety rules* iff the rule set consists of and only of PTE Safety Rule 2.1 and 2.2. As mentioned before, in this work, we shall only focus on wireless CPS whose safety rules belong to the category of PTE safety rules. For simplicity, we call such wireless CPS "*PTE wireless CPS*".

2.4 Design Pattern based Solutions

To guarantee PTE safety rules described in the previous section, we propose a leasing based design pattern approach.

2.4.1 Leasing based Design Pattern

For a PTE wireless CPS, we assume that safety is guaranteed if all its member entities stay in their safe-locations. The challenge arises when a remote entity needs to

enter its risky-locations. When a remote entity ξ_k ($k \in \{1, 2, \ldots, N\}$) of a PTE wireless CPS requests to enter its risky-locations, PTE Safety Rule 2.2 and Ineq. (2.1) imply that entity $\xi_0, \xi_1, \ldots, \xi_k$ must coordinate. This may be achieved through wireless communications (uplink/downlink) via the base station ξ_0. However, wireless communications are by nature unreliable. Messages may be lost, and the states of participating entities may become inconsistent, violating the PTE safety rules.

To deal with the unreliable wireless communications, we propose a "*lease*" based design pattern, and (in the subsequent subsections) show that as long as the PTE wireless CPS design complies with the proposed design pattern, the PTE safety rules are guaranteed.

Specifically, there are three roles for PTE wireless CPS entities: *Supervisor*, *Initializer*, and *Participant*. The base station ξ_0 serves the role of "Supervisor". Initially, all entities stay in their respective safe-locations. We only allow one remote entity to actively request switching to its risky-locations. Such a remote entity is called an "Initializer". *For the time being, let us assume there is only one Initializer; and without loss of generality, assume the Initializer is remote entity* ξ_N.

According to PTE Safety Rule 2.2 and Ineq. (2.1), when ξ_N requests to enter risky-locations, remote entity $\xi_1, \xi_2, \ldots, \xi_{N-1}$ must enter respective risky-locations before ξ_N. Remote entities $\xi_1, \xi_2, \ldots, \xi_{N-1}$ hence play the role of "Participants".

We require that every entity ξ_i's ($i \in \{0, 1, 2, \ldots, N\}$) dwelling in risky-locations is based on a lease, i.e. a contract between the Supervisor and ξ_i. A lease specifies the expiration time of dwelling in the risky-locations, and takes effect upon the entrance to risky-locations. If by the lease expiration, the Supervisor has not yet aborted/cancelled the lease, ξ_i will exit to safe-location automatically.

The above thinking guides us to propose the design of Supervisor, Initializer, and Participant as shown from Table 2.1 to Table 2.3[2]. We respectively denote the Supervisor, Initializer, and (the ith) Participant's defining hybrid automata (see Table 2.1, Table 2.2, and Table 2.3) as A_{supvsr}, A_{initzr}, and $A_{\text{ptcpnt},i}$. These hybrid automata's diagrams (and the respective detailed diagrams in Appendix 5.3 of Chapter 5) are elaborated in the following:

Supervisor:

1. A_{supvsr}'s location set V_{supvsr} include the following locations: "Fall-Back", "Lease ξ_i" (where $i = 1 \sim N$), "Cancel Lease ξ_i" (where $i = 1 \sim N$), and "Abort Lease ξ_i" (where $i = 1 \sim N$).

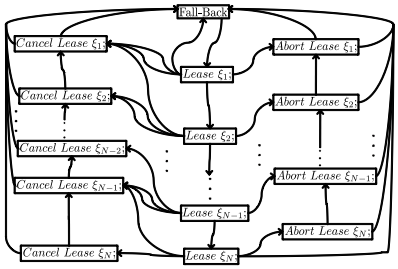
2. Initially, the Supervisor dwells in location "Fall-Back", and all data state variables initial values are zero.

3. When in location "Fall-Back", if an event $evt\xi_N To\xi_0 Req$ is received (which is sent by the Initializer requesting for entering risky-locations, see the descriptions for A_{initzr} in the following paragraph), and the Supervisor has been continuously dwelling in "Fall-Back" for at least $T^{\mathsf{min}}_{\text{fb},0}$, and the application dependent proposition $ApprovalCondition$ holds, then the Supervisor transits to location "Lease ξ_1". Along this transition[3], the Supervisor sends out event $evt\xi_0 To\xi_1 LeaseReq$, requesting leasing Participant ξ_1.

[2] Note as mentioned in Section 2.2.2, all data state variable names and location names are local to the corresponding hybrid automata. For example, A_{supvsr}'s "Fall-Back" location is not A_{initzr}'s "Fall-Back" locations, although the two locations has the same name. Likewise, A_{supvsr}'s t_{clk} data state variables not A_{initzr}'s t_{clk} data state variable of A_{initzr}.

[3] In fact, this "transition" includes two consecutive transitions, the first one is on receiving event $evt\xi_N To\xi_0 Req$, Supervisor enters an *intermediate location* of 0 dwelling time; and then transit from this intermediate location to "Lease ξ_1" and send out $evt\xi_0 To\xi_1 LeaseReq$. For narrative simplicity, in the following, such intermediate locations between two consecutive events are not elaborated.

Table 2.1. Specification of Supervisor

Role	Conceptual Description of Behaviors	Hybrid Automata Specifications
Supervisor	Conceptually, the Supervisor ξ_0 shall start from a "Fall-Back" location. Whenever the Initializer ξ_N requests leasing itself to enter risky-locations, the Supervisor shall lease Participants ξ_1, ξ_2, ..., ξ_{N-1} according to PTE ordering first. After all $\xi_1 \sim \xi_{N-1}$ are leased (i.e. $\xi_1 \sim \xi_{N-1}$ enter respective risky-locations), the Supervisor approves ξ_N's lease request to enter risky-location. The Initializer ξ_N can also request to cancel the leases; or when an application dependent proposition $ApprovalCondition$ is violated (e.g. in laser tracheotomy wireless CPS, $ApprovalCondition$ means blood oxygen level SpO_2 is higher than threshold Θ_{SpO_2}), Supervisor ξ_0 can abort leases. Lease cancellations/aborts are conducted in the reverse PTE order.	

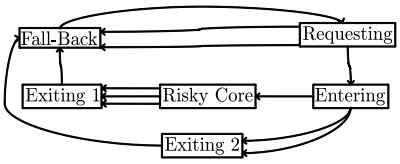
4. When in location "Lease ξ_i" (where $i = 1 \sim N-1$), the behavior of Supervisor can be described by Fig. 2.3 (a).

5. When in location "Lease ξ_N", the behavior of Supervisor can be described by Fig. 2.3 (b).

6. When in location "Cancel Lease ξ_i" (where $i = 1 \sim N$), the behavior of Supervisor can be described by Fig. 2.3 (c).

7. When in location "Abort Lease ξ_i" (where $i = 1 \sim N$), the behavior of Supervisor can also be described by Fig. 2.3 (c), except that every occurrence of "Cancel" is replaced by "Abort".

Table 2.2. Specification of Initializer

Role	Conceptual Description of Behaviors	Hybrid Automata Specifications
Initializer	Conceptually, the Initializer ξ_N shall start from a "Fall-Back" location. It can randomly request to lease itself to enter risky-locations. If this request is approved by the Supervisor ξ_0, ξ_N enters risky-locations. The dwelling in risky-locations can be cancelled by ξ_N or aborted by ξ_0 at any time; otherwise, ξ_N returns to "Fall-Back" when the lease expires.	

Initializer:

1. A_{initzr}'s location set V_{initzr} include the following locations: "Fall-Back", "Requesting", "Entering", "Risky Core", "Exiting 1", and "Exiting 2". $V_{\text{initzr}}^{\text{risky}}$ include

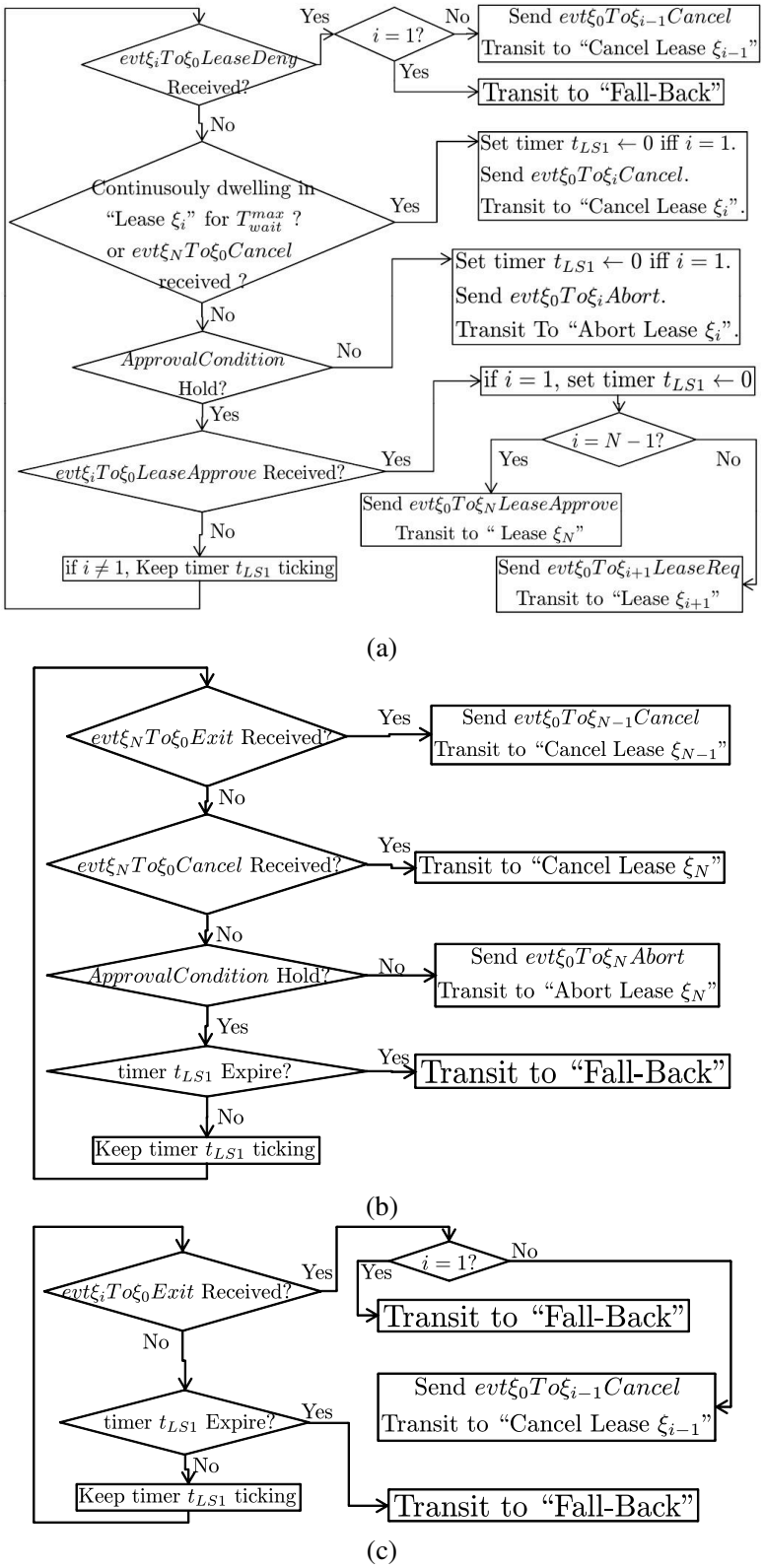


Figure 2.3. Flow block diagram at location (a) "Lease ξ_i" ($i = 1 \sim N-1$); (b) "Lease ξ_N"; (c) "Cancel Lease ξ_i" ($i = 1 \sim N$). Note "t_{LS1} expire" means $t_{LS1} \geqslant T^{\max}_{\mathrm{LS1}}$.

location "Risky Core" and "Exiting 1"; all other locations belong to $V_{\text{initzr}}^{\text{safe}}$.

2. Initially, the Initializer ξ_N dwells in location "Fall-Back"; and all data state variables initial values are zero.

3. When in location "Fall-Back" with continuous dwelling duration over $T_{\text{fb},N}^{\mathsf{min}}$, the Initializer ξ_N can send event $evt\xi_N To\xi_0 Req$ and transit to "Requesting" at any time.

4. When in location "Requesting", the Initializer ξ_N can send event $evt\xi_N To\xi_0 Cancel$ and transit back to "Fall-Back" at any time. Secondly, if ξ_N dwells continuously in "Requesting" for $T_{\text{req},N}^{\mathsf{max}}$, it will automatically transit back to "Fall-Back". Thirdly, if event $evt\xi_0 To\xi_N LeaseApprove$ is received, ξ_N transits to "Entering".

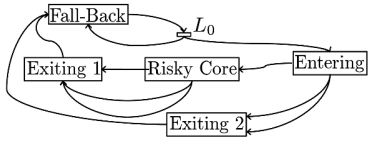
5. When in location "Entering", the Initializer ξ_N can send event $evt\xi_N To\xi_0 Cancel$ and transit to "Exiting 2". Secondly, if $evt\xi_0 To\xi_N Abort$ is received, ξ_N also transits to "Exiting 2". Thirdly, if ξ_N dwells continuously in "Entering" for $T_{\text{enter},N}^{\mathsf{max}}$, it transits to "Risky Core".

6. When in location "Risky Core", the Initializer ξ_N can send event $evt\xi_N To\xi_0 Cancel$ and transit to "Exiting 1". Secondly, if $evt\xi_0 To\xi_N Abort$ is received, ξ_N also transits to "Exiting 1". Thirdly, if ξ_N dwells continuously in "Risky Core" for $T_{\text{run},N}^{\mathsf{max}}$, it also transits to "Exiting 1".

7. When in location "Exiting 1" or "Exiting 2", the Initializer ξ_N must continuously dwell in the location for $T_{\text{exit},N}$, and then transit to "Fall-Back" and send event $evt\xi_N To\xi_0 Exit$.

Participant:

Table 2.3. Specification of Participant

Role	Conceptual Description of Behaviors	Hybrid Automata Specifications
(ith) Participant	Conceptually, a Participant ξ_i ($i = 1 \sim N-1$) shall start from a "Fall-Back" location. Upon receiving lease request from the Supervisor ξ_0, and if the lease is approved, ξ_i enters risky-locations. The dwelling in risky-locations can be cancelled by the Initializer ξ_N or aborted by the Supervisor ξ_0 at any time; otherwise, ξ_i returns to "Fall-Back" when the lease expires.	 Intermediate Location btw two events. Cost 0 time.

1. $A_{\text{ptcpnt},i}$'s location set $V_{\text{ptcpnt},i}$ include the following locations: "Fall-Back", "L0", "Entering", "Risky Core", "Exiting 1", and "Exiting 2". $V^{\text{risky}}_{\text{ptcpnt},i}$ include location "Risky Core" and "Exiting 1"; all other locations belong to $V^{\text{safe}}_{\text{ptcpnt},i}$.

2. Initially, Participant ξ_i dwells in location "Fall-Back"; and all data state variables initial values are zero.

3. When in location "Fall-Back" with continuous dwelling duration over $T^{\mathsf{min}}_{\text{fb},i}$, upon receiving event $evt\xi_0To\xi_iLeaseReq$, ξ_i transits to a temporary location "L0".

4. When in "L0", if an application dependent proposition $ParticipationCondition$ sustains, ξ_i sends event $evt\xi_iTo\xi_0LeaseApprove$ and transits to "Entering"; otherwise, ξ_i sends event $evt\xi_iTo\xi_0LeaseDeny$ and transits back to "Fall-Back".

5. When in location "Entering", if event $evt\xi_0To\xi_iCancel$ or $evt\xi_0To\xi_iAbort$ is received, ξ_i transits to "Exiting 2". Otherwise, if ξ_i dwells continuously in "En-

tering" for $T^{\mathsf{max}}_{\mathrm{enter},i}$, it transits to "Risky Core".

6. When in location "Risky Core", if event $evt\xi_0To\xi_iCancel$ or $evt\xi_0To\xi_iAbort$ is received, ξ_i transits to "Exiting 1". Otherwise, if ξ_i dwells continuously in "Risky Core" for $T^{\mathsf{max}}_{\mathrm{run},i}$, it also transits to "Exiting 1".

7. When in location "Exiting 1" or "Exiting 2", Participant ξ_i must continuously dwell in the location for $T_{\mathrm{exit},i}$, and then transit to "Fall-Back" and send event $evt\xi_iTo\xi_0Exit$.

2.4.2 Design Pattern Validity

We now analyze the validity of the proposed design pattern. As mentioned before, the main threat to PTE wireless CPS is the unreliable wireless communications. Event reception between the Supervisor, Initializer, and Participants can be lossy. If some important events are not received, the holistic system can enter an inconsistent state, which jeopardizes PTE safety rules.

A main contribution of this work is that we prove that by properly configuring the time constants of the aforementioned A_{supvsr}, A_{initzr}, and $A_{\mathrm{ptcpnt},i}$, PTE safety rules are guaranteed despite any communication faults. Specifically, we have the following result.

THEOREM 2.1 (DESIGN PATTERN VALIDITY) *Given a hybrid system $\mathcal{H}$ of ξ_0 as "Supervisor" (i.e. behaves per A_{supvsr}), ξ_N ($N \geq 2$) as "Initializer" (i.e. behaves per A_{initzr}), and ξ_i ($i = 1, 2, \ldots, N-1$) as "Participants" (i.e. behaves per $A_{\mathrm{ptcpnt},i}$).*

Suppose $\mathcal{H}$ starts with all entities (i.e. $\xi_0 \sim \xi_N$) residing in location "Fall-Back", and satisfies conditions c1 $\sim$ c7:

c1. All configuration time constants ($T^{\mathsf{max}}_{\mathrm{wait}}$, $T^{\mathsf{min}}_{\mathrm{fb},0}$, $T^{\mathsf{max}}_{\mathrm{LS1}}$, $T^{\mathsf{max}}_{\mathrm{req},N}$, $T^{\mathsf{min}}_{\mathrm{fb},i}$, $T^{\mathsf{max}}_{\mathrm{enter},i}$, $T^{\mathsf{max}}_{\mathrm{run},i}$, $T_{\mathrm{exit},i}$, where $i = 1 \sim N$) are positive.

c2. $T^{\mathsf{max}}_{\mathrm{LS1}} \stackrel{\mathsf{def}}{=} T^{\mathsf{max}}_{\mathrm{enter},1} + T^{\mathsf{max}}_{\mathrm{run},1} + T_{\mathrm{exit},1} > NT^{\mathsf{max}}_{\mathrm{wait}}$.

c3. $(N-1)T^{\mathsf{max}}_{\mathrm{wait}} < T^{\mathsf{max}}_{\mathrm{req},N} < T^{\mathsf{max}}_{\mathrm{LS1}}$.

c4. $\forall i \in \{1, 2, \ldots, N\}$, *there is*

$$(i-1)T^{\mathsf{max}}_{\mathrm{wait}} + T^{\mathsf{max}}_{\mathrm{enter},i} + T^{\mathsf{max}}_{\mathrm{run},i} + T_{\mathrm{exit},i} \leq T^{\mathsf{max}}_{\mathrm{LS1}}.$$

c5. $\forall i \in \{1, 2, \ldots, N-1\}$, *there is*

$$T^{\mathsf{max}}_{\mathrm{enter},i} + T^{\mathsf{min}}_{\mathrm{risky}:i\to i+1} < T^{\mathsf{max}}_{\mathrm{enter},i+1}.$$

c6. $\forall i \in \{1, 2, \ldots, N-1\}$, *there is*

$$\begin{aligned} T^{\mathsf{max}}_{\mathrm{enter},i} + T^{\mathsf{max}}_{\mathrm{run},i} \quad > \quad & T^{\mathsf{max}}_{\mathrm{wait}} + T^{\mathsf{max}}_{\mathrm{enter},i+1} + T^{\mathsf{max}}_{\mathrm{run},i+1} \\ & + T_{\mathrm{exit},i+1}. \end{aligned}$$

c7. $\forall i \in \{1, 2, \ldots, N-1\}$, *there is* $T_{\mathrm{exit},i} > T^{\mathsf{min}}_{\mathrm{safe}:i+1\to i}$.

Then we have:

Claim 1 (Safety): *Even if events sent* between *entities can be arbitrarily lost, $\mathcal{H}$ still guarantees PTE safety rules. That is, every entity's continuous dwelling time in risky-*

locations is upper bounded by $T^{\mathsf{max}}_{\mathrm{wait}} + T^{\mathsf{max}}_{\mathrm{LS1}}$*, and the PTE full ordering of* $\xi_1 < \xi_2 < \ldots < \xi_N$ *is maintained.*

Claim 2 (Liveness): *Let* $P^{\mathrm{PER}}_{N,0}$ *denote the packet error rate of the communication channel from the "Initializer"* ξ_N *to "Supervisor"* ξ_0*. If* $P^{\mathrm{PER}}_{N,0} < 100\%$*, i.e.* ξ_N *can send events to* ξ_0 *after all. Then i) suppose at* t_0 *all entities (i.e.* $\xi_0 \sim \xi_N$*) reside in "Fall-Back", then starting from* t_0*, every* $T^{\mathsf{min}}_{\mathrm{fb},N} + T^{\mathsf{max}}_{\mathrm{req},N}$ *second,* ξ_N *has at least one chance to send* $evt\xi_N To \xi_0 Req$ *to* ξ_0*, until* ξ_0 *leaves location "Fall-Back"; ii) suppose* ξ_0 *non-zeno-ly leaves location "Fall-Back" at* t_{00} *(i.e.* ξ_0 *is not at "Fall-Back" at* t^+_{00}*), let* $T_{\mathrm{reset}} \overset{\mathsf{def}}{=} (N-1)T^{\mathsf{max}}_{\mathrm{wait}} + T^{\mathsf{max}}_{\mathrm{LS1}} + T^{\mathsf{min}}_{\mathrm{fb},N} + T^{\mathsf{max}}_{\mathrm{req},N} + T^{\mathsf{max}}_{\mathrm{enter},N} + T^{\mathsf{max}}_{\mathrm{run},N} + T_{\mathrm{exit},N}$*, then* $\exists t \in (t_{00}, t_{00} + T_{\mathrm{reset}}]$*, such that all entities (i.e.* $\xi_0 \sim \xi_N$*) return to location "Fall-Back" at* t*.*

Proof: The sketch of the proof is as follows.

First we can prove if the given parameters satisfy Conditions c1 $\sim$ c7, and that all entities start from "Fall-Back" location, the system will reset itself to "Fall-Back" within $T^{\mathsf{max}}_{\mathrm{wait}} + T^{\mathsf{max}}_{\mathrm{LS1}}$ every time $evt\xi_0 To \xi_1 LeaseReq$ happens. This is mainly because of the leases: even if messages are lost, leases will expire to guarantee the return to "Fall-Back" of the Initializer and every Participant.

Second, we prove between any two consecutive $evt\xi_0 To \xi_1 LeaseReq$ events (or the last such event and time ∞), any entity can only dwell in the risky-locations for once.

Third, due to Conditions c1 $\sim$ c7, for each ξ_i and ξ_{i+1} ($i = 1 \sim N-1$), the aforementioned single dwelling intervals of ξ_i and ξ_{i+1} satisfies PTE enter-risky/exit-

risky safeguard interval requirements.

The detailed proof appears in Appendix 5.4 of Chapter 5. ■

2.4.3 Methodology to Transform Design Pattern into Specific Designs

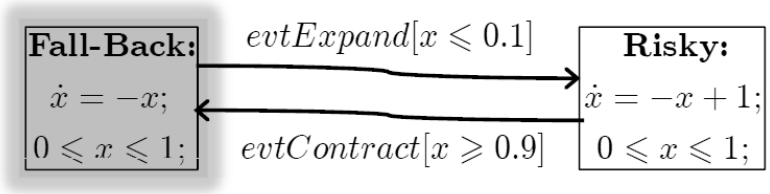
In the conference version of this work [T+13a], we further proposed a methodology to transform the aforementioned design pattern hybrid automata A_{supvsr}, A_{initzr}, and $A_{\text{ptcpnt},i}$ into specific PTE compliant wireless CPS designs. We call this methodology "*elaboration*".

The intuition of elaboration is that every location v of A_{supvsr}, A_{initzr}, and $A_{\text{ptcpnt},i}$ can be expanded with a child hybrid automata A'. As long as A' is sufficiently independent (i.e. orthogonal) from the rest part of A_{supvsr}, A_{initzr}, and $A_{\text{ptcpnt},i}$, it will not interfere the design pattern's guarantee on PTE safety rules.

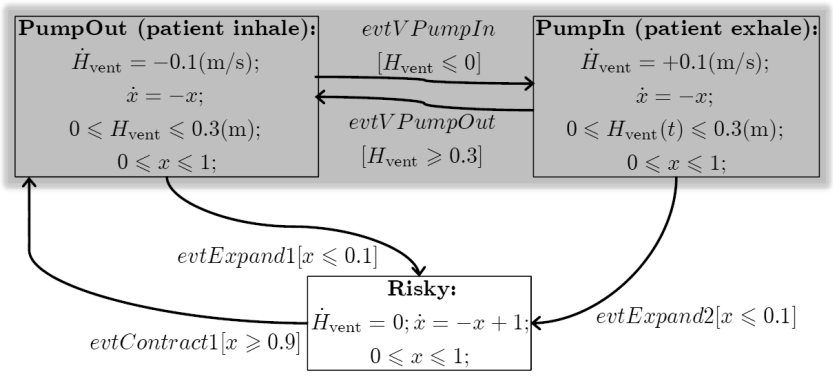
Fig. 2.4 illustrates an example of elaboration. Denote the hybrid automaton of Fig. 2.2 to be A'_{vent}. We use A'_{vent} to elaborate hybrid automaton A of Fig. 2.4 (a) at location "Fall-Back". The resulted elaboration is the hybrid automaton A'' of Fig. 2.4 (b).

The formal description on elaboration is provided in Appendix 5.5 of Chapter 5 for reader's convenience. One important feature of this elaboration methodology is summarized by Theorem 2 in Appendix 5.5 of Chapter 5. Sketch of Theorem 2 is represented in the following for reader's convenience:

Sketch of Theorem 2 (Design Pattern Compliance): if the design pattern hybrid automata (i.e. A_{supvsr}, A_{initzr}, and $A_{\text{ptcpnt},i}$) satisfy Condition c1 $\sim$ c7 of Theorem 2.1, hence guarantee PTE safety rules and liveness described in Theorem 2.1 Claim 1 and



(a)



(b)

Figure 2.4. Elaboration Example (compare the shaded areas in (a) and (b)). (a) Hybrid Automaton A, which has one data state variable x; the shaded location is to be elaborated. (b) Hybrid Automaton A'', which is the elaboration of A (see (a)) at location "Fall-Back" with hybrid automaton A'_{vent} (see Fig. 2.2); note no edge exists from "Risky" to "PumpIn" because "PumpIn" is not an initial location of A'_{vent}.

2, then any specific design resulted from elaborating the design pattern hybrid automata still guarantees the same PTE safety rules and liveness.

2.5 Case Study

Next, we carry out two case studies to validate our proposed leasing based hybrid design pattern approach. The case studies are respectively on medical CPS and control CPS, two major categories of CPS applications.

2.5.1 Laser Tracheotomy Wireless Medical CPS

Scenario and Design:

In laser tracheotomy wireless medical CPS (see Fig. 2.5 (a) for the application layout), a patient is under anesthesia, hence must be connected to a ventilator to breathe oxygen. However, a surgeon may randomly request a laser-scalpel to emit laser, to cut the patient's trachea. Therefore, PTE safety rules apply as follows. Before the emission of laser, the ventilator must have paused for at least $T^{\mathsf{min}}_{\mathrm{risky}:1\to 2}$ (we regard the ventilator as entity ξ_1, the Participant; and the laser-scalpel as entity ξ_2, the Initializer); after the emission of laser, the ventilator must wait for at least $T^{\mathsf{min}}_{\mathrm{safe}:2\to 1}$ before resuming. Otherwise, if high concentration of oxygen in the patient's trachea (due to ventilation) is present when laser emits, the patient's trachea can catch fire. In addition, the durations that the laser-scalpel can continuously emit and that the ventilator can continuously pause shall respectively be upper-bounded by a constant.

The ventilator and the laser-scalpel are wirelessly connected via a base station, which also plays the role of the Supervisor (i.e. entity ξ_0). The supervisor/initializer can

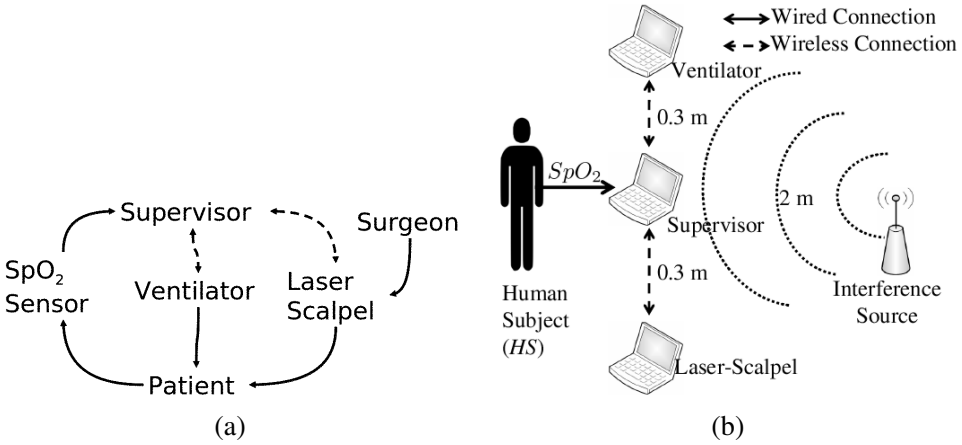


Figure 2.5. (a) Laser tracheotomy wireless medical CPS, figure quoted from [L$^+$12]; (b) Emulation Layout

abort/cancel laser emission at any time (e.g., when the supervisor detects the patient's blood oxygen level SpO_2 reaches below a threshold, it can immediately request aborting laser emission and resuming ventilation), but the PTE safety rules must be maintained.

On the other hand, because the supervisor, laser-scalpel, and ventilator are connected via wireless, message losses are possible. Therefore, we carry out our leasing based design approach, so that even with message losses, the wireless CPS can maintain PTE safety rules.

Interested readers can refer to Appendix 5.6 of Chapter 5 for the resulted detailed design hybrid automata diagrams.

We configure the time parameters of the above detailed design hybrid automata according to common-sense laser tracheotomy requirements [J$^+$12] as follows. For the Supervisor (i.e. the laser tracheotomy supervisor), $T^{\mathsf{min}}_{\mathrm{fb},0} = 13$(s), $T^{\mathsf{max}}_{\mathrm{wait}} = 3$(s). For the Initializer (i.e. the laser-scalpel), $T^{\mathsf{max}}_{\mathrm{req},2} = 5$(s), $T^{\mathsf{max}}_{\mathrm{enter},2} = 10$(s), $T^{\mathsf{max}}_{\mathrm{run},2} = 20$(s),

$T_{\text{exit},2} = 1.5$(s). For Participant 1 (i.e. the ventilator), $T^{\mathsf{max}}_{\text{enter},1} = 3$(s), $T^{\mathsf{max}}_{\text{run},1} = 35$(s), $T_{\text{exit},1} = 6$(s). The PTE enter-risky/exit-risky safeguard intervals are $T^{\mathsf{min}}_{\text{risky:}1\to2} = 3$(s) and $T^{\mathsf{min}}_{\text{safe:}2\to1} = 1.5$(s).

