Unmanned System Technologies

Huafeng Yu
Xin Li
Richard M. Murray
S. Ramesh
Claire J. Tomlin *Editors*

Safe, Autonomous and Intelligent Vehicles

Springer

Unmanned System Technologies

Springer's Unmanned Systems Technologies (UST) book series publishes the latest developments in unmanned vehicles and platforms in a timely manner, with the highest of quality, and written and edited by leaders in the field. The aim is to provide an effective platform to global researchers in the field to exchange their research findings and ideas. The series covers all the main branches of unmanned systems and technologies, both theoretical and applied, including but not limited to:

- Unmanned aerial vehicles, unmanned ground vehicles and unmanned ships, and all unmanned systems related research in:
- Robotics Design
- Artificial Intelligence
- Guidance, Navigation and Control
- Signal Processing
- Circuit and Systems
- Mechatronics
- Big Data
- Intelligent Computing and Communication
- Advanced Materials and Engineering

The publication types of the series are monographs, professional books, graduate textbooks, and edited volumes.

More information about this series at http://www.springer.com/series/15608

Huafeng Yu • Xin Li • Richard M. Murray
S. Ramesh • Claire J. Tomlin
Editors

Safe, Autonomous and Intelligent Vehicles

Springer

Editors
Huafeng Yu
Boeing Research and Technology
Huntsville, AL, USA

Xin Li
Duke University
Durham, NC, USA

Richard M. Murray
California Institute of Technology
Pasadena, CA, USA

S. Ramesh
General Motors R&D
Warren, MI, USA

Claire J. Tomlin
University of California
Berkeley, CA, USA

ISSN 2523-3734 ISSN 2523-3742 (electronic)
Unmanned System Technologies
ISBN 978-3-319-97300-5 ISBN 978-3-319-97301-2 (eBook)
https://doi.org/10.1007/978-3-319-97301-2

Library of Congress Control Number: 2018959861

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

An autonomous and intelligent system generally refers to a system that can automatically sense and adapt to dynamically varying environment. The term broadly covers numerous emerging and critical applications including self-driving vehicles, unmanned aircraft systems, and autonomous ships. A broad application of modern artificial intelligence and machine learning technologies is featured in these systems.

The design, modeling, verification, and validation of today's autonomous and intelligent systems have become increasingly challenging with growing functional complexity in scale and features, the integration of new artificial intelligence and machine learning technologies, the adoption of more distributed and networked architectural platforms, and stringent demands on various design constraints imposed by performance, fault tolerance, reliability, extensibility, and security. The aforementioned trend on growing complexity presents tremendous design and validation challenges to safety assurance and certification and calls for an immediate attention to this emerging area for developing radically new methodologies and practices to address the grand challenges, as well as enormous opportunities that have been rarely explored in the past.

Over the past several years, a large number of academic articles, technical reports, and industrial whitepapers have been published in this area. However, due to the highly interdisciplinary nature of the area, they are often independently reported across diverse technical communities like verification and validation, artificial intelligence, signal processing, system control, computer vision, and circuit design. Recent research and development in these areas has advanced to the point where an organized, integrated account seamlessly integrating the state of the art is immediately needed. This will help in comparing a large body of techniques in the literature and clarifying their trade-offs in terms of performance, cost, utility, etc. For this reason, there is increasing demand to report the state-of-the-art advances recently made by both academic and industrial researchers closely collaborating together in this area.

This book aims to answer this demand and to cover the important aspects of autonomous and intelligent systems, including perception, decision making, and control. It also covers the important application domains of these systems such as automobile and aerospace and, most importantly, how to define and validate the safety requirements as well as the designed systems, in particular machine learning enabled systems. To achieve these goals, rigorous verification and validation methods are developed to address different challenges, based on formal methods, compositional synthesis, machine learning, adaptive stress testing, statistical validation, model-based design, and cyber resilience.

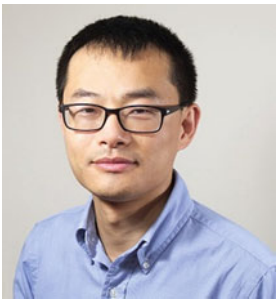
The main objective of this book is to present the major challenges related to safety of next-generation machine learning enabled autonomous and intelligent systems with growing complexity and new applications, discuss new design and validation methodologies to address these safety issues, and offer sufficient technical background to facilitate more academic and industrial researchers to collaboratively contribute to this emerging and promising area.

We anticipate that this book will provide the knowledge and background for the recent research and development and, more importantly, bring together multiple communities for interdisciplinary cross-culture interaction and set the stage for future growth in the field.

Huntsville, AL, USA — Huafeng Yu
Durham, NC, USA — Xin Li
Pasadena, CA, USA — Richard M. Murray
Warren, MI, USA — S. Ramesh
Berkeley, CA, USA — Claire J. Tomlin

Contents

About the Editors



Huafeng Yu is a senior researcher with Boeing Research & Technology. He is currently working on Safety, Assurance and Certification for Unmanned Aircraft Systems and Self-Driving Vehicles and is the technical lead for AI safety and assurance in The Boeing Company. Huafeng's main research interests include formal methods, safety assurance, artificial intelligence, machine learning, model-based engineering, and cyber security. Prior to joining Boeing, he has been working in TOYOTA, ALTRAN, INRIA, Gemplus, and Panasonic. He has more than 15 years of experience in safety research and development in the domains of automobile and aerospace. Huafeng is currently a member of IEEE Technical Committee on Cybernetics for Cyber-Physical Systems (CCPS) and chair of its industry outreach subcommittee. He has been a member of SAE standard committee for AADL. Huafeng serves as associate editor of *IET Journal on Cyber-Physical Systems* and guest editor of *IEEE Transaction on Sustainable Computing* and *ACM Transactions on Cyber-Physical Systems*. He has served on Program Committees of DAC, DATE, ICCAD, SAC, ICPS, DASC, SmartWorld, ARCH, SLIP, WICSA and CompArch, and AVICPS. Huafeng received his PhD from INRIA and the University of Lille 1 (France, 2008) and master's from University Joseph Fourier (France, 2005), both in computer science.

Xin Li received his Ph.D. degree in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, in 2005, and his M.S. and B.S. degrees in electronics engineering from Fudan University, Shanghai, China, in 2001 and 1998, respectively. He is currently a professor in the Department of Electrical and Computer Engineering at Duke University, Durham, NC, and is leading the Institute of Applied Physical Sciences and Engineering (iAPSE) at Duke Kunshan University, Kunshan, Jiangsu, China. In 2005, he co-founded Xigmix Inc. to commercialize his PhD research and served as the Chief Technical Officer until the company was acquired by Extreme DA in 2007. From 2009 to 2012, he was the Assistant Director for FCRP Focus Research Center for Circuit & System Solutions (C2S2), a national consortium working on next-generation integrated circuit design challenges. He is now on the Board of Directors for R&D Smart Devices (Hong Kong) and X&L Holding (Hong Kong). His research interests include integrated circuit, signal processing, and data analytics. Dr. Xin Li is the Deputy Editor-in-Chief of *IEEE TCAD*. He was an Associate Editor of *IEEE TCAD*, *IEEE TBME*, *ACM TODAES*, *IEEE D&T*, and *IET CPS*. He served on the Executive Committee of DAC, ACM SIGDA, IEEE TCCPS, and IEEE TCVLSI. He was the General Chair of ISVLSI, iNIS, and FAC and the Technical Program Chair of CAD/Graphics. He received the NSF CAREER Award in 2012, two IEEE Donald O. Pederson Best Paper Awards in 2013 and 2016, the DAC Best Paper Award in 2010, two ICCAD Best Paper Awards in 2004 and 2011, and the ISIC Best Paper Award in 2014. He also received six Best Paper Nominations from DAC, ICCAD, and CICC. He is a Fellow of IEEE.

Richard M. Murray is the Thomas E. and Doris Everhart Professor of Control and Dynamical Systems and Bioengineering at the California Institute of Technology (Caltech). He received his B.S. degree in electrical engineering from Caltech in 1985 and his M.S. and Ph.D. degrees in electrical engineering and computer sciences from the University of California, Berkeley, in 1988 and 1991, respectively. He joined the Caltech faculty in Mechanical Engineering in 1991 and helped found the Control and Dynamical Systems program in 1993. In 1998–1999, he took a sabbatical leave and served as the director of Mechatronic Systems at the United Technologies Research Center, Hartford, Connecticut. Upon returning to Caltech, he served as the division chair (dean) of Engineering and Applied Science from 2000 to 2005, the director for Information Science and Technology from 2006 to 2009, and interim division chair from 2008 to 2009. He received the Donald P. Eckman Award in 1997, the IFAC Harold Chestnut Textbook Prize (with Karl Åström) in 2011, and the IEEE Bode Lecture Prize in 2016 and is an elected member of the National Academy of Engineering (2013). His research is in the application of feedback and control to networked systems, with applications in biology and autonomy. Current projects include the analysis and design of biomolecular feedback circuits, the synthesis of discrete decision-making protocols for reactive systems, and the design of highly resilient architectures for autonomous systems. He is a cofounder of Synvitrobio, Inc., a cell-free synthetic biology company in San Francisco, and a member of the Defense Innovation Board, which advises the U.S. Secretary of Defense.

S. Ramesh has been with General Motors Global R&D in Warren, MI, where he currently holds the position of Senior Technical Fellow and thrust area lead for model-based embedded software. At General Motors, he is responsible for providing technical leadership for research and development in several areas related to Electronics, Control, and Software processes, methods, and tools. Earlier he was in India Science Lab serving as the lab group manager for two research groups on Software and System Verification and Validation. During this time, he led several projects on next-generation rigorous verification and testing methods for control software, which resulted in proof of concept methods and tools that were piloted on several SW subsystems across different domains like BCM, HVAC, and Active Safety. His broad areas of interests are rigorous software engineering, embedded systems, and real-time systems. He is the author of several patents and has published more than 100 papers in peer-reviewed international journals and conferences. He is on the editorial boards of the *International Journal on Real-Time Systems* and *EURASIP Journal on Embedded Systems* and earlier on *IEEE Journal on Embedded System Letters*. Prior to joining GM R&D, he was on the faculty of the Department of Computer Science and Engineering at IIT Bombay for more than fifteen years. At IIT Bombay, he played a major role in setting up a National Centre for Formal Design and Verification of Software. As the founding head of this Centre, he carried out many projects on verification of embedded software, for several government organizations. He is a fellow of the Indian National Academy of Engineering and was visiting/adjunct faculty of many institutions. Ramesh earned his B.E. degree in electronics and communication engineering from Indian Institute of Science, Bangalore, and his PhD degree in computer science and engineering from Indian Institute of Technology Bombay, India.

Claire J. Tomlin is a professor of electrical engineering and computer sciences at the University of California at Berkeley, where she holds the Charles A. Desoer Chair in Engineering. She was assistant, associate, and full professor at Stanford (1998–2007) and joined Berkeley in 2005. She has been an affiliate at Lawrence Berkeley National Laboratory in the Life Sciences Division since January 2012. She works in hybrid systems and control, with applications to air traffic and unmanned air vehicle systems, robotics, energy, and biology. Dr. Tomlin pioneered methods for computing the reachable set to encompass all behaviors of a hybrid system, which makes it possible to verify that the system stays within a desired safe range of operation and to design controllers to satisfy constraints. She has applied these methods to collision avoidance control for multiple aircraft and to the analysis of switched control protocols in avionics and embedded controllers in aircraft. Her work has been tested in simulation and UAV test flights, and applied to and flown on two large commercial platforms: (1) Boeing aircraft: Her method was used to compute collision zones for two aircraft paired approaches, and was flown on a Boeing T-33 test aircraft, flying close to a piloted F-15. The F-15 pilot flew "blunders" into the path of the T-33, which used Tomlin's algorithm to avoid collision. (2) Driven on Scania trucks: Dr. Tomlin's method was used to derive a minimum safe distance between transport trucks driving in high-speed platoons for fuel savings, and revealed that the relative distance used today can be reduced significantly with this automation. Her work is also being considered for application in the Next Generation Air Transportation System (NextGen) and in Unmanned Aerial Vehicle Traffic Management (UTM). Dr. Tomlin is a MacArthur Foundation, IEEE, and AIMBE fellow. She has received the Donald P. Eckman Award of the American Automatic Control Council in 2003, the Tage Erlander Guest Professorship of the Swedish Research Council in 2009, an honorary doctorate from KTH in 2016, and in 2017 the IEEE Transportation Technologies Award.

Chapter 1
Introduction

Huafeng Yu, Xin Li, Richard M. Murray, S. Ramesh, and Claire J. Tomlin

While design and implementation of autonomous and intelligent systems is not a new area, the recent advances in sensor, machine learning, and other software and hardware bring enormous opportunities as well as complexities in these systems, and they accompany increasingly complex safety requirements as well as constraints. Conventional design and validation methodologies are often not sufficiently equipped to address the grand challenges of safety and assurance required by today's systems. Meanwhile, there is always a high expectation of safer driving/operations provided by modern autonomous systems. Recent Tesla and Uber's fatal accidents, in which autonomous driving features were involved, are serious reminds that research and development on autonomy safety and assurance are still far from being well matured, for experimental or production-level autonomous vehicles.

This book, "Safe, Autonomous and Intelligent Vehicles", presents the start-of-the-art research and development for the emerging topics on design, modeling,

H. Yu (✉)
Boeing Research and Technology, Huntsville, AL, USA
e-mail: huafeng.yu@boeing.com

X. Li
Duke University, Durham, NC, USA
e-mail: xinli.ece@duke.edu

R. M. Murray
California Institute of Technology, Pasadena, CA, USA
e-mail: murray@cds.caltech.edu

S. Ramesh
General Motors R&D, Warren, MI, USA
e-mail: ramesh.s@gm.com

C. J. Tomlin
University of California, Berkeley, Berkeley, CA, USA
e-mail: tomlin@eecs.berkeley.edu

H. Yu et al. (eds.), *Safe, Autonomous and Intelligent Vehicles*,
Unmanned System Technologies, https://doi.org/10.1007/978-3-319-97301-2_1

verification, and validation dedicated to autonomous and intelligent systems, featured by artificial intelligence and machine learning. In particular, it emphasizes the innovative methods and tools to address the grand challenges of safety, security, assurance, and certification. The objective of this book is to present the major challenges and discuss promising solutions and, most importantly, offer the technical background to facilitate more academic and industrial researchers to collaboratively contribute to this area. It is among the first formal-methods-oriented books to offer a broad spectrum of research on autonomous system functionality (including perception, decision making, control, etc.) in various application domains (such as automobile and aerospace).

Chapter 2 presents a novel methodology to validate vision-based autonomous driving systems over different circuit corners with consideration of temperature variation and circuit aging. The proposed approach seamlessly integrates the image data recorded under nominal conditions with comprehensive statistical circuit models to synthetically generate the critical corner cases for which an autonomous driving system is likely to fail. As such, a given automotive system can be robustly validated for these worst-case scenarios that cannot be easily captured by physical experiments. To efficiently estimate the rare failure rate of an autonomous system, a novel Subset Sampling (SUS) algorithm is further proposed. In particular, a Markov Chain Monte Carlo algorithm based on graph mapping is developed to accurately estimate the rare failure rate with a minimal amount of test data, thereby minimizing the validation cost. The numerical experiments show that SUS achieves 15.2× runtime speed-up over the conventional brute-force Monte Carlo method.

Chapter 3 investigates safety and performance implications of the Cyber-Physical Systems subject to cyberattacks from a control systems perspective. To design an attack-resilient controller while not making it excessively conservative, this work proposes a hybrid control framework containing multiple sub-controllers that can adapt to various cyberattacks by switching sub-controllers. Further, the robustness of the controller is achieved by the switching logic which determines the safest sub-controller whose future performance under attack is the least bad. The developed hybrid controller is analytically verified for the finite time horizon case, and further extended to the infinite time horizon case. Simulation results are provided to demonstrate the performance and applicability of the proposed controller design to an unmanned aircraft system subject to cyberattacks.

Chapter 4 first summarizes recent efforts in overcoming two key challenges in the design of safe autonomous systems: inaccurate dynamics model of the system; partially-observable environments. Machine Learning (ML) and Artificial Intelligence (AI)-based controller techniques have the potential to transform real-world autonomous systems because of the data-driven design approach. However, given the safety-critical nature of autonomous vehicle systems, this transformation requires approaches which are robust, resilient, and safe. This chapter presents both the proposed learning-based schemes and the efforts made towards verifying such schemes. Hamilton-Jacobi (HJ) reachability theory is first presented; it is a safety verification tool that can be used to verify a system when its dynamics model and a "model" of the environment are known. Then, the authors move on to the

scenarios where learning-based components are involved in the system, either due to unknown dynamics or uncertain environment, and discuss how HJ reachability-based algorithms can be developed to verify such systems.

Chapter 5 presents an extension of stress testing in simulation, which plays a critical role in the validation of safety-critical systems. The analysis of failure events is important in understanding the causes and conditions of failures, informing improvements to the system, and the estimation and categorization of risks. However, stress testing of safety-critical systems can be very challenging. Finding failure events can be difficult due to the size and complexity of the system, interactions with an environment over many time steps, and rarity of failure events. The authors present adaptive stress testing (AST), an accelerated stress testing method for finding the most likely path to a failure event. Adaptive stress testing formulates stress testing as a sequential decision process and then uses reinforcement learning to optimize it. By using learning during search, the algorithm can automatically discover important parts of the state space and adaptively focus the search. Adaptive stress testing is applied to stress test a prototype of next-generation aircraft collision avoidance system in simulated encounters, where the authors find and analyze the most likely paths to near mid-air collision.

Chapter 6 first explores, in the context of autonomous vehicles, many driver convenience and safety automation systems being introduced into production vehicles. These systems often include controllers designed for individual tasks. This chapter addresses the challenge of designing individual controllers for interacting subsystems, and composing them in a way that preserves the guarantees that each controller provides on each subsystem. The proposed approach is based on formal methods and correct-by-construction controller synthesis. Mechanisms for handling implementation and model imperfections and contract-based composition of functionality are discussed. The ideas are demonstrated in high fidelity simulations in CarSim, and also on a real vehicle in Mcity, the autonomous vehicle testing facility at the University of Michigan.

Chapter 7 addresses the reachable set estimation and safety verification problems for Nonlinear Autoregressive-Moving Average (NARMA) models in the forms of neural networks. The neural networks involved in the model are a class of feed-forward neural networks called Multi-Layer Perceptrons (MLPs). By partitioning the input set of an MLP into a finite number of cells, a layer-by-layer computation algorithm is developed for reachable set estimation of each individual cell. The union of estimated reachable sets of all cells forms an over-approximation of the reachable set of the MLP. Furthermore, an iterative reachable set estimation algorithm based on reachable set estimation for MLPs is developed for NARMA models. The safety verification can be performed by checking the existence of non-empty intersections between unsafe regions and the estimated reachable set. Several numerical examples are provided to illustrate the approach.

Chapter 8 first studies a licensing process through which pilots and drivers are certified to operate vehicles in aviation and surface transportation. The process includes an assessment of physical readiness, a written knowledge exam, and a practical exam. This is a process whereby a human examiner assesses whether

humans can effectively operate autonomously either in actual or simulated conditions. Given the rise of autonomous transportation technologies and the debate as to how such technologies that leverage probabilistic reasoning could and should be certified, could such licensing approaches be extended to certification of autonomous systems? This chapter discusses how machine and human autonomy are similar and different, how and why licensing processes have developed historically across the two transportation domains, and how autonomous transportation systems could adapt the principles of human-based licensing processes for certification.

Chapter 9 introduces a model-based software synthesis flow to address significant challenges in the design of modern cyber-physical systems, including the increasing complexity in both functionality and architecture, the dynamic physical environment and various safety-critical design objectives and system requirements (timing, security, fault-tolerance, etc.). This synthesis flow is conducted in a holistic framework across functional layer and architecture layer. In this framework, the authors introduce task generation and task mapping under various design metrics including timing, security, fault-tolerance, control performance, etc.

Chapter 10 presents a compositional approach for the scalable, formal verification of autonomous systems that contain Deep Neural Network components. The approach uses assume-guarantee reasoning whereby contracts, encoding the input–output behavior of individual components, allow the designer to model and incorporate the behavior of the learning-enabled components working side-by-side with the other components. The approach is also illustrated on an example taken from the autonomous vehicles domain.

Chapter 2
Efficient Statistical Validation of Autonomous Driving Systems

Handi Yu, Weijing Shi, Mohamed Baker Alawieh, Changhao Yan, Xuan Zeng, Xin Li, and Huafeng Yu

2.1 Introduction

The last decade has witnessed tremendous advance in Advanced Driver Assistance System (ADAS) [1] and autonomous driving [2, 3]. Such systems can perform numerous intelligent functions [4] such as collision avoidance, lane departure warning, traffic sign detection, etc. When implementing these systems, machine learning plays an essential role to interpret sensor data and understand surrounding environment. These machine learning systems significantly improve driving safety and comfort, but in turn raise new challenges concerning system robustness and reliability.

One main challenge is that most autonomous driving systems heavily rely on visual perception and are sensitive to environmental variations, e.g., rain, fog, etc.

H. Yu
Duke University, Durham, NC, USA
e-mail: hy126@duke.edu

W. Shi · M. B. Alawieh
Carnegie Mellon University, Pittsburgh, PA, USA
e-mail: weijings@cmu.edu; malawieh@cmu.edu

C. Yan · X. Zeng
Fudan University, Shanghai, P. R. China
e-mail: yanch@fudan.edu.cn; xzeng@fudan.edu.cn

X. Li (✉)
Duke University, Durham, NC, USA

Duke Kunshan University, Kunshan, P.R. China
e-mail: xinli.ece@duke.edu

H. Yu
Boeing Research and Technology, Huntsville, AL, USA
e-mail: huafeng.yu@boeing.com

H. Yu et al. (eds.), *Safe, Autonomous and Intelligent Vehicles*,
Unmanned System Technologies, https://doi.org/10.1007/978-3-319-97301-2_2

An autonomous driving system cannot be perfectly accurate. In order to create a robust autonomous driving system, it is crucial to validate its performance over all possible environmental scenarios. Such a validation process requires extensive on-road records of 10^6–10^8 miles to capture all corner cases [5]. However, physically observing all these corner cases over a long time is almost impossible in practice.

To address this issue, several novel methodologies have been proposed in the literature to synthetically generate important corner cases (e.g., rain [6] and fog [7]) that cannot be easily observed in physical experiments [6–9]. An autonomous driving system is then validated by using both physical test data and synthetic test data. However, the state-of-the-art research on automotive validation mainly focuses on system-level and environmental issues. As an autonomous driving vehicle is equipped with a large amount of electronic modules to facilitate real-time sensing, decision, and control, circuit-level non-idealities (e.g., temperature variation and circuit aging) play an increasingly important role in system performance, robustness, and reliability. The aforementioned aspect is extremely important; yet it has been rarely explored in the literature.

In this chapter, we propose a model-based methodology to synthesize test data over various circuit corners with consideration of temperature variation and circuit aging. Our key idea is to seamlessly integrate the image data recorded under nominal conditions with comprehensive statistical circuit models to synthetically generate all corner cases. Towards this goal, we first carefully study the underlying physics of temperature variation and circuit aging for image sensors, and then accurately capture these non-idealities by circuit-level statistical models. Next, we inject the circuit-level non-idealities into the raw image signals estimated from the nominal recordings by inverting the image processing pipeline for a visual perception system. Finally, the new corner cases incorporating circuit-level non-idealities are synthetically generated by propagating the "contaminated" image signals over the image processing pipeline. As will be demonstrated by our experimental results in Sect. 2.3, the true positive rate of a STOP sign detection system is reduced from 99.69% to 73.20% after considering circuit-level non-idealities.

Based on the aforementioned test case generation methods, an autonomous driving system can be validated over different scenarios. However, there always exists another main challenge in the validation of autonomous driving systems at design stage regarding the extremely small failure rate. In order to create a reliable automotive system, it is crucial to ensure that the machine learning system, including both hardware and software, is highly accurate. For example, for a STOP sign detection system, an extremely small failure rate (e.g., 10^{-12}) must be accomplished so that the system does not fail every few hours.

To validate such a small failure rate, a test vehicle may be built to perform real-time road test, which is costly and may take several months to complete [10]. Alternatively, virtual testing based on environment emulation has been adopted, where emulators are used to generate synthetic inputs (e.g., hundreds of hours of video data) for the systems under test, instead of relying on physical sensing data [8, 11, 12]. However, at design stage, no physical system implementation is available yet. A system design must be validated by simulation, which is computationally expensive especially when an accurate simulation model is used. For example,

simulating a multi-core CPU can be 1800× slower than running the physical hardware system [13]. In this case, if we require to test a large amount of data (e.g., 10^{13} image blocks) in order to capture the extremely small failure rate, the validation procedure becomes prohibitively expensive (e.g., taking more than 10 years).

To reduce the computational cost, we propose to adopt the idea of subset sampling (SUS) [14, 15]. Namely, we represent the rare failure probability as the product of several conditional probabilities. Each conditional probability corresponds to an extended failure rate that is defined by relaxing the failure boundary. As such, the conditional probabilities are relatively large and, hence, can be estimated inexpensively. In addition, we develop several novel ideas and heuristics in order to make SUS of great efficiency for our application of interest. The numerical experiments in Sect. 2.4 demonstrate that the SUS achieves more than 15.2× runtime speed-up over the conventional brute-force Monte Carlo method.

The remainder of this chapter is organized as follows. In Sect. 2.2, we briefly review the background of visual perception system including its camera model and computer vision algorithm. In Sect. 2.3, we present the proposed test data generation method to synthesize important corner cases and then show impact of circuit-level non-idealities in the experimental results. Next, we introduce our proposed SUS approach and then demonstrate its efficiency by numerical experiments in Sect. 2.4. Finally, we conclude in Sect. 2.5.

2.2 Background

A generic visual perception system is composed of three major components: image sensing, image processing, and visual perception. In this section, we briefly review their key functions and roles.

2.2.1 Image Sensing

Figure 2.1a shows the simplified architecture of a CMOS image sensor. The light passing through the lens is filtered by a color filter array (CFA, most commonly, the Bayer CFA with RGB masks [17]), and then the light intensity with the corresponding RGB wavelength is sensed by the pixel detectors where the incident photons are converted to photoelectrons. During this sensing process, automatic control is often applied to determine the appropriate settings such as exposure (e.g., aperture size, shutter speed, etc.) and focal position.

For a given scene, assume that the number of the photoelectrons generated in a detector is n_{SIG}. Based on the pixel circuit schematic in Fig. 2.1b, transferring the electrons to the capacitor C changes the voltage across the capacitor [16]:

$$v_{SIG} = v_0 - v_t = \frac{G_{PIXEL} q}{C} n_{SIG}, \tag{2.1}$$

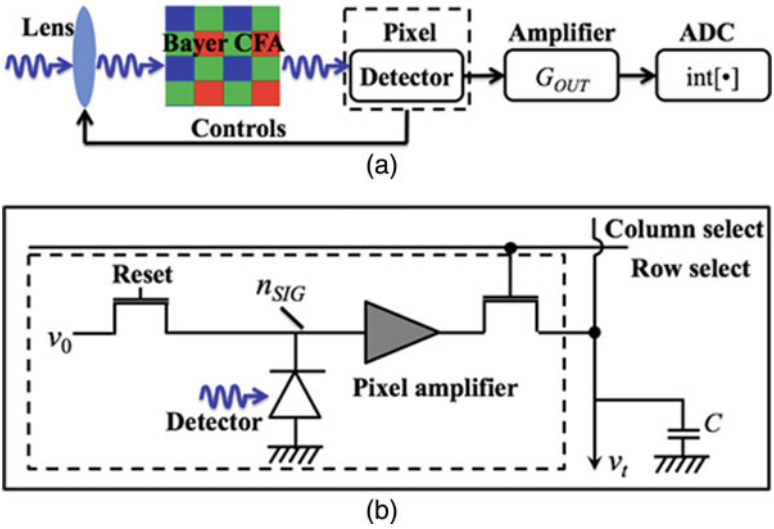


Fig. 2.1 The simplified architecture is shown for a CMOS color sensor: (**a**) overall architecture of the sensing system, and (**b**) simplified circuit schematic of a pixel [16]

where v_0 and v_t denote the input and output voltages of the pixel, v_{SIG} is the voltage change, G_{PIXEL} is the gain of the pixel amplifier, and q is the elementary charge.

To generate a digital signal x_{SIG}, v_{SIG} is further amplified and digitalized by an analog-to-digital converter (ADC) [17]:

$$x_{SIG} = \text{int}\left[\frac{G_{ISO}G_{FIX}v_{SIG}}{v_{MAX}} \times 2^{N_{ADC}}\right], \tag{2.2}$$

where int[•] takes the integer value of a real-valued variable, G_{ISO} is the gain depending on the ISO level, G_{FIX} is a fixed gain, v_{MAX} is the maximum pixel output in analog domain, and N_{ADC} stands for the number of bits provided by the ADC. Combining (2.1) and (2.2) yields:

$$x_{SIG} = \text{int}\left[\frac{G_{ISO}G_{FIX}}{v_{MAX}}\left(\frac{G_{PIXEL}q}{C}\right)n_{SIG} \times 2^{N_{ADC}}\right]. \tag{2.3}$$

2.2.2 Image Processing

In order to produce an accurate representation of the captured scene, the raw image derived from a CMOS image sensor must be processed over an image processing pipeline. Such a pipeline differs from one manufacturer to another; however, several basic operations are commonly used.

A CMOS image sensor is often composed of both active pixels and dark pixels. Active pixels are arrayed for image sensing, while dark pixels are covered by an opaque mask to measure dark noises. Black-level compensation is implemented to remove dark noises from the raw digital signals of active pixels [17]:

$$x_{BC} = x_{SIG} - \overline{x}_{DP}, \tag{2.4}$$

where x_{BC} denotes the pixel data after black-level compensation, and the dark noise $\overline{x}_{DP}$ is calculated as the average digital signal of the dark pixels.

Due to the CFA filter, each signal x_{SIG} or x_{BC} in (2.4) represents the light intensity of one color only: red, green, or blue. Hence, M pixels can be represented as:

$$\mathbf{x}_{SIG} = \left[\mathbf{x}_{SIG,R}\ \mathbf{x}_{SIG,G}\ \mathbf{x}_{SIG,B}\right] = \left[x_{SIG,1}\ x_{SIG,2}\ \cdots\ x_{SIG,M}\right], \tag{2.5}$$

$$\mathbf{x}_{BC} = \left[\mathbf{x}_{BC,R}\ \mathbf{x}_{BC,G}\ \mathbf{x}_{BC,B}\right] = \left[x_{BC,1}\ x_{BC,2}\ \cdots\ x_{BC,M}\right], \tag{2.6}$$

where $\mathbf{x}_{SIG,R} \in \Re^{1\times MR}$, $\mathbf{x}_{SIG,G} \in \Re^{1\times MG}$, $\mathbf{x}_{SIG,B} \in \Re^{1\times MB}$, $\mathbf{x}_{BC,R} \in \Re^{1\times MR}$, $\mathbf{x}_{BC,G} \in \Re^{1\times MG}$, and $\mathbf{x}_{BC,B} \in \Re^{1\times MB}$ denote the pixels corresponding to different colors. When adopting a CFA filter, we can only sense one color for each pixel. To reconstruct a full color image, interpolation is required, for example, bilinear interpolation which can be mathematically represented as [17]:

$$\mathbf{x}_{IP} = \begin{bmatrix} \mathbf{x}_{BC,R}\mathbf{C}_{IP,R} \\ \mathbf{x}_{BC,G}\mathbf{C}_{IP,G} \\ \mathbf{x}_{BC,B}\mathbf{C}_{IP,B} \end{bmatrix}, \tag{2.7}$$

where $\mathbf{x}_{IP} \in \Re^{3\times M}$ represents the RGB values of M pixels, and $\mathbf{C}_{IP,R} \in \Re^{MR\times M}$, $\mathbf{C}_{IP,G} \in \Re^{MG\times M}$, and $\mathbf{C}_{IP,B} \in \Re^{MB\times M}$ denote the interpolation operators.

The interpolated image remains different from what is observed by human eyes. Human vision system is able to map the sensed color of a white object under different lighting conditions to a standard white color. To mimic this functionality, white balance is further applied [18]:

$$\mathbf{x}_{WB} = \mathbf{C}_{WB}\mathbf{x}_{IP}, \tag{2.8}$$

where $\mathbf{x}_{WB}$ stands for the image data after white balance and $\mathbf{C}_{WB} \in \Re^{3\times 3}$ represents the coefficient matrix which is determined by the reference color temperature.

Furthermore, cameras from different manufacturers have their own tones and the images should be normalized by color correction [19]:

$$\mathbf{x}_{CC} = \mathbf{C}_{CC}\mathbf{x}_{WB}, \tag{2.9}$$

where $\mathbf{x}_{CC}$ stands for image data after color correction and $\mathbf{C}_{CC} \in \Re^{3\times 3}$ represents the color correction matrix.

Gamma compensation is another processing step commonly used in cameras to increase the sensitivity of dark tones [19]:

$$\mathbf{x}_{OUT} = gamma\left(\mathbf{x}_{CC}\right), \tag{2.10}$$

where $\mathbf{x}_{OUT}$ stands for image data after Gamma compensation and $gamma\ (\bullet)$ denotes a nonlinear pixel-wise mapping of an N_{ADC}-bit color value to an 8-bit one:

$$x_{OUT} = \text{int}\left[\alpha \cdot (x_{CC})^{\gamma}\right], \tag{2.11}$$

where α and γ are two coefficients for Gamma compensation.

In summary, a typical image processing pipeline can be mathematically expressed as:

$$\mathbf{x}_{OUT} = gamma\left(\mathbf{C}_{CC}\mathbf{C}_{WB}\begin{bmatrix}\mathbf{x}_{BC,R}\mathbf{C}_{IP,R}\\ \mathbf{x}_{BC,G}\mathbf{C}_{IP,G}\\ \mathbf{x}_{BC,B}\mathbf{C}_{IP,B}\end{bmatrix}\right). \tag{2.12}$$

Equation (2.12) will be used in the following sections to synthesize important corner cases for image data with consideration of circuit-level non-idealities.

2.2.3 Visual Perception

After the aforementioned image processing step, visual perception aims to further recognize and understand the surrounding scenes by applying computer vision and/or machine learning algorithms. Hence, computer vision and machine learning algorithms are crucial to many critical functions such as localization and trajectory planning. In this regard, object detection is one of the most common tasks. Consider traffic sign detection as an example. It makes a binary decision to determine whether the input image contains a traffic sign of interest or not [20]:

$$f\left(\mathbf{x}\right)\begin{cases}\geq \delta\ \left(Traffic\ sign\ detected\right)\\ < \delta\ \left(Traffic\ sign\ undetected\right)\end{cases}, \tag{2.13}$$

where $f(\mathbf{x})$ represents the discriminant function, $\mathbf{x}$ stands for the vector containing all pixels of an input image, and δ denotes the threshold value that defines the decision boundary.

To intuitively understand a detection system, we consider traffic sign detection as an example and briefly review its algorithm flow based on cascade classifiers [21], as shown in Fig. 2.2. In such a system, a camera is mounted in front of the vehicle and it captures the front-view video in real time. Each video frame is cropped into a set of image blocks and each image block is sent to the cascade classifiers to decide whether it contains the target traffic sign or not.

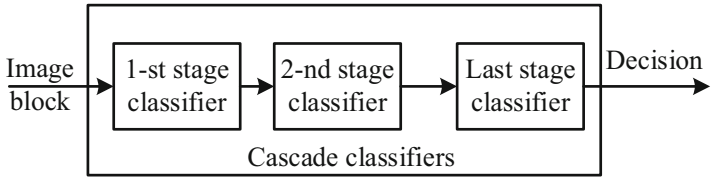


Fig. 2.2 An algorithm flow is shown for traffic sign detection using cascade classifiers

Cascade classifiers are composed of a sequence of classifiers. If one stage of the classifiers decides that the input image block does not contain a traffic sign, this image block is rejected and will not be further passed to the next stage. For more details about the cascade classifiers, please refer to [22].

2.3 Test Data Generation

When designing machine learning systems, it is necessary to validate its accuracy over all scenarios. Hence, a large test dataset must be collected for validation purpose, which is often time-consuming or even financially unaffordable. Motivated by this observation, we develop a novel model-based methodology to efficiently synthesize test data with consideration of temperature variation and circuit aging.

2.3.1 Temperature Variation

Ensuring performance over the operation temperature range is critical for a vision-based autonomous driving system. As temperature increases, dark current in detectors caused by the thermal charge carriers in depletion region [17] may significantly increase, thereby substantially degrading the signal-to-noise ratio of the recorded images and eventually reducing the recognition accuracy. The dark current density j_D is calculated as:

$$j_D = \frac{q n_{IN}}{2\tau} w, \tag{2.14}$$

where n_{IN} denotes the intrinsic concentration of silicon, τ denotes the generation lifetime of electron-hole pairs, and w stands for the width of the depletion region.

In (2.14), the depletion width w can be calculated using the junction capacitance C_J:

$$w = \frac{\varepsilon_{SI}}{C_J}, \tag{2.15}$$

where ε_{SI} denotes the dielectric constant of silicon and the junction capacitance C_J is modeled by [23]:

$$C_J = \frac{C_{J0}}{\sqrt{1 - \frac{v}{v_{BI}}}}, \tag{2.16}$$

where C_{J0} denotes the junction capacitance at zero bias, v is the applied voltage, and v_{BI} represents the built-in potential. Combining (2.14, 2.15, 2.16) yields:

$$j_D = \frac{q n_{IN}}{2\tau} \frac{\varepsilon_{SI}}{C_{J0}} \sqrt{1 - \frac{v}{v_{BI}}}. \tag{2.17}$$

In (2.17), τ and v can be found from the sensor manual, and the value of n_{IN} depends on the temperature [24]:

$$n_{IN} \propto T^{\frac{3}{2}} \cdot e^{-\frac{E_G}{2k_B T}}, \tag{2.18}$$

where T denotes the temperature, E_G denotes the energy bandgap, and k_B is the Boltzmann constant. Substituting (2.18) into (2.17), one can observe the fact that dark current exponentially depends on temperature:

$$j_D \propto \frac{q}{2\tau} \frac{\varepsilon_{SI}}{C_{J0}} \sqrt{1 - \frac{v}{v_{BI}}} \cdot T^{\frac{3}{2}} \cdot e^{-\frac{E_G}{2k_B T}}. \tag{2.19}$$

In addition, C_{J0} and v_{BI} in (2.19) vary with temperature. To accurately capture the dependency between dark current and temperature, we consider the following equations adopted by the BSIM4 model [23]:

$$\begin{aligned} n_{IN} &= 1.45 \times 10^{10} \cdot \left(\frac{T_0}{300.15}\right)^{\frac{3}{2}} \cdot e^{\left(21.56 - \frac{q E_G}{2k_B T}\right)} \\ C_{J0}(T) &= C_{J0}(T_0) \cdot (1 + a_C (T - T_0)) \\ v_{BI}(T) &= v_{BI}(T_0) \cdot (1 + a_V (T - T_0)) \end{aligned}, \tag{2.20}$$

where T_0 denotes the nominal temperature, $C_{J0}(T_0)$ and $v_{BI}(T_0)$ represent the capacitance and build-in potential at T_0 respectively, and a_C and a_V are two coefficients.

Based on (2.17), the expected number of dark electrons ${n_{DARK}}^*$ generated over the exposure time t_{EXP} can be calculated as:

$${n_{DARK}}^* = \frac{j_D A_{PH} t_{EXP}}{q}, \tag{2.21}$$

where A_{PH} is the area of the photodiode. Both A_{PH} and t_{EXP} are found from the sensor manual. In practice, the generation of dark electrons is not deterministic

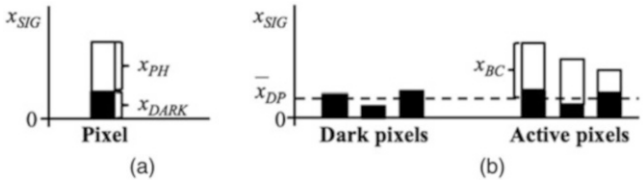


Fig. 2.3 (**a**) The raw digital signal including both the photoelectrons and the dark electrons is shown for one pixel. (**b**) The impact of dark electrons is illustrated with consideration of black-level compensation

[24]. Each dark electron appears independently and randomly with a probability. Therefore, the actual number of dark electrons varies from pixel to pixel and from time to time, although their expected numbers are identical. For each pixel, the actual number of dark electrons n_{DARK} follows a Poisson distribution [24]:

$$\Pr[n_{DARK} = \lambda] = \frac{(n_{DARK}{}^*)^{\lambda}}{\lambda!} e^{-n_{DARK}{}^*} \ (\lambda = 0, 1, 2, \ldots) \cdot \tag{2.22}$$

Figure 2.3a shows the contribution of dark electrons for one pixel. Once generated, the dark electrons transferring to the capacitor C in Fig. 2.1b will contribute to the digital signal x_{SIG} in Fig. 2.3a. According to (2.1, 2.2), v_{SIG} is proportional to n_{SIG} and x_{SIG} is further proportional to v_{SIG}, assuming that the quantization effect by ADC is negligible. Therefore, the mapping from n_{SIG} to x_{SIG} is almost linear and is composed of both photoelectrons and dark electrons:

$$x_{SIG} = a_{CV} n_{SIG} = a_{CV} (n_{PH} + n_{DARK}) = x_{PH} + x_{DARK}, \tag{2.23}$$

where a_{CV} represents the conversion coefficient, n_{PH} is the number of photoelectrons, x_{PH} denotes the digital signal due to photoelectrons, and x_{DARK} denotes the digital signal due to dark electrons:

$$x_{DARK} = \text{int}\left[\frac{G_{ISO} G_{FIX} n_{DARK}}{v_{MAX}} \cdot \left(\frac{G_{PIXEL} q}{C}\right) \times 2^{N_{ADC}}\right]. \tag{2.24}$$

To minimize the impact of dark electrons, black-level compensation is often applied. However, since dark electrons are generated by a random process, dark signals vary from pixel to pixel as shown in Fig. 2.3b. Substituting (2.23) into (2.4), the pixel data after black-level compensation is:

$$x_{BC} = x_{SIG} - \overline{x}_{DP} = x_{PH} + (x_{DARK} - \overline{x}_{DP}) . \tag{2.25}$$

While the aforementioned scheme successfully removes the global dark noise, it cannot accurately compensate the dark noises for individual cells due to device mismatches. Hence, dark noise may substantially influence the signal-to-noise ratio of recorded images especially at high temperature.

2.3.2 Circuit Aging

An autonomous driving system should maintain extremely high reliability over the product lifetime (i.e., more than 15 years). However, similar to other electrical devices, an image sensor ages over time, resulting in defective pixels on the generated images [25]. The accumulation of these defects is another key factor that degrades the quality of an autonomous driving system.

While there are a number of physical reasons for aging, terrestrial cosmic ray radiation is one of the key factors that influence an image sensor. It damages the silicon bulk of photodiodes and causes a variety of defects such as hot pixels [25]. Typically, a hot pixel contains an illumination-independent component that increases linearly with the exposure time t_{EXP}. In addition, an offset is often associated with the hot pixel, yielding the following model [26]:

$$x_{SIG,HOT} = x_{SIG} + G_{ISO} \cdot (a_{HOT} \cdot t_{EXP} + b_{HOT}), \tag{2.26}$$

where $x_{SIG,HOT}$ denotes the raw digital signal of a hot pixel, a_{HOT} is the coefficient of the illumination-independent term, and b_{HOT} stands for the offset. The coefficients a_{HOT} and b_{HOT} depend on the radiation effect. To provide a conservative estimation, we assume that a_{HOT} and b_{HOT} are sufficiently large so that $x_{SIG,HOT}$ reaches the maximum value $x_{SIG,MAX}$ (i.e., a constant):

$$x_{SIG,HOT} = x_{SIG,\mathrm{MAX}}. \tag{2.27}$$

The defects of different pixels are statistically independent. Therefore, the density of defects increases at a constant rate over time. The growth rate of defect density D_{HOT} for CMOS image sensors is empirically modeled as [26]:

$$D_{HOT} = D_0 \cdot {A_{PH}}^{c_A} \cdot {ISO}^{c_{ISO}}, \tag{2.28}$$

where ISO stands for the ISO level, and D_0, c_A, and c_{ISO} are the coefficients as defined in [26]. Based on (2.28), the expected number of defects ${n_{AP,HOT}}^*$ across an active pixel array over the aging time t_{AGE} equals:

$${n_{AP,HOT}}^* = D_{HOT} \cdot A_{AP} \cdot t_{AGE}, \tag{2.29}$$

where A_{AP} denotes the effective area of an image sensor which is the total area of the photodiodes for all active pixels and can be found from the sensor manual.

Hot pixels occur randomly and independently over time and space. The actual number of defects $n_{AP,HOT}$ appearing in active pixels over time t_{AGE} is modelled by a Poisson distribution [27]:

$$\Pr\left[n_{AP,HOT} = \lambda\right] = \frac{({n_{AP,HOT}}^*)^{\lambda}}{\lambda!} e^{-{n_{AP,HOT}}^*} \ (\lambda = 0, 1, 2, \cdots) \cdot \tag{2.30}$$

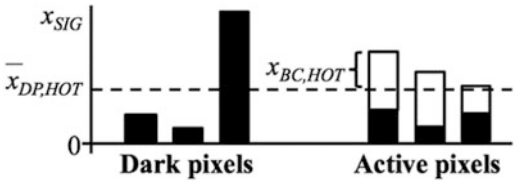


Fig. 2.4 The impact of defects in dark pixels is illustrated for black-level compensation

Given the number of defects in an active pixel array, these defects are distributed randomly across the array. Then the output signals of these defective pixels are replaced by $x_{SIG,MAX}$ as shown in (2.27).

The aforementioned defects influence the output signals of the corresponding pixels significantly. When propagating the defective signals over the image processing pipeline, their impact may spread globally over all pixels in the array. For instance, when applying black-level compensation, the dark signals are calculated based on the dark pixels, as discussed in Sect. 2.2.2. Hence, once there are defects in the dark pixels, these defects will contribute to all pixels after applying black-level compensation, as shown by the example in Fig. 2.4. In this example, the third dark pixel is defective with a large output. The dark noise with consideration of the defect (i.e., $\bar{x}_{DP,HOT}$) increases, thereby resulting in an over-compensation for all active pixels.

Similar to (2.29, 2.30), the actual number of defects in dark pixels $n_{DP,HOT}$ follows a Poisson distribution:

$$\Pr\left[n_{DP,HOT} = \lambda\right] = \frac{\left(n_{DP,HOT}{}^{*}\right)^{\lambda}}{\lambda!} e^{-n_{DP,HOT}{}^{*}} \ (\lambda = 0, 1, 2, \cdots) \cdot \tag{2.31}$$

In (2.31), the average value $n_{DP,HOT}{}^{*}$ equals:

$$n_{DP,HOT}{}^{*} = D_{HOT} \cdot A_{DP} \cdot t_{AGE}, \tag{2.32}$$

where A_{DP} denotes the effective area of dark pixels and can be found from the sensor manual. Once defects appear in the dark pixels, the dark noise $\bar{x}_{DP,HOT}$ is calculated as:

$$\overline{x}_{DP,HOT} = \frac{\sum_{i} x_{DP,i}}{n_{DP}} \approx \overline{x}_{DP} + \frac{n_{DP,HOT} \cdot x_{SIG,\mathrm{MAX}}}{n_{DP}}, \tag{2.33}$$

where n_{DP} represents the number of dark pixels. After black-level compensation, the pixel data can be calculated as:

$$x_{BC,HOT} = x_{SIG} - \overline{x}_{DP,HOT} \approx x_{BC} - \frac{n_{DP,HOT} x_{SIG,\mathrm{MAX}}}{n_{DP}}. \tag{2.34}$$

Furthermore, the interpolation process in Sect. 2.2.2 may introduce spatial dependence between a defective pixel and its neighborhood, as shown in Fig. 2.5. In

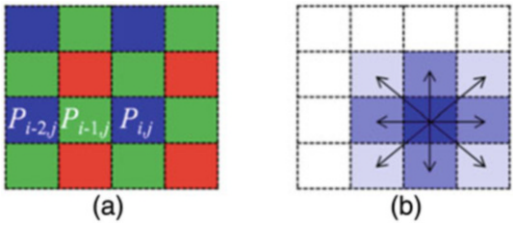


Fig. 2.5 (**a**) A colored image array is shown for illustration purpose. (**b**) A single hot pixel may influence several adjacent pixels due to interpolation

Fig. 2.5a, consider the blue pixel in the middle (i.e., $P_{i,j}$) as an example. Its expected blue value is denoted as $x_{Pi,j}$, and the blue value of its left pixel $x_{Pi-1,j}$ is interpolated by using the two adjacent pixels $P_{i,j}$ and $P_{i-2,j}$:

$$x_{P_{i-1,j}} = \frac{x_{P_{i,j}} + x_{P_{i-2,j}}}{2}, \tag{2.35}$$

If the pixel $P_{i,j}$ is defective, the pixel $P_{i-1,j}$ will get a wrong blue value:

$$x_{P_{i-1,j},HOT} = \frac{x_{SIG,\mathrm{MAX}} + x_{P_{i-2,j}}}{2} = x_{P_{i-1,j}} + \frac{x_{SIG,\mathrm{MAX}} - x_{P_{i,j}}}{2}, \tag{2.36}$$

where the second term represents the error introduced by the hot pixel. Generally, a single hot pixel may influence several adjacent pixels due to interpolation, as shown in Fig. 2.5b.

The aforementioned discussions reveal an important fact that statistical models are required to capture the circuit-level non-idealities due to temperature variation and circuit aging. In the next section, we further propose a novel method for corner case generation in order to simulate the critical corner cases with consideration of circuit-level non-idealities.

2.3.3 Corner Case Generation

To validate an autonomous driving system, a large amount of on-road recordings are often collected. However, most of them are recorded under nominal circuit conditions while the extreme corner cases (e.g., operating at extremely high/low temperature, aging over 10 years, etc.) are difficult to observe. To address this issue, a novel corner case generation method is proposed in this section. Our proposed approach incorporates circuit-level non-idealities with nominal recordings to synthetically generate different corner cases for validation purposes.

We consider the images recorded under nominal conditions to be ideal, i.e., without any defect or dark current. To estimate the nominal raw data $\mathbf{x}_{BC,NOM}$ from the nominal recording $\mathbf{x}_{OUT,NOM}$, the image processing pipeline in (2.12) is considered:

$$\mathbf{x}_{OUT,NOM} = gamma\left(\mathbf{C}_{CC}\mathbf{C}_{WB}\begin{bmatrix}\mathbf{x}_{BC,NOM,R}\mathbf{C}_{IP,R}\\ \mathbf{x}_{BC,NOM,G}\mathbf{C}_{IP,G}\\ \mathbf{x}_{BC,NOM,B}\mathbf{C}_{IP,B}\end{bmatrix}\right). \tag{2.37}$$

Note that all the functions in (2.37) are invertible. Namely, for a given $\mathbf{x}_{OUT,NOM}$, the nominal raw data $\mathbf{x}_{BC,NOM} = [\mathbf{x}_{BC,NOM,R}\ \mathbf{x}_{BC,NOM,G}\ \mathbf{x}_{BC,NOM,B}]$ can be calculated by inverting the image processing pipeline with the following steps.

First, to invert Gamma compensation, the corresponding inverse function $gamma^{-1}(\bullet)$ of the nonlinear mapping $gamma(\bullet)$ in (2.11) can be approximately written as:

$$x_{CC,NOM} = \left(\frac{x_{OUT,NOM}}{\alpha}\right)^{\frac{1}{\gamma}}, \tag{2.38}$$

where $x_{CC,NOM}$ denotes the nominal pixel data before Gamma compensation. By applying (2.38), the nominal image data before Gamma compensation can be estimated as:

$$\mathbf{x}_{CC,NOM} = gamma^{-1}\left(\mathbf{x}_{OUT,NOM}\right). \tag{2.39}$$

Next, we consider color correction, white balance and interpolation in (2.7, 2.8, and 2.9). All of them are modelled as linear functions. Therefore, the nominal image data before interpolation $\mathbf{x}_{BC,NOM}$ can be estimated as:

$$\begin{aligned}
\mathbf{x}_{BC,NOM,R} &= \left(\mathbf{e}_R\mathbf{C}_{WB}^{-1}\mathbf{C}_{CC}^{-1}\mathbf{x}_{CC,NOM}\right)\mathbf{C}_{IP,R}^{T}\left(\mathbf{C}_{IP,R}\mathbf{C}_{IP,R}^{T}\right)^{-1}\\
\mathbf{x}_{BC,NOM,G} &= \left(\mathbf{e}_G\mathbf{C}_{WB}^{-1}\mathbf{C}_{CC}^{-1}\mathbf{x}_{CC,NOM}\right)\mathbf{C}_{IP,G}^{T}\left(\mathbf{C}_{IP,G}\mathbf{C}_{IP,G}^{T}\right)^{-1},\\
\mathbf{x}_{BC,NOM,B} &= \left(\mathbf{e}_B\mathbf{C}_{WB}^{-1}\mathbf{C}_{CC}^{-1}\mathbf{x}_{CC,NOM}\right)\mathbf{C}_{IP,B}^{T}\left(\mathbf{C}_{IP,B}\mathbf{C}_{IP,B}^{T}\right)^{-1}
\end{aligned} \tag{2.40}$$

where $\mathbf{e}_R = [1\ 0\ 0]$, $\mathbf{e}_G = [0\ 1\ 0]$ and $\mathbf{e}_B = [0\ 0\ 1]$ are color selectors.

When the circuit-level non-idealities are taken into account, the perturbation posed on the nominal raw data $\mathbf{x}_{BC,NOM}$ is denoted as $\Delta\mathbf{x}_{BC} = [\Delta\mathbf{x}_{BC,R}\ \Delta\mathbf{x}_{BC,G}\ \Delta\mathbf{x}_{BC,B}]$. Considering temperature variation and circuit aging, we can partition $\Delta\mathbf{x}_{BC}$ into two components:

$$\begin{aligned}
\Delta\mathbf{x}_{BC} = \Delta\mathbf{x}_{BC,DARK} + \Delta\mathbf{x}_{BC,HOT} = \big[&\Delta\mathbf{x}_{BC,DARK,R} + \Delta\mathbf{x}_{BC,HOT,R}\\
&\Delta\mathbf{x}_{BC,DARK,R} + \Delta\mathbf{x}_{BC,HOT,R}\ \ \Delta\mathbf{x}_{BC,DARK,R} + \Delta\mathbf{x}_{BC,HOT,R}\big]
\end{aligned}, \tag{2.41}$$

where $\Delta\mathbf{x}_{BC,DARK}$ and $\Delta\mathbf{x}_{BC,HOT}$ represent the perturbation components due to dark currents and hot pixels, respectively.

First, we study the perturbation of dark signal $\Delta x_{BC,DARK}$ for each pixel. Based on (2.25), it is straightforward to derive:

$$\Delta x_{BC,DARK} = x_{DARK} - \overline{x}_{DP}. \tag{2.42}$$

In (2.42), we assume that the dark signal under nominal condition is negligible. To calculate $\Delta x_{BC,DARK}$, the dark signal x_{DARK} is estimated by using (2.24). When designing a CMOS sensor, the maximum voltage v_{MAX} in (2.24) is often set to match the analog output corresponding to a full well with n_{WELL} photoelectrons at the default ISO level $G_{ISO,DEF}$ specified by the manufacturer. Based on (2.1 and 2.2), v_{MAX} can be written as:

$$v_{MAX} = G_{ISO,DEF} G_{FIX} \left(\frac{G_{PIXEL} q}{C} \right) n_{WELL}. \tag{2.43}$$

Substituting (2.43) into (2.24) yields:

$$x_{DARK} = \text{int} \left[\frac{G_{ISO}}{G_{ISO,DEF} n_{WELL}} n_{DARK} \times 2^{N_{ADC}} \right]. \tag{2.44}$$

In (2.44), the values of $G_{ISO,DEF}$, n_{WELL}, and N can be found from the sensor manual and n_{DARK} can be calculated by following the Poisson distribution in (2.22). However, G_{ISO} is manually or automatically specified during image recording and may differ from image to image. The ISO level of an image cannot be determined from a pre-recorded image stored in a compressed format such as JPEG or AVI. To provide a conservative estimation, multiple corner cases corresponding to different ISO levels are generated for validation purposes. Based on the aforementioned discussions, we can estimate the output signal for each dark pixel in a similar way, calculate the dark noise $\bar{x}_{DP}$, and eventually determine $\Delta \mathbf{x}_{BC,DARK}$ by using (2.42).

Next, we calculate $\Delta \mathbf{x}_{BC,HOT}$ with consideration of circuit aging where the numbers of defects in active pixels and dark pixels, denoted as $n_{AP,HOT}$ and $n_{DP,HOT,}$ respectively, are estimated by following the Poisson distribution in (2.30 and 2.31). These defective pixels are randomly distributed as discussed in Sect. 2.3.2. Based on (2.34), the perturbation for each pixel after black-level compensation can be calculated as:

$$\Delta x_{BC,HOT} = -\frac{n_{DP,HOT} x_{MAX}}{n_{DP}}, \tag{2.45}$$

where n_{DP} can be found from the sensor manual. Substituting $\Delta \mathbf{x}_{BC,DARK}$ and $\Delta \mathbf{x}_{BC,HOT}$ into (2.41) yields the total perturbation $\Delta \mathbf{x}_{BC}$.

Once both $\mathbf{x}_{BC,NOM}$ and $\Delta \mathbf{x}_{BC}$ are obtained, the corresponding image data with perturbation after color correction $\mathbf{x}_{CC,PTB}$ can be directly calculated using (2.7, 2.8, and 2.9):

$$\mathbf{x}_{CC,PTB} = \mathbf{C}_{CC} \mathbf{C}_{WB} \begin{bmatrix} \left(\mathbf{x}_{BC,NOM,R} + \Delta \mathbf{x}_{BC,R} \right) \mathbf{C}_{IP,R} \\ \left(\mathbf{x}_{BC,NOM,G} + \Delta \mathbf{x}_{BC,G} \right) \mathbf{C}_{IP,G} \\ \left(\mathbf{x}_{BC,NOM,B} + \Delta \mathbf{x}_{BC,B} \right) \mathbf{C}_{IP,B} \end{bmatrix}. \tag{2.46}$$

Substituting (2.40) into (2.46) yields:

$$\mathbf{x}_{CC,PTB} = \mathbf{C}_{CC}\mathbf{C}_{WB}\begin{bmatrix}\mathbf{x}_{BC,NOM,R}\mathbf{C}_{IP,R}\\ \mathbf{x}_{BC,NOM,G}\mathbf{C}_{IP,G}\\ \mathbf{x}_{BC,NOM,B}\mathbf{C}_{IP,B}\end{bmatrix} + \mathbf{C}_{CC}\mathbf{C}_{WB}\begin{bmatrix}\Delta\mathbf{x}_{BC,R}\mathbf{C}_{IP,R}\\ \Delta\mathbf{x}_{BC,G}\mathbf{C}_{IP,G}\\ \Delta\mathbf{x}_{BC,B}\mathbf{C}_{IP,B}\end{bmatrix}$$

$$= \mathbf{x}_{CC,NOM} + \mathbf{C}_{CC}\mathbf{C}_{WB}\begin{bmatrix}\Delta\mathbf{x}_{BC,R}\mathbf{C}_{IP,R}\\ \Delta\mathbf{x}_{BC,G}\mathbf{C}_{IP,G}\\ \Delta\mathbf{x}_{BC,B}\mathbf{C}_{IP,B}\end{bmatrix}. \tag{2.47}$$

Equation (2.47) reveals an important fact that there is no need to explicitly estimate the image data $\mathbf{x}_{BC,NOM}$ in (2.40) in order to calculate $\mathbf{x}_{CC,PTB}$ in (2.47). Instead, we only need to know the nominal image data $\mathbf{x}_{CC,NOM}$ and the perturbation term due to circuit-level non-idealities:

$$\Delta\mathbf{x}_{CC} = \mathbf{C}_{CC}\mathbf{C}_{WB}\begin{bmatrix}\Delta\mathbf{x}_{BC,R}\mathbf{C}_{IP,R}\\ \Delta\mathbf{x}_{BC,G}\mathbf{C}_{IP,G}\\ \Delta\mathbf{x}_{BC,B}\mathbf{C}_{IP,B}\end{bmatrix}. \tag{2.48}$$

Combining (2.47 and 2.48) yields:

$$\mathbf{x}_{CC,PTB} = \mathbf{x}_{CC,NOM} + \Delta\mathbf{x}_{CC}. \tag{2.49}$$

In (2.38), the matrices $\mathbf{C}_{CC}$, $\mathbf{C}_{IP,R}$, $\mathbf{C}_{IP,G}$, and $\mathbf{C}_{IP,B}$ are known for a given image sensor; however, the matrix $\mathbf{C}_{WB}$ is often unknown and cannot be determined from pre-reordered images. Similar to the ISO level, we generate multiple corner cases with different white balance settings to provide a conservative estimation. After the "contaminated" image data $\mathbf{x}_{CC,PTB}$ is calculated according to (2.49), Gamma compensation is applied by using (2.10):

$$\mathbf{x}_{OUT,PTB} = gamma\left(\mathbf{x}_{CC,PTB}\right). \tag{2.50}$$

Algorithm 2.1 summarizes the major steps of the proposed corner case generation. Once the corner cases are generated, a vision-based autonomous driving system can then be validated based on these cases. For a given case, the autonomous driving system fails, if it cannot produce the correct outcome. In the next section, we will propose a novel SUS method to efficiently evaluate the rare failure rate of an autonomous driving system.

Algorithm 2.1: Corner Case Generation

1. Identify the values for τ, v, t_{EXP}, A_{PH}, $G_{ISO,DEF}$, n_{WELL}, N_{ADC}, A_{AP}, A_{DP}, and n_{DP} from the sensor manual.
2. Given an input nominal recording, denoted as $\mathbf{x}_{OUT,NOM}$, calculate the nominal image data before Gamma compensation $\mathbf{x}_{CC,NOM}$ based on (2.39).
3. Calculate perturbation $\Delta\mathbf{x}_{BC,DARK}$ with consideration of temperature variation by using (2.42, 2.43, and 2.44) with different ISO levels.

4. Calculate perturbation $\Delta \mathbf{x}_{BC,HOT}$ with consideration of circuit aging by using (2.45) with different ISO levels.
5. Calculate total perturbation $\Delta \mathbf{x}_{BC}$ by combining $\Delta \mathbf{x}_{BC,DARK}$ and $\Delta \mathbf{x}_{BC,HOT}$ in (2.41).
6. Calculate perturbation $\Delta \mathbf{x}_{CC}$ by using (2.48) with different white balance settings.
7. Calculate the "contaminated" image data $\mathbf{x}_{CC,PTB}$ before Gamma compensation by using (2.49).
8. Apply Gamma compensation on $\mathbf{x}_{CC,PTB}$ by using (2.50) to generate corner cases.

2.3.4 Numerical Experiments

In this section, we demonstrate the impact of circuit-level non-idealities by taking a STOP sign detection system as the example. All the numerical experiments are performed on a cluster composed of 500 nodes where each node is a workstation with 1.67 GHz CPU and 4GB memory.

2.3.4.1 Experimental Setup

We consider a CMOS image sensor designed in a 90 nm process. The image resolution is 1280 × 720 pixels. Table 2.1 shows the major parameters of the image sensor. To appropriately cover all possible corner cases, we assume that the ISO level of the image sensor varies from 100 to 800 and the reference color temperature for white balance varies from 2000 K to 10,000 K.