Per Theorem 2 (see Appendix 5.5 of Chapter 5, the above configurations guarantee PTE safety rules. To further validate this, we implemented and carried out emulations of the above design.

Emulation Setup:

Fig. 2.5 (b) illustrates the layout of our emulation. The laser tracheotomy ventilator, supervisor, and (surgeon operated) laser-scalpel are respectively emulated by three computers. The patient is emulated by a real human subject (HS).

Instead of actually ventilating the human subject HS, the ventilator emulator displays its current hybrid automata location ("PumpOut", "PumpIn", etc.). Human subject HS watches the display and breathe accordingly.

We also emulate the following three kinds of events, which cause all other events in the emulated system.

The first is the Initializer event $evt\xi_2 To \xi_0 Req$, triggered when the laser-scalpel is in "Fall-Back" and the surgeon requests to supervisor to emit laser. In the real system, this is triggered by the surgeon's human will. In our emulation, however, this is emulated by (re-)initializing a timer T_{on} (T_{on} follows exponential distribution) whenever the laser-scalpel enters "Fall-Back". When in "Fall-Back" and T_{on} sets off, the (emulated) surgeon requests to emit laser.

The second kind is the Initializer event $evt\xi_2 To \xi_0 Cancel$, triggered when the laser-scalpel is emitting and the surgeon cancels the request to emit laser. Again in

a real system, this is triggered by the surgeon's human will. In our emulation, this is emulated by (re-)initializing a timer T_{off} (T_{off} follows exponential distribution) whenever the laser-scalpel enters "Risky Core" (i.e. starts emission). When in "Risky Core" and T_{off} sets off, the (emulated) surgeon requests to cancel laser emission.

The third kind is the Supervisor event $evt\xi_0 To\xi_i Abort$ ($i = 1 \sim N$), triggered when the supervisor is in "Lease ξ_i" location and $ApprovalCondition$ becomes false. In our emulation, the human subject HS wears an oximeter (Nonin 9843 [non]), which measures HS's blood oxygen level in real-time t ($SpO_2(t)$). The oximeter is wired to the laser tracheotomy supervisor emulator. The $ApprovalCondition$ is that the oximeter reading $SpO_2(t) > \Theta_{SpO_2}$, where Θ_{SpO_2} is set to 92%.

The supervisor, ventilator, and laser-scalpel emulators communicate with each other via wireless, with supervisor as base station, and the other two as clients. Their wireless interfaces are implemented via 2.45GHz ZigBee TMote-Sky motes [Y+08]. In addition, there is an IEEE 802.11g WiFi interference source 2 meters away from the supervisor. The interference source runs Iperf (a standard network evaluation software, see `http://iperf.sourceforge.net`) to generate 3Mbps interfering data traffic to be broadcast through a WiFi radio band overlapping with that of the ZigBee TMote-Sky motes'. Because the interference broadcast is independent from the laser tracheotomy wireless CPS communications, any packets/events between the supervisor, ventilator, and laser-scalpel emulation computers can be lost.

2.5.2 Inverted Pendulum Remote Monitoring Wireless Control CPS

Scenario and Design:

Inverted Pendulum (IP) is a metal rod (the pendulum) with one end hinged on a cart, and the other end free rotating. The cart can move along a rail (the "x-axis") to keep the hinged rod standing up-right still. Due to its inborn instability, IP is a widely adopted test bed for various control strategies, including control CPS [Qua] [F+93].

In our IP remote monitoring case study (see Fig 2.6 for the application layout), the IP (entity ξ_2, the Initializer) may randomly request for a random walk, i.e. randomly adjust the cart's *reference location* (i.e. the target stabilization location) on the rail. Because random walk is considered a risky operation, the entire duration of random walk, including $T^{\mathsf{min}}_{\mathrm{risky}:1\to2}$ seconds right before the random walk, and $T^{\mathsf{min}}_{\mathrm{safe}:2\to1}$ seconds right after the random walk, must be continuously monitored/recorded by a remote video camera (entity ξ_1, the Participant). The video record can be used for real-time decision making, or for future analysis, debugging, or accident-forensics. Meanwhile, we do not allow infinite random walk, hence the duration of each random walk, and the corresponding duration of remote monitoring are upper bounded.

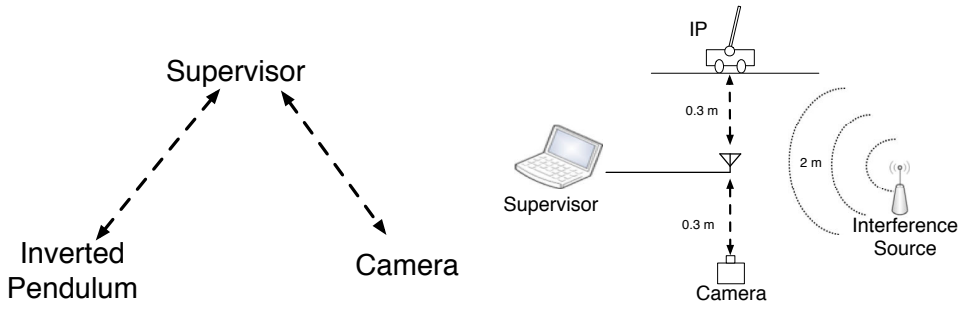


Figure 2.6. (a) Inverted Pendulum (IP) remote monitoring wireless control CPS; (b) Experiment Layout

Similar to the laser tracheotomy case, the monitoring camera and the IP are wirelessly connected to the supervisor (entity ξ_0). The supervisor/initializer can abort/cancel

the random walk at any time, but the PTE safety rules must be maintained.

Meanwhile, as the supervisor, IP, and camera are connected via wireless, message losses are possible. Therefore, we carry out our leasing based design approach, so that even with message losses, the wireless CPS can maintain PTE safety rules.

The detailed design of hybrid automata in IP remote monitoring (see Appendix 5.6 of Chapter 5) is similar to the laser tracheotomy case, except that in "Risky Core" location, the IP conducts random walk, and the camera conducts video recording. We configure the time parameters of the detailed design hybrid automata as follows. For the Supervisor, $T^{\mathsf{min}}_{\mathrm{fb},0} = 0.1$(s), $T^{\mathsf{max}}_{\mathrm{wait}} = 0.1$(s). For the Initializer (i.e. the IP), $T^{\mathsf{max}}_{\mathrm{req},2} = 0.1$(s), $T^{\mathsf{max}}_{\mathrm{enter},2} = 3.0$(s), $T^{\mathsf{max}}_{\mathrm{run},2} = 20$(s), $T_{\mathrm{exit},2} = 2.5$(s). For the Participant 1 (i.e. the camera), $T^{\mathsf{max}}_{\mathrm{enter},1} = 1.0$(s), $T^{\mathsf{max}}_{\mathrm{run},1} = 35.0$(s), $T_{\mathrm{exit},1} = 6$(s). The PTE enter-risky/exit-risky safeguard intervals are $T^{\mathsf{min}}_{\mathrm{risky}:1\to2} = 1.0$(s) and $T^{\mathsf{min}}_{\mathrm{safe}:2\to1} = 1.5$(s). The above settings satisfy condition c1 $\sim$ c7 in Theorem 2.1, meanwhile allow reasonable duration length for random walk and monitoring.

Experiment Setup:

We implemented the IP remote monitoring detailed design, and carried out experiment evaluation. Fig. 2.6 (b) shows our experiment layout. The layout and settings are the same as those of our laser tracheotomy emulation, except that the laser scalpel emulator, ventilator emulator, and (laser tracheotomy) supervisor are respectively replaced by the IP, camera, and the (IP remote monitoring) supervisor.

2.5.3 Trials and Results

For laser tracheotomy wireless medical CPS (IP remote monitoring wireless control CPS), we ran two emulation (experiment) trials, each of 30 minutes duration. During the trials, the PTE safety rules are:

1. Neither ventilator pause (camera monitoring) nor laser emission (IP random walk) can last for more than 1 minute;

2. Ventilator pause (camera monitoring) duration must always properly-temporally-embedding laser emission (IP random walk) duration, with entering/exiting safeguard interval of $T^{\mathsf{min}}_{\mathrm{risky}:1\to 2} = 3$ seconds (1 second) and $T^{\mathsf{min}}_{\mathrm{safe}:2\to 1} = 1.5$ second (1.5 second).

Violation of either of the PTE safety rules is a failure.

As mentioned before, in the two trials, the emulated surgeon (IP) requests to emit/cancel-emit laser (start/cancel random walk) according to timer T_{on} and T_{off}, which are both random numbers following exponential distribution. The expectation of T_{on} is 30 seconds. The expectations of T_{off} are 18 seconds and 6 seconds respectively in the two trials.

Because of the use of our proposed leasing based design pattern, and the configuration of parameters satisfying Theorem 2 (see Appendix 5.5 of Chapter 5), although packets/events between the ventilator emulator (camera), supervisor, and laser-scalpel emulator (IP) can be arbitrarily lost, the PTE safety rules are never violated. This is shown in Table 2.4, the rows corresponding to "with Lease" always have 0 failures.

Table 2.4. PTE Safety Rule Violation (Failure) Statistics

(a) Laser Tracheotomy Emulation

Trial Mode	$E(T_{\text{off}})$ (sec)	# of Laser Emissions	# of Failures	# of $evtRunEnded$
with Lease	18	19	0	5
without Lease	18	11	4	0
with Lease	6	19	0	3
without Lease	6	12	3	0

(b) IP Remote Monitoring Experiment

Trial Mode	$E(T_{\text{off}})$ (sec)	# of IP random walks	# of Failures	# of $evtRunEnded$
with Lease	18	12	0	8
without Lease	18	11	6	0
with Lease	6	15	0	10
without Lease	6	13	7	0

1. Each trial lasts 30 minutes, and is under constant WiFi interference.
2. For each trial, the expectation $E(T_{\text{on}}) \equiv 30$(sec).
3. $evtRunEnded$ occurs when lease expiration forces the laser-scalpel (IP) to stop emitting (random walk), i.e. when lease mechanism takes effect to rescue the system from violating the PTE safety rules.

For comparison, for each case study, we also ran two additional emulation (experiment) trials with the same configurations but without using the leasing mechanism. Specifically, the ventilator (camera) does not set up a lease timer when it starts pausing (monitoring), neither does the laser-scalpel (IP) set up a lease timer when it starts emitting laser (random walk). When the surgeon's cancel laser emission event (the IP's cancel random walk event) is lost or the supervisor's abort event is lost, no one can terminate the ventilator's pause (the camera's monitoring) or the laser's emission (the IP's random walk). Thus, as shown in Table 2.4, the rows corresponding to "without Lease" all result in many failures.

To facilitate understanding of the above results, in the following, we provide some more intuitive explanations.

Without loss of generality, let us focus on the laser tracheotomy case study.

Because of leasing, the ventilator's stay in the pause state (i.e. risky-locations) expires on lease time-out; hence it will automatically return to "Fall-Back" to continue ventilating the patient, even when it is cut-off from communications. Same applies to the laser-scalpel's stay in the emission state (i.e. risky-locations). Conditions c1 $\sim$ c7 of Theorem 2.1 further guarantee that the automatic returns to "Fall-Back" of ventilator and laser-scalpel both conform to proper-temporal-embedding even under arbitrary packet/event losses.

Interested readers can refer to Appendix 5.7 of Chapter 5 for even more intuitive explanations.

2.6 Comparisons

PTE safety is a relatively new issue raised by CPS. To our best knowledge, the state-of-the-art solution is a polling based approach proposed by Kim et al. [K$^+$10].

2.6.1 Polling Based Approach

Kim et al. [K$^+$10]'s polling based approach also adopts a layout of distributed entities, with a base station and several remote entities. The base station also serves the role of the Supervisor; one remote entity is the Initializer, and the other remote entities serve the role of Participants. However, different from our leasing based approach, the Supervisor does not passively wait for messages sent from the Initializer to trigger a sequence of PTE operations. Instead, it periodically polls remote entities for their current states. The polling message is also piggy backed with a plan vector. The plan

vector is basically instructions on what the remote entity shall do in the current and future periods, assuming communication link with the base station will be broken in the future periods. Also, the remote entities cannot change their (cyber) states (though the Initializer can *request* to start/cancel a sequence of PTE operations), unless instructed by the plan vector from the Supervisor.

For example, the plan vector for laser-scalpel may set the laser-scalpel to keep emitting for the next two periods and then deactivate in the third period; whereas the plan vector for ventilator may ask the ventilator to keep pausing in the next four periods. The Supervisor coordinates these plan vectors, ensuring the PTE safety rules are guaranteed. The polling temporal sequence (exemplified by the case of laser tracheotomy) is shown in Fig 2.7.

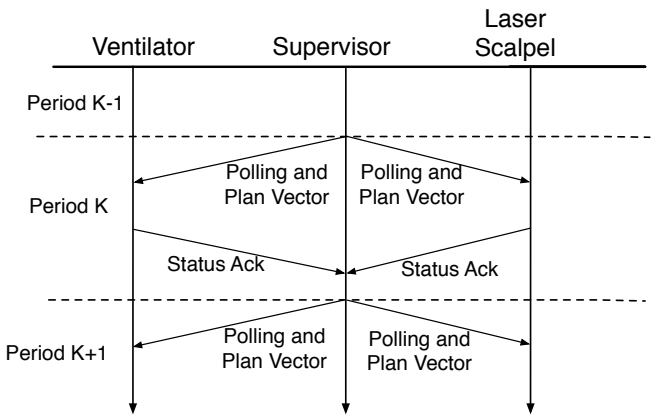


Figure 2.7. The Polling Temporal Sequence for Laser Tracheotomy Wireless CPS (quoted from [K$^+$10]).

2.6.2 Simulation Setup

We compare our leasing based approach with Kim et al. [K+10]'s polling based approach with simulation.

We reuse laser tracheotomy and IP remote monitoring described in Section 2.5 as the application background. Particularly, the PTE safety demands remain the same.

For our leasing based approach, the simulation setup matches what is described in Section 2.5. The only exception is that to improve the approach's robustness, each wireless packet is consecutively retransmitted ten times from the application layer, once per 10ms.

For the polling based approach, we follow the instructions in [K+10]. The detailed designs and parameter configurations are given in Appendix 5.8 of Chapter 5. There are two issues worth particular mentioning.

The first issue is on the choice of polling period. The longer the polling period, the less wireless messages sent per second, and hence better wireless medium occupation efficiency (when the wireless medium is benign or moderately adverse). On the other hand, the polling period can neither be too long, due to two reasons. First, the polling based approach assumes all remote entities' physical state change within a polling period is negligible (unless the plan vector instructs the remote entity to conduct a state change). Second, response time to user requests is at least one polling period; and for good user experience, under benign or moderately adverse wireless medium conditions, the response time should be within tens of milliseconds [G+06] [F+04]. Considering the above factors, we set the polling period at 50(ms).

The second issue is on the choice of other configuration parameters. Note PTE

safety rules only care about worst case time bounds (e.g., the laser emission *must* take place *at least* 1.5 seconds after the ventilator has paused). Within these time bound constraints, many feasible configurations exist for both the polling and the leasing based approaches (e.g. the laser emission *can* take place 2 seconds, or 3 seconds after the ventilator has paused). To make our comparisons fair, the polling and leasing based approach parameters are configured so that the default (i.e. when there is no cancellation nor abort, and no communication packet loss) behaviors of $\xi_1, \ldots, \xi_N$ are the same under both approaches[4].

2.6.3 Results and Analysis

Based on aforementioned analysis, emulations, and experiments, we know that both our leasing based approach and Kim et al. [K$^+$10]'s polling based approach can guarantee PTE safety rules under arbitrary wireless communication failures. However, their performance, specifically, wireless medium occupation efficiency and user experience, may be different.

To strictly quantify the two performance indicators, we further consider three wireless medium conditions: benign, moderately adverse, and adverse, respectively corresponds to a *Packet Error Rate* (PER) of 0.5%, 5%, and 50%.

For each approach (leasing based vs. polling based), each application (laser tracheotomy, IP remote monitoring), and each wireless medium condition, we run 1000 simulation trials. In each trail, all entities start from "Fall-Back"-equivalent locations/states [5] , and then the Initializer will request to run a complete sequence of PTE operations.

[4]Here we ignore the delay differences caused by polling period and packet transmission, which is in $\leqslant$ 50ms range.

[5] For leasing based approach, this means the Supervisor and Initializer both start from the "Fall-Back"

The Initializer will keep requesting until approval is received from the Supervisor [6] . Suppose the Initializer started to request at t_0, and first received the Supervisor's approval at t_1, we call the duration $(t_1 - t_0)$ the "*initialization response time*".

Once the Initializer enters its "Risky-Core" location/state, it (re-)initializes a timer T_{off}. T_{off} follows exponential distribution with an expectation of 18(s). If T_{off} sets off and the Initializer is still in "Risky-Core" location/state, the Initializer requests to cancel the risky activity (i.e. laser emission or IP random walk). Suppose the Initializer requests to cancel the risky activity at t_2, and suppose if no message is lost, the Participant can return to "Fall-Back" location/state at t_3^*. Meanwhile, denote the actual time the Participant returns to "Fall-Back" location/state to be t_3. Then we call the difference $(t_3 - t_3^*)$ the "*extra suffering time*". Extra suffering time quantifies the extra suffering time endured by the Participant due to message losses in the cancellation process. That is, the Initializer has cancelled the risky activity, but the Participant is not notified, hence has to suffer longer, waiting for the leasing or polling mechanism to return it to "Fall-Back" location/state.

We use initialization response time and extra suffering time to quantify user experiences. For both metrics, the shorter means better user experience (quicker response or less extra suffering). Wireless medium occupation, however, is quantified by the ratio of time used for wireless transmission during the whole interval of $[t_0, t_3]$. The higher the ratio, the worse the wireless medium occupation efficiency (i.e. the more wasteful of the wireless medium).

---

location (see Fig. 7.14, 7.15, 7.17, 7.18 in Chapter 5); the Participant starts from "PumpOut" location in the case of laser tracheotomy (see Fig. 7.16 in Chapter 5) , and from "Fall-Back" location in the case of IP remote monitoring (see Fig. 7.19 in Chapter 5). For polling based approach, this means the Supervisor, Initializer, and Participant all start from the "Fall-Back" state (see Fig. 7.20 $\sim$ 7.25 in Chapter 5).

[6] For leasing based approach, this means message $evt\xi_0 To \xi_2 LeaseApprove$ is received by the Initializer (see Fig. 7.15, 7.18 in 5). For polling based approach, this means message $evt\xi_0 Ack$ is received by the Initializer (see Fig. 7.21, 7.24 in 5).

The simulations results for laser tracheotomy are summarized in Fig. 2.8. We can make several observations from these figures.

First, our leasing based approach incurs less wireless medium occupation ratio than polling based approach. As shown in Fig. 2.8 (a)(d), leasing's wireless medium occupation ratio is upper bounded by 0.65%; while polling's is lower bounded by 5.69%. Later we will see the impact of this difference on system scalability. This difference is intuitive: our leasing based approach is an event based approach, no messages are sent unless certain event takes place; while polling based approach sends messages every period no matter what. The benefit of wireless medium occupation efficiency will become more significant when we evaluate system scalability (see later paragraphs).

Second, when wireless medium is benign (e.g. PER = 0.5%) or moderately adverse (e.g. PER = 5%), leasing based approach can provide slightly better user experience. As shown in Fig. 2.8 (b)(e), leasing's initialization response time is upper bounded by 44ms, while polling's is lower bounded by 100ms; and as shown in Fig. 2.8 (c)(f), leasing's extra suffering time statistics (1st/3rd quartile, median, maximum) are all roughly one order of magnitude shorter than polling's. This is because under benign or moderately adverse wireless medium condition, packet loss is rare. Under leasing based approach, an event can be immediately responded to; while under polling based approach, every response must take at least one polling period. However, when wireless medium is severely adverse (e.g. PER = 50%), polling based approach provides better user experience than leasing based approach (see Fig. 2.8 (b)(c)(e)(f), leasing's maximum initialization response time and extra suffering time can respectively reach 10s and 31.03s, while polling's only respectively reach 2.52s and 0.671s). This is intuitive: polling based approach is basically continuously retransmitting messages every period,

hence has better chance of delivering messages when packet loss rate is high[7].

Finally, we also evaluate the scalability of the two approaches. We study an N-IP remote monitoring scenario, where N pairs of IP-camera are being coordinated by a Supervisor. Fig. 2.9 compares the performances of leasing and polling based approaches when N scales up from 1 to 12 (wireless medium is set to moderately adverse, i.e. PER = 5%). We can see that polling based approach uses up wireless bandwidth quickly as N grows; when $N = 12$, nearly all wireless bandwidth is used up (91.72% in the worst case). In contrast, our leasing based approach only uses a small portion of the wireless bandwidth for all N values (5.86% in the worst case). This matches intuition, as our leasing based approach carries out event based interrupt-like communication, which is well-known to be more communication resource thrift than polling.

2.7 Related Work

Lease based design pattern is originally proposed by Gray et al. [G$^+$89] and is used to provide efficient consistent access to cached data in distributed computer systems. In the past decades, various leasing based distributed computer systems have been implemented to achieve system consistency [T$^+$97, A$^+$05, K$^+$08, CWR06, B$^+$07, A$^+$10]. As pointed out in Section 2.1, all these distributed computer systems are fundamentally different from CPS due to following reasons: 1) check-point and roll-back, two funda-

[7] It is worth noting that initialization response time refers to the duration of a request-reply sequence taking place at the very beginning of PTE (and assuming all entities starts from "Fall-Back"). Therefore, it is irrelevant to most of PTE configuration parameters except $T^{\max}_{\mathrm{req},N}$ (request time out). In both leasing and polling schemes, $T^{\max}_{\mathrm{req},N}$ are the same, hence the comparison is fair. Meanwhile, extra suffering time refers to the *difference* between actual response time and the ideal response time when PER = 0. Most PTE parameters are cancelled out due to the subtraction, leaving only ξ_1 (Participant)'s maximum dwelling time in risky-location relevant: in the worst case, the patient (camera) has to suffer this maximum dwelling time longer than the PER = 0 case. Again, this parameter settings are the same for both leasing and polling schemes, hence the comparison is fair.

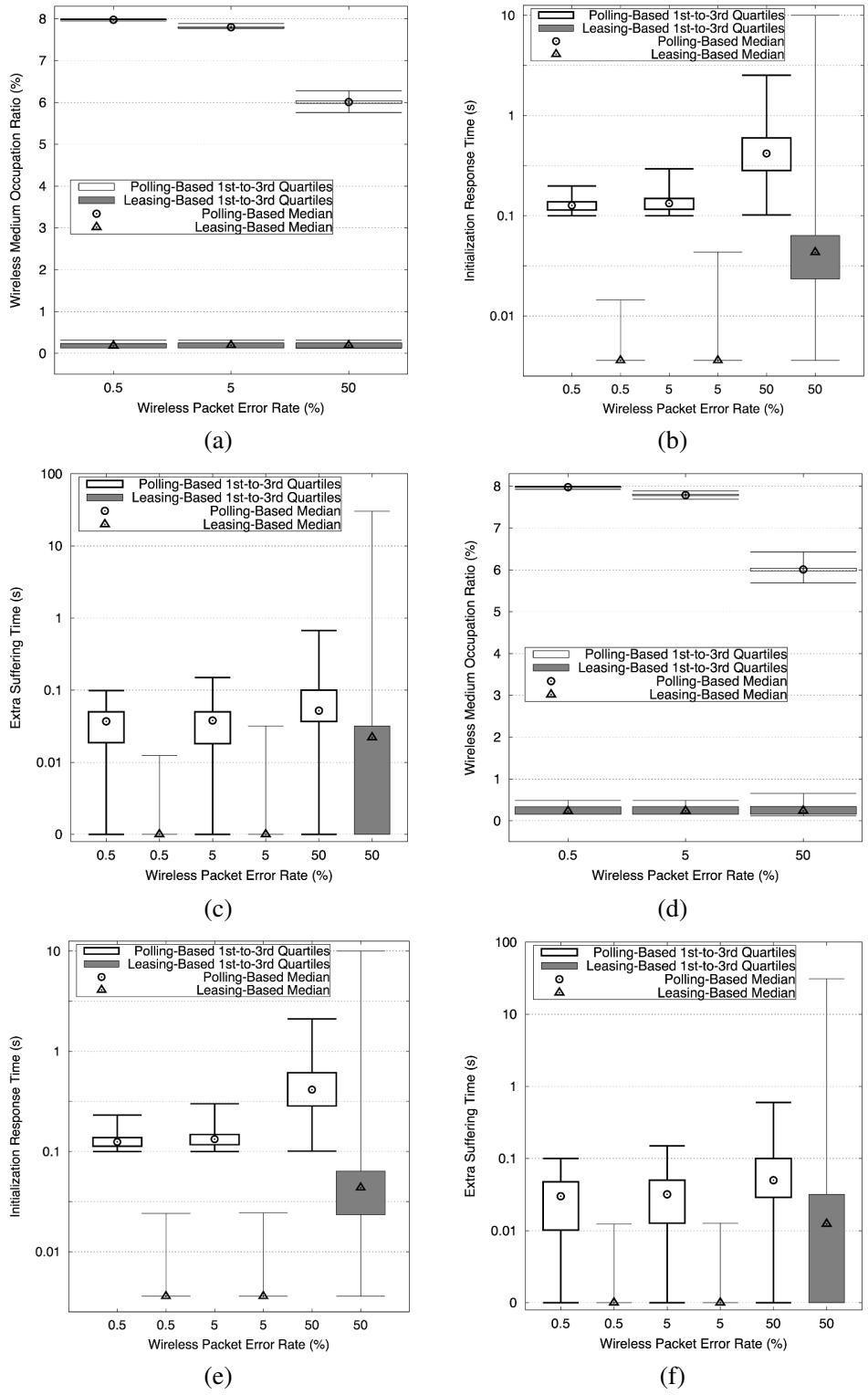


Figure 2.8. Comparisons between Leasing-Based Approach and Polling-Based Approach in Laser Tracheotomy ((a) $\sim$ (c)) and IP Remote Monitoring ((d) $\sim$ (f)) Wireless CPS

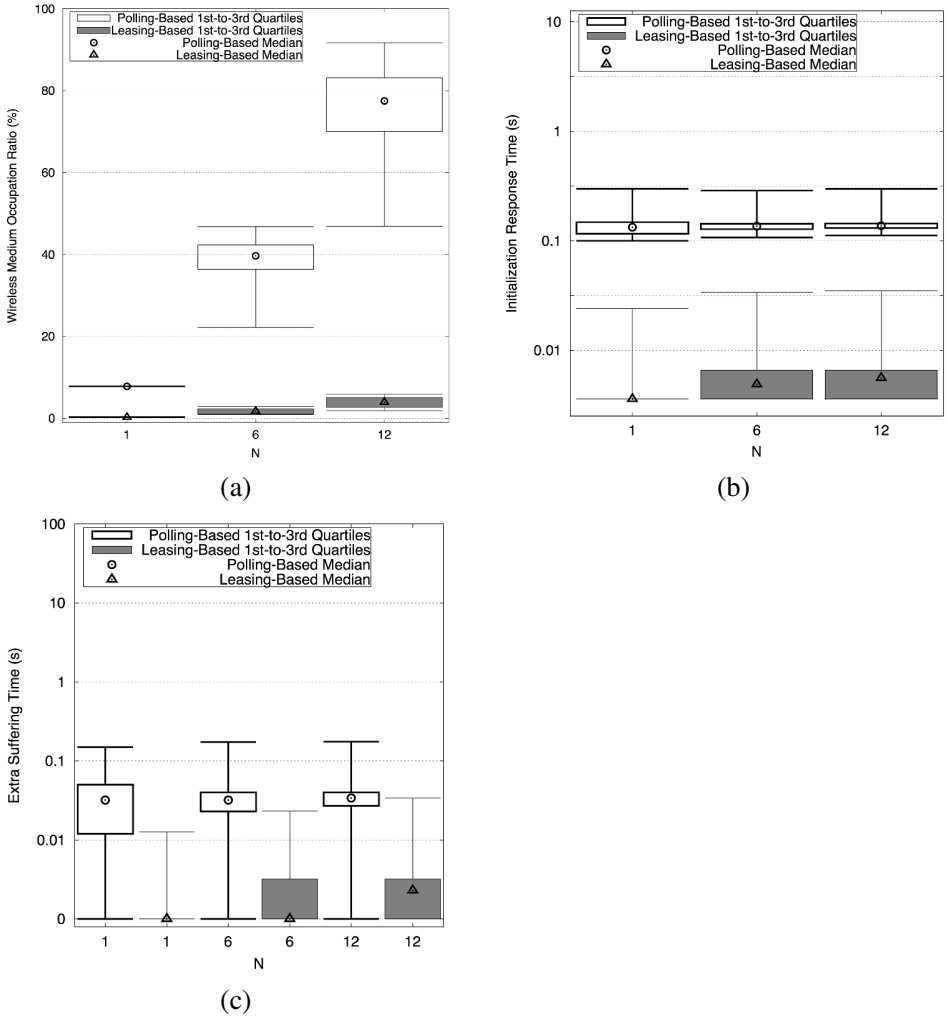


Figure 2.9. Scalability Comparisons between Leasing-Based Approach and Polling-Based Approach (N is the number of IPs being remotely monitored)

mental operations in lease-based distributed computer systems are often impossible for CPS (e.g. we cannot revive a killed patient); 2) PTE temporal ordering, particularly the continuous-time duration requirements (such as the minimal safeguard interval) are usually not present for distributed computer systems (which instead focus on logical-time, aka causal precedences).

Although formal methods have been applied to design pattern research [Gar90, Mik98], hybrid modeling is mostly used for verification [A+93, H+95, A+96, GGH+03, L+12]. Recently, Tichakorn [Tic10] proposes a subclass of hybrid automata for a class of hybrid control systems in which certain control actions occur roughly periodically and applied it to verify the safety of an autonomous vehicle. However, the focus there is still verification, rather than design.

The content of this chapter is published in [T+13a, T+15].

2.8 Conclusion and Future Work

In this chapter, we formalize a temporal interlocking/mutual-exclusion pattern called PTE safety rules for CPS physical component interactions. We propose a leasing based design pattern to guarantee PTE safety rules in wireless CPS, as part of the effort to address challenges arising from poor reliability of wireless communication on CPS' mission/life criticality. We derive a set of closed-form constraints, and prove that as long as system parameters are configured to satisfy these constraints, PTE safety rules are guaranteed under arbitrary wireless communication faults. Furthermore, we develop hybrid modeling approaches to describe the design patterns, and develop a formal methodology to elaborate the design pattern into specific designs that provide PTE safety guarantees.

Our case studies on laser tracheotomy wireless CPS and inverted pendulum remote monitoring validate the proposed design methodology. We also compare our solutions with a polling based solution in terms of wireless resource occupation and user experience metrics. The comparison results show that our solutions has much less wireless resource occupation than the polling based approach; and in terms of user experience metrics, the polling based solution performs better under severely adverse wireless medium conditions, while ours performs better under benign or moderately adverse wireless medium conditions. As our future work, we will investigate additional network protocol technologies to enhance wireless communication reliability, hence to further improve the performance of our leasing based approach and the polling based approach. We will also explore more application domains for the proposed design pattern, such as chemical plant safety, anesthesiology, control, where timing (time duration) is an important parameter in defining safety rules.

Chapter 3

Profiling Cross-Domain Noise for Gray Box Two-Level Control CPS

3.1 Introduction

In this chapter, we focus on another major category of CPS, control CPS, where computer systems control physical objects in real-time. Instead of achieving fault avoidance by proper system design, such as guaranteeing PTE safety rules in Chapter 2, this work attempts to profile the fault propagation in CPS. Specifically, we are interested in studying *cross-domain noise*: that comes from the physical subsystem, propagates through the cyber subsystem, and goes back to the physical subsystem. Cross-domain noise is hard to profile when the cyber subsystem is a gray box, hence cannot be explicitly modeled.

This work focuses on two-level control CPS, a widely used control CPS architecture. Specifically, we assume a classic control CPS architecture described by Fig. 3.1. It

consists of a "physical" control subsystem (simplified as the "*physical subsystem*" in the following) and a "cyber" computing subsystem (simplified as the "*cyber subsystem*" in the following). The physical and cyber subsystems form a two-level control loop. The physical subsystem conducts the *inner control loop*, which carries out fine-time-grained sensing (the "local sensing" in the figure) and actuating of the *plant* (i.e. the physical object being controlled). The cyber subsystem conducts the *outter control loop*, which carries out coarse-time-grained reference point updates. For simplicity, in the following, we call the control CPS architecture of Fig. 3.1 the *two-level control CPS* (2L-CCPS) architecture.

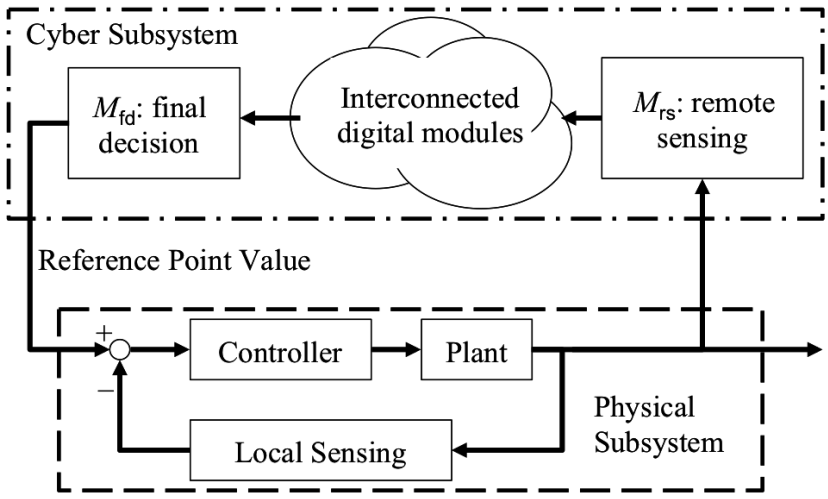


Figure 3.1. 2L-CCPS, a classic control CPS architecture

More specifically, in Fig. 3.1, things within the dashed box constitute the physical subsystem, which is the same as a conventional non-CPS control system. The external input to the physical subsystem is the *reference point* value, a vector that specifies the target state of the plant. Given the reference point value, the physical subsystem takes charge of maneuvering the plant until the plant's state reaches the reference point value.

For example, suppose the plant is a cart, with vector $(x_1, x_2)^\mathsf{T}$ as its state, where x_1 is the cart's current location and x_2 is the cart's current velocity. A reference point value of $(10, 0)^\mathsf{T}$ commands the physical subsystem to move the cart to location 10 and stop there.

Besides the physical subsystem, things within the dot-dashed box in Fig. 3.1 constitute the cyber subsystem. Specifically, the cyber subsystem is a set of interconnected digital modules (can be both software and/or hardware, e.g. digital signal processors). We assume the cyber subsystem is a *gray box*: the internals of the digital modules are unknown, but their external interfaces and interconnections are known. Such assumption is representative, as more and more cyber subsystems are built from *commercial-off-the-shelf* (COTS) components, and/or become too complex to accurately model. This assumption aside, the interconnected digital modules collaborate to holistically form a workflow that remotely senses the plant state, processes the sensed state, and decides the new reference point value. The new reference point value is the output of the cyber subsystem to be fed back to the physical subsystem. We assume the time cost for the above work flow is negligible; and every time the cyber subsystem triggers the above work flow and outputs a new reference point value, a *reference point update event* happens.

The reference point update events happen in coarse-time-granularity: they happen discretely and stochastically. In contrast, the sensing and actuating taking place within the physical subsystem (i.e. the inner control loop) happen in fine-time-granularity. They are carried out in continuous time, or periodically with a period several orders of magnitude shorter than the average interval between two reference point update events.

For example, for a 2L-CCPS to remotely fly a drone, the drone (the physical subsystem) has its on-board fine-time-grained sensing and actuating for attitude control;

while the ground station (the cyber subsystem) uses visuals to conduct remote coarse-time-grained sensing of the drone, and to decide/command the drone where to go. *In the following, unless otherwise denoted, the "sensing" of this chapter refers to the latter, i.e., the coarse-time-grained remote sensing for computing new reference point values by the cyber subsystem.*

In practice, sensed signals are always accompanied with noises. These noises constitute a major source of errors. Noises within conventional control systems (e.g., the physical subsystem of a 2L-CCPS) are well-studied and can be well contained [HC10] [V+10] [Bub05][1]. Hence these noises are out of the scope of this work. Instead, we are interested in the noises that cross the boundaries between the cyber and physical subsystems, i.e. the so called *cross-domain noises*. Specifically, in a 2L-CCPS (see Fig. 3.1), cross-domain noises refer to those generated from the remote sensing of the plant, propagate through the cyber subsystem, and injected back to the physical subsystem as error component of the new reference point value.

Overview of Proposed Framework

This work is interested in profiling how the cross-domain noises, reshaped by the gray box cyber subsystem, impact the plant. The overall idea of our framework is as follows.

We first prepare a benchmark, i.e. a set of sample states of the plant. For each sample state of the benchmark, we carry out Monte Carlo emulation. In each emulation trial, the benchmark sample state, together with cross-domain noise, is entered into the cyber subsystem. The cyber subsystem then outputs the (noisy) next reference point

[1] In a more general sense, these noises include local sensing noises, controller output disturbances, and plant modeling errors. All are well-studied and can be well contained within the conventional control system.

value, which is fed across the domain boundary into a physical subsystem simulator to measure the risk increase. Via the above Monte Carlo emulation, we profile the relationship between cross-domain noise and risk increase. This relationship becomes a metric to evaluate the impact of the cross-domain noise.

Contributions and Basic Insights

In a more general sense, our proposed framework addresses a sub-problem of fault propagation profiling, a hot topic in system dependability research. There are a lot of works on profiling fault propagation in pure software systems [P+15] [PD12] [JL11] [D+11] [OA11] [O+10] [JS05] [H+04], or CPS [S+13] [A+12] [RPA11] [GPM09].

Compared to these existing works, main contributions and insights are summarized as following.

1. We model the physical subsystem at the granularity of differential equation level; and propose a control theory based benchmark framework to profile the 2L-CCPS cross-domain noise.

2. We propose a methodology to effectively shrink the benchmark sampling region, exploiting control domain-specific knowledge of Lyapunov stability.

3. We carry out case study on a representative control testbed. Our profiling framework effectively helps identify the most profitable cyber module to upgrade and our benchmark sampling region shrinking technology effectively reduces the profiling computation.

Chapter Organization

The rest of this chapter is organized as follows. Section 3.2 describes the overall

systems model to set the context for discussion. Section 3.3 elaborates our cross-domain noise profiling framework. Section 3.4 introduces a methodology to shrink the benchmark sampling region for the profiling framework. Section 3.5 carries out a case study to demonstrate how to use the framework, and the results are validated by experiment. Section 3.6 discusses related work. Section 3.7 concludes this chapter.

3.2 Overall Systems Models

We shall first set the context for our discussion by introducing the overall systems model. This includes the physical and cyber aspects of the 2L-CCPS architecture, and the combined systems model.

3.2.1 Physical Subsystem Model

In this chapter, we assume the physical subsystem of a 2L-CCPS is a *Linear Time Invariant* (LTI) control system, which is probably the most widely used control system.

For an LTI control system, the state of the plant at time t is described by a n-dimensional vector $X(t) = (x_1(t), x_2(t), \ldots, x_n(t))^{\mathsf{T}}$. The vector is also called the plant's *state vector* (in the following, we use the term "plant's state" and " plant's state vector" interchangeably), and each element of the state vector is also called a *state variable*. For simplicity of notations, we often omit the parameter t when writing state vector/variables, and use $\dot{X}$ (and respectively $\dot{x_i}$, $i = 1 \sim n$) to denote the derivative $\frac{\mathrm{d}\,X}{\mathrm{d}\,t}$ (and respectively $\frac{\mathrm{d}\,x_i}{\mathrm{d}\,t}$, $i = 1 \sim n$).

The dynamics of the plant is governed by the following systems of differential

equations.

$$\frac{\mathrm{d}(X - O_{\mathsf{ref}})}{\mathrm{d}\,t} = \mathbf{A}(X - O_{\mathsf{ref}}) + \mathbf{B}U, \tag{3.1}$$

$$U = -\mathbf{K}(X - O_{\mathsf{ref}}), \tag{3.2}$$

where $O_{\mathsf{ref}} \in \mathbb{R}_{n\times 1}$ is the reference point value from the cyber subsystem: the objective of control is to maneuver the plant state vector X to O_{ref} (so that $X - O_{\mathsf{ref}} = 0$); $\mathbf{A} \in \mathbb{R}_{n\times n}$ and $\mathbf{B} \in \mathbb{R}_{n\times m}$ are two constant matrices dependent on the plant's physics; $U(t) = (u_1(t), u_2(t), \ldots, u_m(t))^{\mathsf{T}}$ is the controller output created as per Eq. (3.2); $\mathbf{K} \in \mathbb{R}_{m\times n}$ is a constant matrix defining the control strategy.

Denote $\tilde{X} \stackrel{\mathsf{def}}{=} X - O_{\mathsf{ref}}$, the system of Eq. (3.1)(3.2) can be rewritten into the following form.

$$\dot{\tilde{X}} = \mathbf{F}\tilde{X}, \tag{3.3}$$

where $\mathbf{F} = \mathbf{A} - \mathbf{B}\mathbf{K}$.

Besides the above systems of differential equations, the dynamics of the plant are also governed by *allowed region* (or equivalently, the *forbidden region*, i.e. the complement of allowed region) in the state space. Every time X exceeds the allowed region (i.e. reaches the forbidden region), a *plant fault* happens. For example, for drone swarm control CPS, any two drones must maintain a distance of over 500 meters. Dropping below this 500 meters limit means a plant fault happens.

We use A to denote the allowed region and $\bar{A}$ to denote the forbidden region in the plant's state space.

3.2.2 Cyber Subsystem Model

We assume the cyber subsystem of a 2L-CCPS is a gray box: the subsystem's module level information, such as the constituent modules, their interfaces, and interconnections, is visible; though the internals of individual modules are invisible (black box can be regarded as a special case of gray box, where there is only one constituent module).

We assume the gray box cyber subsystem has a single input port and a single output port (see Fig. 3.2). The input port sends the current state of the plant X into the "Remote Sensing" module (denoted as M_{rs}, see Fig. 3.2); and the output port sends the decision from the "Final Decision" module (denoted as M_{fd}, see Fig. 3.2) as the new reference point value O'_{ref} to the physical subsystem. The remote sensing module M_{rs} senses the state of the physical plant, and outputs $M_{\mathsf{rs}}(X) + N$ to the rest of the cyber subsystem, where $M_{\mathsf{rs}}(X)$ is the sensing result without noise, and N is the cross-domain noise *random variable* (RV). The cross-domain noise RV N hence will propagate throughout the gray box cyber subsystem to interfere the final decision making.

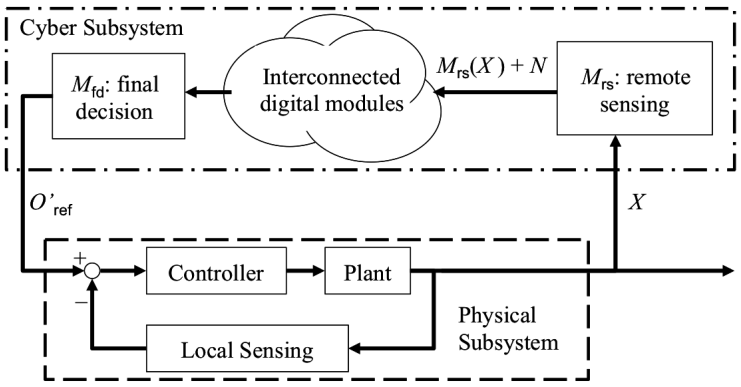


Figure 3.2. 2L-CCPS gray box cyber subsystem model

3.2.3 Combined Model

The hybrid automaton [Tab09] [A+93] of Fig. 3.3, denoted as H, models the combined "cyber" and "physical" aspects of 2L-CCPS.

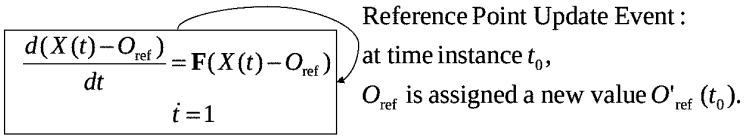


Figure 3.3. Hybrid automaton H that models 2L-CCPS

The continuous behavior of the combined model is described by H's node, which includes Eq. (3.3) and the continuous increase of time: $\dot{t} = 1$. We assume the plant's state vector X is continuous along time axis. This is reflected in H by the fact that the reference point update event cannot update X's value.

The discrete behavior of the combined model is described by H's edge. The edge represents a reference point update event: at any time t_0, the cyber subsystem can change the value of reference point O_{ref}, i.e. the cyber subsystem's output into the physical subsystem. After a reference point update event, O_{ref} takes a new value (denoted as $O'_{\text{ref}}(t_0)$ in Fig. 3.3) and remains constant until the next reference point update event. Note to comply with reality analysis, we assume the triggering of reference point update events is non-zeno.

3.3 Cross-Domain Noise Profiling Framework

As mentioned in Section 3.1, noises in the physical subsystem, such as local sensing noises, controller output disturbances, and plant modeling errors, are well studied

and can be well contained within the physical subsystem. Therefore these noises are out of the scope of this work. Instead, we focus on cross-domain noises (i.e. the noise denoted by RV N in Fig. 3.2), which are not contained within the physical subsystem. *Correspondingly, in the following, unless explicitly denoted, we use the term "noise" and "cross-domain noise" interchangeably.*

Our goal is to propose a framework of methods to profile cross-domain noises in the classic 2L-CCPS (see Fig. 3.1).

3.3.1 Elementary Trial and Reachability Probability

The *physical* subsystem of 2L-CCPS is modeled by Eq. (3.3); hence is memoryless. That is, the future trajectory of the plant $X(t)$ ($t \in (t_0, +\infty)$, where t_0 is the current time) is only dependent on the current state $X(t_0)$ and the current and future reference point values $O_{\mathsf{ref}}(t)$ ($t \in [t_0, +\infty)$). In practice, the derivative on the left hand side of Eq. (3.3) is finite, therefore, we can also say the future trajectory of $X(t)$ ($t \in (t_0, +\infty)$) is only dependent on the current state $X(t_0)$ and future reference point values $O_{\mathsf{ref}}(t)$ ($t \in (t_0, +\infty)$).

We assume the 2L-CCPS *cyber* subsystem is also memoryless. Suppose current time is t_0, and current plant state $X(t_0)$ is given: $X(t_0) = X^0$. We carry out the following *elementary trial*. At t_0, the cyber subsystem samples the current plant state and triggers a reference point update event immediately (see Fig. 3.3), changing O_{ref} to $O'_{\mathsf{ref}}(t_0)$. After that, the cyber subsystem triggers no more reference point update event.

In the elementary trial, the sampling, and hence the cyber subsystem's decision making, are interfered by the cross-domain noise RV N (see Fig. 3.2). Therefore, whether or not a plant fault will happen (i.e. $X(t)$ reaches forbidden region $\bar{A}$ during

$(t_0, +\infty)$) becomes random, and can be represented by a Bernoulli RV of $R(N, X^0)$: $R(N, X^0) = 1$ represents a plant fault will happen; and $R(N, X^0) = 0$ otherwise. We call $R(N, X^0)$ the *reachability RV* under cross-domain noise RV N and given X^0, and denote the *reachability probability* $\mathsf{Pr}(R(N, X^0) = 1)$ as $p(N, X^0)$; and consequently $\mathsf{Pr}(R(N, X^0) = 0) = 1 - p(N, X^0)$. Intuitively, $p(N, X^0)$ reflects the risk of the 2L-CCPS under cross-domain noise RV N and given X^0 (interested readers can refer to Appendix 5.11 to further understand this intuition).

In the following, we may simplify $R(N, X^0)$ as R and $p(N, X^0)$ as p when the context is unambiguous.

3.3.2 Measuring Reachability Probability

Next, we describe how to measure the value of $p(N, X^0)$. Under cross-domain noise RV N and given $X(t_0) = X^0$, we run a campaign of η elementary trials. The value of $p(N, X^0)$ can be estimated by averaging the results of these elementary trials. In practice, the elementary trials are emulated; and we further propose a control stability theory based method to accelerate these emulations.

Specifically, denote the reachability RV for the jth ($j = 1, \ldots, \eta$) elementary trial as R_j. Denote $\bar{R} \stackrel{\mathsf{def}}{=} \frac{1}{\eta} \sum_{j=1}^{\eta} R_j$. According to the well-known central limit theorem, when η is big enough, we can use $\bar{R}$ to estimate $p(N, X^0)$. This is quantitatively elaborated by the following proposition.

PROPOSITION 3.1 (CAMPAIGN SCALE) *Under cross-domain noise RV N, given $X(t_0) = X^0$, $\alpha \in [0,1]$, and $\delta_p \in (0, +\infty)$, if*

$$\eta \geqslant \left(\frac{\Phi^{-1}(1 - \frac{\alpha}{2})}{2\delta_p} \right)^2, \tag{3.4}$$

where Φ is the cumulative distribution function of standard normal distribution and Φ^{-1} is Φ's inverse; then $\bar{R}$ falls within range $p \pm \delta_p$ with confidence level of $(1-\alpha)$. That is, $\Pr(|\bar{R} - p| \leqslant \delta_p) \geqslant 1 - \alpha$.

Proof: Due to the memoryless assumption of the cyber and physical subsystems, R_js are identical independent distribution RVs, and $R_j \sim$ Bernoulli(p). According to central limit theorem, RV $\bar{R}$ therefore conforms to normal distribution Normal$(\mu, \sigma^2/\eta)$, where μ and σ^2 are respectively the expectation and variance of R_j. As $R_j \sim$ Bernoulli(p), $\mu = p$ and $\sigma^2 = p(1-p) \leqslant \frac{1}{4}$ (because $p \in [0,1]$), i.e. $\sigma \leqslant \frac{1}{2}$.

$$\begin{aligned} \text{Also Ineq. (3.4)} \quad &\Rightarrow \quad \sqrt{\eta} \geqslant \frac{\Phi^{-1}(1 - \frac{\alpha}{2})}{2\delta_p} \\ &\Rightarrow \quad \delta_p \geqslant \frac{\Phi^{-1}(1 - \frac{\alpha}{2})}{2\sqrt{\eta}}. \end{aligned} \tag{3.5}$$

Therefore, $\bar{R} \sim \text{Normal}(\mu, \sigma^2/\eta)$

$\Rightarrow \quad \mathsf{Pr}(|\bar{R} - \mu| \leqslant \frac{\sigma}{\sqrt{\eta}}\Phi^{-1}(1 - \frac{\alpha}{2})) \geqslant 1 - \alpha$

$\Rightarrow \quad \mathsf{Pr}(|\bar{R} - p| \leqslant \frac{1}{2\sqrt{\eta}}\Phi^{-1}(1 - \frac{\alpha}{2})) \geqslant 1 - \alpha$

(as $\mu = p$ and $\sigma \leqslant \frac{1}{2}$)

$\Rightarrow \quad \mathsf{Pr}(|\bar{R} - p| \leqslant \delta_p) \geqslant 1 - \alpha$ (due to Ineq. (3.5)). ■

Proposition 3.1 basically says, under cross-domain noise RV N, given $X(t_0) = X^0$, α, and δ_p, after a measurement campaign of η (η satisfies Ineq. (3.4)) elementary trials, we derive an realization $\bar{r}$ of RV $\bar{R}$, *which can be used as an estimation of p, i.e.* $\hat{p} = \bar{r}$, *with confidence level of at least* $(1 - \alpha)$.

As $\bar{R}$'s realization, we have $\bar{r} = \frac{1}{\eta}\sum_{j=1}^{\eta} r_j$, where r_j is RV R_j's realization in the corresponding elementary trail. To get r_j, the brute force way is to emulate the jth elementary trial. This includes two steps.

In step 1, feed the initial plant state X^0 into the real cyber subsystem and derive O'_{ref}. In step 2, simulate the physical subsystem of Eq. (3.3), from simulator time t_0 to simulator time $+\infty$, with initial plant state X^0, and updated reference point value O'_{ref}. If the resulted trajectory $X(t)$ ($t \in [t_0, +\infty)$) reaches forbidden region $\bar{A}$, then $r_j = 1$; otherwise $r_j = 0$.

In practice, infinite time simulation is impossible. Therefore the above brute force way has to be accelerated. This is possible when the physical subsystem (described by Eq. (3.3)) is an LTI control system.

In control engineering, it is a well established practice that LTI control systems in the form of Eq. (3.3) are designed to be stable in the sense of Lyapunov [Bro91]. Specifically, $\mathbf{K}$ of Eq. (3.2) is designed such that a positive definite symmetric matrix $\mathbf{P} \in \mathbb{R}_{n\times n}$ exists to satisfy

$$\mathbf{F}^{\mathsf{T}}\mathbf{P} + \mathbf{P}\mathbf{F} = -\mathbf{I}, \tag{3.6}$$

where $\mathbf{I}$ is the $n \times n$ identity matrix.

Conversely, given control systems of Eq. (3.3) that is stable in the sense of Lyapunov, there are mature tools [Bro91] to derive the aforementioned $\mathbf{P}$.

With $\mathbf{P}$, we can define a *Lyapunov function* $V(X(t), O_{\mathsf{ref}}(t))$ as follows.

$$V(X(t), O_{\mathsf{ref}}(t)) \stackrel{\mathsf{def}}{=} (X(t) - O_{\mathsf{ref}}(t))^{\mathsf{T}}\mathbf{P}(X(t) - O_{\mathsf{ref}}(t)). \tag{3.7}$$

Intuitively, Lyapunov function represents a virtual "potential energy" of the physical plant. If the physical subsystem is stable, this potential energy should monotonically decrease. This is quantified by the following proposition.

PROPOSITION 3.2 (TRAJECTORY BOUNDARY) *Given* $X(t_0) = X^0 \in \mathbb{R}_{n\times 1}$ *and* $O'_{\mathsf{ref}}(t_0) \in \mathbb{R}_{n\times 1}$, *let* $X(t)$ $(t \in [t_0, +\infty))$ *be the trajectory of plant state evolved according to Eq.* (3.3) *when* $O_{\mathsf{ref}}(t) \equiv O'_{\mathsf{ref}}(t_0)$, *then* $\forall t \in [t_0, +\infty)$,

$$\frac{d\,V(X(t), O_{\mathsf{ref}}(t))}{d\,t} \leqslant 0. \tag{3.8}$$

Proof: Proposition 3.2 is already implied in the classic proof of Lyapunov stability [Kha01]. The details are recompiled in Appendix 5.10. ■

Due to Proposition 3.2, in an elementary trial, the plant's Lyapunov function value monotonically drops. Particularly, if it drops below the minimum Lyapunov function value of the forbidden region $\bar{A}$, the plant state can never reach $\bar{A}$ again. Based on this heuristics, we propose the algorithm of Fig. 3.4 to emulate the jth elementary trial ($j = 1, \ldots, \eta$), so as to approximate r_j, the realization of reachability RV R_j.

```
1.  ElementaryTrialEmulation(input: N, X^0; output: r_j){
2.      Input X(t_0) = X^0 into the cyber subsystem to generate O'_ref(t_0);
        // or equivalently, let M_rs output M_rs(X(t_0)) + N to the rest
        // of the cyber subsystem to generate O'_ref(t_0), where X(t_0) = X^0.
3.      Current simulator time t ← t_0;
4.      O_ref ← O'_ref(t_0);
5.      while (true){
6.          Derive X(t) according to Eq. (3.3);
7.          if (X(t) ∈ Ā) { r_j ← 1; break; }
8.          if (V(X(t), O_ref) < inf_{χ∈Ā}{V(χ, O_ref)}) { r_j ← 0; break; }
9.          t ← t + δ_t; // δ_t: per iteration simulator time increment
10.         if (t ⩾ T_sim){ // T_sim: maximum simulation time
11.             r_j ← 1; break;
12.         }
13.     }
14. }
```

Figure 3.4. Pseudo C code to emulate an elementary trial, to calculate r_j. It is an emulation because Line 2 uses the real cyber subsystem.

In Fig. 3.4, Line 7 corresponds to the case that trajectory $X(t)$ is found to reach forbidden region $\bar{A}$, hence $r_j = 1$. In Line 8, as future trajectory $X(t)$'s Lyapunov function value drops below $\inf_{\chi \in \bar{A}}\{V(\chi, O_{\text{ref}})\}$, simple proof with negation can show that due to Ineq. (3.8), $X(t)$ will never reach any points in $\bar{A}$. Line 11 corresponds to the situation that after sufficiently long simulation, we still cannot decide whether $X(t)$ reaches $\bar{A}$; therefore, we pessimistically over approximate with $r_j = 1$.

3.3.3 Quantifying Impact of Cross-Domain Noise with Reachability Probability

Now we can get the η realizations $\{r_j\}$. Let $\hat{p} \stackrel{\text{def}}{=} \bar{r} \stackrel{\text{def}}{=} \frac{1}{\eta}\sum_{j=1}^{\eta} r_j$, as per Proposition 3.1, when η satisfies Ineq. (3.4), $\hat{p} = \bar{r}$ is a $(1-\alpha)$ confident estimation of p. By definition, p is an elementary trial's reachability probability (i.e. probability to reach forbidden region $\bar{A}$) under cross-domain noise RV N and given initial plant state X^0. That is, p's elaborative form is $p(N, X^0)$, and it measures the *risk* of an elementary trial.

The *impact* of cross-domain noise RV N should be the *risk increase* caused by N. Let $I(N, X^0)$ denote the impact of N on the 2L-CCPS with initial plant state $X(t_0) = X^0$. Then we propose to quantify $I(N, X^0)$ as

$$I(N, X^0) \stackrel{\text{def}}{=} p(N, X^0) - p(0, X^0), \tag{3.9}$$

where $p(0, X^0)$ is an elementary trial's reachability probability under 0 cross-domain noise and given initial plant state X^0.

To holistically quantify the impact of N to the 2L-CCPS, ideally, we should evaluate $I(N, X^0)$ for every $X^0 \in \mathbb{R}_{n\times 1}$. Obviously this is impractical. Instead, we propose to use a benchmark $\mathcal{B} = \{X_i^0\}_{i=1,\ldots,b}$ of b sample points in the allowed region A (i.e. $\forall i$, $X_i^0 \in A$). The b sample points in $\mathcal{B}$ are fixed, or the sampling method is fixed (e.g. uniform sampling in A). We call each sampled point X_i^0 a *benchmark point*.

With benchmark $\mathcal{B} = \{X_i^0\}_{i=1,\ldots,b}$, we summarize our 2L-CCPS cross-domain noise profiling framework as follows. Given cross-domain noise RV N, for each benchmark point X_i^0, we run the elementary trial campaign described in Section 3.3.1 and 3.3.2 to get reachability probability $p_i(N, X_i^0)$ and $p_i(0, X_i^0)$, and follow Eq. (3.9) to

get cross-domain noise impact $I_i(N, X_i^0)$. The profile of cross-domain noise RV N is thus the set $\{I_i(N, X_i^0)\}_{i=1,\ldots,b}$.

3.4 Shrinking Benchmark Region

3.4.1 Refined 2L-CCPS Architecure

In Section 3.3, the benchmark points are sampled from the entire allowed region A. This benchmark sampling region (simplified as "*benchmark region*" in the following) is too big. On the other hand, for an initial plant state $X^0 \in A$ sufficiently away from the forbidden region $\bar{A}$, the plant trajectory may never reach $\bar{A}$, even perturbed by large cross-domain noise. It is therefore meaningless to include such X^0 in the benchmark. To make an analogy, to benchmark meteoroids' reachability to the earth, it is sufficient to focus on meteoroids in the solar system; meteoroids in other galaxies are practically irrelevant.

Based on the above heuristics, we propose to shrink the benchmark region as follows.

We refine the classic 2L-CCPS architecture of Fig. 3.1 by adding a *bounding filter* to the input port of the physical subsystem (see Fig. 3.5). This bounding filter rejects extreme new reference point values from the cyber subsystem. Specifically, suppose at time t_0 an reference point update event happened, and $X(t_0) = X^0$. Then the bounding filter will define a hyper *bounding ball* $\mathsf{Ball}(X^0, \gamma)$ in the state space, centered at X^0 with radius $\gamma > 0$. If the new reference point value O'_{ref} from the cyber subsystem is within $\mathsf{Ball}(X^0, \gamma)$, then O'_{ref} is accepted. Otherwise, O'_{ref} is truncated. Formally, the filtered new reference point value O''_{ref} is

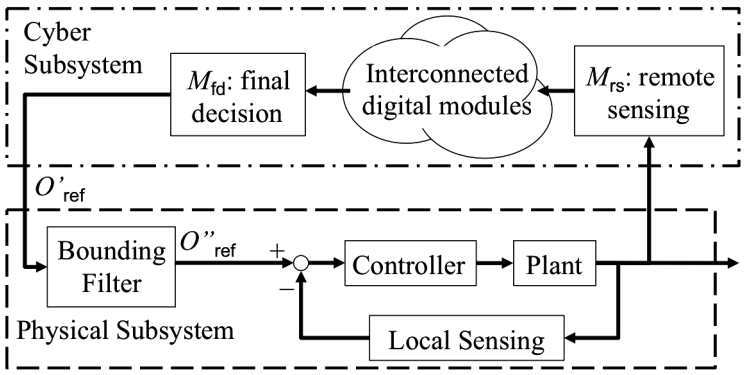


Figure 3.5. Refined 2L-CCPS architecture

$$O''_{\mathsf{ref}} = \begin{cases} \frac{O'_{\mathsf{ref}} - X^0}{||O'_{\mathsf{ref}} - X^0||_2}\gamma + X^0 & (\text{if } ||O'_{\mathsf{ref}} - X^0||_2 \geqslant \gamma) \\ O'_{\mathsf{ref}} & (\text{otherwise}) \end{cases} \tag{3.10}$$

Note Eq. (3.10) implies that the classic 2L-CCPS architecture (see Fig. 3.1) is a special case of the refined 2L-CCPS architecture (see Fig. 3.5), where $\gamma = +\infty$.