A STOP sign detection system is implemented with three-stage cascade classifiers that are trained in [22, 28] by using the MATLAB toolbox PMT [29] based on three public databases: BelgiumTS [30], GTSRB [31], and GTSDB [32]. Each image block is converted to grayscale and resized to 32 × 32 pixels. Therefore, each

Table 2.1 Image sensor settings

Parameter	Symbol	Value	Parameter	Symbol	Value
Generation lifetime	τ	2 ms	Well capacity	n_{WELL}	20,000
Applied voltage	v	−1.2 V	ADC resolution	N	12 bit
Exposure time	t_{EXP}	1/60s	Area of active pixels	A_{AP}	1280 × 720 × A_{PH}
Pixel size	A_{PH}	1.4 μm × 1.4 μm	Area of dark pixels	A_{DP}	25 × A_{PH}
Default ISO level	$G_{\mathrm{ISO,DEF}}$	1	Number of dark pixels	n_{DP}	25

Table 2.2 Number of features for each stage of cascade classifiers

Stage	1st-stage	2nd-stage	3rd-stage	Overall
# of features	16	48	144	208

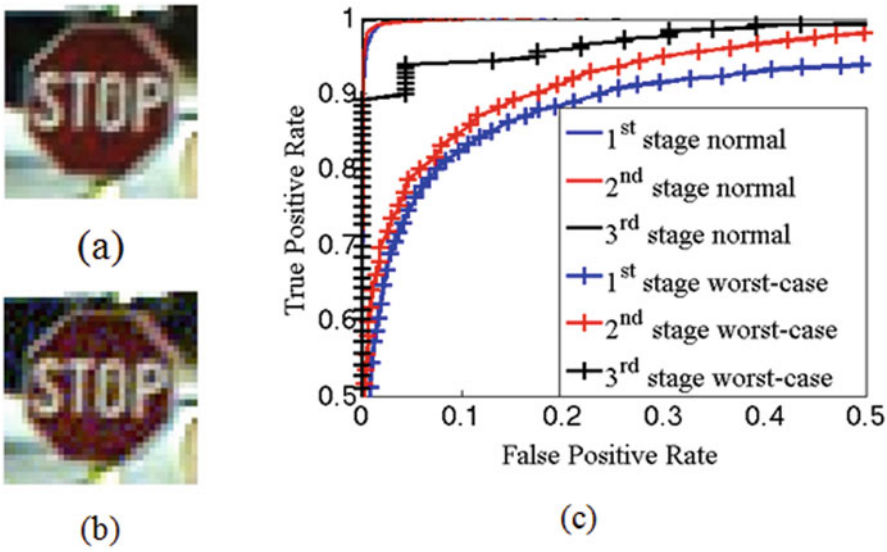


Fig. 2.6 (**a**) A stop sign is recorded under the nominal condition without dark current. (**b**) A corner case is synthetically generated with the temperature of 400 K, the reference color temperature for white balance of 6500 K and the ISO level of 100. (**c**) Receiver operating characteristic (ROC) curves are shown for both the nominal and worst cases with consideration of temperature variation

input data point for the detection system is a vector containing $32 \times 32 = 1024$ pixels. Features are extracted for STOP sign detection, as described in Sect. 2.2.3. The number of features for each stage is shown in Table 2.2.

The aforementioned detection system is appropriately trained using the images collected under nominal conditions. For each nominal image, our proposed flow is applied for worst-case validation. Namely, a set of different corner cases with consideration of temperature variation, circuit aging, ISO level variation, and white balance is tested. If the detection system fails to generate the correct output at any corner, a failure case is reported. Our objective is to compare the nominal accuracy against the worst-case accuracy in order to quantitatively demonstrate the impact of circuit-level non-idealities.

2.3.4.2 Temperature Variation

Figure 2.6c shows the receiver operating characteristic (ROC) curves of three-stage classifiers for both the nominal and worst cases. For the nominal case, the false

Table 2.3 Accuracy comparison with consideration of temperature variation

Failure rate		1st-stage (%)	2nd-stage (%)	3rd-stage (%)	Overall (%)
Nominal case	TP	99.90	99.95	99.84	99.69
	FP	7.18	8.04	5.34	0.03
Worst case	TP	81.82	95.82	93.37	73.20
	FP	8.97	34.89	4.35	0.14

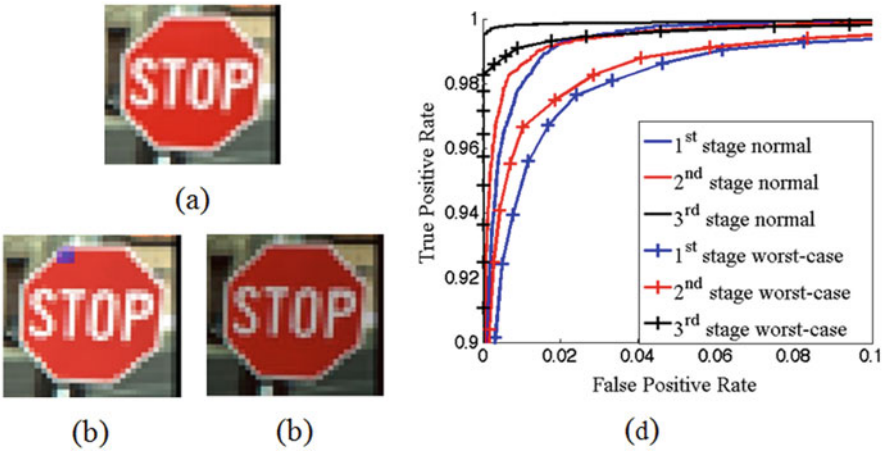


Fig. 2.7 (**a**) A stop sign is recorded under the nominal condition without any defect. (**b**) A corner case is synthetically generated with a defect in an active pixel of blue color. (**c**) A corner case is synthetically generated with a defect in a dark pixel. (**d**) ROC curves are shown for both the nominal and worst cases with consideration of circuit aging

positive rates are close to 0 and the true positive rates are close to 1 in all three stages, meaning that these classifiers are accurate in detecting both positives and negatives cases. However, the ROC curves are substantially degraded in worst cases. Given the same false positive rate, the corresponding true positive rate is greatly reduced, thereby demonstrating the significant impact posed by temperature variation. Table 2.3 numerically compares the false positive rates and true positive rates at a given setting. The overall true positive rate of the detection system is reduced from 99.69% to 73.20% in the worst case.

2.3.4.3 Circuit Aging

Figure 2.7d shows the ROC curves for both nominal and worst cases with consideration of circuit aging. In this example, the worst-case ROC curves slightly deviate from the nominal ones. However, to guarantee safety for an automotive system, the effect of circuit aging cannot be simply ignored. Table 2.4 numerically compares the false positive rates and true positive rates at a given setting. The overall true

Table 2.4 Accuracy comparison with consideration of circuit aging

Failure rate		1st (%)	2nd (%)	3rd (%)	Total
Nominal case	TP	99.90	99.95	99.84	99.69%
	FP	7.18	8.04	5.34	3.08×10^{-4}
Worst case	TP	99.31	99.38	99.08	97.78%
	FP	8.27	9.24	6.17	4.7×10^{-4}

positive rate of the detection system is reduced from 99.69% to 97.78% in the worst case. Alternatively speaking, the corresponding false negative rate increases from (1–99.69%) = 0.31% to (1–97.78%) = 2.22% (7× larger).

2.4 Subset Simulation

2.4.1 Mathematical Formulation

The failure rate of a machine learning system is the probability P_f that the system receives an input (e.g., an image block) and produces the wrong outcome. Such an input is referred to as the failure point in this chapter. The set containing all failure points, represented by Ω, is referred to as the failure region of the system. If the failure probability is estimated by the brute-force Monte Carlo method, we randomly generate N independent samples $\{\mathbf{x}_n; n = 1, 2, \ldots, N\}$ and then calculate:

$$P_f = P(\mathbf{x} \in \Omega) \approx \frac{N_f}{N}, \tag{2.51}$$

where $P(\bullet)$ is the probability of an event and N_f is the number of failure points.

If the failure rate is extremely small (e.g. 10^{-12}), a large number of samples (e.g. 10^{13}) must be generated for high accuracy, resulting in expensive computational cost. To address this, we adopt SUS [14, 15] to efficiently evaluate the rare failure rate. SUS defines a sequence of intermediate failure regions:

$$\Omega = \Omega_K \subset \Omega_{K-1} \subset \cdots \subset \Omega_1, \tag{2.52}$$

where Ω_K is identical to the original failure region Ω. A 2-D example with three intermediate failure regions (i.e., $K = 3$) is shown in Fig. 2.8 for illustration purpose.

Considering the discriminant function in (2.13), we can define the intermediate failure regions by adjusting the threshold value δ. When the false-detection rate is estimated, a smaller threshold value results in more false-detection events and, hence, a larger failure region. The intermedia failure regions can be defined by choosing a sequence of threshold values:

$$\delta = \delta_K > \delta_{K-1} > \cdots > \delta_1, \tag{2.53}$$

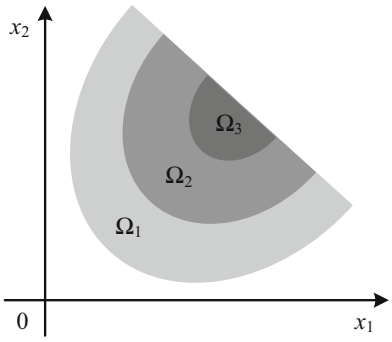


Fig. 2.8 A simple 2-D example with three intermediate failure regions is shown for illustration purpose where $\Omega = \Omega_3 \subset \Omega_2 \subset \Omega_1$

where δ_k denotes the threshold value corresponding to the failure region Ω_k. To estimate the miss-detection rate, a similar sequence of threshold values can be chosen:

$$\delta = \delta_K < \delta_{K-1} < \cdots < \delta_1. \tag{2.54}$$

After the intermediate failure regions are defined, the failure rate P_f can be computed as the product of a set of conditional probabilities:

$$P_f = P(\mathbf{x} \in \Omega_1) \cdot \prod_{k=2}^{K} P(\mathbf{x} \in \Omega_k | \mathbf{x} \in \Omega_{k-1}). \tag{2.55}$$

If each Ω_k is appropriately chosen, each conditional probability in (2.55) is relatively large and can be accurately estimated by a small number of samples:

$$P(\mathbf{x} \in \Omega_k | \mathbf{x} \in \Omega_{k-1}) \approx \frac{N_f^{(k)}}{N^{(k)}}, \tag{2.56}$$

where $N^{(k)}$ is the number of samples inside the failure region Ω_{k-1} and ${N_f}^{(k)}$ is the number of samples inside the failure region Ω_k.

However, randomly sampling a given failure region is non-trivial in practice. For our application of interest, a large amount of input data (e.g. image blocks) are often given in advance to validate the system. Therefore, random sampling must be performed for these "discrete" data points, while the conventional SUS method is designed to sample continuous random variables only [14, 15]. In what follows, we will propose a novel sampling scheme to address this issue.

2.4.2 Random Sampling

To compute the small failure rate P_f by using (2.55), we need random samples for each failure region. To estimate $P(\mathbf{x} \in \Omega_1)$, we use the brute-force Monte Carlo

method. Given a set of image blocks $\{\mathbf{x}_n; n = 1, 2, \ldots, N\}$, we randomly choose $N^{(1)}$ image blocks from dataset. For each chosen image block $\mathbf{x}_n$, we calculate:

$$f(\mathbf{x}_n) \begin{cases} \geq \delta_1 \ (Traffic\ sign\ detected) \\ < \delta_1 \ (Traffic\ sign\ undetected) \end{cases}, \tag{2.57}$$

to determine whether $\mathbf{x}_n$ is in the failure region Ω_1 or not. Next, the failure rate $P(\mathbf{x} \in \Omega_1)$ is calculated as:

$$P(\mathbf{x} \in \Omega_1) \approx \frac{N_f^{(1)}}{N^{(1)}}. \tag{2.58}$$

Because the failure rate $P(\mathbf{x} \in \Omega_1)$ is substantially greater than P_f, only a small number of samples are needed to accurately estimate $P(\mathbf{x} \in \Omega_1)$.

Once the failure rate $P(\mathbf{x} \in \Omega_1)$ is estimated, we should further estimate the conditional probabilities $P(\mathbf{x} \in \Omega_k \mid \mathbf{x} \in \Omega_{k-1})$ $(k = 2, 3, \ldots, K)$. Considering $P(\mathbf{x} \in \Omega_2 \mid \mathbf{x} \in \Omega_1)$ as an example, we need to generate samples inside Ω_1. Even though $N_f{}^{(1)}$ failure points are already generated for Ω_1 when calculating $P(\mathbf{x} \in \Omega_1)$, those data points are not sufficient and we must collect additional samples inside Ω_1. Towards this goal, we adopt the idea of MCMC described in [20]. We assume that extra failure points for Ω_1 can be found in the neighborhood of known failure points. Hence, starting from $N_f{}^{(1)}$ failure points $\{\mathbf{x}_n; n = 1, 2, \ldots, N_f{}^{(1)}\}$, we explore other data points around $\mathbf{x}_n$ and check if they belong to the failure region Ω_1.

Such a simple idea, however, is non-trivial to implement, because the given dataset $\{\mathbf{x}_n; n = 1, 2, \ldots, N\}$ is high-dimensional and extremely large. Hence, searching the neighborhood for a given point $\mathbf{x}_n$ is computationally expensive. In order to address this complexity issue, we map the data points to an undirected graph consisting of a uniform grid. Using this graph, the neighborhood of a vertex can be explored with low computational cost by randomly walking over the vertices through the graph edges. Figure 2.9 shows a simple 2-D example where four data points are mapped to a 2-D uniform grid.

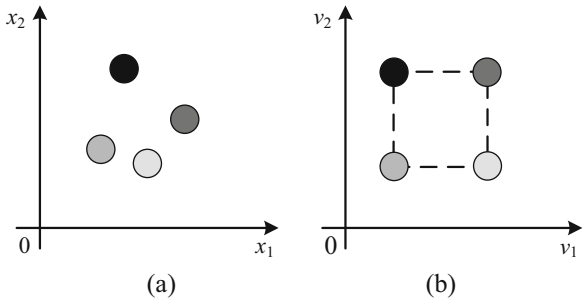


Fig. 2.9 A simple 2-D example where the data points are mapped to a 2-D uniform grid: (**a**) four data points and (**b**) a uniform 2-D with four vertices. Different data points and vertices are shown in different colors

To implement this graph mapping idea, we investigate the probability formulation of MCMC [20]. Starting from a data point $\mathbf{x}_A$, the probability of reaching a new point $\mathbf{x}_B$ in the neighborhood is [20]:

$$P\left(\mathbf{x}_B|\mathbf{x}_A\right) \propto \exp\left(-s_1 \cdot \|\mathbf{x}_A - \mathbf{x}_B\|_2^2\right), \tag{2.59}$$

where $||\bullet||_2$ represents the L2 norm of a vector and s_1 is a scaling factor. Equation (2.49) implies that the probability of reaching $\mathbf{x}_B$ is high if $\mathbf{x}_B$ is close to $\mathbf{x}_A$. Namely, we want to search the neighborhood of the failure point $\mathbf{x}_A$, because it is likely to observe other failure points in its neighborhood. By introducing another scaling factor s_2, we can re-write (2.59) as:

$$P\left(\mathbf{x}_B|\mathbf{x}_A\right) = \exp\left(s_2\right) \cdot \exp\left(-s_1 \cdot \|\mathbf{x}_A - \mathbf{x}_B\|_2^2\right). \tag{2.60}$$

After the data points are mapped to a graph, we randomly walk through the graph by starting from a vertex (say, $\mathbf{v}_A$). The probability to reach a new vertex $\mathbf{v}_B$ is approximately equal to:

$$P\left(\mathbf{v}_B|\mathbf{v}_A\right) = {P_0}^{d(\mathbf{v}_A, \mathbf{v}_B)}, \tag{2.61}$$

where P_0 represents the probability of walking through a single edge in the graph, and $d(\mathbf{v}_A, \mathbf{v}_B)$ denotes the "distance" between $\mathbf{v}_A$ and $\mathbf{v}_B$ measured by the number of edges between them. In (2.61), we assume that all edges are assigned with the same probability P_0 for random walk. The distance $d(\mathbf{v}_A, \mathbf{v}_B)$ can be mathematically modeled by the L1 norm up to a scaling factor s_3:

$$d\left(\mathbf{v}_A, \mathbf{v}_B\right) = s_3 \cdot \|\mathbf{v}_A - \mathbf{v}_B\|_1, \tag{2.62}$$

where $||\bullet||_1$ represents the L1 norm of a vector.

In order to implement MCMC by using random walk based on this graph, we must match the probability models in (2.60 and 2.61). Taking the logarithm on both sides of (2.60 and 2.61), we have:

$$\log\left(P\left(\mathbf{x}_B|\mathbf{x}_A\right)\right) = -s_1 \cdot \|\mathbf{x}_A - \mathbf{x}_B\|_2^2 + s_2, \tag{2.63}$$

$$\log\left(P\left(\mathbf{v}_B|\mathbf{v}_A\right)\right) = \log\left(P_0\right) \cdot d\left(\mathbf{v}_A, \mathbf{v}_B\right). \tag{2.64}$$

Substituting (2.62) into (2.64) and then matching (2.63 and 2.64) yield:

$$s_1 \cdot \|\mathbf{x}_A - \mathbf{x}_B\|_2^2 - s_2 = -\log\left(P_0\right) \cdot s_3 \cdot \|\mathbf{v}_A - \mathbf{v}_B\|_1. \tag{2.65}$$

Equation (2.65) can be re-written as:

$$s_4 \cdot \|\mathbf{x}_A - \mathbf{x}_B\|_2^2 - s_5 = \|\mathbf{v}_A - \mathbf{v}_B\|_1, \tag{2.66}$$

where

$$s_4 = -\frac{s_1}{\log(P_0) \cdot s_3}, \tag{2.67}$$

$$s_5 = -\frac{s_2}{\log(P_0) \cdot s_3}. \tag{2.68}$$

Equation (2.66) indicates that the distance $||\mathbf{x}_A - \mathbf{x}_B||_2{}^2$ between data points $\mathbf{x}_A$ and $\mathbf{x}_B$ should be matched to the distance $||\mathbf{v}_A - \mathbf{v}_B||_1$ between graph vertices $\mathbf{v}_A$ and $\mathbf{v}_B$ up to two scaling factors s_4 and s_5. In other words, when mapping the data points to the graph, we should preserve the pairwise distance between these data points.

To mathematically formulate the graph mapping problem, we define the distance matrix $\mathbf{D}_x$ for the data points:

$$\mathbf{D}_x = \begin{bmatrix} \|\mathbf{x}_1 - \mathbf{x}_1\|_2^2 & \cdots & \|\mathbf{x}_1 - \mathbf{x}_N\|_2^2 \\ \vdots & \cdots & \vdots \\ \|\mathbf{x}_N - \mathbf{x}_1\|_2^2 & \cdots & \|\mathbf{x}_N - \mathbf{x}_N\|_2^2 \end{bmatrix}. \tag{2.69}$$

Similarly, we define the distance matrix $\mathbf{D}_v$ for the graph vertices:

$$\mathbf{D}_v = \begin{bmatrix} \|\mathbf{v}_1 - \mathbf{v}_1\|_1 & \cdots & \|\mathbf{v}_1 - \mathbf{v}_N\|_1 \\ \vdots & \cdots & \vdots \\ \|\mathbf{v}_N - \mathbf{v}_1\|_1 & \cdots & \|\mathbf{v}_N - \mathbf{v}_N\|_1 \end{bmatrix}. \tag{2.70}$$

Our goal is to find the optimal one-to-one mapping between each data point and each graph vertex. The mapping can be mathematically encoded by a permutation applied to the graph vertices. Namely, we aim to find the optimal permutation so that the m-th graph vertex is matched to the m-th data point after permutation. When the graph vertices are permutated, the distance matrix in (2.70) can be re-written as:

$$\mathbf{P}^T \cdot \mathbf{D}_v \cdot \mathbf{P}, \tag{2.71}$$

where $\mathbf{P}$ is a permutation matrix.

Matching the distance matrices in (2.69 and 2.71) according to (2.66) yields:

$$\underset{s_4, s_5, \mathbf{P}}{\text{minimize}} \left\| s_4 \cdot \mathbf{D}_x - s_5 - \mathbf{P}^T \cdot \mathbf{D}_v \cdot \mathbf{P} \right\|_F, \quad \text{subject to } \mathbf{P} \in \Phi \tag{2.72}$$

where $||\bullet||_F$ denotes the Frobenius norm of a matrix and Φ denotes the set containing all permutation matrices. The optimization problem in (2.72) is NP-hard [33] and, hence, solving it is non-trivial. To address this issue, several novel heuristics are

applied to efficiently find a sub-optimal solution for (2.72), and consequently, find the mapping between data points and graph vertices. The pairwise distance is accurately captured in a low-dimensional space [34–37]. Hence, the graph mapping task is formulated as a matching problem between two reduced-size distance matrices, and it can be efficiently solved by a fast matching scheme based on K-D tree [38]. For more details, please refer to [22].

2.4.3 Summary

The overall flow for our proposed SUS method is summarized in Algorithm 2.2. Note that the first step depends on the dataset $\{\mathbf{x}_n;\ n = 1, 2, \ldots, N\}$ only and is independent of the machine learning system under validation. The step can be finished in advance before the machine learning system is designed.

Algorithm 2.2: Failure Rate Estimation by Subset Sampling

1. Start from a given dataset $\{\mathbf{x}_n;\ n = 1, 2, \ldots, N\}$ and a given uniform grid $\{\mathbf{v}_n;\ n = 1, 2, \ldots, N\}$, and match their column vectors based on K-D tree [22].
2. Estimate the failure probability $P(\mathbf{x} \in \Omega_1)$ in (2.58) by using the brute-force Monte Carlo method.
3. Generate a set of random samples in each failure region $\{\Omega_k;\ k = 2, 3, \ldots, K\}$ by random walk based on the uniform grid.
4. Estimate the conditional probabilities $\{P(\mathbf{x} \in \Omega_k \mid \mathbf{x} \in \Omega_{k-1});\ k = 2, 3, \ldots, K\}$ by using (2.56).
5. Compute the rare failure rate P_f by using (2.55).

2.4.4 Numerical Experiments

In this section, we demonstrate the efficacy of our proposed SUS method by a STOP sign detection system. All experiments are performed on a Linux server with a Xeon(R) E5649 2.53GHz CPU and 64 GB memory.

2.4.4.1 Experimental Setup

To estimate the rare false-detection rate for the STOP sign detection system, we collect a set of test data based on the videos taken by front-view cameras on vehicles. No image block in the test dataset contains a STOP sign. The total number of image blocks is equal to $2^{23} = 8{,}388{,}608$. For testing and comparison purposes, we exhaustively evaluate all image blocks by our STOP sign detection system and the estimated "golden" failure rate is equal to 10^{-5}.

Table 2.5 Failure rate estimation results for SUS

Failure region	# of classifier evaluations	# of unique failure points	"Golden" conditional failure rate	Estimated conditional failure rate
Ω_1	1000	1000	1.1×10^{-1}	1.0×10^{-1}
Ω_2	3345	753	0.9×10^{-1}	1.0×10^{-1}
Ω_3	4189	158	0.8×10^{-1}	1.0×10^{-1}
Ω_4	601	19	0.6×10^{-1}	1.1×10^{-1}
Ω	46	2	8.0×10^{-1}	5.0×10^{-1}

When applying our proposed SUS algorithm to estimate the rare failure rate, we map the aforementioned 8,388,608 image blocks to a 23-D uniform grid. The coordinate of each vertex in the grid is expressed as:

$$\mathbf{v}_n = \left[v_{n,1} \; v_{n,2} \cdots v_{n,23} \right]^T, \tag{2.73}$$

where $v_{n,i} \in \{0, 1\}$ $(i = 1, 2, \ldots, 23)$ is a binary variable. Since the uniform grid is 23-dimensional, it contains $2^{23} = 8{,}388{,}608$ vertices in total. The total number of vertices in the uniform grid is equal to the total number of image blocks in the test dataset. In this example, we empirically choose $\varepsilon = 10\%$ as the desired value for $P(\mathbf{x} \in \Omega_1)$ and $\{P(\mathbf{x} \in \Omega_k \mid \mathbf{x} \in \Omega_{k-1}); k = 2, 3, \ldots, K\}$.

2.4.4.2 Experimental Results

Table 2.5 shows the failure rate estimation results for SUS, including the numbers of classifier evaluations, the numbers of unique failure points, the "golden" conditional failure rates and the estimated conditional failure rates. The final estimated failure rate is 5.4×10^{-5} and it is close to the "golden" failure rate (i.e., 10^{-5}) in this example. Here, the number of classifier evaluations is an important metric that represents the overall computational cost for system validation.

To investigate the trade-off between estimation accuracy and computational cost, we vary the number of classifier evaluations of SUS by changing the number of random samples in all failure regions $\{\Omega_k; k = 1, 2, \ldots, K\}$. Figure 2.10 shows the estimation errors corresponding to those different numbers of classifier evaluations. It can be observed that the estimation error decreases, as the number of classifier evaluations increases. This observation is consistent with our intuition. Namely, Monte Carlo analysis becomes more accurate, as more random samples are used.

To further demonstrate the estimation accuracy and computational cost of SUS, we repeatedly run SUS 1000 times. Figure 2.11a, b shows the histograms for the numbers of classifier evaluations and the estimated failure rates corresponding to these 1000 runs. Based on Fig. 2.11b, the 95% confidence interval is $[1.1 \times 10^{-5}, 7.7 \times 10^{-5}]$ and the average number of classifier evaluations is 9.9×10^3. To achieve the same accuracy, the brute-force Monte Carlo method requires 1.5×10^5

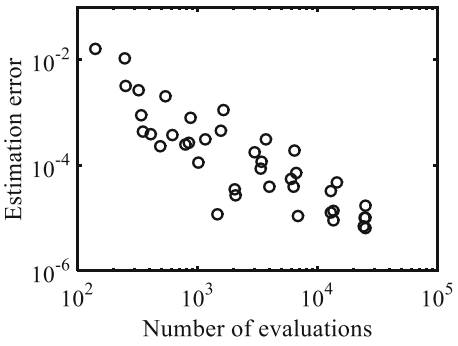


Fig. 2.10 The estimation errors are shown for different numbers of classifier evaluation

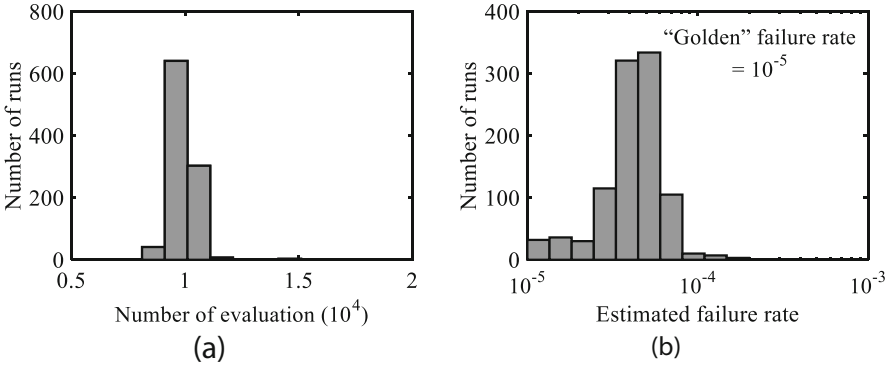


Fig. 2.11 (**a**) The histogram for the numbers of classifier evaluations is shown for 1000 runs. (**b**) The histogram for the estimated failure rates is shown for 1000 runs

random samples (i.e., 1.5×10^5 classifier evaluations). Hence, our proposed SUS method achieves $15.2\times$ runtime speed-up over the conventional brute-force Monte Carlo method in this example.

2.5 Conclusions

In this chapter, we first propose a novel method to effectively validate vision-based autonomous driving systems over different circuit corners with consideration of circuit-level non-idealities, e.g., temperature variation and circuit aging. The proposed approach synthetically generates the critical corner cases that are difficult to record by physical experiments. Next, we develop a novel SUS method to efficiently estimate the rare failure rate for the automotive machine learning system. Towards this goal, we propose a novel MCMC algorithm based on graph mapping. A number of heuristic methods are developed to make our proposed approach computationally efficient and, hence, practically feasible. From our experimental

results, the circuit-level non-idealities show a significant impact on the overall performance of autonomous driving systems, and to evaluate the rare failure rate of the autonomous driving system, our SUS achieves 15.2× runtime speed-up over the conventional brute-force Monte Carlo method without surrendering any accuracy.

Acknowledgments This research work has been supported in part by Toyota InfoTechnology Center, Cadence Design Systems, National Key Research and Development Program of China 2016YFB0201304, and National Natural Science Foundation of China (NSFC) research project 61376040, 61574046 and 61674042.

References

1. A. Lindgren, F. Chen, in State of the art analysis: an overview of advanced driver assistance systems (ADAS) and possible human factors issues. Human Factors and Economics Aspects on Safety (2006), pp. 38–50
2. M. Aeberhard et al., Experience, results and lessons learned from automated driving on Germany's highways. IEEE Intell. Transp. Syst. Mag. **7**(1), 42–57 (2015)
3. J. Ziegler et al., Making bertha drive—an autonomous journey on a historic route. IEEE Intell. Transp. Syst. Mag. **6**(2), 8–20 (2014)
4. M. Fu, Y. Huang, in A survey of traffic sign recognition. *ICWAPR* (2010), pp. 119–124
5. N. Kalra, S.M. Paddock, Driving to safety: how many miles of driving would it take to demonstrate autonomous vehicle reliability. Transp. Res. A Pol. Pract. **94**, 182–193 (2016)
6. D. Hospach et al., in Simulation of falling rain for robustness testing of video-based surround sensing systems. *DATE* (2016), pp. 233–236
7. R. Gallen et al., Nighttime visibility analysis and estimation method in the presence of dense fog. IEEE Trans. Intell. Transp. Syst. **16**(1), 310–320 (2015)
8. M. Nentwig, M. Stamminger, in Hardware-in-the-loop testing of computer vision based driver assistance systems. *IEEE Intelligent Vehicles Symposium* (2011), pp. 339–344
9. M. Nentwig et al., in Concerning the applicability of computer graphics for the evaluation of image processing algorithms. *ICVES* (2012), pp. 205–210
10. A. Broggi et al., Extensive tests of autonomous driving technologies. IEEE Trans. Intell. Transp. Syst. **14**(3), 1403–1415 (2013)
11. E. Roth, T. Calapoglu, in Advanced driver assistance system testing using OptiX. NVIDIA GTC (2012)
12. L. Raffaëlli et al., in Facing ADAS validation complexity with usage oriented testing. ERTS (2016)
13. M. Lankamp et al., MGSim-simulation tools for multi-core processor architectures, arXiv (2013)
14. S. Au, J. Beck, Estimation of small failure probabilities in high dimensions by subset simulation. Probabilist. Eng. Mech. **16**(4), 263–277 (2001)
15. S. Sun, X. Li, in Fast statistical analysis of rare circuit failure events via subset simulation in high-dimensional variation space. ICCAD (2014), pp. 324–331
16. T. Kuroda, *Essential Principles of Image Sensors* (CRC Press, Boca Raton, 2014)
17. G.C. Holst, *CCD Arrays, Cameras, and Displays* (JCD Publication, Winter Park, 1998)
18. R. Ramanath et al., Color image processing pipeline. IEEE Signal Process. Mag. **22**(1), 34–43 (2005)
19. C.M. Kyung, *Theory and Applications of Smart Cameras* (Springer, Dordrecht, 2016)
20. C. Bishop, *Pattern Recognition and Machine Learning* (Prentice Hall, Upper Saddle River, 2007)

21. P. Viola, M. Jones, in Rapid object detection using a boosted cascade of simple features. CVPR (2001), pp. I511–I518
22. W. Shi et al., in Efficient statistical validation of machine learning systems for autonomous driving. ICCAD (2016)
23. X. Xi et al., in *BSIM4.3.0 MOSFET Model—User's Manual*, vol. 2, no. 1 (University of California, Berkley, 2003), pp. 351–354
24. P. Seitz, A.J. Theuwissen, *Single-Photon Imaging* (Springer, Berlin, 2011)
25. G.G. Nampoothiri et al., in Ageing effects on image sensors due to terrestrial cosmic radiation. IS&T/SPIE Electronic Imaging, vol. 7875 (2011)
26. G.H. Chapman et al., in Empirical formula for rates of hot pixel defects based on pixel size, sensor area and ISO. IS&T/SPIE Electronic Imaging, vol. 8659, (2013)
27. H.T. Sencar, N. Memon, *Digital Image Forensics* (Springer, New York, 2013)
28. W. Shi et al., An FPGA-based hardware accelerator for traffic sign detection. IEEE Trans. Very Large Scale Integr. Syst. **25**(4), 1362–1372 (2017)
29. P. Dollar, Piotr's computer vision MATLAB toolbox (PMT), http://vision.ucsd.edu/~pdollar/toolbox/doc/index.html
30. R. Timof et al., in Traffic sign recognition—how far are we from the solution? IJCNN (2013)
31. J. Stallkamp et al., Man vs. computer: benchmarking machine learning algorithms for traffic sign recognition. Neural Netw. **32**, 323–332 (2012)
32. S. Houben et al., in Detection of traffic signs in real-world images: The German traffic sign detection benchmark. IJCNN (2014)
33. S. Sahni, T. Gonzalez, P-complete approximation problems. J. ACM **23**(3), 555–565 (1976)
34. D. Harel, Y. Koren, in Graph Drawing by High Dimensional Embedding. International Symposium on Graph Drawing (Springer, Berlin, 2002). pp. 207–219
35. S. Banerjee, A. Roy, *Linear Algebra and Matrix Analysis for Statistics* (Chapman and Hall/CRC, Hoboken, 2014)
36. I. Borg, *Modern Multidimensional Scaling: Theory and Applications* (Springer, New York, 2005)
37. P. Gopalakrishnan, X. Li, L. Pileggi, in Architecture-aware FPGA placement using metric embedding. DAC (2006), pp. 460–465
38. J. Bentley, Multidimensional binary search trees used for associative searching. Commun. ACM **18**(9), 509–517 (1975)

Chapter 3
Cyberattack-Resilient Hybrid Controller Design with Application to UAS

Cheolhyeon Kwon and Inseok Hwang

3.1 Introduction

Recent years have witnessed a significant growth of autonomous systems used in a wide range of cyber-physical systems (CPSs) domains, but exposed to the threat of cyberattacks due to their close integration of computational resources, physical processes, and communication capabilities [1, 2]. Traditionally, cyberattacks have been largely studied in the computer science field where security is mainly achieved by guaranteeing integrity of data [3–5]. While the computer security parts are key components of the CPS, these methods alone are not sufficient for assuring CPS performance within the unified cyber and physical layers. For example, if the attacker has access to the automatic control layer in the CPS such as programmable logic controllers (PLCs), encryption in the communication network layer cannot protect the CPS [6]. Although these types of attacks could be addressed with cryptographic tools that cover the lower-level control layer, resource constraints inherent in many CPS domains may prevent heavy-duty security approaches from being deployed. There are also multiple sensor attacks that can corrupt the control loop of a CPS without breaking encryption [7, 8]. These highlight the need of a control system approach which extends the existing computer-oriented CPS security architecture to include the resilience of CPS control in coordination of the physical dynamics [9–11].

From a control systems perspective, cyberattacks on CPSs can be broadly categorized into two types [11, 12]: *Denial of Service (DoS) attacks* and *integrity attacks*. Firstly, DoS attacks refer to an attempt to make a network resource unavailable to its intended users, thereby obstructing communication between networked agents

C. Kwon · I. Hwang (✉)
Purdue University, West Lafayette, IN, USA
e-mail: kwonc@purdue.edu; ihwang@purdue.edu

H. Yu et al. (eds.), *Safe, Autonomous and Intelligent Vehicles*,
Unmanned System Technologies, https://doi.org/10.1007/978-3-319-97301-2_3

[13]. These attacks can jam communication channels, attack routing protocols, etc. [14, 15]. On the other hand, integrity attacks compromise the integrity of data packets fed into sensors and/or actuators [16]. In this case, the attacker obtains the secret key, exploits a backdoor communication channel, or physically alters what a sensor is measuring to inject false data. Most research on integrity attacks has focused on the stealthy cyberattack strategies that are able to partially or completely bypass traditional anomaly detectors embedded in a CPS, some examples of which are the replay attack [17], the false data injection attack [18, 19], the zero-dynamics attack [20], and the covert attack [21, 22].

To address the cyberattack countermeasures within the control systems framework, the architectural vulnerabilities of CPS have been investigated to proactively reveal and diagnose the potential threat of attacks [23–26]. These proactive measures can be used for reliable CPS design prior to being deployed, whereby one can identify the critical CPS components and reduce their vulnerabilities by increasing the integrity of major communication channels, installing redundant network agents, etc. As reactive measures for the CPS subject to cyberattacks, there have been developed a number of monitoring systems focusing on the attack-resilient estimators [27–29] and attack detection schemes [30, 31]. Additionally, reachability analysis has been used to assure the CPS state estimation under cyberattacks [32, 33].

Alongside the CPS vulnerability analysis and monitoring system development, the controller design problem is equally important for mitigating the impact of cyberattacks. However, the conventional robust/fault-tolerant control approach is not directly applicable to assure the safety of the CPS under attack. This is mainly due to the unpredictable nature of cyberattacks, which can make the response of a controller designed to react to common disturbances and faults inappropriate in many instances. Therefore, most related research has been limited to designing controllers that are robust to a specific attack type, e.g., DoS attacks [15, 34, 35], without rigorous consideration to the attackers' ability to change their attack strategies over time. One way to address varying cyberattacks is an adaptive control idea that can adjust the controller in response to different attacks [36, 37]. Still, the adaptive control approach alone is not sufficient, especially for safety critical CPSs, since the controller adapted to the present attack may not be safe for future attacks. Another promising way to manage such uncertain cyberattacks is the game theoretical approach where the attacker(s) and the protector(s) are players competing for goals in a dynamic game [38, 39]. In this game, the attacker is assumed to launch the worst attack among all possible attack strategies. Then, although the resulting controller is optimal under the worst attack scenario, it could be excessively conservative when the actual attack deviates from the assumed worst attack. This highlights the difficulty of a controller design that is robust to arbitrary cyberattacks while not significantly degrading the performance.

This research develops an attack-resilient controller that pursues both performance and safety of the CPS by integrating the adaptive control idea and the game-theoretic approach. Inspired by a hybrid system model, we consider a hybrid control framework consisting of a set of sub-controllers, each designed to counter

a specific type of cyberattack. Here, the multiple sub-controllers can adaptively switch among themselves to achieve better control performance with respect to various attacks, including those that have not been assumed for sub-controller design. Further, the robustness of the controller is attained by the switching logic such that the hybrid controller switches to the safest sub-controller. For this task, we introduce a performance measure to evaluate the attack impact and control performance simultaneously. Using this measure, it can be said that a certain controller is safer than another if its evaluated performance is better than the performance of another under a given cyberattack. In other words, the safest sub-controller at a certain moment is the one whose future performance is the best under possible future cyberattacks. Since future attack behavior is unpredictable and arbitrary, we employ the game-theoretic approach and figure out the worst attack scenario of each sub-controller. From the computed worst future performances of the individual sub-controllers, the safest sub-controller is determined to be the one with the "least bad performance," i.e., the best among the worst. The controller then switches to this sub-controller accordingly. Under this switching logic, the proposed hybrid controller is proven to be safer than the adaptive control approach and less conservative than the game-theoretic approach, and also to maintain its stability throughout the switching.

The rest of this paper is organized as follows: In Sect. 3.2, we describe the CPS dynamics subject to cyberattacks and formulate a cyberattack mitigation problem. Section 3.3 presents a hybrid control framework with a switching logic. Specifically, the implemented switching logic enables the hybrid controller to switch to the safest sub-controller based on the current CPS state and past attack history. The performance of the proposed controller is analytically verified in Sect. 3.4, and further extension to infinite time horizon is discussed in Sect. 3.5. Our hybrid controller is demonstrated with an illustrative unmanned aircraft system (UAS) example in Sect. 3.6, for which a set of sub-controllers is exemplified by a pair of optimal H_2 and H_∞ controllers. Conclusions are given in Sect. 3.7.

3.2 Problem Formulation

3.2.1 System and Cyberattack Models

This section presents the mathematical models of the CPS dynamics subject to cyberattacks and an attack model that specifies the attacker's capability. We consider a discrete-time linear CPS model subject to a priori-unknown cyberattacks:

$$x_a(k+1) = Ax_a(k) + Bu(k) + B_a a(k), \quad x_a(0) = x_0 \tag{3.1}$$

where $x_a(k) \in \mathbb{R}^n$ (the subscript "a" denotes the system under attack) and $u(k) \in \mathbb{R}^p$ are the CPS's state and input, respectively, A and B are the system matrices of

appropriate dimensions, and $k \in \mathcal{N}$ denotes the discrete-time index, taking values from the time horizon (possibly infinite) $\mathcal{N} = \{0, 1, 2, \cdots, N\}$. Then, various *attack sequences* $a(k) \in \mathbb{R}^s$ can be injected into the CPS with the *attack matrices* B_a of compatible dimension. It is assumed that the system matrix pairs (A, B) satisfy the controllability condition. Practically, the injected attacks are realized through the physical structure of actuator components. Thus, the attack matrix B_a should satisfy:

$$\text{span}\{B_a\} \subseteq \text{span}\{B\}$$

where "span" denotes the column space of the matrix. For simplicity, let us use the following notation henceforth:

$$x_{[\tau_1,\tau_2]} := \left[x^{\mathrm{T}}(\tau_1)\, x^{\mathrm{T}}(\tau_1+1)\, \cdots\, x^{\mathrm{T}}(\tau_2)\right]^{\mathrm{T}}, \quad 0 \le \tau_1 \le \tau_2 \le N$$

With perfect state measurements, the state history of the CPS, $x_{a[0,k]}$, and the past attack record, $a_{[0,k-1]}$, are available for each time step k, though the future attack signals $a_{[k,N]}$ are unknown. Using this information, the control input can be designed as:

$$u(k) = \mu(x_{a[0,k]}, a_{[0,k-1]})$$

where μ is a mapping function, commonly called the *control law* or *strategy*.

Assumption 1 The attacker is assumed to have full knowledge of the CPS dynamics A and B, and the control strategy μ.

Under this assumption, we consider the worst security problem where the well-informed attacker can exploit the controller's action for his or her attack strategy.

Assumption 2 The attack sequence $a(k)$ is assumed to be *signals with bounded energy* in a finite time window $T \ge 1$, i.e., $\|a_{[0,T]}\|^2 < \rho^2$ where ρ is a measurable constant.

Assumption 2 is concerned with the practical constraints in the CPS. That is, due to the physical limitation of CPS components such as power capacity, operation range, communication bandwidth, etc., an attacker cannot carry out arbitrarily large attack energy. This constraint is commonly applied to the external input (in this case cyberattack) rejection problems [40].

To measure the impact of cyberattacks on the CPS, we introduce a cost function $J(u_{[\tau_1,\tau_2]}, a_{[\tau_1,\tau_2]})$ that simultaneously evaluates the control performance and the attack severity over the time interval $[\tau_1, \tau_2]$. Note that J can be established differently according to the CPS's operational objective, and thus motivating different control strategies. In this research, we consider the CPS regulation objective, i.e., stabilizing the CPS's state subject to cyberattacks, for which the cost function during the time interval $[\tau_1, \tau_2]$ is defined by the following quadratic form:

$$J(u_{[\tau_1,\tau_2]}, a_{[\tau_1,\tau_2]}) := \sum_{k=\tau_1}^{\tau_2} \left(x_a^{\mathrm{T}}(k) Q x_a(k) + u^{\mathrm{T}}(k) R u(k)\right) + x_a^{\mathrm{T}}(\tau_2+1) Q x_a(\tau_2+1) \tag{3.2}$$

where $Q \succeq \mathbf{0}$ and $R \succ \mathbf{0}$ are design parameters including the information about scaling factors, security weights, control efforts, etc.

Without loss of generality, the control strategy μ for stabilizing the state of CPS can be designed as a linear state feedback control:

$$\mu(x_{a[0,k]}, a_{[0,k-1]}) := K(k)x_a(k)$$

where $K(k)$ is a time-varying state feedback gain matrix. Then, the resulting closed-loop system can be represented as:

$$x_a(k+1) = (A + BK(k))\, x_a(k) + B_a a(k) \tag{3.3}$$

Applying the above feedback control into (3.2), the performance measure J can be equivalently represented as the function of feedback gain and attack such that:

$$\begin{aligned} J(K_{[\tau_1,\tau_2]}, a_{[\tau_1,\tau_2]}) := &\sum_{k=\tau_1}^{\tau_2-1} x_a^{\mathrm{T}}(k) \left(Q + K^{\mathrm{T}}(k)RK(k)\right) x_a(k) \\ &+ x_a^{\mathrm{T}}(\tau_2+1)\, Q x_a(\tau_2+1) \end{aligned} \tag{3.4}$$

Then, the attack-resilient control problem turns out to be the design of feedback gain history $K(k)$, $\forall k \in \mathcal{N}$ to optimally mitigate the impact of a priori-unknown cyberattacks.

3.2.2 Cyberattack Mitigation Problem

Based on the closed-loop CPS dynamics (3.3) with the feedback control gain, the attack mitigation problem is formulated as the following optimal control problem:

Problem 1 (General Attack Mitigation Problem) For a given finite time horizon $[0, T]$, we look for the optimal feedback gain history $K_{[0,T]}$ which minimizes the cost function J associated with the cyberattack severity:

$$\begin{aligned} &\min_{K_{[0,T]}} \quad J(K_{[0,T]}, a_{[0,T]}) \\ &\text{Subject to} : \|a_{[0,T]}\|^2 < \rho^2 \\ &\qquad\qquad \text{System dynamics (3.3)} \end{aligned} \tag{3.5}$$

Even though the CPS model and attack model are well-specified, Problem 1 is nontrivial as the attack sequence $a_{[0,T]}$ is not available in advance. The existing research have attempted to address this difficulty by assuming a specific attack scenario so that the attack sequence can be treated as a settled parameter of the

problem. However, the solution $K_{[0,T]}$ is optimal only if the attacker behaves as assumed by the controller. For instance, the original problem can be posed by the game-theoretic approach where the attack sequence $a_{[0,T]}$ is assumed to be the worst attack scenario, i.e., maximizing the cost function J as follows.

Problem 2 (Game-Theoretic Attack Mitigation Problem [40]) For a given finite time horizon $[0, T]$, we look for the optimal feedback gain history $K_{[0,T]}$ which minimizes the cost function J while $a_{[0,T]}$ is maximizing J:

$$\begin{aligned} &\min_{K_{[0,T]}} \max_{a_{[0,T]}} \; J(K_{[0,T]}, a_{[0,T]}) \\ &\text{Subject to : } \|a_{[0,T]}\|^2 < \rho^2 \\ &\qquad\qquad \text{System dynamics (3.3)} \end{aligned} \tag{3.6}$$

The main result to be derived from this linear quadratic game is a linear feedback controller that achieves the optimal (minimax) performance bound. In reality, the attacker's behavior is independent of the controller's actions and may even change over time. Then, the solution $K_{[0,T]}$ to Problem 2, which is optimal under the worst attack scenario, could be excessively conservative to other arbitrary attack sequences. Our work therefore studies an alternative control approach with the ability to adapt the controller in response to varying cyberattacks.

3.3 Hybrid Controller Design

In this section, we seek to design a controller that accounts for the both performance and robustness against an arbitrary attack injection. Inspired by the hybrid system which consists of multiple dynamical models, we propose a hybrid control framework consisting of multiple sub-controllers that are given by:

$$K(k) \in \mathscr{K} = \left\{ K_q \middle| q \in \{1, 2, \cdots, r\} \right\}, \quad k \in \mathscr{N} \tag{3.7}$$

where $K_q, q \in \{1, 2, \cdots, r\}$ individually denotes the sub-controller feedback gain that stabilizes the CPS's state, and can be designed for countering a specific attack strategy. Figure 3.1 illustrates a schematic of the proposed hybrid controller framework. Note that the actual cyberattacks may not match any of the attack strategies considered by these r controllers, and can further change over time.

Mathematically, the hybrid controller can be structurized by a collection of the following five tuples $\mathscr{H} = (\mathscr{Q}, \mathscr{X}_a, \mathscr{A}, f, \gamma)$:

- $\mathscr{Q} = \{1, 2, \cdots, r\}$ is a finite set of the discrete states (also called as modes), each corresponds to the sub-controller.
- $\mathscr{X}_a \in \mathbb{R}^n$ is the continuous state space.
- $\mathscr{A} \in \mathbb{R}^s$ is the attack signal space.

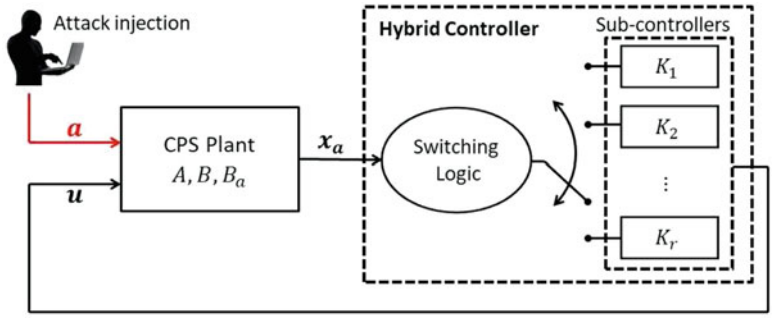


Fig. 3.1 Attack-resilient hybrid controller framework

- $f : \mathscr{Q} \times \mathscr{X}_a \to \mathscr{X}_a$ is a function that describes the evolution of the continuous state. With (3.7), f is defined as:

$$f(q, x) := (A + BK_q)x$$

Consequently, from (3.3), the continuous state dynamics are given by the following difference equation:

$$\begin{aligned} x_a(k+1) &= f(q(k), x_a(k)) + B_a a(k) \\ &= \left(A + BK_{q(k)}\right) x_a(k) + B_a a(k) \end{aligned} \tag{3.8}$$

- $\gamma : \mathscr{Q}^k \times \mathscr{X}_a^{k+1} \times \mathscr{A}^k \times \mathscr{N} \to \mathscr{Q}$ is a function that determines the evolution of the discrete state, called a discrete state transition function:

$$q(k) = \gamma(q_{[0,k-1]}, x_{a[0,k]}, a_{[0,k-1]}, k), \quad q(0) = q_0 \tag{3.9}$$

Let $\zeta = (q, x_a)$, where $q \in \mathscr{Q}$ and $x_a \in \mathscr{X}_a$, be the hybrid state of $\mathscr{H}$ under cyberattacks. An execution of $\mathscr{H}$ then generates a *hybrid state evolution* $\zeta(k)$ where the continuous state x_a is governed by (3.8) and the discrete state q is governed by (3.9), respectively.

Based on the above hybrid controller structure, our main interest is how to design the switching logic, i.e., the discrete state transition function (3.9) that switches the sub-controllers to improve the performance and robustness of the closed-loop CPS control under varying cyberattacks. Algorithmically, adjusting the controller on the basis of the previous observations is the main feature of an *adaptive control approach*. Thus, it is feasible to implement a kind of adaptive control strategy for switching sub-controllers in response to the past attack record. One heuristic example is shown in Fig. 3.2, where the switching logic is to choose the sub-controller designed to counter the past attack strategy. However, such an adaptive control approach could not be sufficiently robust to cyberattacks mainly for two

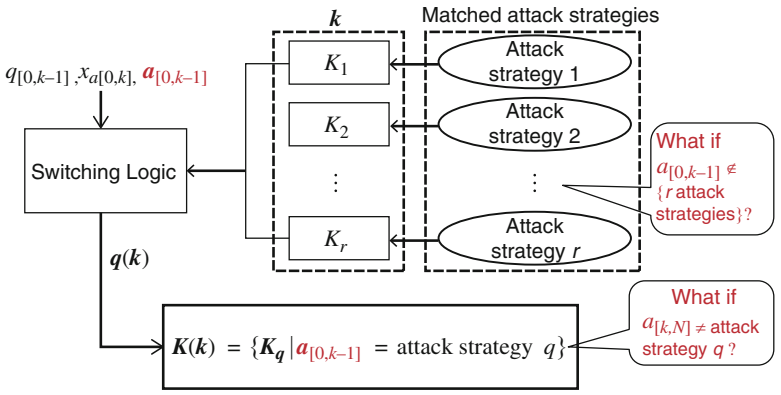


Fig. 3.2 Block diagram of adaptive control-based switching logic for the hybrid controller

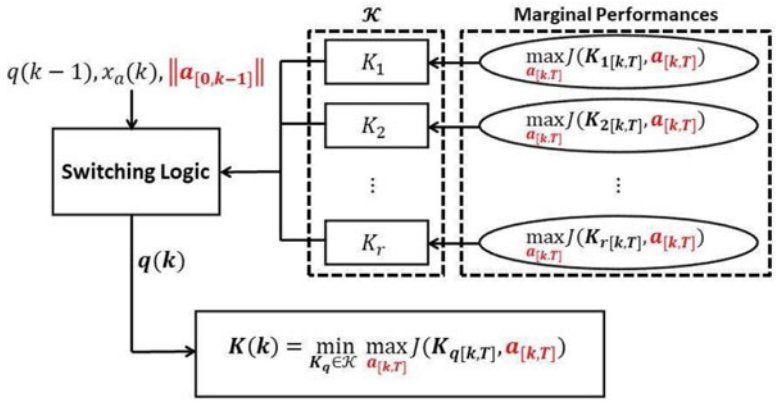


Fig. 3.3 Block diagram of the proposed switching logic for the attack-resilient hybrid controller

reasons: (1) switching logic could be in trouble when the actual attack does not match any of the r attack strategies assumed in individual sub-controller designs, i.e., $a_{[0,k-1]} \notin \{r \text{ attackstrategies}\}$; and (2) since the attacker can change the injected attack strategy over time, the chosen sub-controller for the present attack may not be the best for future attacks, i.e., $a_{[k,T]} \neq \text{attacktype } q$.

To enhance the robustness of the hybrid controller, we employ the game-theoretic approach that takes into account the individual sub-controller's marginal performance as shown in Fig. 3.3. Then, the proposed switching logic can be posed as a (sub)controller-wise minimax problem where the most robust sub-controller is selected as the one with the least bad future performance. For evaluating the performance bound of the each sub-controller in the presence of its own worst

cyberattacks, we consider the attack energy $\|a_{[0,k-1]}\|^2 \in \mathscr{E}$ where $\mathscr{E} = [0, \rho^2]$ is the allowable attack energy space. The amount of the past attack energy can be recursively computed through the past attack history:

$$\|a_{[0,k-1]}\|^2 = \|a_{[0,k-2]}\|^2 + \|a(k-1)\|^2, \quad \forall\, 2 \le k \le T \tag{3.10}$$

We then make use of Assumption 2 that imposes the limit of the energy of the future attack sequences $a_{[k,T]}$ such that:

$$\|a_{[k,T]}\|^2 < \rho^2 - \|a_{[0,k-1]}\|^2 \tag{3.11}$$

Thus, the amount of the available attack energy can be updated at each time step k. Based on (3.11), we further define a function $\bar{J} : \mathscr{Q} \times \mathscr{X}_a \times \mathscr{E} \times \mathscr{N} \to \mathbb{R}$ described by the following optimization problem:

$$\begin{aligned} \bar{J}(q, x, \alpha^2, k) := & \max_{a_{[k,T)]}} J(K_{q[k,T]}, a_{[k,T]}) \\ \text{Subject to} : & \|a_{[k,T]}\|^2 < \rho^2 - \alpha^2 \\ & \text{System dynamics (3.3) where } x_a(k) = x \end{aligned} \tag{3.12}$$

Here, (3.12) expresses the upper bound of the future cost increment in the next $T-k$ time steps from the current state. Note that $\bar{J}$ is only dependent on the current state $x_a(k)$ and the past attack energy consumption $\|a_{[0,k-1]}\|$ instead of the entire history of hybrid states and attack signals. Therefore, the proposed switching logic gives the discrete state transition as:

$$q(k) = \gamma_{\mathscr{H}}(q(k-1), x_a(k), \|a_{[0,k-1]}\|^2, k), \quad q(0) = q_0 \tag{3.13}$$

where $\gamma_{\mathscr{H}} : \mathscr{Q} \times \mathscr{X}_a \times \mathscr{E} \times \mathscr{N} \to \mathscr{Q}$ denotes the discrete state transition function for the proposed hybrid controller.[1] Within the hybrid system framework, the discrete state transition can be addressed by a concept of "guard condition" specified as follows [41].

Definition 1 $G : \mathscr{Q} \times \mathscr{Q} \times \mathscr{N} \to 2^{\mathscr{X}_a \times \mathscr{E}}$ is a time-varying guard condition that assigns to each $(i, j) \in \mathscr{Q} \times \mathscr{Q}$ a guard $G(i, j, k) \subset \mathscr{X}_a \times \mathscr{E}$ such that:

- $G(i, j, k)$ is a measurable subset of $\mathscr{X}_a \times \mathscr{E}$ (possibly empty, unbounded); and
- for each $i \in \mathscr{Q}$, the set $\{G(i, j, k) | j \in \mathscr{Q}\}$ is a disjoint partition of $\mathscr{X}_a \times \mathscr{E}$, that is:

$$G(i, j, k) \cap G(i, l, k) = \emptyset, \quad \forall j, l \in \mathscr{Q},\ j \neq l$$

[1] In this paper, for generality, we call $q(k-1) = i$ to $q(k) = j$ as a discrete state transition even if i is equal to j, i.e., the discrete state q could "transit" to itself.

and

$$\bigcup_{j=1}^{r} G(i, j, k) = \mathscr{X}_a \times \mathscr{E}, \quad \forall j \in \mathscr{Q}$$

Accounting for our switching logic, the set of guards $\{G(i, j, k) \big| i \in \mathscr{Q}\}$ at each time k is designed by (3.12):

$$G(i, j, k) = \left\{(x, \alpha^2) \big| x \in \mathscr{X}_a, \alpha^2 \in \mathscr{E}, \ l \in \mathscr{Q}, \ l \neq j, \ \bar{J}(j, x, \alpha^2, k) \leq \bar{J}(l, x, \alpha^2, k)\right\} \tag{3.14}$$

and $\gamma_{\mathscr{H}}$ is defined as:

$$\gamma_{\mathscr{H}}(i, x, \alpha^2, k) := j \quad \text{if} \quad (x, \alpha^2) \in G(i, j, k)$$

Each guard $G(i, j, k)$ describes a partition of the space $\mathscr{X}_a \times \mathscr{E}$ into a number of closed cells, and governs the transition function to activate the most robust feedback gain $K_{q(k)}$ at each time step k. As depicted in Fig. 3.4, given that the previous discrete state $q(k-1) = i$, if the continuous state and attack energy state point $(x_a(k), \|a_{[0,k-1]}\|^2)$ is inside (including the boundary) of guard $G(i, j, k)$, the discrete state transition from i to j occurs, that yields $q(k) = j$.

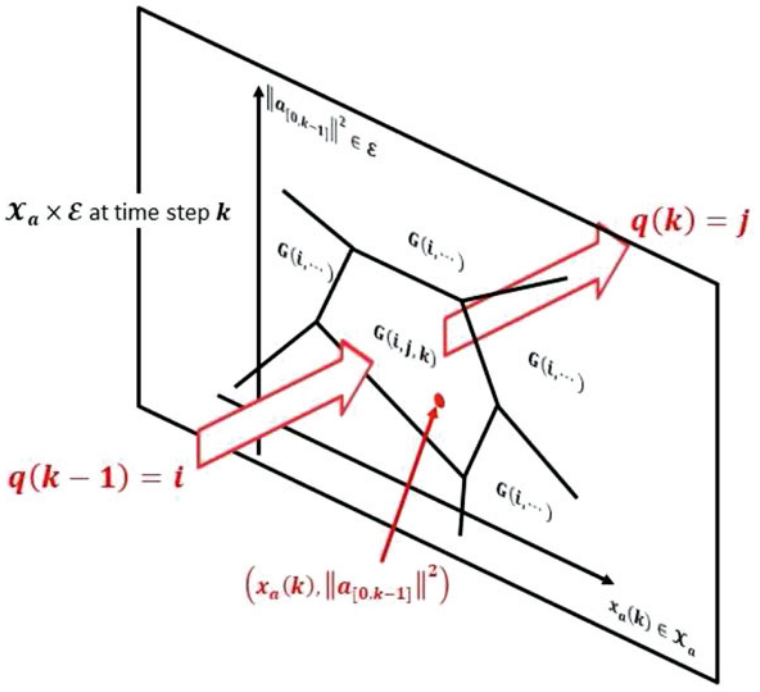


Fig. 3.4 Proposed discrete state transition according to guard conditions

It is easy to make the individual guards interiorly disjoint by selecting appropriate inequalities between $\bar{J}(j, x, \alpha^2, k)$, $j \in \mathscr{Q}$. Then, $\gamma_{\mathscr{H}}$ must be single valued at all times, and thus every execution of $\mathscr{H}$ is well-defined under any possible attack sequence. Also, the guard involves the computationally demanding $\bar{J}$ computations (solving the constrained optimization problem (3.12)) and is implemented offline before running the controller. Therefore, it is computationally efficient and suitable for the online switching process.

3.4 Analytical Performance Verification

This section theoretically verifies the performance and stability of the proposed hybrid controller. First, the performance of the hybrid controller $\mathscr{H}$, whose switching is governed by the guard condition (3.14), is presented by the following theorem.

Theorem 1 *Let $J_{\mathscr{H}}$ be the cost function associated with the hybrid state response $\zeta(k)$ of the proposed hybrid controller $\mathscr{H}$ to an arbitrary cyberattack sequence $a(k)$, which is defined as:*

$$J_{\mathscr{H}}(\zeta_{[\tau_1,\tau_2]}, a_{[\tau_1,\tau_2]}) := J(K^*_{[\tau_1,\tau_2]}, a_{[\tau_1,\tau_2]}) \tag{3.15}$$

where $K^*(k) \equiv K_{q(k)}$ is the feedback gain of the sub-controller selected by the switching logic (3.13). Then, for the given finite time horizon $[0, T]$, the following inequality holds:

$$\max_{a_{[0,T]}} J_{\mathscr{H}}(\zeta_{[0,T]}, a_{[0,T]}) \leq \min_{\{q\in\mathscr{Q}\}} \max_{a_{[0,T]}} J(K_{q[0,T]}, a_{[0,T]}) \tag{3.16}$$

Proof The theorem is proved by induction. Using (3.12), the right-hand side of (3.16) can be rewritten as:

$$\min_{\{q\in\mathscr{Q}\}} \max_{a_{[0,T]}} J(K_{q[0,T]}, a_{[0,T]}) = \arg\min \left\{ \bar{J}(q, x_a(0), 0, 0) \middle| q \in \mathscr{Q} \right\} \tag{3.17}$$

Under an arbitrary admissible attack sequence, the initial discrete state $q(0)$ is chosen according to the guard condition $(x(0), 0) \in G(i, q(0), 0)$ in (3.14) such that:

$$\begin{aligned} &J_{\mathscr{H}}(\zeta_{[0,0]}, a_{[0,0]}) + \bar{J}(q(0), x_a(1), \|a_{[0,0]}\|^2, 1) \\ &\quad \leq \bar{J}(q(0), x_a(0), 0, 0) = \arg\min \left\{ \bar{J}(q, x_a(0), 0, 0) \middle| q \in \mathscr{Q} \right\} \end{aligned} \tag{3.18}$$

And, the every discrete state transition at each time step $k = 1, 2, \cdots, T-1$ satisfies the following inequality:

$$\begin{aligned} &J_{\mathscr{H}}(\zeta_{[0,k]}, a_{[0,k]}) + \bar{J}(q(k), x_a(k+1), \|a_{[0,k]}\|^2, k+1) \\ &\quad \leq J_{\mathscr{H}}(\zeta_{[0,k-1]}, a_{[0,k-1]}) + \bar{J}(q(k-1), x_a(k), \|a_{[0,k-1]}\|^2, k) \end{aligned} \tag{3.19}$$

For the final discrete state $q(T)$, the following inequality can be induced by the guard condition:

$$J_{\mathscr{H}}(\zeta_{[0,T]}, a_{[0,T]}) \leq J_{\mathscr{H}}(\zeta_{[0,T-1]}, a_{[0,T-1]}) + \bar{J}(q(T-1), x_a(T), \|a_{[0,T-1]}\|^2, T) \tag{3.20}$$

Arranging inequalities (3.18), (3.19), and (3.20) in the order of their magnitudes, one can obtain the inequality (3.16). This represents the marginal performance of the proposed hybrid controller. □

In the right-hand-side of inequality (3.16), the cost $J(K_{j[0,T]}, a_{[0,T]})$ indicates the performance when adopting the jth sub-controller only. Thus, Theorem 1 guarantees that the performance of the proposed hybrid controller is equivalent to or better than any of individual sub-controllers, implying that the hybrid controller performs more safely than the individual attack-resilient controller which considers only a single control strategy.