With the bounding filter, no matter what the cross-domain noise RV N is, given the current plant state X^0, any reference point update event can only change reference point to a value within $\mathsf{Ball}(X^0, \gamma)$.

Therefore, under whatever RV N, for an elementary trial starting from plant state X^0, the reachable state space of all possible future trajectories is constrained. Denote this reachable state space as $\mathsf{Traj}(N, X^0)$. Denote

$$\begin{aligned} \bar{B}^* \quad \stackrel{\mathsf{def}}{=} \quad & \{X^0 | X^0 \in A, \text{ and } \mathsf{Traj}(N, X^0) \cap \bar{A} \equiv \varnothing \\ & \text{for whatever RV } N \}. \end{aligned}$$

Then for whatever RV N, $\forall X^0 \in \bar{B}^*$, $p(N, X^0) \equiv 0$ and $I(N, X^0) \equiv 0$. Therefore, if we can explicitly identify $\bar{B}^*$, then we do not need to benchmark test any point in $\bar{B}^*$. A point in $\bar{B}^*$ is thus an "*irrelevant benchmark point*".

Correspondingly, the (relevant) benchmark points only need to be sampled from $B^* \stackrel{\text{def}}{=} A - \bar{B}^*$. More specifically, we call B^* the "*tight shrunk benchmark region*", and call any $B \supseteq B^*$ ($B \subseteq A$) a "*shrunk benchmark region*". Correspondingly, we call $\bar{B}^*$ the "*tight irrelevant benchmark region*", and call any $\bar{B} \subseteq \bar{B}^*$ an "*irrelevant benchmark region*".

3.4.2 Heuristics to Shrink Benchmark Region

Now, the question is how to find B, or equivalently $\bar{B}$, given the bounding filter (see Fig. 3.5).

Our solution heuristics is still based on Proposition 3.2. Basically, for a well designed LTI physical subsystem, the plant's Lyapunov function $V(X(t), O_{\mathsf{ref}}(t))$ exists, and is monotonically decreasing when $O_{\mathsf{ref}}(t)$ is a constant, which is the case for elementary trials. According to Proposition 3.2, at time t_0, given initial plant state $X(t_0) = X^0 \in A$ and bounding filtered new reference point value $O''_{\mathsf{ref}}(t_0) \in \mathsf{Ball}(X^0, \gamma)$, the trajectory of an elementary trial $X(t)$ ($t \in [t_0, +\infty)$) is confined by the hyper-ellipsoid $E(X^0, O''_{\mathsf{ref}}(t_0))$ of

$$\begin{aligned} E(X^0, O''_{\mathsf{ref}}(t_0)) \stackrel{\text{def}}{=} & \{Y | Y \in \mathbb{R}_{n\times 1} \text{ and} \\ & (Y - O''_{\mathsf{ref}}(t_0))^{\mathsf{T}} \mathbf{P} (Y - O''_{\mathsf{ref}}(t_0)) \\ & \leqslant V(X^0, O''_{\mathsf{ref}})\}, \end{aligned} \tag{3.11}$$

where $\mathbf{P}$ is the positive definite symmetric matrix in the Lyapunov function of Eq. (3.7). We call $E(X^0, O''_{\mathsf{ref}}(t_0))$ a "*Lyapunov hyper-ellipsoid*".

As shown by Fig. 3.6, *if none of such confining Lyapunov hyper-ellipsoids intersects with $\bar{A}$, then $X^0 \in \bar{B}^*$.* Consequently, the set of such X^0s constitute a $\bar{B} \subseteq \bar{B}^*$.

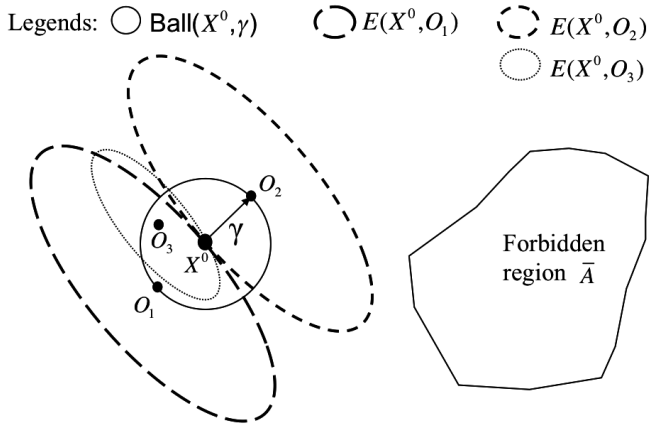


Figure 3.6. Confining Lyapunov hyper-ellipsoids and forbidden region

Formally, let us define

$$V^{\mathsf{sup}}_{X^0,\mathsf{Ball}(X^0,\gamma)} \stackrel{\mathsf{def}}{=} \sup_{\forall O''_{\mathsf{ref}} \in \mathsf{Ball}(X^0,\gamma)} \{V(X^0, O''_{\mathsf{ref}})\}, \tag{3.12}$$

and for arbitrary $S \subseteq \mathbb{R}_{n\times 1}$, define

$$V^{\mathsf{inf}}_{S,\mathsf{Ball}(X^0,\gamma)} \stackrel{\mathsf{def}}{=} \inf_{\forall O''_{\mathsf{ref}} \in \mathsf{Ball}(X^0,\gamma)} \{V(Y, O''_{\mathsf{ref}}) | \forall Y \in S\}.$$

Then the intuition of Fig. 3.6 is formalized by Lemma 3.1.

LEMMA 3.1 (IRRELEVANT BENCHMARK POINT) *For any state* $X^0 \in A$, *if* $V^{sup}_{X^0,Ball(X^0,\gamma)} < V^{inf}_{\bar{A},Ball(X^0,\gamma)}$, *then* $X^0 \in \bar{B}^*$.

Proof: For any elementary trial starting with $X(t_0) = X^0$, no matter what RV N is, the resulted new reference point after bounding filtering, denoted as $O''_{\mathsf{ref}}(t_0)$, is within $\mathsf{Ball}(X^0, \gamma)$. If $V^{\mathsf{inf}}_{\bar{A},\mathsf{Ball}(X^0,\gamma)} > V^{\mathsf{sup}}_{X^0,\mathsf{Ball}(X^0,\gamma)}$, then the elementary trial plant state trajectory's initial Lyapunov function value $V(X(t_0), O''_{\mathsf{ref}}(t_0))$ is less than that of any state in $\bar{A}$. As per Proposition 3.2, the elementary trial plant state trajectory can never reach $\bar{A}$. This is true for any elementary trial starting with $X(t_0) = X^0$ under whatever RV N. Therefore $\mathsf{Traj}(N, X^0) \cap \bar{A} \equiv \varnothing$ for whatever RV N. ■

3.4.3 Closed-Form Definition of Shrunk Benchmark Region

This subsection shall extend Lemma 3.1 to find a closed-form $\bar{B}$, hence B.

Our heuristics is to first find the closed-form formula for $V^{\mathsf{sup}}_{X^0,\mathsf{Ball}(X^0,\gamma)}$. Using this formula, we then find a sufficient condition for $V^{\mathsf{sup}}_{X^0,\mathsf{Ball}(X^0,\gamma)} < V^{\mathsf{inf}}_{\bar{A},\mathsf{Ball}(X^0,\gamma)}$. Then any X^0 satisfying the sufficient condition should belong to $\bar{B}^*$. Consequently, the set of such X^0s constitute a $\bar{B} \subseteq \bar{B}^*$.

Fig. 3.7 gives the intuition to find the closed-form formula to calculate $V^{\mathsf{sup}}_{X^0,\mathsf{Ball}(X^0,\gamma)}$. Given X^0 and $\forall O''_{\mathsf{ref}} \in \mathsf{Ball}(X^0, \gamma)$, the maximum Lyapunov function value $V(X^0, O''_{\mathsf{ref}})$ is achieved when we choose $O''_{\mathsf{ref}} = O_1$, so that the radius of $\mathsf{Ball}(X^0, \gamma)$ exactly overlaps with the semi-minor axis of Lyapunov hyper-ellipsoid $E(X^0, O''_{\mathsf{ref}})$ (see Eq. (3.11)). Note the directions and lengths ratio of the major/minor axes of all Lyapunov hyper-ellipsoids are fixed once $\mathbf{P}$ is given; and $E(X^0, O''_{\mathsf{ref}})$ is centered on O''_{ref} and with X^0 on surface.

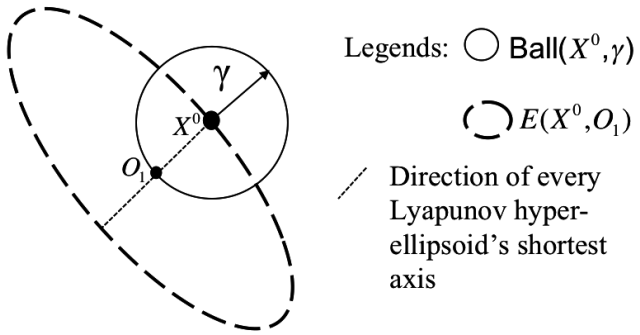


Figure 3.7. Intuition of $V^{\mathsf{sup}}_{X^0,\mathsf{Ball}(X^0,\gamma)}$

Fig. 3.7's intuition to find the closed-form formula of $V^{\mathsf{sup}}_{X^0,\mathsf{Ball}(X^0,\gamma)}$ is formalized by Lemma 3.2.

LEMMA 3.2 (CLOSED-FORM VALUE OF $V^{\mathsf{SUP}}_{X^0,\mathsf{BALL}(X^0,\gamma)}$) *We have* $V^{\mathsf{sup}}_{X^0,\mathsf{Ball}(X^0,\gamma)} = \lambda^{\mathsf{max}}(\mathbf{P})\gamma^2$, *where* $\lambda^{\mathsf{max}}(\mathbf{P})$ *is the maximal eigenvalue of* $\mathbf{P}$ *in Lyapunov function of Eq.* (3.7).

Proof: According to definition (see Eq. (3.12)), $V^{\mathsf{sup}}_{X^0,\mathsf{Ball}(X^0,\gamma)}$ is the optimal objective function value for the following optimization problem:

$$\begin{aligned} \max_{O''_{\mathsf{ref}}} \quad & f_{X^0}(O''_{\mathsf{ref}}) = V(X^0, O''_{\mathsf{ref}}) \\ & = (X^0 - O''_{\mathsf{ref}})^{\mathsf{T}}\mathbf{P}(X^0 - O''_{\mathsf{ref}}) \\ \text{s.t.} \quad & (X^0 - O''_{\mathsf{ref}})^{\mathsf{T}}(X^0 - O''_{\mathsf{ref}}) \leqslant \gamma^2, \end{aligned} \tag{3.13}$$

where O''_{ref} is the only optimization variable.

Problem (3.13) is a typical Quadratic Constrained Quadratic Optimization (QCQP) problem [B+04]. As this problem has a single constraint and the constraint itself is a hyper ball, a special form of quadratic function, we can solve it as follows.

First, denote $\tilde{O}_{\mathsf{ref}} \stackrel{\mathsf{def}}{=} X^0 - O''_{\mathsf{ref}}$, and $f'_{X^0}(\tilde{O}_{\mathsf{ref}}) \stackrel{\mathsf{def}}{=} -f_{X^0}(O''_{\mathsf{ref}}) = -\tilde{O}_{\mathsf{ref}}^{\mathsf{T}}\mathbf{P}\tilde{O}_{\mathsf{ref}}$. Then problem (3.13) is equivalent to problem

$$\begin{array}{ll} \min\limits_{\tilde{O}_{\mathsf{ref}}} & f'_{X^0}(\tilde{O}_{\mathsf{ref}}) \\ \text{s.t.} & \tilde{O}_{\mathsf{ref}}^{\mathsf{T}}\tilde{O}_{\mathsf{ref}} \leqslant \gamma^2. \end{array} \tag{3.14}$$

The Lagrangian of optimization problem (3.14) is

$$L(\tilde{O}_{\mathsf{ref}}, \nu) = \tilde{O}_{\mathsf{ref}}^{\mathsf{T}}(\nu\mathbf{I} - \mathbf{P})\tilde{O}_{\mathsf{ref}} - \nu\gamma^2,$$

and the dual function is

$$\begin{array}{rcl} g(\nu) & = & \inf\limits_{\tilde{O}_{\mathsf{ref}}}\{L(\tilde{O}_{\mathsf{ref}}, \nu)\} \\ & = & \begin{cases} -\nu\gamma^2 & (\text{if } \nu\mathbf{I} - \mathbf{P} \succeq 0) \\ -\infty & (\text{otherwise}) \end{cases} \end{array}$$

where "$\succeq 0$" means the matrix on the left hand side is positive semidefinite. Using a Schur complement [B$^+$04], the Lagrange dual problem to problem (3.14) is

$$\begin{array}{ll} \max\limits_{\nu} & h \\ \text{s.t.} & \nu \geqslant 0 \\ & \begin{bmatrix} \nu\mathbf{I} - \mathbf{P} & 0 \\ 0 & -\nu\gamma^2 - h \end{bmatrix} \succeq 0 \end{array} \tag{3.15}$$

As problem (3.14) is strictly feasible, i.e. there exists some $\tilde{O}_{\mathsf{ref}}$ (e.g. $\tilde{O}_{\mathsf{ref}} = 0$) s.t. $\tilde{O}_{\mathsf{ref}}^{\mathsf{T}}\tilde{O}_{\mathsf{ref}} < \gamma^2$, problem (3.15) holds strong duality to problem (3.14) [B$^+$04]. Hence,

the two problems' optimal values are equal. By solving problem (3.15), we have the optimal value

$$h^* = -\lambda^{\max}(\mathbf{P})\gamma^2,$$

where $\lambda^{\max}(\mathbf{P})$ is the maximal eigenvalue of matrix $\mathbf{P}$. Then we have

$$f_{X^0}(O''_{\text{ref}})^* = -f'_{X^0}(\tilde{O}_{\text{ref}})^* = -h^* = \lambda^{\max}(\mathbf{P})\gamma^2. \qquad \blacksquare$$

Now we know that given $X^0 \in A$, $V^{\sup}_{X^0,\text{Ball}(X^0,\gamma)} = \lambda^{\max}(\mathbf{P})\gamma^2$. Then, it is possible to find a sufficient condition to make $V^{\sup}_{X^0,\text{Ball}(X^0,\gamma)} < V^{\inf}_{\bar{A},\text{Ball}(X^0,\gamma)}$. To find such sufficient condition, let us first define the distance between a point $X^0 \in \mathbb{R}_{n\times 1}$ and a region $S \subseteq \mathbb{R}_{n\times 1}$ as

$$\text{Dis}(X, S) \stackrel{\text{def}}{=} \inf\{||X - Y||_2 | \forall Y \in S\}.$$

Then a sufficient condition is described by Lemma 3.3.

LEMMA 3.3 (IRRELEVANCE DISTANCE) *Given $S \subseteq \mathbb{R}_{n\times 1}$, state $X^0 \in A$, and an arbitrarily small positive constant $\varepsilon > 0$, if*

$$\textit{Dis}(X^0, S) > \sqrt{\frac{\lambda^{\max}(\mathbf{P})}{\lambda^{\min}(\mathbf{P})}}\gamma + \gamma + \varepsilon \stackrel{\text{def}}{=} \Gamma, \tag{3.16}$$

where $\lambda^{\max}(\mathbf{P})$ and $\lambda^{\min}(\mathbf{P})$ are respectively the maximum and minimum eigenvalues of the positive definite symmetric matrix $\mathbf{P}$ of Eq. (3.7)*, then $V^{\sup}_{X^0,\textit{Ball}(X^0,\gamma)} < V^{\inf}_{S,\textit{Ball}(X^0,\gamma)}$.*

Proof: $\forall O''_{\mathsf{ref}} \in \mathsf{Ball}(X^0, \gamma), \forall Y \in S$,

$$V(Y, O''_{\mathsf{ref}}) = (Y - O''_{\mathsf{ref}})^{\mathsf{T}} \mathbf{P} (Y - O''_{\mathsf{ref}}). \tag{3.17}$$

Due to the bounding filter, we know that

$$(O''_{\mathsf{ref}} - X^0)^{\mathsf{T}} (O''_{\mathsf{ref}} - X^0) \leqslant \gamma^2.$$

Also, as $\mathsf{Dis}(X^0, S) > \Gamma$, we have

$$(Y - X^0)^{\mathsf{T}} (Y - X^0) > \Gamma^2.$$

From Eq. (3.17), we get

$$\begin{aligned}
& V(Y, O''_{\mathsf{ref}}) \\
\geqslant\ & \lambda^{\mathsf{min}}(\mathbf{P}) (Y - O''_{\mathsf{ref}})^{\mathsf{T}} (Y - O''_{\mathsf{ref}}) \\
=\ & \lambda^{\mathsf{min}}(\mathbf{P}) \\
& [(Y - X^0) - (O''_{\mathsf{ref}} - X^0)]^{\mathsf{T}} [(Y - X^0) - (O''_{\mathsf{ref}} - X^0)] \\
>\ & \lambda^{\mathsf{min}}(\mathbf{P}) (\Gamma - \gamma)^2 \qquad \text{(see Lemma 5.4 in Appendix 5.9)} \\
>\ & \lambda^{\mathsf{max}}(\mathbf{P}) \gamma^2 + \lambda^{\mathsf{min}}(\mathbf{P}) \varepsilon^2 \\
=\ & V^{\mathsf{sup}}_{X^0, \mathsf{Ball}(X^0, \gamma)} + \lambda^{\mathsf{min}}(\mathbf{P}) \varepsilon^2.
\end{aligned}$$

That is, $\forall O''_{\mathsf{ref}} \in \mathsf{Ball}(X^0, \gamma)$, $\forall Y \in S$, we have $V(Y, O''_{\mathsf{ref}}) > V^{\mathsf{sup}}_{X^0, \mathsf{Ball}(X^0, \gamma)} + \lambda^{\mathsf{min}}(\mathbf{P}) \varepsilon^2$. Therefore, $V^{\mathsf{sup}}_{X^0, \mathsf{Ball}(X^0, \gamma)} < V^{\mathsf{inf}}_{S, \mathsf{Ball}(X^0, \gamma)}$. ■

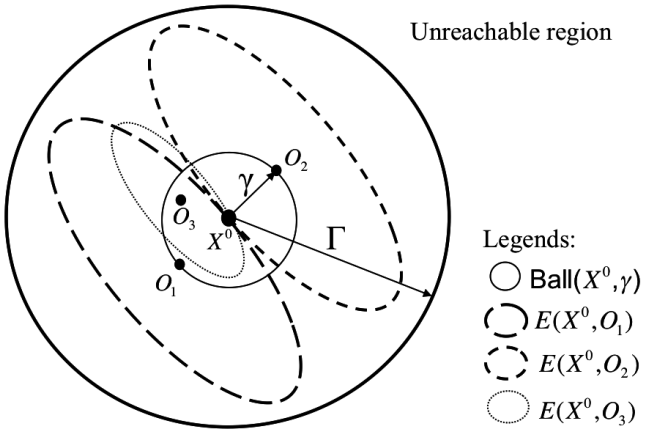


Figure 3.8. Visual intuition of irrelevance distance

Fig. 3.8 visualizes the intuition of irrelevance distance Γ. Basically, if $\mathsf{Dis}(X^0, S) > \Gamma$, then no Lyapunov hyper-ellipsoid $E(X^0, O''_{\mathsf{ref}})$ ($\forall O''_{\mathsf{ref}} \in \mathsf{Ball}(X^0, \gamma)$) can intersect with S. Hence elementary trial trajectories starting from X^0 can never reach S. In case $S = \bar{A}$ and $X^0 \in A$, X^0 thus is an irrelevant benchmark point: $X^0 \in \bar{B}^*$.

Lemma 3.3 thus helps us to find a closed-form shrunk benchmark region B, as described by Theorem 3.1.

THEOREM 3.1 (SHRUNK BENCHMARK REGION) *For the refined 2L-CCPS architecture,*

$$B \stackrel{\mathsf{def}}{=} \{X^0 | X^0 \in A, \textit{ and } \mathsf{Dis}(X^0, \bar{A}) \leqslant \Gamma\} \tag{3.18}$$

is a shrunk benchmark region.

Proof: $\forall X^0 \in \bar{B} = A - B$, $\mathsf{Dis}(X^0, \bar{A}) > \Gamma$. Due to Lemma 3.3, we know that $V^{\mathsf{sup}}_{X^0,\mathsf{Ball}(X^0,\gamma)} < V^{\mathsf{inf}}_{\bar{A},\mathsf{Ball}(X^0,\gamma)}$. Due to Lemma 3.1, we know $X^0 \in \bar{B}^*$. Therefore,

$\bar{B} \subseteq \bar{B}^*$. That is $B \supseteq B^*$. ■

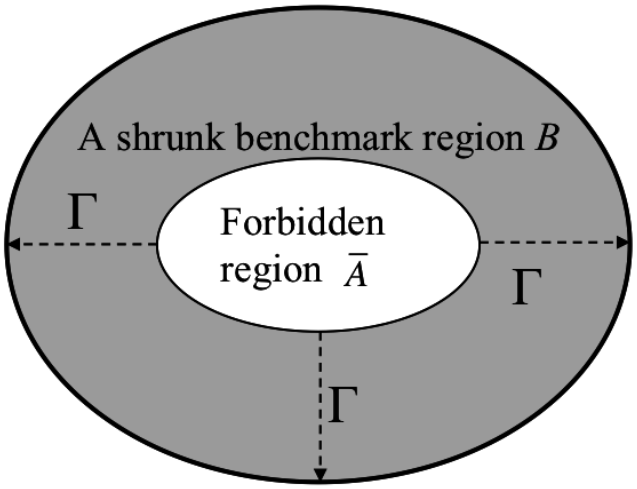


Figure 3.9. A shrunk benchmark region derived via Theorem 3.1

Fig. 3.9 illustrates an example of shrunk benchmark region derived via Theorem 3.1. Now, to build benchmark $\mathcal{B}$, instead of sampling the entire allowed region A, we only need to sample the shrunk benchmark region B.

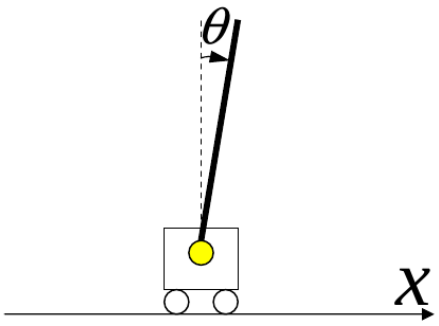
3.5 Case Study

In this section, we carry out a case study to demonstrate the usage and validity of our proposed framework in Section 3.3 and 3.4. Specifically, the case study uses the framework to profile cross-domain noise impacts for two software upgrade alternatives. By comparing the two profiles, a better alternative is chosen. Experiments are then carried out to verify the choice. We also show that Section 3.4's benchmark region shrinking method can save 24.1% of the profiling computation, meanwhile achieving the same profiling goal.

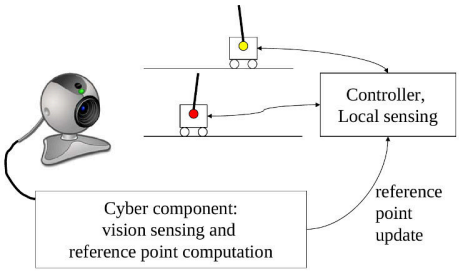
3.5.1 Testbed Setup

Our case study is about a 2L-CCPS testbed that runs vision based parallel inverted pendulums [Bro91] (see Fig. 3.10).

The physical subsystem of the testbed consists of two inverted pendulums: IP_1 and IP_2. An inverted pendulum is a metal rod with one end hinged on a cart, and the other end free to rotate around the hinge (see Fig. 3.10 (a)). The cart can move along a piece of metal rail. The controller of the inverted pendulum takes charge of moving the cart back and forth along the rail to keep the hinged metal rod (the inverted pendulum) standing upright.



(a) An inverted pendulum



(b) Holistic testbed view

Figure 3.10. Parallel-inverted-pendulum testbed

For IP_i ($i = 1, 2$), let $X_{ip_i}(t)$ denote its plant state. X_{ip_i} then includes four

state variables (see Fig. 3.10(a)): respectively the current location $x_{\mathsf{ip}i}(t)$ (m) and velocity $\dot{x}_{\mathsf{ip}i}(t)$ (m/sec) of the cart, and the current angular displacement $\theta_{\mathsf{ip}i}(t)$ (rad) and velocity $\dot{\theta}_{\mathsf{ip}i}(t)$ (rad/sec) of the rod from the upright position. That is, $X_{\mathsf{ip}i}(t) = (x_{\mathsf{ip}i}(t), \theta_{\mathsf{ip}i}(t), \dot{x}_{\mathsf{ip}i}(t), \dot{\theta}_{\mathsf{ip}i}(t))^{\mathsf{T}}$.

As an LTI control system[2], the physical dynamics of IP_i is governed by the following systems of differential equations [Ltd].

$$\frac{\mathrm{d}(X_{\mathsf{ip}i} - O_{\mathsf{ipref}i})}{\mathrm{d}\,t} = \mathbf{A}_{\mathsf{ip}i}(X_{\mathsf{ip}i} - O_{\mathsf{ipref}i}) + \mathbf{B}_{\mathsf{ip}i}U_{\mathsf{ip}i},$$
$$U_{\mathsf{ip}i} = -\mathbf{K}_{\mathsf{ip}i}(X_{\mathsf{ip}i} - O_{\mathsf{ipref}i}),$$

where $X_{\mathsf{ip}i}$, $O_{\mathsf{ipref}i}$, $U_{\mathsf{ip}i}$, $\mathbf{A}_{\mathsf{ip}i}$, $\mathbf{B}_{\mathsf{ip}i}$, $\mathbf{K}_{\mathsf{ip}i}$ respectively correspond to X, O_{ref}, U, $\mathbf{A}$, $\mathbf{B}$, $\mathbf{K}$ in Eq. (3.1) and (3.2). The specific inverted pendulums we use are made by Googol [Ltd], and have the following configurations (for both $i = 1$ and 2).

$$\mathbf{A}_{\mathsf{ip}i} = \begin{pmatrix} 0.000 & 1.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 1.000 \\ 0.000 & 0.000 & 29.400 & 0.000 \end{pmatrix},$$
$$\mathbf{B}_{\mathsf{ip}i} = (0.000, 1.000, 0.000, 3.000)^{\mathsf{T}},$$
$$\mathbf{K}_{\mathsf{ip}i} = (-5.0505, -5.8249, 35.2502, 6.2750).$$

As we have two inverted pendulums, the holistic plant of our testbed can be de-

[2]Strictly speaking, an inverted pendulum control system is not linear, but when $\theta_{\mathsf{ip}i}$ is reasonably small (e.g. $\leq \frac{\pi}{6}$ (rad)), the system can be regarded as linear.

scribed by the following differential equation systems.

$$\frac{\mathrm{d}(X_{\mathsf{tb}} - O_{\mathsf{tbref}})}{\mathrm{d}\,t} = \mathbf{A}_{\mathsf{tb}}(X_{\mathsf{tb}} - O_{\mathsf{tbref}}) + \mathbf{B}_{\mathsf{tb}}U_{\mathsf{tb}}, \tag{3.19}$$

$$U_{\mathsf{tb}} = -\mathbf{K}_{\mathsf{tb}}(X_{\mathsf{tb}} - O_{\mathsf{tbref}}), \tag{3.20}$$

where $X_{\mathsf{tb}} = \begin{pmatrix} X_{\mathsf{ip1}} \\ X_{\mathsf{ip2}} \end{pmatrix}$, $O_{\mathsf{tb}} = \begin{pmatrix} O_{\mathsf{ipref1}} \\ O_{\mathsf{ipref2}} \end{pmatrix}$, $\mathbf{A}_{\mathsf{tb}} = \begin{pmatrix} \mathbf{A}_{\mathsf{ip1}} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{\mathsf{ip2}} \end{pmatrix}$, $\mathbf{B}_{\mathsf{tb}} = \begin{pmatrix} \mathbf{B}_{\mathsf{ip1}} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_{\mathsf{ip2}} \end{pmatrix}$, and $\mathbf{K}_{\mathsf{tb}} = \begin{pmatrix} \mathbf{K}_{\mathsf{ip1}} & \mathbf{0} \\ \mathbf{0} & \mathbf{K}_{\mathsf{ip2}} \end{pmatrix}$.

As both IPs move along the x-axis, the allowed region A for our testbed is[3]

$$A = \{X_{\mathsf{tb}} | X_{\mathsf{tb}} \in \mathbb{R}_{8\times 1}, \text{ and } 0.15 \leq x_{\mathsf{ip2}} - x_{\mathsf{ip1}} \leq 0.2\}. \tag{3.21}$$

That is, IP_1 and IP_2's carts cannot go too close nor too apart[4].

The cyber subsystem of our testbed takes charge of computing new reference points for the plant (i.e. IP_1 and IP_2) using computer vision sensing inputs. Its gray box details are known and depicted in Fig. 3.11.

Note reference point represents the equilibrium state that we aim to achieve. For inverted pendulum IP_i ($i = 1, 2$), we always want the equilibrium taking the form $O_{\mathsf{ipref}i} = (x_{\mathsf{ipref}i}, 0, 0, 0)^{\mathsf{T}}$. That is, at equilibrium, the inverted pendulum cart should stop at $x_{\mathsf{ipref}i}$, and the rod should stand still at upright angle. Therefore, the only possible update we would make to reference point is the cart's equilibrium location $x_{\mathsf{ipref}i}$: at different time, we may want to move the cart to different locations. Therefore, the cyber subsystem of our testbed is focusing on computing new $x_{\mathsf{ipref}i}$s.

[3]Here we are assuming the rail of the IPs are long enough. Otherwise, a more strict definition of A should also include the rail length constraints.

[4]In the actual implementation, IP_1 and IP_2 are moving along two parallel rails. Therefore, the two inverted pendulums will not really crash. However, for evaluation purposes, we still enforce the allowed region of Ineq. (3.21), regarding IP_1 and IP_2 as if moving along a same rail.

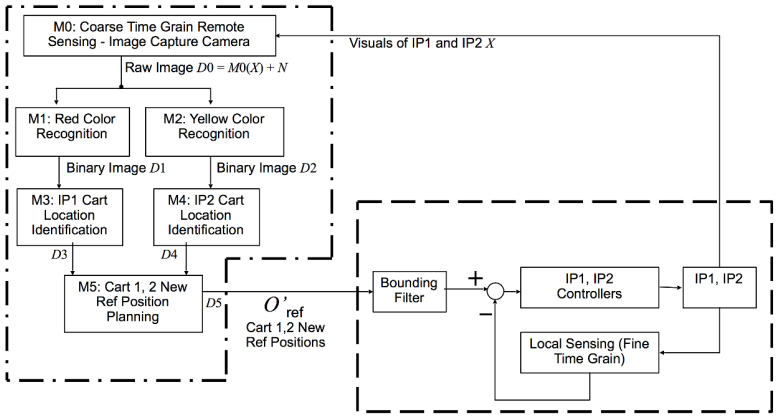


Figure 3.11. Testbed gray box details

As shown in Fig. 3.11, the cyber subsystem's computation data flow starts from M_0, the "Remote Sensing" module, where a USB 2Mega pixel camera captures 640×480 pixels raw images of IP_1 and IP_2. We denote the raw image captured as $D_0 = M_0(X) + N$, where X is the current plant state, and N is the cross-domain noise. D_0 is then fed to module M_1 and M_2 respectively for red and yellow color recognition. M_1's output D_1 is a binary image: a pixel of 1 means the corresponding pixel in D_0 is recognized as red; and 0 otherwise. Same is to M_2 and D_2, except the color to recognize is yellow.

The reason why we carry out red/yellow color recognition is because IP_1 and IP_2's carts respectively bear a red and a yellow label. By recognizing the red/yellow label, we can identify x_{ip1} and x_{ip2}, the current locations of the two carts. This is realized by feeding D_1, D_2 respectively to M_3 and M_4 for IP_1 and IP_2 cart localization.

The output of M_3, D_3, is the estimation of x_{ip1}; while the output of M_4, D_4, is the estimation of x_{ip2}. D_3 and D_4 are fed to M_5, the "Final Decision" module to compute the new reference point values, specifically, x_{ipref1} and x_{ipref2}.

3.5.2 Offline Profiling

We are given the following case study scenario.

There are two roles in the case study: *i)* the *manager* of our 2L-CCPS testbed is in charge of planning the budget to maintain the testbed; *ii)* the *vendors* sell *commercially-off-the-shelf* (COTS) digital modules to the manager.

All digital modules in the cyber subsystem of our testbed are COTS, bought from vendors by the manager. These modules are black boxes to the manager. Nevertheless, their respective API are open, so that the manager can replace/upgrade individual digital modules. Hence the cyber subsystem as a whole is a gray box to the manager.

On the other hand, the manager knows that the raw image data (i.e. D_0) captured from M_0 are noisy. The cross-domain noise propagates through the network of digital modules, and finally affects the plant. There is a need to upgrade the digital modules to make the testbed more robust to the cross-domain noise.

Specifically, the manager is offered two alternatives by the vendor: to upgrade M_1 to M_1'; or to upgrade M_3 to M_3'. To purchase either alternative costs 100 dollars; however, the manager only has 100 dollars of budget. To make a choice, the manager can carry out the computer emulation based profiling framework of Section 3.3 and 3.4, to compare the cross-domain noise impacts of both alternatives.

The first step of the framework is to prepare a benchmark $\mathcal{B} = \{X_i^0\}_{i=1,\ldots,b}$. We choose $b = 1000$. For the time being, let us first try the framework without benchmark region shrinking. That is, we sample $b = 1000$ benchmark points from the entire allowed region A.

For each benchmark point X_i^0 ($i = 1, \ldots, b$), the framework asks us to emulate

η elementary trials following the algorithm of Fig. 3.4. Particularly, we implement Line 2 according to the alternative way described in the comment. That is, we output $M_0(X_i^0) + N$ to the rest of the cyber subsystem to generate $O'_{\mathsf{ref}}(t_0)$.

For each $X_i^0 \in \mathcal{B}$, we prepare a high quality 640×480 pixels picture P_i as M_0's noiseless output. That is, $P_i = M_0(X_i^0)$. Let N denote the cross-domain noise RV; and $D_{0,i}$ denote the noisy output of M_0 corresponding to X_i^0. Then $D_{0,i} = M_0(X_i^0) + N = P_i + N$.

Indeed $D_{0,i}$ is also a 640×480 pixels picture, with each pixel inflicted by RV N. We can generate $D_{0,i}$ pixel by pixel. Let $P_i(j,k) \in [0, 255]$ ($j = 1, 2, \ldots, 640$; $k = 1, 2, \ldots, 480$) denote P_i's red-color value of the pixel at coordinate (j,k). Let $N(j,k) \in \mathbb{R}$ denote the component of cross-domain noise N at pixel coordinate (j,k). Let $D_{0,i}(j,k)$ denote the noisy raw image red-color value at pixel (j,k). Then $D_{0,i}(j,k) = P_i(j,k) + N(j,k)$ (in practice, $D_{0,i}(j,k)$'s value is rounded to the closest integer in $[0, 255]$).

Without loss of generality, our case study focuses/generates the cross-domain noise RV N per Gaussian distribution, i.e. $N(j,k) \sim \text{Normal}(0, \sigma^2)$. We define the *level* of N, denoted as $\|N\|$, with *mean square error* (MSE), a well-known concept in image processing.

$$\text{MSE} \stackrel{\mathsf{def}}{=} \frac{1}{J \cdot K} \sum_{j=1}^{J} \sum_{k=1}^{K} N^2(j,k), \tag{3.22}$$

where J and K are respectively the width and length of an image in pixels. It can be proven that $\mathsf{E}(\text{MSE}) = \sigma^2$.

We then discretize $10 \log_{10}$ MSE's value range into 5 intervals, respectively $(-\infty, -10)$, $[-10, 0)$, $[0, 10)$, $[10, 20)$, $[20, 30)$. Suppose the $10 \log_{10}$ MSE derived from the current

N falls in the lth ($l \in \{1, 2, \ldots, 5\}$) interval, then we say $\|N\| = l$.

With the above methodology to generate $D_{0,i} = M_0(X_i^0) + N$ for each benchmark point X_i^0, we can implement the elementary trial emulation described by Fig. 3.4 entirely.

Now we are ready to profile the impact of cross-domain noise to our test bed. We examine three cyber subsystem settings: no upgrade, upgrade M_1 only, upgrade M_3 only.

For each setting, for each benchmark point $X_i^0 \in \mathcal{B}$ ($i = 1, \ldots, 1000$) and each noise level $||N|| = l, l \in \{1, 2, \ldots, 5\}$, we carry out a campaign of $\eta = 1000$ elementary trial emulations, and derive the cross-domain noise impact value as per Eq. (3.9). According to Proposition 3.1, this guarantees a confidence level of 95% that the derived impact value error is within ± 0.032. For the bounding filter in the physical subsystem, we set its radius $\gamma = 0.001$ (see Fig. 3.11). All the emulations are carried out on a HP workstation with Intel Core I7-3610QM and 8G RAM.

The statistics of impact values over all benchmark points are shown and compared in Fig.3.12.

As the impact value indicate the increase of plant fault probability due to cross-domain noise N. The smaller the impact value, the more robust the system. Therefore, Fig.3.12 clearly favors upgrading M_1.

3.5.3 Offline Profiling with Shrunk Benchmark Region

In Section 3.5.2's profiling, the benchmark points are sampled from the entire allowed region A. By applying the benchmark region shrinking methodology proposed

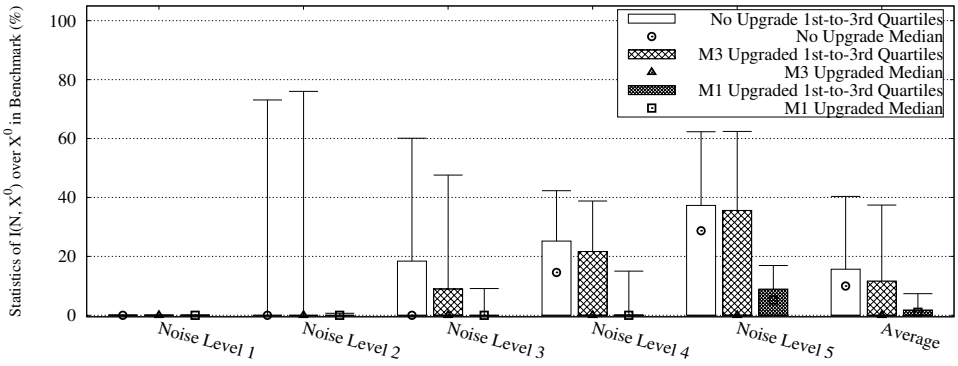


Figure 3.12. Statistics of cross-domain noise impact values $\{I(N, X^0)\}_{\forall X^0 \in \mathcal{B}}$, without shrinking benchmark region

in Section 3.4, we can sample less. Specifically, using existing LTI control Lyapunov analysis methodology [Bro91], we find for our test bed of Eq. (3.19)(3.20),

$$\mathbf{P} \;=\; \begin{pmatrix} \mathbf{Q} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q} \end{pmatrix},$$

where

$$\mathbf{Q} \;=\; \begin{pmatrix} 190.2853 & -50.0013 & 29.3842 & 10.9965 \\ -50.0013 & 436.0298 & -10.9938 & 442.5856 \\ 29.3842 & -10.9938 & 23.9030 & -50.0135 \\ 10.9965 & 442.5856 & -50.0135 & 639.884 \end{pmatrix}.$$

We choose $\varepsilon = 0.0002$, so the irrelevance distance $\Gamma = \sqrt{\frac{\lambda^{\max}(\mathbf{P})}{\lambda^{\min}(\mathbf{P})}}\gamma + \gamma + \varepsilon = 0.016$ (see Eq. (3.16)), which defines the shrunk benchmark region B via Eq. (3.18).

We reuse the benchmark points used in Section 3.5.2, but excluding all those outside of B. In this way, the shrunk benchmark region B removes 241 of the original 1000 benchmark points (i.e., 24.1% of the profiling computation effort is saved). The statistics of cross-domain noise impact values over the reduced benchmark are shown and compared in Fig. 3.13. The results also apparently favor upgrading M_1.

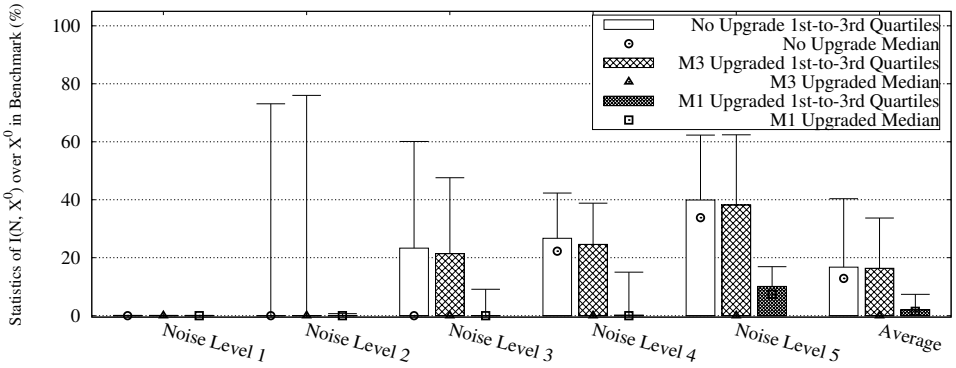


Figure 3.13. Statistics of cross-domain noise impact values $\{I(N, X^0)\}_{\forall X^0 \in \mathcal{B}}$, with shrunk benchmark region

3.5.4 Experiment Validation

We carry out experiments to validate the choice made by offline profiling in Section 3.5.2 and Section 3.5.3.

Specifically, we evaluate three scenarios of the testbed. In the first scenario, no digital module is upgraded. In the second scenario, the manager spends the 100 dollar budget to only upgrade M_1 to M_1'. In the third scenario, the manager spends the 100 dollar budget to only upgrade M_3 to M_3'.

For each scenario, we set the cross-domain noise level $\|N\|$ to 1, 2, 3, 4, and 5 (see Section 3.5.2 and Eq. (3.22) for the definition of these values; in our experiment implementation, module M_0, a noisy camera, is realized by appending a noise generator to a high quality camera's output). For each noise level, 20 elementary trial experiments are carried out. In each experiment, IP_1 and IP_2 start from random initial state uniformly picked from the allowed region A, and run for 1 minute. We record whether during this 1 minute, IP_1 and IP_2's state ever exceeds A. If so, a plant fault occurs.

Table 3.1 lists the experiment result: the total number of plant faults and the percentage of trials that involves faults. According to the table, upgrading M_1 apparently performs better than upgrading M_3 in terms of fault reduction. This matches the prediction made by offline analysis of Section 3.5.2 and Section 3.5.3, hence validates the usefulness of our proposed cross-domain noise profiling framework.

Table 3.1. Percentage of Trials that Encounter Plant Fault(s)

Scenario	Total Number of Faults	Faulty Trial Percentage
No Upgrade	49	49%
Upgrade M_1	20	20%
Upgrade M_3	39	39%

3.6 Related Work

The work of this chapter is a sub-problem of fault propagation profiling, a hot topic in system dependability research. Works of Hiller et al. [H+04] propose using conditional probability to profile the permeability, exposure, and impact of faults in a network of software modules. Johansson et al. [JS05] profile the transient data level

fault propagation in layered operating systems, also using the conditional probability models. Oliner et al. [O+10] [OA11] propose using principal component analysis and temporal correlations to discover influence relationships between software modules, to profile anomaly propagation. Distefano et al. [D+11] propose a compositional calculus to analyse software fault propagation with closed form formulae. Jhumka et al. [JL11] use software fault propagation profiling results to guide the placement of fault detector assertions. Pham et al. [P+15] [PD12] propose a UML based annotation and inference framework to analyze concurrent fault propagations in component based software systems.

However, all the above works focus on pure software system, rather than CPS.

There are works on profiling CPS fault propagation. Sierla et al. [S+13] study CPS fault propagation with an explicit object-oriented and event based model. Ge et al. [GPM09] analyse CPS failure probability using the PRISM [KNP02] probabilistic model checker. Remenyte et al. [RPA11] instead propose to use Petri nets. Anghel et al. [AWM07] propose using stochastic process models to analyse CPS fault propagation's temporal behaviors. There are also works on using various artificial intelligence/statistics tools to quantify CPS fault propagation [A+12] [J+09] [KL09] [S+05].

However, none of the above works models the control physical subsystem at the granularity of differential equation level, and neither do they exploit differential equation level control domain specific knowledge, Lyapunov stability theory in particular, to conduct the profiling.

As cross-domain noise profiling is a subtask of holistic system analysis/evaluation, the framework proposed by this work can be plugged into holistic system analysis/evaluation frameworks, such as FMEA/FMECA [US 15], FTA [US 14], PRA [PRA11], and FFIP [T+11].

For example, in FMEA, we can use this work's results as the input on the system failure rate due to cross-domain noise.

The work of this chapter is also related to fault-tolerant control CPS. Conventional fault-tolerant control CPS works deal with sensing errors, actuating errors, system parameter errors, or even system model changes. They typically require white box models of the cyber subsystem [G+15a] [HC10] [V+10] [Bub05]. Research on fault-tolerant control CPS with gray box cyber subsystems is relatively young. There are works on using redundancy to deal with faults in such control CPS [WHS13] [Kni12] [L+08]. Such topic is apparently orthogonal with this work's topic. Model predictive control [CB13] intends to learn an empirical model of the control CPS. That is, to empirically reverse-engineer the gray box cyber subsystem into white box. But this may not work for all cases, especially when the cyber subsystem is too complex. These cases (where the model predictive control does not work) fall in the context of this work. There are also works on using data mining, machine learning and/or inference to diagnose the cause of faults [G+15b] [T+13b] [M+10], so as to mitigate the cause. In contrast, our work is not about diagnosis. The cause of fault is given: the cross-domain noise. We want to profile its impact on the physical subsystem given different noise levels and various initial plant states. On the other hand, our profiling results can be used as a training set for data mining, machine learning. In this sense, this work complements the diagnosis works.

The content of this chapter is an extension of [T+14]. The content of this chapter is under review for journal publication.

3.7 Conclusion and Future Work

In this chapter, we propose a framework of methodology to profile the cross-domain noise in a generic two-level control CPS (2L-CCPS) architecture, whose cyber subsystem is gray box. In a more general sense, the framework addresses a fault propagation profiling problem in the control CPS context. Compared to existing control CPS fault propagation profiling solutions, our contributions lie in

1. The framework models the physical subsystem at differential equation level; and exploits differential equation level control domain specific knowledge.
2. We exploit the control domain-specific knowledge of Lyapunov stability to effectively reduce the profiling computation (24.1% in our case study).
3. We conducted case study on representative 2L-CCPS platforms to show case the usage of the profiling framework. Decisions made by the profiling framework are further verified by experiments.

The case study only reveals a small portion of our framework's potential. As future work, we will further investigate the use of our profiling framework for various stages of control CPS system engineering, including design stage, implementation stage, system integration stage, and maintenance stage. We will also investigate the use of our profiling framework for various activities of control CPS system engineering, including risk analysis, decision making, testing, debugging, and resource planning.

Chapter 4

Conclusion and Future Work

We understand that CPS is the result of the inevitable convergence between the cyber and the physical world. As many CPSs are safety/mission critical, dependability is a top concern. Conventionally, there are three main approaches to guarantee dependability: fault prevention, fault tolerance, and fault removal. However, in the context of CPS, we face new challenges for these fault handling methods. First, the tight coupling between the cyber and the physical world invalidates many conventional assumptions. Second, the inter-disciplinary nature calls for new solutions that obey and exploit the cross-domain constraints and knowledge. How to address these challenges for CPS fault prevention, fault tolerance, and fault removal is a huge problem space.

In this thesis, we made some initial attempts to explore this problem space.

The study to guarantee PTE safety rule in wireless CPS (see Chapter 2) is an attempt toward CPS fault prevention. PTE safety rule is a characteristic and important safety rule for distributed CPS, as it involves real-time temporal behavior and lays foundations for CPS mutual exclusion/interlocking. We propose a design pattern, which pre-

vents PTE safety rule violation faults by design, despite of arbitrary possible wireless communication failures in runtime. The design pattern follows leasing design philosophy, and is formalized by hybrid automata. Mechanical elaboration methods are also proposed to turn the design pattern into specific designs. This lays the foundation for future programming automation. Our case studies on laser tracheotomy wireless CPS and inverted pendulum remote control CPS validate the proposed method. The proposed method also outperforms existing methods under benign or moderately adverse wireless channel conditions. Following this study, as future work, we plan to investigate additional safety rules for more CPS application contexts, and propose a comprehensive framework for CPS collaboration interlocking/mutual exclusion.

The study to profile cross-domain noise in control CPS provides fundamental tools for CPS fault tolerance and fault removal (see Chapter 3). The proposed hybrid automata reachability based metric enables quantifications of CPS fault tolerance. Treating the cyber subsystem as a gray box matches the fact that cyber subsystems are becoming more complex, exceeding the reach of explicit modeling and analysis. This makes benchmarking an indispensable tool for CPS testing and fault removal. The proposed benchmark shrinking technology exploits the Lyapunov stability theories of control CPS, hence effectively reduces the workload of benchmarking. This lays the foundation for more efficient testing, and hence more efficient fault removal for control CPS. The proposed framework is validated by multiple case studies and experiments. Following this study, as future work, we plan to carry out more case studies, and investigate the use of the proposed framework in other system engineering stages, particularly in CPS debugging.

Chapter 5

Appendix

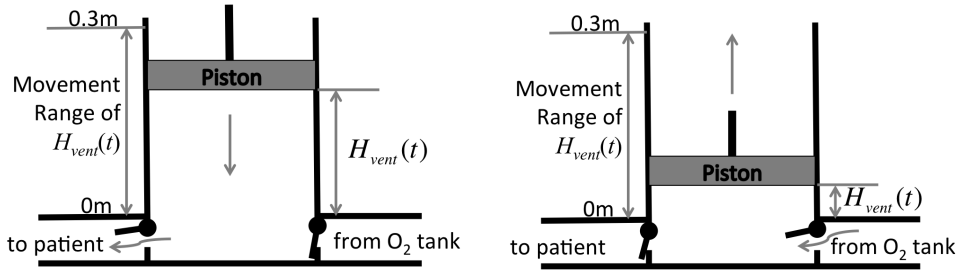
5.1 Ventilator Working Mechanism

The working mechanism of our ventilator is illustrated by Fig. 5.1.

Basically, our ventilator consists of a cylinder, a piston, and two valves. The piston moves up/down the cylinder to pump oxygen from oxygen tank into patient. When the piston moves downward, the valve toward the patient is opened and the valve from the oxygen tank is closed, hence oxygen is pumped out to the patient, forcing the patient to inhale. When the piston moves upward, the valve from the oxygen tank is opened, hence oxygen is pumped into the cylinder; meanwhile the valve to the patient is closed, allowing the patient to naturally exhale due to his/her chest weight.

We denote the current height of the piston as $H_{\text{vent}}(t)$; its movement range is from 0(m) to 0.3(m). $\dot{H}_{\text{vent}}(t)$ is the velocity of the piston. When the piston moves downward, $\dot{H}_{\text{vent}}(t) = -0.1$(m/s). When the piston moves upward, $\dot{H}_{\text{vent}}(t) = +0.1$(m/s).

The piston movement changes direction when $H_{\text{vent}}(t)$ hits its bottom 0(m) or ceiling 0.3(m).



(a) Pumping Out (b) Pumping In

Figure 5.1. Ventilator Working Mechanism. $H_{\text{vent}}(t)$ **is the piston height at time** t**.** $\dot{H}_{\text{vent}}(t)$ **is the piston velocity at time** t**. (a) when the piston moves downward, oxygen is to pumped out to patient (forcing patient to inhale); (b) when the piston moves upward, oxygen is pumped in from tank (meanwhile patient exhales naturally due to chest weight).**

5.2 Formal Definitions of Hybrid Automata

As one goal of this work is to provide *formal* descriptions and analysis, it is necessary to first give the formal definition of hybrid automaton. We use the hybrid automaton of Fig. 2.2 to explain the following abstract definitions.

According to [A$^+$93, H$^+$95, A$^+$96], a hybrid automaton A is a tuple $(\vec{x}(t), V, \mathsf{inv}, F, E, g, R, L, \mathsf{syn}, \Phi_0)$ of following components:

1. A *data state variables vector* $\vec{x}(t) = (x_1(t), x_2(t), \ldots, x_n) \in \mathbb{R}^n$ of n *data state variables* of time t, where n is called the *dimension* of A. A possible evaluation of $\vec{x}(t)$, denoted as $\vec{s} \in \mathbb{R}^n$, is called a *data state* of A (at time t). In the example of Fig. 2.2, the data state variables vector is $(H_{vent}(t))$, i.e. it contains only one data state

variable: $H_{vent}(t)$, which is the height of the ventilator piston at time t.

2. A finite set V of vertices called *locations*. The *state* of A (at time t) is a tuple $\phi(t) = (\ell(t), \vec{x}(t))$ of two variables of time t: the aforementioned data state variables vector $\vec{x}(t)$, and the *location counter* $\ell(t) \in V$, which indicates the current location that A dwells at. In the example of Fig. 2.2, the ventilator hybrid automaton has two locations: PumpOut and PumpIn.

3. A function inv that assigns to each $v \in V$ a subset of $\mathbb{R}^n$, aka. the *invariant set*. As long as the location counter $\ell(t) = v$, $\vec{x}(t)$ must satisfy $\vec{x}(t) \in \mathsf{inv}(v)$. In the example of Fig. 2.2, in location PumpOut, the invariant is that the ventilator piston height $H_{vent}(t)$ stays in the range $0 \leq H_{vent}(t) \leq 0.3$(m).

4. A set of *flow maps* $F = \{f_v | f_v : \mathbb{R}^n \mapsto \mathbb{R}^n, \forall v \in V\}$, with each element f_v defining a set of *differential equations* $\dot{\vec{x}} = f_v(\vec{x})$ over data state variables vector $\vec{x}(t)$ for each location $v \in V$. These differential equations specify the *continuous dynamics* of $\vec{x}(t)$ when $\ell(t) = v$. In the example of Fig. 2.2, in location PumpOut, the flow maps only involve one differential equation: $\dot{H}_{vent}(t) = -0.1$(m/s), i.e. the ventilator piston pushes downward at a velocity of -0.1(m/s).

5. A finite set of *edges* E. Each edge $e \in E$ identifies a *discrete transition* (v, v') from a source location $v \in V$ to a destination location $v' \in V$. We denote the source location of edge e as $\mathsf{src}(e)$; while the destination location as $\mathsf{des}(e)$. An edge $e = (v, v')$ specifies the possible *discrete dynamics* of A's state: it can switch from $\ell(t) = v$ to $\ell(t^+) = v'$. In the example of Fig. 2.2, there are two edges: from location PumpOut to PumpIn, and vice versa.

6. A *guard* function $g : E \mapsto \mathbb{R}^n$ that assigns each $e \in E$ a *guard set* $g(e) \subseteq \mathsf{inv}(\mathsf{src}(e))$. Discrete transition e can only take place when $\vec{x}(t) \in g(e)$. In the example

of Fig. 2.2, the guard condition for the edge (transition) from PumpOut to PumpIn is that the ventilator piston reaches the bottom of its movement range, i.e. $H_{vent}(t) = 0$.

7. A finite set of *reset functions* $R = \{r_e | r_e : \mathsf{inv}(\mathsf{src}(e)) \mapsto 2^{\mathsf{inv}(\mathsf{des}(e))}, \forall e \in E\}$. When the A's state switches from $\ell(t) = \mathsf{src}(e)$ to $\ell(t^+) = \mathsf{des}(e)$ via transition $e \in E$, $\vec{x}(t^+)$ is assigned a new data state from set $r_e(\vec{x})$. In the example of Fig. 2.2, the reset functions for both edges are the identity function, i.e., the state variables vector $((H_{vent}(t))$ does not change value after each transition (edge). We hence omit the reset functions in the figure.

8. A finite set L of *synchronization labels* and a *synchronization labeling function* syn that assigns to each edge $e \in E$ a synchronization label $\mathsf{syn}(e) \in L$. A synchronization label consists of a *root* and a *prefix*, which respectively represent a *event* and the *role* of the hybrid automaton for that event.

When entity ξ_1 (whose hybrid automaton is A_1) sends an event l to entity ξ_2 (whose hybrid automaton is A_2), a transition e_1 in A_1 takes place; and on receiving the event, transition e_2 is triggered in A_2. Correspondingly, we put a synchronization label $!l$ to e_1 and $?l$ to e_2. We respectively add the prefixes ! and ? to the root l, to distinguish the sender and the receiver of event l. In case l is received unreliably, which is typical for wireless, we use $??$ instead of a single ? prefix. Synchronization labels with different prefixes or roots are regarded as different. For example, $!l$, $?l$, $??l$ are considered three different synchronization labels, though they are related to a same event by the root l.

If an event (correspondingly, a synchronization label root) is communicated across multiple hybrid automata, then the corresponding synchronization labels are *external*; otherwise, the corresponding synchronization labels are *internal*. For an internal synchronization label whose corresponding event does not have receiver(s), prefix ! is omit-

ted.

In the example of Fig. 2.2, when the transition from location PumpOut to PumpIn happens, event $evtVPumpIn$ happens; in the other way around, event $evtVPumpOut$ happens. The ! prefix to $evtVPumpIn$ and $evtVPumpOut$ in the figure indicates the events are broadcast. If there are other hybrid automata in the system, some transitions may be triggered on receiving these events, the corresponding transitions are labeled with $?evtVPumpIn$ or $?evtVPumpOut$. In case the reception of events are via unreliable (e.g. wireless) communication links, the corresponding labels should be $??evtVPumpIn$ or $??evtVPumpOut$.

9. A set of *possible initial states* $\Phi_0 \subseteq \{(v, \vec{s}) \in V \times \mathbb{R}^n | v \in V, \vec{s} \in \mathsf{inv}(v)\}$. We also call Φ_0's projection on location set V as *initial locations*, denoted as $\Phi_0|_V$. In the example of Fig. 2.2, the possible initial states can be $\Phi_0 = \{(\text{PumpOut}, (h_0))\}$, where $h_0 \in [0, 0.3]$; i.e. starting from location PumpOut and piston height $H_{vent}(0) \in [0, 0.3]$(m).

5.3 Detailed Diagrams for Design Pattern Hybrid Automata

Please see Fig. 5.2, 5.3, and 5.4 for the detailed diagram of A_{supvsr}, A_{initzr}, and $A_{\text{ptcpnt},i}$, the hybrid automaton for Supervisor, Initializer, and the ith Participant respectively. Note all hybrid automata's initial locations are "Fall-Back", and all data state variables are initialized to 0.

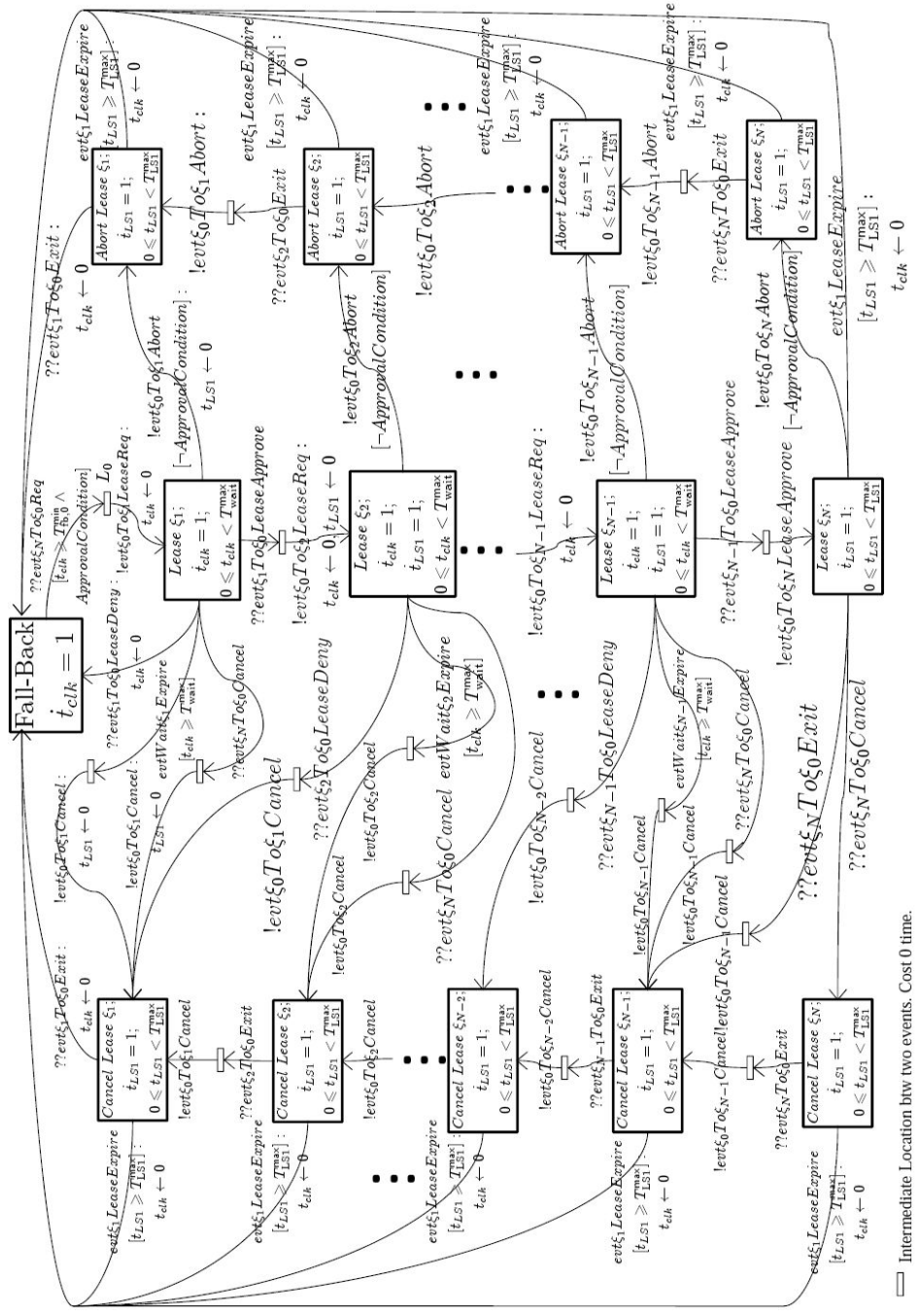


Figure 5.2. Diagram of Hybrid Automaton A_{supvsr}, the Design Pattern for Supervisor. Each rectangle box indicates a location. Inside the box, the first line is the location's name (note state variable names and location names are local to their respective hybrid automata; hence two distinct locations of two distinct hybrid automata may have the same name). Annotations to each edge (aka transition) comply with the following conventions. Before the ':' are the synchronization label and the guard formula (quoted by brackets "[]") for the edge. After the ':' are the data state value resets ("$x \leftarrow a$" means "assigning x of value a"). The above notational conventions also apply to Fig. 5.3 and 5.4.