Remark 1 Suppose one of the sub-controllers in the hybrid controller is designed by the game-theoretic approach, i.e., the solution to Problem 2 based on the assumption of the worst attack scenario. Then, by Theorem 1, our hybrid controller grants the less conservative control performance than the game-theoretic approach against arbitrary cyberattacks.

We further show that the switching logic governed by the guard condition (3.14) guarantees the stability of the hybrid controller.

Theorem 2 *Suppose that each sub-controller of the given hybrid controller can individually stabilize the CPS's state or, equivalently, satisfy the following asymptotic stability in the sense of Lyapunov:*

$$\left(A + BK_q\right)^{\mathrm{T}} P_q \left(A + BK_q\right) - P_q \prec \mathbf{0}, \quad \forall q \in \mathscr{Q} \tag{3.21}$$

where $P_q \succ \mathbf{0}$, $q \in \mathscr{Q}$ represents an arbitrary positive definite matrix. Then, the proposed hybrid controller that consists of these sub-controllers is also asymptotically stable in the sense of Lyapunov.

Proof Consider a collection of quadratic Lyapunov-like functions defined as:

$$V_q(x) := x^{\mathrm{T}} \left(Q + K_q^{\mathrm{T}} R K_q\right) x, \quad \forall q \in \mathscr{Q} \tag{3.22}$$

They correspond to individual sub-controllers q and are concatenated together to make a time-varying Lyapunov function for the hybrid controller such that:

$$V_{\mathscr{H}}(x_a(k), k) := x_a^{\mathrm{T}}(k)(Q + C^{\mathrm{T}} K_{q(k)}^{\mathrm{T}} R K_{q(k)} C) x_a(k) \tag{3.23}$$

Note that $V_{\mathscr{H}}$ is positive definite, but may not be monotonically decreasing over the successive switchings. Using (3.22), the cost function can be written as the sum of the Lyapunov-like functions:

$$J(K_{q[0,T]}, a_{[0,T]}) = \sum_{k=0}^{T-1} V_q(x_a(k)) + x_a^{\mathrm{T}}(T) Q x_a(T) \tag{3.24}$$

Similarly, from (3.15), we know that:

$$J_{\mathscr{H}}(\zeta_{[0,T]}, a_{[0,T]}) = \sum_{k=0}^{T-1} V_{\mathscr{H}}(x_a(k), k) + x_a^{\mathrm{T}}(T) Q x_a(N) \tag{3.25}$$

Now, let $P_q = Q + K_q^{\mathrm{T}} R K_q \succ \mathbf{0}$. Subject to the energy-bounded attack sequence along the infinite time horizon $\mathscr{N} = [0, \infty)$, the asymptotic stability condition (3.21) yields:

$$\lim_{k \to \infty} V_q(x_a(k)) = 0, \quad \forall q \in \mathscr{Q} \tag{3.26}$$

On the other hand, substituting (3.24) and (3.25) for (3.16) gives:

$$\sum_{k=0}^{\infty} V_{\mathscr{H}}(x_a(k), k) \leq \min_{\{q \in \mathscr{Q}\}} \max_{a_{[0,\infty]}} \sum_{k=0}^{\infty} V_q(x_a(k)) \tag{3.27}$$

From (3.26), $\sum_{k=0}^{\infty} V_q(x_a(k))$ is bounded regardless of the attack sequence. Hence, $\sum_{k=0}^{\infty} V_{\mathscr{H}}(x_a(k), k)$ is also bounded, which leads to:

$$\lim_{k \to \infty} V_{\mathscr{H}}(x_a(k), k) = 0 \tag{3.28}$$

By the Lyapunov theorem, the existence of a positive definite Lyapunov function $V_{\mathscr{H}}$ satisfying (3.28) guarantees the asymptotic stability of the hybrid controller.

□

3.5 Extension to Infinite Time Horizon: Receding Horizon Controller

Sections 3.3 and 3.4 discuss an attack-resilient hybrid controller and its performance within a finite time horizon $[0, T]$. Note that T is the time window required for exhibiting the attack energy ρ. This prompts the question to an implementation of the proposed switching logic for the infinite time horizon $[0, N]$, as $N \to \infty$. The following question is whether the resulting hybrid controller performs

adequately for an infinite time horizon attack mitigation problem. The answers to these questions are subject to appropriate modification of the switching logic as will be established in this section.

First, the attack energy constraint for an infinite time horizon needs to be replaced by an *attack power constraints* as follows:

$$\lim_{N\to\infty}\frac{1}{N}\sum_{k=0}^{N}\|a(k)\|^2=\lim_{N\to\infty}\frac{1}{N}\|a_{[0,N]}\|^2<\rho_{\text{inf}}^2 \tag{3.29}$$

where ρ_{inf} denotes a measurable attack power bound. To address the switching logic over the infinite time horizon, we adopt the idea of receding horizon control (RHC), also known as model predictive control (MPC) [42, 43]. The basic concept of RHC is to design a control strategy based on a finite time horizon from the current time k, say $[k, k+T]$, and repeat the process at the next time, say $[k+1, k+1+T]$. Thus, the term "*receding horizon*" indicates that a finite time horizon recedes as time goes by, and the controller only considers that finite window for its strategy without dealing with the infinite time horizon. Specifically, we define the receding time horizon for our switching logic as follows.

Definition 1 $T_p > 0$ and $T_f > 0$ are finite time indices, respectively, used to truncate the past and future time window from the current time step, such that the receding time horizon to be considered at time k is $[k-T_p, k+T_f-1]$.

Applying the receding horizon to the power-bounded attack sequence (3.29), the amount of available attack energy within T_h+T_f time steps can be recursively updated by:

$$\|a_{[k,k+T_f-1]}\|^2<(T_p+T_f)\rho_{\text{inf}}^2-\|a_{[k-T_p,k-1]}\|^2 \tag{3.30}$$

Therefore, $\|a_{[k,k+T_f-1]}\|^2\in\mathscr{E}_{\text{RHC}}$ where $\mathscr{E}_{\text{RHC}}=[0,(T_h+T_f)\rho_{\text{inf}}^2]$ is the allowable attack energy space within the finite receding time horizon $[k-T_p, k+T_f-1]$.

Now, let $\mathscr{H}_{\text{RHC}}$ be the hybrid controller used for the infinite time horizon which includes the same $\mathscr{Q}$, $\mathscr{X}_a$, $\mathscr{A}$, and f of $\mathscr{H}$. Using (3.30), the upper bound of the future cost increment in the finite receding horizon is defined by a function $\bar{\mathbf{J}}:\mathscr{Q}\times\mathscr{X}_a\times\mathscr{E}_{\text{RHC}}\to\mathbb{R}$ given as:

$$\begin{aligned}\bar{\mathbf{J}}(q,x,\alpha^2):=&\max_{a_{[k,k+T_f-1]}} J(K_{q[k,k+T_f-1]},a_{[k,k+T_f-1]})\\ &\text{Subject to}:\|a_{[k,k+T_f-1]}\|^2<(T_h+T_f)\rho_{\text{inf}}^2-\alpha^2\\ &\qquad\text{System dynamics (3.3) where } x_a(k)=x\end{aligned} \tag{3.31}$$

Compared to $\bar{J}$ in (3.12), $\bar{\mathbf{J}}$ computes the future cost increment within constant T_f time steps while receding the time window with respect to the current time step k. As a result, the guard condition $\mathbf{G}$ designed for $\mathscr{H}_{\text{RHC}}$ is time-invariant such that the

set of guards $\{\mathbf{G}(i,\, j)\big|i \in \mathscr{Q}\}$ is given by:

$$\mathbf{G}(i,\, j) = \left\{(x, \alpha^2)\big|x \in \mathscr{X}_a, \alpha^2 \in \mathscr{E}_{\text{RHC}},\ l \in \mathscr{Q},\ j \neq l, \bar{\mathbf{J}}(j, x, \alpha^2) \leq \bar{\mathbf{J}}(l, x, \alpha^2)\right\} \tag{3.32}$$

Similar to G, each guard $\mathbf{G}(i,\, j)$ partitions the space $\mathscr{X}_a \times \mathscr{E}_{\text{RHC}}$ into closed cells that are the bases of the discrete state transition for $\mathscr{H}_{\text{RHC}}$, which is given by:

$$q(k) = \gamma_{\mathscr{H}_{\text{RHC}}}(q(k-1), x_a(k), \|a_{[k-T_p,k-1]}\|^2) \tag{3.33}$$

where the discrete state transition function $\gamma_{\mathscr{H}_{\text{RHC}}} : \mathscr{Q} \times \mathscr{X}_a \times \mathscr{E}_{\text{RHC}} \to \mathscr{Q}$ is defined was:

$$\gamma_{\mathscr{H}_{\text{RHC}}}\left(i, x, \alpha^2\right) := j \qquad \text{if } (x, \alpha^2) \in \mathbf{G}(i,\, j)$$

Clearly, the RHC-based approach significantly reduces the computation effort to come up with the control strategy while not dealing with the infinite time horizon. The performance of $\mathscr{H}_{\text{RHC}}$, however, is considered in the piecewise manner for the finite time window with respect to the current time step k. Thus, concatenating the switching solutions over the infinite time horizon may not lead to the globally optimal performance. This issue is analogous to RHC or MPC that rely on real-time optimization to determine the control strategy based on the current finite time horizon. The stability of $\mathscr{H}_{\text{RHC}}$ can be shown similarly to Theorem 2 but needs a more restrictive condition as follows:

Theorem 3 *Over an infinite time horizon, $\mathscr{H}_{\text{RHC}}$ is asymptotically stable in the sense of Lyapunov regardless of its switching sequence if the contained sub-controllers satisfy:*

$$\left(A + BK_q\right)^{\text{T}} P \left(A + BK_q\right) - P \prec \mathbf{0},\ \ \forall q \in \mathscr{Q} \tag{3.34}$$

where $P \succ \mathbf{0}$ is a positive definite matrix.

Proof For brevity, the detailed proof is not shown. Note that it can be easily proved by considering the Lyapunov function $V_{\mathscr{H}_{\text{RHC}}}(x_a(k)) := x_a^{\text{T}}(k) P x_a(k)$. □

Note that the length of receding time horizon is determined by T_p and T_f which are the design parameters for implementing $\mathscr{H}_{\text{RHC}}$. Therefore, appropriate selection of T_p and T_f is crucial for managing the trade-off between robustness and performance of $\mathscr{H}_{\text{RHC}}$. For instance, longer T_f would increase the robustness of the controller by further examining the future attack capacity, but could decrease the performance as it is likely to be more conservative under an arbitrary cyberattack sequence.

Remark 2 Regarding the predictive time step T_f, two extreme cases can be thought: i) $T_f = 1$ only takes care of the present attack for the controller's action. In this case, $\mathscr{H}_{\text{RHC}}$ performs an one-step brute-force optimal control at every time step without

considering future cyberattacks; and ii) $T_f \to \infty$ covers the entire future attack strategy over the infinite time horizon. Then, regardless of the past attack history, $\mathscr{H}_{\text{RHC}}$ results in the identical sub-controller all the time, which is the most robust within the next "infinite" time horizon. This is similar to the infinite time version of Problem 2 provided in [40].

In summary, a cycle of the recursive hybrid state evolution in $\mathscr{H}_{\text{RHC}}$ is outlined as follows (see also Fig. 3.5):

- Initial setup: Compute the upper bound function $\bar{\mathbf{J}}$ for all $q \in \mathscr{Q}$, $x \in \mathscr{X}_a$, $\alpha^2 \in \mathscr{E}_{\text{RHC}}$ using (3.31) and design the guard condition $\mathbf{G}$ using (3.32) that formulates the discrete state transition function $\gamma_{\mathscr{H}_{\text{RHC}}}$.
- At every k time step,
  1. *Attack evaluation:* Compute the previous attack $a(k-1)$ using the previous discrete state $q(k-1)$ and continuous state $x_a(k-1)$, and the current continuous state $x_a(k)$.
  2. *Attack energy update:* Update the amount of the past attack energy from $\|a_{[k-T_p-1,k-2]}\|$ to $\|a_{[k-T_p,k-1]}\|$ using $a(k-1)$.
  3. *Discrete state transition:* Determine the discrete state $q(k)$ using (3.33).
  4. *Continuous state evolution:* The continuous state $x_a(k+1)$ is given by (3.8) with an arbitrary attack vector $a(k)$.

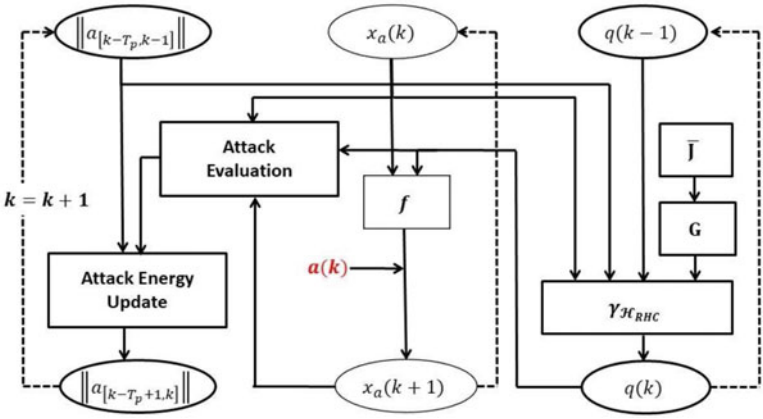


Fig. 3.5 Recursive algorithm structure of hybrid controller

3.6 Illustrative Example

In this section, we present an illustrative example of the proposed hybrid controller that contains two sub-controllers: H_2 and H_∞ optimal controllers, each of which is designed based on the assumption of the specific attack strategy. Our hybrid controller allows switchings between the two sub-controllers to mitigate the effects of varying cyberattacks.

3.6.1 H_2 Optimal Controller

H_2 optimal control focuses on the average effect of cyberattacks on the CPS performance instead of a specific attack sequence. Therefore, it is suitable for countering a *disturbance or noise-like attack sequence* [44]. To design the H_2 optimal controller, we define z as a performance output of a feedback control K subject to attack injection. When the controller is asymptotically stable, each element of z is square integrable and the L_2 norm $\|z\|_2$ of z is finite and is given by:

$$\|z\|_2 := \left(\sum_{k=0}^{\infty} \mathrm{tr} \left[B_a^{\mathrm{T}} (A + BK)^{k\mathrm{T}} (Q + K^{\mathrm{T}} R K)(A + BK)^{\mathrm{T}} B_a \right] \right)^{\frac{1}{2}}$$

We refer to $\|z\|_2$ as the H_2 norm of the system. This norm can be regarded as a measure of the ability to mitigate the average effect of cyberattack a on the CPS performance. Therefore, the optimal feedback gain K_{H_2} which minimizes the above H_2 norm of z is given by:

$$K_{H_2} = -(B^{\mathrm{T}} P_{H_2} B + R)^{-1} B^{\mathrm{T}} P_{H_2} A \tag{3.35}$$

where P_{H_2} is the solution to the following discrete-time algebraic Riccati equation:

$$P_{H_2} = A^{\mathrm{T}} P_{H_2} A + Q - A^{\mathrm{T}} P_{H_2} B_a \left(R + B_a^{\mathrm{T}} P_{H_2} B_a \right)^{-1} B_a^{\mathrm{T}} P_{H_2} A$$

Note that this controller could lead to severe performance loss when the actual cyberattack follows the particular attack strategy, e.g., aims at maximizing the cost function J.

3.6.2 H_∞ Optimal Controller

The H_∞ optimal control is the solution to the game-theoretic approach, addressing the controller in the presence of *the worst attack strategy* [40]. To do so, we want to minimize the H_∞ norm of system that is defined as:

$$\|z\|_\infty := \sup_{a\neq 0, \|a\|_2<\infty} \frac{\|za\|_2}{\|a\|_2}$$

This can be interpreted as the following linear–quadratic dynamic game:

$$\min_K \max_{a_{[0,\infty]}} \left(J(K_{[0,\infty]}, a_{[0,\infty]}) - \eta^2 \left\| a_{[0,\infty]} \right\|^2 \right) \tag{3.36}$$

where the controller (attacker) attempts to minimize (maximize) the cost. The nonnegative quantity η denotes the desired attack attenuation level. For the existence of a unique feedback saddle-point solution, it is assumed that:

$$\eta^2 I - B_a^{\mathrm{T}} P_{H_\infty} B_a \succ \mathbf{0} \tag{3.37}$$

where I denotes an identity matrix with appropriate dimension and P_{H_∞} satisfies the following generalized algebraic Riccati equation:

$$P_{H_\infty} = Q + A^{\mathrm{T}} P_{H_\infty} \left(I + (BB^{\mathrm{T}} - \eta^{-2} B_a B_a^{\mathrm{T}}) P_{H_\infty} \right)^{-1} A \tag{3.38}$$

As the solution to (3.36), the H_∞ optimal feedback gain K_{H_∞} is given by:

$$K_{H_\infty} = -B^{\mathrm{T}} P_{H_\infty} \left(I + (BB^{\mathrm{T}} - \eta^{-2} B_a B_a^{\mathrm{T}}) P_{H_\infty} \right)^{-1} A \tag{3.39}$$

Notice also that the performance of H_∞ optimal control could be degraded (excessively conservative) if the actual cyberattack were not to be the assumed worst attack strategy.

3.6.3 UAS Model

The effectiveness of the hybrid controller with the H_2 and H_∞ sub-controllers is demonstrated with an example of an unmanned aircraft system (UAS) under cyberattacks. In this example, we consider the linearized longitudinal motion of a rotorcraft, for which a detailed dynamical model can be found in [45]. Let the state $x_a = \begin{bmatrix} V_{\mathrm{h}} & V_{\mathrm{v}} & \dot{\theta} & \theta \end{bmatrix}^{\mathrm{T}}$ consist of the horizontal velocity V_{h}(knots), vertical velocity V_{v}(m/s), pitch rate $\dot{\theta}$(deg/s), and pitch angle θ(deg). The control inputs $u = \begin{bmatrix} \delta_c & \delta_l \end{bmatrix}^{\mathrm{T}}$ are collective pitch control δ_c(deg) and longitudinal cyclic pitch control δ_l(deg), both of which are associated with the angle of the rotorcraft's rotor(s). The dynamics of the rotorcraft is linearized at the cruise flight with the horizontal velocity $V_{\mathrm{h}} = 170$ knots and discretized with sampling time $T_s = 0.1$ s, which yields in the following the system matrices:

$$A = \begin{bmatrix} 0.9641 & 0.0025 & -0.0004 & -0.0452 \\ 0.0044 & 0.9036 & -0.0188 & -0.3841 \\ 0.0096 & 0.0465 & 0.9435 & 0.2350 \\ 0.0005 & 0.0024 & 0.0969 & 1.0120 \end{bmatrix}, \quad B = \begin{bmatrix} 0.0440 & 0.0164 \\ 0.4898 & -0.7249 \\ -0.5227 & 0.4172 \\ -0.0266 & 0.0214 \end{bmatrix}$$

For the performance evaluation, the quadratic cost function J is parameterized by:

$$Q = \begin{bmatrix} 1 & 5 & 10 & 10 \\ 5 & 25 & 50 & 50 \\ 10 & 50 & 100 & 100 \\ 10 & 50 & 100 & 100 \end{bmatrix}, \quad R = \begin{bmatrix} 100 & 0 \\ 0 & 100 \end{bmatrix}$$

Without loss of generality, the attack matrix is assumed to be $B_a = B$ since the attack signal $a(k)$ and control input sequence $u(k)$ share the same actuator. Then, the RHC-based hybrid controller $\mathscr{H}_{\text{RHC}}$ is implemented where the sub-controllers K_{H_2} and K_{H_∞} are, respectively, designed by (3.35) and (3.39), and the receding time horizon are set to be $T_p = 10$ and $T_f = 10$.

3.6.4 Simulation Results

In the simulation, we specifically consider the attack scenario in which the attacker has access to the collective pitch control δ_c and injects the sinusoidal signal with the power $\rho_{\text{inf}} = 1(\text{deg})$ as shown in Fig. 3.6. Note that this attack scenario is not anticipated by the both H_2 and the H_∞ optimal controllers which are, respectively, targeting the average effect of the attack and the worst attack strategy. Nevertheless, the proposed hybrid $H_2 - H_\infty$ controller is able to adapt to those attacks that have not been addressed in the design of sub-controllers through the switching between them.

Figure 3.7 shows each state of the UAS regulated by the hybrid $H_2 - H_\infty$ controller in response to the sinusoidal attack signal over the 10 s simulation time (100 time steps). Subject to the power-bounded attack injection, the state evolution of the UAS converges to the bounded periodic variation, which agrees with the theoretical guarantee for the stability of our hybrid controller. For a comparative analysis, we separately test the individual H_2 and H_∞ optimal controllers under the same attack scenario and evaluate their performances (See Figs. 3.8 and 3.9). Figure 3.10 compares the increment of cost J for the hybrid $H_2 - H_\infty$ controller with those of the individual H_2 and the H_∞ optimal controllers over the simulation period. While each controller performs poorly as it is subject to the cyberattack different from the one assumed for its design, the hybrid controller can achieve the better performance by properly switching its sub-controllers from one to another as presented in Fig. 3.11. This clearly validates that the performance of the proposed hybrid controller is equivalent or better than that of its individual sub-controllers (in this case the H_2 and H_∞ optimal controllers).

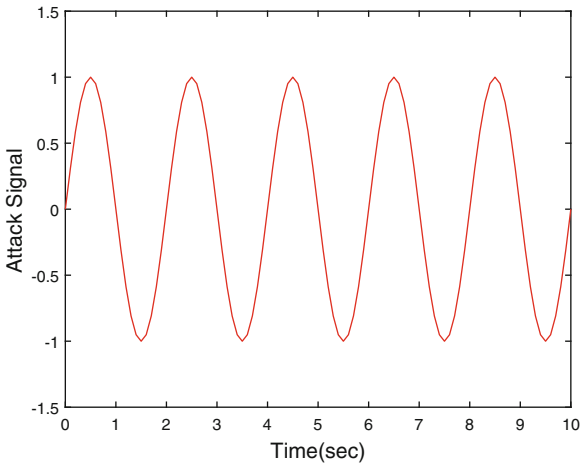


Fig. 3.6 Sinusoidal attack signal injected into the collective pitch control δ_c

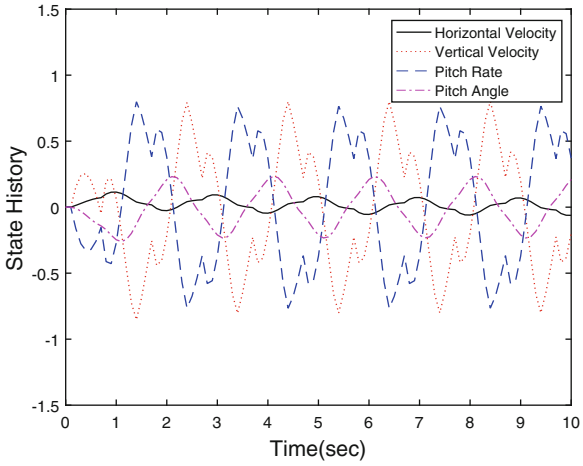


Fig. 3.7 UAS states evolution regulated by the hybrid $H_2 - H_\infty$ controller subject to sinusoidal cyberattack

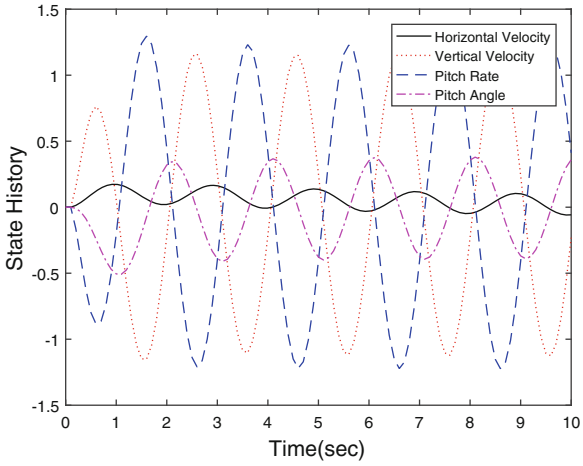


Fig. 3.8 UAS states evolution regulated by the H_2 optimal controller subject to sinusoidal cyberattack

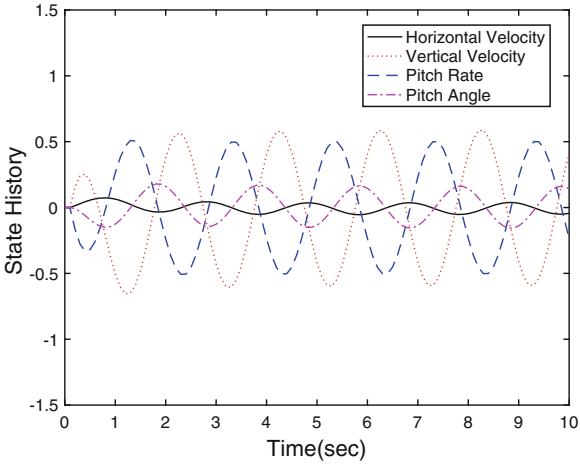


Fig. 3.9 UAS states evolution regulated by the H_∞ optimal controller subject to sinusoidal cyberattack

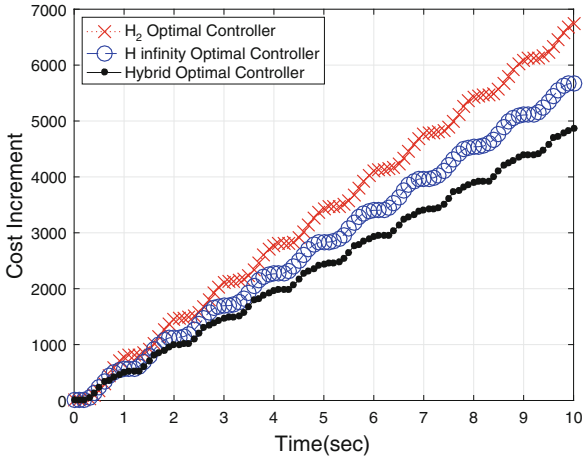


Fig. 3.10 Performance comparison: cost increments of the H_2, H_∞, and the proposed hybrid $H_2 - H_\infty$ controllers

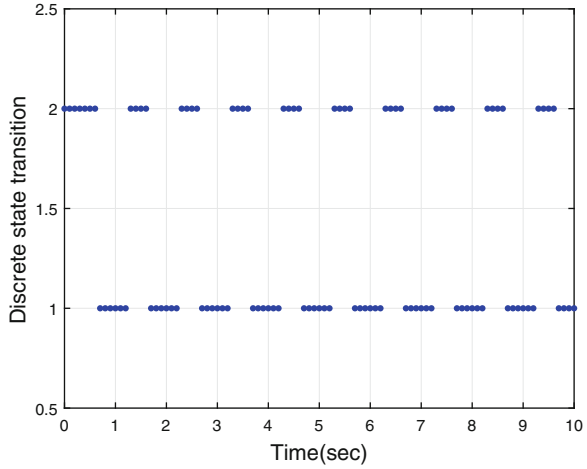


Fig. 3.11 Discrete state transition of the hybrid controller between discrete state 1 (H_2 optimal controller) and discrete state 2 (H_∞ optimal controller)

3.7 Conclusions

This research has investigated the safety and performance issues of cyber-physical systems (CPS) subject to cyberattacks from a control systems perspective. To address the attack-resilient controller while enhancing both robustness and performance, a hybrid control framework has been proposed which is capable of adapting its control strategy in response to a priori-unknown and possibly time-varying cyberattacks. The proposed hybrid controller, consisting of a set of sub-controllers each designed for counteracting a specific attack strategy, can switch among the sub-controllers to achieve the better performance than any single controller. The switching logic then accounts for the robustness of the controller by computing the worst future performances of the individual sub-controllers and determining the safest sub-controller as the one with the least worst-case performance. The developed hybrid controller has been analytically verified for its performance and stability, and further extended to the infinite time horizon case based on the receding horizon control approach. As a case study, an illustrative class of the hybrid controller has been implemented in which the H_2 and H_∞ optimal controllers are designed for the sub-controllers. Simulation results with an unmanned aircraft system (UAS) example have demonstrated that the hybrid $H_2 - H_\infty$ controller performs better than either the individual H_2 or H_∞ optimal controller against cyberattacks.

References

1. A. Humayed, J. Lin, F. Li, B. Luo, Cyber-physical systems security–a survey. IEEE Internet of Things J. **4**, 1802–1831 (2017)
2. V.L. Do, L. Fillatre, I. Nikiforov, P. Willett et al., Security of SCADA systems against cyber–physical attacks. IEEE Aerosp. Electron. Syst. Mag. **32**(5), 28–45 (2017)
3. A. Avizenis, J. Laprie, B. Randell, C. Landwehr, Basic concepts and taxonomy of dependable and secure computing. IEEE Trans. Dependable Secure Comput. **1**(1), 11–32 (2004)
4. J. Saltzer, M. Schroeder, The protection of information in computer systems. Proc. IEEE **63**, 1278–1308 (1975)
5. T. Ahmed, A.R. Tripathi, Security policies in distributed CSCW and workflow systems. IEEE Trans. Syst. Man Cybern. Syst. Hum. **40**, 1220–1231 (2010)
6. A. Teixeira, D. Pérez, H. Sandberg, K.H. Johansson, Attack models and scenarios for networked control systems, in *Proceedings of the 1st International Conference on High Confidence Networked Systems* (ACM, New York, 2012), pp. 55–64
7. D.P. Shepard, J.A. Bhatti, T.E. Humphreys, A.A. Fansler, Evaluation of smart grid and civilian UAV vulnerability to GPS spoofing attacks, in *Proceedings of the ION GNSS Meeting*, vol. 3 (2012), pp. 3591–3605
8. N.O. Tippenhauer, C. Pöpper, K.B. Rasmussen, S. Capkun, On the requirements for successful GPS spoofing attacks, in *Proceedings of the 18th ACM Conference on Computer and Communications Security* (ACM, New York, 2011), pp. 75–86
9. H. Qi, X. Wang, L.M. Tolbert, F. Li, F.Z. Peng, P. Ning, M. Amin, A resilient real-time system design for a secure and reconfigurable power grid. IEEE Trans. Smart Grid **2**(4), 770–781 (2011)

10. M. Pajic, J. Weimer, N. Bezzo, O. Sokolsky, G.J. Pappas, I. Lee, Design and implementation of attack-resilient cyberphysical systems: with a focus on attack-resilient state estimators. IEEE Control. Syst. **37**(2), 66–81 (2017)
11. F. Pasqualetti, F. Dorfler, F. Bullo, Attack detection and identification in cyber-physical systems. IEEE Trans. Autom. Control **58**, 2715–2729 (2013)
12. C. Kwon, W. Liu, I. Hwang, Security analysis for cyber-physical systems against stealthy deception attacks, in *American Control Conference (ACC), 2013* (IEEE, Piscataway, 2013), pp. 3344–3349
13. V. Gligor, A note on denial-of-service in operating systems. IEEE Trans. Softw. Eng. **SE-10**(3), 320–324 (1984)
14. A. Gupta, C. Langbort, T. Basar, Optimal control in the presence of an intelligent jammer with limited actions, in *49th IEEE Conference on Decision and Control* (2010), pp. 1096–1101
15. S. Amin, G. Schwartz, S. Sastry, Security of interdependent and identical networked control systems. Automatica **49**, 186–192 (2013)
16. C. Kwon, W. Liu, I. Hwang, Analysis and design of stealthy cyber attacks on unmanned aerial systems. AIAA J. Aerosp. Inf. Syst. **11**, 525–539 (2014)
17. Y. Mo, R. Chabukswar, B. Sinopoli, Detecting integrity attacks on SCADA systems. IEEE Trans. Control Syst. Technol. **22**(4), 1396–1407 (2014)
18. Y. Liu, P. Ning, M. Reiter, False data injection attacks against state estimation in electric power grids, in *16th ACM Conference on Computer and Communications Security* (2009), pp. 21–32
19. Y. Mo, B. Sinopoli, False data injection attacks in cyber physical systems, in *1st Workshop on Secure Control Systems*, no. 7 (2010)
20. A. Teixeira, I. Shames, H. Sandberg, K.H. Johansson, Revealing stealthy attacks in control systems, in *50th Annual Allerton Conference on Communication, Control, and Computing (Allerton), 2012* (IEEE, Piscataway, 2012), pp. 1806–1813
21. R.S. Smith, A decoupled feedback structure for covertly appropriating networked control systems. IFAC Proc. Vol. **44**(1), 90–95 (2011)
22. A. de Sa, L. Carmo, R. Machado, Covert attacks in cyber-physical control systems. IEEE Trans. Ind. Inf. **13**, 1641–1651 (2017)
23. A. Teixeira, S. Amin, H. Sandberg, K. Johansson, S. Sastry, Cyber security analysis of state estimators in electric power systems, in *49th IEEE Conference on Decision and Control* (2010), pp. 5991–5998
24. H. Sandberg, A. Teixeira, K. Johansson, On security indices for state estimators in power networks, in *1st Workshop on Secure Control Systems*, no. 8 (2010)
25. S. Amin, X. Litrico, S. Sastry, A.M. Bayen, Cyber security of water SCADA systems. Part I: analysis and experimentation of stealthy deception attacks. IEEE Trans. Control Syst. Technol. **21**(5), 1963–1970 (2013)
26. L. Hu, Z. Wang, Q.-L. Han, X. Liu, State estimation under false data injection attacks: security analysis and system protection. Automatica **87**, 176–183 (2018)
27. M. Pajic, I. Lee, G.J. Pappas, Attack-resilient state estimation for noisy dynamical systems. IEEE Trans. Control Netw. Syst. **4**(1), 82–92 (2017)
28. H. Fawzi, P. Tabuada, S. Diggavi, Secure state-estimation for dynamical systems under active adversaries, in *49th Annual Allerton Conference on Communication, Control, and Computing (Allerton), 2011* (IEEE, Piscataway, 2011), pp. 337–344
29. Y. Shoukry, P. Nuzzo, A. Puggelli, A.L. Sangiovanni-Vincentelli, S.A. Seshia, P. Tabuada, Secure state estimation for cyber physical systems under sensor attacks: a satisfiability modulo theory approach. IEEE Trans. Autom. Control **62**, 4917–4932 (2017)
30. Y. Chen, S. Kar, J.M. Moura, Dynamic attack detection in cyber-physical systems with side initial state information. IEEE Trans. Autom. Control **62**(9), 4618–4624 (2017)
31. F. Miao, Q. Zhu, M. Pajic, G.J. Pappas, Coding schemes for securing cyber-physical systems against stealthy data injection attacks. IEEE Trans. Control Netw. Syst. **4**(1), 106–117 (2017)
32. C. Kwon, S. Yantek, I. Hwang, Safety assessment of unmanned aerial systems subject to stealthy cyber attacks. AIAA J. Aerosp. Inf. Syst. **13**, 27–45 (2016)

33. C. Kwon, I. Hwang, Reachability analysis for safety assurance of cyber-physical systems against cyber attacks. IEEE Trans. Autom. Control **63**, 2272–2279 (2017)
34. A.-Y. Lu, G.-H. Yang, Input-to-state stabilizing control for cyber-physical systems with multiple transmission channels under denial-of-service. IEEE Trans. Autom. Control **63**, 1813–1820 (2017)
35. V. Dolk, P. Tesi, C. De Persis, W. Heemels, Event-triggered control systems under denial-of-service attacks. IEEE Trans. Control Netw. Syst. **4**(1), 93–105 (2017)
36. X. Jin, W.M. Haddad, T. Yucelen, An adaptive control architecture for mitigating sensor and actuator attacks in cyber-physical systems. IEEE Trans. Autom. Control **62**, 6058–6064 (2017)
37. A. Abbaspour, M. Sanchez, A. Sargolzaei, K. Yen, N. Sornkhampan, Adaptive neural network based fault detection design for unmanned quadrotor under faults and cyber attacks, in *25th International Conference on Systems Engineering (ICSEng)* (IEEE, Piscataway, 2017)
38. T. Basar, G. Olsder, *Dynamic Noncooperative Game Theory*, 2nd edn. (Society for Industrial and Applied Mathematics, Philadelphia, 1999)
39. Q. Zhu, T. Basar, Robust and resilient control design for cyber-physical systems with an application to power systems, in *50th IEEE Conference on Decision and Control and European Control Conference* (2011), pp. 4066–4071
40. T. Basar, A dynamic games approach to controller design: disturbance rejection in discrete-time. IEEE Trans. Autom. Control **36**(8), 936–952 (1991)
41. C. Seah, I. Hwang, Stochastic linear hybrid systems: modeling, estimation, and application in air traffic control. IEEE Trans. Control Syst. Technol. **17**, 563–575 (2009)
42. C.E. Garcia, D.M. Prett, M. Morari, Model predictive control: theory and practice—a survey. Automatica **25**, 335–348 (1989)
43. E. Scholte, M.E. Campbell, Robust nonlinear model predictive control with partial state information. IEEE Trans. Control Syst. Technol. **16**, 636–651 (2008)
44. G.E. Dullerud, F. Paganini, *A Course in Robust Control Theory: A Convex Approach* (Springer, New York, 2005)
45. K.S. Narenda, S.S. Tripathi, Identification and optimization of aircraft dynamics. AIAA J. Aircraft **10**(4), 193–199 (1973)

Chapter 4
Control and Safety of Autonomous Vehicles with Learning-Enabled Components

Somil Bansal and Claire J. Tomlin

Due to recent surge of interest in the use of unmanned aerial systems (UASs) for civil applications such as package delivery, aerial surveillance, and disaster response, among many others [1–5], our airspace may in the near future contain up to thousands of unmanned aerial vehicles (UAVs), potentially in close proximity of humans, other UAVs, and other important assets. The story is very similar on roads: recently, there has been a huge push on developing fully autonomous cars. The potential advantages include improved traffic efficiency, and low pollution, among others. Since these systems are safety critical, it is important to ensure that the system satisfies safety specifications at all times. For example, for UASs, we may want to ensure that the vehicle reaches its destination while avoiding collision with buildings and other vehicles during its flight. Similarly, for an autonomous car, we want to ensure that it stops at a STOP sign or a red light, does not collide with other cars, and stays clear of pedestrians. Thus, verifying the safety specifications for general complex systems, such as autonomous cars, is an important but a challenging problem.

There are three key challenges in verifying real-world autonomous systems: First, most of the available verification tools rely on an accurate dynamics model of the system. However, for complex real-world systems, it might be hard to obtain an accurate dynamics model. In other cases, a dynamics model might be available but it might be too complex. In such scenarios, it might be challenging to even satisfactorily control the system; verifying a system is only an additional layer of complexity. For example, if an accurate model of UAS is not available, it might be hard to design a controller that takes it from one position to another, even when no other obstacles or vehicles are present in the system.

S. Bansal · C. J. Tomlin (✉)
Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA, USA
e-mail: somil@berkeley.edu; tomlin@berkeley.edu

H. Yu et al. (eds.), *Safe, Autonomous and Intelligent Vehicles*,
Unmanned System Technologies, https://doi.org/10.1007/978-3-319-97301-2_4

Second, real-world systems often operate in partially observable environments, which are unknown beforehand. For example, for an autonomous car, it is hard to predict all the cars/pedestrians that it will encounter during its operation. It is only during the actual operation that the car will get to see what is out there on the road. We thus need to ensure that the designed controller: (a) can handle such a priori unknown situations and (b) satisfies the safety specifications for *all* such possible environment configurations. This is again a very challenging problem since specifying all possible environment configurations itself might be extremely hard, let alone their verification.

Finally, real-world autonomous systems often consist of multiple agents and verifying multi-agent systems is in general harder, especially when agents have different objectives.

In this chapter, we summarize some of the recent efforts in overcoming the first and the second challenge. In particular, when the system dynamics model is unknown or when the system is operating in an unknown environment, Machine Learning (ML) and Artificial Intelligence (AI)-based techniques have been proposed to design a controller for the system. These techniques have the potential to transform real-world autonomous systems because of their ability to design a controller based on the data collected on the system. However, given the safety-critical nature of autonomous vehicle systems, this transformation requires approaches which are robust, resilient, and safe. Here, we present both the proposed learning-based schemes as well as efforts made toward verifying such schemes. We start with presenting Hamilton–Jacobi (HJ) reachability theory, which is a safety verification tool that can be used to verify a system when its dynamics model and a "model" of the environment are known. Then, we move on to the scenarios where learning-based components are involved in the system, either due to unknown dynamics or uncertain environment, and discuss how HJ reachability-based algorithms can be extended to verify such systems.

4.1 Hamilton–Jacobi Reachability

Hamilton–Jacobi (HJ) reachability analysis is an important formal verification method for guaranteeing performance and safety properties of dynamical systems. In general, verification of systems is challenging for many reasons, even when the system dynamics model and the environment it is operating in are known perfectly. First, all possible system behaviors must be accounted for. This makes most simulation-based approaches insufficient, and thus formal verification methods are needed. Second, many practical systems are affected by disturbances in the environment, which can be unpredictable, and may even contain adversarial agents. In addition, these systems often have high-dimensional state spaces and evolve in continuous time with complex, nonlinear dynamics.

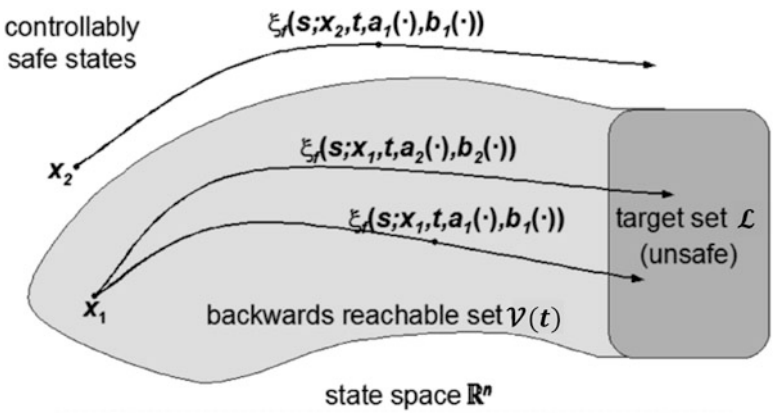


Fig. 4.1 Target set and backward reachable set. Several trajectories are shown starting at the same time t but from different states x and subject to different input signals $a(\cdot)$ and $b(\cdot)$. Input signal $a(\cdot)$ is chosen to drive the trajectory away from the target set, while input signal $b(\cdot)$ is chosen to drive the trajectory toward the target. Figure taken from [6]

HJ reachability can overcome some of the above challenges and has been used for the safety analysis of many autonomous systems in the past decade. Its advantages include compatibility with general nonlinear system dynamics, formal treatment of bounded disturbances, and the availability of well-developed numerical tools that can account for all possible system behaviors. In this section, we first present an overview of the basic HJ reachability theory. We then demonstrate an application of HJ reachability for provably safe and scalable trajectory planning for multi-vehicle systems: how a group of vehicles in the same vicinity can reach their destinations while avoiding situations which are considered dangerous, such as collisions.

4.1.1 Backward Reachable Set (BRS)

In reachability theory, we are often interested in computing the *backward reachable set* of a dynamical system. This is the set of states such that the trajectories that start from this set can reach some given target set (see Fig. 4.1). If the target set consists of those states that are known to be unsafe, then the BRS contains states which are potentially unsafe and should therefore be avoided. As an example, consider collision avoidance protocols for two aircraft in En-Route airspace. The target set would contain those states that are already "in loss of separation," such as those states in which the aircraft are within the five-mile horizontal separation distance mandated by the Federal Aviation Administration. The backward reachable set contains those states which could lead to a collision, despite the best possible control actions. We typically formulate such safety-critical scenarios in terms of a two-player game, with Player 1 and Player 2 being the control inputs. For example,

Player 1 could represent one aircraft, Player 2 another, with Player 1's control input being treated as the control input of the joint system, and with Player 2's control input being treated as the "disturbance."

Mathematically, let $x \in \mathbb{R}^n$ be the system state, which evolves according to the ordinary differential equation (ODE):

$$\dot{x}(s) = f(x(s), a(s), b(s)), s \in [t, 0], a(s) \in \mathcal{A}, b(s) \in \mathcal{B}, \tag{4.1}$$

where $a(s)$ and $b(s)$ denote the input for Player 1 and Player 2, respectively. We assume that the control functions $a(\cdot)$, $b(\cdot)$ are drawn from the set of measurable functions[1]:

$$a(\cdot) \in \mathbb{A}(t) = \{\phi : [t, 0] \to \mathcal{A} : \phi(\cdot) \text{ is measurable}\}$$
$$b(\cdot) \in \mathbb{B}(t) = \{\phi : [t, 0] \to \mathcal{B} : \phi(\cdot) \text{ is measurable}\}$$

where $\mathcal{A} \subset \mathbb{R}^{n_u}$ and $\mathcal{B} \subset \mathbb{R}^{n_d}$ are compact and $t < 0$. The system dynamics, or flow field, $f : \mathbb{R}^n \times \mathcal{A} \times \mathcal{B} \to \mathbb{R}^n$, is assumed to be uniformly continuous, bounded, and Lipschitz continuous in x uniformly in[2] a and b. Therefore, given $a(\cdot) \in \mathbb{A}$ and $b(\cdot) \in \mathbb{B}$, there exists a unique trajectory solving (4.1) [7]. We will denote solutions, or trajectories of (4.1) starting from state x at time t under control $a(\cdot)$ and $b(\cdot)$ as $\zeta(s; x, t, a(\cdot), b(\cdot)) : [t, 0] \to \mathbb{R}^n$. ζ satisfies (4.1) with an initial condition almost everywhere:

$$\begin{aligned} \frac{d}{ds}\zeta(s; x, t, a(\cdot), b(\cdot)) &= f(\zeta(s; x, t, a(\cdot), b(\cdot)), a(s), b(s)) \\ \zeta(t; x, t, a(\cdot), b(\cdot)) &= x \end{aligned} \tag{4.2}$$

Intuitively, a BRS represents the set of states $x \in \mathbb{R}^n$ from which the system can be driven into some set $\mathcal{L} \subseteq \mathbb{R}^n$ *within* a time horizon of duration $|t|$. We call $\mathcal{L}$ the "target set." We assume that Player 1 will try to steer the system away from the target with her input, and Player 2 will try to steer the system toward the target with her input. Consequently, we want to compute the following BRS:

$$\mathcal{V}(t) = \{x : \exists \gamma \in \Gamma(t), \forall a(\cdot) \in \mathbb{A}, \exists s \in [t, 0], \zeta(s; x, t, a(\cdot), \gamma[a](\cdot)) \in \mathcal{L}\}, \tag{4.3}$$

where $\Gamma(\cdot)$ in (4.3) denotes the feasible set of strategies for Player 2.

[1] A function $f : X \to Y$ between two measurable spaces (X, Σ_X) and (Y, Σ_Y) is said to be measurable if the preimage of a measurable set in Y is a measurable set in X, that is: $\forall V \in \Sigma_Y$, $f^{-1}(V) \in \Sigma_X$, with Σ_X, Σ_Y σ-algebras on X,Y.

[2] We will omit the notation (s) from variables such as x and a when referring to function values here on.

The computation of the BRS in (4.3) requires solving a differential game between Player 1 and Player 2 [6, 8], as discussed later in this section, but since we are in a differential game setting, it is important to first address what information the players know about each other's decisions. This information pattern directly affects their strategies and, consequently, the outcome of the game. In reachability problems, we assume that the Player 2 uses only non-anticipative strategies $\Gamma(\cdot)$ [6], defined as follows:

$$\begin{aligned}\gamma \in \Gamma(t) := \{\mathcal{N} : \mathbb{A}(t) \to \mathbb{B}(t) : a(r) = \hat{a}(r) \text{ a.e. } r \in [t, s] \\ \Rightarrow \mathcal{N}[a](r) = \mathcal{N}[\hat{a}](r) \text{ a.e. } r \in [t, s]\}\end{aligned} \tag{4.4}$$

That is, Player 2 cannot respond differently to two Player 1 controls until they become different. Yet, in this setting, Player 2 has the advantage of factoring in Player 1's choice of input at every instant t and adapting its own accordingly. Thus, Player 2 has an *instantaneous informational advantage*, which allows us to establish safety guarantees under the worst-case scenarios. One particular class of problems in which the notion of non-anticipative strategies is applicable is robust control problems, in which one wants to obtain the robust control (Player 1) with respect to the worst-case disturbance (Player 2), which can then be modeled as an adversary with the instantaneous informational advantage (not because this disturbance is in fact reacting to the controller's input, but rather, because out of all possible disturbances, there will be one that will happen to be the worst possible given the chosen control).

Under the assumption of non-anticipative strategies, one can more generally compute the so-called *reach-avoid set* [9], the set of all states such that the trajectories that start from this set can reach a given target set while avoiding some other given set. For example, it will be natural to compute a reach-avoid set for a vehicle, where the target set could be its destination and avoid set could be the set of obstacles or other vehicles on its way to the destination. Mathematically, we want to compute the following set:

$$\begin{aligned}\mathcal{V}(t) =& \{x : \forall \gamma \in \Gamma(t), \exists a(\cdot) \in \mathbb{A}, \forall \tau \in [t, 0], \zeta(\tau; x, t, a(\cdot), \gamma[a](\cdot)) \notin \mathcal{G}, \\ & \exists s \in [t, 0], \zeta(s; x, t, a(\cdot), \gamma[a](\cdot)) \in \mathcal{L}\},\end{aligned} \tag{4.5}$$

Finally, the obstacle function $\mathcal{G}$ in (4.5), and hence the set to be avoided, can change over time.

The differential game that must be solved in order to compute the BRS in (4.3) or in (4.5) is a "game of kind" rather than a "game of degree," i.e., games in which the outcome is determined by *whether or not* the state of the system reaches a given configuration under specified constraints at any time within the duration of the game. The good news is that an approach known as the *level set method* can transform these games of kind into games of degree in an analytically sound and computationally tractable way. In particular, it can be shown that the BRS can be

computed by solving the following final value double-obstacle Hamilton–Jacobi variational inequality (HJ VI) [10]:

$$\max\left\{\min\{D_t V(t,x)+H(t,x,\nabla V(t,x)), l(x)-V(t,x)\}, -g(t,x)-V(t,x)\right\}=0, \quad t \leq t_f$$
$$V(t_f,x) = \max\{l(x), -g(t_f,x)\} \tag{4.6}$$

In (4.6), the function $l(x)$ is the implicit surface function representing the target set $\mathcal{L} = \{x : l(x) \leq 0\}$. Similarly, the function $g(t,x)$ is the implicit surface function representing the time-varying obstacles $\mathcal{G}(t) = \{x : g(t,x) \leq 0\}$. The BRS $\mathcal{V}(t,t_f)$ is given by:

$$\mathcal{V}(t,t_f) = \{x : V(t,x) \leq 0\}. \tag{4.7}$$

The Hamiltonian, $H(t,x,\nabla V(t,x))$, depends on the system dynamics, and the role of control and disturbance. In addition, the Hamiltonian is an optimization that produces the optimal control $a^*(t,x)$ and optimal disturbance $b^*(t,x)$, once V is determined. For BRSs, whenever the existence of a control ("$\exists a$") or disturbance is sought, the optimization is a minimum over the set of controls or disturbance. Whenever a BRS characterizes the behavior of the system for all controls ("$\forall a$") or disturbances, the optimization is a maximum. For the reach-avoid set in (4.5), the Hamiltonian is given by:

$$H(t,x,\nabla V(t,x)) = \min_{a\in\mathcal{A}} \max_{b\in\mathcal{B}} \nabla V(t,x)\cdot f(x,a,b). \tag{4.8}$$

We refer the interested readers to [11] for further details on the computation of BRS.

4.1.2 *Application: Provably Safe Multi-Vehicle Trajectory Planning*

We now illustrate how reachability can be used for safe trajectory planning for large-scale autonomous systems. In particular, we simulate a 50-vehicle UAV system where UAVs are flying through a city environment. This setup can be representative of many UAV applications, such as package delivery, aerial surveillance, etc.

We grid the airspace of the city of San Francisco (SF) in California, USA and use it as the position space for the UAVs, as shown in Fig. 4.2. Each box in Fig. 4.2 represents a 1-km^2 area of SF. The origin point for the vehicles is denoted by the blue star. Four different areas in the city are chosen as the destinations for the vehicles. Mathematically, the target set $\mathcal{L}_i$ of the vehicle i, Q_i, is a circle of radius r in the position space, i.e., each vehicle is trying to reach some desired set of destination positions. In terms of the state space x_i, the target sets are defined as:

$$\mathcal{L}_i = \{x_i : \|p_i - c_i\|_2 \leq r\} \tag{4.9}$$

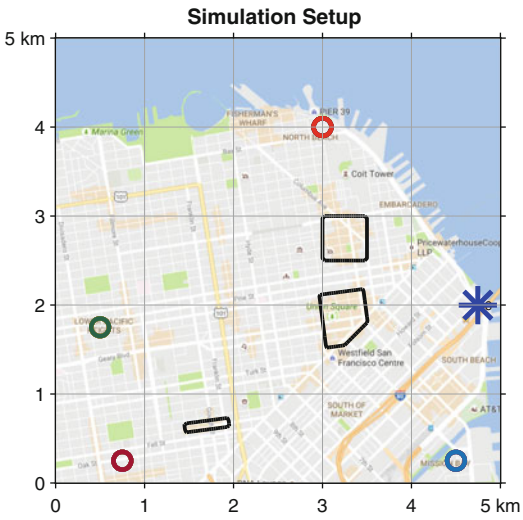


Fig. 4.2 Simulation setup. A 25-km^2 area in the city of San Francisco is used as the space for the UAVs. The vehicles originate from the blue star and go to one of the four destinations, denoted by circles. Tall buildings in the downtown area are used as static obstacles, represented by the black contours

where p_i is the position component of the state and c_i is the center of the target circle. In this simulation, we use $r = 100$ m. The four targets are represented by four circles in Fig. 4.2. The destination of each vehicle is chosen randomly from these four destinations. Finally, some areas in downtown SF and the city hall are used as representative static obstacles for the vehicles, denoted by black contours in Fig. 4.2.

To make our simulations as close as possible to real scenarios, we assume that the disturbances are present too, for example, wind that affects the vehicles' position evolution. We also impose the control bounds on vehicles that align with the modern UAV specifications. The goal of the vehicles is to reach their destinations while avoiding a collision with the other vehicles or the static obstacles, despite these disturbances and control bounds.

Although reachability is well-suited for these robustness requirements, simultaneous analysis of all vehicles is intractable, since the joint state space of this 50-vehicle system is very high-dimensional. Instead, vehicles are assigned a strict priority ordering, with lower-priority vehicles treating higher-priority vehicles as moving obstacles. Robust path planning around these induced "obstacles" is done using time-varying formulation of reachability discussed in Sect. 4.1.1. The result is a reserved "space–time" in the airspace for each vehicle, which is dynamically feasible to track even when the vehicle experiences disturbances and performs collision avoidance against all other vehicles. Simulations of the algorithm for different combinations of wind speeds and UAV densities are shown in Fig. 4.3. Depending on the vehicle density and the wind speed, different trajectory patterns automatically emerge out of the reachability analysis to ensure that the vehicles remain clear of all static obstacles as well as of each other, despite the disturbance in the dynamics. Thus, reachability gives us a state-feedback control policy for each vehicle that is guaranteed to ensure safety at all times. Further details about the algorithm and simulations can be found in [12–14].

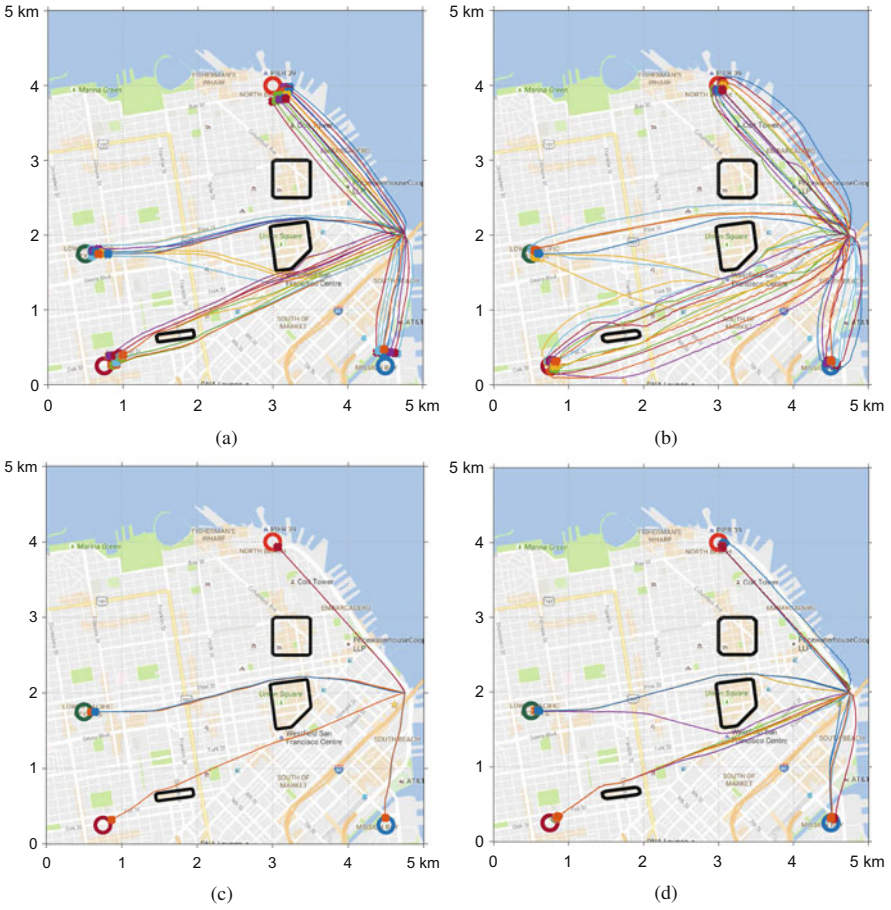


Fig. 4.3 Natural lane forming of UAVs due to disturbance rejection and vehicle density. Figures taken from [12]. (**a**) 6 m/s wind, high UAV density. (**b**) 11 m/s wind, high UAV density. (**c**) 6 m/s wind, medium UAV density. (**d**) 11 m/s wind, medium UAV density

4.1.3 Limitations of HJ Reachability

Even though HJ reachability is a very powerful tool for a formal safety analysis of autonomous systems, it relies on a few assumptions:

(1) A *validated* system dynamics model must be known to perform reachability analysis. For example, to use reachability for safe trajectory planning in Sect. 4.1.2, a 3D Dubins car dynamics model is used. If the actual system has a different dynamic behavior, then the controller designed using reachability is not guaranteed to be safe for the actual system. This assumption can be rather restrictive for real-world autonomous systems, where obtaining a dynamics model from first principles could be too hard or too cumbersome depending on the system complexity.

(2) Either the environment is fully observable or a conservative model of the environment is available. The term *environment* here refers to the part of the system that we cannot control directly. For example, static obstacles, wind, intruder vehicles, etc., are the parts of the environment in the multi-vehicle case study presented in Sect. 4.1.2. There it is assumed that all static obstacles (the Black contours) are known beforehand [13]. For wind, it is assumed that bounds on the wind speed are known beforehand. Similarly, for intruder vehicles, it is assumed that their dynamics model is known. These assumptions, although often required for verification purposes, can be impractical and may lead to a very conservative control performance on real-world systems.

A natural question to ask is what sort of guarantees can we make when an accurate model is not known beforehand and/or when the environment is only partially observable? Even simpler question to ask is can we even control the system to a satisfactory level if the above assumptions do not hold, let alone the safety guarantees?

When the system model is unknown or only partially known, one possibility is to "refine" the model using the data collected on the system. Similarly, when the environment is only partially known, learning-based methods provide a promising alternative to learn the appropriate "reflexes" that a system should have to handle the unforeseen situations it encounters while operating in the environment. In the remainder of this chapter, we present some concrete learning architectures to control systems when the above assumptions do not hold. Finally, we discuss how safety can be incorporated in these learning architectures.

4.2 Learning-Based Model Refinement

When a dynamics model of the system is unknown or only partially known (e.g., only a nominal model is available), learning-based methods can be used to design a controller for the system. These methods can be broadly divided into two categories: Model-Based (MB) approaches and Model-Free (MF) approaches. Model-based approaches refine a dynamics model using data or learn it from scratch. This data-driven model is subsequently used for designing a controller. Model-free approaches aim to directly design a controller using data without maintaining a dynamics model. Both of these approaches have different strengths and limitations [15]. Typically, MF approaches are very effective at learning complex controllers, but the convergence might require millions of trials on the actual system and lead to (globally suboptimal) local minima. Moreover, the learned controller is hard to generalize to new tasks and environments. This renders MF methods impractical for most real-world systems. On the other hand, MB approaches have the theoretical benefit of being able to better generalize to new tasks and environments, and in practice they can drastically reduce the number of trials required [16, 17]. Moreover, the learned model can be used in conjunction with verification tools, such as HJ reachability, for performing a safety analysis of the system, something that remains challenging for MF approaches.

In this section, we discuss some learning-based approaches to obtain a data-driven model of the system, and use it for control purposes. In the next section, we will discuss how these data-driven models can be further used for a safety analysis of the system.

4.2.1 Function Approximator-Based Model Learning

Data-driven model learning has been studied in the literature mostly under the banner of system identification. System identification focuses on both building a mathematical model of the system from measured data and designing the optimal experiments for efficiently generating informative data for fitting such models [18–21]. System identification can be used for either gray-box modeling or black-box modeling. In gray-box modeling, it is assumed that a model structure is known a priori but some model parameters are unknown which are estimated based on data. Parameter estimation is relatively easier if the model form is known but this is rarely the case, in which case, a black-box modeling is required which also learns the model form. Several approaches have been proposed for black-box modeling of both linear and highly complex nonlinear models under the assumption that the system can be represented as a NARMAX model. NARMAX model essentially assumes that the output at a given time is an expansion of past inputs, outputs, and noise terms. The another model that is widely used for system identification and what we focus on here is state-space model, which can also be thought of as a special case of NARMAX model when the system state is treated as its output. Similar to NARMAX models, there are several ways to identify state-space models [18].

In the simplest case, the state-space model identification or dynamics learning problem can be formalized as a regression problem: given a state $x(t)$ and a control input $a(t)$ at time t, the goal is to predict the next state $x(t+1)$, the state of the system at time $(t+1)$. More formally, given a system $\mathcal{S}$, our goal is to find a model $\mathcal{M}$ of $\mathcal{S}$ that minimizes the one-step prediction error:

$$\mathcal{M} := \arg\min_{f \in \mathcal{F}} \sum_{i,t} \| f(\hat{x}(i,t), \hat{a}(i,t)) - \hat{x}(i,t+1) \|, \tag{4.10}$$

where $\{\hat{x}(\cdot,t), \hat{a}(\cdot,t), \hat{x}(\cdot,t+1)\}$ is the state-control transition data collected from the actual system (for example, this can be done by executing some nominal controller on the actual system) and $\mathcal{F}$ denotes the class of functions over which we optimize. Note that instead of minimizing a single-step prediction error in (4.10), one can also minimize a multistep prediction error.

Different parametric and nonparametric function classes can be chosen in (4.10) to approximate the system dynamics. Recently, function approximators-based state-space models, such as Neural Networks (NN), and Gaussian Processes (GP), have received a lot of attention, primarily because they are conceptually simple, easy to train and to use, and have excellent approximation properties [22–28]. NNs

Fig. 4.4 The Crazyflie 2.0

have recently been used to learn the dynamics models for a range of autonomous systems, such as aerial vehicles [29, 30], robotic arms [31], walking robots [32], etc. For example, in [29], authors learned an NN-based 12-dimensional dynamics model for Crazyflie 2.0, a nanoquadrotor developed by Bitcraze (see Fig. 4.4). A simple 2-layer feed-forward network was trained during the identification process using supervised learning on the data collected on Crazyflie. This model was then successfully used to control Crazyflie on complex maneuvers, indicating the potential of NNs as a system identification tool. Similarly, other deterministic and probabilistic function approximators, such as Recurrent Neural Networks (RNNs), Bayesian Neural Networks (BNNs), Gaussian Processes (GPs), etc., have also been used to learn the dynamics models [33].

4.2.2 Goal-Driven Model Learning

Even though the model-based approaches described above are generally data-efficient in terms of the amount of data they need from the real-system before being able to design a good controller, learning a globally accurate model from data is still a challenging task. Therefore, it might be desirable to take a goal-driven approach where one focuses on learning the minimal model required to achieve a good control performance on a particular task at hand, rather than learning a globally accurate prediction model [34–39]. This might also be desirable when a dynamics model of the actual system is known, but it is too complex to design a controller directly. In such cases, a goal-driven approach can be used to obtain a simpler "abstraction" of the actual system that is sufficient to design a good controller for the actual system. For example, in [40] a Bayesian Optimization (BO)-based active learning framework, aDOBO, has been proposed to learn a dynamics model that achieves the best control performance for a given task on the actual system, directly based on the performance observed in experiments on the system. The aDOBO framework is illustrated in Fig. 4.5. The current linear dynamics model, together

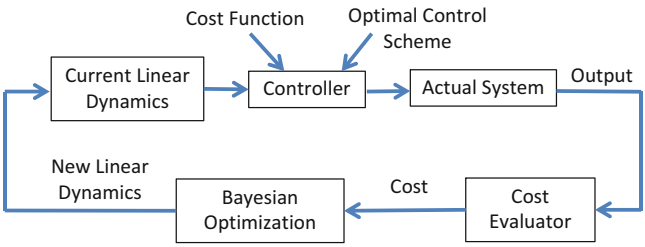


Fig. 4.5 aDOBO: a Bayesian optimization-based active learning framework for optimizing the dynamics model for a given cost function, directly based on the observed cost values

with the cost function (also referred to as task), is used to design a controller with an appropriate optimal control scheme. The cost (or performance) of the controller is evaluated in closed-loop operation with the actual (unknown) physical plant. BO uses this performance information to iteratively update the dynamics model to improve the performance. This procedure corresponds to optimizing the linear system dynamics with the purpose of maximizing the performance of the final controller. Hence, unlike traditional system identification approaches, it does not necessarily correspond to finding the most accurate dynamics model, but rather the model yielding the best controller performance when provided to the optimal control method used to design the controller. It is also illustrated that the true dynamics of the system may not even be required as far as the control performance for a given objective is concerned, and a much simpler dynamics model can be used for the control purposes. For example, it is illustrated that a linear time-invariant model can be sufficient for controlling complex unstable nonlinear systems, such as the cart-pole system, and their might exist a continuum of such linear models.

4.3 Safety Analysis of Learned Models

In Sect. 4.2, we discussed how the proven efficacy of learning-based control schemes strongly motivates their application to robotic systems operating in the physical world. However, for safety-critical systems, it is often important to guarantee correct operation during the data collection and the learning process itself. Moreover, it is important to ensure that the learned model accurately captures the system behavior in the region of interest, if this model is being used for further analysis of the system, for example, to compute reachable sets. In this section, we present some approaches proposed in the literature to address these two challenges.

4.3.1 Safety During Model Learning

We first start with the problem of ensuring safety during the learning process. One line of approaches that is proposed in the literature for this endeavor is to start with an initial exploration region that is known to be safe. Data is then collected in the safe exploration region and the safe exploration region is slowly expanded based on the collected data while ensuring safety at all times, until a desired level of accuracy is achieved. The proposed methods, however, differ in how this initial safe exploration region is obtained as well as how the safe exploration region is expanded.

In [41, 42], such a safety framework is proposed based on HJ reachability methods that can work in conjunction with an arbitrary learning algorithm. There it is assumed that a nominal dynamics model of the system is available initially, and a bound on the model uncertainty is known. The initial safe exploration region is computed using HJ reachability on the nominal model. This exploration region guarantees constraint satisfaction while minimally interfering with the learning process. The authors further introduce a Bayesian mechanism that refines this exploration region as the system acquires new evidence and refines its dynamics model, thus reducing the initial conservativeness when appropriate. The result is a least-restrictive, safety-preserving control law that intervenes only when: (a) the computed safety guarantees require it, or (b) confidence in the computed guarantees decays in light of new observations. The authors provide safety guarantees combining probabilistic and worst-case analysis and demonstrate the proposed framework experimentally on a quadrotor vehicle. Even though safety analysis is based on a simple point-mass model, the quadrotor is able to successfully run policy-gradient reinforcement learning without crashing, and safely retracts away from a strong external disturbance introduced during one of the experiments.

However, when a nominal model of the system is not available beforehand or if it is too conservative, alternative safety-ensuring schemes might be required. To that end, SAFEOPT has been proposed [43], where safety is ensured under the assumption that the dynamics model satisfies regularity conditions expressed via a Gaussian process prior. SAFEOPT has also been applied to tune the controller parameters in a safety-aware fashion [44]. In particular, the algorithm starts with an initial, low-performance controller that is known to be safe. SAFEOPT explores in the safe region using a Bayesian mechanism, and automatically optimizes the parameters of the controller while guaranteeing safety with a high probability.

4.3.2 Model Validation Before Deployment

Other than ensuring that the system satisfies the safety constraints during the data collection process, we also need to make sure that the learned model is indeed representing the actual system accurately. Thus, a learned model needs to be

validated before it is deployed for controller synthesis or safety analysis. Broadly, validation refers to formally quantifying the quality/ability of a model to mimic the (dynamic) behavior of the actual system. However, the sophistication of the required validation certificate depends on the objective behind obtaining a model. For example, if a vehicle model is obtained only to design a trajectory-tracking controller, then it is sufficient to ensure that the model behaves as intended around the desired trajectory. On the other hand, if the model is obtained for designing a provably safe controller, it is important to ensure that the model behavior matches with that of the actual system in the entire state space of operation.

Several methods have been proposed in the literature to validate an abstraction (or model), $\mathcal{M}$, of the actual system, $\mathcal{S}$, utilizing the notion of simulation metric, where the quality of an abstraction is quantified via metrics that specify how close the trajectories of $\mathcal{S}$ and $\mathcal{M}$ are [45–54]. In particular, the worst-case distance between the system and the model trajectories across all possible control laws is computed. This distance bound can then be used for a safety analysis of the actual system using the model. For example, suppose that the actual system is a vehicle and we want to design a controller for the vehicle to ensure that it avoids a particular region in the position space (e.g., an obstacle) at all times. This position space can be augmented by the computed distance bound and then HJ reachability can be used on the model to design a controller that avoids the augmented region at all times. The same controller will then ensure that the actual system avoids the obstacle region at all times.

However, since the above approaches validate the open-loop behavior of the model against that of the system, they may lead to quite conservative bounds on the quality of $\mathcal{M}$ as an abstraction of $\mathcal{S}$ when the model is used for controller synthesis for a specific set of tasks. To overcome this limitation, authors in [55] propose a context-specific framework, CuSTOM, to quantify the quality of an abstraction for the controller synthesis purposes. Here, context refers to a specific (potentially continuous) set of control tasks for which the system designer wants to synthesize a controller. As opposed to the open-loop validation, the validation of closed-loop systems is considered, where the controller has been designed using $\mathcal{M}$ for the particular task. Thus, only those behaviors of the model are validated that are relevant for the underlying task rather than all system behaviors. This subtlety of CuSTOM proves to be particularly important for validating data-driven models where obtaining a good open-loop abstraction across the entire state space is extremely challenging; thus, such models will hardly pass a typical simulation metric-based validation test, even when the abstraction is good enough for synthesizing a controller for the actual system.

4.4 Learning in Partially Observable environments

In Sect. 4.2, we discussed how one can use learning-based schemes to obtain a dynamics model of the system if it is not known a priori. In Sect. 4.3, we discussed how one can ensure safety during the learning process itself. Moreover, we discussed

some validation schemes that can be used to validate the learned model before it is subsequently used for control or verification purposes. Now, we turn our attention to how a learned model or a first-principle model can be used to control a system in a partially observable environment.

To address this problem, we first need to discuss what information is available to the system while it is operating in the environment. When a system operates in an environment, the environment information is often captured through various perception systems, which is often useful or even essential to design a controller. One of the most common sensors used in robotics to obtain this information is a camera, primarily because of its cost efficiency. To highlight the challenges associated with partially observable environments, we will assume that the system dynamics are known as well as its state is full observable. This dynamics model, for example, can be obtained either using the first principles or the methods outlined in the previous sections. The cases where these assumptions do not hold only complicate the challenges further.

Even though both computer vision and control theory have made a significant progress in the last few years, it remains a challenge to incorporate vision-based perception systems in the model-based control schemes [56] in a general manner. In fact, one of the important reasons for recent popularity of model-free approaches is their ability to learn a controller directly from the raw images [57, 58], enabling these approaches to control the system in partially observable environments. A learning architecture that has been particularly successful at this endeavor is the so called end-to-end learning [58], wherein the perception and control systems are trained simultaneously. In particular, a Convolutional Neural Network (CNN) is trained that takes raw image as input and outputs the control command to be applied to the system; thus, one need not to "hardcode" the features learned by the perception system. This facilitates the learning of the "right features" by the perception systems, i.e., the ones that are essential for the control purposes. This approach has been successfully used for various robotic tasks that require a close coordination between perception and control systems [58–60]. However, end-to-end model-free approach has several limitations: (1) these approaches may not be robust to changes in the system, which is inevitable for real-world systems, (2) a direct learning of control commands could be significantly harder due to its high frequency nature and is often unnecessary, and (3) the safety analysis of such architectures is quite challenging.

On the other hand, there are model-based approaches that design/learn the perception component in a disjoint fashion, and then incorporate it with the planning module. For example, in [61], a new algorithm is developed called FaSTrack, Fast and Safe Tracking, for navigation purposes in partially observable environments. A trajectory planner using simplified dynamics to plan quickly can be incorporated into the FaSTrack framework, which provides a safety controller for the vehicle along with a guaranteed tracking error bound. By formulating a differential game and leveraging HJ reachability's flexibility with respect to nonlinear system dynamics, this tracking error bound is computed in the error coordinates, which evolve according to the error dynamics, and captures all possible deviations due to dynamic

infeasibility of the planned path and external disturbances. The simplified planner allows to solve the planning problem in real time as it gathers information about the environment and updates the environment map, while maintaining safety at all times. Thanks to the known dynamics model and the reachability, the safety can be ensured despite a partially observable environment. However, FaSTrack has several limitations compared to end-to-end approaches: (1) it hardcodes the features that the perception component outputs (in this case, a map of the environment) and (2) it does not learn the semantics of the environment, which requires it to build the map from scratch in a new environment and renders it impractical for dynamics environments. These are precisely the limitations that end-to-end learning approaches are equipped to address.

Thus, a natural question to ask is can we use learning to obtain semantics-level information about the environment and combine it with the system dynamics model for control purposes and safety analysis? More importantly, can we do this in a general manner, where one need not hardcode the features that the perception component end-up learning, possibly training the perception component in a system-aware fashion? Such a framework will be able to leverage the full potential of learning while maintaining safety. In particular, it can learn representations that are general and required to solve the tasks at hand, but at the same time compatible with safety tools, such as HJI reachability. Since the planning is performed using a system dynamics model, such a framework will also be robust to small changes in the system. Finally, the learning process itself becomes easier as the learning architecture does not need to reason about the planning process anymore.

Controlling systems in partially observable environments based on raw images is an active area of research and different approaches have been proposed in the literature to address some of the above questions (see [56, 62–74] and references therein); however, a lot still remains to be answered. Regardless, it is very clear that this is an exciting era in autonomy, and machine learning will play a key role in this era. However, the questions posed in this chapter will be instrumental in deciding where exactly and in what capacity will learning be useful.

References

1. B.P. Tice, Unmanned aerial vehicles: The force multiplier of the 1990s. Airpower Journal **5**(1), 41–55 (1991)
2. W. DeBusk, Unmanned aerial vehicle systems for disaster relief: Tornado alley, in *Infotech@ Aerospace Conferences* (2010)
3. Amazon.com, Inc., Amazon Prime Air, 2016. Available: http://www.amazon.com/b?node=8037720011
4. AUVSI News, UAS aid in South Carolina tornado investigation, 2016. Available: http://www.auvsi.org/blogs/auvsi-news/2016/01/29/tornado
5. BBC Technology, Google plans drone delivery service for 2017, 2016. Available: http://www.bbc.com/news/technology-34704868
6. I. Mitchell, A. Bayen, C. Tomlin, A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games. IEEE Trans. Autom. Control **50**(7), 947–957 (2005)

7. E. Coddington, N. Levinson, *Theory of Ordinary Differential Equations* (Tata McGraw-Hill Education, 1955)
8. J. Lygeros, On reachability and minimum cost optimal control. Automatica **40**(6), 917–927 (2004)
9. K. Margellos, J. Lygeros, Hamilton-Jacobi formulation for reach–avoid differential games. IEEE Trans. Autom. Control **56**(8), 1849–1861 (2011)
10. J. Fisac, M. Chen, C. Tomlin, S. Sastry, Reach-avoid problems with time-varying dynamics, targets and constraints, in *Conference on Hybrid Systems: Computation and Control* (2015)
11. S. Bansal, M. Chen, S. Herbert, C. Tomlin, Hamilton-Jacobi reachability: a brief overview and recent advances, in *Conference on Decision and Control* (2017)
12. M. Chen, S. Bansal, K. Tanabe, C. Tomlin, Provably safe and robust drone routing via sequential path planning: a case study in San Francisco and the Bay Area, 2017. Available: http://arxiv.org/abs/1705.04585
13. M. Chen, S. Bansal, J. Fisac, C. Tomlin, Robust sequential path planning under disturbances and adversarial intruder. IEEE Trans. Control Syst. Technol. (2018)
14. S. Bansal, M. Chen, J. Fisac, C. Tomlin, Safe sequential path planning of multi-vehicle systems under presence of disturbances and imperfect information, in *American Control Conference* (2017)
15. M.P. Deisenroth, G. Neumann, J. Peters, A survey on policy search for robotics. Found. Trends Robot. **2**(1–2), 1–142 (2013)
16. M.P. Deisenroth, D. Fox, C.E. Rasmussen, Gaussian processes for data-efficient learning in robotics and control. IEEE Trans. Pattern Anal. Mach. Intell. **37**(2), 408–423 (2015)
17. S. Levine, P. Abbeel, Learning neural network policies with guided policy search under unknown dynamics, in *Advances in Neural Information Processing Systems* (2014)
18. L. Ljung, System identification, in *Signal Analysis and Prediction* (Springer, 1998)
19. T. Söderström, P. Stoica, System identification (1989)
20. K.J. Åström, P. Eykhoff, System identification–a survey. Automatica **7**(2), 123–162 (1971)
21. J.-N. Juang, Applied system identification (1994)
22. O. Nelles, Nonlinear system identification: from classical approaches to neural networks and fuzzy models (2013)
23. S. Chen, S. Billings, Neural networks for nonlinear dynamic system modelling and identification. Int. J. Control **56**(2), 319–346 (1992)
24. S. Haykin, Neural networks: a comprehensive foundation (1998)
25. K.S. Narendra, K. Parthasarathy, Identification and control of dynamical systems using neural networks. IEEE Trans. Neural Netw. **1**(1), 4–27 (1990)
26. K.J. Hunt, D. Sbarbaro, R. Żbikowski, P.J. Gawthrop, Neural networks for control systems–a survey. Automatica **28**(6), 1083–1112 (1992)
27. R. Fierro, F.L. Lewis, Control of a nonholonomic mobile robot using neural networks. IEEE Trans. Neural Netw. **9**(4), 589–600 (1998)
28. A. Yeşildirek, F.L. Lewis, Feedback linearization using neural networks. Automatica **31**(11), 1659–1664 (1995)
29. S. Bansal, A. Akametalu, F. Jiang, F. Laine, C. Tomlin, Learning quadrotor dynamics using neural network for flight control, in *Conference on Decision and Control* (2016)
30. A. Punjani, P. Abbeel, Deep learning helicopter dynamics models, in *Conference on Robotics and Automation* (2015)
31. I. Lenz, R.A. Knepper, A. Saxena, DeepMPC: learning deep latent features for model predictive control, in *Robotics: Science and Systems* (2015)
32. A. Nagabandi, G. Yang, T. Asmar, G. Kahn, S. Levine, R. Fearing, Neural network dynamics models for control of under-actuated legged millirobots (2017, Preprint), arXiv:1711.05253
33. M. Deisenroth, C. Rasmussen, PILCO: a model-based and data-efficient approach to policy search, in *International Conference on Machine Learning* (2011)
34. J. Joseph, A. Geramifard, J. Roberts, J. How, N. Roy, Reinforcement learning with misspecified model classes, in *International Conference on Robotics and Automation* (2013)

35. P. Donti, B. Amos, J. Kolter, Task-based end-to-end model learning. Adv. Neural Inf. Proces. Syst. (2017)
36. C. Atkeson, Nonparametric model-based reinforcement learning. Adv. Neural Inf. Proces. Syst. (1998)
37. P. Abbeel, M. Quigley, A. Ng, Using inaccurate models in reinforcement learning, in *International Conference on Machine Learning* (2006)
38. M. Gevers, Identification for control: from the early achievements to the revival of experiment design. Eur. J. Control **11**(4–5), 335–352 (2005)
39. H. Hjalmarsson, M. Gevers, F. De Bruyne, For model-based control design, closed-loop identification gives better performance. Automatica **32**(12), 1659–1673 (1996)
40. S. Bansal, R. Calandra, T. Xiao, S. Levine, C. Tomlin, Goal-driven dynamics learning via Bayesian optimization, in *Conference on Decision and Control* (2017)
41. A. Akametalu, J. Fisac, J. Gillula, S. Kaynama, M. Zeilinger, C. Tomlin, Reachability-based safe learning with Gaussian processes, in *Conference on Decision and Control* (2014)
42. J. Fisac, A. Akametalu, M. Zeilinger, S. Kaynama, J. Gillula, C. Tomlin, A general safety framework for learning-based control in uncertain robotic systems (2017, Preprint), arXiv:1705.01292
43. Y. Sui, A. Gotovos, J. Burdick, A. Krause, Safe exploration for optimization with Gaussian processes, in *International Conference on Machine Learning* (2015)
44. F. Berkenkamp, A. Schoellig, A. Krause, Safe controller optimization for quadrotors with Gaussian processes, in *International Conference on Robotics and Automation* (2016)
45. R. Alur, T. A. Henzinger, G. Lafferriere, G.J. Pappas, Discrete abstractions of hybrid systems. Proc. IEEE **88**(7), 971–984 (2000)
46. C. Baier, J. Katoen, K.G. Larsen, *Principles of Model Checking* (MIT press, 2008)
47. A. Girard, G.J. Pappas, Approximate bisimulation: a bridge between computer science and control theory. Eur. J. Control **17**(5–6), 568–578 (2011)
48. G. Pola, A. Girard, P. Tabuada, Approximately bisimilar symbolic models for nonlinear control systems. Automatica **44**(10), 2508–2516 (2008)
49. A. Girard, G. Pola, P. Tabuada, Approximately bisimilar symbolic models for incrementally stable switched systems. IEEE Trans. Autom. Control **55**(1), 116–126 (2010)
50. M.L. Bujorianu, J. Lygeros, M.C. Bujorianu, Bisimulation for general stochastic hybrid systems, in *International Workshop on Hybrid Systems: Computation and Control* (Springer, 2005), pp. 198–214
51. J. Desharnais, A. Edalat, P. Panangaden, Bisimulation for labelled Markov processes. Inf. Comput. **179**(2), 163–193 (2002)
52. K.G. Larsen, A. Skou, Bisimulation through probabilistic testing. Inf. Comput. **94**(1), 1–28 (1991)
53. S. Strubbe, A. Van Der Schaft, Bisimulation for communicating piecewise deterministic Markov processes (CPDPs), in *International Workshop on Hybrid Systems: Computation and Control* (Springer, 2005), pp. 623–639
54. A. Abate, Approximation metrics based on probabilistic bisimulations for general state-space Markov processes: a survey. Electron. Notes Theor. Comput. Sci. **297**, 3–25 (2013)
55. S. Bansal, S. Ghosh, A. Sangiovanni Vincentelli, S. Seshia, C. Tomlin, Context-specific validation of data-driven models (2018, Preprint), arXiv:1802.04929
56. M. Watter, J. Springenberg, J. Boedecker, M. Riedmiller, Embed to control: a locally linear latent dynamics model for control from raw images. Adv. Neural Inf. Proces. Syst. (2015)
57. V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller, Playing atari with deep reinforcement learning (2013, Preprint), arXiv:1312.5602
58. S. Levine, C. Finn, T. Darrell, P. Abbeel, End-to-end training of deep visuomotor policies. J. Mach. Learn. Res. **17**(39), 1–40 (2016)
59. S. Gupta, J. Davidson, S. Levine, R. Sukthankar, J. Malik, Cognitive mapping and planning for visual navigation, in *Conference on Computer Vision and Pattern Recognition* (2017)
60. P. Agrawal, A. Nair, P. Abbeel, J. Malik, S. Levine, Learning to poke by poking: experiential learning of intuitive physics. Adv. Neural Inf. Proces. Syst. (2016)

61. S. Herbert, M. Chen, S. Han, S. Bansal, J. Fisac, C. Tomlin, FaSTrack: a modular framework for fast and guaranteed safe motion planning, in *Conference on Decision and Control* (2017)
62. K. Hashimoto, A review on vision-based control of robot manipulators. Adv. Robot. **17**(10), 969–991 (2003)
63. M. Achtelik, M. Achtelik, S. Weiss, R. Siegwart, Onboard IMU and monocular vision based control for MAVs in unknown in-and outdoor environments. Int. Conf. Robot. Autom. (2011)
64. A. Beyeler, J. Zufferey, D. Floreano, Vision-based control of near-obstacle flight. Auton. Robot. **27**(3), 201 (2009)
65. O. Shakernia, Y. Ma, T. Koo, S. Sastry, Landing an unmanned air vehicle: vision based motion estimation and nonlinear control. Asian Journal of Control **1**(3), 128–145 (1999)
66. G. Ros, A. Sappa, D. Ponsa, A. Lopez, Visual SLAM for driverless cars: a brief survey, in *Intelligent Vehicles Symposium (IV) Workshops*, vol. 2, 2012
67. A. Kim, R. Eustice, Perception-driven navigation: active visual SLAM for robotic area coverage, in *International Conference on Robotics and Automation* (2013)
68. J. Fuentes-Pacheco, J. Ruiz-Ascencio, J. Rendón-Mancha, Visual simultaneous localization and mapping: a survey. Artif. Intell. Rev. **43**(1), 55–81 (2015)
69. J. Aulinas, Y. Petillot, J. Salvi, X. Lladó, The SLAM problem: a survey. CCIA **184**(1), 363–371 (2008)
70. C. Finn, I. Goodfellow, S. Levine, Unsupervised learning for physical interaction through video prediction, in *Advances in Neural Information Processing Systems* (2016)
71. C. Finn, X. Tan, Y. Duan, T. Darrell, S. Levine, P. Abbeel, Deep spatial autoencoders for visuomotor learning, in *International Conference on Robotics and Automation* (2016)
72. T. Dreossi, A. Donzé, S.A. Seshia, Compositional falsification of cyber-physical systems with machine learning components, in *NASA Formal Methods Symposium* (Springer, Cham, 2017), pp. 357–372
73. S.A. Seshia, D. Sadigh, S.S. Sastry, Towards verified artificial intelligence. arXiv preprint arXiv:1606.08514
74. T. Dreossi, S. Jha, S.A. Seshia, Semantic adversarial deep learning. arXiv preprint arXiv:1804.07045

Chapter 5
Adaptive Stress Testing of Safety-Critical Systems

Ritchie Lee, Ole J. Mengshoel, and Mykel J. Kochenderfer

5.1 Introduction

Autonomy promises performance improvements and cost reductions that will revolutionize many industries. A variety of applications are currently being developed, including driverless cars, delivery drones, robotic servants, and automatic traffic management systems. These emerging systems are unprecedented in their scale and complexity and also in how they are being trusted to make decisions that are *safety-critical*. Because failures of these systems can result in loss of life and property, ensuring the safety of these systems is essential for deployment.

Validating the safety of autonomous systems can be very challenging, because these systems are inherently more complex and less understood than traditional systems. These systems generally have large state spaces and exhibit complex behavior. They also interact with external, stochastic, partially observable, and multi-agent environments over many time steps, which results in an exponential number of possible futures. Exhaustive consideration of all possibilities is generally intractable. Furthermore, because these systems are designed with safety in mind, failures can be extremely rare and difficult to reach.

Existing methods for validation of safety-critical systems can be broadly separated into two categories. In the first category, there are formal methods that construct a mathematical model of the system and check safety properties on the model using exhaustive enumeration or mathematical proofs. While these methods provide completeness guarantees, they have difficulty scaling to systems with very

R. Lee (✉) · O. J. Mengshoel
Carnegie Mellon University Silicon Valley, NASA Ames Research Park, Moffett Field, CA, USA
e-mail: ritchie.lee@sv.cmu.edu; ole.mengshoel@sv.cmu.edu

M. J. Kochenderfer
Stanford University, Stanford, CA, USA
e-mail: mykel@stanford.edu

H. Yu et al. (eds.), *Safe, Autonomous and Intelligent Vehicles*,
Unmanned System Technologies, https://doi.org/10.1007/978-3-319-97301-2_5

large and complex state space. In contrast, the second category includes simulation-based methods that evaluate a black box simulator at a finite number of paths. These methods do not require the internal details of the models used, so they are more flexible and can scale to larger systems and higher-fidelity models. While simulation-based methods cannot guarantee completeness, they are often very effective at finding realistic examples of failures. For very large and complex systems, stress testing in simulation may be the only practical alternative.

Monte Carlo is one of the most widely used methods for simulation-based testing. The algorithm involves drawing random samples from a stochastic model of the system's environment and observing whether a failure occurs. While this approach can generate failure events, when sampling is undirected, it can be very inefficient. The inefficiency is due to the size and complexity of the state space, rarity of failure events, and exponential explosion of searching over sequences of states. The combination of these factors makes serendipitously encountering a failure event extremely unlikely. Furthermore, while this method returns some path to a failure event, it does not necessarily give the most likely path.

Adaptive stress testing (AST) is an accelerated search algorithm for finding the most likely path to a failure event [15]. AST formulates stress testing as a sequential decision process and then optimizes it using a reinforcement learning algorithm. The approach uses learning to guide the search, which contrasts with traditional Monte Carlo approaches that are undirected. The key idea is to define an appropriate reward function to give evaluative feedback to the reinforcement learner so that it can explore and learn through interactions with the simulator and automatically discover important parts of the state space. The result is an accelerated search algorithm that is focused, scalable, and adaptive. We present formulations for stress testing both fully observable systems, where the state is available to the reinforcement learner, and partially observable systems, where part or all of the state is not available.

We apply AST to stress test a prototype aircraft collision avoidance system in simulated encounters and show how AST can be used to find and analyze likely encounters of near mid-air collision (NMAC). This chapter is an extended and revised version of a conference paper [15]. The remaining sections are organized as follows. Section 5.2 reviews the related literature on stress testing approaches. Section 5.3 reviews definitions and provides an overview of sequential decision processes and Monte Carlo tree search (MCTS). Section 5.4 presents the AST framework and an implementation of AST suitable for partially observable systems. Finally, Sect. 5.5 describes an application of AST to aircraft collision avoidance.

5.2 Related Work

Formal Methods Formal methods construct mathematical models of the system and rigorously prove whether a safety property holds. These methods can often provide counterexamples when a property does not hold. Probabilistic model checking (PMC) is a formal method that verifies properties over Markov models

[4, 17]. Markov models with discrete state in discrete time are often used in practice. PMC provides complete evaluations of the properties and their probabilities of occurrence over all states and paths. Automatic theorem proving (ATP) uses computer algorithms to automatically generate mathematical proofs. Systems and assumptions are modeled using mathematical logic, and then ATP is applied to prove whether a property holds. If a proof is generated successfully, one can conclude with certainty that the property holds over the entire model. However, if the algorithm fails to generate a proof, then it is uncertain whether the property holds. Hybrid systems theorem proving (HSTP) extends ATP to differential dynamic logic, which is a real-valued, first-order dynamic logic for hybrid systems [7, 13]. A hybrid system model can capture both continuous and discrete dynamic behavior. The continuous behavior is described by a differential equation and the discrete behavior is described by a state machine or automaton. PMC and HSTP both evaluate properties over the entire state space and thus give guarantees of completeness. However, they have difficulty scaling to the size and complexity of large autonomous systems. Consequently, applications tend to test only a subset of the system and use simple dynamics models. Furthermore, constructing formal method models can be very challenging. For example, the complexities of autonomous systems and their operating environments may not be easily modeled as a hybrid system.

Simulation-Based Testing Simulation-based methods use a simulation model and evaluate the system at a finite number of simulation paths. Simulation models are generally easy to create, because there are very few constraints on their form. While simulation-based methods cannot provide exhaustive coverage of the space as formal methods do, they are often the most practical approach for finding realistic examples of failure events. Simulation-based methods vary in how simulation paths are chosen. The paths can be manually crafted by a domain expert or they can sweep over a low-dimensional parametric model [2]. An alternative approach is to run simulations drawn from a stochastic model of the system's operating environment [5, 9]. However, simple Monte Carlo sampling can be very inefficient given the very large search space, rarity of failure events, and undirected nature of the search. Our approach also samples from a stochastic model of the system's environment. However, instead of using undirected Monte Carlo sampling, our approach formulates stress testing as a sequential decision process and explicitly searches for failure events using reinforcement learning.

5.3 Background

5.3.1 Definitions

The current state of the simulator is denoted s and the next state is denoted s'. The action is denoted a. In cases where the time index is relevant, we use subscript to indicate the time, e.g., the initial state is s_0, the state at time t is s_t, and the action

at time t is a_t. We assume that the simulator is *episodic* in that it terminates in a finite number of steps. The *terminal time* T is the first time the simulation enters a terminal state. We assume that terminal states are absorbing so that the simulator remains in the terminal state for all subsequent time steps but does not collect any additional reward. Consequently, $t \geq T$ indicates that the simulation has terminated and $t < T$ indicates that it has not terminated. The simulator terminates either at an event or when a maximum time is reached. A simulation *path* is a sequence of simulation states from initial to terminal state resulting from a forward simulation. The *reward* is the result of evaluating the reward function on a single transition of the simulator. The *total reward* is the sum of all the rewards collected over a path. An *event* E is the subset of state space where the event of interest occurs. While this paper focuses on failure events, an event can be arbitrarily defined. We use the notation $s \in E$ to indicate that an event has occurred. Alternatively, we may use the Boolean variable e to indicate whether an event has occurred. A *pseudorandom seed*, or just *seed*, is a vector of numbers used to initialize a pseudorandom number generator. We assume that all random processes in the simulator are derived from the pseudorandom number generator and seed, so that the result of sampling from these processes is deterministic given the seed.

5.3.2 Sequential Decision Process

A sequential decision process is a mathematical framework for modeling situations where an agent makes a sequence of decisions in an environment to maximize a reward function [8]. If the environment is known and its state is fully observable, then the problem can be formulated as a Markov decision process (MDP). In an MDP, the agent chooses an action a at time t based on observing the current state s. The system evolves probabilistically to the next state s' according to a transition function based on the current state s and action a. The agent then receives reward r for the transition. The assumption that the transition function depends only on the current state and action is known as the *Markov property*. In cases where the underlying process is Markov, but the agent only observes part of the state, the problem can be represented as a partially observable Markov decision process (POMDP) [8]. The model is identical to that of an MDP, except that the agent does not observe s exactly when choosing a. This paper considers models that have a finite number of steps.

Reinforcement learning algorithms can be used to optimize sequential decision problems where the transition function is not fully known. Reinforcement learning algorithms do not require full knowledge of the transition function—only a means to draw samples from it. The algorithms learn through interaction with the environment [8, 16]. The learners adapt sampling during the search, enabling them to efficiently search large and complex state spaces. Reinforcement learners incrementally estimate the *state–action value function* $Q(s, a)$, which is the expected sum of rewards resulting from choosing action a in state s and following an optimal policy

thereafter. The *optimal action* a^* is the action that maximizes the state–action value function at state s, i.e., $a^* = \text{argmax}_a Q(s, a)$. The *optimal policy* $\pi^*(s)$ is the function that gives the optimal action a^* for each state s.

5.3.3 Monte Carlo Tree Search

Monte Carlo tree search (MCTS) is a state-of-the-art heuristic search algorithm for optimizing sequential decision processes [1, 12]. MCTS incrementally builds a search tree using a combination of directed sampling based on estimates of the state–action value function and undirected sampling based on a fixed distribution, called *rollouts*. During the search, sampled paths from the simulator are used to incrementally estimate $Q(s, a)$ and the optimal policy. To account for the uncertainty in the estimates, which may lead to premature convergence, MCTS encourages exploration by adding an exploration term to the state–action value function to encourage choosing paths that have not been explored as often. The exploration term optimally balances the selection of the best action estimated so far with the need for exploration to improve the quality of current value estimates. By doing so, MCTS adaptively focuses the search towards more promising areas of the search space. The effect of the exploration term diminishes as the number of times the state is visited increases.

This paper uses a variant of MCTS called Monte Carlo tree search with double progressive widening (MCTS-DPW), which extends MCTS to large or continuous state and action spaces [3]. When the state and action spaces are large (or infinite), visited states and actions are not revisited sufficiently through sampling alone, which hinders the quality of value estimates. The benefit of MCTS-DPW's progressive widening is that it forces revisits to existing states and slowly allows new states to be added as the total number of visits increases. Progressive widening stabilizes value estimates and prevents explosion of the branching factor of the tree.

5.4 Adaptive Stress Testing

Adaptive stress testing (AST) involves finding the most likely path to a failure event [15]. The idea is to formulate the problem as a sequential decision process and then use reinforcement learning to optimize it. Figure 5.1 shows a conceptual overview of the AST framework. The system under test is placed in a simulation where it interacts with a simulated environment. A reinforcement learner interacts with the simulator over multiple time steps to maximize the reward it receives. To do so, the learner manipulates the stochastic elements of the environment to trigger a failure event and maximize its path probability. In other words, the reinforcement learner is searching for a sequence of environment variables that is most *adversarial* to the system under test. The output of the optimization is the most likely sequence of states that results in a failure event.

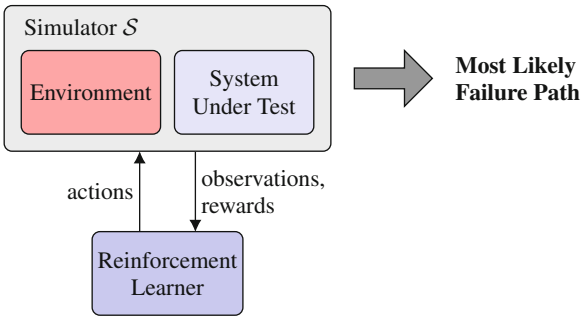


Fig. 5.1 Adaptive stress testing. A reinforcement learner interacts with a simulator and chooses environment variables to maximize the rewards received. Maximization of the reward function leads to finding the most likely path to a failure event

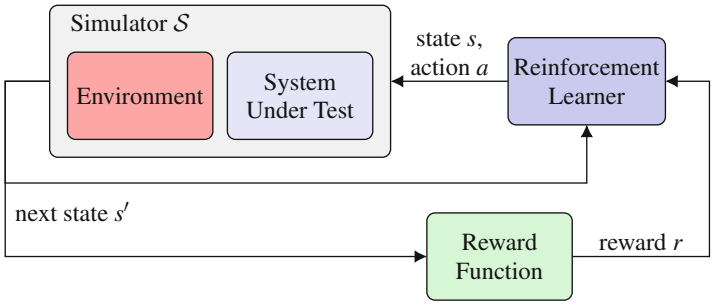


Fig. 5.2 Adaptive stress testing of a fully observable system

5.4.1 Full Observability

When the state of the system is fully observable, we can formulate the problem as a Markov decision process (MDP). The state s of the decision process is the state of the simulator, which includes both the state of the environment and the state of the system under test. The action a of the decision process directly controls the values of the stochastic elements in the environment. The actions are chosen to maximize a reward function that favors finding failure events and maximizing the path probability that leads to it. The state transition function is given by the simulator, where the next state can be sampled given the current state and action. The decision process has a finite time horizon.

Figure 5.2 illustrates the AST framework for the case where the state is fully observable by the reinforcement learner. The system under test is placed in a simulator where it interacts with an environment that can be controlled by the input actions a. The actions control the stochastic variables of the environment. The simulator samples a transition and returns the state at the next time step. The reward function is evaluated on the state to provide a reward to the reinforcement learner. The learner chooses actions to maximize the total reward received given all previous interactions with the simulator.

The reward function is designed to find failure events as the primary objective and to maximize the path probability as a secondary objective. The design corresponds to the stress testing objective of finding the most likely path to a failure event. There are three components of the reward function.

$$R(s, a, s') = \begin{cases} R_E & \text{if } s \in E \\ -d & \text{if } s \notin E,\ t \geq T \\ \log(P(s' \mid s, a)P(a \mid s)) & \text{if } s \notin E,\ t < T \end{cases} \tag{5.1}$$

The first term of Eq. (5.1) assigns a nonnegative reward R_E if the path terminates and a failure event occurs. If the path terminates and an event does not occur, the second term of Eq. (5.1) penalizes the learner by assigning the negative of the miss distance d to the learner. The miss distance is some measure defined by the user that is indicative of how close the simulation came to a failure. If such a measure is not available, then $-d$ can be set to a large negative constant. However, providing an appropriate miss distance can greatly accelerate the search by giving the learner the ability to distinguish the desirability of two paths that do not contain failure events. The third term of Eq. (5.1) maximizes the overall path probability by awarding the log probability of each transition. The transition probability is given by the product of the probability of the transition dynamics $P(s' \mid s, a)$ and the probability of choosing a under the model $P(a \mid s)$. The probability of choosing action a can, in general, depend on the current state s. Recall that reinforcement learning maximizes the expected sum of rewards. By choosing a reward of the log probability at each step, the reinforcement learning algorithm maximizes the sum of the log probabilities, which is equivalent to maximizing the product of the probabilities.

Our definitions of state, action, transition function, and reward function conform to an MDP that can be optimized using standard reinforcement learning algorithms [8, 16]. Existing reinforcement learning algorithms such as Monte Carlo tree search (MCTS) [12] and Q-Learning [18] can be applied to optimize the decision process and find the optimal path.

5.4.2 Partial Observability

The MDP formulation discussed previously requires the state of the simulator s. However, in many applications, some or all of the state variables may not be accessible. For example, black box components in the simulator, such as software libraries and binaries, can maintain state without exposing it externally. Incomplete state information leads to different states being aliased to the same observation, which can confuse the learner, hinder learning, and lead to poor optimization performance.

We introduce an abstraction that relaxes the need for the simulator to expose its state. Instead of explicitly representing and passing the state into and out of

the simulator, we assume that the simulator maintains state internally and the state is updated in-place. Furthermore, instead of passing in the stochastic variables of the environment as the reinforcement learning actions, we use a pseudorandom seed $\bar{a}$ as a proxy. The simulator uses $\bar{a}$ to seed an internal random process that draws a sample of the environment variables a and a transition simultaneously. We assume that setting the seed makes the sampling process deterministic and sampling uniformly over all pseudorandom seeds returns the natural distribution of the stochastic models in the simulator.

Seed–Action Simulator A *seed–action simulator* $\bar{\mathcal{S}}$ is a stateful simulator that uses a pseudorandom seed input to update an internal state in-place. The state s is not exposed externally, making the simulator appear non-Markovian to external processes, such as the reinforcement learner. The simulator can draw a sample of the next state using it to replace the current state in-place. The simulator returns the transition probability p given by $P(s' \mid s, a)P(a \mid s)$; a Boolean indicating whether an event occurred e; and the miss distance d. While the state cannot be observed or set, we assume that the simulator transitions are deterministic given the pseudorandom seed $\bar{a}$. This assumption allows a previously visited state to be revisited by replaying the sequence of pseudorandom seeds $\bar{a}_0, ..., \bar{a}_t$ that leads to it starting from the initial state. In other words, we use the sequence of pseudorandom seeds as the state. We use s to denote the true internal state of the simulator and $\bar{s}$ to denote the sequence of seeds that induces s.

The seed–action simulator $\bar{\mathcal{S}}$ exposes the following simulation control functions:

- INITIALIZE($\bar{\mathcal{S}}$) resets the simulator $\bar{\mathcal{S}}$ to initial state s_0. The simulation state is modified in-place.
- STEP($\bar{\mathcal{S}}, \bar{a}$) advances the state of the simulator by pseudorandom sampling. First, the pseudorandom seed $\bar{a}$ is used to set the state of the simulator's pseudorandom process. Second, an action a is sampled according to the environment's stochastic model. The action a can generally depend on the current state s. Third, the next state s' is sampled given the current state s and action a and the state of the simulator is updated. The simulator returns the transition probability p given by $P(s' \mid s, a)P(a \mid s)$; whether an event occurred e; and the miss distance d.
- ISTERMINAL($\bar{\mathcal{S}}$) returns true if the current state of the simulator is terminal and false otherwise.

Figure 5.3 illustrates the AST framework under the pseudorandom seed abstraction, where the simulator has been replaced by a seed–action simulator. The seed–action simulator $\bar{\mathcal{S}}$ maintains its own state, but does not output it. The seed input is used to draw a random action and state transition and the state is updated in-place. The simulator outputs the transition probability p, a Boolean indicating whether an event occurred e, and the miss distance d. The reward function translates the simulator outputs into a reward. The optimizer learns from the reward to optimize over pseudorandom seeds.

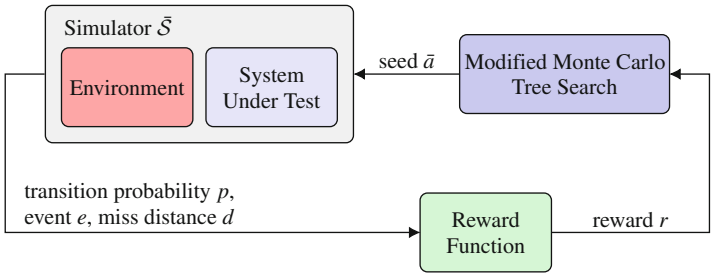


Fig. 5.3 Adaptive stress testing of a partially observable system

In addition to handling partial observability, this abstraction also provides a practical convenience. Simulators are often written in a distributed and modular fashion where each component maintains its own state. The simulator may consist of many of these components. Since the simulator state is the concatenation of the states of all the individual components, explicitly assembling and handling the state can break modularity and be a major implementation inconvenience. This abstraction, which uses in-place state update and pseudorandom seeds as a proxy to the actions, enables the simulator to maintain modularity of the components.

Modified MCTS Algorithm We adapt the MCTS algorithm for optimizing seed–action simulators. We base the algorithm on the double progressive widening variant of MCTS because the actions of the reinforcement learner are now pseudorandom seeds, which are vast [3]. The transition behavior of the simulator is deterministic given a pseudorandom seed input. Consequently, only a single next state is possible and there is no need to limit the number of next states. We simplify the algorithm by removing the progressive widening of the state and its associated variables, k' and α'. The action space is the space of all pseudorandom seeds. Since there is no need to distinguish between seeds, we choose the rollout policy and the action expansion function of MCTS to uniformly sample over all seeds. Sampling seeds uniformly generates unbiased samples of the next state from the simulator. We use U_{seed} to denote the discrete uniform distribution over all possible seeds. The state s of the simulator is not available to the reinforcement learner. However, since the simulator is deterministic given the pseudorandom seed input $\bar{a}$, we can revisit a previous state by replaying the sequence of seeds that leads to it starting from the initial state. As a result, we use the sequence of actions $\bar{a}_0, \ldots, \bar{a}_t$ as the state $\bar{s}$ in the algorithm. The path with the highest total reward, that is, the sum of all the rewards received over the entire path, may, and likely will, be from a path that is encountered during a rollout. Since rollouts are not individually recorded, information about the best path can be lost. To ensure that the algorithm returns the best path seen over the entire execution of the search, we explicitly track the highest total reward seen, R^*, and the corresponding action sequence, $\bar{s}^*$.

The modified MCTS algorithm is shown in Algorithm 1. The algorithm consists of a main loop that repeatedly performs forward simulations of the system while building the search tree and updating the state–action value estimates. The search

tree $\mathcal{T}$ is initially empty. Each simulation runs from initial state to terminal state. The path is determined by the sequence of pseudorandom seeds chosen by the algorithm, which falls into three stages for each simulation:

Algorithm 1 MCTS for seed–action simulators

▷ Inputs: Seed–action simulator $\bar{\mathcal{S}}$
▷ Returns: Sequence of seeds that induces path with highest total reward $\bar{s}^*$
function MCTS($\bar{\mathcal{S}}$)
  global $\bar{s}_T \leftarrow \emptyset$
  $(\bar{s}^*, R^*) \leftarrow (\emptyset, -\infty)$
  loop
    $\bar{s} \leftarrow \emptyset$
    INITIALIZE($\bar{\mathcal{S}}$)
    $R_{total} \leftarrow$SIMULATE($\bar{\mathcal{S}}, \bar{s}$)
    if $R_{total} > R^*$ **then**
      $(\bar{s}^*, R^*) \leftarrow (\bar{s}_T, R_{total})$
  return $\bar{s}^*$
function SIMULATE($\bar{\mathcal{S}}, \bar{s}$)
  if ISTERMINAL($\bar{\mathcal{S}}$) **then**
    $\bar{s}_T \leftarrow \bar{s}$
    return 0
  if $\bar{s} \notin \mathcal{T}$ **then**
    $\mathcal{T} \leftarrow \mathcal{T} \cup \{\bar{s}\}$
    $(N(\bar{s}), A(\bar{s})) \leftarrow (0, \emptyset)$
    return ROLLOUT($\bar{\mathcal{S}}, \bar{s}$)
  $N(\bar{s}) \leftarrow N(\bar{s}) + 1$
  if $\|N(\bar{s}, \bar{a})\| < kN(\bar{s})^\alpha$ **then**
    $\bar{a} \sim U_{seed}$
    $(N(\bar{s}, \bar{a}), V(\bar{s}, \bar{a})) \leftarrow (0, \emptyset)$
    $Q(\bar{s}, \bar{a}) \leftarrow 0$
    $A(\bar{s}) \leftarrow A(\bar{s}) \cup \{\bar{a}\}$
  $\bar{a} \leftarrow \text{argmax}_a \, Q(\bar{s}, a) + c\sqrt{\frac{\log N(\bar{s})}{N(\bar{s}, a)}}$
  $(p, e, d) \leftarrow$STEP($\bar{\mathcal{S}}, \bar{a}$)
  $r \leftarrow$REWARD(p, e, d)
  $\bar{s}' \leftarrow [\bar{s}, \bar{a}]$
  $q \leftarrow r +$SIMULATE($\bar{\mathcal{S}}, \bar{s}'$)
  $N(\bar{s}, \bar{a}) \leftarrow N(\bar{s}, \bar{a}) + 1$
  $Q(\bar{s}, \bar{a}) \leftarrow Q(\bar{s}, \bar{a}) + \frac{q - Q(\bar{s}, \bar{a})}{N(\bar{s}, \bar{a})}$
  return q
function ROLLOUT($\bar{\mathcal{S}}, \bar{s}$)
  if ISTERMINAL($\bar{\mathcal{S}}$) **then**
    $\bar{s}_T \leftarrow \bar{s}$
    return 0
  $\bar{a} \sim U_{seed}$
  $(p, e, d) \leftarrow$STEP($\bar{\mathcal{S}}, \bar{a}$)
  $r \leftarrow$REWARD(p, e, d)
  $\bar{s}' \leftarrow [\bar{s}, \bar{a}]$
  return $r +$ROLLOUT($\bar{\mathcal{S}}, \bar{s}'$)

- *Search.* In the search stage, which is implemented by SIMULATE, the algorithm starts at the root of the tree and recursively selects a child to follow. At each visited state node, the progressive widening criteria determine whether to choose among existing actions (seeds) or to expand the number of children by sampling a new action. The criterion limits the number of actions at a state s to be no more than polynomial in the total number of visits to that state. Specifically, a new action $\bar{a}$ is sampled from a discrete uniform distribution over all seeds U_{seed} if $\|A(\bar{s})\| < kN(\bar{s})^{\alpha}$, where k and α are parameters, $\|A(\bar{s})\|$ is the number of previously visited actions at state $\bar{s}$, and $N(\bar{s})$ is the total number of visits to state $\bar{s}$. Otherwise, the existing action that maximizes

$$Q(\bar{s}, \bar{a}) + c\sqrt{\frac{\log N(\bar{s})}{N(\bar{s}, \bar{a})}} \tag{5.2}$$

  is chosen, where c is a parameter that controls the amount of exploration in the search, and $N(\bar{s}, \bar{a})$ is the total number of visits to action $\bar{a}$ in state $\bar{s}$. The second term in Eq. (5.2) is an *exploration bonus* that encourages selecting actions that have not been tried as frequently. The action is used to advance the simulator to the next state and the reward is evaluated. The search stage continues in this manner until the system transitions to a state that is not in the tree.
- *Expansion.* Once we have reached a state that is not in the tree $\mathcal{T}$, we create a new node for the state and add it. The set of previously taken actions from this state, $A(\bar{s})$, is initially empty and the number of visits to this state $N(\bar{s})$ is initialized to zero.
- *Rollout.* Starting from the state created in the expansion stage, we perform a *rollout* that repeatedly samples state transitions until the desired termination is reached. State transitions are drawn from the simulator with actions chosen according to a rollout policy, which we set as the discrete uniform distribution over all seeds U_{seed}.

At each step in the simulation, the reward function is evaluated and the reward is used to update estimates of the state–action values $Q(\bar{s}, \bar{a})$. The values are used to direct the search. At the end of each simulation, the best total reward R^* and best path $\bar{s}^*$ are updated. Simulations are run until the stopping criterion is met. The criterion is a fixed number of iterations for all our experiments except for the performance study where we used a fixed computational budget. The algorithm returns the path with the highest total return represented as a sequence of pseudorandom seeds.

Computational Complexity Each iteration of the MCTS main loop simulates a path from initial state to terminal state. As a result, the number of calls to the simulator is linear in the number of loop iterations. The computation time thus varies as $O(N_{loop} \cdot (T_{\text{INITIALIZE}} + N_{steps} \cdot T_{\text{STEP}}))$, where N_{loop} is the number of loop iterations, $T_{\text{INITIALIZE}}$ is the computation time of INITIALIZE, N_{steps} is the average number of steps in the simulation, and T_{STEP} is the computation time of the STEP function.

5.5 Aircraft Collision Avoidance Application

Aircraft collision avoidance systems are mandated on all large transport and cargo aircraft in the USA to help prevent mid-air collisions. Its operation has played a crucial role in the exceptional level of safety in the national airspace [14]. Airborne collision avoidance systems monitor the airspace around an aircraft and issue alerts to the pilot if a conflict with another aircraft is detected. These alerts, called resolution advisories (RAs), instruct the pilot to maneuver the aircraft to a certain target vertical velocity and maintain it. The advisories are typically issued when the aircraft are within approximately 20–40 s to a potential collision.

The Traffic Alert and Collision Avoidance System (TCAS) is currently deployed in the USA and many countries around the world and has been very successful at protecting aircraft from mid-air collisions. However, studies have revealed fundamental limitations in TCAS that prevent it from operating effectively in the next-generation airspace [11]. To address the growing needs of the national airspace, the Federal Aviation Administration (FAA) has decided to create a new aircraft collision avoidance system. The next-generation Airborne Collision Avoidance System (ACAS X) is currently being developed and tested and promises a number of improvements over TCAS including a reduction in collision risk while simultaneously reducing the number of unnecessary alerts [11].

One of the primary safety metrics of airborne collision avoidance systems is the likelihood of near mid-air collision (NMAC), defined as two aircraft coming closer than 500 ft horizontally and 100 ft vertically. While studies have shown that NMACs are extremely rare, the risk of NMAC cannot be completely eliminated due to factors such as surveillance noise, pilot response delay, and the need for an acceptable alert rate [11]. Understanding the nature of the residual NMAC risk is important for certification and to inform the development of future iterations of the system.

There are several versions of ACAS X under development. In this paper, we refer to ACAS Xa version 0.8.5, which is designed to be a direct replacement to TCAS. Table 5.1 lists the possible primary RAs of ACAS X. We use $\dot{z}_{own}$ to denote the current vertical velocity of the aircraft.

Table 5.1 Primary ACAS X advisories

Abbreviation	Description	Rate to maintain (ft/min)
COC	Clear of conflict	N/A
DND	Do not descend	0
DNC	Do not climb	0
MAINTAIN	Maintain current rate	$\dot{z}_{own}$
DS1500	Descend at 1500 ft/min	-1500
CL1500	Climb at 1500 ft/min	$+1500$
DS2500	Descend at 2500 ft/min	-2500
CL2500	Climb at 2500 ft/min	$+2500$

The COC advisory stands for "clear of conflict" and is equivalent to no advisory. The pilot is free to choose how to control the aircraft. The DND and DNC advisories stand for "do not descend" and "do not climb," respectively. They restrict the pilot from flying in a certain direction. The MAINTAIN advisory is preventative and instructs the pilot to maintain the current vertical rate of the aircraft. The advisories DS1500 and CL1500 instruct the pilot to descend or climb at 1500 ft/min. The pilot is expected to maneuver the aircraft at $\frac{1}{4}g$ acceleration until the target vertical rate is reached and then maintain that vertical rate. The DS2500 and CL2500 advisories instruct the pilot to descend or climb at an increased rate of 2500 ft/min. These advisories are strengthened advisories and a stronger response from the pilot is expected. For these strengthened RAs, the pilot is expected to maneuver at $\frac{1}{3}g$ acceleration until the target vertical rate is reached and then maintain that vertical rate. Strengthened RAs must follow a weaker RA of the same vertical direction. For example, a CL1500 advisory must precede a CL2500 advisory. Advisories issued by collision avoidance systems on different aircraft are not completely independent. When an RA is issued, coordination messages are sent to nearby aircraft to prevent other collision avoidance systems from accidentally issuing an RA in the same vertical direction.

5.5.1 Experimental Setup

Figure 5.4 shows AST applied to aircraft collision avoidance. The MCTS algorithm interacts with a seed–action simulator that models various components of a mid-air encounter. The simulator models the initial distributions of the aircraft, sensors, collision avoidance system, pilot response, and aircraft dynamics. A reward function provides feedback to the reinforcement learner, informing it of the desirability of the current state.

Initial State The initial state of the encounter includes initial positions, velocities, and headings of all the aircraft. The initial state is drawn from a distribution that gives realistic initial configurations of aircraft that are likely to lead to NMAC. In our experiments, the encounters are initialized using the Lincoln Laboratory Correlated Aircraft Encounter Model (LLCEM) [10]. LLCEM is a statistical model learned from a large body of radar data of the entire national airspace.

Sensor Model The sensor model captures how the collision avoidance system perceives the world. We assume beacon-based radar with no noise. The aircraft measures their own vertical rate, barometric altitude, heading, and altitude above ground. For each intruding aircraft, the sensor measures slant range (relative distance to intruder), bearing (relative horizontal angle to intruder), and relative altitude.

Collision Avoidance System The collision avoidance system is the system under test. We use a prototype of ACAS X in the form of a binary library obtained from

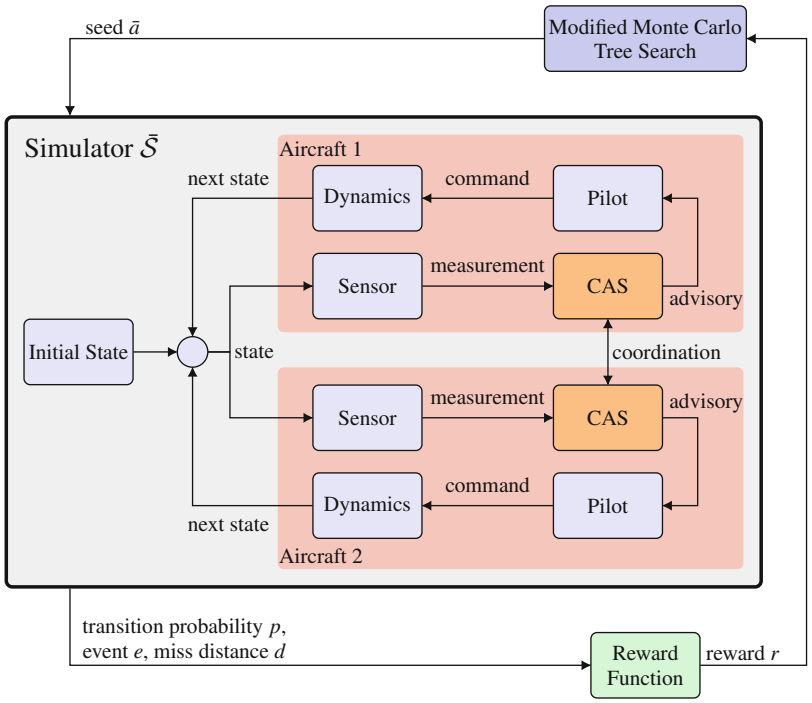


Fig. 5.4 System diagram for pairwise encounters

the FAA. The binary has a minimal interface that allows initializing and stepping the state forward in time. The collision avoidance system maintains internal state, but does not expose it. The primary output of the ACAS X system is the RA. ACAS X has a coordination mechanism to ensure that issued RAs from different aircraft are compatible with one another, i.e., that two aircraft are not instructed to maneuver in the same vertical direction. Coordination messages are communicated to all nearby aircraft.

Pilot Model The pilot model formalizes the pilot's intent and how the pilot responds to an RA. The pilot's intent is how the pilot would fly the aircraft if there were no RAs. To model intended commands, we use the pilot command model in LLCEM, which gives a realistic stochastic model of aircraft commands in the airspace [10]. The pilot response model defines how pilots respond to an issued RA. Pilots are assumed to respond deterministically to an RA with perfect compliance after a fixed delay [6]. Pilots respond to initial RAs with a 5 s delay and subsequent RAs (i.e., strengthenings and reversals) with a 3 s delay. During the initial delay period, the pilot continues to fly their intended trajectory. During response delays from subsequent RAs, the pilot continues responding to the previous RA. Multiple RAs issued successively are queued so that both their order and timing are preserved. In the case where a subsequent RA is issued within 2 s

or less of an initial RA, the timing of the subsequent RA is used and the initial RA is skipped. The pilot command includes a specified airspeed acceleration, vertical rate, and turn rate.

Aircraft Dynamics Model The aircraft dynamics model determines how the state of the aircraft propagates with time. The aircraft state includes the airspeed, position north, position east, altitude, roll angle, pitch angle, and heading angle. The aircraft state is propagated forward at 1 Hz using forward Euler integration.

We use the reward function defined in Eq. (5.1). The event E is defined to be an NMAC, which occurs when two aircraft are closer than 100 ft vertically and 500 ft horizontally. We define the miss distance d to be the distance of closest approach, which is the Euclidean distance of the aircraft at their closest point in the encounter. The distance at the point of closest approach is an appropriate metric, because it is monotonically decreasing as trajectories get closer and reach a minimum at an NMAC. Because the models in the simulator use continuous distributions, we use the transition probability density instead of the transition probability in the reward function. We simulate encounters to a maximum of 50 time steps. The MCTS parameters in Algorithm 1 are set as follows: exploration constant $c = 100$; $k = 0.5$; $\alpha = 0.85$; and number of loop iterations $N_{loop} = 2000$.

5.5.2 Results

We apply AST to analyze NMACs in encounters involving two aircraft. We searched 100 encounters initialized using samples from LLCEM. Of the 100 encounters searched, 18 encounters contained an NMAC, yielding an empirical find rate of 18%. When the optimization algorithm completes, it returns the path with the highest reward regardless of whether the path contains an NMAC. When the returned path does not contain an NMAC, it is uncertain whether an NMAC exists and the algorithm was unable to find it, or whether no NMAC is reachable given the initial state of the encounter. We manually cluster the NMAC encounters and present our findings. The results of our study are reported to the ACAS X development team for further study and to inform development.

Crossing Time We observed a number of NMACs resulting from vertical maneuvers that occur at specific times. In particular, aircraft crossing in altitude during the delay period of an initial RA can lead to an NMAC. Figure 5.5 shows one such encounter that eventually ends in an NMAC at 36 s into the encounter. The probability density of the encounter evaluated under LLCEM is $5.3 \cdot 10^{-18}$. This measure can be used as a relative measure of likelihood of occurrence. In this encounter, the aircraft cross in altitude during pilot 1's delay period. The crossing leads to aircraft 1 starting the climb from below aircraft 2. The subsequent reversal later in the encounter is unable to resolve the conflict due to the pilot response delay.

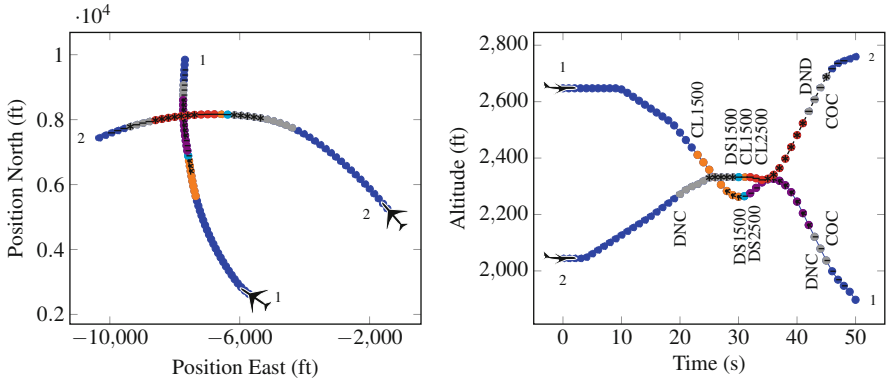


Fig. 5.5 An NMAC encounter where the aircraft vertically cross during pilot delay

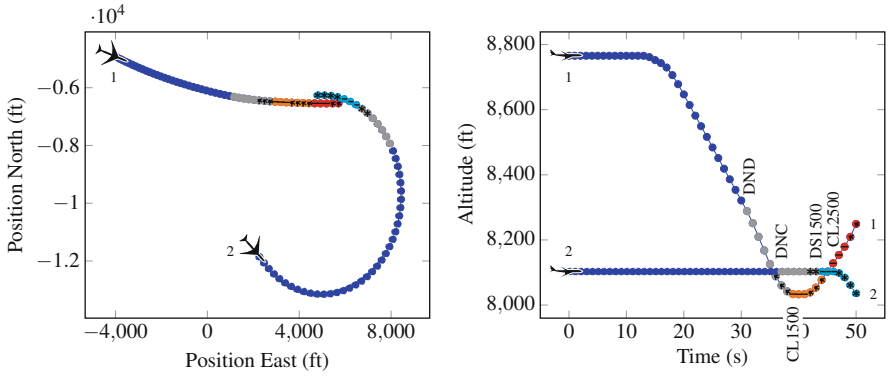


Fig. 5.6 An NMAC encounter where one aircraft is turning at high rate

High Turn Rates Turns, especially those at higher rates, tend to complicate the conflict resolution process by quickly shortening the time to closest approach. ACAS X does not have full state information about its intruder and must estimate it by tracking relative distance, relative angle, and the intruder altitude. Figure 5.6 shows an example of an encounter that has similar crossing behavior as Fig. 5.5 but exacerbated by the high turn rate of aircraft 2 (approximately 1.5 times the standard turn rate). In this scenario, the aircraft become almost head-on at the time of closest approach and a reversal is not attempted. An NMAC with a probability density of $6.5 \cdot 10^{-17}$ occurs at 48 s into the encounter.