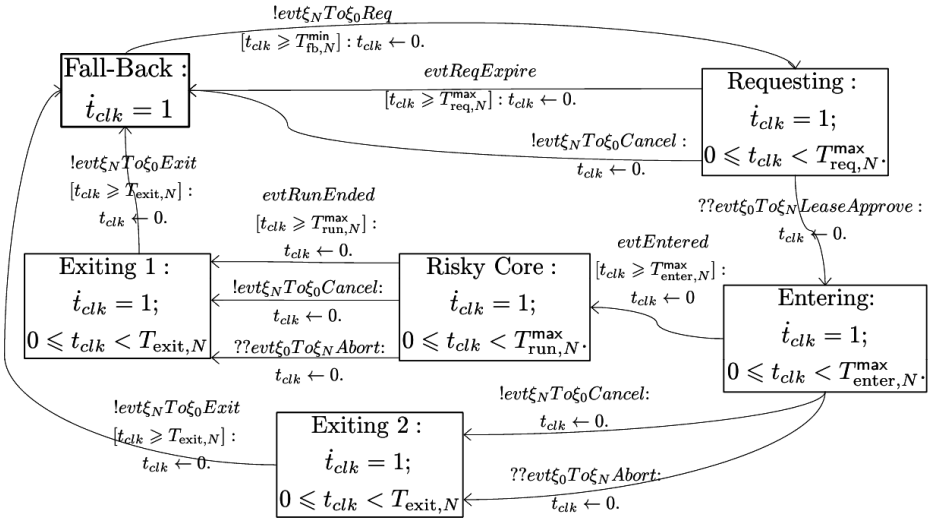


Figure 5.3. Diagram of Hybrid Automaton A_{initzr}, the Design Pattern for Initializer

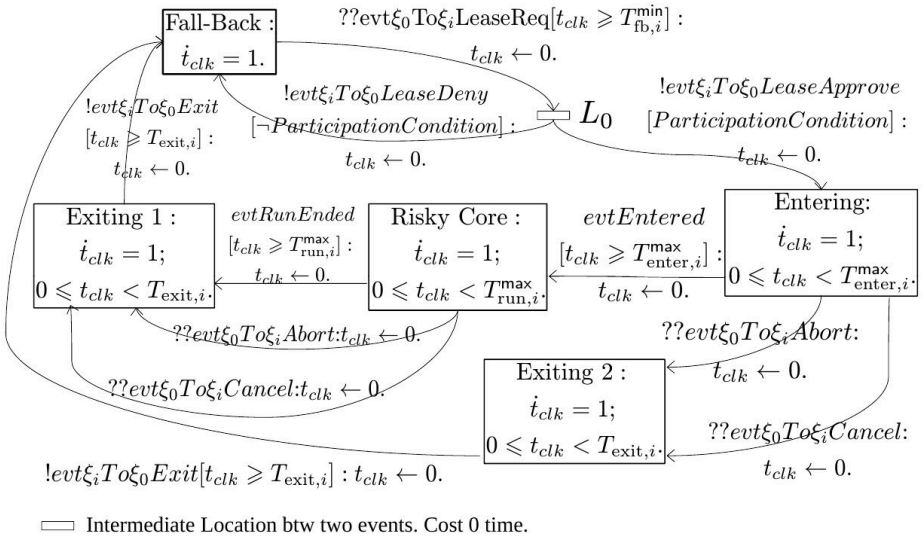


Figure 5.4. Diagram of Hybrid Automaton $A_{\text{ptcpnt},i}$, the Design Pattern for the ith Participant.

5.4 Proof of Theorem 2.1

First, we can prove the following lemma.

LEMMA 5.1 (GUARANTEED RESETTING) *Suppose* $evt\xi_0To\xi_1LeaseReq$ *happens at time* t_0*, then we have*

$\forall i \in \{0, 1, \ldots, N\} \cdot \ell_i(t_0^-) \equiv$ *"Fall-Back".*

Proof: First, because $evt\xi_0To\xi_1LeaseReq$ happens at t_0 and ξ_0's location "L0" allows 0 dwelling time, we have $\ell_0(t_0^-) =$ "Fall-Back". ξ_0's location "L0" allows 0 dwelling time also implies $evt\xi_NTo\xi_0Req$ happens at t_0, so $\ell_N(t_0^-) =$ "Fall-Back" (see Fig. 5.3).

Now all we need to prove is

$$\ell_i(t_0^-) = \text{ "Fall-Back" } (i = 1 \sim N - 1). \tag{5.1}$$

We can prove this inductively. As all hybrid automata of $\mathcal{H}$ start from respective "Fall-Back" locations, if t_0 is the first time $evt\xi_0To\xi_1LeaseReq$ happens, Claim (5.1) sustains.

Suppose Claim (5.1) sustains for t_0, a time instance that $evt\xi_0To\xi_1LeaseReq$ happens. Suppose $t_1 > t_0$ is the next time instance that $evt\xi_0To\xi_1LeaseReq$ happens. Then there can be two cases for the interval of $[t_0, t_1)$.

Case 1: During $[t_0, t_1)$, $evt\xi_1 LeaseExpire$ never happens.

Suppose during $[t_0, t_1)$, ℓ_0 ever reaches location "Lease ξ_j" but not location "Lease ξ_{j+1}" ($j \in \{1, 2, \ldots, N-1\}$).

Then throughout $[t_0, t_1)$, ξ_{j+1}, ξ_{j+2}, ..., ξ_{N-1} never get a chance to leave their respective "Fall-Back" locations.

By checking all possible exit paths from "Lease ξ_j" in Fig. 5.2, and knowing that $evt\xi_1 LeaseExpire$ never happens, we find for each entity ξ_k (where $k = j, j-1, \ldots, 1$), $\exists t \in [t_0, t_1)\cdot$ either $evt\xi_k To\xi_0 LeaseDeny$ or $evt\xi_k To\xi_0 Exit$ happens at t. For otherwise, ℓ_0 cannot be at "Fall-Back" at t_1^-, which has already been proven at the beginning of this proof. Meanwhile, according to Fig. 5.4, as soon as $evt\xi_k To\xi_0 LeaseDeny$ or $evt\xi_k To\xi_0 Exit$ happens, ℓ_k enters "Fall-Back", and stays there till t_1^- because there is no more $evt\xi_0 To\xi_1 LeaseReq$ (hence $evt\xi_0 To\xi_k LeaseReq$) during $[t, t_1)$.

The same analysis applies to the case when ℓ_0 ever reaches location "Lease ξ_N" during $[t_0, t_1)$.

Therefore, Claim (5.1) sustains for t_1 in Case 1.

Case 2: $\exists t \in [t_0, t_1)$, $evt\xi_1 LeaseExpire$ happens. Then according to Fig. 5.2, 5.3, and 5.4, $t_0 + T_{\mathrm{LS1}}^{\mathsf{max}} \leq t$. As defined, $t < t_1$, so $t_0 + T_{\mathrm{LS1}}^{\mathsf{max}} < t_1$.

On the other hand, the latest time during $[t_0, t_1)$ for ξ_i ($i \in \{1, 2, \ldots, N-1\}$) to leave "Fall-Back" (if it ever leaves) is at $t_0 + (i-1)T_{\mathrm{wait}}^{\mathsf{max}}$ (see Fig. 5.2, 5.4). After that, ξ_i's maximal stay outside of "Fall-Back" is $T_{\mathrm{enter},i}^{\mathsf{max}} + T_{\mathrm{run},i}^{\mathsf{max}} + T_{\mathrm{exit},i}$ (see Fig. 5.4). Because of Condition c4, this means by $t_0 + T_{\mathrm{LS1}}^{\mathsf{max}} < t_1$, ξ_i should have returned "Fall-Back". Due to the same reason as in Case 1, after the return, ξ_i should have stayed in "Fall-Back" till t_1^-.

Therefore, Claim (5.1) sustains for t_1 in Case 2.

Due to Case 1 and 2, Claim (5.1) sustains for t_1. Induction sustains.

Note no matter there are infinite, or finite occurrences of $evt\xi_0To\xi_1LeaseReq$, the above induction based proof sustains. ■

Then we have the following lemma.

LEMMA 5.2 (SINGLE VISIT BETWEEN RESETS) *Let t_0, t_1 ($t_0 < t_1$) be the time instances that two consecutive $evt\xi_0To\xi_1LeaseReq$ happen; or let t_0 be the last time that $evt\xi_0To\xi_1LeaseReq$ happens and $t_1 = \infty$. Either way,*

$$\forall t \in (t_0, t_1) \cdot \textit{no } evt\xi_0To\xi_1LeaseReq \textit{ happens at } t. \tag{5.2}$$

We then have

Claim 5.2.1: $\forall i \in \{1, 2, \ldots, N-1\}$, throughout interval $[t_0, t_1)$, ξ_i can respectively enter its location (set) "L0", "Entering", {"Risky Core", "Exiting 1"}, and "Exiting 2" (see Fig. 5.4) for at the most once, and continuously dwell there for no more than 0, $T^{\mathsf{max}}_{\mathrm{enter},i}$, $T^{\mathsf{max}}_{\mathrm{run},i} + T_{\mathrm{exit},i}$, and $T_{\mathrm{exit},i}$ respectively.

Claim 5.2.2: Throughout interval $[t_0, t_1)$, ξ_N can respectively enter its location (set) "Entering", {"Risky Core", "Exiting 1"}, and "Exiting 2" (see Fig. 5.3) for at the most once, and continuously dwell there for no more than $T^{\mathsf{max}}_{\mathrm{enter},N}$, $T^{\mathsf{max}}_{\mathrm{run},N} + T_{\mathrm{exit},N}$, and $T_{\mathrm{exit},N}$ respectively.

Proof: Due to Lemma 5.1, we have $\forall i \in \{0, 1, \ldots, N\} \cdot \ell_i(t_0^-) =$ "Fall-Back".

$\forall i \in \{1, 2, \ldots, N-1\}$, for ℓ_i to enter "L0" twice in interval $[t_0, t_1)$, $evt\xi_0To\xi_iLeaseReq$ must happen twice (see Fig. 5.4), which implies $evt\xi_0To\xi_1LeaseReq$ happen twice (see Fig. 5.2), which implies $\exists t_2 \in (t_0, t_1) \cdot evt\xi_0To\xi_1LeaseReq$ happens at t_2. This contradicts Formula (5.2) (note due to c1, $T_{\text{fb},0}^{\text{min}} > 0$, hence no zeno can happen). The continuous dwelling time upper bound of 0 follows naturally due to "L0"'s dwelling constraint.

Same reasoning also applies to location (set) "Entering", {"Risky Core", "Exiting 1"}, and "Exiting 2". Hence Claim 5.2.1 is proven.

Same way we can prove Claim 5.2.2, where $evt\xi_0To\xi_NLeaseApprove$ replaces $evt\xi_0To\xi_iLeaseReq$. ■

Furthermore, we have the following lemma.

LEMMA 5.3 (PTE COMPLIANCE) *Let t_0, t_1 ($t_0 < t_1$) be the time instances when two consecutive $evt\xi_0To\xi_1LeaseReq$ happen; or let t_0 be the last time $evt\xi_0To\xi_1LeaseReq$ happens and $t_1 = \infty$. In both cases, $\forall t \in [t_0, t_1)$, Ineq.* (2.1) *sustains.*

Proof: Without loss of generality, let us first focus on entity ξ_i and ξ_{i-1} (where $i \in \{2, 3, \ldots, N-1\}$).

If ξ_i never entered risky-locations throughout interval $[t_0, t_1)$, then the PTE ordering of $\xi_{i-1} < \xi_i$ trivially sustains.

Otherwise, there must be a maximal interval $[t_2, t_3) \subseteq [t_0, t_1)$ that ℓ_i stays in risky-locations. That is, $evtEntered$ happens at t_2, $evt\xi_i To \xi_0 Exit$ happens at $t_3 < t_1$ (no matter $t_1 < \infty$ or $t_1 = \infty$, we know $t_3 < t_1$ due to Lemma 5.1 and 5.2), and $\forall t \in (t_2, t_3) \cdot \ell_i(t) \in \{$"Risky Core", "Exiting 1"$\}$.

Due to Lemma 5.2, (t_2, t_3) is the only maximal interval in $[t_0, t_1)$ that ℓ_i stays in risky-locations. Due to Lemma 5.1, $\ell_i(t_0^-) =$ "Fall-Back". By exhaustively examining all possible paths in $A_{\text{ptcpnt},i}$ of Fig. 5.4, we have

$$t_0 + T^{\mathsf{max}}_{\text{enter},i} \leq \quad t_2 \quad \leq t_0 + (i-1)T^{\mathsf{max}}_{\text{wait}} + T^{\mathsf{max}}_{\text{enter},i}; \tag{5.3}$$

$$\text{and } t_2 + T_{\text{exit},i} \leq \quad t_3 \quad \leq t_2 + T^{\mathsf{max}}_{\text{run},i} + T_{\text{exit},i}. \tag{5.4}$$

Now let us check the duration that ξ_{i-1} may stay in its risky-locations within $[t_0, t_1)$.

By exhaustively examining all possible paths in $A_{\text{ptcpnt},i}$ of Fig. 5.4, for ξ_i's $evtEntered$ to happen at t_2, $evt\xi_i To \xi_0 LeaseApprove$ must happen at

$$t_4 = t_2 - T^{\mathsf{max}}_{\text{enter},i}. \tag{5.5}$$

Note $t_4 \geq t_0$ because of Ineq. (5.3). According to A_{supvsr} of Fig. 5.2, because $\ell_0(t_0^-) =$ "Fall-Back", for $evt\xi_i To \xi_0 LeaseApprove$ to happen at $t_4 \in [t_0, t_2)$, $evt\xi_{i-1} To \xi_0 LeaseApprove$ must have happened at some time instance t_5, where

$$t_0 \leq t_5 \leq t_4. \tag{5.6}$$

Due to Condition c5,

$$\begin{aligned} & t_5 + T^{\mathsf{max}}_{\mathrm{enter},i-1} + T^{\mathsf{min}}_{\mathrm{risky}:i-1\to i} \\ < \;& t_4 + T^{\mathsf{max}}_{\mathrm{enter},i} = t_2 \quad \text{(due to Eq. (5.5)).} \end{aligned} \tag{5.7}$$

On the otherhand, because after t_0, the first occurrence of $evt\xi_i To\xi_0 Exit$ happens at t_3, by exhaustively checking all possible paths in A_{supvsr} of Fig. 5.2, this implies the following proposition.

PROPOSITION 5.1 *No $evt\xi_0 To\xi_j Abort$ nor $evt\xi_0 To\xi_j Cancel$ ever happened during $[t_0, t_3)$ ($\forall j = i-1, i-2, \ldots, 1$).* □

Because of Proposition 5.1, Ineq. (5.6)(5.7) imply that after ℓ_{i-1} enters location "Entering" at t_5, it enters "Risky Core", i.e. risky-locations, at

$$t_6 = t_5 + T^{\mathsf{max}}_{\mathrm{enter},i-1}. \tag{5.8}$$

Due to Ineq. (5.7), we have

$$t_6 < t_2 - T^{\mathsf{min}}_{\mathrm{risky}:i-1\to i} < t_3. \tag{5.9}$$

On the other hand, from Fig. 5.2, we see

$$t_5 \geq t_4 - T^{\mathsf{max}}_{\mathrm{wait}}. \tag{5.10}$$

Ineq. (5.10) and Condition c6 together imply

$$\begin{aligned} & t_5 + T^{\mathsf{max}}_{\mathrm{enter},i-1} + T^{\mathsf{max}}_{\mathrm{run},i-1} \\ > \;& t_4 + T^{\mathsf{max}}_{\mathrm{enter},i} + T^{\mathsf{max}}_{\mathrm{run},i} + T_{\mathrm{exit},i} \\ = \;& t_2 + T^{\mathsf{max}}_{\mathrm{run},i} + T_{\mathrm{exit},i} \quad \text{(due to Eq. (5.5))} \\ \geq \;& t_3. \quad \text{(due to Ineq. (5.4))} \end{aligned} \tag{5.11}$$

Due to Proposition 5.1, during $[t_5, t_3)$, ξ_{i-1} never receives $evt\xi_0 To\xi_{i-1} Abort$ or $evt\xi_0 To\xi_{i-1} Cancel$. This fact, combined with Ineq. (5.11), implies the earliest time instance after t_5 that ξ_{i-1} may receive $evt\xi_0 To\xi_{i-1} Abort$ or $evt\xi_0 To\xi_{i-1} Cancel$ is t_3. Let $t_7 \in [t_0, t_1)$ be the time instance that ℓ_{i-1} exits risky-locations, then

$$\begin{aligned} t_7 \;\geq\; & \mathsf{min}\{t_5 + T^{\mathsf{max}}_{\mathrm{enter},i-1} + T^{\mathsf{max}}_{\mathrm{run},i-1} + T_{\mathrm{exit},i-1}, \\ & \quad t_3 + T_{\mathrm{exit},i-1}\} \\ = \;& t_3 + T_{\mathrm{exit},i-1} \quad \text{(due to Ineq. (5.11))} \\ > \;& t_3 + T^{\mathsf{min}}_{\mathrm{safe}:i\to i-1} \quad \text{(due to Condition c7)} \end{aligned} \tag{5.12}$$

Note Ineq. (5.9)(5.12) implies $t_6 < t_7$; and no matter $t_1 < \infty$ or $t_1 = \infty$, Lemma 5.1 and 5.2 imply $t_7 < t_1$. Therefore, to summarize, ξ_{i-1} must have visited risky-locations for once during interval $[t_0, t_1)$; and the visit starts at t_6, and ends at t_7, where t_6 compiles with Ineq. (5.9), and t_7 complies with Ineq. (5.12). In other words, the PTE ordering of $\xi_{i-1} < \xi_i$ sustains during interval $[t_0, t_1)$.

Using the same approach, we can also prove $\xi_{N-1} < \xi_N$ during $[t_0, t_1)$. ■

With the above lemmas, we can prove Theorem 2.1 as follows.

Proof of Claim 1:

Throughout the execution of hybrid system $\mathcal{H}$, if $evt\xi_0To\xi_1LeaseReq$ never happens, as all entities are initialized from "Fall-Back" locations, the PTE safety rules trivially sustain.

If infinite number of $evt\xi_0To\xi_1LeaseReq$ happen. Then before the first $evt\xi_0To\xi_1LeaseReq$, all entities stay in "Fall-Back" locations, PTE safety rules trivially sustain. After that, between any two consecutive $evt\xi_0To\xi_1LeaseReq$, due to Lemma 5.1 and 5.2, PTE Safety Rule 2.1 sustains, and a risky-locations continuous dwelling time upper bound is $T^{\mathsf{max}}_{\mathrm{run},i} + T^{\mathsf{max}}_{\mathrm{exit},i}$ for ξ_i ($i = 1, 2, \ldots, N$); due to Lemma 5.3, PTE Safety Rule 2.2 also sustains. Therefore, PTE safety rules sustain.

The same proving approach can be applied to the scenario where finite number of $evt\xi_0To\xi_1LeaseReq$ happen (we need to check the special case: the interval between the last occurrence of $evt\xi_0To\xi_1LeaseReq$ and time ∞; but the same proving approach can be applied, and the conclusion is the same). ■

Proof of Claim 2.i:

We can prove by contradiction. Suppose in any duration of length $T^{\mathsf{min}}_{\mathrm{fb},N} + T^{\mathsf{max}}_{\mathrm{req},N}$ in $[t_0, +\infty)$, ξ_N never get a chance to send $evt\xi_NTo\xi_0Req$. Then ξ_0 can never leave location "Fall-Back" in $[t_0, +\infty)$ (see Fig. 5.2, 5.3 in Appendix 5.3); hence can never send $evt\xi_0To\xi_NLeaveApprove$; hence ξ_N can never leave the location set of {"Fall-Back", "Requesting"} in $[t_0, +\infty)$. Then in any duration $[t_a, t_b]$ of length $T^{\mathsf{min}}_{\mathrm{fb},N} + T^{\mathsf{max}}_{\mathrm{req},N}$ in $[t_0, +\infty)$, suppose

a) at t_a, ξ_N resides in "Fall-Back", then by $t_a + T^{\mathsf{min}}_{\mathrm{fb},N} \in [t_a, t_b]$, ξ_N get the chance to send $evt\xi_NTo\xi_0Req$, contradiction reached;

b) at t_a, ξ_N resides in "Requesting", then by $t_a+T^{\mathsf{max}}_{\mathrm{req},N} \in [t_a, t_b]$, ξ_N must have returned to "Fall-Back", suppose the return time instance is t_c, then $t_a \leq t_c \leq t_a + T^{\mathsf{max}}_{\mathrm{req},N} \leq t_b$, then by $t_c+T^{\mathsf{min}}_{\mathrm{fb},N} \leq t_a+T^{\mathsf{max}}_{\mathrm{req},N}+T^{\mathsf{min}}_{\mathrm{fb},N} = t_b$, ξ_N get the chance to send $evt\xi_N To\xi_0 Req$, also reached contradiction. ■

Proof of Claim 2.ii:

According to Supervisor's hybrid automata A_{supvsr} (see Fig. 5.2 in Appendix 5.3), every location other than "Fall-Back" has a dwelling time upper bound. By checking all possible paths of A_{supvsr}, we know that ξ_0 can continuously stay away from "Fall-Back" for at the most $T_{\mathrm{reset},0} \stackrel{\mathsf{def}}{=} (N-1)T^{\mathsf{max}}_{\mathrm{wait}} + T^{\mathsf{max}}_{\mathrm{LS1}}$. Therefore, once ξ_0 non-zenoly leaves "Fall-Back" at t_{00}, ξ_0 will return to "Fall-Back" by $t^+_{00} + T_{\mathrm{reset},0}$. That is, $\exists t_a \in (t^+_{00}, t^+_{00} + T_{\mathrm{reset},0}]$, such that ξ_0 first returns to "Fall-Back" at t_a.

Meanwhile, according to Initializer's hybrid automata A_{initzr} (see Fig. 5.3 in Appendix 5.3), every location other than "Fall-Back" has a dwelling time upper bound. By checking all possible paths of A_{initzr}, we know that ξ_N can continuously stay away from "Fall-Back" for at the most $T_{\mathrm{reset},N} \stackrel{\mathsf{def}}{=} T^{\mathsf{max}}_{\mathrm{req},N} + T^{\mathsf{max}}_{\mathrm{enter},N} + T^{\mathsf{max}}_{\mathrm{run},N} + T_{\mathrm{exit},N}$.

Therefore, $\exists t_b \in [t_a, t_a + T_{\mathrm{reset},N} + T^{\mathsf{min}}_{\mathrm{fb},N}]$, such that t_b is the first time instance in $[t_a, t_a + T_{\mathrm{reset},N} + T^{\mathsf{min}}_{\mathrm{fb},N}]$ that ξ_N have continuously resided in "Fall-Back" for at least $T^{\mathsf{min}}_{\mathrm{fb},N}$. In other words, t_b is the first time instance in $[t_a, t_a + T_{\mathrm{reset},N} + T^{\mathsf{min}}_{\mathrm{fb},N}]$ that ξ_N can send $evt\xi_N To\xi_0 Req$ to ξ_0.

As at t_a, ξ_0 resides in "Fall-Back", and ξ_0 cannot leave "Fall-Back" unless receiving $evt\xi_N To\xi_0 Req$ from ξ_N. Suppose ξ_N sends ξ_0 event $evt\xi_N To\xi_0 Req$ at t_b, and ξ_0 receives the event, then this will trigger ξ_0 to send $evt\xi_0 To\xi_1 LeaseReq$ at t_b. Then according to Lemma 5.1, we can infer that at t^-_b, all entities ($\xi_0 \sim \xi_N$) reside in "Fall-

Back". As all entities' residing location at t_b^- is not determined by events happening in t_b, therefore, even if ξ_0 does not receive $evt\xi_N To\xi_0 Req$ at t_b, we can still conclude that at t_b^-, all entities are residing in "Fall-Back".

As $t_b \geq t_a > t_{00}^+$, we have $t_b^- > t_{00}$.

As $t_b \leq t_a + T_{\mathrm{fb},N}^{\mathsf{min}} + T_{\mathrm{reset},N} \leq t_{00}^+ + T_{\mathrm{reset},0} + T_{\mathrm{fb},N}^{\mathsf{min}} + T_{\mathrm{reset},N} = t_{00}^+ + T_{\mathrm{reset}}$, we have $t_b^- \leq t_{00} + T_{\mathrm{reset}}$.

Therefore, we conclude that $\exists t \in (t_{00}, t_{00} + T_{\mathrm{reset}}]$, such that all entities ($\xi_0 \sim \xi_N$) return to location "Fall-Back" at t (where $t = t_b$). ■

5.5 Formal Description on Atomic Elaboration of Hybrid Automaton

In the following, we first propose the formal concept of *independence* between hybrid automata. We then propose a formal methodology on *elaborating* locations of design pattern hybrid automata with independent child hybrid automata. Finally, we prove following the proposed elaboration method, the resulted specific designs maintains the PTE safety rules guarantees.

Unless explicitly denoted, the rest of the chapter assumes every hybrid automaton to be time-block-free and non-zeno[1].

We now define *hybrid automata independence*.

[1] For the aforementioned design pattern hybrid automata A_{supvsr}, A_{initzr}, and $A_{\mathrm{ptcpnt},i}$, as long as Condition c1 $\sim$ c7 hold, they are time-block-free and non-zeno. Besides, time-block-free and non-zeno are well-known concepts in formal modeling, and most practical hybrid automata are time-block-free and non-zeno. Due to above reasons, we are not going to elaborate the definitions of these two concepts in this work.

DEFINITION 5.1 (HYBRID AUTOMATA INDEPENDENCE) *Given hybrid automata* $A = (\vec{x}(t), V, \mathsf{inv}, F, E, g, R, L, \mathsf{syn}, \Phi_0)$ *and* $A' = (\vec{x}'(t), V', \mathsf{inv}', F', E', g', R', L', \mathsf{syn}', \Phi'_0)$*, we say "*$A$ and A' are independent*" iff*

1. $\mathsf{elements}(\vec{x}(t)) \cap \mathsf{elements}(\vec{x}'(t)) = \varnothing$*;*
2. *and* $V \cap V' = \varnothing$*;*
3. *and* $L \cap L' = \varnothing$*.*

*Furthermore, we say "*a set of hybrid automata A_1, A_2, …, A_k are mutually independent*", iff* $\forall i, j \in \{1, 2, \ldots, k\}$ *and* $i \neq j$*,* A_i *and* A_j *are independent.*

We further define *simple* hybrid automaton.

DEFINITION 5.2 (SIMPLE HYBRID AUTOMATON) *A hybrid automaton* $A = (\vec{x}(t), V, \mathsf{inv}, F, E, g, R, L, \mathsf{syn}, \Phi_0)$ *is* simple *iff*

1. $\forall v_1, v_2 \in V, \mathsf{inv}(v_1) = \mathsf{inv}(v_2)$.
2. $\forall v \in \Phi_0|_V \cdot \forall \vec{s} \in \mathsf{inv}(v) \cdot (v, \vec{s}) \in \Phi_0$*, where* $\Phi_0|_V$ *means* Φ_0*'s projection on* V*.*
3. $\forall v \in \Phi_0|_V \cdot (v, \mathbf{0}) \in \Phi_0$*, where* $\mathbf{0}$ *is the zero data state vector.*
4. *A has no time-convergent paths, no timelock locations, and no zeno paths.*

Let us first describe the intuition on how to *elaborate* a given hybrid automaton at one location with one child hybrid automaton.

Atomic Elaboration of Hybrid Automaton (Intuition):

Given a hybrid automaton $A = (\vec{x}(t), V, \mathsf{inv}, F, E, g, R, L, \mathsf{syn}, \Phi_0)$, location $v \in V$, and a simple hybrid automaton $A' = (\vec{x}'(t), V', \mathsf{inv}', F', E', g', R', L', \mathsf{syn}', \Phi'_0)$ such that A and A' are independent, then we can create the "(atomic) elaboration of A at v with A'", i.e. a hybrid automaton $A'' = (\vec{x}''(t), V'', \mathsf{inv}'', F'', E'', g'', R'', L'', \mathsf{syn}'', \Phi''_0)$, according to the following intuitions.

1. Location v of hybrid automaton A is replaced by simple hybrid automaton A'.
2. All former ingress edges to v in A become ingress edges to A' (A''s initial locations to be more specific).
3. All former egress edges from v in A become egress edges from A'.
4. When in A', the data state variables $\vec{x}(t)$ of A maintain their continuous behavior as if they are in v.
5. When out of A', the data state variables $\vec{x}'(t)$ of A' remain unchanged (until return to A' again in the future).

The above intuitive methodology can be formalized as follows.

Formal Description on Atomic Elaboration of Hybrid Automaton:

The formal description assumes the following.

1. Given $\vec{x}^a = (x_1^a, x_2^a, \ldots, x_n^a) \in \mathbb{R}^n$ and $\vec{x}^b = (x_1^b, x_2^b, \ldots, x_m^b) \in \mathbb{R}^m$, $(\vec{x}^a, \vec{x}^b) \stackrel{\mathsf{def}}{=} (x_1^a, x_2^a, \ldots, x_n^a, x_1^b, x_2^b, \ldots, x_m^b) \in \mathbb{R}^{n+m}$.

2. Given an n-element tuple $X = (x_1, x_2, \ldots, x_n)$, we use $X|_i$ ($i \in \{1, 2, \ldots, n\}$) to denote X's ith element: x_i. We use $\mathsf{elements}(X) \stackrel{\mathsf{def}}{=} \{x_1, x_2, \ldots, x_n\}$ to denote the set of all elements of X.

Under the above assumptions, the formal description on how to carry out atomic elaboration of hybrid automaton runs as follows.

Given a hybrid automaton $A = (\vec{x}(t), V, \mathsf{inv}, F, E, g, R, L, \mathsf{syn}, \Phi_0)$, location $v \in V$, and a simple hybrid automaton $A' = (\vec{x}'(t), V', \mathsf{inv}', F', E', g', R', L', \mathsf{syn}', \Phi_0')$ such that A and A' are independent, then we can create the "atomic elaboration of A at v with A'", i.e. a hybrid automaton $A'' = (\vec{x}''(t), V'', \mathsf{inv}'', F'', E'', g'', R'', L'', \mathsf{syn}'', \Phi_0'')$, according to the following steps.