Maneuvering Against RA A number of NMAC encounters involve the pilot initially moving against the issued RA. During the initial pilot response delay, the pilot is flying the intended trajectory. If the pilot maneuvers against the RA during this time, an NMAC may occur despite complying with the RA later on. In most cases, the disagreement must be severe to result in an NMAC, where the

pilot aggressively accelerates against the RA. While it is a difficult case for the collision avoidance system to resolve, operationally, it is extremely rare for a pilot to maneuver so aggressively against an RA.

Sudden Aggressive Maneuvering Sudden maneuvers can lead to NMACs when they are sufficiently aggressive and occurring at just the right time. In particular, we observed some encounters where two aircraft are approaching one another separated in altitude and flying level, then one aircraft suddenly accelerates vertically towards the other aircraft as they are about to pass. Under these circumstances, given the pilot response delays and dynamic limits of the aircraft, there is insufficient time and distance remaining for the collision avoidance system to resolve the conflict. Operationally, the chances of such maneuvers occurring are extremely low and, in fact, they are slightly exaggerated by our model. ACAS X issues traffic alerts (TAs) to alert pilots to nearby traffic, making them aware of intruding aircraft well before the initial RA. These advance warnings increase the pilot's situational awareness and reduce blunders like these. Our simulator does not model the effect of TAs.

One course of action would be for ACAS X to intervene preemptively. While this reduces the risk of possible sudden behavior, it also increases the overall alert rate of the system. Ultimately, the designer must assess the probabilities of various scenarios and find the delicate balance between risk of collision and issuing too many advisories.

Combined Factors In our experiments, it is rare for an NMAC to be attributable to a single cause. More commonly, a combination of factors contribute to the NMAC. Figure 5.5 shows an example of such an encounter. Although crossing time played a crucial role, there are a number of other factors that are important. The horizontal behavior where they are turning into each other is significant as it reduces the time to closest approach. The two vertical maneuvers of aircraft 1 before receiving an RA are also important. Similar observations can be made on many of the NMAC encounters found.

5.5.3 Performance Comparison

We compare the performance of MCTS against simple Monte Carlo sampling given a fixed computational budget. The algorithms are given a fixed amount of wall clock time and the best path found at the end of that time is returned. We compare the wall clock time of the algorithms rather than number of samples to account for the additional computation performed by the MCTS algorithm. Figure 5.7 shows the performance of the two algorithms as computation time varies. Figure 5.7a compares the total reward of the best encounters found by the algorithms. Each data point shows the mean and standard error of the mean of 100 pairwise encounters. Figure 5.7b shows the NMAC find rate of NMACs out of the 100 encounters searched. In both cases, MCTS clearly performs better than the baseline Monte Carlo search. The effectiveness of MCTS in finding NMACs

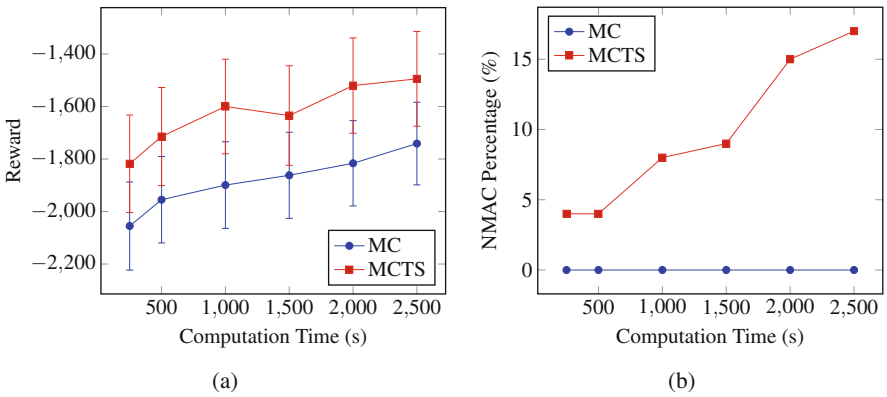


Fig. 5.7 Performance of MCTS and Monte Carlo with computation time

is particularly important and we see that MCTS greatly outperforms the baseline in this regard. As the computational budget increases, MCTS is able to find more NMACs, whereas with these computational budgets, Monte Carlo is unable to find any.

5.6 Conclusion

This paper presented adaptive stress testing (AST), a reinforcement learning-based stress testing approach for finding the most likely path to a failure event. We described AST formulations for the case where the state of the simulator is fully observable and also for the case where the state is partially or not available. For the latter case, we presented a modified MCTS algorithm that uses the pseudorandom seed of the simulator to overcome partial observability. We applied AST to stress test a prototype of the next-generation Airborne Collision Avoidance System (ACAS X) in an aircraft encounter simulator and successfully found a number of interesting examples of near mid-air collisions. Our implementation of AST is available as an open source Julia package at https://github.com/sisl/AdaptiveStressTesting.jl.

Acknowledgements We thank Neal Suchy at the Federal Aviation Administration (FAA); Michael Owen, Robert Klaus, and Cindy McLain at MIT Lincoln Laboratory; Joshua Silbermann, Anshu Saksena, Ryan Gardner, and Rachel Szczesiul at Johns Hopkins Applied Physics Laboratory; and others in the ACAS X team. We thank Guillaume Brat at NASA and Corina Pasareanu at Carnegie Mellon University for their invaluable feedback. This work was supported by the Safe and Autonomous Systems Operations (SASO) Project under NASA Aeronautics Research Mission Directorate (ARMD) Airspace Operations and Safety Program (AOSP).

References

1. C.B. Browne, E. Powley, D. Whitehouse, S.M. Lucas, P.I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, S. Colton, A survey of Monte Carlo tree search methods. IEEE Trans. Comput. Intell. AI Games **4**(1), 1–43 (2012)
2. B.J. Chludzinski, Evaluation of TCAS II version 7.1 using the FAA fast-time encounter generator model. Project Report ATC-346, Massachusetts Institute of Technology, Lincoln Laboratory (2009)
3. A. Couëtoux, J.B. Hoock, N. Sokolovska, O. Teytaud, N. Bonnard, Continuous upper confidence trees, in *Learning and Intelligent Optimization (LION)* (2011), pp 433–445
4. R.W. Gardner, D. Genin, R. McDowell, C. Rouff, A. Saksena, A. Schmidt, Probabilistic model checking of the next-generation airborne collision avoidance system, in *Digital Avionics Systems Conference (DASC)* (2016)
5. J.E. Holland, M.J. Kochenderfer, W.A. Olson, Optimizing the next generation collision avoidance system for safe, suitable, and acceptable operational performance. Air Traffic Control Q. **21**(3), 275–297 (2013)
6. International Civil Aviation Organization, Surveillance, radar and collision avoidance, in International Standards and Recommended Practices, vol IV, annex 10, 4th edn (2007)
7. J.B. Jeannin, K. Ghorbal, Y. Kouskoulas, R. Gardner, A. Schmidt, E. Zawadzki, A. Platzer, A formally verified hybrid system for the next-generation airborne collision avoidance system, in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)* (2015)
8. M.J. Kochenderfer, Decision making under uncertainty: theory and application. MIT Press (2015)
9. M.J. Kochenderfer, J.P. Chryssanthacopoulos, A decision-theoretic approach to developing robust collision avoidance logic, in *IEEE International Conference on Intelligent Transportation Systems (ITSC)* (2010), pp 1837–1842
10. M.J. Kochenderfer, L.P. Espindle, J.K. Kuchar, J.D. Griffith, Correlated encounter model for cooperative aircraft in the national airspace system. Project Report ATC-344, Massachusetts Institute of Technology, Lincoln Laboratory (2008)
11. M.J. Kochenderfer, J.E. Holland, J.P. Chryssanthacopoulos, Next-generation airborne collision avoidance system. Lincoln Lab. J. **19**(1), 17–33 (2012)
12. L. Kocsis, C. Szepesvári, Bandit based Monte-Carlo planning, in *European Conference on Machine Learning (ECML)* (2006), pp 282–293
13. Y. Kouskoulas, D. Genin, A. Schmidt, J. Jeannin, Formally verified safe vertical maneuvers for non-deterministic, accelerating aircraft dynamics, in *8th International Conference on Interactive Theorem Proving* (2017), pp 336–353
14. J.K. Kuchar, A.C. Drumm, The traffic alert and collision avoidance system. Lincoln Lab. J. **16**(2), 277–296 (2007)
15. R. Lee, M.J. Kochenderfer, O.J. Mengshoel, G.P. Brat, M.P. Owen, Adaptive stress testing of airborne collision avoidance systems, in *Digital Avionics Systems Conference (DASC)* (2015)
16. R.S. Sutton, A.G. Barto, *Reinforcement Learning: An Introduction* (MIT Press, Cambridge, 1998)
17. C. von Essen, D. Giannakopoulou, Probabilistic verification and synthesis of the next generation airborne collision avoidance system. Int. J. Softw. Tools Technol. Transfer **18**(2), 227–243 (2016)
18. C.J.C.H. Watkins, P. Dayan, Technical note: Q-learning. Mach. Learn. **8**, 279–292 (1992)

Chapter 6
Provably-Correct Compositional Synthesis of Vehicle Safety Systems

Petter Nilsson and Necmiye Ozay

6.1 Introduction

Modern passenger vehicles are prime examples of everyday complex engineered systems—a single car can contain hundreds of electronic control units that work together to provide safety and comfort to passengers and drivers. The trend toward increasing autonomy further adds to this complexity, and also increases the safety-criticality of the system. Unintended acceleration incidents and software-related recalls in recent years signify the need for principled approaches that can handle the complexity in design, verification, and operation of these systems. Due to the high complexity, a monolithic design approach that takes all the interactions and specifications into account at the same time is not feasible. Therefore, compositional design methodologies with correctness guarantees are needed.

In this chapter, we propose a contract-based framework for compositional design of safety controllers. Assume-guarantee reasoning and contract-based design are proposed as principled means for analysis and synthesis of complex systems in a modular way [1–4]. The key idea in these approaches is to equip each subsystem with local contracts that capture assumptions on their interaction with the other subsystems, and guarantees on their desired behavior. The particular assume-guarantee formalism varies based on the type of specifications the system has to satisfy. For instance, the assumptions and guarantees could be given as automata [1], temporal logic formulas [3, 4], supply/demand rates [5], or invariant sets [6, 7].

P. Nilsson
California Institute of Technology, Pasadena, CA, USA
e-mail: pettni@caltech.edu

N. Ozay (✉)
University of Michigan, Ann Arbor, MI, USA
e-mail: necmiye@umich.edu

H. Yu et al. (eds.), *Safe, Autonomous and Intelligent Vehicles*,
Unmanned System Technologies, https://doi.org/10.1007/978-3-319-97301-2_6

Specifically, we encode safety contracts as polytopic robustly controlled-invariant sets [8, 9] and show how such contracts and controllers realizing them can be constructed.

The proposed framework is demonstrated on two advanced driver assistance systems: adaptive cruise control and lane keeping. These functions, which typically constitute the basis of autonomous driving functionality, have been studied individually using several different techniques [10–15]. Interactions between longitudinal and lateral dynamics render compositional design of the two functions nontrivial. Our design flow starts with dynamic models of the systems together with specifications, and proceeds with decomposition of the problem via contracts and synthesis of local controllers. The contracts are constructed in such a way that the composed system is guaranteed to function correctly. We validate the controllers in high-fidelity simulations, and also implement them on an actual vehicle in Mcity. We discuss several challenges and trade-offs in the design process and highlight some of the advantages of contracts in mitigating some of these challenges.

The rest of the chapter is structured as follows. Section 6.2 introduces the adaptive cruise control and lane-keeping problems. Invariant sets, composition, and contracts are discussed in Sect. 6.3, where we also state the problem we address. Section 6.4 presents theoretical underpinnings of compositional computation of invariant sets together with algorithms. Design results and experiments are presented in Sects. 6.5 and 6.6, respectively. Finally, the chapter is concluded in Sect. 6.7.

Notation For two sets A and B, their *Minkowski sum* is denoted by $A \oplus B := \{a + b : a \in A,\ b \in B\}$. The *Minkowski difference*, denoted $A \ominus B$, is the set $\{c : \{c\} \oplus B \subset A\}$. The infinity-norm ball around a with radius r will be denoted by $\mathcal{B}_\infty(a, r) := \{x : \|a - x\|_\infty \leq r\}$. For a mapping $f : \mathbb{R}^n \to \mathbb{R}^m$, the *image* of $A \subset \mathbb{R}^n$ is $f(A) := \{f(a) : a \in A\} \subset \mathbb{R}^m$, and the *pre-image* of $B \subset \mathbb{R}^m$ is $f^{-1}(B) := \{x \in \mathbb{R}^n : f(x) \in B\}$.

We will denote the k-simplex as $\Delta_k := \{\boldsymbol{\alpha} \in \mathbb{R}^k_+ : \sum_{i=1}^k \alpha_i = 1\}$. The convex hull of a set A can then be written as:

$$\text{conv}(A) := \bigcup_{k \geq 1} \left\{ \sum_{i=1}^{k} \alpha_i a_i,\ \boldsymbol{\alpha} \in \Delta_k,\ a_i \in A\ \forall\, i \right\}.$$

If the set A is finite, i.e., $A = \{a_1, \ldots, a_k\}$ for some $k \in \mathbb{N}$, the convex hull is $\text{conv}(A) = \left\{ \sum_{i=1}^k \alpha_i a_i \ :\ \boldsymbol{\alpha} \in \Delta_k \right\}$.

6.2 Autonomous Driving Functions

In this section, we describe two control systems implementing two primitive autonomous driving functions, namely adaptive cruise control and lane keeping. While the control software architecture of a fully autonomous car involves many more systems, we use these two systems to highlight the potential integration challenges and how to overcome them within a principled contract-based framework.

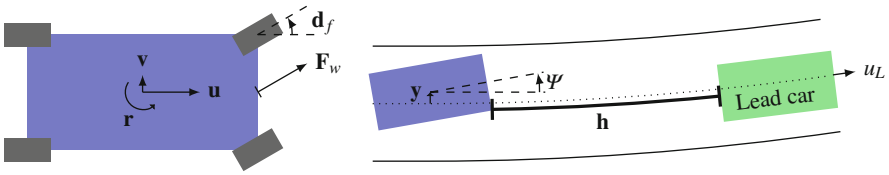


Fig. 6.1 Illustration of the states in the lateral and longitudinal models. The states **v**, **u**, and **r** are local to the car body frame (left), whereas the states **y**, Ψ, and **h** are quantities in the road frame (right). The dotted line is the center of the lane

6.2.1 Adaptive Cruise Control

Adaptive cruise control (ACC) is responsible for regulating the longitudinal dynamics of the vehicle. A linearized force-balance equation for the longitudinal dynamics is given by (see, e.g., [10] for details):

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\mathbf{u}\\ \mathbf{h}\end{bmatrix} = \begin{bmatrix}-f_1/m & 0\\ -1 & 0\end{bmatrix}\begin{bmatrix}\mathbf{u}\\ \mathbf{h}\end{bmatrix} + \begin{bmatrix}1/m\\ 0\end{bmatrix}\mathbf{F}_w + \begin{bmatrix}-f_0/m - \mathbf{vr}\\ u_L\end{bmatrix}, \tag{6.1}$$

where, as illustrated in Fig. 6.1, **u** is the forward speed of the ego vehicle, u_L is the speed of the lead vehicle, **h** is the headway (i.e., the distance to the lead vehicle), m is the vehicle mass, the parameters f_0 and f_1 are related to air drag and friction, and the parameters **v** and **r** are related to the lateral dynamics described in the next section.

The goal of ACC is to compute the applied longitudinal force $\mathbf{F}_w$ either so that a desired longitudinal speed $\mathbf{u} = u_{des}$ is achieved or so that the headway **h** stays above some nonnegative minimal value, whichever speed is lower.

6.2.2 Lane Keeping

The lane-keeping (LK) control system is responsible for regulating the lateral dynamics of the vehicle. A linearized model of the lateral dynamics is given by (see, e.g., [11] for details):

$$\frac{d}{dt}\begin{bmatrix}\mathbf{y}\\ \mathbf{v}\\ \Psi\\ \mathbf{r}\end{bmatrix} = \underbrace{\begin{bmatrix}0 & 1 & \mathbf{u} & 0\\ 0 & -\frac{C_{\alpha f}+C_{\alpha r}}{m\mathbf{u}} & 0 & \frac{bC_{\alpha r}-aC_{\alpha f}}{m\mathbf{u}} - \mathbf{u}\\ 0 & 0 & 0 & 1\\ 0 & \frac{bC_{\alpha r}-aC_{\alpha f}}{I_z\mathbf{u}} & 0 & -\frac{a^2C_{\alpha f}+b^2C_{\alpha r}}{I_z\mathbf{u}}\end{bmatrix}}_{A_{LK}(\mathbf{u})}\begin{bmatrix}\mathbf{y}\\ \mathbf{v}\\ \Psi\\ \mathbf{r}\end{bmatrix} + \begin{bmatrix}0\\ \frac{C_{\alpha f}}{m}\\ 0\\ a\frac{C_{\alpha f}}{I_z}\end{bmatrix}\mathbf{d}_f + \begin{bmatrix}0\\ 0\\ -1\\ 0\end{bmatrix}\mathbf{r}_d, \tag{6.2}$$

where the states $\mathbf{y}$, $\mathbf{v}$, Ψ, and $\mathbf{r}$ describe lateral displacement from the center of the lane, lateral velocity, yaw angle relative to the road, and yaw rate, respectively (again see Fig. 6.1). The input to the LK system is the steering angle $\mathbf{d}_f$, and there is also an exogenous disturbance $\mathbf{r}_d$ stemming from the curvature of the road. The parameters $C_{\alpha f}, C_{\alpha r}, I_z, a$, and b depend on the geometry and tire characteristics of the car.

The goal of LK is to control the wheel steering angle $\mathbf{d}_f$ in a way that prevents lane departure, that is, the LK controller should at all times keep the state $\mathbf{y}$ within some bound that depends on the widths of the lane and the car itself.

6.2.3 Challenges in Composition

Although the goals of the LK and ACC modules can be stated independently, the two problems are interdependent. As can be seen, the state $\mathbf{u}$ pertaining to the ACC system appears in the system matrix of (6.2). Conversely, the states $\mathbf{v}$ and $\mathbf{r}$ from the LK system appear in the offset term of the ACC system. Furthermore, both of these interdependencies are nonlinear.

Therefore, a design process that overlooks these interdependencies cannot guarantee that these two systems operate correctly when implemented simultaneously on the car. Motivated by this challenge, in what follows we develop a methodology for synthesizing provably correct local controllers for such interdependent subsystems. The key idea is to quantify the effect of each controlled subsystem onto its environment (i.e., other subsystems that it influences), and, conversely, take into account the effects of the environment onto the subsystem in consideration.

6.3 Composition of Invariant Sets Via Contracts

One could, in principle, try to synthesize ACC and LK controllers by combining the longitudinal and lateral dynamics given by (6.1) and (6.2) into a single system model, and search for a monolithic controller that imposes the ACC and LK objectives simultaneously. However, as the systems get more and more complex, such an approach fails to scale. Instead, a natural way to approach a large-scale problem is to decompose it into smaller subproblems, such that each subproblem is of lower dimensionality and thus easier to solve. Crucially, the decomposition must be done in a principled way in order for the sub-solutions to constitute a valid solution to the original problem. A popular framework for compositional analysis is *assume-guarantee reasoning*, where each subcomponent is endowed with a "contract" specifying its allowed behavior [16, 17]. In particular, an assume-guarantee contract for a subsystem contains assumptions on the behavior of other subsystems, and guarantees on the subsystem's own behavior as it relates to other subsystems. Also, the external system environment can be incorporated as a component with behavior restricted by an assumption.

In what follows, we focus on contracts related to safety of dynamical systems. We consider dynamical systems of the form:

$$\mathbf{x}(k+1) = f(\mathbf{x}(k), \mathbf{u}(k), \mathbf{d}(k)), \tag{6.3}$$

with state $\mathbf{x}$, control input $\mathbf{u} \in \mathcal{U}$, and disturbance input $\mathbf{d} \in \mathcal{D}$, where the set $\mathcal{U}$ is the set of admissible control inputs and the disturbance $\mathbf{d}$ is assumed to be restricted to a set $\mathcal{D}$. We model any uncontrolled effects on the system with the disturbance input $\mathbf{d}$, which can capture external inputs from the system environment, unknown/uncertain parameters of the system model, or, in a compositional setting, the effects of other subsystems on the subsystem under consideration. For instance, the states $\mathbf{v}$ and $\mathbf{r}$ of the lateral dynamics act as a disturbance in the ACC submodule when the dynamics in (6.1) are considered. For a system of the form (6.3), consider the control goal of keeping the state $\mathbf{x}$ within some safe set X. The safe set can, for instance, represent that the headway $\mathbf{h}$ is positive for ACC, or for LK that the lateral displacement $\mathbf{y}$ is less than some bound. This type of invariance specification can be encoded with an assume-guarantee contract:

$$\square\,(\mathbf{d} \in \mathcal{D}) \implies \square\,[(\mathbf{x} \in X) \wedge (\mathbf{u} \in \mathcal{U})]\,, \tag{6.4}$$

where the notation $\square$ is adopted from linear temporal logic and denotes the "always" operator, and the symbols $\wedge$ and $\implies$ are the logical "and" and "implication" operators. When the set $\mathcal{U}$ is clear from context, we simply write $\square\,(\mathbf{d} \in \mathcal{D}) \implies \square\,(\mathbf{x} \in X)$. The contract (6.4) essentially means that under the assumption that the disturbance input $\mathbf{d}$ remains in the set $\mathcal{D}$ at all times, the system state must remain in the safe set X at all times, and control inputs can only be selected from the admissible control input set $\mathcal{U}$. If there exist an initial state $\mathbf{x}(0) \in X$ and a sequence of (possibly state- and disturbance-dependent) control inputs $\mathbf{u} \in \mathcal{U}$ that guarantee that the state of the system remains in X for all times for all possible realizations of the disturbance sequence within the disturbance set $\mathcal{D}$, we say that the contract (6.4) is *realizable*. When the system is controlled by a controller that guarantees this property, we say that the closed-loop system *satisfies* the contract (6.4).

In order to study realizability of contracts of the abovementioned form, we use controlled-invariant sets. Given a dynamical system of the form (6.3), a set $\mathcal{U}$ of admissible inputs, and a set $\mathcal{D}$ of possible disturbance signals, a subset $\mathcal{C}$ of the state space is called *controlled invariant* if for all $x \in \mathcal{C}$, there exists a control input $u \in \mathcal{U}$ such that $f(x, u, \mathcal{D}) \subset \mathcal{C}$. It directly follows from the definitions that if $\mathcal{C}$ is a controlled-invariant set for a given system, the contract $\square\,(\mathbf{d} \in \mathcal{D}) \implies \square\,(\mathbf{x} \in \mathcal{C})$ is realizable. Moreover, the realizability of a contract of the form (6.4) is equivalent to the existence of a controlled-invariant set $\mathcal{C}$ inside X (i.e., $\mathcal{C} \subseteq X$). Therefore, we focus on computation of controlled-invariant sets inside a given safe set X, and computation of the corresponding invariance-enforcing controllers. There are several techniques to construct invariant sets and controllers that enforce invariance, such as sums-of-squares optimization [18, 19], finite abstractions [20], polytopic controlled-invariant set computations, and analytical solutions in cases

where it is possible (e.g., [14, 21]). In this chapter, we focus on polytopic controlled-invariant sets; as with much linear systems theory, this method is attractive by being accessible, fairly general, and by having a well-understood underlying theory.

The complexity of computing a controlled-invariant set $\mathcal{C}$ and the corresponding invariance-inducing control inputs scales with the state space dimension, making the computation prohibitively expensive for large-scale systems. System decomposition mitigates the scalability issue in computation of invariance contracts by restricting attention to invariant sets that are *separable*, i.e., that are products of sets in lower dimensions. However, in the case of interdependent subsystems and decomposition, there are more aspects to invariance contracts beyond scalability. First of all, depending on how the assume-guarantee contracts are synthesized, the compositional approach may be conservative compared to a monolithic solution as shown in the following example:

Example 1 Consider the following two-dimensional system:

$$\begin{bmatrix}\mathbf{x}_1(k+1)\\ \mathbf{x}_2(k+1)\end{bmatrix} = \begin{bmatrix}1 & 0.2\\ 0.2 & 1\end{bmatrix}\begin{bmatrix}\mathbf{x}_1(k)\\ \mathbf{x}_2(k)\end{bmatrix} + \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}\mathbf{u}_1(k)\\ \mathbf{u}_2(k)\end{bmatrix}, \quad -0.1 \le \mathbf{u}_1(k), \mathbf{u}_2(k) \le 0.1. \tag{6.5}$$

Below, separable and centralized computation of an invariant set contained in $[-1, 1]^2$ are compared.

The separable case is symmetric; consider subsystem 1 which is as follows:

$$\mathbf{x}_1(k+1) = \mathbf{x}_1(k) + \mathbf{u}_1(k) + \underbrace{0.2\mathbf{x}_2(k)}_{\text{disturbance}}, \quad -0.1 \le \mathbf{u}_1(k) \le 0.1. \tag{6.6}$$

Restricting attention to symmetric contracts, the objective is to find the largest number $\bar{x}$ such that $[-\bar{x}, \bar{x}]$ is controlled invariant for subsystem (6.6) under the assumption that $-\bar{x} \le \mathbf{x}_2 \le \bar{x}$. Elementary calculations reveal that the maximal such number is $\bar{x} = 0.5$, so the maximal symmetric separable controlled-invariant set is $\mathcal{C}_1 = [-0.5, 0.5]^2$ with volume 1.

In the centralized case, techniques described later in the chapter can be employed which results in a (non-separable) controlled-invariant set $\mathcal{C}_2$ with volume 3. The two invariant sets are displayed in Fig. 6.2. The example manifests that separability comes at the cost of a smaller invariant set. ■

Although conservatism is undesirable in itself, system modularity may in many cases be desirable. For instance, if the plant of one subsystem is modified, the local controller can be updated to satisfy the same contract without the need to redesign other components. Assume-guarantee contracts may also facilitate debugging: by monitoring satisfaction of contracts, it is easy to determine where in the system a fault first occurs.

In addition to assume-guarantee contracts, a second crucial aspect of a decomposition framework is assumptions on communication. If the current state of one subsystem is made available, other subsystems can use that information in

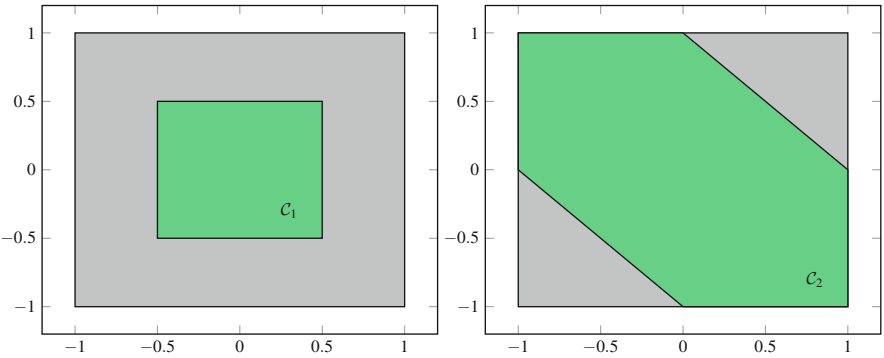


Fig. 6.2 Illustration of separable and centralized invariant sets. Invariant sets are plotted in green and are contained in the box $[-1, 1]^2$ (gray). The separable set (left) is $\mathcal{C}_1 = [-0.5, 0.5] \times [-0.5, 0.5]$; the interval $[-0.5, 0.5]$ is controlled invariant for (6.6). The larger centralized invariant set $\mathcal{C}_2$ (right) is controlled invariant for (6.5)

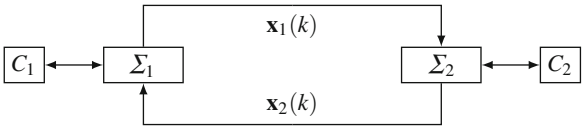


Fig. 6.3 Example of an assume-guarantee interconnection. If the controller C_1 for Σ_1 enforces $\square(\mathbf{x}_2 \in X_2) \implies \square(\mathbf{x}_1 \in X_1)$ and the controller C_2 for Σ_2 enforces $\square(\mathbf{x}_1 \in X_1) \implies \square(\mathbf{x}_2 \in X_2)$, the composition satisfies $\square(\mathbf{x}_1 \in X_1 \wedge \mathbf{x}_2 \in X_2)$ for appropriate initial conditions

their pursuit to satisfy their own contracts. Again, there is a trade-off between conservatism and modularity: transmission of state information has the potential to make the approach less conservative (compared to a monolithic controller), but comes with increased subsystem interdependence.

Example 2 Let Σ_1 and Σ_2 be two discrete-time systems with states $\mathbf{x}_1$ and $\mathbf{x}_2$ and state update functions $f_1(x_1, x_2, u_1)$ and $f_2(x_1, x_2, u_2)$ as illustrated in Fig. 6.3. Suppose that they are endowed with two assume-guarantee contracts:

$$\text{Contract for } \Sigma_1 : \ \square\,(\mathbf{x}_2 \in X_2) \Longrightarrow \square\,(\mathbf{x}_1 \in X_1)\,,$$
$$\text{Contract for } \Sigma_2 : \ \square\,(\mathbf{x}_1 \in X_1) \Longrightarrow \square\,(\mathbf{x}_2 \in X_2)\,.$$

Consider the task of subsystem Σ_1: it needs to keep its state $\mathbf{x}_1$ inside of X_1 under the assumption that $\mathbf{x}_2$ is kept in X_2. If the current state $\mathbf{x}_2(k)$ is available, a controller for Σ_1 must enforce invariance of a set $Y_1 \subset X_1$ with the property that for all $x_1 \in Y_1$:

$$\forall x_2 \in X_2 \ \exists u_1 \in \mathcal{U}_1 \ f_1(x_1, x_2, u_1) \in Y_1. \tag{6.7}$$

On the other hand, if the current state $\mathbf{x}_2(k)$ is unknown, the controller for Σ_1 must find an input that works for all possible $x_2 \in X_2$;

$$\exists u_1 \in \mathcal{U}_1 \; \forall x_2 \in X_2 \; f_1(x_1, x_2, u_1) \in Y_1. \tag{6.8}$$

The switch of quantifiers illustrates that (6.8) is a more difficult control problem: a u_1 satisfying (6.8) also satisfies (6.7), but the opposite is in general not true. ■

A system decomposition is essentially a partition of the system states into subsystems, where each subsystem is responsible for controlling its own states. Given a system decomposition and an overall specification, the distributed synthesis problem is to find a set of feasible assume-guarantee contracts along with local controllers that enforce the contracts under appropriate information transmission assumptions.

6.3.1 Contract Realizability Problem

In this chapter, we are interested in the following problem. Given a system of the form (6.3), a partition of the state $\mathbf{x}$ into N subsystem states (i.e., $\mathbf{x}^T = [\mathbf{x}_1^T, \ldots, \mathbf{x}_N^T]$), information constraints among the states dictating which subsystem controller has access to what other states and disturbances, and an assume-guarantee contract of the form:

$$\square\,(\mathbf{d} \in \mathcal{D}) \implies \square \bigwedge_{i=1}^{N} (\mathbf{x}_i \in X_i)\,, \tag{6.9}$$

find, for each $n = 1, \ldots, N$, local contracts of the form:

$$\square[(\mathbf{d} \in \mathcal{D}) \wedge \bigwedge_{m \neq n} (\mathbf{x}_m \in \mathcal{C}_m)] \implies \square\,(\mathbf{x}_n \in \mathcal{C}_n)\,, \tag{6.10}$$

with $\mathcal{C}_n \subset X_n$ that are locally realizable. In particular, we search for sets $\mathcal{C}_n$ that are controlled invariant. Provided that initial conditions are within these sets, i.e., $\mathbf{x}_n(0) \in \mathcal{C}_n$ for all $n \in 1, \ldots, N$, and that each subsystem n enforces its local contract (6.10), it follows that the overall invariance condition (6.9) is satisfied.

A contract of the form (6.10) contains inherent trade-offs. In general, it is desirable that the sets $\mathcal{C}_n$ are as large as possible since they represent the set of initial conditions from where the overall invariance specification can be enforced, and, furthermore, it is easier to enforce invariance of a larger set. However, if one of the left-hand side sets $\mathcal{C}_m$ in (6.10) grows, subsystem n must be robust to a wider range of effects from subsystem m, and is therefore in general able to

enforce invariance of a comparatively smaller $\mathcal{C}_n$[1]. This implies that—as opposed to a centralized invariant set—there is not a unique maximum when searching for this type of separable invariant set.

One other interesting aspect of the local contracts is how they capture contracts between subsystems ($\Box \bigwedge_{m \neq n} (\mathbf{x}_m \in \mathcal{C}_m)$) and contracts with the environment ($\Box (\mathbf{d} \in \mathcal{D})$). In particular, if two subsystems have access to additional common information like the state of a third subsystem or of the disturbance, they can condition their local contract onto this additional information and coordinate accordingly. This is related to parametric assume-guarantee contracts [22] where common information acts as a parameter.

6.3.2 Contract Refinement Heuristic

In the next section, we describe computational procedures to evaluate whether a local contract of the form:

$$\Box[(\mathbf{d} \in \mathcal{D}) \wedge \bigwedge_{m \neq n} \Box (\mathbf{x}_m \in X_m)] \implies \Box (\mathbf{x}_n \in X_n)$$

can be upheld, by computing controlled-invariant sets $\mathcal{C}_n \subset X_n$. In case this is not possible, i.e., one or more of the sets $\mathcal{C}_n$ turn out as the empty set, the original contract needs to be refined. Based on the observation that it is easier for subsystem n to enforce invariance of X_n if the set X_m is smaller, the following heuristic can be applied for contract refinement:

1. For all n such that $\mathcal{C}_n \neq \emptyset$, set $X_n = \mathcal{C}_n$,
2. If $\mathcal{C}_n = \emptyset$ or step 1. did not result in additional nonempty sets, shrink all X_m for which $\mathcal{C}_m \neq \emptyset$ by a predefined factor.

6.4 Contract Realizability Via Polyhedral Controlled-Invariant Sets

We now describe one method for determining whether a given contract is realizable via computation of maximal controlled-invariant sets. We start with the general procedure for linear parametrized systems in Sect. 6.4.1, and then explain in Sect. 6.4.2 how the behavior of a system with nonlinear parameter dependencies can be over-approximated by the former kind.

[1]Suppose that $\mathcal{D}$ is the set of disturbances and that $\mathcal{C}$ is the maximal controlled-invariant set contained in X. Then, $\mathcal{C}$ grows with the size of X and decreases with the size of $\mathcal{D}$.

6.4.1 Computation of Polyhedral Controlled-Invariant Sets

We start by considering a discrete-time system Σ defined as:

$$\Sigma : \mathbf{x}(k+1) = f(\mathbf{x}(k), \mathbf{d}^1(k), \mathbf{u}(k), \mathbf{d}^2(k)), \tag{6.11}$$

for

$$f(x, d^1, u, d^2) = \left(A + \sum_{j=1}^{n_{d^1}} d_j^1 A_j^1 + \sum_{j=1}^{n_{d^2}} d_j^2 A_j^2 \right) x + Bu + \sum_{j=1}^{n_{d^1}} d_j^1 F_j^1 + \sum_{j=1}^{n_{d^2}} d_j^2 F_j^2,$$

together with sets $\mathcal{D}_i = \{d^i : H_{d^i} d^i \leq h_{d^i}\}$ for $i = 1, 2$ and $\mathcal{U} = \{u : H_u u \leq h_u\}$ that describe disturbance and input constraints, for $H_{d^i} \in \mathbb{R}^{\mathcal{N}_{d^i} \times n_{d^i}}$, and $H_u \in \mathbb{R}^{\mathcal{N}_u \times n_u}$. Here, $\mathbf{d}^1$ models measurable disturbance and $\mathbf{d}^2$ nonmeasurable disturbance; that is, the input $\mathbf{u}(k)$ can depend on $\mathbf{d}^1(k)$ but not on $\mathbf{d}^2(k)$. This model is more general than typical linear system models in that the disturbance can affect the system matrix.

Remark 1 The distinction between measurable and nonmeasurable disturbance can be phrased as a turn-based game: first an adversary picks the value of the measurable disturbance, then the controller selects the control input, and finally an adversary picks the nonmeasurable disturbance. Whether a given disturbance signal should be treated as measurable or nonmeasurable is a design choice. If Σ is a discrete-time representation of a continuous system, it is reasonable to treat the disturbance as measurable if the sampling rate is faster than the typical time constant of the disturbance signal.

Given a set $X \subset \mathbb{R}^{n_x}$, we are interested in computing the maximal controlled-invariant set $\mathcal{C}_\infty(X)$ contained in X. This set is characterized by the fact that whenever the system state $\mathbf{x}(k)$ is inside $\mathcal{C}_\infty(X)$, there exists an admissible control input $\mathbf{u}(k)$ that for any possible disturbance enforces $\mathbf{x}(k+1) \in \mathcal{C}_\infty(X)$. The input $\mathbf{u}(k)$ can be a function of the measurable disturbance $\mathbf{d}_1(k)$, but must be robust to any nonmeasurable disturbance $\mathbf{x}_2(k)$ inside of $\mathcal{D}_2$. Formally, $\mathcal{C}_\infty(X)$ must satisfy the following quantified property:

$$\forall x \in \mathcal{C}_\infty(X), \forall d^1 \in \mathcal{D}_1, \exists u \in \mathcal{U}, \forall d^2 \in \mathcal{D}_2,\ f(x, d_1, u, d_2) \in \mathcal{C}_\infty(X). \tag{6.12}$$

Remark 2 For simplicity, we assume in this section that the input constraint set $\mathcal{U}$ and disturbance constraint sets $\mathcal{D}_1$ and $\mathcal{D}_2$ are independent of the current value of $\mathbf{x}(k)$. It is straightforward to incorporate state-dependent input constraints by making $\mathcal{U}$ a polyhedron in (x, u) space. Also, state-dependent disturbance constraints can be incorporated, but only for disturbance signals d^i that do not affect the A matrix (i.e., $A^i = 0$).

It is well known that the set sequence computed by the following iterations converges toward $\mathcal{C}_\infty(X)$ [23]:

$$\begin{cases} V_0 = X, \\ V_{k+1} = X \cap \mathrm{Pre}^\Sigma(V_k). \end{cases} \tag{6.13}$$

Here, the backwards reachability operator $\mathrm{Pre}^\Sigma(X)$ returns the set of all initial states from where the system Σ can be steered to X in one time step, regardless of the values of measurable and nonmeasurable disturbance:

$$\mathrm{Pre}^\Sigma(X) = \left\{ x : \forall d^1 \in \mathcal{D}_1 \; \exists u \in \mathcal{U} \; \forall d^2 \in \mathcal{D}_2, \;\; f(x, d^1, u, d^2) \in X \right\}.$$

However, (6.13) does in general not terminate in finite time. Furthermore, sets in intermediate iterations are not themselves controlled invariant and can therefore not be used to enforce a contract. It is known that (6.13) converges in finite time under certain conditions [24]. If such conditions are not fulfilled, it is possible to approximate $\mathcal{C}_\infty(X)$ to arbitrary precision by imposing an artificial contraction factor ρ, which in addition results in a set that is itself controlled invariant [9, 25]. In particular, the following algorithm terminates for any $\rho > 0$, and for decreasing values of ρ yields a controlled-invariant result $\mathcal{C}_\infty^\rho \subset \mathcal{C}_\infty(X)$ that is arbitrarily close to $\mathcal{C}_\infty(X)$:

Algorithm 1: Compute inner approximation of $\mathcal{C}_\infty(X)$

```
V = X;
while V ⊄ Pre^Σ (V ⊖ B(0, ρ)) ⊕ B(0, ρ) do
    V = X ∩ Pre^Σ (V ⊖ B(0, ρ));
end
return Pre^Σ (V ⊖ B(0, ρ));
```

Proposition 1 *Algorithm 1 terminates after a finite number of iterations and returns a controlled-invariant set.*

Proof The proposition is evidently true if $V = \emptyset$ at some point in the algorithm, hence we assume that $V \neq \emptyset$ throughout the algorithm. First, we show termination in finite time. The mapping $V \mapsto X \cap \mathrm{Pre}^\Sigma(V \ominus \mathcal{B}(0, \rho))$ is monotonic with respect to set inclusion; it follows that the sequence $V_0, V_1, \ldots, V_k, \ldots$ generated by Algorithm 1 is monotonically decreasing and hence converges from the outside in Hausdorff distance to some set $\mathcal{C}_\infty^\rho(X)$. Thus, there exists a $\bar{k}$ such that

$$V_{\bar{k}} \subset \mathcal{C}_\infty^\rho(X) \oplus \mathcal{B}(0, \rho) \subset V_k \oplus \mathcal{B}(0, \rho) = X \cap \mathrm{Pre}^\Sigma\left(V_{k-1} \ominus \mathcal{B}(0, \rho)\right) \oplus \mathcal{B}(0, \rho)$$

for all $k \geq 0$. In particular, this holds for $k = \bar{k} + 1$ which shows that the algorithm finishes in at most $\bar{k}$ steps.

Secondly, the termination criterion $V \subset \text{Pre}^{\Sigma}(V \ominus \mathcal{B}(0, \rho)) \oplus \mathcal{B}(0, \rho)$ implies controlled invariance of $\text{Pre}^{\Sigma}(V \ominus \mathcal{B}(0, \rho))$. Indeed, for any $x \in \text{Pre}^{\Sigma}(V \ominus \mathcal{B}(0, \rho))$ by definition of Pre^{Σ} the system can be steered to the set $V \ominus \mathcal{B}(0, \rho)$, which by the termination criterion is a subset of the initial set $\text{Pre}^{\Sigma}(V \ominus \mathcal{B}(0, \rho))$. □

The remaining ingredient required to compute or inner-approximate $\mathcal{C}_{\infty}(X)$ is computation of Pre^{Σ}. Computation amounts to eliminating the quantifiers over disturbance and control which can be reformulated as intersection and projection operations. For a set $V \subset \mathbb{R}^x \times \mathbb{R}^u$, let $\text{proj}_x(V)$ denote the projection of V onto $\mathbb{R}^x$. Also, let $\mathcal{V}(\mathcal{P})$ denote the set of vertices of a polyhedron $\mathcal{P}$. An exact way to compute Pre^{Σ} is given as follows:

Theorem 1

$$\text{Pre}^{\Sigma}(X) = \bigcap_{d^1 \in \mathcal{V}(\mathcal{D}_1)} \text{proj}_x \left\{ (x, u) : H_x f(x, d^1, u, d^2) \leq h_x, H_u u \leq h_u \ \ \forall d^2 \in \mathcal{V}(\mathcal{D}_2) \right\}.$$

With this method, $\text{Pre}^{\Sigma}(X)$ can be computed via $|\mathcal{V}(\mathcal{D}_1)|$ projections of $(n_x + n_u)$-dimensional polyhedra, each with $\mathcal{N}_x|\mathcal{V}(\mathcal{D}_2)| + \mathcal{N}_u$ inequalities. Polyhedral projections can be computed numerically via, e.g., Fourier–Motzkin elimination implemented in toolboxes such as MPT [26]. The sets of vertices $\mathcal{V}(\mathcal{D}_i)$ are finite but can potentially be very large—just the problem of enumerating the vertex set can be computationally challenging. An alternative way of computing an inner approximation of Pre^{Σ} that does not require vertex enumeration is to use the method of robust counterparts [27].

6.4.2 Over-Approximation of Nonlinear Parametrizations

We now discuss how to over-approximate a system with nonlinear interconnections with a system of the form (6.11) in order to leverage the method of the previous subsection. Consider a subsystem Σ_1 with state $\mathbf{x}_1 \in \mathbb{R}^{n_x}$ with dynamics that are linear in its own states, but have potential nonlinear parameter dependencies on the states of other subsystems $\Sigma_2, \ldots, \Sigma_N$.

$$\Sigma_1 : \ \mathbf{x}_1(k+1) = \left(A_1 + \sum_{m=2}^{N} A_m(\mathbf{x}_m(k)) \right) \mathbf{x}_1(k) + B_1 \mathbf{u}_1(k) + \sum_{m=2}^{N} F_m(\mathbf{x}_m(k)). \tag{6.14}$$

Assume that for all $m = 2, \ldots, N$, there is a finite collection $\{f_m^j\}_{j=1}^{q_m}$ of q_m nonconstant functions $f_m^j : X_m \to \mathbb{R}$, linearly independent of X_m, such that $A_m(x_m)$ can be represented as $A_m(x_m) = \sum_{j=1}^{q_m} f_m^j(x_m) A_m^j$ and $F_m(x_m) = \sum_{j=1}^{q_m} f_m^j(x_m) F_m^j$, where A_m^j and F_m^j are constant matrices for all $j \in 1, \ldots, q_m$. Note that such a collection $\{f_m^j\}_{j=1}^{q_m}$ with at most $n_x^2 + n_x$ members always exists, but q_m is often much smaller than $n_x^2 + n_x$. Define the function $f_m : \mathcal{C}_m \to \mathbb{R}^{q_m}$ as:

$$f_m : x_m \mapsto [f_m^1(x_m)\ f_m^2(x_m) \ldots f_m^{q_m}(x_m)]^T,$$

which accounts for all the nonlinearities from subsystem m.

We divide the subsystems $2, \ldots, N$ into two information set indices J^1 (information available) and J^2 (information unavailable) in order to distinguish between subsystems for which the states are available, and those for which the only information is that the states are within the sets X_m. Then, the following generalized backwards reachable set operator is required to do invariant set computations for Σ_1:

$$\mathrm{Pre}^{\Sigma_1}(X) = \left\{ \begin{array}{l} x_1 : (\forall m \in J^1,\ \forall x_m \in X_m)(\exists u_1 \in \mathcal{U}_1)\ (\forall m \in J^2,\ \forall x_m \in X_m) \\ \left(A_1 + \sum_{m=2}^{N} \sum_{j=1}^{q_m} f_m^j(x_m) A_m^j \right) x_1 + B_1 u_1 + \sum_{m=2}^{N} \sum_{j=1}^{q_m} f_m^j(x_m) F_m^j \in X \end{array} \right\}. \tag{6.15}$$

That is, the subsystems in J^1 are treated as a *measurable* disturbance, so the control is allowed to depend on these state values. On the contrary, the only information about the *nonmeasurable* subsystems in J^2 is that they are contained in the sets X_m, so the control action must be robust with respect to all possible values in these sets.

The computation of (6.15) is not possible with the theory in Sect. 6.4.1 due to the nonlinear terms f_m^j. In the following, we show how to remove these by first moving the nonlinearities to the set descriptions, and then over-approximating nonlinear sets with convex hulls. We also show that the latter is without loss of generality.

6.4.3 Removal of Nonlinearities via Convexification

The lemma below shows that the nonlinearities associated with f_m can be accounted for in a linear system where the nonlinearities have been moved to a set description $f_m(X_m)$. In addition, there is no loss of precision if the convex hull of $f_m(X_m)$ is considered. Once this result is established, it is extended to a reformulation of the backwards reachable set operator in Corollary 1 below.

Lemma 1 *Let X be a convex set, then*

$$\forall x_m \in X_m, \quad \left(A_1 + \sum_{j=1}^{q_m} f_m^j(x_m) A_m^j \right) x_1 + \sum_{j=1}^{q_m} f_m^j(x_m) F_m^j \in X, \tag{6.16}$$

if and only if

$$\forall d_m \in \text{conv}(f_m(X_m)), \quad \left(A_1 + \sum_{j=1}^{q_m} d_m^j A_m^j\right) x_1 + \sum_{j=1}^{q_m} d_m^j F_m^j \in X. \tag{6.17}$$

Proof Statement (6.16) can be rewritten as:

$$\forall d_m \in f_m(X_m), \quad \left(A_1 + \sum_{j=1}^{q_m} d_m^j A_m^j\right) x_1 + \sum_{j=1}^{q_m} d_m^j F_m^j \in X. \tag{6.18}$$

The implication from (6.17) to (6.16) is now obvious since $f_m(X_m) \subset \text{conv}\,(f_m(X_m))$. Suppose that (6.16) holds but not (6.17). Then, there is $d_m \in \text{conv}(f_m(X_m))$ such that

$$\left(A_1 + \sum_{j=1}^{q_m} d_m^j A_m^j\right) x_1 + \sum_{j=1}^{q_m} d_m^j F_m^j \notin X.$$

By definition of the convex hull, $d_m = \sum_{l=1}^{K} \alpha_l d_l$ for $\alpha \in \Delta_K$ and $d_l \in f_m(X_m)$. But, from (6.18):

$$\begin{aligned}&\left(A_1 + \sum_{j=1}^{q_m} d_m^j A_m^j\right) x_1 + \sum_{j=1}^{q_m} d_m^j F_m^j \\ &\quad = \sum_{l=1}^{K} \alpha_l \left(\left(A_1 + \sum_{j=1}^{q_m} d_l^j A_m^j\right) x_1 + \sum_{j=1}^{q_m} d_l^j F_m^j\right) \in X\end{aligned}$$

by convexity of X—a contradiction. □

Corollary 1 *For a convex $\mathcal{U}_1$, the backwards reachable set operator (6.15) can be written as follows:*

$$\text{Pre}^{\Sigma_n}(X) = \left\{ \begin{array}{l} x_n : (\forall m \in J^1,\ \forall d_m \in \text{conv}(f_m(X_m)))\ (\exists u_n \in \mathcal{U}_n) \\ \qquad (\forall m \in J^2,\ \forall d_m \in \text{conv}(f_m(X_m))) \\ \left(A_1 + \sum_{m=2}^{N} \sum_{j=1}^{q_m} d_m^j A_m^j\right) x_n + B_1 u_1 + \sum_{m=2}^{N} \sum_{j=1}^{q_m} d_m^j F_m^j \in X \end{array} \right\}. \tag{6.19}$$

Proof (Sketch) The corollary follows by twice applying Lemma 1. The first application to convexify the sets associated with J^2 is straightforward since other

parts (i.e., terms related to J^1 and u_1) can be taken as constant. To substitute the sets associated with J^1, nonrelevant parts of the expression can be moved to the set description X by eliminating the inner $\forall$ and $\exists$ quantifiers, to arrive at an expression like in Lemma 1. By convexity of $\mathcal{U}_1$, the modified set X is convex. □

Corollary 2 *If* $\text{conv}(f_m(X_m))$ *is replaced by* V_m *in* (6.19) *for some* $V_m \supset \text{conv}(f_m(X_m))$*, then an inner approximation of* $\text{Pre}^{\Sigma_n}(X)$ *is obtained.*

As a consequence of these results, the nonlinearities in f_m can be moved to the set description if the convexified image of X_m under f_m can be computed. The dynamics in (6.19) are on the form (6.11) as desired. However, the convex hulls $\text{conv}(f_m(X_m))$ are not necessarily polyhedra as required in Sect. 6.4.1.

Utilizing over-approximations of the sets $\text{conv}\,(f_m(X_m))$ results in inner approximations of $\text{Pre}^{\Sigma_n}(X)$. Such inner approximations still yield sound (but not complete) winning sets when employed in the fixed-point computations (6.13). Next, we present two explicit techniques for computing polyhedral over-approximations of the convex hull $\text{conv}(f(X))$ for a multivariate function $f : X \to \mathbb{R}^{n_d}$ and a convex set $X \subset \mathbb{R}^p$. Any of these procedures can be utilized in order to compute (6.19) using the results in Sect. 6.4.1.

6.4.3.1 Convex-Hull Computation With Monotone Functions

The first technique computes an over-approximation of $\text{conv}(f(X))$ provided that f satisfies certain monotonicity conditions, and that X is a hyper rectangle.

Monotonicity is defined with respect to a given *cone*, which is a set $K \subset \mathbb{R}^n$ such that $x, y \in K$ implies $x + y \in K$, and such that $x \in K$ implies $tx \in K$ for all scalars $t \geq 0$. A cone K induces a partial ordering $\leq_K$ on $\mathbb{R}^n$ given by:

$$x \leq_K y \quad \Longleftrightarrow \quad y - x \in K.$$

An *interval* with respect to a cone K is a convex set X for which there exist extreme points $x^-, x^+ \in X$ such that $x^- \leq_K x \leq_K x^+$ for all $x \in X$; intervals are denoted $[x^-, x^+]_K$. For instance, when the cone K corresponds to one of the orthants in $\mathbb{R}^n$, intervals are hyper rectangles.

Given two cones $K_1 \subset \mathbb{R}^p$ and $K_2 \subset \mathbb{R}^q$, a function $f : \mathbb{R}^p \to \mathbb{R}^q$ is *monotone* on the set X with respect to K_1 and K_2 if for all $x, y \in X$, $x \leq_{K_1} y$ implies that $f(x) \leq_{K_2} f(y)$. A simple condition to verify monotonicity of a function $f^i : \mathbb{R}^p \to \mathbb{R}$ is the sign stability of its gradient:

Proposition 2 *Given a hyper rectangle* $\mathcal{C} \subset \mathbb{R}^p$*, a mapping* $f^i : \mathcal{C} \to \mathbb{R}$ *is monotone on* $\mathcal{C}$ *if each element of the gradient of* f^i *maintains its sign on* $\mathcal{C}$*. That is, for all* j*, there exists* $\sigma_j \in \{0, 1\}$ *such that*

$$(-1)^{\sigma_j} \frac{\partial f^i}{\partial x_j}(x) \geq 0 \quad \forall x \in \mathcal{C}, \tag{6.20}$$

then f^i is monotone with respect to the cones $K_1^i = \{x = [x^1, \ldots, x^p]^\intercal \in \mathbb{R}^p \mid (-1)^{\sigma_j} x_j \geq 0\}$, $K_2 = [0, \infty)$. Moreover, if all components f^i, $i = 1, \ldots, q$, of a function $f : \mathcal{C} \to \mathbb{R}^q$ are monotone with respect to some cones K_1^i and $K_2 = [0, \infty)$, then $f(\mathcal{C}) \subset \prod_{i=1}^p [f^i(\underline{x}^i), f^i(\overline{x}^i)]$, where $\underline{x}^i$ and $\overline{x}^i$ are extreme points of $\mathcal{C}$ with respect to the order induced by K_1^i.

Note that since $\prod_{i=1}^p [f^i(\underline{x}^i), f^i(\overline{x}^i)]$ in the above proposition is a convex set, it contains $\text{conv}(f(\mathcal{C}))$. Given f and $\mathcal{C}$, if the conditions in the above proposition fail, i.e., the gradients of component functions of f are not sign-stable on $\mathcal{C}$, but there is a finite cover $\{\mathcal{C}_l\}_{l=1}^L$ of $\mathcal{C}$ with $\mathcal{C} = \bigcup_{l=1}^L \mathcal{C}_l$ and each $\mathcal{C}_l$ is a hyper rectangle where the conditions of the proposition hold, then it is still possible to find a convex over-approximation of $\text{conv}(f(\mathcal{C}))$. This is done by over-approximating the convex hull of each $f(\mathcal{C}_l)$ with a hyper rectangle using the proposition above, and then taking the convex hull of these hyper rectangles. The next result shows that, under certain conditions, it is possible to obtain an arbitrarily tight over-approximation of $\text{conv}(f(\mathcal{C}))$ by covering $\mathcal{C}$ with hyper-boxes of decreasing size. The idea is illustrated in the left part of Fig. 6.4.

Theorem 2 *Let $\{\mathcal{C}_l\}_{l=1}^L$ be a hyper-rectangular cover of $\mathcal{C}$, i.e., $\mathcal{C} = \bigcup_{l=1}^L \mathcal{C}_l$, and each $\mathcal{C}_l$ is a hyper rectangle. Assume that the gradients of the component functions of f are sign-stable on each $\mathcal{C}_l$. Also, let $[\mathcal{C}_l]_\epsilon$ be a finite hyper-rectangular cover of $\mathcal{C}_l$, where the size of the maximum edge of each hyper rectangle $R \in [\mathcal{C}_l]_\epsilon$ is at most ϵ and $\mathcal{C}_l = \bigcup_{R \in [\mathcal{C}_l]_\epsilon} R$. Then,* $\text{conv}(f(\mathcal{C})) = \lim_{\epsilon \to 0} \text{conv}(\{\{f(R)\}_{R \in [\mathcal{C}_l]_\epsilon}\}_{l=1}^L)$.

Proof The result follows from the following claims: i) for any sets A and B, $\text{conv}(A \cup B) = \text{conv}(\text{conv}(A) \cup \text{conv}(B))$, and ii) for sets A and B such that $h(A, B) \leq \epsilon$, it holds that $h(\text{conv}(A), \text{conv}(B)) \leq \epsilon$, where h is the Hausdorff distance function.

Claim i) follows directly from the definition of convex hulls. To show ii), take any $a \in \text{conv}(A)$; it can be written as $a = \sum_{i \in I} \alpha_i a_i$ with $a_i \in A$ and $\sum_{i \in I} \alpha_i = 1$, $\alpha_i \geq 0$. For each a_i, there is a $b_i \in B$ s.t. $\|a_i - b_i\| \leq \epsilon$. Evidently, $b = \sum_{i \in I} \alpha_i b_i \in \text{conv}(B)$, and $\|a - b\| = \|\sum_i \alpha_i (a_i - b_i)\| \leq \sum_{i \in I} \alpha_i \|a_i - b_i\| \leq \epsilon$, which shows $h(\text{conv}(A), \text{conv}(B)) \leq \epsilon$. □

6.4.3.2 Convex-Hull Computation With Convex Projections

Ideas similar to those for monotonicity can also be used when the component functions f^i of f are convex or concave on some convex polyhedral sets $\mathcal{C}_l$ that cover the set $\mathcal{C}$. Note that when f^i is convex (concave), the minimum (maximum) of f^i on each $\mathcal{C}_l$ can be found by convex programming and the maximum (minimum) can be found by evaluating the function f^i on the vertices of $\mathcal{C}_l$. Such a strategy again creates a collection of hyper rectangles that over-approximate $f(\mathcal{C})$. However, the convex hull of these hyper rectangles can potentially have a large number of vertices. Next, attention is restricted to a very special case with a single parameter, valid for the lane-keeping application studied later, for which over-approximations with smaller numbers of vertices can be computed.

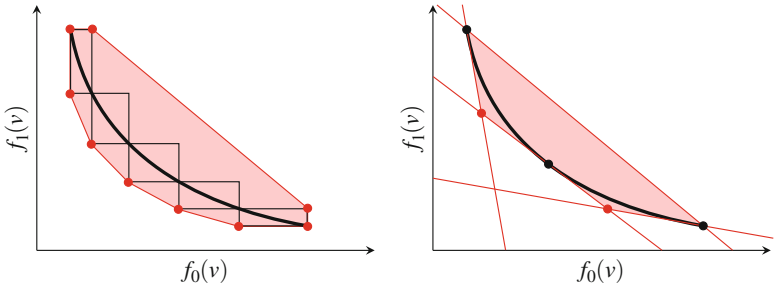


Fig. 6.4 Convex-hull over-approximation of images of a function. The convex hull is constructed around a nonlinear curve $f : [a, b] \to \mathbb{R}^2$ using the monotonicity (left) and gradient (right) methods

Consider the special case of a map $f = (f^0, f^1) : [\underline{x}, \overline{x}] \to \mathbb{R}^2$ where f^1 is convex in f^0 over $[\underline{x}, \overline{x}]$, i.e., the function $f^1 \circ (f^0)^{-1} : f^0([\underline{x}, \overline{x}]) \to \mathbb{R}$ is a convex function. In this case, the line connecting $f(\underline{x})$ to $f(\overline{x})$ lies completely above the graph of $f([\underline{x}, \overline{x}])$. Furthermore, any tangent line to $f([\underline{x}, \overline{x}])$ lies completely below the graph of $f([\underline{x}, \overline{x}])$. The right part of Fig. 6.4 illustrates the idea, where one hyperplane constraint is constructed using the former fact, and three hyperplane constraints are constructed using the latter fact. In particular, the former constraint is of the form:

$$-mf^0(x) + f^1(x) \leq f^1(\underline{x}) - mf^0(\underline{x}), \quad m = \frac{f^1(\overline{x}) - f^1(\underline{x})}{f^0(\overline{x}) - f^0(\underline{x})},$$

and the latter constraints are of the form:

$$m_0 f^0(x) + m_1 f^1(x) \geq m_0 f^0(x_i) + m_1 f^1(x_i),$$

where $m_0 \frac{\mathrm{d}f^0}{\mathrm{d}x}(x_i) + m_1 \frac{\mathrm{d}f^1}{\mathrm{d}x}(x_i) = 0$ and $x_i \in [\underline{x}, \overline{x}]$.

This method can also be applied in higher dimensions, i.e., when $f : [\underline{x}, \overline{x}] \to \mathbb{R}^q$, provided that the components $f^0, \ldots, f^{q-1}$ of f pairwise satisfy the convexity property described above. In this case, two-dimensional sets $\mathcal{P}_{ij}$ can be computed for all i, j with $i \neq j$, and then lifted to q dimensions to obtain $\mathrm{conv}(f(\mathcal{C})) \subset \mathcal{P} = \{[x^1, \ldots, x^q]^\intercal : (x^i, x^j) \in \mathcal{P}_{ij} \; \forall \, i \neq j\}$.

6.5 Design Flow for the Case Study

We now apply the theory developed in the chapter to synthesize LK and ACC controllers with compositional guarantees. The system models (6.1) and (6.2) are given in continuous time, in order to apply the methods from previous sections we

therefore time-discretize the models using Euler forward with a time step of 0.1 s, which preserves the base $\{f_i\}_{i=1}^{q}$ from which the system matrices are composed. When implementing a correct discrete-time controller in continuous time, inter-sample constraint violations may occur as well as differences from the omission of higher-order terms in the Euler forward approximation (as opposed to the exact exponential map). It is possible to account for these small errors by adding certain robustness margins [28], but we omit those details here.

6.5.1 Constraints

The fundamental objectives of the LK and ACC systems are to stay in the lane, and to keep a safe distance to the lead car. To quantify these requirements, we impose a constraint $|\mathbf{y}| \leq 0.7$ to prevent lane departure, and a constraint $\mathbf{h} \geq 5$ that limits the minimal distance to the lead car. The headway bound is adjusted for the low speeds suitable for the autonomous test track Mcity.

In addition to state safety constraints, for comfort reasons we impose control input bounds $\mathbf{F}_w \in [-3m, 2m]$ and $|\mathbf{d}_f| \leq 45\pi/180$, respectively. This yields the overall specification:

$$\varphi = \square\,(\mathbf{F}_w \in [-3m, 2m]) \wedge \square\left(|\mathbf{d}_f| \leq 45\pi/180\right) \wedge \square(|\mathbf{y}| \leq 0.7) \wedge \square(\mathbf{h} \geq 5). \tag{6.21}$$

6.5.2 Contracts

In order to decompose the problem in a formal way, we introduce contracts between the two subsystems. The desired outcomes of the contracts are the bounds from the previous section. To achieve these, we impose the following contracts that are also illustrated in Fig. 6.5:

1. Given item 3, the ACC subsystem promises that the forward velocity is within a range $\mathbf{u} \in [6.94, 8.33]$,
2. Given item 3, the ACC subsystem promises that the headway constraint $\mathbf{h} \geq 5$ is satisfied,
3. Given item 1, the LK subsystem promises that $|\mathbf{v}| \leq 1$ and that $|\mathbf{r}| \leq 0.78$,
4. Given item 1, the LK subsystem promises that the lane departure constraint $|\mathbf{y}| \leq 0.7$ is satisfied.

Items 2 and 4 in the list above are the state invariance constraints in the original specification (6.21), whereas items 1 and 3 are *internal* assume-guarantee contracts between the subsystems. The numbers in the contracts are again selected for driving at relatively low speeds; a more capable implementation would likely need different contracts for varying driving conditions. In Table 6.1, the state bounds imposed by the contracts are listed, together with system parameters that we use in the following.