1. $\vec{x}''(t) \stackrel{\mathsf{def}}{=} (\vec{x}(t), \vec{x}'(t))$; denote A and A''s dimensions as respectively n and n', then $\vec{x}''(t) \in \mathbb{R}^{n+n'}$.

2. $V'' \stackrel{\mathsf{def}}{=} (V \cup V')\backslash\{v\}$.

3. $\forall u \in V\backslash\{v\}$, $\mathsf{inv}''(u) \stackrel{\mathsf{def}}{=} \mathsf{inv}(u) \times \mathsf{inv}'(v')$, where "$\times$" means Cartesian product, v' is an arbitrary location in V'; $\forall u \in V'$, $\mathsf{inv}''(u) \stackrel{\mathsf{def}}{=} \mathsf{inv}(v) \times \mathsf{inv}''(u)$.

4. $F'' \stackrel{\mathsf{def}}{=} \{f_u''|f_u'' : \mathbb{R}^{n+n'} \mapsto \mathbb{R}^{n+n'}, \forall u \in V''\}$ such that at location $u \in V''$, we have

4.1. if $u \in V \backslash \{v\}$, then

$$f''_u\big((x_1, x_2, \ldots, x_n, x'_1, x'_2, \ldots, x'_{n'})\big)|_i \stackrel{\text{def}}{=} \begin{cases} f_u\big((x_1, x_2, \ldots, x_n)\big)|_i & \\ & \text{(when } 1 \leq i \leq n\text{)}, \\ 0 & \text{(otherwise);} \end{cases}$$

4.2. if $u \in V'$, then

$$f''_u\big((x_1, x_2, \ldots, x_n, x'_1, x'_2, \ldots, x'_{n'})\big)|_i \stackrel{\text{def}}{=} \begin{cases} f_v\big((x_1, x_2, \ldots, x_n)\big)|_i & \\ & \text{(when } 1 \leq i \leq n\text{)}, \\ f'_u\big((x'_1, x'_2, \ldots, x'_{n'})\big)|_{i-n} & \text{(otherwise).} \end{cases}$$

5. $L'' \stackrel{\text{def}}{=} L \cup L'$.

6. E'', g'', R'', syn'' are created according to the following process.

6.1. Initially $E'' = \varnothing$.

6.2. For each $e = (v_1, v_2) \in E$, where $v_1, v_2 \neq v$ (hence $v_1, v_2 \in V''$), we add edge $e'' = (v_1, v_2)$ into E''. Furthermore, we define $g''(e'') \stackrel{\text{def}}{=} g(e) \times \mathbb{R}^{n'}$;

$$r''_{e''}\big((s_1, s_2, \ldots, s_n, s'_1, s'_2, \ldots, s'_{n'})\big) \stackrel{\text{def}}{=} r_e\big((s_1, s_2, \ldots, s_n)\big) \times \{(s'_1, s'_2, \ldots, s'_{n'})\}, \quad \forall (s_1, s_2, \ldots, s_n, s'_1, s'_2, \ldots, s'_{n'}) \in \mathsf{inv}''(v_1);$$

and $\mathsf{syn}''(e'') \stackrel{\mathsf{def}}{=} \mathsf{syn}(e)$.

6.3. For each $e = (v_1, v) \in E$, where $v_1 \neq v$ (hence $v_1 \in V''$), we add for each $v' \in \Phi'_0|_{V'}$ (i.e. Φ'_0's projection on V') an edge $e'' = (v_1, v')$ into E''. Furthermore, we define $g''(e'') \stackrel{\mathsf{def}}{=} g(e) \times \mathbb{R}^{n'}$;

$$\begin{aligned} & r''_{e''}\big((s_1, s_2, \ldots, s_n, s'_1, s'_2, \ldots, s'_{n'})\big) \\ \stackrel{\mathsf{def}}{=} \; & r_e\big((s_1, s_2, \ldots, s_n)\big) \times \{(s'_1, s'_2, \ldots, s'_{n'})\}, \\ & \forall (s_1, s_2, \ldots, s_n, s'_1, s'_2, \ldots, s'_{n'}) \in \mathsf{inv}''(v_1); \end{aligned}$$

and $\mathsf{syn}''(e'') \stackrel{\mathsf{def}}{=} \mathsf{syn}(e)$.

6.4. For each $e = (v, v_2) \in E$, where $v_2 \neq v$ (hence $v_2 \in V''$), we add for each $v' \in V'$ an edge $e'' = (v', v_2)$ into E''. Furthermore, we define $g''(e'') \stackrel{\mathsf{def}}{=} g(e) \times \mathbb{R}^{n'}$;

$$\begin{aligned} & r''_{e''}\big((s_1, s_2, \ldots, s_n, s'_1, s'_2, \ldots, s'_{n'})\big) \\ \stackrel{\mathsf{def}}{=} \; & r_e\big((s_1, s_2, \ldots, s_n)\big) \times \{(s'_1, s'_2, \ldots, s'_{n'})\}, \\ & \forall (s_1, s_2, \ldots, s_n, s'_1, s'_2, \ldots, s'_{n'}) \in \mathsf{inv}''(v'); \end{aligned}$$

and $\mathsf{syn}''(e'') \stackrel{\mathsf{def}}{=} \mathsf{syn}(e)$.

6.5. For each $e = (v, v) \in E$, we add for each $v' \in V'$ an edge $e'' = (v', v')$ into E''. Furthermore, we define $g''(e'') \stackrel{\mathsf{def}}{=} g(e) \times \mathbb{R}^{n'}$;

$$\begin{aligned} & r''_{e''}\big((s_1, s_2, \ldots, s_n, s'_1, s'_2, \ldots, s'_{n'})\big) \\ \stackrel{\mathsf{def}}{=} \; & r_e\big((s_1, s_2, \ldots, s_n)\big) \times \{(s'_1, s'_2, \ldots, s'_{n'})\}, \\ & \forall (s_1, s_2, \ldots, s_n, s'_1, s'_2, \ldots, s'_{n'}) \in \mathsf{inv}''(v'); \end{aligned}$$

and $\mathsf{syn}''(e'') \stackrel{\mathsf{def}}{=} \mathsf{syn}(e)$.

6.6. For each $e' = (v_1, v_2) \in E'$ (hence $v_1, v_2 \in V''$), we add an edge $e'' = (v_1, v_2)$ into E''. Furthermore, we define $g''(e'') \stackrel{\mathsf{def}}{=} \mathbb{R}^n \times g'(e')$;

$$\begin{aligned} & r''_{e''}\big((s_1, s_2, \ldots, s_n, s'_1, s'_2, \ldots, s'_{n'})\big) \\ \stackrel{\mathsf{def}}{=} \; & \{(s_1, s_2, \ldots, s_n)\} \times r'_{e'}\big((s'_1, s'_2, \ldots, s'_{n'})\big); \\ & \forall (s_1, s_2, \ldots, s_n, s'_1, s'_2, \ldots, s'_{n'}) \in \mathsf{inv}''(v_1); \end{aligned}$$

and $\mathsf{syn}''(e'') \stackrel{\mathsf{def}}{=} \mathsf{syn}'(e)$.

7. Φ''_0 is created according to the following process.

7.1. Initially $\Phi''_0 = \varnothing$.

7.2. For each $(v_1, \vec{s}) \in \Phi_0$, where $v_1 \neq v$, we add $(v_1, (\vec{s}, \underbrace{0, 0, \ldots, 0}_{n' \text{ zeros}}))$ into Φ''_0.

7.3. For each $(v, \vec{s}) \in \Phi_0$, we add for each $(v', \vec{s}') \in \Phi'_0$ a state value $(v', (\vec{s}, \vec{s}'))$ into Φ''_0.

8. For PTE CPS, there is the issue of partitioning V'' into safe-locations V''^{safe} and risky-locations V''^{risky}. In case $v \in V^{\mathrm{safe}}$, $V''^{\mathrm{safe}} \stackrel{\mathsf{def}}{=} V' \cup (V^{\mathrm{safe}} \backslash \{v\})$ and $V''^{\mathrm{risky}} \stackrel{\mathsf{def}}{=} V'' \backslash V''^{\mathrm{safe}}$; otherwise, $V''^{\mathrm{risky}} \stackrel{\mathsf{def}}{=} V' \cup (V^{\mathrm{risky}} \backslash \{v\})$ and $V''^{\mathrm{safe}} \stackrel{\mathsf{def}}{=} V'' \backslash V''^{\mathrm{risky}}$.

We denote A'', the atomic elaboration of A at v with A', as

$$A'' = \mathcal{E}(A, v, A').$$

With atomic elaboration at hand, we can go further.

Given hybrid automaton A, k distinct locations $v_1 \sim v_k \in V$ (where V is A's location set), and k simple hybrid automata $A_1 \sim A_k$ such that A, A_1, ..., A_k are mutually independent, then we can carry out "(parallel) elaboration of A at v_1, v_2, ..., v_k with A_1, A_2, ..., A_k", denoted as

$$\begin{aligned} & \mathcal{E}(A, (v_1, v_2, \ldots, v_k), (A_1, A_2, \ldots, A_k)) \\ \stackrel{\text{def}}{=} \quad & \underbrace{\mathcal{E}(\ldots \mathcal{E}(\mathcal{E}(}_{\text{repeat } k \text{ times}} A, v_1, A_1), v_2, A_2) \ldots), v_k, A_k). \end{aligned}$$

Denote $A' = \mathcal{E}(A, (v_1, v_2, \ldots, v_k), (A_1, A_2, \ldots, A_k))$, we also say "*$A'$ elaborates A at v_1, v_2, ..., v_k with A_1, A_2, ..., A_k respectively*".

Intuitively, parallel elaboration of A at v_1, v_2, ..., v_k with A_1, A_2, ..., A_k can be implemented by elaborating A at v_1 with A_1, v_2 with A_2, so on and so forth, until v_k with A_k.

If a specific wireless CPS design, described by hybrid system $\mathcal{H}'$, has its member hybrid automata respectively elaborating the Supervisor, Initializer, and Participant design pattern hybrid automata (i.e. A_{supvsr}, A_{initzr}, and $A_{\text{ptcpnt},i}$), then the design $\mathcal{H}'$ maintains the properties of our design pattern and guarantee of PTE safety rules. Formally, this is expressed in the form of the following theorem.

THEOREM 5.1 (DESIGN PATTERN COMPLIANCE) *Given a hybrid system $\mathcal{H}'$ consisting of entities $\xi'_0, \xi'_1, \ldots, \xi'_N$, which respectively corresponds to hybrid automata of $A'_0, A'_1, \ldots, A'_N$. If the following conditions are satisfied:*

1. *There are distinct locations v^0_1, v^0_2, ..., $v^0_{k_0} \in V_{\text{supvsr}}$, and simple hybrid automata*

A_1^0, A_2^0, ..., $A_{k_0}^0$, *such that* A_{supvsr} *and* A_j^0 *(*$j = 1 \sim k_0$*) are independent, and* A_0' *elaborates* A_{supvsr} *at* v_1^0, v_2^0, ..., $v_{k_0}^0$ *with* A_1^0, A_2^0, ..., $A_{k_0}^0$ *respectively;*

2. *For each* $i \in \{1, 2, \ldots, N-1\}$, *there are distinct locations* v_1^i, v_2^i, ..., $v_{k_i}^i \in V_{\text{ptcpnt},i}$, *and simple hybrid automata* A_1^i, A_2^i, ..., $A_{k_i}^i$, *such that* $A_{\text{ptcpnt},i}$ *and* A_j^i *(*$j = 1 \sim k_i$*) are independent, and* A_i' *elaborates* $A_{\text{ptcpnt},i}^i$ *at* v_1^i, v_2^i, ..., $v_{k_i}^i$ *with* A_1^i, A_2^i, ..., $A_{k_i}^i$ *respectively;*

3. *There are distinct locations* v_1^N, v_2^N, ..., $v_{k_N}^N \in V_{\text{initzr}}$, *and simple hybrid automata* A_1^N, A_2^N, ..., $A_{k_N}^N$, *such that* A_{initzr} *and* A_j^N *(*$j = 1 \sim k_N$*) are independent, and* A_N' *elaborates* A_{initzr} *at* v_1^N, v_2^N, ..., $v_{k_N}^N$ *with* A_1^N, A_2^N, ..., $A_{k_N}^N$ *respectively;*

4. *Hybrid automata* A_j^i *are mutually independent, where* $i = 0, 1, \ldots, N$, $j = 1, 2, \ldots, k_i$*;*

5. *Condition c1* $\sim$ *c7 of Theorem 2.1 sustain;*

where V_{supvsr}, $V_{\text{ptcpnt},i}$, *and* V_{initzr} *are respectively* A_{supvsr}, $A_{\text{ptcpnt},i}$, *and* A_{initzr}*'s location sets, then* $\mathcal{H}'$ *satisfies PTE safety rules and liveness described in Theorem 2.1 Claim 1 and 2.*

Proof: If not, there must be an execution trace $\phi'(t)$ (see [A$^+$96] for the rigorous definition of "execution trace", aka "trajectory" of a hybrid system) of $\mathcal{H}'$ that violates PTE safety rules (liveness). According to the methodology we elaborate hybrid automata, $\phi'(t)$ corresponds to an execution trace $\phi(t)$ of $\mathcal{H}$ (the hybrid system of A_{supvsr}, $A_{\text{ptcpnt},i}$ ($i = 1, 2, \ldots, N-1$), and A_{initzr}) that also violates PTE safety rules (liveness). This contradicts Theorem 2.1. ■

5.6 Detailed Design of Leasing Based Approach for Case Studies

We start our design of laser tracheotomy wireless CPS per proposed leasing-based design approach.

First, we see the wireless laser tracheotomy CPS consists of three entities (i.e. $N = 2$): the laser tracheotomy supervisor (together with the SpO_2 sensor wired to it) plays the role of Supervisor, hence entity ξ_0; the (surgeon operated) laser-scalpel plays the role of Initializer, hence entity ξ_2; and the ventilator plays the role of Participant 1, hence entity ξ_1.

Next, we design the hybrid automata for the laser tracheotomy supervisor, laser-scalpel, and ventilator by respectively elaborating A_{supvsr}, A_{initzr}, and $A_{\text{ptcpnt},1}$.

Take the ventilator detailed design for example. The detailed design of a stand-alone ventilator has already been described by the simple hybrid automaton A'_{vent} of Fig. 2.2. The stand-alone design of A'_{vent}, however, is not aware of the communications/collaborations with supervisor and laser-scalpel; hence cannot guarantee PTE safety rules. In order to guarantee PTE safety rules, we revise the ventilator design by elaborating the Participant Design Pattern hybrid automaton $A_{\text{ptcpnt},i}$ (see Section 2.4.1-Participant; also see Fig. 5.4 for the diagram of the hybrid automaton) at location "Fall-Back" with A'_{vent}, using the elaboration method described in Section 2.4.3.

The Initializer hybrid automaton A_{initzr} and Supervisor hybrid automaton A_{supvsr} do not need to be further elaborated. They can be directly used to describe the behavior of laser-scalpel and laser tracheotomy supervisor respectively.

The resulted detailed designs for the wireless laser tracheotomy entities are shown in Fig. 5.5, 5.6, and 5.7 respectively. Some data state variable names and/or locations names in the corresponding design patterns are modified to better reflect their meanings in laser tracheotomy.

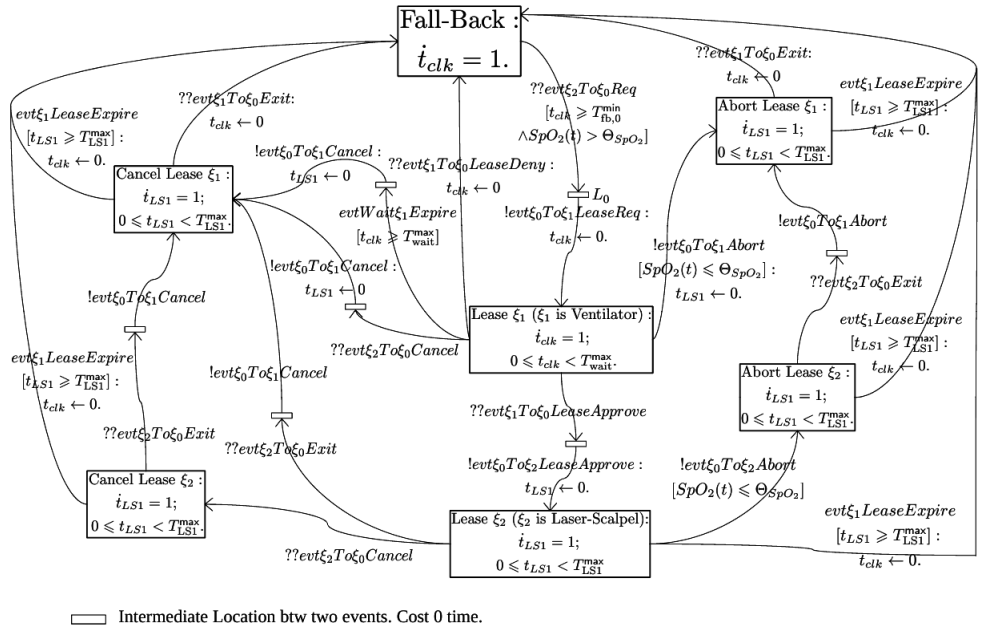


Figure 5.5. Laser Tracheotomy Supervisor Detailed Design. Note entity ξ_1 refers to the ventilator, and ξ_2 refers to the laser-scalpel.

Via the same approach, we derive the detailed designs for the wireless IP remote monitoring entities, which are shown in Fig. 5.8, 5.9, and 5.10 respectively. Some data state variable names and/or locations names in the corresponding design patterns are modified to better reflect their meanings in IP remote monitoring.

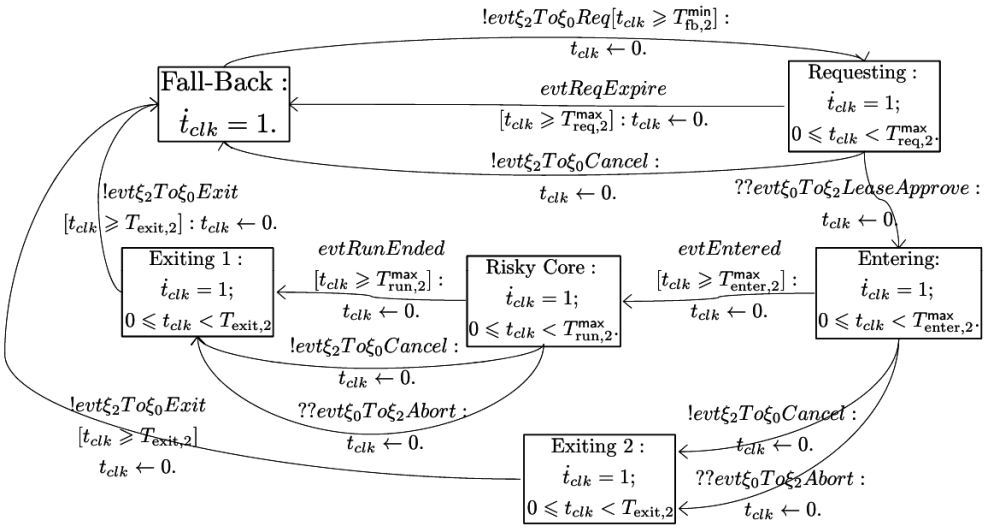


Figure 5.6. Laser Tracheotomy Laser-Scalpel Detailed Design. Note the laser-scalpel emits and only emits laser when dwelling in location "Risky Core".

5.7 Example Scenarios where Leasing Protects PTE Safety Rules

Let us further consider a number of typical scenarios to get better intuitions on how leasing approach works in the laser tracheotomy case study.

One scenario is that after the ventilator is paused and the laser-scalpel is emitting, the surgeon may forget to cancel laser emission until too late (e.g. T_{off} is set to 1 hour). Without leasing, only the abort request from the supervisor can stop laser emission and resume ventilator before it is too late. However, this requires a sequence of correct send/receive of events through wireless: $evt\xi_0To\xi_2Abort$, followed by $evt\xi_2To\xi_0Exit$, and followed by $evt\xi_0To\xi_1Abort$. Losing any one of these events at the receiver end will cause PTE safety rules violation. For example, losing $evt\xi_2To\xi_0Exit$, the supervi-

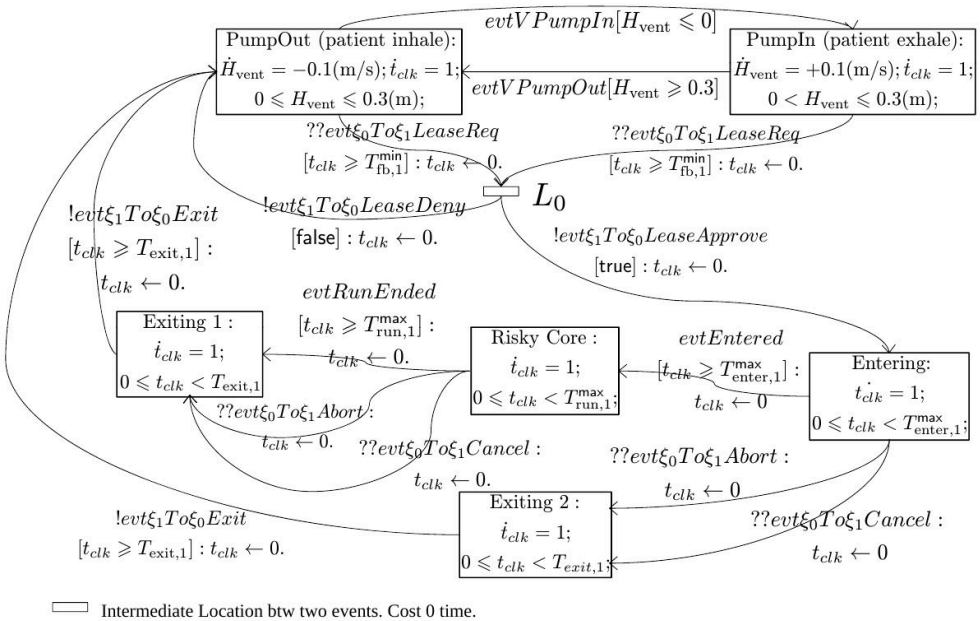


Figure 5.7. Laser Tracheotomy Ventilator Detailed Design, by elaborating Participant Design Pattern.

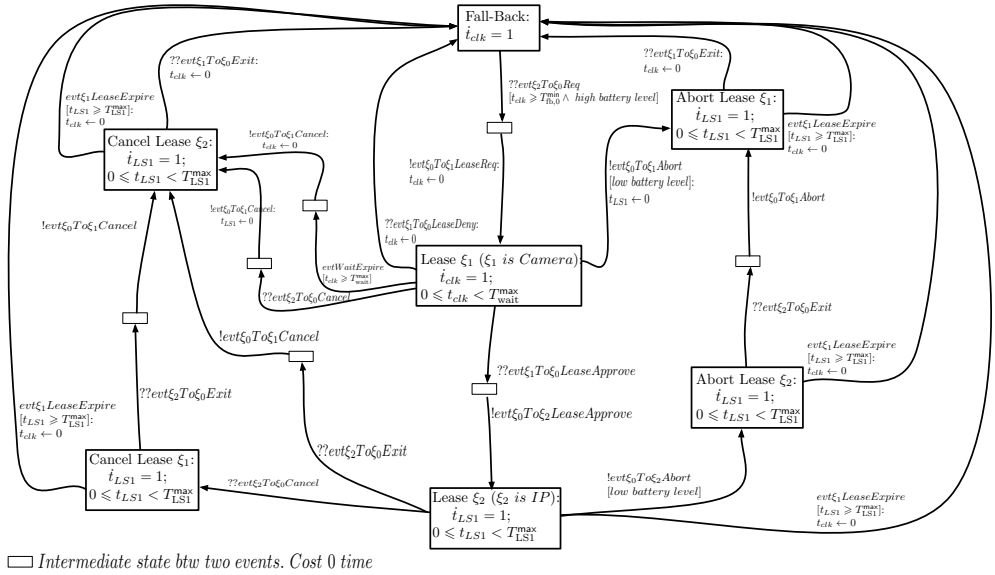


Figure 5.8. IP Remote Monitoring Supervisor Detailed Design. Note entity ξ_1 refers to the camera, and ξ_2 refers to the IP.

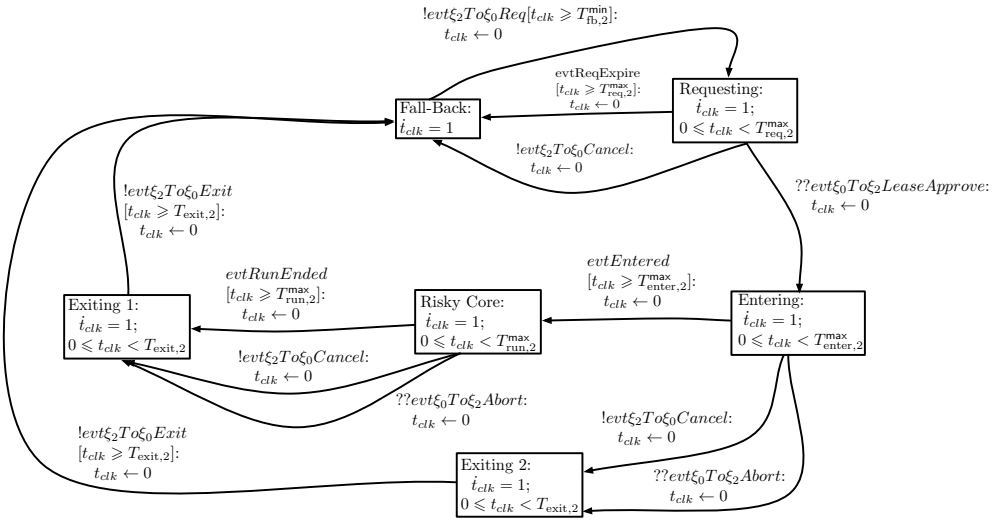


Figure 5.9. IP Remote Monitoring IP Detailed Design. Note the IP conducts random walk when and only when dwelling in location "Risky Core".

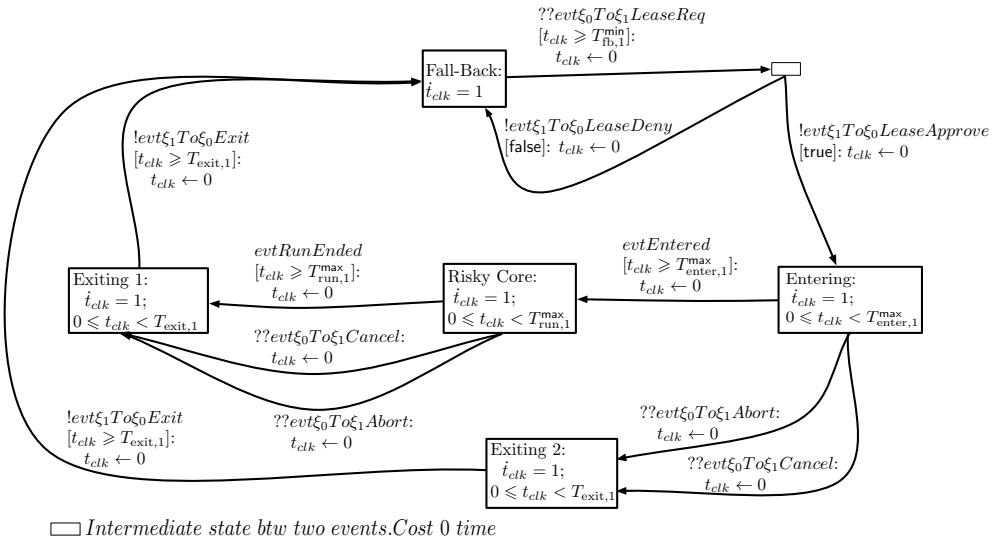


Figure 5.10. IP Remote Monitoring Camera Detailed Design.

sor may think the laser-scalpel is stuck and cannot stop laser emission, hence ventilator shall keep pausing.

With leasing, the laser emission terminates within the lease $T^{\mathsf{max}}_{\mathrm{run},2} = 20$(s) with or without surgeon's request to cancel; and the ventilator resumes within the lease $T^{\mathsf{max}}_{\mathrm{run},1} = 35$(s) with or without supervisor's requests. Hence PTE safety rules are protected.

Similar analysis applies to the scenario that the surgeon remembers to cancel laser emission, but his/her cancelling request (i.e. $evt\xi_2To\xi_0Cancel$) is not received at the supervisor. Without lease, the ventilator may keep pausing till for too long; with lease, the ventilator will keep pausing for $T^{\mathsf{max}}_{\mathrm{run},1} = 35$(s) at the most, hence cannot suffocate the patient.

A third scenario involves the parameter configuration constraints. Suppose we set $T^{\mathsf{max}}_{\mathrm{enter},2} = T^{\mathsf{max}}_{\mathrm{enter},1} = 0$(s) (or any other value so that $T^{\mathsf{max}}_{\mathrm{enter},2} = T^{\mathsf{max}}_{\mathrm{enter},1}$), then because $T^{\mathsf{min}}_{\mathrm{risky}:1\to2} = 3\text{(s)} > 0$, Condition c5 of Theorem 2.1 is violated. Under such design,

immediately after the ventilator is paused, the laser-scalpel can emit laser, violating the PTE requirement of $T^{\mathsf{min}}_{\mathrm{risky}:1\to 2} = 3$(s): that the laser-scalpel must wait for another 3(s) after the ventilator pauses, and then can it emit laser.

In summary, if we follow the proposed lease based design approach, Theorem 2.1 and 5.1 can guarantee PTE safety rules.

5.8 Detailed Design of Polling based Approach for Case Studies

The detailed design state diagrams of laser tracheotomy and IP remote monitoring per polling based approach [K$^+$10] are shown in Fig 5.11 $\sim$ 5.16 respectively.

In these figures, $CurrentTime$ refers to the wall clock time, $T_{period} = 50$(ms) is the polling period. For laser tracheotomy (see Fig. 5.11, 5.12, and 5.13), $T^{\xi_1}_{length} = 44$(s), $T^{\xi_1}_{Entering} = 3$(s), $T^{\xi_2}_{length} = 31.5$(s), $T^{\xi_2}_{Entering} = 10$(s), and $T^{\xi_2}_{Exiting} = 1.5$(s). For IP remote monitoring (see Fig. 5.14, 5.15, and 5.16), $T^{\xi_1}_{length} = 42$(s), $T^{\xi_1}_{Entering} = 1$(s), $T^{\xi_2}_{length} = 25.5$(s), $T^{\xi_2}_{Entering} = 3$(s), and $T^{\xi_2}_{Exiting} = 2.5$(s).