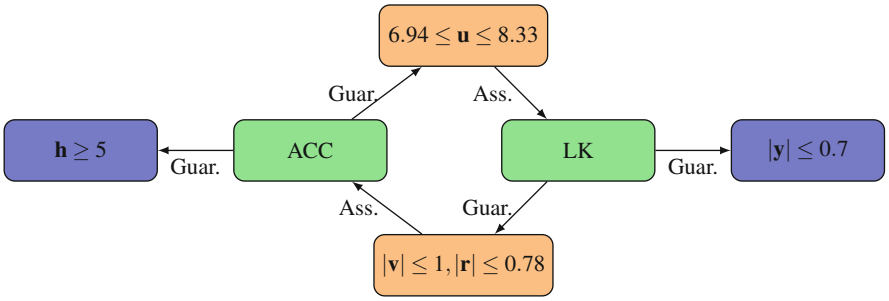


Fig. 6.5 Illustration of contract relationships between system components

Table 6.1 Parameter values (left) and state constraints (right) for ACC and LK models

Parameters				State	Lower bound	Upper bound
m	1800 kg	$C_{\alpha f}$	140,000 N	$\mathbf{u}$	6.94 m/s	8.33 m/s
a	1.20 m	$C_{\alpha r}$	120,000 N	$\mathbf{h}$	5 m	∞
b	1.65 m	u_L	25.2 km/h	$\mathbf{y}$	−0.7 m	0.7 m
f_0	74.63 N	I_z	3270 kg m^2	$\mathbf{v}$	−1 m/s	1 m/s
f_1	40.59 Ns/m			$\mathbf{r}$	−0.78 rad/s	0.78 rad/s

6.5.3 *Realizability of LK Contract*

Given the decomposition via contracts, we now separately synthesize LK and ACC controllers that enforce the local contracts. To satisfy its contract, the LK subsystem must satisfy the specification $\varphi_{LK}^{ASS} \implies \varphi_{LK}$, where

$$\varphi_{LK}^{ASS} = \square\,(6.94 \leq \mathbf{u} \leq 8.33) \wedge \square\,(|\mathbf{r}_d| \leq 0.5)\,, \tag{6.22a}$$

$$\varphi_{LK} = \square\left(|\mathbf{d}_f| \leq 45\pi/180\right) \wedge \square\,(|\mathbf{y}| \leq 0.7) \wedge \square\,(|\mathbf{v}| \leq 1) \wedge \square\,(|\mathbf{r}| \leq 0.78)\,. \tag{6.22b}$$

Here, $|\mathbf{r}_d| \leq 0.5$ is an assumption on the maximal road curvature affecting the LK subsystem, which can be treated as measurable disturbance. This is an invariance specification and $\mathbf{u}$ can be treated as measurable disturbance to the LK subsystem; a controller that enforces the specification can therefore be computed with the method in Sect. 6.4. In particular, the entries of A_{LK} in (6.2) are composed of two linearly independent nonlinearities: u and $1/u$. Using the method in Sect. 6.4.3.2, we compute a polyhedron $\mathcal{P}_{LK} \subset \mathbb{R}^2$ that is an over-approximation of all values the function $u \mapsto [u, 1/u]$ can take when u varies in the interval $[6.94, 8.33]$. Subsequently applying Algorithm 1 with Pre^{Σ} computed via Theorem 1 yields the set $\mathcal{C}_{LK}$ depicted in Fig. 6.6, which is robustly controlled invariant under the assumption (6.22a).

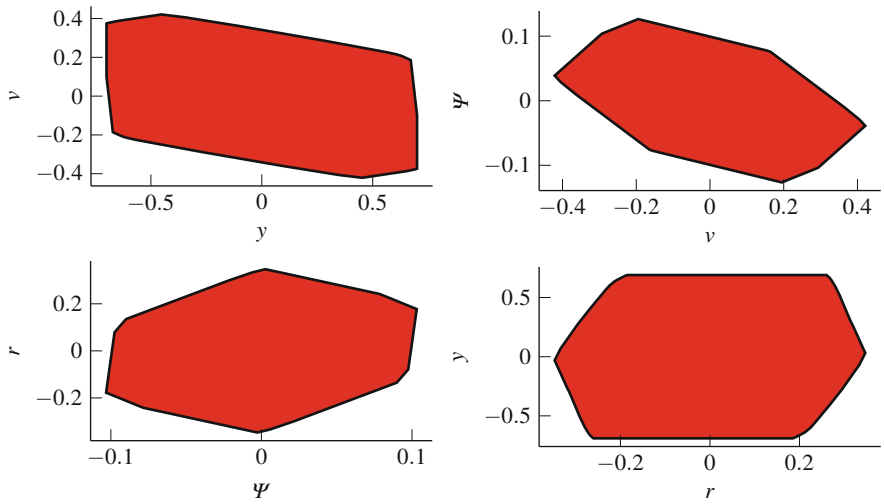


Fig. 6.6 Two-dimensional sections through the origin of the four-dimensional controlled-invariant set $\mathcal{C}_{LK}$ for the LK system (6.2)

6.5.4 Realizability of ACC Contract

We proceed by synthesizing an ACC controller that satisfies the corresponding part of the contract. In particular, the local ACC specification is $\varphi_{ACC}^{ASS} \implies \varphi_{ACC}$, for

$$\varphi_{ACC}^{ASS} = \square\,(|\mathbf{v}| \leq 1) \wedge \square\,(|\mathbf{r}| \leq 0.78)\,, \tag{6.23a}$$

$$\varphi_{ACC} = \square\,(\mathbf{F}_w \in [-3m, 2m]) \wedge \square\,(\mathbf{h} \geq 5) \wedge \square\,(6.94 \leq \mathbf{u} \leq 8.33)\,. \tag{6.23b}$$

Again, this local synthesis problem fits into the framework of Sect. 6.4. For illustration purposes, we describe in some detail how the effect of the LK subsystem on the ACC subsystem can be over-approximated. In the ACC system (6.1), there is an offset term $\mathbf{vr}$ that is a function of the state of the LK system. Our goal is to find a polyhedron $\mathcal{P}_{ACC} \subset \mathbb{R}$ that is an over-approximation of the range of the function $f : [v, r] \mapsto vr$ where $v \in [-1, 1]$ and $r \in [-0.78, 0.78]$. The exact range of this function can easily be seen to be $[-0.78, 0.78]$; however, we will formalize the derivation of this range using the monotonicity method described in Sect. 6.4.3.1. In fact, since f maps to $\mathbb{R}$, this method will also produce the exact range of f. We first note that $\nabla f = [r, v]$, which is sign-stable on each quadrant of the (v, r) plane. Consider quadrant one, where f is monotone w.r.t. the cones $K_1 = \{x \in \mathbb{R}^2 \mid x_j \geq 0 \;\; \forall j = 1, 2\}$, $K_2 = [0, \infty)$. Therefore, the maximum value of f on the first quadrant occurs at $(v, r) = (1, 0.78)$, and the minimum value occurs at the origin so that $f([0, 1]) \times [0, 0.78]) = [f(0, 0), f(1, 0.78)] = [0, 0.78]$. Similarly, the range of f on quadrant three is found to be $[0, 0.78]$, and the range

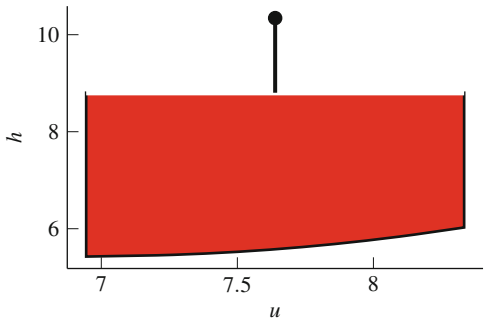


Fig. 6.7 Controlled-invariant set $\mathcal{C}_{ACC}$ for the ACC system (6.1). The line indicates that the set extends to positive infinity in the h direction

of f on quadrants two and four is identically $[-0.78, 0]$. Thus, the range of f is precisely $\text{conv}([-0.78, 0], [0, 0.78]) = [-0.78, 0.78]$. Using this bound, we then proceed as with the LK system to obtain the controlled-invariant two-dimensional polyhedron $\mathcal{C}_{ACC}$ shown in Fig. 6.7.

6.5.5 Low-Fidelity Simulation Results

Once the two sets $\mathcal{C}_{LK}$ and $\mathcal{C}_{ACC}$ that satisfy the two contracts (6.22) and (6.23) are obtained, controllers that enforce the overall specification (6.21) can be implemented by choosing for a given state $\mathbf{x}_{ACC}(k) \in \mathcal{C}_{ACC}$ and $\mathbf{x}_{LK}(k) \in \mathcal{C}_{LK}$ an input $\mathbf{F}_w(k)$ such that $A(\mathbf{x}_{LK}(k))\mathbf{x}_{ACC}(k) + B\mathbf{F}_w(k) + F(\mathbf{x}_{LK}(k)) \in \mathcal{C}_{ACC}$, and conversely for the LK input $\mathbf{d}_f(k)$. Finding such inputs amounts to enforcing certain linear constraints which by construction are feasible.

We first demonstrate the controller performance via simultaneous simulations of the design models (6.1) and (6.2). As the method is provably correct with respect to these models, we expect all constraints from Sect. 6.5.1 to be satisfied. To illustrate the effect of constraint guarantees, we select steering inputs in a "lazy" way: we only steer when absolutely necessary to avoid exiting the lane bounds. In Fig. 6.8, a simultaneous simulation of the ACC and LK systems is shown, where the road curvature $\mathbf{r}_d$ is in the form of a sinusoidal signal with maximal amplitude 0.25. As can be seen in the plot for $\mathbf{y}$, the car steers as little as possible which makes it bounce between the lane edges, but still all bounds are respected throughout the simulation.

6.5.6 CarSim Simulation Results

Although the controllers are provably correct with respect to the design models, further validation is mandated to ensure that the design models capture the essential dynamics of the real systems of interest. As a first such validation, we test them in CarSim [29]—an industry-grade simulation package. Recent versions of CarSim

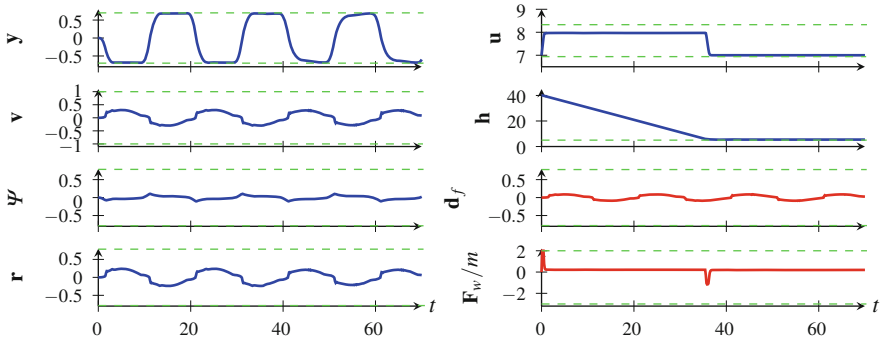


Fig. 6.8 Simultaneous simulation of the LK and ACC subsystems on a curvy road. States are shown in blue, and computed control inputs in red. The dashed green lines indicate state and control bounds that are guaranteed to hold. The ACC controller is programmed to maintain a desired speed of 8 m/s when possible. However, it is forced to slow down at $t = 35$ to avoid future violation of the headway constraint

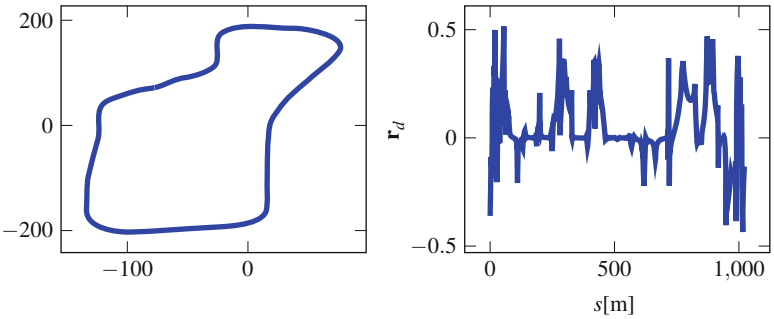


Fig. 6.9 Illustration of Mcity path used in testing. Left: outline of the path that is driven counterclockwise. Right: curvature along the path which is absolutely bounded by 0.5 as in (6.22a)

allow for driving in a virtual Mcity. We test the controllers by driving the outer path illustrated in Fig. 6.9 without a lead car. State trajectories for a complete lap are shown in Fig. 6.10, a video of this simulation is also available at http://web.eecs.umich.edu/~necmiye/figs/mcity.mp4.

6.6 Implementation in Mcity

We also report on our initial tests with a real vehicle in Mcity—the connected and automated vehicle testing facility at the University of Michigan. The vehicle platform used is a Lincoln MKZ equipped with various sensors and a drive-by-wire system shown in Fig. 6.11 (left). The platform is running Polysync middleware

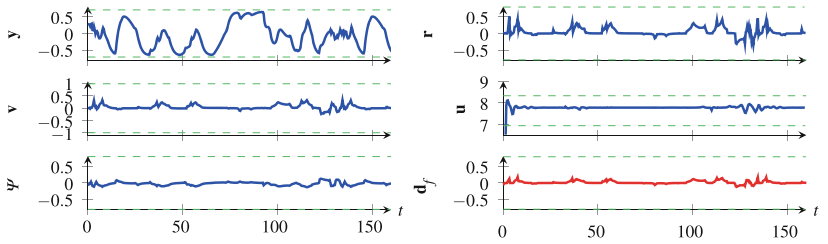


Fig. 6.10 CarSim simulation results from the virtual Mcity test track. Although the controllers are provably correct only with respect to the design models, they still enforce all constraints in the high-fidelity CarSim simulation, which validates our choice of design models

Fig. 6.11 Left: Lincoln MKZ used in the experiments. Right: The desired path in blue and the actual path in red overlaid on a satellite image of Mcity highway portion

[30] that provides easy communication with sensors and actuators. The principal actuators are the steering wheel, throttle, and brake pedals. A high-precision RTK GPS sensor is used to provide global position and orientation information to the controllers. Our design flow allows us to directly deploy the algorithms tested in CarSim as Polysync-compatible C-code via MATLAB Simulink code generation.

There were some challenges in realizing provably-correct longitudinal control on the vehicle platform. The input to the longitudinal dynamics model (6.1) is wheels force (torque). Applying a given wheel torque requires knowing the (engine-speed-dependent) mapping from throttle and brake pedal positions to the wheel torque, which is not available publicly for Lincoln MKZ. Moreover, since the vehicle is hybrid, reverse-engineering such a mapping is even more challenging. Therefore, we opted for implementing only the LK component of the synthesized controllers and used a simple proportional integral (PI) controller for longitudinal control. Thanks to the modular framework provided by contracts, replacing the ACC with a PI controller does not require redesigning the LK controller. As long as the PI

controller satisfies the same local ACC contract, correctness of the composition is still preserved. Under the assumption that the PI controller guarantees to keep the forward velocity of the car in some range,[2] the LK controller is guaranteed to keep the vehicle inside the lane.

For test purposes, we constructed a virtual lane in Mcity traveling from south to north. The center line of the virtual lane is plotted in blue in Fig. 6.11 (right), which essentially involves a lane change at the highway portion of Mcity followed by a right curve. The actual trajectory under the control of the invariance-enforcing LK controller is depicted in red, which can be seen to closely follow the center line of the virtual lane.

6.7 Conclusions

In this chapter, we presented a contract-based compositional framework for synthesizing provably-correct safety controllers for interacting subsystems. Our approach can handle certain types of parametric uncertainty and takes into account information constraints, thus providing a means to analyze the trade-off between conservatism and scalability. We demonstrated the design flow on two autonomous driving functions, and tested the resulting controllers in both simulations and on an actual car in Mcity. Future work will consider richer contracts beyond safety that take preview information into account.

Acknowledgments The authors would like to thank Kwesi Rutledge for help with CarSim simulations and the Mcity implementation, Jessy Grizzle for useful discussions on vehicle safety systems, Stanley Smith for his contributions at the early stages of the work, and the Mcity team for assistance with experiments. This work is supported in part by NSF grants CNS-1239037, CNS-1446298, and ECCS-1553873, and by Toyota Research Institute (TRI). TRI provided funds to assist the authors with their research, but this chapter solely reflects the opinions and conclusions of the authors and not those of TRI or any other Toyota entity.

References

1. G. Frehse, Z. Han, B. Krogh, Assume-guarantee reasoning for hybrid i/o-automata by over-approximation of continuous interaction, in *Proceedings of IEEE CDC* (2004), pp. 479–484
2. L. Benvenuti, A. Ferrari, E. Mazzi, A.L. Sangiovanni-Vincentelli, Contract-based design for computation and verification of a closed-loop hybrid system, in *Proceedings of HSCC* (2008), pp. 58–71

[2] Although whether the PI controller can guarantee this assumption or not has not been formally verified, the contract provides us a means to check the satisfaction or violation of the assumption at run time.

3. P. Nuzzo, H. Xu, N. Ozay, J.B. Finn, A.L Sangiovanni-Vincentelli, R.M. Murray, A. Donzé, S.A Seshia, A contract-based methodology for aircraft electric power system design. IEEE Access **2**, 1–25 (2014)
4. I. Filippidis, Decomposing formal specifications into assume-guarantee contracts for hierarchical system design. Ph.D. thesis, California Institute of Technology, 2018
5. E.S. Kim, M. Arcak, S.A. Seshia, Compositional controller synthesis for vehicular traffic networks, in *Proceedings of IEEE CDC* (2015), pp. 6165–6171
6. P. Nilsson, N. Ozay, Synthesis of separable controlled invariant sets for modular local control design, in *Proceedings of ACC* (2016), pp. 5656–5663
7. S. Smith, P. Nilsson, N. Ozay, Interdependence quantification for compositional control synthesis: an application in vehicle safety systems, in *Proceedings of IEEE CDC* (2016), pp. 5700 – 5707
8. F. Blanchini, Set invariance in control. Automatica **35**(11), 1747–1767 (1999)
9. E. De Santis, M.D. Di Benedetto, L. Berardi, Computation of maximal safe sets for switching systems. IEEE Trans. Autom. Control **49**(2), 184–195 (2004)
10. P. Nilsson, O. Hussien, A. Balkan, Y. Chen, A. Ames, J. Grizzle, N. Ozay, H. Peng, P. Tabuada, Correct-by-construction adaptive cruise control: two approaches. IEEE Trans. Control Syst. Technol. **24**(4), 1294–1307 (2016)
11. E.J. Rossetter, J. Christian Gerdes, Lyapunov based performance guarantees for the potential field lane-keeping assistance system. J. Dyn. Syst. Meas. Control. **128**(3), 510–522 (2006)
12. G.J.L. Naus, J. Ploeg, M.J.G. Van de Molengraft, W.P.M.H. Heemels, M. Steinbuch, Design and implementation of parameterized adaptive cruise control: an explicit model predictive control approach. Control. Eng. Pract. **18**(8), 882–892 (2010)
13. S. Ishida, J.E. Gayko, Development, evaluation and introduction of a lane keeping assistance system, in *IEEE Intelligent Vehicles Symposium* (2004), pp. 943–944
14. D. Hoehener, G. Huang, D. Del Vecchio, Design of a lane departure driver-assist system under safety specifications, in *Proceedings of IEEE CDC* (2016), pp. 2468–2474
15. T. Korssen, V. Dolk, J. van de Mortel-Fronczak, M. Reniers, M. Heemels, Systematic model-based design and implementation of supervisors for advanced driver assistance systems. IEEE Trans. Intell. Transp. Systems **PP**, 1–12 (2017)
16. K. Chatterjee, T.A. Henzinger, Assume-Guarantee Synthesis, in *Proceedings of TACAS* (2007), pp. 261–275
17. L. Benvenuti, A. Ferrari, E. Mazzi, A.L. Sangiovanni-Vincentelli, Contract-based design for computation and verification of a closed-loop hybrid system, in *Proceedings of HSCC*, ed. by M. Egerstedt, B. Mishra (2008), pp. 58–71
18. X. Xu, J.W. Grizzle, P. Tabuada, A.D. Ames, Correctness guarantees for the composition of lane keeping and adaptive cruise control. IEEE Trans. Autom. Sci. Eng. **PP**(99), 1–14 (2017)
19. M. Korda, D. Henrion, C.N. Jones, Convex computation of the maximum controlled invariant set for polynomial control systems. SIAM J. Control Optim. **52**(5), 2944–2969 (2014)
20. P. Tabuada, *Verification and Control of Hybrid Systems: A Symbolic Approach* (Springer Science & Business Media, New York, 2009)
21. A.D. Ames, X. Xu, J.W. Grizzle, P. Tabuada, Control barrier function based quadratic programs for safety critical systems. IEEE Trans. Autom. Control. **62**(8), 3861–3876 (2017)
22. E.S. Kim, M. Arcak, S.A. Seshia, A small gain theorem for parametric assume-guarantee contracts, in *Proceedings of HSCC* (ACM, New York, 2017), pp. 207–216
23. D.P. Bertsekas, Infinite time reachability of state-space regions by using feedback control. IEEE Trans. Autom. Control **17**(5), 604–613 (1972)
24. E.G. Gilbert, K. Tin Tan, Linear systems with state and control constraints: the theory and application of maximal output admissible sets. IEEE Trans. Autom. Control **36**(9), 1008–1020 (1991)
25. M. Rungger, P. Tabuada, Computing robust controlled invariant sets of linear systems. IEEE Trans. Autom. Control **62**(7), 3665–3670 (2017)
26. M. Herceg, M. Kvasnica, C.N. Jones, M. Morari, Multi-parametric toolbox 3.0, in *Proceedings of the European Control Conference* (2013), pp. 502–510

27. P. Nilsson, Correct-by-construction control synthesis for high-dimensional systems. Ph.D. thesis, University of Michigan, 2017
28. J. Liu, N. Ozay, Finite abstractions with robustness margins for temporal logic-based control synthesis. Nonlinear Anal. Hybrid Syst. **22**, 1–15 (2016)
29. CarSim (Mechanical Simulation). http://www.carsim.com/
30. PolySync. http://www.polysync.io/

Chapter 7
Reachable Set Estimation and Verification for Neural Network Models of Nonlinear Dynamic Systems

Weiming Xiang, Diego Manzanas Lopez, Patrick Musau, and Taylor T. Johnson

7.1 Introduction

Artificial neural networks have been widely used in machine learning and artificial intelligence systems. Applications include adaptive control [12, 15], pattern recognition [19, 26], game playing [27], autonomous vehicles [6], and many others. Neural networks are trained over finite amounts of input and output data, and are expected to be able to generalize to produce desirable outputs for given inputs even including previously unseen inputs. Though neural networks have been showing effectiveness and powerful ability in resolving complex problems, they are confined to systems that comply only to the lowest safety integrity levels since, most of the time, a neural network is viewed as a *black box* without effective methods to assure robustness or safety specifications for its outputs. For nonlinear dynamic systems whose models are difficult or even impossible to establish, using neural network models that are inherently derived from input and output data to approximate the nonlinear dynamics is an efficient and practical way. One standard employment of neural networks is to approximate the Nonlinear Autoregressive-Moving Average (NARMA) model which is a popular model for nonlinear dynamic systems. However, once the NARMA model in the form of neural networks is established, a problem naturally arises: *How to compute the reachable set of an NARMA model that is essentially expressed by neural networks and, based on that, how to verify properties of an NARMA model?* For computing or estimating the reachable set for a nonlinear system starting from an initial set and with an input set, the numbers of inputs and initial state that need to be checked are infinite, which is impossible only

W. Xiang · D. M. Lopez · P. Musau · T. T. Johnson (✉)
Department of Electrical Engineering and Computer Science, Vanderbilt University, Nashville, TN, USA
e-mail: weiming.xiang@vanderbilt.edu; diego.manzanas.lopez@vanderbilt.edu; patrick.musau@vanderbilt.edu; taylor.johnson@vanderbilt.edu

H. Yu et al. (eds.), *Safe, Autonomous and Intelligent Vehicles*,
Unmanned System Technologies, https://doi.org/10.1007/978-3-319-97301-2_7

by performing experiments. Moreover, it has been observed that neural networks can react in unexpected and incorrect ways to even slight perturbations of their inputs [28], which could result in unsafe systems. Hence, methods that are able to provide formal guarantees are in a great demand for verifying specifications or properties of systems involving neural networks. Verifying neural networks is a hard problem, even simple properties about them have been proven NP-complete problems [17]. The difficulties mainly come from the presence of activation functions and the complex structures, making neural networks large-scale, nonlinear, non-convex and thus incomprehensible to humans. Until now, only few results have been reported for verifying neural networks. The verification for feed-forward multi-layer neural networks is investigated based on *Satisfiability Modulo Theory* (SMT) in [14, 24]. In [23] an abstraction-refinement approach is proposed for verification of specific networks known as *Multi-Layer Perceptrons* (MLPs). In [17, 41], a specific kind of activation functions called *Rectified Linear Unit* (ReLU) is considered for the verification problem of neural networks. A simulation-based approach is developed in [39], which turns the reachable set estimation problem into a neural network maximal sensitivity computation problem that is described in terms of a chain of convex optimization problems. Additionally, some recent reachable set/state estimation results are reported for neural networks [30, 42, 44, 45, 47], these results that are based on Lyapunov functions analogous to stability [32–37, 43] and reachability analysis of dynamical systems [38, 40] certainly have potentials to be further extended to safety verification.

In this paper, we will use neural networks to represent the forward dynamics of the nonlinear systems that are in the form of NARMA models. Due to the non-convex and nonlinearity existing in the model and inspired by some simulation-based ideas for verification problems [2, 3, 9, 11], a simulation-based approach will be developed to estimate the reachable set of state responses generated from a NARMA model. The core step of the approach is the reachable set estimation for a class of feed-forward neural networks called Multi-Layer Perceptron (MLP). By discretizing the input space of an MLP into a finite-number of regularized cells, a layer-by-layer computation process is developed to establish an over-approximation of the output set for each individual cell. The union of output set of all cells is the reachable set estimation for the MLP with respect to a given input set. On the basis of the reachable set estimation method for MLPs, the reachable set over any finite-time interval for an NARMA model can be estimated in a recursive manner. Safety verification can be performed if an estimation for the reachable set of an NARMA model is established, by checking the existence of intersections between the estimated reachable set and unsafe regions.

The remainder of this paper is organized as follows. Neural network model of nonlinear systems, that is the NARMA model, is introduced in Sect. 7.2. The problem formulation is presented in Sect. 7.3. The main results, reachable set estimation for MLPs and NARMA models, are given in Sects. 7.4 and 7.5, respectively. An example for magnetic levitation systems is presented in Sect. 7.6. Conclusions are made in Sect. 7.7.

Notations: $\mathbb{R}$ denotes the field of real numbers, $\mathbb{R}^n$ stands for the vector space of all n-tuples of real numbers, $\mathbb{R}^{n\times n}$ is the space of $n \times n$ matrices with real entries. $\|\mathbf{x}\|_\infty$ stands for infinity norm for vector $\mathbf{x} \in \mathbb{R}^n$ defined as $\|\mathbf{x}\|_\infty = \max_{i=1,\ldots,n} |x_i|$. $\mathbf{A}^\top$ denotes the transpose of matrix $\mathbf{A}$. For a set $\mathcal{A}$, $|\mathcal{A}|$ denotes its cardinality.

7.2 Neural Network Models of Nonlinear Dynamic Systems

Neural networks are commonly used for data-driven modeling for nonlinear systems. One standard model to represent discrete-time nonlinear systems is the Nonlinear Autoregressive-Moving Average (NARMA) model. Given a discrete-time process with past states $\mathbf{x}(k), \mathbf{x}(k-1), \ldots, \mathbf{x}(k-d_x)$ and inputs $\mathbf{u}(k), \mathbf{u}(k-1), \ldots, \mathbf{u}(k-d_u)$, an NARMA model is in the form of

$$\mathbf{x}(k+1) = f\left(\mathbf{x}(k), \mathbf{x}(k-1), \ldots, \mathbf{x}(k-d_x), \mathbf{u}(k), \mathbf{u}(k-1), \ldots, \mathbf{u}(k-d_u)\right), \tag{7.1}$$

where the nonlinear function $f(\cdot)$ needs to be approximated by training neural networks. The initial state of NARMA model (7.1) is $\mathbf{x}(0), \ldots, \mathbf{x}(d_x)$, which is assumed to be in set $\mathcal{X}_0 \times \cdots \times \mathcal{X}_{d_x}$, and the input set is $\mathcal{U}$. We assume that the initial state $\{\mathbf{x}(0), \ldots, \mathbf{x}(d_x)\} \in \mathcal{X}_0 \times \cdots \times \mathcal{X}_{d_x}$ and input satisfies $\mathbf{u}(k) \in \mathcal{U}, \forall k \in \mathbb{N}$.

A neural network consists of a number of interconnected neurons. Each neuron is a simple processing element that responds to the weighted inputs it received from other neurons. In this paper, we consider the most popular and general feed-forward neural network, MLP. Generally, an MLP consists of three typical classes of layers: An input layer, that serves to pass the input vector to the network, hidden layers of computation neurons, and an output layer composed of at least a computation neuron to produce the output vector.

The action of a neuron depends on its activation function, which is described as

$$y_i = h\left(\sum\nolimits_{j=1}^{n} \omega_{ij} v_j + \theta_i\right), \tag{7.2}$$

where v_j is the jth input of the ith neuron, ω_{ij} is the weight from the jth input to the ith neuron, θ_i is called the bias of the ith neuron, y_i is the output of the ith neuron, $h(\cdot)$ is the activation function. The activation function is generally a nonlinear function describing the reaction of ith neuron with inputs v_j, $j = 1, \cdots, n$. Typical activation functions include Rectified Linear Unit (ReLU), logistic, tanh, exponential linear unit, linear functions, etc. In this work, our approach aims at dealing with activation functions regardless of their specific forms, only the following monotonic assumption needs to be satisfied.

Assumption 1 *For any $v_1 \le v_2$, the activation function satisfies $h(v_1) \le h(v_2)$.*

Assumption 1 is a common property that can be satisfied by a variety of activation functions. For example, it is easy to verify that the most commonly used such as logistic, tanh, ReLU, all satisfy Assumption 1.

An MLP has multiple layers, each layer ℓ, $1 \leq \ell \leq L$, has $n^{[\ell]}$ neurons. In particular, layer $\ell = 0$ is used to denote the input layer and $n^{[0]}$ stands for the number of inputs in the rest of this paper, and of course, $n^{[L]}$ stands for the last layer, that is the output layer. For a neuron i, $1 \leq i \leq n^{[\ell]}$ in layer ℓ, the corresponding input vector is denoted by $\mathbf{v}^{[\ell]}$ and the weight matrix is

$$\mathbf{W}^{[\ell]} = \left[\boldsymbol{\omega}_1^{[\ell]}, \ldots, \boldsymbol{\omega}_{n^{[\ell]}}^{[\ell]}\right]^\top, \tag{7.3}$$

where $\boldsymbol{\omega}_i^{[\ell]}$ is the weight vector. The bias vector for layer ℓ is

$$\boldsymbol{\theta}^{[\ell]} = \left[\theta_1^{[\ell]}, \ldots, \theta_{n^{[\ell]}}^{[\ell]}\right]^\top$$

The output vector of layer ℓ can be expressed as

$$\mathbf{y}^{[\ell]} = h_\ell(\mathbf{W}^{[\ell]}\mathbf{v}^{[\ell]} + \boldsymbol{\theta}^{[\ell]}), \tag{7.4}$$

where $h_\ell(\cdot)$ is the activation function for layer ℓ.

For an MLP, the output of $\ell - 1$ layer is the input of ℓ layer, and the mapping from the input of input layer $\mathbf{v}^{[0]}$ to the output of output layer $\mathbf{y}^{[L]}$ stands for the input–output relation of the MLP, denoted by

$$\mathbf{y}^{[L]} = H(\mathbf{v}^{[0]}), \tag{7.5}$$

where $H(\cdot) \triangleq h_L \circ h_{L-1} \circ \cdots \circ h_1(\cdot)$.

According to the *Universal Approximation Theorem* [13], it guarantees that, in principle, such an MLP in (7.5), namely the function $F(\cdot)$, is able to approximate any nonlinear real-valued function. To use MLP (7.5) to approximate NARMA model (7.1), we can let the input of (7.5) as

$$\mathbf{v}^{[0]} = [\mathbf{x}^\top(k), \mathbf{x}^\top(k-1), \ldots, \mathbf{x}^\top(k-d_x), \mathbf{u}^\top(k), \mathbf{u}^\top(k-1), \ldots, \mathbf{u}^\top(k-d_u)]^\top, \tag{7.6}$$

and output as

$$\mathbf{y}^{[L]} = \mathbf{x}(k+1). \tag{7.7}$$

With the input and output data of original nonlinear systems, an approximation of NARMA model (7.1) can be obtained by standard feed-forward neural network training process. Despite the impressive ability of approximating nonlinear functions, much complexities represent in predicting the output behaviors of MLP (7.5) as well as NARMA model (7.1) because of the nonlinearity and non-convexity of

MLPs. In most of the real applications, an MLP is usually viewed as a *black box* to generate a desirable output with respect to a given input. However, regarding property verifications such as the safety verification, it has been observed that even a well-trained neural network can react in unexpected and incorrect ways to even slight perturbations of their inputs, which could result in unsafe systems. Thus, to validate the neural network NARMA model for a nonlinear dynamics, it is necessary to compute the reachable set estimation of the model, which is able to cover all possible values of output, to assure that the state trajectories of the model will not attain unreasonable values that is inadmissible for the original system. It is also necessary to estimate all possible values of state for safety verification of a neural network NARMA model.

7.3 Problem Formulation

Consider initial set $\mathcal{X}_0 \times \cdots \times \mathcal{X}_{d_x}$ and input set $\mathcal{U}$, the reachable set of NARMA model in the form of (7.1) is defined as follows.

Definition 1 Given an NARMA model in the form of (7.1) with initial set $\mathcal{X}_0 \times \cdots \times \mathcal{X}_{d_x}$ and input set $\mathcal{U}$, the reachable set at a time instant k is:

$$\mathcal{X}_k \triangleq \{\mathbf{x}(k) \mid \mathbf{x}(k) \text{ satisfies (7.1) and } \{\mathbf{x}(0), \cdots, \mathbf{x}(d_x)\} \in \mathcal{X}_0 \times \ldots \times \mathcal{X}_{d_x}, \mathbf{u}(k) \in \mathcal{U},\ \forall k \in \mathbb{N}\}, \quad (7.8)$$

and the reachable set over time interval $[0, k_f]$ is defined by

$$\mathcal{X}_{[0,k_f]} = \bigcup_{s=0}^{k_f} \mathcal{X}_s. \quad (7.9)$$

Since MLPs are often large, nonlinear, and non-convex, it is extremely difficult to compute the exact reachable set $\mathcal{X}_k$ and $\mathcal{X}_{[0,k_f]}$ for an NARMA model with MLPs. Rather than directly computing the exact output reachable set for an NARMA model, a more practical and feasible way is to derive an over-approximation of $\mathcal{X}_k$, which is called reachable set estimation.

Definition 2 A set $\tilde{\mathcal{X}}_k$ is called a reachable set estimation of NARMA model (7.1) at time instant k, if $\mathcal{X}_k \subseteq \tilde{\mathcal{X}}_k$ holds and, moreover, $\tilde{\mathcal{X}}_{[0,k_f]} = \bigcup_{s=0}^{k} \tilde{\mathcal{X}}_s$ is a reachable set estimation for NARMA model (7.1) over time interval $[0, k_f]$.

Based on Definition 2, the problem of reachable set estimation for an NARMA model is given as below.

Problem 1 How does one find the set $\tilde{\mathcal{X}}_k$ such that $\mathcal{X}_k \subseteq \tilde{\mathcal{X}}_k$, given a bounded initial set $\mathcal{X}_0 \times \cdots \times \mathcal{X}_{d_x}$ and an input set $\mathcal{U}$ and an NARMA model (7.1)?

In this work, we will focus on the safety verification for NARMA models. The safety specification for output is expressed by a set defined in the state space, describing the safety requirement.

Definition 3 Safety specification $\mathcal{S}$ formalizes the safety requirements for state $\mathbf{x}(k)$ of NARMA model (7.1), and is a predicate over state $\mathbf{x}$ of NARMA model (7.1). The NARMA model (7.1) is safe over time interval $[0, k_f]$ if and only if the following condition is satisfied:

$$\mathcal{X}_{[0,k_f]} \cap \neg\mathcal{S} = \emptyset, \tag{7.10}$$

where $\neg$ is the symbol for logical negation.

Therefore, the safety verification problem for NARMA models is stated as follows.

Problem 2 How can the safety requirement in (7.10) be verified given an NARMA model (7.1) with a bounded initial set $\mathcal{X}_0 \times \cdots \times \mathcal{X}_{d_x}$ and an input set $\mathcal{U}$ and a safety specification $\mathcal{S}$?

Before ending this section, a lemma is presented to show that the safety verification of an MLP can be relaxed by checking with the over-approximation of output reachable set.

Lemma 1 *Consider an NARMA model (7.1) and a safety specification $\mathcal{S}$, the NARMA model is safe in time interval $[0, k_f]$ if the following condition is satisfied*

$$\tilde{\mathcal{X}}_{[0,k_f]} \cap \neg\mathcal{S} = \emptyset, \tag{7.11}$$

where $\mathcal{X}_{[0,k_f]} \subseteq \tilde{\mathcal{X}}_{[0,k_f]}$.

Proof Since $\mathcal{X}_{[0,k_f]} \subseteq \tilde{\mathcal{X}}_{[0,k_f]}$, condition (7.11) directly leads to $\mathcal{X}_{[0,k_f]} \cap \neg\mathcal{S} = \emptyset$. The proof is complete.

Lemma 1 implies that it is sufficient to use the estimated reachable set for the safety verification of an NARMA model, thus the solution of Problem 1 is also the key to solve Problem 2.

7.4 Reachable Set Estimation for MLPs

As (7.5)–(7.7) in previous section, the state of an NARMA model is computed through an MLP recursively. Therefore, the first step for the reachable set estimation for an NARMA model is to estimate the output set of MLP (7.5).

Given an MLP $\mathbf{y}^{[L]} = H(\mathbf{v}^{[0]})$ with a bounded input set $\mathcal{V}$, the problem is how to compute a set $\mathcal{Y}$ as below:

$$\mathcal{Y} \triangleq \{\mathbf{y}^{[L]} \mid \mathbf{y}^{[L]} = H(\mathbf{v}^{[0]}),\ \mathbf{v}^{[0]} \in \mathcal{V} \subset \mathbb{R}^n\}. \tag{7.12}$$

Due to the complex structure and nonlinearities in activation functions, the estimation of output reachable set of MLP represents much difficulties if only using analytical methods. One possible way to circumvent those difficulties is to employ the information produced by a finite number of simulations.

Definition 4 Given a set $\mathcal{V} \subset \mathbb{R}^n$, a finite collection of sets $\mathscr{P} = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_N\}$ is said to be a partition of $\mathcal{V}$ if (1) $\mathcal{P}_i \subseteq \mathcal{V}$; (2) $\text{int}(\mathcal{P}_i) \cup \text{int}(\mathcal{P}_j) = \emptyset$; (3) $\mathcal{V} \subseteq \bigcup_{i=1}^{N} \mathcal{P}_i, \forall i \in \{1, \ldots, N\}$. Each element $\mathcal{P}_i$ of partition $\mathscr{P}$ is called a cell.

In this paper, we use cells defined by intervals which are given as follows: For any bounded set $\mathcal{V} \subset \mathbb{R}^n$, we have $\mathcal{V} \subseteq \bar{\mathcal{V}}$,where $\bar{\mathcal{V}} = \{\mathbf{v} \in \mathbb{R}^n \mid \underline{\mathbf{v}} \leq \mathbf{v} \leq \bar{\mathbf{v}}\}$, in which $\underline{\mathbf{v}}$ and $\bar{\mathbf{v}}$ are defined as the lower and upper bounds of elements of $\mathbf{v}$ in $\mathcal{V}$ as $\underline{\mathbf{v}} = [\inf_{\mathbf{v}\in\mathcal{V}}(v_1), \ldots, \inf_{\mathbf{v}\in\mathcal{V}}(v_n)]^\top$ and $\bar{\mathbf{v}} = [\sup_{\mathbf{v}\in\mathcal{V}}(v_1), \ldots, \sup_{\mathbf{v}\in\mathcal{V}}(v_n)]^\top$, respectively. Then, we are able to partition interval $\mathcal{I}_i = [\inf_{\mathbf{v}\in\mathcal{V}}(v_i), \sup_{\mathbf{v}\in\mathcal{V}}(v_i)]$, $i \in \{1, \ldots, n\}$ into M_i segments as $\mathcal{I}_{i,1} = [v_{i,0}, v_{i,1}]$, $\mathcal{I}_{i,2} = [v_{i,1}, v_{i,2}]$, $\ldots$, $\mathcal{I}_{i,M_i} = [v_{i,M_i-1}, v_{i,M_i}]$, where $v_{i,0} = \inf_{\mathbf{v}\in\mathcal{V}}(v_i)$, $v_{i,M_i} = \sup_{\mathbf{v}\in\mathcal{V}}(v_i)$ and $v_{i,n} = v_{i,0} + \frac{m(v_{i,M_i} - v_{i,0})}{M_i}$, $m \in \{0, 1, \ldots, M_i\}$. The cells then can be constructed as $\mathcal{P}_i = \mathcal{I}_{1,m_1} \times \cdots \times \mathcal{I}_{n,m_n}, i \in \{1, 2, \ldots, \prod_{s=1}^{n} M_s\}, \{m_1, \ldots, m_n\} \in \{1, \ldots, M_1\} \times \cdots \times \{1, \ldots, M_n\}$. To remove redundant cells, we have to check if the cell has empty intersection with $\mathcal{V}$. Cell $\mathcal{P}_i$ should be removed if $\mathcal{P}_i \cap \mathcal{V} = \emptyset$. The cell construction process is summarized by `cell` function in Algorithm 1.

Algorithm 1 Partition an input set

Require: Set $\mathcal{V}$, partition numbers $M_i, i \in \{1, \ldots, n\}$
Ensure: Partition $\mathscr{P} = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_N\}$
function CELL $\mathcal{V}, M_i, i \in \{1, \ldots, n\}$
  $v_{i,0} \leftarrow \inf_{\mathbf{v}\in\mathcal{V}}(v_i)$, $v_{i,M_i} \leftarrow \sup_{\mathbf{v}\in\mathcal{V}}(v_i)$
  for $i = 1 : 1 : n$ **do**
    for $j = 1 : 1 : M_i$ **do**
      $v_{i,j} \leftarrow v_{i,0} + \frac{j(v_{i,M_i} - v_{i,0})}{M_i}$
      $\mathcal{I}_{i,j} \leftarrow [v_{i,j-1}, v_{i,j}]$
    end for
  end for
  $\mathcal{P}_i \leftarrow \mathcal{I}_{1,m_1} \times \cdots \times \mathcal{I}_{n,m_n}, \{m_1, \ldots, m_n\} \in \{1, \ldots, M_1\} \times \cdots \times \{1, \ldots, M_n\}$
  if $\mathcal{P}_i \cap \mathcal{V} = \emptyset$ **then**
    Remove $\mathcal{P}_i$
  end if
  return $\mathscr{P} = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_N\}$
end function

With the cells constructed by `cell` function, the next step is to develop a function that is able to estimate the output reachable set for each individual cell as the input to the MLP. A layer-by-layer approach is developed.

Theorem 1 *For a single layer* $\mathbf{y} = h(\mathbf{W}\mathbf{v} + \boldsymbol{\theta})$*, if the input set is a cell described by* $\mathcal{I}_1 \times \cdots \times \mathcal{I}_{n_v}$ *where* $\mathcal{I}_i = [\underline{v}_i, \bar{v}_i]$*,* $i \in \{1, \ldots, n_v\}$*, the output set can be over-approximated by a cell in the expression of intervals* $\mathcal{I}_1 \times \cdots \times \mathcal{I}_{n_y}$*, where* $\mathcal{I}_i$*,*

$i \in \{1, \ldots, n_y\}$ *can be computed by*

$$\mathcal{I}_i = [h(\underline{z}_i + \theta_i),\ h(\bar{z}_i + \theta_i)], \tag{7.13}$$

where $\underline{z}_i = \sum_{j=1}^{n_v} \underline{g}_{ij}$, $\bar{z}_i = \sum_{j=1}^{n_v} \bar{g}_{ij}$ *with* $\underline{g}_{ij}$ *and* $\bar{g}_{ij}$ *defined by*

$$\underline{g}_{ij} = \begin{cases} \omega_{ij}\underline{v}_j & \omega_{ij} \geq 0 \\ \omega_{ij}\bar{v}_j & \omega_{ij} < 0 \end{cases}, \quad \bar{g}_{ij} = \begin{cases} \omega_{ij}\bar{v}_j & \omega_{ij} \geq 0 \\ \omega_{ij}\underline{v}_j & \omega_{ij} < 0 \end{cases}. \tag{7.14}$$

Proof By (7.14), one can obtain that

$$\underline{z}_i = \min_{\mathbf{v} \in \mathcal{I}_1 \times \cdots \times \mathcal{I}_{n_v}} \left(\sum_{j=1}^{n_v} \omega_{ij} v_j \right), \tag{7.15}$$

$$\bar{z}_i = \max_{\mathbf{v} \in \mathcal{I}_1 \times \cdots \times \mathcal{I}_{n_v}} \left(\sum_{j=1}^{n_v} \omega_{ij} v_j \right). \tag{7.16}$$

Consider neuron i, its output is $y_i = h\left(\sum_{j=1}^{n_v} \omega_{ij} v_j + \theta_i\right)$. Under Assumption 1, we can conclude that

$$\min_{\mathbf{v} \in \mathcal{I}_1 \times \cdots \times \mathcal{I}_{n_v}} \left(h\left(\sum_{j=1}^{n_v} \omega_{ij} v_j + \theta_i \right) \right) = h(\underline{z}_i + \theta_i), \tag{7.17}$$

$$\max_{\mathbf{v} \in \mathcal{I}_1 \times \cdots \times \mathcal{I}_{n_v}} \left(h\left(\sum_{j=1}^{n_v} \omega_{ij} v_j + \theta_i \right) \right) = h(\bar{z}_i + \theta_i). \tag{7.18}$$

Thus, it leads to

$$y_i \in [h(\underline{z}_i + \theta_i),\ h(\bar{z}_i + \theta_i)] = \mathcal{I}_i. \tag{7.19}$$

and therefore, $\mathbf{y} \in \mathcal{I}_1 \times \cdots \times \mathcal{I}_{n_y}$. The proof is complete.

Theorem 1 not only demonstrates the output set of one single layer can be approximated by a cell if the input set is a cell, it also gives out an efficient way to calculate the cell, namely by (7.13) and (7.14). For multi-layer neural networks, Theorem 1 plays the key role for the layer-by-layer approach. For an MLP which essentially has $\mathbf{v}^{[\ell]} = \mathbf{y}^{[\ell-1]}$, $\ell = 1, \ldots, L$, if the input set is a set of cells, Theorem 1 assures the input set of every layer can be over-approximated by a set of cells, which can be computed by (7.13) and (7.14) layer-by-layer. The output set of layer L is thus an over-approximation of reachable set of the MLP.

Function `reachMLP` given in Algorithm 2 illustrates the layer-by-layer method for reachable set estimation for an MLP.

Example 1 An MLP with 2 inputs, 2 outputs, and 1 hidden layer consisting of 5 neurons is considered. The activation function for the hidden layer is chosen as

Algorithm 2 Reachable set estimation for MLP

Require: Weight matrices $\mathbf{W}^{[\ell]}$, bias $\boldsymbol{\theta}^{[\ell]}$, $\ell \in \{1, \ldots, L\}$, set $\mathcal{V}$, partition numbers M_i, $i \in \{1, \ldots, n\}$
Ensure: Reachable set estimation $\tilde{\mathcal{Y}}$.
function REACHMLP $\mathbf{W}^{[\ell]}$, $\boldsymbol{\theta}^{[\ell]}$, $\ell \in \{1, \ldots, L\}$, $\mathcal{V}$, $M_i, i \in \{1, \ldots, n\}$
  $\mathscr{P} \leftarrow \text{cell}(\mathcal{V}, M_i, i \in \{1, \ldots, n\})$
  for $p = 1 : 1 : |\mathscr{P}|$ **do**
    $\mathcal{I}_1^{[1]} \times \cdots \times \mathcal{I}_{n^{[1]}}^{[1]} \leftarrow \mathcal{P}_p$
    for $j = 1 : 1 : L$ **do**
      for $i = 1 : 1 : n^{[j]}$ **do**
        $\underline{g}_{ij} \leftarrow \begin{cases} \omega_{ij}\underline{v}_j & \omega_{ij} \geq 0 \\ \omega_{ij}\bar{v}_j & \omega_{ij} < 0 \end{cases}$, $\bar{g}_{ij} \leftarrow \begin{cases} \omega_{ij}\bar{v}_j & \omega_{ij} \geq 0 \\ \omega_{ij}\underline{v}_j & \omega_{ij} < 0 \end{cases}$
        $\underline{z}_i \leftarrow \sum_{j=1}^{n_v} \underline{g}_{ij}$, $\bar{z}_i \leftarrow \sum_{j=1}^{n_v} \bar{g}_{ij}$
        $\mathcal{I}_i^{[j+1]} \leftarrow [h_j(\underline{z}_i + \theta_i),\ h_j(\bar{z}_i + \theta_i)]$
      end for
    end for
    $\tilde{\mathcal{Y}}_p \leftarrow \mathcal{I}_1^{[L]} \times \cdots \times \mathcal{I}_{n^{[L]}}^{[L]}$
  end for
  $\tilde{\mathcal{Y}} \leftarrow \bigcup_{p=1}^{|\mathscr{P}|} \tilde{\mathcal{Y}}_p$
  return $\tilde{\mathcal{Y}}$
end function

`tanh` function and `purelin` function is for the output layer. The weight matrices and bias vectors are given as below:

$$\mathbf{W}^{[1]} = \begin{bmatrix} 0.2075 & -0.7128 \\ 0.2569 & 0.7357 \\ -0.6136 & -0.3624 \\ 0.0111 & 0.1393 \\ -1.0872 & -0.2872 \end{bmatrix},\ \boldsymbol{\theta}^{[1]} = \begin{bmatrix} -1.1829 \\ -0.6458 \\ 0.4619 \\ -0.0499 \\ 0.3405 \end{bmatrix},$$

$$\mathbf{W}^{[2]} = \begin{bmatrix} -0.5618 & -0.0851 & -0.4529 & -0.8230 & 0.5651 \\ 0.7861 & -0.0855 & 1.1041 & 1.6385 & -0.3859 \end{bmatrix},\ \boldsymbol{\theta}^{[2]} = \begin{bmatrix} -0.2489 \\ -0.1480 \end{bmatrix}.$$

In this example, the input set is considered as below:

$$\mathcal{V} = \{\mathbf{v} \in \mathbb{R}^2 \mid \|\mathbf{v}\|_\infty \leq 1\}.$$

Then, the partition numbers are chosen to be $M_1 = M_2 = 20$, which means there are in total 400 cells, $\mathcal{P}_i$, $i \in \{1, \ldots, 400\}$, produced for the reachable set estimation.

Executing function `reachMLP` for input set $\mathcal{V}$, the estimated output reachable set is given in Fig. 7.1, in which it can be seen that 400 reachtubes are obtained and the union of them is the over-approximation of reachable set.

Moreover, we choose a different partition number discretizing state space to show how the choice of partitioning input set affects the estimation outcome.

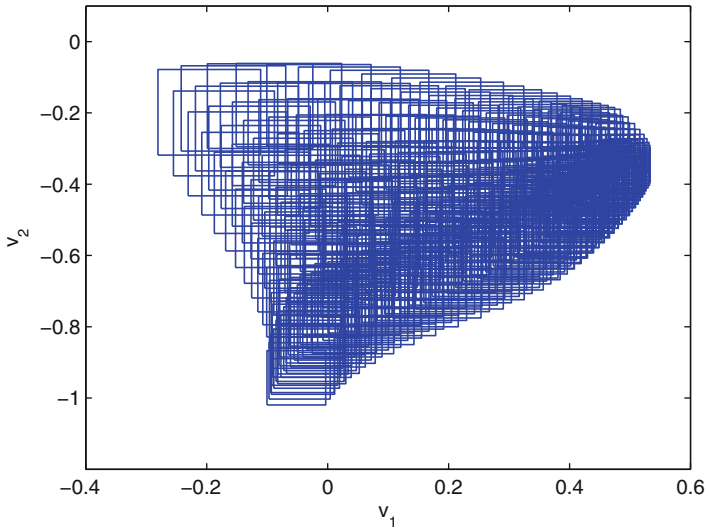


Fig. 7.1 Output reachable set estimation with input set $\mathcal{V} = \{\mathbf{v} \in \mathbb{R}^2 \mid \|v\|_\infty \leq 1\}$ and partition number $M_1 = M_2 = 20$. 400 reachtubes are computed for the reachable set estimation of the MLP

Explicitly, larger partition numbers will produce more cells and generate preciser approximations of input sets and are supposed to achieve preciser estimations. Here, we adjust the partition numbers from 10 to 50 for the different estimation results. With this finer discretization, more computation efforts are required for running function `reachMLP`, but a tighter estimation for the reachable set can be obtained. The reachable set estimations are shown in Fig. 7.2. Comparing those results, it can be observed that larger partition numbers can lead to a better estimation result at the expense of more computation efforts. The computation time and number of reachtubes with different partition numbers are listed in Table 7.1.

To validate the result, 5000 random outputs are generated, it is clear to see in Fig. 7.2 that all the outputs are included in the estimated reachable set, showing the effectiveness of the proposed approach.

7.5 Reachable Set Estimation for NARMA Models

Based on the developed approach for reachable set estimation for MLP, this section will extend the result to NARMA models. As in previous sections, NARMA models employ MLP to approximate the nonlinear relation between $\mathbf{x}(k), \mathbf{x}(k-1), \ldots, \mathbf{x}(k-d_x), \mathbf{u}(k), \mathbf{u}(k-1), \ldots, \mathbf{u}(k-d_u)$ and state $\mathbf{x}(k+1)$. Without loss of generality, we assume $d_x \geq d_u$, thus the model is valid for any $k \geq d_x$. Thus, with

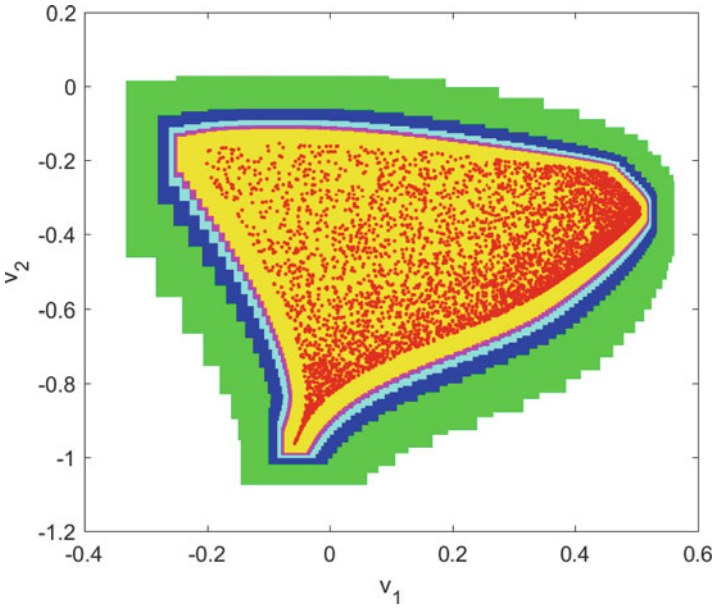


Fig. 7.2 Output reachable set estimation with input set $\mathcal{V} = \{\mathbf{v} \in \mathbb{R}^2 \mid \|v\|_\infty \leq 1\}$ and partition number $M_1 = M_2 = 10$ (green + blue + cyan + magenta + yellow), $M_1 = M_2 = 20$ (blue + cyan + magenta + yellow), $M_1 = M_2 = 30$ (cyan + magenta + yellow), $M_1 = M_2 = 40$ (magenta + yellow), and $M_1 = M_2 = 50$ (yellow). It can be observed that tighter estimations can be obtained with larger partition numbers. 5000 random outputs (red spots) from input set are all located in the estimated reachable set

Table 7.1 Computation time and number of reachtubes with different partition numbers

Partition number	Computation time	Number of reachtubes
$M_1 = M_2 = 10$	0.062304 s	100
$M_1 = M_2 = 20$	0.074726 s	400
$M_1 = M_2 = 30$	0.142574 s	900
$M_1 = M_2 = 40$	0.251087 s	1600
$M_1 = M_2 = 50$	0.382729 s	2500

the aid of reachable set estimation results for MLP, the reachable set of NARMA (7.1) at time instant k can be estimated by recursively using functions `cell` and `reachMLP` for $k - d_x$ times.

Since the reachable sets $\mathcal{X}_k$, $k \in \{0, 1, \ldots, d_x\}$, are given as initial set, let us start with $k = d_x + 1$. In the employment of function `reachMLP` with input of $\mathcal{X}_0 \times \ldots \times \mathcal{X}_{d_x}$ and $\mathcal{U}^{d_u}$, $\tilde{\mathcal{X}}_{d_x+1} = \text{reachMLP}(\mathbf{W}^{[\ell]}, \boldsymbol{\theta}^{[\ell]}, \ell \in \{1, \ldots, L\}, \mathcal{X}_0 \times \cdots \times \mathcal{X}_{d_x}, M_i, i \in \{1, \ldots, n^{[0]}\})$ is an over-approximation of $\mathcal{X}_{d_x+1}$, namely $\mathcal{X}_{d_x+1} \subseteq \tilde{\mathcal{X}}_{d_x+1}$. Then, repeating using function `reachMLP` from $d_x + 1$ to k_f, we can obtain an over-approximation of $\mathcal{X}_k$, $k = d_x + 1, \ldots, k_f$, and $\mathcal{X}_{[0,k_f]}$.

Proposition 1 *Consider NARMA model (7.1) with initial set $\mathcal{X}_0 \times \cdots \times \mathcal{X}_{d_x}$ and input set $\mathcal{U}$, the reachable set $\mathcal{X}_k$, $k > d_x$ can be recursively over-approximated by*

$$\tilde{\mathcal{X}}_k = \text{reachMLP}(\mathbf{W}^{[\ell]}, \boldsymbol{\theta}^{[\ell]}, \ell \in \{1, \ldots, L\}, \tilde{\mathcal{X}}_{k-d_x-1} \times \cdots \times \tilde{\mathcal{X}}_{k-1} \times \mathcal{U}^{d_u}, M_i, i \in \{1, \ldots, n^{[0]}\}), \tag{7.20}$$

where $\tilde{\mathcal{X}}_k = \mathcal{X}_k$, $k \in \{0, \ldots, d_x\}$. *The reachable set over time interval* $[0, k_f]$ *can be estimated by*

$$\tilde{\mathcal{X}}_{[0,k_f]} = \bigcup_{s=0}^{k_f} \tilde{\mathcal{X}}_s. \tag{7.21}$$

The iterative algorithm for estimating reachable set $\mathcal{X}_k$ and $\mathcal{X}_{k_f}$ is summarized as function `reachNARMA` in Algorithm 3.

Algorithm 3 Reachable set estimation for NARMA model

Require: Weight matrices $\mathcal{W}^{[\ell]}$, bias $\boldsymbol{\theta}^{[\ell]}$, $\ell = 1, \ldots, L$, initial set $\mathcal{X}_0 \times \ldots \times \mathcal{X}_{d_x}$, input set $\mathcal{U}$, partition numbers $M_i, i \in \{1, \ldots, n^{[0]}\}$

Ensure: Reachable set estimation $\mathcal{X}_k$, $\mathcal{X}_{[0,k_f]}$.

1: **function** REACHNARMA $\mathcal{W}^{[\ell]}, \boldsymbol{\theta}^{[\ell]}, \ell = 1, \ldots, L, \mathcal{X}_0 \times \ldots \times \mathcal{X}_{d_x}, \mathcal{U}, M_i, i \in \{1, \ldots, n^{[0]}\}$
2: **for** $k = d_u + 1 : 1 : k_f$ **do**
3: $\mathcal{V} \leftarrow \mathcal{X}_{k-d_u-1} \times \ldots \times \mathcal{X}_{k-1} \times \mathcal{U}^{d_u}$
4: $\mathcal{X}_k \leftarrow \text{reachMLF}(\mathcal{W}^{[\ell]}, \boldsymbol{\theta}^{[\ell]}, \ell = 1, \ldots, L, \mathcal{V}, M_i, i \in \{1, \ldots, n^{[0]}\}$.
5: **end for**
6: $\mathcal{X}_{[0,k_f]} \leftarrow \bigcup_{s=0}^{k_f} \mathcal{X}_s$
7: **return** $\mathcal{X}_k, k = 0, 1 \ldots, k_f, \mathcal{X}_{[0,k_f]}$
8: **end function**

Function `reachNARMA` is sufficient to solve the reachable set estimation problem for an NARMA model, that is Problem 1. Then, we can move forward to Problem 2, the safety verification problem for an NARMA model with a given safety specification $\mathcal{S}$ over a finite interval $[0, k_f]$, with the aid of estimated reachable set $\mathcal{X}_{[0,k_f]}$. Given a safety specification $\mathcal{S}$, the empty intersection between over-approximation $\tilde{\mathcal{X}}_{[0,k_f]}$ and $\neg\mathcal{S}$, namely $\tilde{\mathcal{X}}_{[0,k_f]} \cap \neg\mathcal{S} = \emptyset$, naturally leads to $\mathcal{X}_{[0,k_f]} \cap \neg\mathcal{S} = \emptyset$ due to $\mathcal{X}_{[0,k_f]} \subseteq \tilde{\mathcal{X}}_{[0,k_f]}$. The safety verification result is summarized by the following proposition.

Proposition 2 *Consider NARMA model (7.1) with initial set* $\mathcal{X}_0 \times \cdots \times \mathcal{X}_{d_x}$, *input set* $\mathcal{U}$, *and a safety specification* $\mathcal{S}$, *the NARMA model (7.1) is safe in interval* $[0, k_f]$, *if* $\tilde{\mathcal{X}}_{[0,k_f]} \cap \neg\mathcal{S} = \emptyset$, *where* $\tilde{\mathcal{X}}_{[0,k_f]} = \text{reachNARMA}(\mathcal{W}^{[\ell]}, \boldsymbol{\theta}^{[\ell]}, \ell = 1, \ldots, L, \mathcal{X}_0 \times \cdots \times \mathcal{X}_{d_x}, \mathcal{U}, M_i, i \in \{1, \ldots, n^{[0]}\})$ *obtained by Algorithm 3.*

Function `verifyNARMA` is developed based on Proposition 2 for Problem 2, the safety verification problem for NARMA model. If function `verifyNARMA` returns SAFE, then the NARMA model is safe. If it returns UNCERTAIN, caused by the fact $\tilde{\mathcal{X}}_{[0,k_f]}$, that means the safety property is unclear for this case.

A numerical example is provided to show the effectiveness of our developed approach.