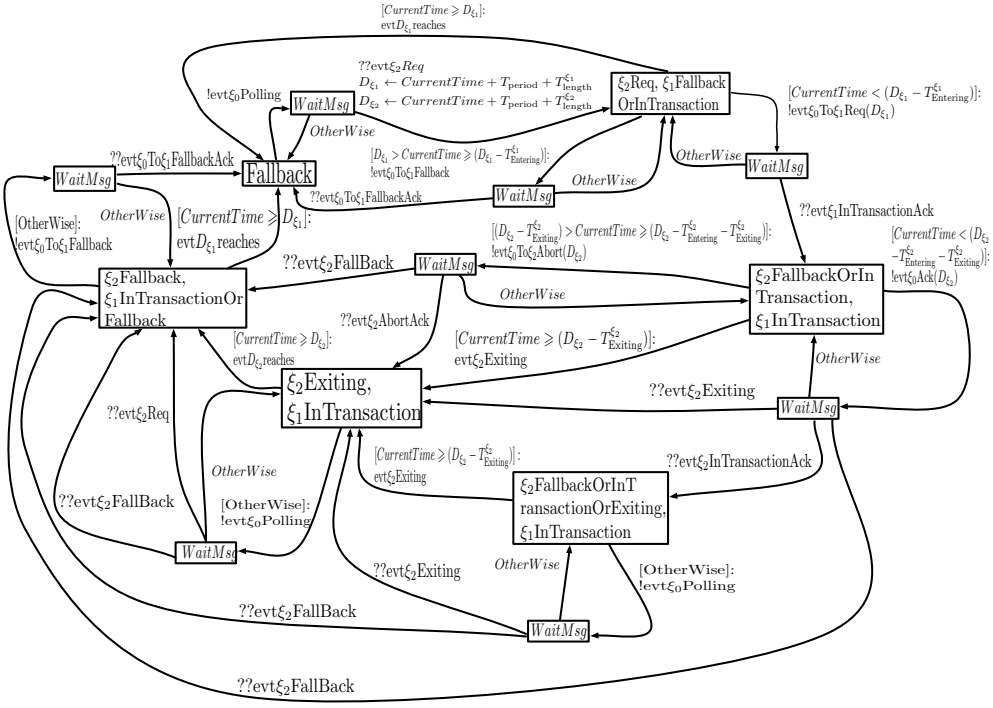


Figure 5.11. Laser Tracheotomy Supervisor (aka Entity ξ_0) Detailed Design, per Kim et al [K$^+$10]'s Polling-Based Approach

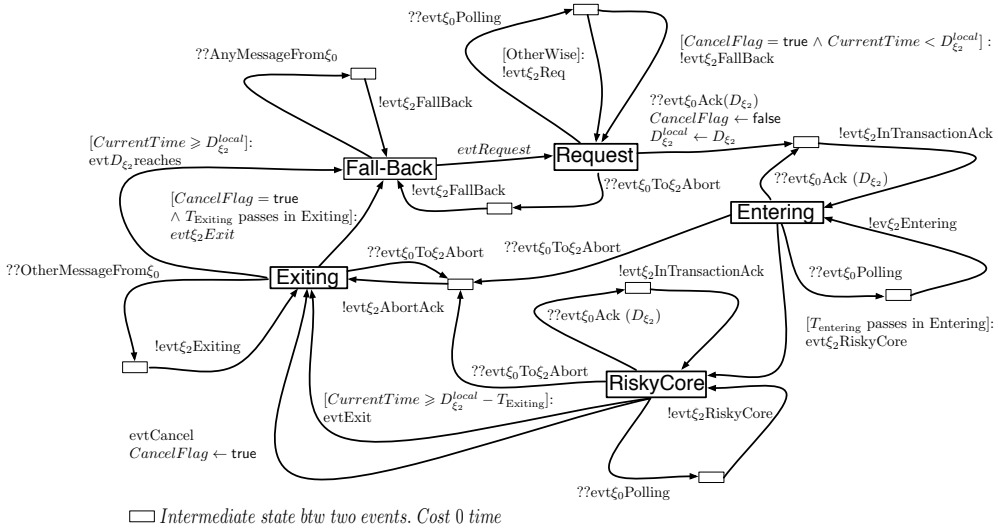


Figure 5.12. Laser Tracheotomy Laser Scalpel (the Initializer, aka Entity ξ_2) Detailed Design, per Kim et al [K$^+$10]'s Polling-Based Approach

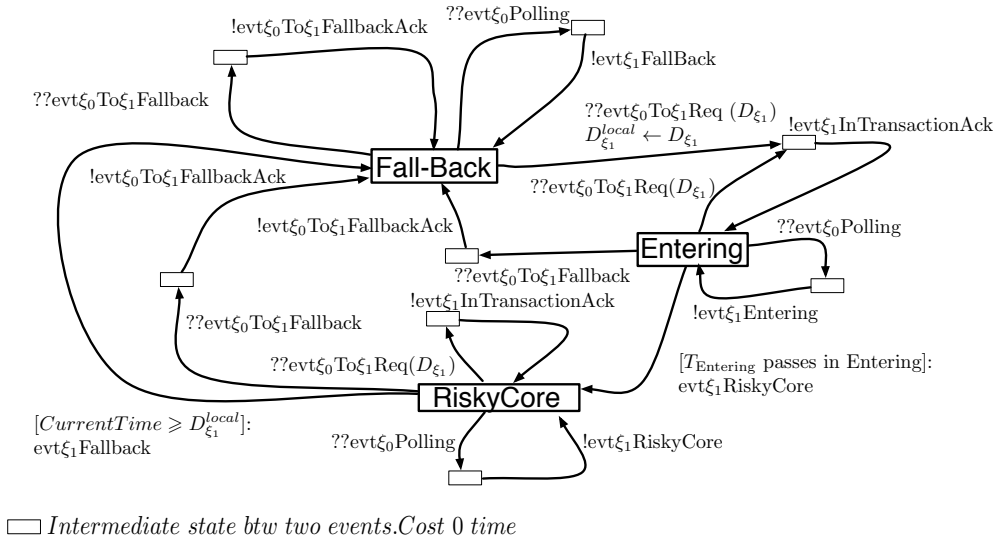


Figure 5.13. Laser Tracheotomy Ventilator (the Participant, aka Entity ξ_1) Detailed Design, per Kim et al [K$^+$10]'s Polling-Based Approach

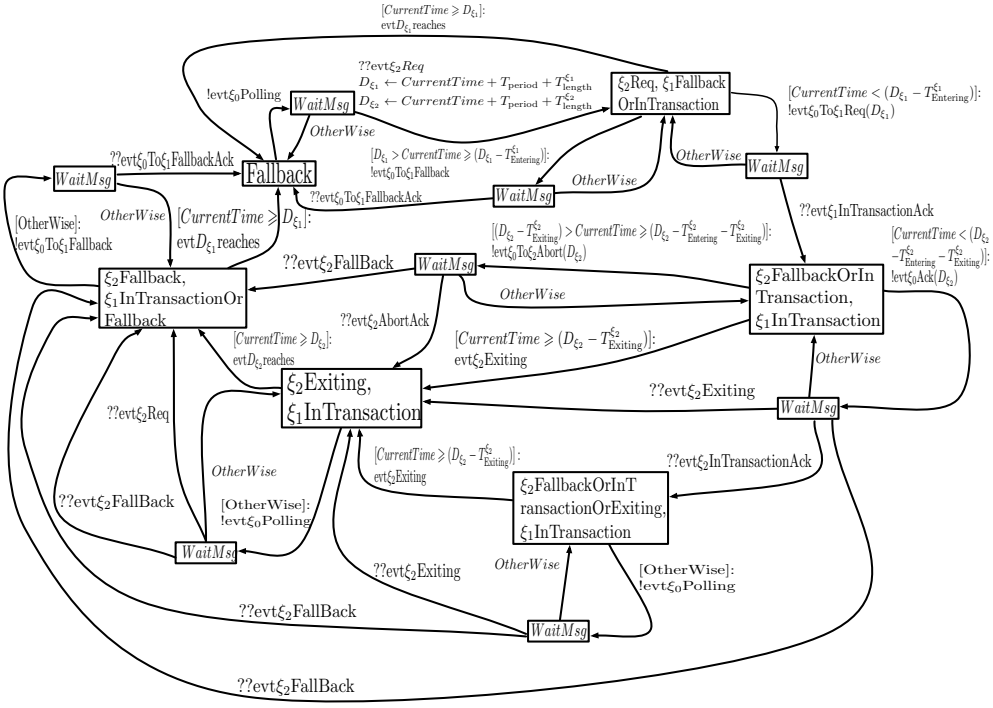


Figure 5.14. IP Remote Monitoring Supervisor (aka Entity ξ_0) Detailed Design, per Kim et al [K$^+$10]'s Polling-Based Approach

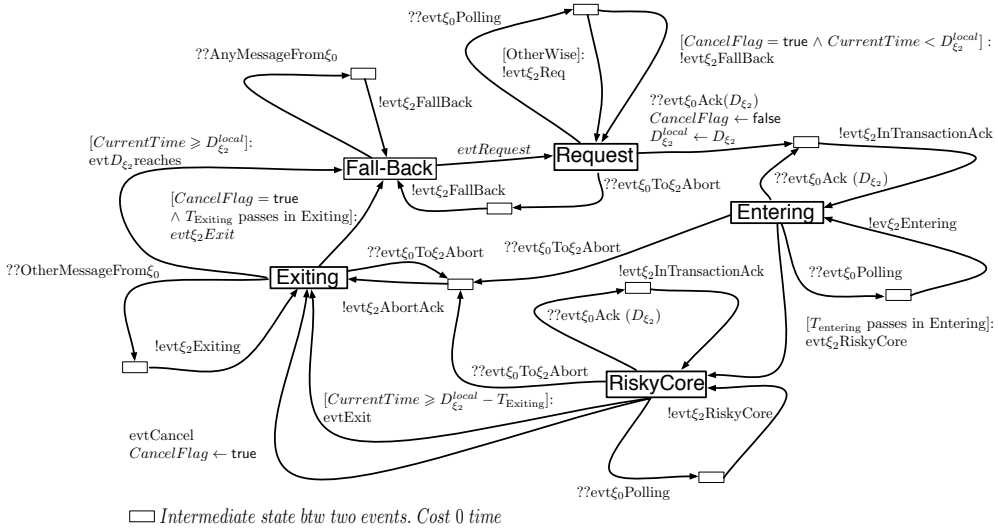


Figure 5.15. IP Remote Monitoring IP (the Initializer, aka Entity ξ_2) Detailed Design, per Kim et al [K$^+$10]'s Polling-Based Approach

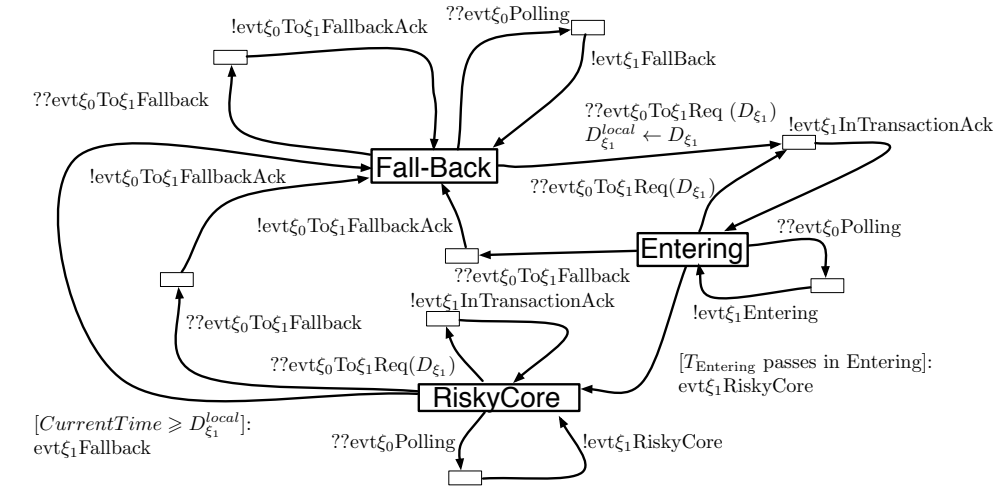


Figure 5.16. IP Remote Monitoring Camera (the Participant, aka Entity ξ_1) Detailed Design, per Kim et al [K$^+$10]'s Polling-Based Approach

5.9 Shortest Distance from a Ball to a Concentric Ball Complement

$\forall X, Y \in \mathbb{R}_{n\times 1}$, denote $\mathsf{dis}(X,Y) \stackrel{\mathsf{def}}{=} ||X-Y||_2 = \sqrt{(X-Y)^{\mathsf{T}}(X-Y)}$. We have the following lemma:

LEMMA 5.4 *Given $\Gamma \geqslant \gamma > 0$, then $\forall X, Y \in \mathbb{R}_{n\times 1}$ s.t. $X^{\mathsf{T}}X \leqslant \gamma^2$ and $Y^{\mathsf{T}}Y > \Gamma^2$, we have $\mathsf{dis}(X,Y) > \Gamma - \gamma$.*

Proof: Define $f_Y(X) \stackrel{\mathsf{def}}{=} (X-Y)^{\mathsf{T}}(X-Y)$, let us first solve the following optimization problem:

$$\begin{aligned} \min_X \quad & f_Y(X) \\ \text{s.t.} \quad & X^{\mathsf{T}}X \leqslant \gamma^2. \end{aligned}$$

For this problem, we have its Lagrangian $L(X,\nu) = ||X-Y||_2^2 + \nu(||X||_2^2 - \gamma^2)$. Using the Karush-Kuhn-Tucker(KKT) conditions, we have

$$||X^*||_2 - \gamma \leqslant 0 \tag{5.13}$$

$$\nu^* \geqslant 0$$

$$\nu^*(||X^*||_2 - \gamma) = 0 \tag{5.14}$$

$$(1+\nu^*)X^* - Y = 0 \tag{5.15}$$

Substituting X^* from Eq. (5.15) into Eq. (5.14), we have

$$\nu^*(||X^*||_2 - \gamma) = \frac{\nu^*}{1+\nu^*}(||Y||_2 - (1+\nu^*)\gamma) = 0 \tag{5.16}$$

As we know $Y^{\mathsf{T}}Y > \Gamma^2$ and $\Gamma \geqslant \gamma > 0$, then we have $||Y||_2 > \Gamma \geqslant \gamma > 0$. From Eq. (5.16), we know either $\nu^* = 0$ or $(||Y||_2 - (1+\nu^*)\gamma) = 0$. If $\nu^* = 0$, we have $X^* = Y$ from Eq. (5.15), and $||Y||_2 = ||X^*||_2 \leqslant \gamma$ from Eq. (5.13), which contradicts the fact that $||Y||_2 > \gamma$. Thus, we have

$$||Y||_2 - (1+\nu^*)\gamma = 0 \Rightarrow 1 + \nu^* = \frac{||Y||_2}{\gamma}.$$

Substituting $(1+\nu^*) = ||Y||_2/\gamma$ into Eq. (5.15), we derive

$$X^* = \frac{\gamma}{||Y||_2} Y$$

Then, we have

$$f_Y(X)^* = ||\frac{\gamma}{||Y||_2} Y - Y||_2^2 = (||Y||_2 - \gamma)^2.$$

Here Y is a given parameter to the optimization problem. As $||Y||_2 > \Gamma \geqslant \gamma > 0$, we have $f_Y(X)^* = (||Y||_2 - \gamma)^2 > (\Gamma - \gamma)^2$. That is, $\forall X, Y \in \mathbb{R}_{n\times 1}$, if $X^{\mathsf{T}}X \leqslant \gamma^2$, $Y^{\mathsf{T}}Y > \Gamma^2$, and $\Gamma \geqslant \gamma > 0$, $\mathsf{dis}(X,Y) = \sqrt{f_Y(X)} \geqslant \sqrt{f_Y(X)^*} > \Gamma - \gamma$. ■

The idea of Lemma 5.4 is illustrated by Fig. 5.17.

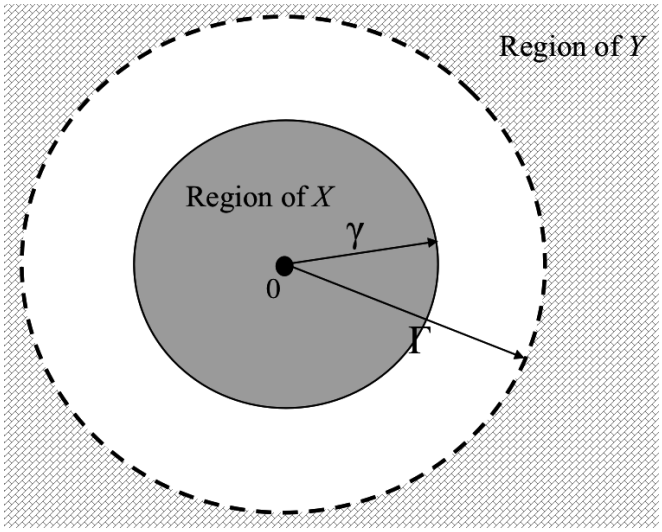


Figure 5.17. Minimal distance from a ball to a concentric ball complement

5.10 Proof of Proposition 3.2

$$
\begin{aligned}
& \frac{\mathrm{d}\, V(X(t), O_{\mathsf{ref}}(t))}{\mathrm{d}\, t} \quad \text{(where } O_{\mathsf{ref}}(t) \equiv O'_{\mathsf{ref}}(t_0)\text{)} \\
= \;& \dot{X}^{\mathsf{T}}\mathbf{P}(X(t) - O'_{\mathsf{ref}}(t_0)) \\
& +(X(t) - O'_{\mathsf{ref}}(t_0))^{\mathsf{T}}\mathbf{P}\dot{X} \qquad \text{(see Eq. (3.7))} \\
= \;& (\mathbf{F}(X(t) - O'_{\mathsf{ref}}(t_0)))^{\mathsf{T}}\mathbf{P}(X(t) - O'_{\mathsf{ref}}(t_0)) + \\
& (X(t) - O'_{\mathsf{ref}}(t_0))^{\mathsf{T}}\mathbf{P}\mathbf{F}(X(t) - O'_{\mathsf{ref}}(t_0)) \text{ (see Eq. (3.3))} \\
= \;& (X(t) - O'_{\mathsf{ref}}(t_0))^{\mathsf{T}}(\mathbf{F}^{\mathsf{T}}\mathbf{P} + \mathbf{P}\mathbf{F})(X(t) - O'_{\mathsf{ref}}(t_0)) \\
= \;& -(X(t) - O'_{\mathsf{ref}}(t_0))^{\mathsf{T}}\mathbf{I}(X(t) - O'_{\mathsf{ref}}(t_0)) \quad \text{(see Eq. (3.6))} \\
= \;& -(X(t) - O'_{\mathsf{ref}}(t_0))^{\mathsf{T}}(X(t) - O'_{\mathsf{ref}}(t_0)) \leqslant 0. \qquad \blacksquare
\end{aligned}
$$

5.11 Meaning of p

PROPOSITION 5.2 (RISK OF TRAJECTORY) *Given cross-domain noise RV N, suppose during $[t_0, +\infty)$, a 2L-CCPS undergoes k ($k \geqslant 1$) reference point update events, respectively happened at $t_0 < t_1 < \ldots < t_{k-1}$. Let X_i ($i = 0, \ldots, k-1$) denote the plant state right before the ith reference point update event. Let R_i denote the reachability RV for X_i under N, and $p_i = \mathsf{Pr}(R_i = 1)$. Let ϖ denote the probability that the trajectory of $X(t)$ ($t \in [t_0, +\infty)$) never reaches $\bar{A}$ (i.e. the 2L-CCPS never encounters plant fault). Then $\varpi \geqslant \Pi_{i=1}^{k-1}(1 - p_i)$.*

Proof: Starting from X_i, what happens during $[t_i, t_{i+1})$ ($i = 0, \ldots, k-1$, where $t_k \stackrel{\mathsf{def}}{=} +\infty$) is exactly what happens to an elementary trial starting from X_i during $[0, t_{i+1} - t_i)$ (suppose the elementary trial starts from time 0). Therefore, the probability of not reaching $\bar{A}$ during $[t_i, t_{i+1})$ is no less than $(1 - p_i)$. As per Eq. (3.3), $X(t)$ is continuous on $[t_0, +\infty)$, therefore, $\varpi \geqslant \Pi_{i=0}^{k-1}(1 - p_i)$. ■

Particularly, if p_is are upper bounded by p^{max}, then $\varpi \geqslant (1 - p^{\mathsf{max}})^k$. In the extreme case, if $p^{\mathsf{max}} = 0$, then $\varpi = 1$. That is, the control CPS has 0 probability of encountering plant fault.

Bibliography

[A+93] Rajeev Alur et al. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. *Hybrid Systems*, 1993.

[A+96] Rajeev Alur et al. Automatic symbolic verification of embedded systems. *IEEE Trans. on Software Engineering*, 22(3):181–201, 1996.

[A+05] Siddhartha Annapureddy et al. Shark: scaling file servers via cooperative caching. *Proc. of the NSDI'05*, pages 129–142, 2005.

[A+10] Atul Adya et al. Centrifuge: Integrated lease management and partitioning for cloud services. *Proc. of the NSDI'10*, 2010.

[A+12] Manu Augustine et al. Cognitive map-based system modeling for identifying interaction failure modes. *Res. Eng. Design*, 23:105-124, 2012.

[AWM07] Marian Anghel, Kenneth A. Werley, and Adilson E. Motter. Stochastic model for power grid dynamics. *Proc. of the 40th Intl. Conf. on System Sci.*, January 2007.

[B+04] Stephen Boyd et al. *Convex Optimization*. Cambridge Univ. Press, 2004.

[B+07] Elisa Gonzalez Boix et al. Context-aware leasing for mobile ad hoc networks. *3rd Workshop on OT4AmI co-located at ECOOP'07*, 2007.

[Bro91] William L. Brogan. *Modern Control Theory (3rd Ed.)*. Prentice Hall, 1991.

[Bub05] Zdzislaw Bubnicki. *Modern Control Theory*. Springer, 2005.

[CB13] Eduardo F. Camacho and Carlos Bordons. *Model Predictive Control in the Process Industry (Advances in Industrial Control)*. Springer, 2013.

[CWR06] Xin Chen, Haining Wang, and Shansi Ren. DNScup: Strong cache consistency protocol for dns. *Proc. of the ICDCS'06*, pages 40–48, 2006.

[D+11] S. Distefano et al. A compositional method for reliability analysis of workflows affected by multiple failure modes. *Proc. of CBSE*, June 2011.

[F+93] G.F. Franklin et al. Feedback contrl of dynamical systems. *Addison-Wesey*, Nov. 1993.

[F+04] B. Fisher et al. Seeing, hearing, and touching: Putting it all together. *ACM SIGGRAPH 2004 Course Notes*, 2004.

[G+89] C. G. Gray et al. Leases: An efficient fault-tolerant mechanism for distributed file cache consistency. *Proc. of ACM SOSP'89*, 1989.

[G+06] M. Gleancros et al. Exploiting perception in high fidelity virtual environments additional. *ACM SIGGRAPH 2006 Courses*, 2006.

[G+15a] Zhiwei Gao et al. A survey of fault diagnosis and fault-tolerant techniques part i: Fault diagnosis with model-based and signal-based approaches. *IEEE Trans. on Ind. Electronics*, (preprint) 2015.

[G+15b] Zhiwei Gao et al. A survey of fault diagnosis and fault-tolerant techniques part ii: Fault diagnosis with knowledge-based and hybrid/active approaches. *IEEE Trans. on Ind. Electronics*, (preprint) 2015.

[Gar90] David Garlan. The role of formal reusable frameworks. *SIGSOFT Softw. Eng. Notes*, 15(4):42–44, 1990.

[GGH+03] M. Gribaudo, M. Gribaudo, A. Horvth, A. Bobbio, E. Tronci, E. Ciancamerla, and M. Minichino. Fluid petri nets and hybrid model-checking: A comparative case study. *Reliability Engineering And System Safety*, 81:239–257, 2003.

[GPM09] Xiaocheng Ge, Richard F. Paige, and John A. McDermid. Probabilistic failure propagation and transformation analysis. *Proc. of the 28th Intl. Conf. on Computer Safety, Reliability, and Security*, pages 215–228, 2009.

[H+95] Thomas A. Henzinger et al. Hytech: The next generation. *Proc. of the RTSS'95*, pages 56–65, 1995.

[H+04] Martin Hiller et al. EPIC: Profiling the propagation and effect of data errors in software. *IEEE Trans. on Computers*, 53(5):1-19, 2004.

[HC10] Naira Hovakimyan and Chengyu Cao. *L1 Adaptive Control Theory: Guaranteed Robustness with Fast Adaptation*. SIAM, 2010.

[J+09] David C. Jensen et al. Design of an electrical power system using a functional failure and flow state logic reasoning methodology. *Proc. of Prognostics and Health Management Society Annual Conf.*, 2009.

[J+12] Townsend Jr. et al. Sabiston textbook of surgery: The biological basis of modern surgical practice. *19th ed, Elsevier Saunders*, 2012.

[JL11] Arshad Jhumka and Matthew Leeke. The early identification of detector locations in dependable software. *Proc. of IEEE Intl. Symp. on Software Reliability Engineering*, 2011.

[JS05] Andreas Johansson and Neeraj Suri. Error propagation profiling of operating systems. *Proc. of DSN*, pages 86–95, 2005.

[K+08] C. Kotselidis et al. Distm: A software transactional memory framework for clusters. *Proc. of the ICPP'08*, pages 51–58, 2008.

[K+10] C. Kim et al. A framework for the safe interoperability of medical devices in the presence of network failures. *ICCPS'10*, April 2010.

[Kha01] Hassan K. Khalil. *Nonlinear Systems (3rd Ed.)*. Prentice Hall, 2001.

[KL09] Daniel Krus and Katie Grantham Lough. Function-based failure propagation for conceptual design. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 23:409–426, November 2009.

[Kni12] John Knight. *Fundamentals of Dependable Computing for Software Engineers*. CRC Press, 2012.

[KNP02] W. Z. Kwiatkowska, G. Norman, and D. Parker. PRISM: Probabilistic cymbolic model checker. *TOOLS 2002*, 2324:200–204, 2002.

[L+08] Xue Liu et al. ORTEGA: An efficient and flexible online fault tolerance architecture for real-time control systems. *IEEE Trans. on Industrial Informatics*, 4(4):213-224, 2008.

[L+12] Tao Li et al. From offline toward real-time: A hybrid systems model checking and CPS co-design approach for medical device plug-and-play (MDPnP). *Proc. of the ICCPS'12*, pages 13–22, 2012.

[LL09] J. Lunze and F. Lamnabhi Lagarrigue. *Handbook of Hybrid Systems Control: Theory, Tools, Applications*. 2009.

[Ltd] Googol Tech. Ltd. *Linear Inverted Pendulum*. `http://www.googoltech.com`.

[M+10] Ming-Da Ma et al. Fault detection based on statistical multivariate analysis and microarray visualization. *IEEE Trans. on Industrial Informatics*, 6(1), 2010.

[Mik98] Tommi Mikkonen. Formalizing design patterns. *Proc. of ICSE*, 1998.

[non] Nonin 9843 oximeter/co2 detector. *http://www.nonin.com*.

[O+10] A. J. Oliner et al. Using correlated surprise to infer shared influence. *Proc. of DSN*, pages 191–200, 2010.

[OA11] Adam J. Oliner and Alex Aiken. Online detection of multi-component interactions in production systems. *Proc. of Dependable Systems and Networks (DSN)*, pages 49–60, 2011.

[P+15] T.-T. Pham et al. Reliability prediction for component-based software systems: Dealing with concurrent and propagating errors. *Science of Computer Programming*, 97:426–457, 2015.

[PD12] Thanh-Trung Pham and Xavier Defago. Reliability prediction for component-based systems: Incorporating error propagation analysis and different execution models. *Proc. of Intl. Conf. on Quality Software*, 2012.

[PRA11] *Probabilistic Risk Assessment Procedures Guide for NASA Managers and Practitioners (2nd Ed.).* NASA/SP-2011-3421, December 2011.

[Qua] Quanser. *http://www.quanser.com*.

[RLSS10] Ragunathan (Raj) Rajkumar, Insup Lee, Lui Sha, and John Stankovic. Cyber-physical systems: The next computing revolution. *Proceedings of the 47th Design Automation Conference*, pages 731–736, 2010.

[RPA11] Rasa Remenyte-Prescott and John D. Andrews. Modeling fault propagation in phased mission systems using petri nets. *RAMS*, 2011.

[S+05] Robert B. Stone et al. Linking product functionality to historic failures to improve failure analysis in design. *Res. in Eng. Design*, 16:96-108, 2005.

[S+08] Lui Sha et al. Cyber-physical systems: A new frontier. *IEEE SUTC*, pages 1–9, 2008.

[S+13] Seppo Sierla et al. Common cause failure analysis of cyber-physical systems situated in constructed environments. *Research in Engineering Design*, 24(4):375-394, 2013.

[T+97] Chandramohan A. Thekkath et al. Frangipani: a scalable distributed file system. *Proc. of ACM SOSP'97*, pages 224–237, 1997.

[T+11] Irem Y. Tumer et al. Integrated design-stage failure analysis of software-driven hardware systems. *IEEE Trans. on Computers*, 60(8), 2011.

[T+13a] Feng Tan et al. Guaranteeing proper-temporal-embedding safety rules in wireless cps: A hybrid formal modeling approach. *Proc. of DSN'13*, 2013.

[T+13b] Du-Ming Tsai et al. Defect detection in solar modules using ica basis images. *IEEE Trans. on Industrial Informatics*, 9(1), 2013.

[T+14] Feng Tan et al. Wip abstract: A framework on profiling cross-domain noise propagation in control cps. *Proc. of ICCPS'14*, 2014.

[T+15] Feng Tan et al. A lease based hybrid design pattern for proper-temporal-embedding of wireless cps interlocking. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 2015.

[Tab09] Paulo Tabuada. *Verification and Control of Hybrid Systems: A Symbolic Approach*. Springer, 2009.

[Tic10] Wongpiromsarn Tichakorn. Formal methods for design and verification of embedded control systems : application to an autonomous vehicle. *Dissertation (Ph.D.),California Institute of Technology*, 2010.

[US 14] US Nuclear Regulatory Commission. *Fault Tree Handbook*. NUREG-0492. June 2014.

[US 15] US Dept. of the Army. *TM 5-698-4: Failure Modes, Effects and Criticality Analyses (FMECA) for Command, Control, Communications, Computer, Intelligence, Surveillance, and Reconnaissance (C4ISR) Facilities*. 2015.

[V+10] M. Verhaegen et al. Chapter 2: Fault tolerant flight control - a survey. *Fault Tolerant Flight Control: A Benchmark Challenge*, 2010.

[W+07] Qixin Wang et al. Building robust wireless LAN for industrial control with the DSSS-CDMA cell phone network paradigm. *IEEE Transactions on Mobile Computing*, 6(6):706–719, June 2007.

[W+11] Yufei Wang et al. Wicop: Engineering wifi temporal white-spaces for safe operations of wireless body area networks in medical applications. *Proc. of the RTSS'11*, pages 170 –179, 2011.

[WHS13] Xiaofeng Wang, Naira Hovakimyan, and Lui Sha. L1simplex: fault-tolerant control of cyber-physical systems. *Proc. of ICCPS*, pages 41–50, 2013.

[Y+08] Jennifer Yick et al. Wireless sensor network survey. *Computer Networks*, 52(12):2292 – 2330, 2008.