Example 2 In this example, we consider an NARMA model as below:

$$\mathbf{x}(k+1) = f(\mathbf{x}(k), \mathbf{u}(k)), \tag{7.22}$$

where $\mathbf{x}(k), \mathbf{u}(k) \in \mathbb{R}$. We use an MLP with 2 inputs, 1 output and 1 hidden layer consisting of 5 neurons to approximate f with weight matrices and bias vectors below:

$$\mathbf{W}^{[1]} = \begin{bmatrix} 0.1129 & 0.4944 \\ 2.2371 & 0.4389 \\ -1.1863 & -0.7365 \\ 0.2965 & 0.3055 \\ -0.6697 & 0.5136 \end{bmatrix}, \ \boldsymbol{\theta}^{[1]} = \begin{bmatrix} -13.8871 \\ -8.2629 \\ 5.8137 \\ -3.2035 \\ -0.6697 \end{bmatrix},$$

$$\mathbf{W}^{[2]} = \begin{bmatrix} -3.3067 & 1.3905 & -0.6422 & 2.5221 & 1.8242 \end{bmatrix}, \ \boldsymbol{\theta}^{[2]} = \begin{bmatrix} 5.8230 \end{bmatrix}$$

Algorithm 4 Safety verification for NARMA model

Require: Weight matrices $\mathcal{W}^{[\ell]}$, bias $\boldsymbol{\theta}^{[\ell]}$, $\ell = 1, \ldots, L$, initial set $\mathcal{X}_0 \times \ldots \times \mathcal{X}_{d_x}$, input set $\mathcal{U}$, partition numbers $M_i, i \in \{1, \ldots, n^{[0]}\}$, safety specification $\mathcal{S}$
Ensure: SAFE or UNCERTAIN.
1: **function** VERIFYNARMA $\mathcal{W}^{[\ell]}$, $\boldsymbol{\theta}^{[\ell]}$, $\ell = 1, \ldots, L$, $\mathcal{X}_0 \times \ldots \times \mathcal{X}_{d_x}$, $\mathcal{U}$, $M_i, i \in \{1, \ldots, n^{[0]}\}$, $\mathcal{S}$
2: $\mathcal{X}_{[0,k_f]} \leftarrow$ reachNARMA($\mathcal{W}^{[\ell]}, \boldsymbol{\theta}^{[\ell]}, \ell = 1, \ldots, L, \mathcal{X}_0 \times \ldots \times \mathcal{X}_{d_x}, \mathcal{U}, M_i, i \in \{1, \ldots, n^{[0]}\}$)
3: **if** $\mathcal{X}_{[0,k_f]} \cap \mathcal{S} = \emptyset$ **then**
4: **return** SAFE
5: **else**
6: **return** UNCERTAIN
7: **end if**
8: **end function**

The activation function for the hidden layer is to choose `tanh` function and `purelin` function is for the output layer. The initial set and input set are given by the following set

$$\mathcal{X}_0 = \{\mathbf{x}(0) \in \mathbb{R} \mid -0.2 \leq \mathbf{x}(0) \leq 0.2\}, \tag{7.23}$$

$$\mathcal{U} = \{\mathbf{u}(k) \in \mathbb{R} \mid 0.8 \leq \mathbf{u}(k) \leq 1.2, \ \forall k \in \mathbb{N}\}. \tag{7.24}$$

We set the partition numbers to be $M_1 = M_2 = 10$, where M_1 is for input $\mathbf{u}$ and M_2 is for state $\mathbf{x}$. The time horizon for the reachable set estimation is set to be $[0, 50]$. Using function `reachNARMA`, the reachable set can be estimated, which is shown in Fig. 7.3. To show the effectiveness of our proposed approach, we randomly generate 100 state trajectories that are all within the estimated reachable set.

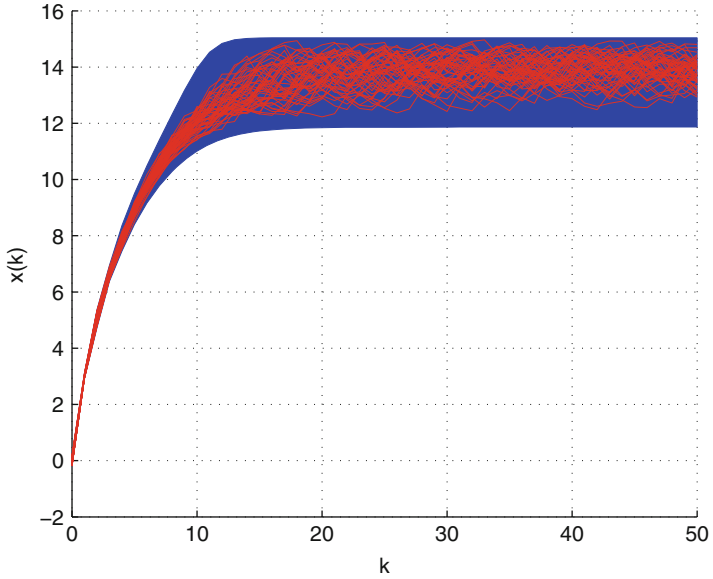


Fig. 7.3 Reachable set estimation for NARMA model. Blue area is the estimated reachable set and red solid lines are 100 randomly generated state trajectories. All the randomly generated state trajectories are in the reachable set estimation area

Furthermore, with the estimated reachable set, the safety verification can be easily performed. For example, if the safety region is assumed to be $\mathcal{S} = \{\mathbf{x} \in \mathbb{R} \mid \mathbf{x} \leq 16\}$, it is easy to verify that $\tilde{\mathcal{X}}_{[0,50]} \cap \neg\mathcal{S} = \emptyset$ which means the NARMA model is safe.

7.6 Magnetic Levitation Systems (Maglev)

7.6.1 Brief Introduction

Magnetic Levitation Systems, which are called Maglev Systems in short, are systems in which an object is suspended exclusively by the presence of magnetic fields. In such schemes, the force exerted by the presence of magnetic fields is able to counteract gravity and any other forces acting on the object [7]. In order to achieve levitation, there are two principal concerns. The first concern is to exert a sufficient lifting force with which to counteract gravity and the second concern is stability. Once levitation has been achieved, it is critical to ensure that the system does not move into a configuration in which the lifting forces are neutralized [25]. However, attaining stable levitation is a considerably complex task, and in his famous theorem, Samuel Earnshaw demonstrated that there is no

static configuration of stability for magnetic systems [8]. Intuitively, the instability of magnetic systems lies in the fact that magnetic attraction or repulsion increases or decreases in relation to the square of distance. Thus, most control strategies for Maglev Systems make use of servo-mechanisms [31] and a feedback linearization [29] around a particular operating point of the complex nonlinear differential equations [46] describing the sophisticated mechanical and electrical dynamics. Despite their intrinsic complexity, these systems have exhibited utility in numerous contexts and in particular Maglev System have generated considerable scientific interest in transportation due to their ability to minimize mechanical loss, allow faster travel [18], minimize mechanical vibration, and emit low levels of noise[16]. Other application domains of such systems include wind tunnel levitation [31], contact-less melting, magnetic bearings, vibrator isolation systems, and rocket-guiding designs [10]. Consequently, Maglev Systems have been extensively studied in control literature [7].

Due to their unstable, complex, and nonlinear nature, it is difficult to build a precise feedback control model for the dynamic behavior of complex Maglev System. In most cases, a linearization of the nonlinear dynamics is susceptible to a great deal of inaccuracy and uncertainty. As the system deviates from an assumed operating point, the accuracy of the model deteriorates [1]. Additionally, models based on simplifications are often unable to handle the presence of disturbance forces. Thus, to improve control schemes, a stricter adherence to the complex nonlinear nature of the Maglev Systems is needed. In the last several years, neural network control systems have received significant attention due to their ability to capture complex nonlinear dynamics and model nonlinear unknown parameters [46].

In the control of magnetic levitation systems the nonlinear nature can be modeled by a neural network that is able to describe the input–output nature of the nonlinear dynamics [31]. Neural networks have shown the ability to approximate any nonlinear function to any desired accuracy [20]. Using the neural network model of the plant we wish to control, a controller can be designed to meet system specifications. While neural control schemes have been successful in creating stable controllers for nonlinear systems, it is essential to demonstrate that these systems do not enter undesirable states. As an example, in the requirements for a Maglev train system developed in 1997 by the Japanese Ministry of transportation, the measurements of the 500 km/h train's position and speed could deviate by a maximum of 3 cm and 1 km/h, respectively, in order to prevent derailment and contact with the railway [22]. As magnetic systems become more prevalent in transportation and in other domains, the verification of these systems is essential. Thus, in this example, we perform a reachable set estimation of an NARMA neural network model (7.1) of a Maglev System.

7.6.2 Neural Network Model

The Maglev System we consider consists of a magnet suspended above an electromagnet where the magnet is confined to only moving in the vertical direction [7]. Using the results of De Jésus et al. [7], the nonlinear equation of motion for the system is

$$\frac{d^2y(t)}{dt^2} = -g + \frac{\alpha}{M}\frac{i^2(t)}{y(t)} - \frac{\beta}{M}\frac{dy(t)}{dt}, \tag{7.25}$$

where $y(t)$ is the vertical position of the magnet above the electromagnet in mm, $i(t)$, in Amperes, is the current flowing in the electromagnet, M is the mass of the magnet, g is the gravitational constant, β is the frictional coefficient, and α is the field strength constant. The frictional coefficient β is dictated by the material in which the magnet moves. In our case, the magnet moves through air. The field strength constant α is determined by the number of turns of wire in our electromagnet and by the strength of the magnet being levitated [7].

To capture the nonlinear input–output dynamics of the system, we trained an NARMA neural network (7.1) to predict the magnet's future position values. In order to predict the magnet's future position values, two inputs are supplied to the network: the first is the past value of the current flowing in the electromagnet $i(k-1)$ and the second input is the magnet's previous position value $y(k-1)$. The output of the neural network is the current position $y(k)$. The network consists of one hidden layer with eight neurons and an output layer with one neuron. The transfer function of the first layer is `tanh` and `purelin` for the output layer.

The network is trained using a data set consisting of 4001 target position values for the output and 4001 input current values. The Levenberg-Marquard algorithm [21] is used to train the network using batch training. Using batch training, the weights and biases of the NARMA model (7.1) are updated after all the inputs and targets are supplied to the network and a gradient descent algorithm is used to minimize error [4]. To avoid over-fitting the network, the training data is divided randomly into three sets: the training set, which consists of 2801 values, the validation set, which consists of 600 values, and a test set which is the same size as the validation set. The training set is used to adjust the weight and bias values of the network as well as to compute the gradient, while the validation set is used to measure the network's generalization. Training of the networks ceases when the network's generalization to input data stops improving. The testing data does not take part into the training, but it is used to check the performance of the net during and after training.

In this example, we set the minimum gradient to 10^{-7}, and set the number of validation checks to 6. Thus, training ceases if the error on the validation set increases for 6 consecutive iterations or the minimum gradient achieves a value of 10^{-7}. In our case, the training stopped when the validation checks reached its limit of 6, obtaining a performance of 0.000218. Initially, before the training begins, the

values of the weights, biases, and training set are initialized randomly. Thus, the value of the weights and the biases may be different every time that the network is trained. The weights and biases of the hidden layer are

$$W^{[1]} = \begin{bmatrix} -68.9367 & -3.3477 \\ -0.0802 & -2.1460 \\ 0.1067 & -3.7875 \\ 0.1377 & -1.5763 \\ -0.3954 & -1.4477 \\ -0.4481 & -6.9485 \\ 0.0030 & 1.5819 \\ 5.9623 & -5.5775 \end{bmatrix}, \boldsymbol{\theta}^{[1]} = \begin{bmatrix} 47.8492 \\ 2.2129 \\ 1.9962 \\ -0.0091 \\ -0.0727 \\ -3.8435 \\ 1.7081 \\ 7.5619 \end{bmatrix}$$

and in the output layer, the weights and the biases are

$$W^{[2]} = \begin{bmatrix} -0.0054 & -0.3285 & -0.0732 & -0.4019 & -0.1588 & -0.0128 & 0.5397 & -0.0279 \end{bmatrix},$$

$$\boldsymbol{\theta}^{[2]} = \begin{bmatrix} 0.1095 \end{bmatrix}.$$

Once the NARMA network model (7.1) is trained and the weight and bias values are adjusted to the values shown above, the reachable set estimation of the system can be computed and a safety requirement $\mathcal{S}$ could be verified. This computation is executed following the process described in the previous section.

7.6.3 Reachable Set Estimation

In order to compute the reachable set and verify if the given specification is satisfied, Algorithm 3 is employed. First, the reachable set estimation using 5 partitions is computed, followed by the reachable set estimation using 20 partitions. After both reachable set estimations are calculated, 200 random trajectories are generated and plotted into Fig. 7.4.

The reachable set estimations and the random trajectories are computed with an initial set and input set that are assumed to be given by

$$\mathcal{X}_0 = \{\mathbf{x}(0) \in \mathbb{R} \mid 4.00 \leq \mathbf{x}(0) \leq 5.00\}, \tag{7.26}$$

$$\mathcal{U} = \{\mathbf{u}(k) \in \mathbb{R} \mid 0.10 \leq \mathbf{u}(k) \leq 1.10,\ \forall k \in \mathbb{N}\}. \tag{7.27}$$

As is observed from Fig. 7.4, all the randomly generated trajectories lie within the estimated reachable set. Also, it can be noted that the area of the reachable set estimation using a larger partition number, that is 20, represented in green, it is

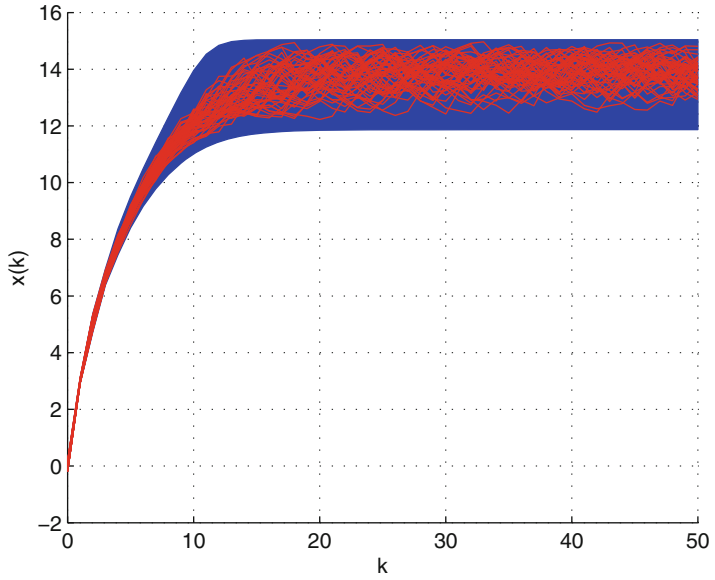


Fig. 7.4 Reachable set estimation using 5 and 20 partitions. The blue area corresponds to the estimated reachable set using 5 partitions, the tighter green area corresponds to the reachable set estimation using 20 partitions, and the red lines correspond to 200 randomly generated state trajectories, which all of them lie within the estimated reachable set area

Table 7.2 Computational time for different partition numbers

Partition number	Computation time
$M_1 = M_2 = 5$	0.048700 s
$M_1 = M_2 = 20$	0.474227 s

smaller than the blue area, which corresponds to the reachable set estimation using a lower partition number ($M_1 = M_2 = 5$) . This is especially noticeable as the time k increases to 40–50, where the difference between the blue region and green region increases as the lower limit of the state $\mathbf{x}(k)$ using 5 partitions keeps decreasing towards 0.6, while the lower limit of the green area maintains a more steady line at 0.7 approximately.

In Table 7.2, the computational time has been recorded for each reachable set estimation. It can be observed that the computational time increases as the partition number increases. For this system, the computational time is approximately 10 times greater when 20 partitions are used. This means that every approach has its different advantages. For the cases when a more precise estimation is needed, we can increase the number of partitions, while for the cases when a larger over-approximation is enough, the number of partitions may be decreased to reduce its computational cost.

The reachable set estimation for the NARMA neural network model (7.1) of the Maglev Systems shows that all system responses to inputs are contained within the reachable set. Thus, our over-approximation of the reachable states is valid. Given

a safety specification $\mathcal{S}$ and the reachable set calculated using Algorithm 3, we are able to determine whether our system model satisfies $\mathcal{S}$. In our example, we did not perform a safety analysis but rather demonstrated the robustness of Algorithm 3 in capturing a large number of possible predictions of the NARMA network model (7.1). The magnet in our example was confined to moving in one dimension. In magnetic levitation systems that are not physically constrained to a set of axes, there are 6° of freedom (three rotational and three transnational) [5]. Thus, while we have demonstrated that our algorithm is robust for two-dimensional systems, it will be good to demonstrate its efficacy on higher dimensional systems. However, as the dimensionality and size of the neural networks increases, the computation time needed to compute the reachable set increases significantly as well.

7.7 Conclusions

This paper studies the reachable set estimation problem for neural network NARMA models of nonlinear dynamic systems. By partitioning the input set into a finite number of cells, reachable set estimation for MLPs can be done for each individual cells and get the union of output set of cells to form an over-approximation of output set. Then, the reachable set estimation for NARMA models can be performed by iterating the reachable set estimation process for MLP step-by-step to establish an estimation of the state trajectories of an NARMA model. Safety properties of NARMA models can then be verified by checking that the intersection between the estimated reachable set and unsafe regions (sets) is empty. The approach is demonstrated by a Maglev System, for which the reachable set of its NARMA neural network model is estimated. The approach is applicable for a variety of neural network models with different activation functions. However, since the estimation is an over-approximation and error will accumulate at each layer, much finer discretization for the input space is required for deep neural networks that necessarily have large numbers of layers, which will introduce a large computational effort, as otherwise the estimation results will be too conservative to be useful. Reducing the conservativeness caused by the increase of layers and generalizing it to deep neural networks will be a future focus for our approach.

Acknowledgements The material presented in this paper is based upon work supported by the National Science Foundation (NSF) under grant numbers CNS 1464311, CNS 1713253, SHF 1527398, and SHF 1736323, the Air Force Office of Scientific Research (AFOSR) through contract numbers FA9550-15-1-0258, FA9550-16-1-0246, and FA9550-18-1-0122, the Defense Advanced Research Projects Agency (DARPA) through contract number FA8750-18-C-0089, and the Office of Naval Research through contract number N00014-18-1-2184. The US government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation thereon. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of AFOSR, DARPA, NSF, or ONR.

References

1. J.I. Baig, A. Mahmood, Robust control design of a magnetic levitation system, in *2016 19th International Multi-Topic Conference (INMIC)* (2016), pp. 1–5
2. S. Bak, P. Sridhar Duggirala, HyLAA: a tool for computing simulation-equivalent reachability for linear systems, in *Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control* (ACM, New York, 2017), pp. 173–178
3. S. Bak, P. Sridhar Duggirala, Rigorous simulation-based analysis of linear hybrid systems, in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems* (Springer, Berlin, 2017), pp. 555–572
4. M.H. Beale, M.T. Hagan, H.B. Demuth, Neural network toolbox user's guide, in *R2016a*. The MathWorks, Inc., Natick (2012). www.mathworks.com
5. P.J. Berkelman, R.L. Hollis, Lorentz magnetic levitation for haptic interaction: device design, performance, and integration with physical simulations. Int. J. Robot. Res. **19**(7), 644–667 (2000)
6. M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L.D. Jackel, M. Monfort, U. Muller, J. Zhang, et al. End to end learning for self-driving cars (2016). Arxiv preprint arXiv:1604.07316
7. O. De Jesus, A. Pukrittayakamee, M.T. Hagan, A comparison of neural network control algorithms, in *International Joint Conference on Neural Networks, 2001. Proceedings. IJCNN '01.*, vol. 1 (2001), pp. 521–526
8. R.J. Duffin, Free suspension and earnshaw's theorem. Arch. Ration. Mech. Anal. **14**(1), 261–263 (1963)
9. P.S. Duggirala, S. Mitra, M. Viswanathan, M. Potok, C2E2: a verification tool for stateflow models, in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems* (Springer, Berlin, 2015), pp. 68–82
10. A. El Hajjaji, M. Ouladsine, Modeling and nonlinear control of magnetic levitation systems. IEEE Trans. Ind. Electron. **48**(4), 831–838 (2001)
11. C. Fan, B. Qi, S. Mitra, M. Viswanathan, P. Sridhar Duggirala, Automatic reachability analysis for nonlinear hybrid models with C2E2, in *International Conference on Computer Aided Verification* (Springer, Berlin, 2016), pp. 531–538
12. S.S. Ge, C.C. Hang, T. Zhang, Adaptive neural network control of nonlinear systems by state and output feedback. IEEE Trans. Syst. Man Cybern. Part B (Cybern.) **29**(6), 818–828 (1999)
13. K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators. Neural Netw. **2**(5), 359–366 (1989)
14. X. Huang, M. Kwiatkowska, S. Wang, M. Wu, Safety verification of deep neural networks (2016). Arxiv preprint arXiv:1610.06940
15. K. Jetal Hunt, D. Sbarbaro, R. Żbikowski, P.J Gawthrop, Neural networks for control systems: a survey. Automatica **28**(6), 1083–1112 (1992)
16. J. Kaloust, C. Ham, J. Siehling, E. Jongekryg, Q. Han, Nonlinear robust control design for levitation and propulsion of a maglev system. IEE Proc. Control Theory Appl. **151**(4), 460–464 (2004)
17. G. Katz, C. Barrett, D. Dill, K. Julian, M. Kochenderfer, Reluplex: an efficient SMT solver for verifying deep neural networks (2017). Arxiv preprint arXiv:1702.01135
18. C.H. Kim, J. Lim, J.M. Lee, H.S. Han, D.Y. Park, Levitation control design of super-speed maglev trains, in *2014 World Automation Congress (WAC)* (2014), pp. 729–734
19. S. Lawrence, C. Lee Giles, Ah. Chung Tsoi, A.D Back, Face recognition: a convolutional neural-network approach. IEEE Trans. Neural Netw. **8**(1), 98–113 (1997)
20. X.-D. Li, J.K.L. Ho, T.W.S. Chow, Approximation of dynamical time-variant systems by continuous-time recurrent neural networks. IEEE Trans. Circuits Syst. Express Briefs **52**(10), 656–660 (2005)
21. L.S.H. Ngia, J. Sjoberg, Efficient training of neural nets for nonlinear adaptive filtering using a recursive levenberg-marquardt algorithm. IEEE Trans. Signal Process. **48**(7), 1915–1927 (2000)

22. M. Ono, S. Koga, H. Ohtsuki, Japan's superconducting maglev train. IEEE Instrum. Meas. Mag. **5**(1), 9–15 (2002)
23. L. Pulina, A. Tacchella, An abstraction-refinement approach to verification of artificial neural networks, in *International Conference on Computer Aided Verification* (Springer, Berlin, 2010), pp. 243–257
24. L. Pulina, A. Tacchella, Challenging SMT solvers to verify neural networks. AI Commun. **25**(2), 117–135 (2012)
25. D.M. Rote, Y. Cai, Review of dynamic stability of repulsive-force maglev suspension systems. IEEE Trans. Mag. **38**(2), 1383–1390 (2002)
26. J. Schmidhuber, Deep learning in neural networks: an overview. Neural Netw. **61**, 85–117 (2015)
27. D. Silver, A. Huang, C.J Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. Nature **529**(7587), 484–489 (2016)
28. C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, R. Fergus, Intriguing properties of neural networks (2013). Arxiv preprint arXiv:1312.6199
29. R. Uswarman, A.I. Cahyadi, O. Wahyunggoro, Control of a magnetic levitation system using feedback linearization, in *2013 International Conference on Computer, Control, Informatics and Its Applications (IC3INA)* (2013), pp. 95–98
30. M. Viet Thuan, H. Manh Tran, H. Trinh, Reachable sets bounding for generalized neural networks with interval time-varying delay and bounded disturbances. Neural Comput. Appl. **29**(10), 783–794 (2018)
31. R.J. Wai, J.D. Lee, Robust levitation control for linear maglev rail system using fuzzy neural network. IEEE Trans. Control Syst. Technol. **17**(1), 4–14 (2009)
32. W. Xiang, On equivalence of two stability criteria for continuous-time switched systems with dwell time constraint. Automatica **54**, 36–40 (2015)
33. W. Xiang, Necessary and sufficient condition for stability of switched uncertain linear systems under dwell-time constraint. IEEE Trans. Automatic Control **61**(11), 3619–3624 (2016)
34. W. Xiang, Parameter-memorized Lyapunov functions for discrete-time systems with time-varying parametric uncertainties. Automatica **87**, 450–454 (2018)
35. W. Xiang, J. Xiao, Stabilization of switched continuous-time systems with all modes unstable via dwell time switching. Automatica **50**(3), 940–945 (2014)
36. W. Xiang, J. Lam, J. Shen, Stability analysis and $\mathcal{L}_1$-gain characterization for switched positive systems under dwell-time constraint. Automatica **85**, 1–8 (2017)
37. W. Xiang, H.-D. Tran, T.T. Johnson, Robust exponential stability and disturbance attenuation for discrete-time switched systems under arbitrary switching. IEEE Trans. Autom. Control (2017). https://doi.org/10.1109/TAC.2017.2748918
38. W. Xiang, H.-D. Tran, T.T. Johnson, On reachable set estimation for discrete-time switched linear systems under arbitrary switching, in *American Control Conference (ACC), 2017* (IEEE, New York, 2017), pp. 4534–4539
39. W. Xiang, H.-D. Tran, T.T. Johnson, Output reachable set estimation and verification for multi-layer neural networks (2017). Arxiv preprint arXiv:1708.03322
40. W. Xiang, H.-D. Tran, T.T. Johnson, Output reachable set estimation for switched linear systems and its application in safety verification. IEEE Trans. Autom. Control **62**(10), 5380–5387 (2017)
41. W. Xiang, H.-D. Tran, T.T. Johnson, Reachable set computation and safety verification for neural networks with ReLU activations (2017). Arxiv preprint arXiv: 1712.08163
42. Z. Xu, H. Su, P. Shi, R. Lu, Z.-G. Wu, Reachable set estimation for Markovian jump neural networks with time-varying delays. IEEE Trans. Cybern. **47**(10), 3208–3217 (2017)
43. L. Zhang, W. Xiang, Mode-identifying time estimation and switching-delay tolerant control for switched systems: an elementary time unit approach. Automatica **64**, 174–181 (2016)
44. L. Zhang, Y. Zhu, W.X. Zheng, Synchronization and state estimation of a class of hierarchical hybrid neural networks with time-varying delays. IEEE Trans. Neural Netw. Learn. Syst. **27**(2), 459–470 (2016)

45. L. Zhang, Y. Zhu, W.X. Zheng, State estimation of discrete-time switched neural networks with multiple communication channels. IEEE Trans. Cybern. **47**(4), 1028–1040 (2017)
46. S.T. Zhao, X.W. Gao, Neural network adaptive state feedback control of a magnetic levitation system, in *The 26th Chinese Control and Decision Conference (2014 CCDC)* (2014), pp. 1602–1605
47. Z. Zuo, Z. Wang, Y. Chen, Y. Wang, A non-ellipsoidal reachable set estimation for uncertain neural networks with time-varying delay. Commun. Nonlinear Sci. Numer. Simul. **19**(4), 1097–1106 (2014)

Chapter 8
Adaptation of Human Licensing Examinations to the Certification of Autonomous Systems

M. L. Cummings

8.1 Introduction

Recent advances in sensor development have enabled new artificial intelligence techniques in the post-processing of sensor data such as deep learning, which have greatly enhanced the perceptual abilities of autonomous systems, most notably, autonomous cars and unmanned aerial vehicles (aka drones). Both systems are expected to reach operational capabilities in the near-term, with the promise of flying cars in the distant, but realizable, future [1].

With the imminent arrival of such vehicles, there is significant discussion around how such systems should be certified as safe to operate, given that humans will no longer be directly responsible for their operation. In aviation and surface (i.e., driving) transportation domains, operators are typically certified to control vehicles through an examination process which typically includes a physical readiness assessment, a written knowledge exam, and a practical examination in the actual vehicle.

These practical exams are operational in nature and in aviation, take the form of a "checkride" whereby a human examiner assesses whether humans can effectively and safely operate an aircraft autonomously. While driving practical exams are not as focused on emergencies as aviation checkrides, they also are attempting to ensure that a vehicle will be safely operated by the human behind the wheel. In these domains, when an operator is licensed, they are effectively certified by a regulatory agency as safe to operate their vehicles across a variety of circumstances. Additional

M. L. Cummings (✉)
Duke University, Durham, NC, USA
e-mail: mary.cummings@duke.edu

H. Yu et al. (eds.), *Safe, Autonomous and Intelligent Vehicles*,
Unmanned System Technologies, https://doi.org/10.1007/978-3-319-97301-2_8

specialized licenses are needed in both driving and aviation when a person wishes to conduct commercial operations, for example, or operate vehicles under special cases.

The intensity of training and difficulty of the examination process varies with the increasing complexity of operating a vehicle. For example, it is much more difficult to gain a pilot's license for a large commercial passenger aircraft as opposed to a small four seat recreational plane. The same is true for large tractor trailers versus a standard automobile, with increasing requirements for increasing gross vehicle weights. However, the licensing of both aviation and surface transportation vehicles, commercial or otherwise, share the same basic structure in that there are three basic elements of the licensing process: (1) Basic physical standards, (2) A written exam that assesses knowledge, and (3) the practical exam where an examiner accompanies the licensee in real-world conditions.

The licensing processes for aviation and surface transportation will be detailed in the following sections, followed by discussions of how such licensing processes could or should inform certification of autonomous vehicles in their respective settings. This chapter focuses primarily on autonomous cars with no required human supervision, i.e., cars with Levels 4 or 5 of automation as defined by SAE standard J3016. For clarity purposes, the term licensing will be reserved for the independent verification that a single human can effectively and safely operate a class of vehicles, while certification will mean that one or a group of homogenous vehicles (in terms of capabilities, sensors, and hardware) can be effectively and safely operated by one or more onboard computers.

8.1.1 Driving Licensing Exams

In the Unites States, the driver's licensing process typically begins with a vision screening test, since vision is the sense responsible for 95% of driving-related inputs [2]. While each state has its own set of standards, typical requirements include corrected visual acuity of 20/40, but people who exceed this could be allowed a license under limited circumstances and with special on-road assessments [3]. It is not uncommon for physicians to recommend limited driving, particularly at night and in low-light conditions for drivers with diseases and conditions that affect their vision. While there are many other medical conditions that can limit a person's driving ability such as hearing loss, epilepsy, and insulin use (49 CFR 391.41), waivers are given on a case by case basis. While general physicals are not required for basic licensing, they are for commercial licenses.

Once basic visual acuity is established, drivers typically take a knowledge examination, which again varies by state, but generally focuses on traffic laws, rules of the road, and safe driving practices. Once the knowledge exam has been passed, the final portion is the on-road practical exam. While each state is responsible for

Table 8.1 Written driving exam knowledge categories with examples [4]

	Trip planning		Crossing/entering with vision obstructed
Pre/post driving	Donning seat belts	Positioning vehicle	Avoiding blind spot
Vehicle control	Anticipating stops	Handling emergencies	Quick stop
	Adjusting speed for turn		Escape paths
Rules of the road	Shared left-turn lanes	Sharing the road	Bicyclists
	Work zone signs		Work zones
Visual search	Avoiding distraction	Special driving situations	Rural road driving
	Lane changing		Mountain driving
Communication	Uses hand signals when appropriate	Driver preparation	Sunlight/sunset
	Horn		What to do when a driver is aggressive
Adjusting speed	Hydroplaning	Vehicle readiness	Displays (legibility)
	Roadside activity		Controls (ease of reach, operation)

their own licensing programs and the exact details can vary for both personal and commercial licensing, there are common guidelines that US states can reference in the design of these exams [4].

These guidelines maintain that the purpose of the knowledge exam is to ensure that driver license applicants possess the information required to operate vehicles in a way that is consistent with public safety and mobility. According to these guidelines, the primary source of this information should be made readily available through a driving manual, with alternate forms of presentation for people who cannot read and other disabilities. The categories that make up the core body of testable knowledge for written driving exams are detailed in Table 8.1, with example topics.

The examples in Table 8.1 were selected to highlight areas where either humans or autonomous cars experience difficulties. For example, human drivers can struggle with anticipating stops under the vehicle control category, or performing quick stops in emergency situations. These are actions that autonomous cars perform with a much higher degree of reliability. However, autonomous cars struggle in poor lighting conditions such as sunset since sensor washout is a problem and at present, they also have difficulties operating in and around work zones due to wide variability in signage and placement of maintenance equipment. These issues will be further explored in a later section.

Written knowledge exams typically assess licensees through multiple choice questions that are selected from a larger database of questions that cover the knowledge domains in Table 8.1. Some states administer their knowledge exams in two parts that address both general questions about the topics in Table 8.1, and then another section that addresses an examinee's understanding of road signs. Figure 8.1 is an example of road signs that examinees would have to identify.

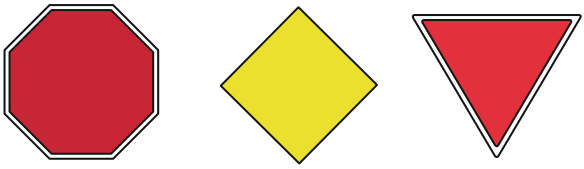


Fig. 8.1 Example signs to be identified in a written driving exam [5]

Table 8.2 Skills that should be assessed during a practical driving test [4]

Attentional	Routine motor skills
Attention-sharing	Acceleration, slowing
Attention-shifting	Turning
Perceptual	Maintaining speed
Spatial judgment	Shifting
Gap judgment	Lane keeping
Distance judgment	Stopping, backing
Hazard detection	Adjusting to limited traction

The last element of a typical driver's licensing exam is the practical test which assesses whether an examinee has the skills to operate an automobile "in a manner consistent with the safety and mobility of the motoring public [4]." The AAMVA breaks down the required skills into three categories: perceptual, attentional, and motor, which are detailed in Table 8.2. Per these licensing guidelines, navigation is a cognitive attribute and should not be part of skills testing, nor should advanced skills like controlled skids be assessed in any practical exam.

These practical skills can be assessed through both on and off road tests, as well as simulation, although the on-road test with an accompanying human examiner is the most common form of driving skill assessment. Such a "check ride" where the examiner accompanies a driver through a course on public roads is an inherently subjective and uncontrolled process, where conditions will naturally vary due to weather, traffic, and the examiner's particular approach to testing. Thus it is not meant to be a scientific process that can predict who will likely crash in the future, but rather exists to verify an examinee possesses the minimal knowledge and skills necessary to drive [4].

8.1.2 Aviation Licensing Exams

Just as driving licenses can fall into two categories, personal and commercial aviation licensing follows a similar structure. Human pilots are "rated" in various aircraft which can range from unmanned aircraft (where pilots can have minimal aviation experience) to a commercial pilot, multi-engine airline transport rating which is the most difficult to achieve and requires substantial experience. Regardless of the type of rating, pilots must also undergo physical, knowledge, and practical exams similar to that of drivers.

Table 8.3 Elements of the airman certification standards

Expected pilot behaviors	Testable elements
Know	Aeronautical knowledge
Consider	Aeronautical decision-making & special emphasis
Do	Flight proficiency

Physical exam requirements generally become more stringent as the size and complexity of the aircraft increases. Glider pilots, for example, only have to write a statement that they have no physical defects that would prevent them from safely operating an unpowered aircraft, but private pilots must pass a physical examination administered by an FAA-authorized aviation medical doctor before they can fly alone. This must be renewed every 5 years until pilots are 40 years of age, then they must renew every 2 years. Passenger airline transport pilots must obtain a first class medical certificate, which is a more comprehensive exam, and it must be renewed every 6 months after a pilot's 40th birthday (14 CFR Part 61).

Just as in drivers' licensing, the physical exam exists to ensure that humans are able to use their required sensors to obtain information about the world and make action decisions in a timely manner. As the FAA's requirements for increasing frequency of physicals indicates, there is enough evidence to suggest that natural aging processes can degrade a person's ability to respond both correctly and in a timely fashion in an aircraft. Because of the possible high consequences of mistakes, commercial airline transport pilots must retire at 65 years, although there is no age limit for private pilots.

Similar to driving, pilots must pass knowledge tests that demonstrate that they understand the knowledge, skill, or risk management elements defined in the Airman Certification Standards.

(ACS) document for the rating they are trying to achieve. The ACS also serves as the document that guides examiners during the "check ride," i.e., the practical exam. The ACS is organized such that areas for operation/tasks are listed, accompanied with information that elucidates what pilots should "Know/Consider/Do" in these circumstances (Table 8.3).

Table 8.4 is an ACS for a pilot undertaking the task of diverting an aircraft when the original destination is unavailable due to weather or some other unforeseen problem. It tells pilots what they need to know, what skills are required, and what considerations are needed for risk management when they are attempting to conduct a successful divert to a new destination. It also clearly underscores what errors of omission lead to failed risk management. In the ACS, a commercial pilot is responsible for 11 areas of operation that include, for example, preflight procedures, navigation, and emergency operation. In total they must exhibit knowledge, skills, and risk management abilities for at least 60 tasks.

While there is a pilot written knowledge exam similar to that in the driving domain, there is also a knowledge portion of the practical exam that takes the

Table 8.4 Airman certification standards for the task of diversion in the area of operation of navigation [6]

Task	C. diversion
References	FAA-H-8083-2, FAA-H-8083-25; AIM; Navigation charts
Objective	To determine that the applicant exhibits satisfactory knowledge, risk management, and skills associated with diversion
Knowledge	The applicant demonstrates understanding of
CA.VI.C.K1	Selecting an alternate destination
CA.VI.C.K2	Situations that require deviations from flight plan and/or ATC instructions
Risk management	The applicant demonstrates the ability to identify, assess and mitigate risks, encompassing
CA.VI.C.R1	Collision hazards, to include aircraft, terrain, obstacles, and wires
CA.VI.C.R2	Distractions, loss of situational awareness, and/or improper task management
CA.VI.C.R3	Failure to make a timely decision to divert.
CA.VI.C.R4	Failure to select an appropriate airport.
CA.VI.C.R5	Failure to utilize all available resources (e.g., automation, ATC, and flight deck planning aids)
Skills	The applicant demonstrates the ability to:
CA.VI.C.S1	Select a suitable airport and route for diversion
CA.VI.C.S2	Make a reasonable estimate of heading, groundspeed, arrival time, and fuel consumption to the divert airport
CA.VI.C.S3	Maintain the appropriate altitude, $\pm$100 feet and heading, $\pm$10°
CA.VI.C.S4	Update/interpret weather in flight
CA.VI.C.S5	Explain and use flight deck displays of digital weather and aeronautical information, as applicable

form of an oral exam prior to the in-flight skills check ride. The written exam tests applicants' knowledge of the rating they are applying for, and the oral exam allows evaluators to further assess missed questions on the written test through scenario-based questioning. This scenario-based questioning is especially relevant to assessment of risk mitigation since there are no set of absolute procedures for every possible contingency and emergency. This more loosely structured assessment of risk mitigation does not exist in the driving domain, and as will be discussed later, will be a key issue in the certification of autonomous systems.

Once applicants pass the written and oral exam, they proceed to the in-flight practical test which tests applicants' required skills in the performance of various tasks and maneuvers. During this portion of the exam, the evaluator can further assess not just the skills but the abilities of pilots to conduct safe risk mitigation through simulation of emergencies (such as retarding one throttle of a two engine plane to simulate an engine flameout.) The evaluator can then witness first hand as the applicant maneuvers the actual plane as well as communicate with air traffic control to effect a safe landing. The ability of a human evaluator to alter his or her test strategy in flight to adjust to differences in human performance is a key element of this evaluation process. Just as in driving, the point of these exams is not

to determine predictors for eventual problems, but to ensure applicants can safely operate aircraft.

8.2 SRKE Taxonomy

The previous discussions of driving and pilot licensing procedures demonstrate that there is a clear commonality between these programs. Generally, the licensing of humans to operate vehicles focuses on the ability of the human to perceive the world around them correctly (tested through physical exams), understand the rules and guiding principles of the overarching systems (knowledge exam), and then demonstration of the skills required to safely operate in these setting (the practical exam/check ride). The one remarkable difference between driving and pilot licensing is that pilots of all levels must explicitly demonstrate risk management in their operations, where drivers do not. As will be illustrated, this should be a significant consideration in the certification and implementation of autonomous cars.

To better understand why this seemingly small difference is so important, and why autonomous vehicle (AV) certifications will need to look much more like aviation certifications than those of driving, the SRKE (skills, rules, knowledge, and expertise) taxonomy is introduced in Fig. 8.2 [7]. This framework demonstrates how and when functions should be allocated between humans and autonomous systems, and what the implications are for certification of autonomous systems of all types.

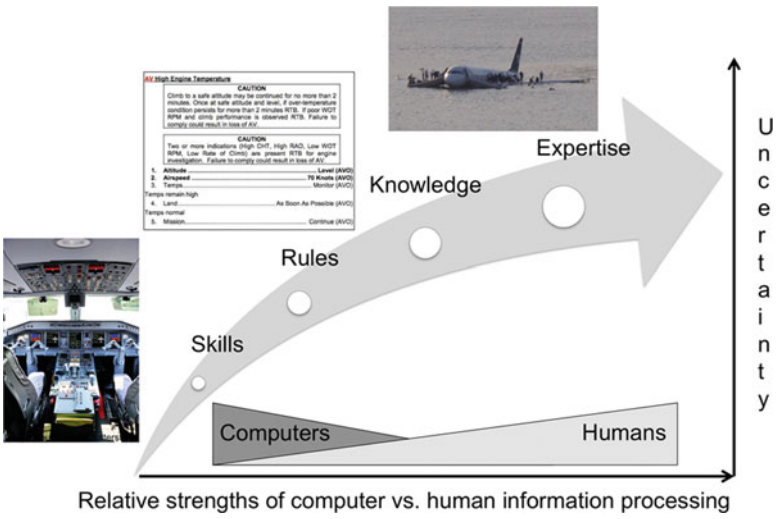


Fig. 8.2 Skill, rule, knowledge, & expert behaviors and the relationship to uncertainty

In the SRKE taxonomy, skill-based behaviors are sensory-motor actions that are highly automatic, and for humans are typically acquired after some period of training [8]. In flying and driving, the bulk of a human operator's work is a set of motor responses that should become routine and nearly effortless with practice. In Fig. 8.2, an example of skill-based control is the act of flying an aircraft. Student pilots spend the bulk of their training learning to scan dials and gauges so that they can instantly recognize the state of an aircraft and adjust as needed. However, automation is superior in this task because such tasks require motor memory with a feedback error correction loop. In most commercial aircraft, automation does the bulk of flying and pilots only typically manipulate controls 3–7 min for flights of any length [9].

There is no question that the bulk of car and aviation accidents are due to human error, often because the correct skills were not applied in a timely fashion [10, 11]. Because the automaticity of human skill-based control can be brittle due to problems like distraction, vigilance, fatigue, and the neuromuscular lag [12], they arguably should be replaced with automation in environments requiring these skills. However, as will be discussed further, whether skill-based reasoning can be executed reliably by AVs across the breadth of required scenarios is still an open question.

Once a set of skills is mastered, both computers and operators can then attend to higher cognitive tasks called rule-based behaviors (Fig. 8.2), which are effectively those actions guided by subroutines, stored rules, or procedures. For example, when a fire light illuminates or some other sub-system indicates a problem, pilots recognize that they should consult a manual to determine the correct procedure (since there are far too many procedures to be committed to memory), and then follow the steps to completion. Some interpretation is required, particularly for multiple system problems, which is common during a catastrophic failure like the loss of thrust in one engine. Recognizing which procedure to follow is not always obvious, particularly in warning systems where one aural alert can indicate different failure modes.

By the very nature of their if-then-else structures, rule-based behaviors are potentially good candidates for automation but uncertainty management is key, as depicted in Fig. 8.2 on the vertical axis. If there is low uncertainty in the world, computer rule-based reasoning is typically superior than humans such as the ability of a car's navigation system to find the quickest path in a much faster time than humans. However, if there is traffic congestion or roadwork unknown to the automation, which are sources of uncertainty, human path planning is often superior.

This example highlights a conundrum of automation in the face of uncertainty. While fast and able to handle complex computation far better than humans, computer algorithms, which work primarily at the rule-based level, are notoriously brittle in that they can only take into account those quantifiable variables identified in the design stages that were deemed to be critical [13]. In complex systems with inherent uncertainties (weather, traffic delays), it is not possible to include a priori every single variable that could impact the final solution.

Fig. 8.3 Using black and white stickers and a general attack algorithm called Robust Physical Perturbations, researchers can cause a computer vision system to see the stop sign as a 45 mph speed limit sign [15]

Another problem for automation of rule-based behaviors is similar to one for humans, which is the selection of the right rule or procedure for a given set of stimuli. Computers will reliably execute a procedure more consistently than any human, but the assumption is that the computer selects the correct procedure, which is highly dependent on correct sensing. It is this reliance on near-perfect sensing that can make autonomous car rule-based reasoning extremely brittle, especially if the sensors do not work correctly. For example, poor computer vision was a major factor in the death of a Tesla driver who relied on the car's autopilot for obstacle detection [14]. The car could not execute its automated braking rule set, which generally is excellent, because it never "saw" a tractor-trailer crossing the road ahead and thus was never triggered.

While the ability of autonomous car perceptual sensors to map the world in a reliable and repeatable manner under all expected driving conditions is still very much an open engineering question, another relatively new area of concern for these perception systems is hacking. Even if autonomous car sensors work correctly under all weather conditions and in all traffic configurations, they are still not guaranteed to work correctly, especially if individuals with expertise know how to "trick" such systems.

Figure 8.3 depicts a stop sign that was "hacked" by researchers placing four black and white stickers in strategic locations on the sign [15], such that the computer vision system, enabled through a deep learning network, "saw" the sign as a 45 mph sign instead of a stop sign. If this were to happen in the real world, the car could select its increase-speed-to-45-mph algorithm instead of its stopping algorithm. While such demonstrations currently occur in research settings, it is inevitable that they will occur in the real world.

This example highlights that as uncertainty grows, both humans and computers will be required to reason inductively, as opposed to deductively because not all information can be known and estimates will have to be made. This is illustrated

in Fig. 8.2 as knowledge-based reasoning. Humans resolve these scenarios through judgment and intuition, and often rely on past experience to resolve such situations. This was the case in the landing of USAIR 1549 in 2009 in the Hudson River, as depicted in Fig. 8.2. The Captain had to decide whether to ditch the aircraft or attempt to land it at a nearby airport. Given his mental model, the environment, and the state of the aircraft, his knowledge-based reasoning made him choose the ditching option, which led to a successful outcome.

The last level in the SRKE taxonomy is expertise, in which knowledge-based behaviors are a prerequisite for gaining expertise, but cannot be achieved without significant experience under the highest levels of uncertainty, which is difficult to replicate in test environments. Because of the aforementioned brittleness problems in computer algorithms and the inability to replicate the intangible concepts of judgment and intuition, knowledge-based reasoning and true expertise, for now, are outside the realm of computers. Advances are being made in areas of machine learning, including deep learning, which may help computers approximate something like knowledge-based reasoning. However, as demonstrated by the stop sign example in Fig. 8.3, advancements in these areas are still very preliminary and significant research is needed to determine more reliable and robust methods for autonomous systems to reason under uncertainty.

8.3 Implications of Human Licensing on Autonomous Vehicle Certification

Given current licensing and certification processes of humans in both driving and aviation domains, and understanding the SRKE framework, what then can be learned from human licensing that could be applied to autonomous systems? Given that AVs will be increasingly introduced on the roadways, both federal and state agencies will quickly need answers to certification and licensing questions, which will also be increasingly needed in future drone operations.

8.3.1 Vision Tests for AVs?

One important parallel that could be drawn between human and autonomous systems licensing is the very first test of any operator certification, the physical screening stage. More specifically, it is likely that licensing and certification tests of new autonomous systems will need to consider just how well these vehicles "see" the world around them.

As demonstrated in Fig. 8.3, computer vision systems can fail miserably in what humans would see as a very simply inductive reasoning task. And such failures are not limited to intentional forms of hacking. Snow on top of signs, long shadows

falling across signs in the late afternoon, graffiti, and foliage obscuring part of a sign or roadway elements are just some of the issues that can cause computer vision systems to not "see" correctly. Sensor washout in the early morning or late afternoon is another problem as well as faulty obstacle detections, such as the inability of many vision systems to distinguish writing in the sides of trucks from static signs.

How then can licensing agencies ensure that autonomous vehicles can "see," particularly in known areas of difficulty? Will new driverless cars be taken to test tracks and given partially obscured signs in the late afternoon or in snowing conditions to ensure that they operate as intended? Are there a set of images that can be "shown" to a computer vision system quickly, much like the Ishihara Color Plate Test where numbers made up of dots of one color are embedded in a field of other colors to ensure human can discriminate colors? Such types of tests that can quickly detect anomalies in software systems powered by probabilistic reasoning would likely be part of a larger multi-stage approach to testing, but as in all systems, early identification of significant issues is central to faster and less expensive development processes.

8.3.2 Knowledge Tests and Checkrides for AVs?

In terms of a written knowledge test, there are likely no direct certification parallels for the licensing of autonomous. However, just as there is industry consensus of what knowledge all human operators should have in present-day flying and driving domains (i.e., Tables 8.1 and 8.4), there should be a similar set of guidelines developed by NHTSA, who is responsible for regulating automobiles, that dictates the core body of "knowledge" that all Level 4 and 5 cars should have before they are safe to operate under various conditions. Once this minimum set of knowledge "requirements" are elucidated, then tests can be constructed either in simulation or in special test track environments that allow manufacturers to demonstrate that their cars meet such requirements.

These tests of knowledge, either in the real world or in simulation, then begin to approximate what humans experience as checkrides. Given the low level skill-based nature of driving practical exams which most autonomous cars today could easily pass (although their ability to follow verbal instructions would likely be abysmal), practical autonomous driving tests should model the checkride format of pilot licensing exams, especially the risk mitigation elements. Figure 8.2 highlights that as uncertainty and risks grow, the demand for knowledge and expert based reasoning also grows. Therefore, autonomous vehicle certification tests should focus on how the underlying software and hardware components perform under conditions of high uncertainty that could significantly jeopardize passenger safety as well as those individuals outside the car.

So if we start to combine the information in Table 8.1 with the format of an FAA licensing checkride (Table 8.4), then a set of requirements would emerge that dictate that AVs should be able to, for example, identify, assess, and mitigate

risks for positioning the car while crossing/entering and intersection with vision obstructed or identify, assess, and mitigate risks when driving through shared work zones. Evaluators would then look for evidence from the manufacturers that models of AVs with various software and hardware combinations could indeed detect they are in one of these high risk areas of operations, and then develop an accurate world model of the environment, with *good enough* estimates of the potential risks and outcomes so that a safe course of action is generated by the car.

This emphasis on good enough is rooted in the decision theoretic concept known as satisficing [16], in which humans cannot have perfect knowledge of the world and so will generate solutions that are at least good enough to meet minimum constraints. The notion of licensing someone who performs not necessarily perfectly is well established in aviation. A critical element of these checkrides is that the examiner is attempting to assess the likelihood that an examinee will be able to perform correctly under even the highest levels of uncertainty. While there are clear standards for how checkrides should be administered, ultimately the examiner must make a judgment as to whether he or she *feels* the examinee is a safe *enough* operator. Thus there is an element of trust that is developed throughout the oral exam and checkride, and while the examinations are rooted in objective criteria, there are clearly subjective elements as well.

It is generally accepted that for AVs to be widely accepted, they will need to generate equivalent or better levels of safety (EBLS) than those of human-driven cars [17]. Thus autonomous vehicles will effectively need to be assessed on their ability to not necessarily behave perfectly in all settings, but to satisfice in a way that is superior to humans, especially in low probability but high consequence settings. Formally defining EBLS so that they can be assessed through either formal testing or naturalistic studies is still a step manufacturers and regulators have yet to address.

This need for regulators and evaluators to understand if and how AVs can generate plans and actions that meet EBLS is especially difficult given the opaque nature of the underlying machine learning algorithms, which are notoriously difficult to understand [18]. The complexity of such approaches coupled with human difficulties in reasoning under uncertainty [19] has motivated a relatively new field of inquiry into explainable or interpretable artificial intelligence (AI). Significant work is needed to develop more transparent algorithms, visualizations, and interaction modalities such that people with advanced degrees in machine learning can understand the limits of such approaches.

8.3.3 Graduated Licensing

In the United States and in many other countries like Australia and the United Kingdom, teenage drivers are certified through a staged process called graduated licensing (GL), although these programs vary in requirements. In GL programs, teen drivers are typically first allowed to drive only when supervised, and this license is often called a learner's permit. This stage can be followed by a provisional licensing

period where drivers are allowed to operate a vehicle unsupervised, but are subject to many restrictions such as limited hours of operation (e.g., daylight only), number and ages of passengers in the car, no cell phone in the car, etc. When experience and/or age requirements are met, the teen driver can obtain a full license.

Autonomous vehicles have arguably been undergoing a testing period similar to the learner's permit process, with many states requiring a safety driver in the car to monitor the system and take over as necessary. However, in California AVs are now being allowed to operate on public roads with no human driver in the car, but with an operator remotely monitoring the vehicle's progress and intervening as necessary through teleoperation. While such operations will undoubtedly yield valuable test data, they also need to be carefully considered. Relying on remote teleoperation to take over a vehicle in the case of an urgent or emergent scenario is extremely dangerous when required response times are on the order of seconds, which is typical of driving scenarios, especially those on the highway.

Humans suffer from an immutable physical limitation known as the neuromuscular lag, which is the 0.5 s delay that exists between the time of stimulus onset and the ability to react [12]. This inherent time delay is made worse in remote control of AVs given both communication delays and time required for remote operators to gain situation awareness in an information-impoverished remote environment. Indeed at one point in time, the US Air Force lost one third of its Predator unmanned aerial vehicle fleet because of the inability of remote operators to successfully control these vehicles, especially on landing which occurs at speeds similar to those of highway speeds [20]. It is highly unlikely that at highway speeds, a remote AV operator can effectively intervene to prevent an accident.

Given both the perceptual vision problems of AVs as outlined previously, as well as known problems with teleoperation of vehicles in time-critical scenarios, AVs should be held to a similar GL process with very clear restrictions on permissible areas, times, speeds, and weather conditions until such time that the AVs can demonstrate reliable operation under those conditions. Moreover, in other countries like England, Australia, and India, inexperienced drivers are required to put highly visible signs on their cars with letters like L (learner) and P (provisional). The point of these signs is to heighten the awareness of nearby drivers to potentially unsafe behaviors, so they can adjust accordingly. Going forward, AVs should be required to have such clear markings, not only to alert human drivers of potential problems but also because in these early days which clearly constitute testing, human drivers should be made explicitly aware that they are taking part in a test [21].

8.3.4 Certifying Machine Learning Algorithms Is Unprecedented

One important difference between the licensing of human-operated and autonomous vehicles is the lack of individual differences. In both driving and flying, individuals

are certified as safe to operate their vehicles and in the more complex case of flying, the practical exam can be tailored to various individuals as an examiner sees fit. Given that all models of a specific car or drone will presumably be identical, autonomous vehicles of the future will not need to be certified individually. Instead, they will likely need to be certified in groups and classes.

For example, various manufacturers have already proposed different driverless car designs, which means that groups of cars will have different hardware and software solutions between manufacturers but it is also possible that manufacturers will have different solutions on different models. This raises the question of component testing and certification, i.e., if a LIDAR is certified on one model of a Ford car, are all LIDARs on all their cars certified? What if cars use the same hardware, but different software post-processing algorithms?

Software upgrades are another issue that will need to be addressed in licensing and certification. Tesla is the first car company to introduce automatic over-the-air software updates for its cars, but as self-driving cars come online with potentially unreliable capabilities, will major upgrades that introduce new capabilities need certification before pushed to owners who will not likely read the owner's manuals and any associated warnings?

Software bugs and failures are commonplace in current cars, without any advanced autonomy or embedded machine learning. Failures to detect software bugs were implicated in the Toyota unintended acceleration cases leading to a 1.2 B settlement with the US government [22]. Software bugs in general were responsible for 15% of recalls in 2015, and the number of software-related components involved in recalls grew by almost 600% between 2011 and 2015 [23]. These problems occurred despite car companies having access to well-established test practices for traditional software. Given that there are no established test standards for probabilistic reasoning software like machine learning, this rate of software errors with resulting recalls will likely significantly increase.

The automotive industry will likely need to model how the FAA certifies new software upgrades for fly-by-wire aircraft, which share many commonalities with AVs. The FAA relies on an office in the Aviation Safety branch, the Aircraft Certification Service, for approval of software and airborne electronic hardware for airborne systems including autopilots and flight controls, which have direct parallels in the automotive industry. One key aspect of the FAA's software certification process is their partnership with a not-for-profit association called the RTCA (Radio Technical Commission for Aeronautics). This group works with the FAA and the aviation industry to develop consensus-driven performance standards and guidance materials that serve as a partial basis for certification of critical systems and equipment used in the conduct of air transportation.

This certification of complex hardware and software aviation systems through consensus has been generally successful, but there are examples of incomplete or faulty certifications such as the Boeing 787 Battery fire in 2013, where the FAA's certification process was called into question [24]. However, as software grows in complexity, it is not clear that the consensus approach to autonomous system certification is going to be sufficient. Current testing approaches for such systems

rely heavily on deterministic testing, meaning that for every set of inputs, the system will act or react the same way within some small confidence interval. Such testing processes are highly repeatable and for both hardware and software components in the air and on the road today, there are widely accepted industry test standards,

Autonomous systems are stochastic, as opposed to deterministic ones, which means they rely on probabilistic reasoning and significant estimation. They heavily leverage machine learning, aka deep learning, which is a data-intensive approach to developing an autonomous system world model, which serves as the core set of assumptions about who, what, and where agents in the system are and what their likely next set of behaviors and actions will be. To date, there exists no industry consensus on how to test such systems, particularly in safety-critical environments of high uncertainty. Given this lack expertise, the FAA's approach to software certification may serve as a starting point for future autonomous vehicle certifications, but significantly work is needed in academic and industry to address a significant knowledge gap in certifying control algorithms that rely on imperfect sensors and data-driven reasoning to make decisions in safety-critical systems. As mentioned previously, significant more work is needed in the emerging field of Explainable AI in order to fill this knowledge gap.

8.4 Conclusion

Currently there is significant legislation pending at both state and federal levels to introduce autonomous vehicles into the marketplace. Regardless of the outcomes, autonomous vehicles will be introduced into the everyday driving landscape, but significant questions remain as to how such vehicles could or should be certified as safe enough to operate at equivalent or better levels of safety demonstrated by human drivers.

In both driving and aviation settings, humans, who are autonomous agents, are certified through a three-tiered process which focuses on physical readiness, demonstration of broad system knowledge of operation, and practical exams, aka checkrides, where operators are tested on their skills and in the case of aviation, their ability to mitigate risk in the face of uncertainty. While such certification processes will not be needed for humans in the case of Level 4 and Level 5 vehicles, there is still much that can be learned from how humans are certified to safely operate vehicles and aircraft and then applied to autonomous vehicles of all types.

First, while AVs will not need to be assessed on their physical readiness per se, their vision systems contain many weaknesses, with more emerging in basic research demonstrations almost daily. Such weaknesses are due to not only sensor limitations but also software-driven post-processing which makes them vulnerable to hacking. Given that these perception systems are the heart of any autonomous car, it is critical that weaknesses in these systems are both known and mitigated, making testing of these systems especially critical.

Second, the body of knowledge that human drivers are tested on is highly relevant when determining EBLOS of AVs. Moreover, just as the FAA requires human pilots to exhibit demonstrable knowledge across various areas of operations including how to mitigate risk for known contingencies, AVs should be required to demonstrate their boundaries of competence in driving scenarios that range from mundane to life threatening. More specifically, given the known weaknesses of perception systems, it is particularly important that both engineers and regulators have a clear understanding of how the probabilistic algorithms embedded in AVs perceive and mitigate risk. Unfortunately, research into explainable AI is just beginning and so the limits of such approaches are still not well understood.

It is important to point out that these arguments and comparisons are primarily targeting the certification of Level 4 and 5 vehicles, which assume the driver is not relied upon for intervention. The licensing/certification of Level 3 vehicles contains elements of both human licensing and vehicle certification, which complicates which agencies would be ultimately responsible since NHTSA typically is responsible for vehicle safety while state agencies are responsible for driver licensing.

Level 3 vehicle certification is very difficult and beyond the scope of this chapter since such operations require coordinating the handing of vehicle control back and forth between the computer and human. Because human interaction with autonomous systems is plagued with known serious issues such as mode confusion [25], boredom and complacency [26], and distractions and delayed reaction times [27], licensing and certification are not separable issues. In these cases, one small vehicle design change can have dramatic effects on driver behavior and thus licensing and certification should be seen as an integrated effort as opposed to mutually exclusive efforts.

Self-driving and driverless cars carry great promise both in terms of reduced driver deaths and increased accessibility for drivers with a myriad of disabilities or a lack of access to a personal vehicle. However, those technologies that will enable these achievements are very immature, with no industry consensus of a minimum set of safety standards, or how such cars should be tested including identification of the corner cases that define worse possible scenarios. Significantly more work is needed to develop principled testing protocols and there are many important lessons to be learned from how humans have been licensed in driving and aviation domains.

One critical advancement that is needed for both engineers and regulators of future autonomous systems of all types is the development of explainable AI methods that give both engineers and regulators insight into both how such probabilistic systems achieve their solutions and when such solutions are brittle and likely to fail. While accidents and deaths ultimately lead to safer systems, especially for emerging technologies [28], we can and should identify major weaknesses before deadly problems materialize in the population at large.

Acknowledgements This work was inspired by Dr. Michael Francis and supported in part by the US Department of Transportation and the University of North Carolina's Collaborative Sciences Center for Road Safety (CSCRS).

References

1. V. Nneji, A. Stimpson, M.L. Cummings, K. Goodrich, in *Exploring Concepts of Operations for on-Demand Passenger Air Transportation*. Paper Presented at the AIAA Aviation, Denver, CO (2017)
2. D. Shinar, F. Schieber, Visual requirements for safety and mobility of older drivers. Hum. Factors **33**(5), 507–519 (1991)
3. American Medical Association, in *Physician's Guide to Assessing and Counseling Older Drivers*. Chapter 9: Medical conditions and medications that may impair driving (National Highway Transportation and Safety Administration, Washington, 2003)
4. American Association of Motor Vehicle Administrators, AAMVA Guidelines for Noncommercial Knowledge and Skills Test Development. Retrieved from Arlington, VA (2014)
5. Virginia Department of Motor Vehicles, Virginia Driver's Manual. Commonwealth of Virginia (2017)
6. Flight Standards Service, Commercial Pilot – Airplane Airman Certification Standards. FAA-S-ACS-7 (Changes 1 & 2). (U.S. Department of Transportation, Washington, 2017)
7. M.L. Cummings, Man vs. Machine or Man + Machine? IEEE Intell. Syst. **29**(5), 62–69 (2014)
8. J. Rasmussen, Skills, rules, and knowledge: Signals, signs, and symbols, and other distinctions in human performance models. IEEE Trans. Syst. Man Cybern. **13**(3), 257–266 (1983)
9. M.L. Cummings, A. Stimpson, M. Clamann, in Functional Requirements for Onboard Intelligent Automation in Single Pilot Operations. Paper Presented at the AIAA SciTech Conference, San Diego, CA (2016)
10. National Center for Statistics and Analysis, *Traffic Safety Facts: Research Note* (Department of Transportation, Washington, 2017)
11. D.A. Wiegmann, S.A. Shappell, *A Human Error Analysis of Commercial Aviation Accidents Using the Human Factors Analysis and Classification System (HFACS)* (Federal Aviation Administration, Washington, 2001)
12. R.J. Jagacinski, J.M. Flach, *Control Theory for Humans: Quantitative Approaches to Modeling Performance* (Lawrence Erlbaum Associates, New Jersey, 2003)
13. P.J. Smith, C.E. McCoy, C. Layton, in Brittleness in the Design of Cooperative Problem-Solving Systems: The Effects on User Performance. Paper Presented at the IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans (1997)
14. NTSB, *Highway Accident Report: Collision Between a Car Operating with Automated Vehicle Control Systems and a Tractor-Semitrailer Truck near Williston, Florida May 7, 2016*. NTSB/HAR-17/02 PB2017–102600. Washington (2017)
15. I. Evtimov, K. Eykholt, E. Fernandes, T. Kohno, B. Li, A. Prakash, D. Song, Robust physical-world attacks on deep learning models. arXiv preprint 1707.08945 (2017)
16. H.A. Simon, *Models of Bounded Rationality*, vol 2 (MIT Press, Cambridge, 1982)
17. L. Fraade-Blanar, N. Kalra, Autonomous Vehicles and Federal Safety Standards: An Exemption to the Rule? Retrieved from Santa Monica, CA (2017)
18. T. Miller, P. Howe, L. Sonenberg, in Explainable AI: Beware of Inmates Running the Asylum. Paper Presented at the IJCAI-17 Workshop on Explainable AI (XAI) Proceedings, Melbourne, Australia (2017)
19. A. Tversky, D. Kahneman, Judgment under uncertainty: Heuristics and biases. Science **185**(4157), 1124–1131 (1974)
20. S. Erwin, *UAV Programs Illustrate DoD's broken Procurement System*. National Defense (2009)
21. M.L. Cummings, *The Brave New World of Driverless Cars: The Need for Interdisciplinary Research and Workforce Development* (TR News, 2017), pp. 34–37
22. D. Douglas, M.A. Fletcher Toyota Reaches $1.2 Billion Settlement to End Probe of Accelerator Problems. *Washington Post* (2014)
23. N. Steinkamp, in 2016 Automotive Warranty & Recall Report (2016)

24. NTSB Auxiliary power unit battery fire Japan Airlines Boeing 787–8, JA829J Boston, Massachusetts January 7, 2013. NTSB/AIR-14/01 PB2014–108867 (National Transportation Safety Board, Washington, 2014)
25. J. Bredereke, A. Lankenau, in A rigorous view of mode confusion. Paper presented at the SafeComp, Bremen, Germany (2002)
26. M.L. Cummings, F. Gao, Boredom in the workplace: A new look at an old problem. Hum. Factors **58**(2), 279–300 (2016)
27. T.A. Ranney, *Driver Distraction: A Review of the Current State-of-Knowledge* (Department of Transportation, Washington, 2008)
28. H. Petroski, *To Engineer is Human* (Vintage Books, New York, 1992)

Chapter 9
Model-Based Software Synthesis for Safety-Critical Cyber-Physical Systems

Bowen Zheng, Hengyi Liang, Zhilu Wang, and Qi Zhu

9.1 Software Challenges in Safety-Critical Cyber-Physical Systems

Cyber-physical systems (CPS) integrate cyber elements, i.e., embedded sensing, computation and communication components, with physical processes such as mechanical components, physical environment, and human activities [28]. The cyber elements execute algorithms to closely monitor and control the physical processes in a feedback loop [28, 43]. Typical CPS such as autonomous and semi-autonomous vehicles, industrial robots, medical devices, and smart buildings have shown immense potential economic and societal benefits.

In many CPS, software has become increasingly critical and been driving the system innovations [27, 45]. The design of safe and reliable CPS software, however, faces tremendous challenges from the fast increase of functional complexity, the adoption of more advanced and distributed architectural platforms, the uncertainty of physical environment, and the diverse and stringent design requirements.

Functional Complexity Modern cyber-physical systems have increasingly complex functionalities. Take automotive systems as an example, the driving functions have been evolving from traditional engine, transmission and brake control to functions for Advanced Driver Assistance Systems (ADAS) such as adaptive cruise control, lane keeping assist, and collision avoidance warning, and then to fully

B. Zheng
University of California, Riverside, Riverside, CA, USA
e-mail: bzhen003@ucr.edu

H. Liang · Z. Wang · Q. Zhu (✉)
Northwestern University, Evanston, IL, USA
e-mail: hengyiliang2018@u.northwestern.edu; zhilu.wang@u.northwestern.edu; qzhu@northwestern.edu

H. Yu et al. (eds.), *Safe, Autonomous and Intelligent Vehicles*,
Unmanned System Technologies, https://doi.org/10.1007/978-3-319-97301-2_9

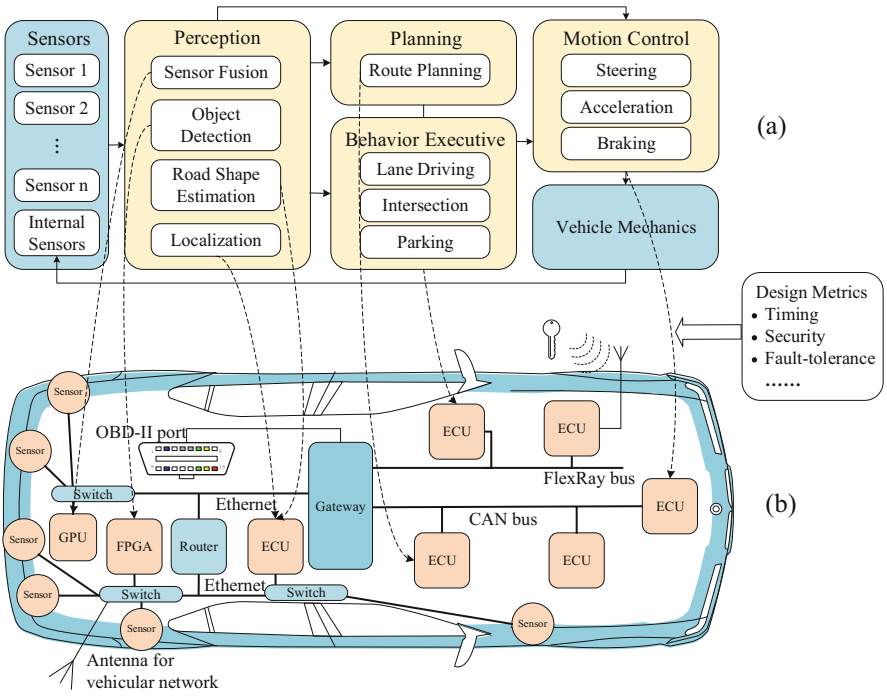


Fig. 9.1 Typical autonomous driving functions and architecture

autonomous driving functions. To enable autonomy, new advanced functions need to be implemented and integrated to work collaboratively, including perception, planning, behavioral executive, and motion control modules. Figure 9.1a shows the basic functions of a typical autonomous vehicle [23, 52]. The perception function is to collect and fuse information from various sensors and understand the environment. The planning function is to arrange routes to travel destinations. The behavioral executive function is to deal with driving behaviors such as lane change, intersection crossing, and parking. The motion control function is to physically execute the behavior generated from the behavioral executive function. In addition to these basic functions, there may also be service functions for task management, vehicular network communication, system configuration, fault and error handling, etc.

As software functionality becomes more complex and provides more features, the volume and value of software in CPS also increase. From year 2000 to 2010, automotive software increased from 2% to 13% of a vehicle's total value, and the number of lines of code increased from one million to more than ten million [8, 37, 45]. It is estimated by Morgan Stanley that the software layer can take 40% of the value of a self-driving vehicle [10]. Such increasing functional complexity also leads to the change of CPS architectural platforms, as discussed below.

Architectural Complexity We again take automotive systems as an example. In the traditional design of automotive electronic systems, each function is provided as a black-box by a Tier-1 supplier and typically implemented on an individual Electrical Control Unit (ECU). Multiple ECUs are connected through buses such as Controller Area Networks (CAN) [6]. With the increase of functional complexity, such *federated architecture* has resulted in the increase of number of ECUs, number of buses, total length of wires, and overall architectural complexity. Currently, there are 50 to 100 ECUs in a standard vehicle controlling dozens of processes [1, 7], increased from around 20 ECUs in just two decades ago. To address this challenge, there is a trend to adopt more powerful computation components and reduce the number of ECUs. In this new *integrated architecture*, one function can be distributed over multiple ECUs, and multiple functions can be executed on one ECU [17]. For computationally-intensive functions (e.g., imaging and video processing functions for autonomous or semi-autonomous driving), accelerators such as GPUs and FPGAs may be used.

Figure 9.1b shows an illustration of possible future automotive architecture. Various sensors are placed to collect real-time information from the environment, and computation components (e.g., single- or multi-core ECUs, GPUs, FPGAs) are connected with buses and gateways to execute perception, planning, behavior executive and motion control functions. New automotive bus protocols such as Time-Sensitive Networking (formerly known as Ethernet AVB [18, 44]) and Time-Triggered Ethernet [22] may be used to address the message size, speed, and bandwidth limitations of traditional CAN buses (autonomous driving functions may need to process hundreds of megabytes or even gigabytes of data per second [20]).

With functional and architectural complexity continuously increasing, there will be more sharing and contention among software functions implemented on the same architectural platform. This presents significant challenges for software synthesis, especially given the uncertain environment and stringent design requirements.

Environment Uncertainty Another major challenge for CPS software design comes from the uncertainty of physical environment. For instance, a vehicle's surrounding environment may change from rural roads to crowded urban areas, from daylight to dark night time, and from clear day to snow or rain. Such drastic change of physical environment leads to high variability of sensing data input, in terms of both volume and characteristics. Consequently, corresponding data processing functions' workload and requirements change rapidly and significantly [11]. The traditional design approach that is solely base on worst-case analysis may lead to inferior or infeasible designs in such cases.

Diverse and Stringent Design Requirements There are often requirements on a variety of objectives/metrics in CPS software design, which may lead to conflicting synthesis choices. In our work, we have considered a number of design metrics, including performance, security, extensibility, fault tolerance, reusability, modularity, memory usage, etc. In the following, we will particularly focus on timing (which relates to many design metrics), security, and fault tolerance.

Timing is at the core of CPS software design, and has critical impacts on both functional correctness and various design metrics such as control performance, extensibility, fault tolerance, and security [30, 40, 46, 55]. Despite significant advances in capturing timing for CPS modeling and simulation [2, 12, 16, 29, 51, 64], the synthesis of CPS software remains hindered by timing-related issues: (1) *diversity of timing requirements* from different design metrics, some with conflicting constraints; (2) *complexity of timing analysis* under complex scale, hierarchy and concurrency of computation and communication; and (3) *uncertainty of timing behavior* resulting from dynamic environment, data input, and platform conditions.

CPS security has become a pressing issue recently, as evidenced by the Stuxnet worm attack [24], the experiments on automotive security [9, 26], and the Maroochy water breach [49]. These attacks come from various cyber and physical surfaces, and are difficult to defend against. For example, in automotive systems, the attacks may come from the On Board Diagnostics (OBD) port, bluetooth communication interface, key-less entry system, multi-media system, and so on [9].

Fault tolerance with respect to soft errors has also become a major concern. The number of soft errors in systems is rapidly increasing, due to the continuous scaling of technology, high energy cosmic particles and radiation from the application environment [5, 53]. In many systems, it is difficult to detect and correct those errors with limited computation and communication resources.

Design metrics such as control performance, fault tolerance, and security often put conflicting requirements on software synthesis [13, 14, 21, 57, 58]. In [14], it is shown that shorter sampling periods lead to better control performance, but may result in worse schedulability. In [58], it is shown that encrypting more messages can enhance security but may worsen control performance and schedulability. Therefore, it is challenging and yet critical to address these design metrics in a holistic fashion across the entire synthesis flow.

Next, we will introduce our model-based software synthesis flow that addresses the above challenges.

9.2 Model-Based Software Synthesis Flow

Model-Based Design (MBD) has been increasingly adopted for designing complex cyber-physical systems and embedded systems. It uses models throughout the production process including design, simulation, code generation, and verification [3, 38]. In particular, the usage of formal or semi-formal models in MBD enables early validation and verification, and also provides a starting point for automated software synthesis. However, for many CPS, such automated flow either does not exist yet or is not mature (i.e., lacks sufficient design space exploration, trade-off, and optimization).

Our model-based software synthesis facilitates the MBD paradigm, and provides an automated flow that starts with formal functional models and conducts holistic task generation and mapping. Our approach also leverages the *Platform-Based*

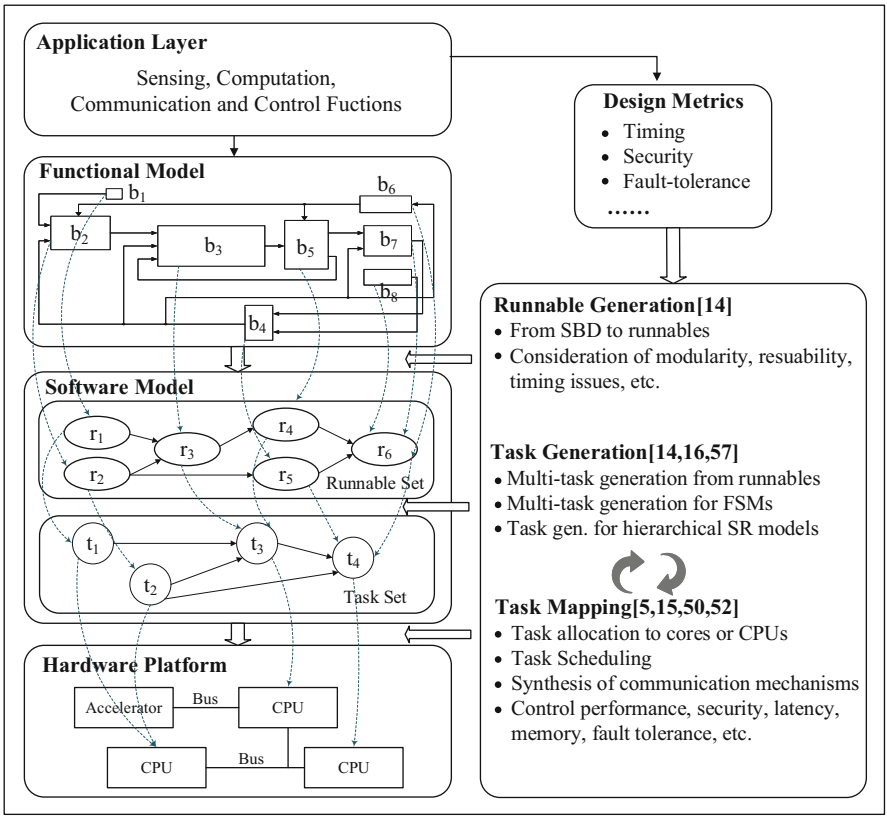


Fig. 9.2 Our model-based software synthesis flow

Design paradigm, which initially separates the modeling of functionality and architecture and then maps the functional model onto the architectural platform through design space exploration [15, 25, 42, 45, 56, 61–63].

We have been targeting automotive software as the main application domain. Our model-based synthesis flow is shown in Fig. 9.2, and goes across multiple system layers as explained in below.

- *Application Layer:* Our flow starts with the application layer that defines system functionality (e.g., various sensing, computation, communication and control functions), often in an informal fashion. A variety of design objectives and constraints are also defined and propagated to lower layers for driving the design process.
- *Functional Models:* The layer of functional models captures the application with formal or semi-formal syntax and semantics. In our work, we consider Synchronous Reactive (SR) models, which dominate current practices for modeling control-centric applications in domains such as automotive and avionics (the

popular Simulink/Stateflow toolset [48] and SCADE suite [47] are based on the SR semantics). In SR models, synchronized processes communicate with each other and react to events [13]. More specifically, an SR model is typically represented by a network of functional blocks in a synchronous block diagram (SBD), where some blocks may be macro blocks that include lower-level blocks and captured by lower-level SBDs. Functional blocks can be of two types: dataflow blocks or extended finite state machine (FSM) blocks. Certain timing behavior, such as worst-case execution time and activation periods of functional blocks, can be captured within the SR models.

- *Software Models:* In our synthesis flow, software tasks are automatically generated from the functional models in the task generation step, with respect to various constraints and optimization objectives. Furthermore, in automotive systems that follow the AUTOSAR [4] standard, there is another layer of *runnables* between functional models and tasks. Following AUTOSAR, Tier-1 suppliers (e.g., Bosch, Denso, Continental) will develop software functions and deliver them to OEMs (e.g., Toyota, General Motors, Volkswagen) in the form of runnables—this is the runnable generation step. From the runnables, the OEMs will then generate tasks and implement them on the hardware platform. In our synthesis flow, we developed formalisms and algorithms for runnable generation from the SR models (in particular the SBDs) and for task generation from runnables. More details will be introduced later in the chapter.
- *Hardware Platform:* The layer of hardware platform can be captured by architectural modeling frameworks such as the Architecture Analysis and Design Language (AADL) or by customized formalism to facilitate the mapping of tasks onto the platform. We need to capture the capabilities and constraints of various computation components (e.g., single- or multi-core ECUs, GPUs, or FPGAs) and the communication network linking them (e.g., CAN, Time-Triggered Ethernet, FlexRay). Given the software tasks (from the task generation step) and the hardware architectural platform, our synthesis flow conducts task mapping that explores task to ECU/core allocation, message to bus allocation, task and message scheduling, etc.

Our model-based synthesis flow provides the following capabilities that are critical for designing safe, secure, efficient, and reliable CPS software.

- *Holistic Timing Consideration:* Our synthesis flow holistically addresses system timing behavior across runnable generation, task generation, and task mapping. We have developed unified mathematical formalism to capture the timing behavior of functional blocks, runnables, and software tasks. Such holistic consideration enables us to start addressing timing-related constraints and trade-offs from the early design stages (as early as runnable generation) and quantitatively analyze timing across the entire synthesis process. More details will be introduced later in the chapter.
- *Multi-Objective Optimization:* Across our synthesis flow, we formulate and solve a series of multi-objective optimization problems to explore runnable generation, task generation, and task mapping, with respect to the constraints and objectives

on multiple design metrics such as performance, schedulability, security, and fault-tolerance. This capability to quantitatively optimize and trade off multiple objectives is critical for ensuring design feasibility and quality.

- *Cross-Layer Codesign.* As our synthesis flow addresses design variables, constraints, and objectives across multiple system layers, it formulates cross-layer synthesis problems. This capability is particularly important for CPS, since design aspects/objectives at different layers are often closely related and have to be codesigned in a holistic formulation. One example is that shorter controlling period may improve control performance at the functional layer, but worsen the schedulability at software and hardware layers. Such trade-off has to be quantitatively analyzed in a cross-layer codesign formulation.

Next, we will introduce more details of our synthesis flow, and demonstrate these three capabilities it has.

9.3 Holistic Timing-Driven Synthesis

In [13], we apply our synthesis flow to automotive software development that follows the AUTOSAR standard. Starting from SR functional models, our software synthesis flow generates runnables from functional blocks in SBDs, generates software tasks from runnables, and then maps tasks onto computation resources. The synthesis results should satisfy a variety of timing constraints such as schedulability deadlines and end-to-end path latency deadlines, ensure the correct execution order and data dependency, and address multiple objectives including modularity, reusability, code size, path latency, etc.

One key focus and novelty of our approach in [13] is to holistically address timing throughout runnable generation, task generation and task mapping stages. This is a challenging goal. In particular, at the runnable generation stage, system timing behavior and schedulability cannot be accurately measured since task-level decisions have not been made yet. To address this challenge and enable holistic timing analysis, we propose a unified mathematical formalism called Firing and Execution Time Automaton (FETA) to capture the worst-case execution time (WCET) of each activation of a periodic runnable or a periodic task (we assume period executions for both in our model). Furthermore, we define an *approximate* schedulability metric based on FETAs, called alpha ratio α, to estimate the impact of runnable generation on system schedulability.

Utilizing the definitions of FETA and alpha ratio, we developed a top-down method and a bottom-up method for runnable synthesis, and then a heuristic algorithm and a simulated annealing based algorithm for task generation and mapping.

Figure 9.3 shows the entire synthesis flow to the left, from SBDs to generated runnable set and to the final task implementation. The diagrams to the right illustrate an example: The top right figure shows an SBD that consists of five

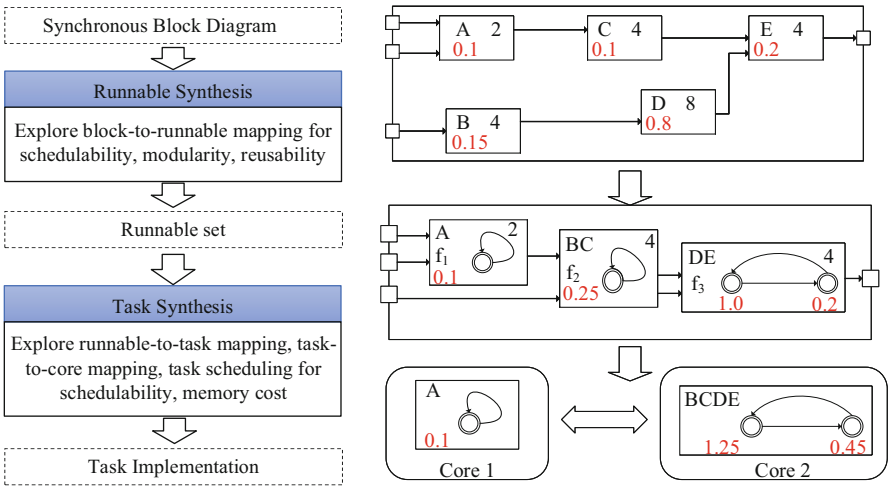


Fig. 9.3 Our holistic timing-driven synthesis flow for automotive software that follows AUTOSAR standard

functional blocks. The middle right figure shows a runnable generation result with three runnables, each with its FETA shown. The bottom right figure shows a task implementation result, with two tasks generated and mapped onto two ECU cores (their FETAs are also shown). Next, we will explain more details of our system model, the FETA formulation, and our synthesis algorithms.

System Model A hierarchical system may consist of a set of subsystems (components) $\mathscr{S} = \{C_j\}$, each of which can be captured by an SBD in SR semantics. Each component C_j consists of a network of functional blocks $B_j = \{b_1^j, b_2^j, \ldots, b_{m^j}^j\}$ connected by links representing functional dependencies, a set of inputs $X^j = \{x_1^j, x_2^j, \ldots, x_{p^j}^j\}$, and a set of outputs $Y^j = \{y_1^j, y_2^j, \ldots, y_{q^j}^j\}$. Each functional block b_i^j has an activation period t_i^j and a WCET γ_i^j. In the top-right of Fig. 9.3, a component (SBD) with five blocks is shown. The block periods are shown in the top-right corner and the WCETs are in the bottom-left corner.

In a runnable view as shown in the middle-right of Fig. 9.3, the implementation of each component C_j consists of a set of runnables. Each runnable is a code implementation of a subset of blocks in component C_j. The union of all the runnable blocksets should cover all the blocks in C_j. Blocks may be allocated to more than one runnables under the condition that blocks implemented by multiple runnables is only executed once by the runnable executing first. Each runnable r_k^j has a base period which is the greatest common divisor of all the blocks implemented in the runnable.

In a task view as in the bottom-right of Fig. 9.3, all the runnables need to be mapped to tasks for execution. Each task also has a period, activation offset and is scheduled according to its priority. The WCET of task τ_i^p is denoted as ω_i^p.

Firing and Execution Time Automaton (FETA) The concept of FETA is an extension to the Firing Time Automaton (FTA) [36], which describes activation events for components that result from the union of synchronously-activated periodic systems. FTA defines a base period, a set of vertices, a set of firing vertices, and transitions between vertices. However, FTA description is not sufficient for capturing runnable timing. FETA is thus proposed to represent the requested execution time at each runnable activation instant. An FETA $\mathscr{A}$ is a pair $\mathscr{A} = (\theta, S)$, where θ is the base period and $S = (V, v_0, F, \delta)$. $V = \{v_0, v_2, \ldots, v_n\}$ is the set of vertices, where v_0 is the initial vertex and each vertex v_i is associated with a WCET specification list of the type $W_p = \{(c_{p0}, -), (c_{pk}, r_k)*\}$. $(c_{p0}, -)$ is the first unconditional WCET while the other WCETs in the list are conditional to the previous execution of the runnable r_k. $F \in V$ is a subset of firing vertices indicating a point in time when one or more blocks are triggered. $\delta : v_{i-1} \rightarrow v_i$ is a link of transition.

The middle-right of Fig. 9.3 shows a sample runnable generation of the SBD above. Functional block A is mapped to runnable f_1; blocks B and C are mapped to runnable f_2; and blocks D and E are mapped to runnable f_3. Take runnable f_3 as an example. Its FETA $\mathscr{A}_3$ contains two vertices. At the first vertex, blocks D and E are both executed and the unconditional WCET of the first vertex is $0.2 + 0.8 = 1.0$. Note that the base period of runnable f_3 is 4; the activation period of E is also 4; while the period of E is 8. Therefore, at the second vertex that represents the second activation of the runnable, only E is executed and the unconditional WCET is 0.2. After that, the FETA transitions back to the first vertex, which means the next runnable activation again will execute both D and E. Such timing behavior repeats and can be captured accurately by FETA.

Optimization Constraints and Objectives Various constraints need to be met during the synthesis process to ensure functional correctness. For runnable generation, the execution order between blocks and activation constraints should be considered. At task synthesis level, the runnable executed by a task must guarantee the synchronous assumption, that is, the amount of execution time requested at each activation must complete before the arrival of any following event associated with the runnable FETA.

The objectives being considered at runnable generation include schedulability (measured by the approximated schedulability metric alpha ratio as stated before), modularity, code size, and reusability. A runnable generation hides information of the internal block structure if the number of runnables (and their dependencies) is significantly smaller than the number of internal blocks. Thus, modularity is measured by the number of runnables generated. An optimal modularity can be obtained by simply implementing all blocks as a single task. Such a solution also provides the minimum code size, however may introduce false input–output dependencies. A runnable generation is reusable if it does not introduce any false input–output dependencies. The trade-offs among these objectives are addressed in runnable generation.

The objectives being considered at task generation and mapping include exact schedulability and memory cost. To measure schedulability, we have to evaluate whether response time of each task τ_i exceeds its deadline. The response time of a task can be calculated as following, where the second term represents the preemption from higher priority tasks.

$$r_{\tau_i} = \omega_{\tau_i} + \sum_{\tau_j \in hp(\tau_i)} \left\lceil \frac{r_{\tau_i}}{T_{\tau_j}} \right\rceil \omega_{\tau_j} \tag{9.1}$$

Furthermore, to ensure system safety and performance, there may be deadline constraints on end-to-end latency of functional paths. For more details of the various design constraints and objectives, please refer to [13].

Synthesis Algorithms Based on the concepts of FETA and alpha ratio, we developed a top-down method and a bottom-up method for runnable synthesis. In the top-down method, we first use a mixed integer linear programming (MILP) formulation to find the runnable generation with optimal modularity while achieving the maximum reusability. Next, a top-down heuristic algorithm gradually decomposes the runnables to improve schedulability until the desired alpha ratio is obtained. At each step, the top-down algorithm finds the runnable that has the maximum local utilization (as defined in [13]) among those that contain blocks with different periods. It then tries to decompose the runnable by moving some of the blocks to a new runnable without introducing cyclic dependencies.

Figure 9.4 shows the results of runnable generation for an industrial example and a number of synthetic examples [13]. The trade-off between modularity and schedulability (measured by approximated metric alpha ratio) is clear.

Note that alpha ratio is only a conservative approximation of the exact schedulability. Based on alpha ratio and FETAs, we can evaluate whether a runnable generation solution is *potentially schedulable* (details in [13]). If it is not, we do

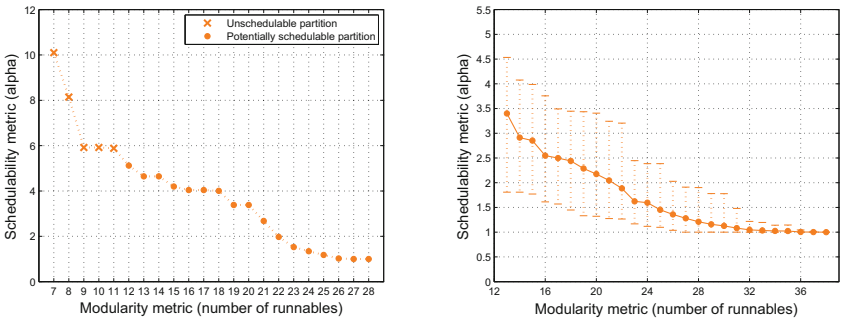


Fig. 9.4 Experimental results for runnable synthesis using top-down method. (**a**) Runnable synthesis for fuel injection example. (**b**) Runnable synthesis for synthetic examples

not even need to consider it for task generation and mapping. In fact, the leftmost points in Fig. 9.4a, b are the solutions with optimal modularity obtained by the MILP formulation, and they are not schedulable based on the alpha ratio and FETAs. The other solutions to the right are results obtained during the top-down decomposition process, and the approximated schedulability (alpha ratio) is indeed being improved. We can see that for the industrial example, the solutions generated after the first five are considered as potentially schedulable. It turns out they are indeed schedulable after task generation and mapping, which indicate that alpha ratio is an effective approximation of the exact schedulability.

The bottom-up method for runnable synthesis starts with the solution that has the best/lowest alpha ratio, and gradually merges runnables to improve modularity.

After runnable generation, we then developed a greedy heuristic algorithm to generate tasks from runnables and map them onto a multi-core platform. The task generation and mapping heuristic first finds the runnables that do not have incoming edges and are placed in an allocation list sorted by their maximum local utilization. Then the heuristic iteratively picks the runnable on top of this list and tries to find a core for its execution. The core is selected based on communication affinity as defined in [13]. When a runnable is successfully allocated, it is removed from the graph and the allocation list. All its outgoing edges in the precedence graph are removed, and a new set of runnables possibly becomes available and are placed in the allocation list. We also developed a simulated annealing based algorithm for task generation and mapping as a comparison to the greedy heuristic. The results can be found in [13] as well.

9.4 Multi-Objective Optimization

Apart from timing, there are other metrics that are also essential in designing cyber-physical systems, such as security and fault-tolerance. These objectives are closely related to timing, as more timing *slacks* provides more room for adding security and fault-tolerance techniques. Therefore, we address these objectives in our synthesis flow, together with timing in a holistic fashion.

9.4.1 Fault-Tolerance

Soft errors have become one of the major concerns of cyber-physical systems. The high energy cosmic particles, the continuous scaling of technology, and the radiations from the environment are some of the sources for soft errors [5, 53]. Soft errors may cause systematic failures, such as application crashes, control flow violations (illegal branches), and silent data corruptions, which increase risks for safety-critical applications.

In order to address soft errors, a lot of techniques have been proposed in the literature. As in [19, 57], we categorize the techniques into embedded error detection (EED) and explicit output comparison (EOC).

- EED refers to the techniques that detect and cover part of the errors with built-in computation overhead. Examples of EED techniques include watchdog timers [39], control flow checking (CFC) [41], instruction signature checking, etc. The state-of-the-art CFC techniques may cover 70% of total errors with about 30% overhead in execution time. In practice, the performance overhead of EED may vary greatly depending on the EED implementation and application [19].
- EOC refers to the techniques that rely on redundant execution to detect and recover errors. For example, running a task twice (with the same input) and comparing the outputs. If the outputs mismatch, an error may have occurred. EOC can achieve almost 100% error detection at the cost of 100% execution time overhead (temporal redundancy) or 100% resource overhead (spatial redundancy). In this chapter, we only focus on temporal redundancy, and more details for spatial redundancy can be found in [57].

Applying EED and EOC techniques can improve the reliability of the system, however, the overhead of EED and EOC may have negative impact on system timing behavior or even fail to meet the real-time constraints (e.g., task deadline, end-to-end latency deadline, etc.). An example of trade-off between EED, EOC, and schedulability is shown in Fig. 9.5. We assume EED error detection rate $\alpha_{\tau_i} = 70\%$ and EOC detection rate $\beta_{\tau_i} = 100\%$. In (a), EOC is deployed for both tasks to achieve higher error coverage, but this solution is infeasible when an error occurs in Task 2 and causes deadline miss. Solution (b) deploys EED for both tasks to satisfy timing constraints, but the detection strength is also weaken. The approximated error coverage of this solution is 73% based on Eq. (9.2). Solution (c) adopted a hybrid solution: EED for Task1 and EOC for Task2. It can recover any single detected error, while achieving a higher error coverage of 91%.

In the rest of the section, we will quantitatively model the impact of EED and EOC techniques on timing and error coverage to ensure system safety.

System Error Coverage We define system error coverage within the hyperperiod T_{hyper} of a task set $\mathscr{T}$ (i.e., the least common multiple of the task periods) where $K \geq 0$ errors may occur. System error coverage is defined as the probability that all errors are either (1) detected and recovered within hyperperiod while all timing constraints are satisfied or (2) happened during idle time [57]. During idle time, the system may still be affected by cosmic ray strikes and memory errors, etc., but we assume the probability to produce erroneous outputs is negligible.[1]

To precisely evaluate the system error coverage in a closed form formulation is difficult, due to the need for the specific error occurrence profile and timing pattern.

[1]Memory is typically well protected. Our formulation can also be extended to address idle-time errors.

	C	T
Task1	0.2	1
Task2	0.45	2

EOC	Λ	Cdec	Crec
Task1	0.02	0.42	0.2
Task2	0.05	0.95	0.45

EED	ΔC	Cdec	Crec
Task1	0.1	0.3	0.3
Task2	0.15	0.6	0.6

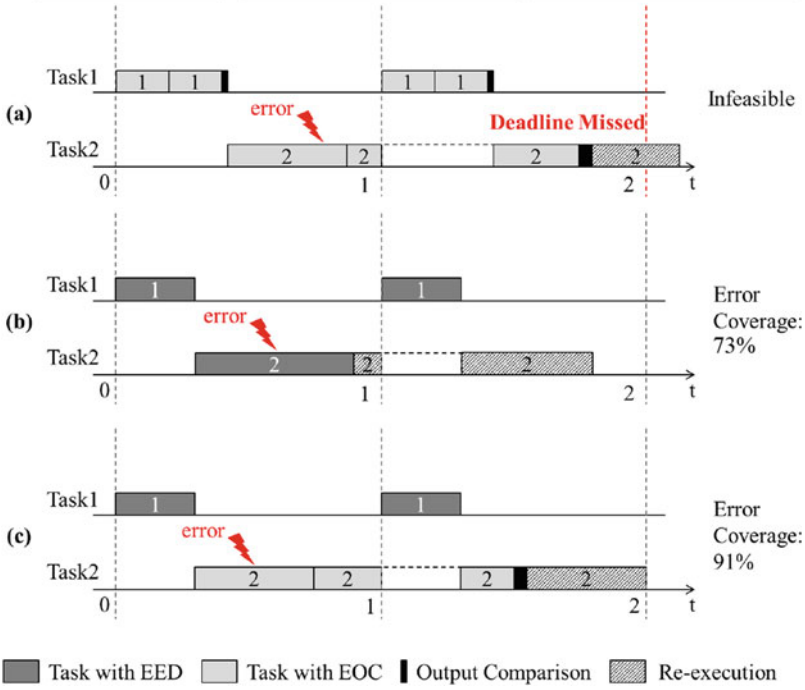


Fig. 9.5 A fault-tolerant design example for single-core CPU platform: (**a**) only EOC is used, (**b**) only EED is used, (**c**) EOC/EED selection

Therefore, we derive an approximated error coverage formulation as below. We use t_{eoc}, t_{eed}, t_{none} to denote the accumulative execution time (including re-execution) needed by tasks employing EOC, EED, and no error detection techniques, and t_{idle} denotes the total idle time. We assume that K arbitrary errors of uniform distribution may occur within a hyperperiod on a single-core ECU. The system error coverage P is then approximated as:

$$P \approx \sum_{i=0}^{K} \sum_{j=0}^{i} \binom{K}{i} \binom{i}{j} \left(\frac{\alpha \cdot t_{eed}}{T_{hyper}}\right)^{j} \left(\frac{\beta \cdot t_{eoc}}{T_{hyper}}\right)^{i-j} \left(\frac{t_{idle}}{T_{hyper}}\right)^{K-i} \tag{9.2}$$

where α and β represent the average error detection rate of EED and EOC, respectively.

Optimization Formulations for Task Mapping For the rest of the section, boolean variables ρ_{τ_i} and o_{τ_i} are used to represent the selection of error detection

mechanism for task τ_i. Specifically, ρ_{τ_i} is 1 if either EOC or EED is used for τ_i, and 0 if neither is used. For tasks using EED or EOC, o_{τ_i} is 1 if EOC is chosen, and 0 if EED is chosen. p_{τ_i,τ_j} is 1 if τ_j has higher priority than τ_i.

Task Execution Time with Error Detection and Error Recovery Time As described at the beginning of Sect. 9.4.1, EED and EOC detect and recover errors with computational overhead. We use $C_{dec_{\tau_i}}$ to denote the *execution time with detection* for task τ_i. If EOC is used for τ_i, $C_{dec_{\tau_i}} = 2C_{\tau_i} + \Lambda_i$, where Λ_i denotes the time for comparing outputs and it is typically much smaller than C_{τ_i}. This represents running a task twice and comparing the outputs. If EED is used, $C_{dec_{\tau_i}} = C_{\tau_i} + \Delta C_{\tau_i}$, where ΔC_{τ_i} is the computation overhead of EED. As shown in the equation, we only need to run the task once, but with the timing overhead.

We use $C_{rec_{\tau_i}}$ to denote the error recovery time for task τ_i and assume the re-execution of a task is scheduled immediately if error(s) is detected. For EOC, $C_{rec_{\tau_i}} = C_{\tau_i} + \Lambda_i$. For EED, $C_{rec_{\tau_i}} = C_{\tau_i} + \Delta C_{\tau_i}$.

Incorporating the boolean variables, we can rewrite $Cdec_{\tau_i}$ and $C_{rec_{\tau_i}}$ as follows.

$$Cdec_{\tau_i} = C_{\tau_i} + \left[o_{\tau_i}(C_{\tau_i} + \Lambda_{\tau_i}) + (1 - o_{\tau_i})\Delta C_{\tau_i}\right]\rho_{\tau_i} \tag{9.3}$$

$$Crec_{\tau_i} = (C_{\tau_i} + o_{\tau_i}\Lambda_{\tau_i} + (1 - o_{\tau_i})\Delta C_{\tau_i})\rho_{\tau_i} \tag{9.4}$$

Worst-Case Fault-Tolerant Task Response Time With error detection and error recover, the original task response time in (9.1) needs to be incorporated with $Cdec_{\tau_i}$ and $C_{rec_{\tau_i}}$. For the single-core system, the worst-case fault-tolerant response time r_{τ_i} is determined by the higher priority task that has the largest $Crec_{\tau_j}$ and all the K errors are located on this task, as shown below in Eq. (9.5).

$$r_{\tau_i} = Cdec_{\tau_i} + K \max_{\tau_j \in \mathscr{T}} \left\{Crec_{\tau_j} p_{\tau_i,\tau_j}\right\} + \sum_{\substack{\tau_k \in \mathscr{T} \\ \wedge \tau_i \neq \tau_k}} \lceil \frac{r_{\tau_i}}{T_{\tau_k}} \rceil Cdec_{\tau_k} p_{\tau_i,\tau_k} \tag{9.5}$$

Optimization Objective To optimize the system error coverage, we further approximate the Eq. (9.2) with a linear formulation in below by (1) setting $K = 1$,[2] and (2) assuming total task re-execution time is negligible compared to total time for regular executions (this is typically true when K is small). The details of the approximation can be found in [57].

$$P \approx 1 - \sum_{\tau_i \in \mathscr{T}} (1 - \varepsilon_{\tau_i}) \frac{Cdec_{\tau_i}}{T_{\tau_i}} \tag{9.6}$$

[2]We also use the same optimization objective K error cases, based on the observation that in practice K is usually very small during the hyperperiod and the amount of time spent on re-execution is also small compared to regular executions.

where ε_{τ_i} is the error detection rate for task τ_i that depends on the choice of error detection mechanisms:

$$\varepsilon_{\tau_i} = \begin{cases} 0 & \text{no detection is used} \\ \alpha_{\tau_i} & \text{EED is used} \\ \beta_{\tau_i} & \text{EOC is used} \end{cases} \tag{9.7}$$

The optimization problem for single-core case can be formulated as follows. The first constraint requires that the response time of each task should be within its deadline. The second constraint enforces that the end-to-end latency on each path should not exceed the deadline for each path. The message response time is modeled as a small constant in single-core case, as the signal is transmitted through memory.

$$\text{maximize: } P \approx 1 - \sum_{\tau_i \in \mathscr{T}} (1 - \varepsilon_{\tau_i}) \frac{Cdec_{\tau_i}}{T_{\tau_i}} \quad \text{s.t.} \tag{9.8}$$

$$\forall \tau_i \;\; r_{\tau_i} = Cdec_{\tau_i} + K \max_{\tau_j \in \mathscr{T}} \left\{ Crec_{\tau_j} p_{\tau_i,\tau_j} \right\} + \sum_{\substack{\tau_k \in \mathscr{T} \\ \wedge \tau_i \neq \tau_k}} \lceil \frac{r_{\tau_i}}{T_{\tau_k}} \rceil Cdec_{\tau_k} p_{\tau_i,\tau_k} \leq T_{\tau_i} \tag{9.9}$$

$$\forall p \;\; l_p = \sum_{\tau_i \in p} (r_{\tau_i} + T_{\tau_i}) + \sum_{s_i \in p} (r_{s_i} + T_{s_i}) \leq D_p \tag{9.10}$$

Fault-Tolerant Task Generation for Single-Rate Systems Given a set of blocks $\mathscr{B} = \{b_0, b_1, b_2 \ldots b_k\}$ in the functional model, we try to map these blocks into a set of tasks $\mathscr{T} = \{\tau_0, \tau_1, \tau_2 \ldots \tau_n\}$, and assign each task an error detection technique so that the error coverage is maximized. In this section, only single rate models (all the blocks having the same period) are considered.

If there is no error detection, the execution time of a task (C_{τ_i}) is the time taken to execute a task. As Eq. (9.11) shows, C_{τ_i} will be the sum of execution time of blocks in τ_i. Boolean variable $f_{b_i,\tau_j} = 1$ indicating that block b_i is mapped to τ_j.

$$C_{\tau_i} = \sum_{b_k \in \mathscr{B}} f_{b_k,\tau_i} C_{b_k} \tag{9.11}$$

The causality of an SBD is the dependencies between blocks inside. In a real-time model, a task will execute only if its inputs have completed their works. We use Q_{b_i,b_j} to indicate that b_j depends on b_i (b_i is an input of b_j), and use q_{τ_m,τ_n} to indicate that τ_n depends on τ_m (τ_m is an input of τ_n). Tasks will inherit dependencies from blocks inside. There must not be any cycle between tasks.

When mapping blocks to tasks, constraints (9.12)–(9.14) must be followed in order to avoid dependency cycle between tasks (i.e., if B depends on A and C depends on B, then we cannot group block A and C into one task).

Table 9.1 System error coverage under different number of cores for the industrial example (multicore)

Core #	≤ 4	5	6	7	8	9
Coverage	Infeasible	0.614	0.774	0.876	0.959	0.961

$$q_{\tau_m,\tau_n} \geq f_{b_i,\tau_m} + f_{b_j,\tau_n} + Q_{b_i,b_j} - 2 \tag{9.12}$$

$$q_{\tau_m,\tau_n} \geq q_{\tau_m,\tau_l} + q_{\tau_l,\tau_n} - 1 \tag{9.13}$$

$$q_{\tau_m,\tau_n} + q_{\tau_n,\tau_m} \leq 1 \tag{9.14}$$

Since we only consider single rate models, the tasks are executed according to causalities between the tasks. In this case, if we apply topological sort on $\mathscr{T}$, as long as the last task can be run successfully within deadline, there is no timing violation. Therefore, the equation can be simplified as below:

$$\sum_{\tau_i \in \mathscr{T}} Cdec_{\tau_i} + K \max_{\tau_i \in \mathscr{T}} \{Crec_{\tau_i}\} \leq Period \tag{9.15}$$

Industrial Case Study and Synthetic Examples The industrial case study is derived from an experimental vehicle that incorporates advanced active safety functions. The subsystem has 41 tasks, among which 16 tasks are critical ones and are assumed to require EOC for higher soft error coverage, and others may use either EOC or EED. The tasks communicate through 81 signals, and there are 171 paths with deadlines ranging from 100 to 300 ms. We can see from Table 9.1 that the error coverage improvement slows down when the number of cores reaches 8.

We also speed up the ECU by a factor between 1.0 to 2.0. For original ECU speed (factor is 1.0), our study shows that at least 4 ECU cores are needed to generate a feasible design. If we increase the number of cores to 8, the system error coverage can reach 98.4% under one error. If the ECU is 2 times faster, only 2 cores are required for a feasible design and only 4 cores are needed to reach nearly 100% error coverage under one error.

We use TGFF to generate synthetic examples with 30–50 tasks each, and are assigned with random periods and WCETs. The comparison among the five methods under different original core utilizations ($\sum_{\tau_i} C_{\tau_i}/T_{\tau_i}$) for a single-core platform under one error is shown in Fig. 9.6. This figure shows that combining EED and EOC in the synthesis flow output the best error coverage rate.

9.4.2 Security

In this subsection, we use automotive system as an example to explain how we consider security in software synthesis, i.e., building a security-aware synthesis flow.

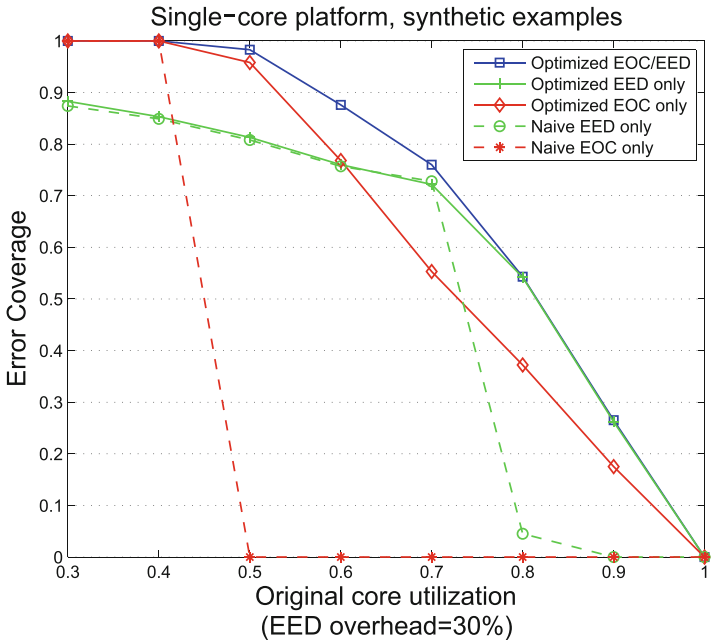


Fig. 9.6 Comparison of various fault-tolerance optimization methods for synthetic examples on a single-core platform

In [26], the authors successfully compromised a real vehicle by hacking into its engine control system, brake control system, and other electronic components. The attacks are conducted through internal CAN buses via packet sniffing, targeted probing, fuzzing, and reverse engineering. The venerabilities of CAN buses are discussed in a number of studies [26, 31, 50, 54]. The major issues of CAN buses include (1) every message is broadcasted, thus enabling eavesdropping by other components connected to the same bus; (2) the scheduling of messages is based on static priorities, which makes it fragile to denial-of-service attacks; and (3) the limited resource makes it difficult to add existing security protection mechanisms.

Take the example shown in Fig. 9.7, if messages have no authentication, a malicious node can easily masquerade the trusted node and send a“Lock brake" message to brake. However, if we are able to add message authentication codes (MACs) to the messages and assign a same symmetric key between the brake and the trusted node, it will be much more difficult for the malicious node to conduct masquerade attacks.

Adding security mechanisms should be addressed together with other timing-related objectives (in particular schedulability), since they may significantly affect system timing by introducing computation overhead for authentication and communication overhead on CAN buses with added fields for authentication.

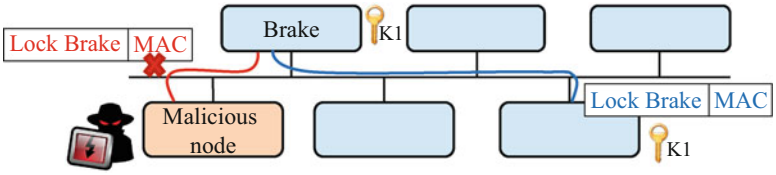


Fig. 9.7 The security challenges in automotive systems

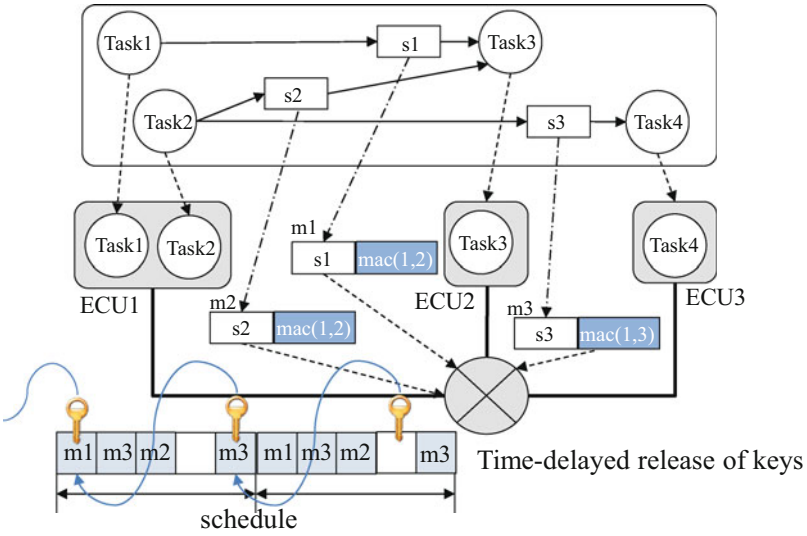


Fig. 9.8 Security-aware mapping for TDMA-based systems

In [32, 35], we developed a security-aware mapping framework for CAN-based systems, where we explore adding MACs to CAN messages together with task allocation and scheduling. More specifically, we explore how to add MACs to individual messages, how many bits should be used for MACs, and how to assign key sharing groups (details in [32]). We formulate these decisions together with the choices for task allocation, task scheduling, signal packing, and message scheduling in an integrated formulation. The formulation ensures that both security and timing requirements are met.

We observed from our work in [32] that even with the holistic consideration of security in the synthesis process, it is still very challenging to add security protection mechanisms to CAN buses due to their limited bandwidth and payload sizes. Fortunately, the automotive industry has been actively developing next-generation in-vehicle communication protocols. In [33, 34], we developed a framework for security-aware mapping for next-generation TDMA-based in-vehicle communication protocols such as Time-Triggered Ethernet, as shown in Fig. 9.8.

The major difference between TDMA-based protocols and CAN is that the message transmission is based on pre-defined time slots, which is better for handling denial-of-service attacks and also enables us to use a more efficient message authentication mechanism with delayed release of keys [33]. We explore the design of such security mechanism (in particular the choice of key-release interval) together with task allocation, task scheduling, and network scheduling. From our experiments, we found that it is more feasible to add security protection mechanisms to these new protocols with faster speed, higher bandwidth, and larger payload. However, we still need to conduct rigorous quantitative analysis in the synthesis process, otherwise the added overhead may still violate timing constraints.

9.5 Cross-Layer Codesign

For various applications for cyber-physical systems, some metrics can only be captured at the application layer. However, these metrics may also largely depend on the implementation from the lower layers. For example, control performance can only be captured at the application layer but it also depends on the sampling period and task scheduling at lower layers. Another example is that the safety properties of vehicle-to-vehicle applications should be captured at the application layer, but the timing-related requirements for safety should drive the software synthesis flow.

In [58], we use the design of secure cyber-physical system to demonstrate the importance of cross-layer codesign in a holistic framework. The adoption of security techniques introduces overhead on computation and communication. In turn, this may affect the system performance, safety, schedulability, and other timing related metrics. To build correct, efficient, and secure cyber-physical systems, we combine control-theoretic methods at the application layer and cybersecurity techniques at the embedded platform layer, and address security together with other design metrics such as control performance under resource and real-time constraints.

The cross-layer codesign problem considering security is shown in Fig. 9.9. In the system, multiple control loops share an embedded platform, with messages transmitted from sensors (vision sensors, GPS, ultrasound, etc.) to controllers and from controllers to actuators. If a message is encrypted, a dedicated *decryption task* is used for decrypting the message and sending the result to the receiving tasks. Such approach is more efficient than decrypting the same message within each receiving task. The attackers may be able to eavesdrop on the communication medium and further reconstruct the system state. This results not only in a loss of privacy, but can further be used as the basis for other malicious attacks.

The key design variables are the sampling period of each control task and the selection of messages for encryption. When the sampling period of a control task increases, its control performance decreases, and platform schedulability becomes easier with less frequent activation of the control task. On the other hand, when the number of messages being encrypted increases, the system security level increases, and platform schedulability becomes harder because of the increased overhead.

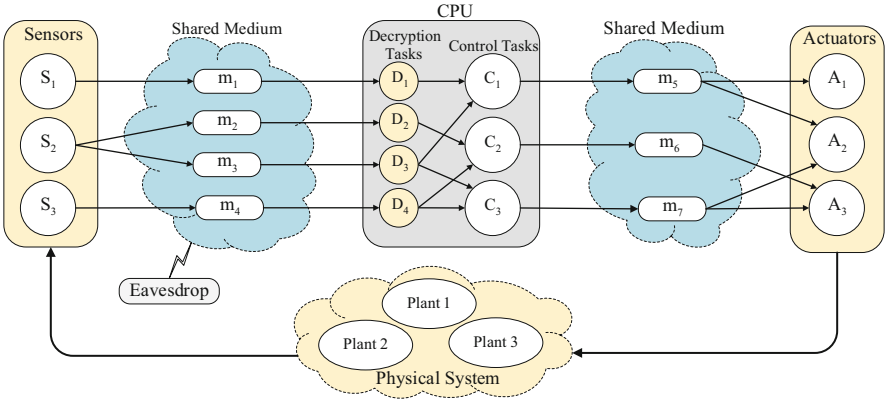


Fig. 9.9 A cyber-physical system with multiple control loops sharing an embedded platform. Attackers may eavesdrop on the communication medium and apply various attacks

Using the formulations in [58], we can quantitatively trade off between control performance and security while guaranteeing the real-time constraints are met.

For applications utilizing vehicle-to-vehicle and vehicle-to-infrastructure communications like cooperative adaptive cruise control (CACC) and autonomous intersection management, our work in [59, 60] analyzes the impact of message delay and message loss on system-level performance and safety properties. We found that to guarantee the system safety and performance, the applications themselves have to be codesigned with the lower-layer computation and communication components.

9.6 Conclusion

Software design has become increasingly challenging and important for cyber-physical systems. In this chapter, we introduced our model-based software synthesis flow that holistically addresses timing throughout the synthesis process; optimizes and trades off multiple design objectives such as performance, security, fault tolerance, schedulability, and modularity; and conducts cross-layer codesign for addressing different cyber and physical aspects. Our synthesis flow has been applied to automotive and transportation domains, and shown its effectiveness in generating correct, efficient, secure, and reliable CPS software implementations.

Acknowledgements This work has been supported by the National Science Foundation grants CCF-1553757, CCF-1646381, and CNS-1646641, and the Office of Naval Research grants N00014-14-1-0815 and N00014-14-1-0816.

References

1. U. Abelein, H. Lochner, D. Hahn, S. Straube, Complexity, quality and robustness - the challenges of tomorrow's automotive electronics, in *2012 Design, Automation Test in Europe Conference Exhibition (DATE)* (2012), pp. 870–871
2. Z. Al-Bayati, Y. Sun, H. Zeng, M. Di Natale, Q. Zhu, B. Meyer, Task placement and selection of data consistency mechanisms for real-time multicore applications, in *Real-Time and Embedded Technology and Applications Symposium (RTAS), 2015 IEEE* (IEEE, New York, 2015), pp. 172–181
3. Automakers opting for model-based design. http://www.designnews.com
4. AUTOSAR. http://www.autosar.org
5. R.C. Baumann, Radiation-induced soft errors in advanced semiconductor technologies. IEEE Trans. Device Mater. Reliab. **5**(3), 305–316 (2005)
6. R. Bosch, CAN specification, version 2.0
7. A. Canedo, J. Wan, A. Faruque, M. Abdullah, Functional modeling compiler for system-level design of automotive cyber-physical systems, in *2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)* (IEEE, New York, 2014), pp. 39–46
8. R.N. Charette, This car runs on code, in *IEEE Spectrum* (2009)
9. S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, T. Kohno et al., Comprehensive experimental analyses of automotive attack surfaces, in *USENIX Security Symposium*, San Francisco, 2011
10. L.M. Clements, K.M. Kockelman, Economic effects of automated vehicles. Transp. Res. Rec. J. Transp. Res. Board **2606**, 106–114 (2017)
11. A. Das, A. Kumar, B. Veeravalli, R. Shafik, G. Merrett, B. Al-Hashimi, Workload uncertainty characterization and adaptive frequency scaling for energy minimization of embedded systems, in *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*. EDA Consortium, San Jose, 2015, pp. 43–48
12. P. Deng, Q. Zhu, M. Di Natale, H. Zeng, Task synthesis for latency-sensitive synchronous block diagram, in *2014 9th IEEE International Symposium on Industrial Embedded Systems (SIES)* (IEEE, Piscataway, 2014), pp. 112–121
13. P. Deng, F. Cremona, Q. Zhu, M.D. Natale, H. Zeng, A model-based synthesis flow for automotive CPS, in *2015 ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)* (2015), pp. 198–207
14. P. Deng, Q. Zhu, A. Davare, A. Mourikis, X. Liu, M.D. Natale, An efficient control-driven period optimization algorithm for distributed real-time systems. IEEE Trans. Comput. **65**(12), 3552–3566 (2016)
15. D. Densmore, A. Simalatsar, A. Davare, R. Passerone, A. Sangiovanni-Vincentelli, Umts mpsoc design evaluation using a system level design framework, in *Design, Automation & Test in Europe Conference & Exhibition, 2009. DATE'09* (IEEE, Piscataway, 2009), pp. 478–483
16. P. Derler, E.A. Lee, A.S. Vincentelli, Modeling cyber-physical systems. Proc. IEEE **100**(1), 13–28 (2012)
17. M. Di Natale, A. Sangiovanni-Vincentelli, Moving from federated to integrated architectures in automotive: the role of standards, methods and tools. Proc. IEEE **98**(4), 603 –620 (2010)
18. J. Diemer, D. Thiele, R. Ernst, Formal worst-case timing analysis of Ethernet topologies with strict-priority and AVB switching, in *2012 7th IEEE International Symposium on Industrial Embedded Systems (SIES)* (IEEE, Piscataway, 2012), pp. 1–10
19. Y. Gao, S.K. Gupta, M.A. Breuer, Using explicit output comparisons for fault tolerant scheduling (FTS) on modern high-performance processors, in *DATE 2013* (2013)
20. Google car: data hog at speeds topping 2700gb per hour. https://goo.gl/9w6LJs

21. L. Guo, Q. Zhu, P. Nuzzo, R. Passerone, A. Sangiovanni-Vincentelli, E. Lee, Metronomy: a function-architecture co-simulation framework for timing verification of cyber-physical systems, in *2014 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)* (2014), pp. 1–10
22. P. Hank, T. Suermann, S. Müller, Automotive Ethernet, a holistic approach for a next generation in-vehicle networking standard, in *Advanced Microsystems for Automotive Applications 2012* (Springer, Berlin, 2012), pp. 79–89
23. K. Jo, J. Kim, D. Kim, C. Jang, M. Sunwoo, Development of autonomous car—part I: distributed system architecture and development process. IEEE Trans. Ind. Electron. **61**(12), 7131–7140 (2014)
24. S. Karnouskos, Stuxnet worm impact on industrial cyber-physical system security, in *IECON 2011-37th Annual Conference on IEEE Industrial Electronics Society* (IEEE, Piscataway, 2011), pp. 4490–4494
25. K. Keutzer, A.R. Newton, J.M. Rabaey, A. Sangiovanni-Vincentelli, System-level design: orthogonalization of concerns and platform-based design. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **19**(12), 1523–1543 (2000)
26. K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham et al., Experimental security analysis of a modern automobile, in *2010 IEEE Symposium on Security and Privacy (SP)* (IEEE, Piscataway, 2010), pp. 447–462
27. E. Lee, Cyber physical systems: design challenges, in *2008 11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing (ISORC)* (2008), pp. 363–369
28. E.A. Lee, Cyber physical systems: design challenges, in *2008 11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing (ISORC)* (IEEE, Piscataway, 2008), pp. 363–369
29. E.A. Lee, The past, present and future of cyber-physical systems: a focus on models. Sensors **15**(3), 4837–4869 (2015)
30. E.A. Lee, S.A. Seshia, *Introduction to Embedded Systems: A Cyber-Physical Systems Approach*, Lee & Seshia (MIT Press, Cambridge, 2011)
31. C.-W. Lin, A. Sangiovanni-Vincentelli, Cyber-security for the Controller Area Network (CAN) communication protocol, in *2012 International Conference on Cyber Security (CyberSecurity)* (IEEE, Piscataway, 2012), pp. 1–7
32. C. Lin, Q. Zhu, C. Phung, A. Sangiovanni-Vincentelli, Security-aware mapping for can-based real-time distributed automotive systems, in *2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)* (2013), pp. 115–121
33. C. Lin, Q. Zhu, A. Sangiovanni-Vincentelli, Security-aware mapping for TDMA-based real-time distributed systems, in *2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)* (2014), pp. 24–31
34. C. Lin, B. Zheng, Q. Zhu, A. Sangiovanni-Vincentelli, Security-aware design methodology and optimization for automotive systems. ACM Trans. Des. Autom. Electron. Syst. **21**(1), 18:1–18:26 (2015)
35. C. Lin, Q. Zhu, A. Sangiovanni-Vincentelli, Security-aware modeling and efficient mapping for CAN-based real-time distributed automotive systems. IEEE Embed. Syst. Lett. **7**(1), 11–14 (2015)
36. R. Lublinerman, S. Tripakis, Modular code generation from triggered and timed block diagrams, in *14th IEEE Real-Time and Embedded Technology and Applications Symposium, RTAS '08* (2008)
37. J.P. MacDuffie, T. Fujimoto, Why dinosaurs will keep ruling the auto industry. Harv. Bus. Rev. **88**(6), 23–25 (2010)
38. Mathworks, Why adopt model-based design for embedded control software development? https://goo.gl/i6itpf
39. G. Miremadi, J. Harlsson, U. Gunneflo, J. Torin, Two software techniques for on-line error detection, in *Twenty-Second International Symposium on Fault-Tolerant Computing, 1992. FTCS-22. Digest of Papers* (1992), pp. 328–335

40. F. Mueller, Challenges for cyber-physical systems: security, timing analysis and soft error protection, in *High-Confidence Software Platforms for Cyber-Physical Systems (HCSP-CPS) Workshop, Alexandria* (2006), p. 4
41. N. Oh, P. Shirvani, E. McCluskey, Control-flow checking by software signatures. IEEE Trans. Reliab. **51**(1), 111–122 (2002)
42. A. Pinto, A. Bonivento, A.L. Sangiovanni-Vincentelli, R. Passerone, M. Sgroi, System level design paradigms: platform-based design and communication synthesis. ACM Trans. Des. Autom. Electron. Syst. **11**(3), 537–563 (2006)
43. R. Poovendran, K. Sampigethaya, S.K. Gupta, I. Lee, K.V. Prasad, D. Corman, J. Paunicka, Special issue on cyber-physical systems [scanning the issue]. Proc. IEEE **100**(1), 6–12 (2012)
44. R. Queck, Analysis of Ethernet AVB for automotive networks using network calculus, in *2012 IEEE International Conference on Vehicular Electronics and Safety (ICVES)* (IEEE, Piscataway, 2012), pp. 61–67
45. A. Sangiovanni-Vincentelli, Quo vadis, SLD? Reasoning about the trends and challenges of system level design. Proc. IEEE **95**(3), 467–506 (2007)
46. A. Sangiovanni-Vincentelli, M. Di Natale, Embedded system design for automotive applications. Computer **40**(10), 42 –51 (2007)
47. SCADE. http://www.esterel-technologies.com/products/scade-suite/
48. Simulink - simulation and model-based design. http://www.mathworks.com/products/simulink/. Accessed: 20 April 2016
49. J. Slay, M.Miller, *Lessons Learned from the Maroochy Water Breach* (Springer, Berlin, 2008)
50. I. Studnia, V. Nicomette, E. Alata, Y. Deswarte, M. Kaâniche, Y. Laarouchi, Survey on security threats and protection mechanisms in embedded automotive networks, in *2013 43rd Annual IEEE/IFIP Conference on Dependable Systems and Networks Workshop (DSN-W)* (IEEE, Piscataway, 2013), pp. 1–12
51. J. Sztipanovits, X. Koutsoukos, G. Karsai, N. Kottenstette, P. Antsaklis, V. Gupta, B. Goodwine, J. Baras, S. Wang, Toward a science of cyber-physical system integration. Proc. IEEE **100**(1), 29 –44 (2012)
52. C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer et al., Autonomous driving in urban environments: Boss and the urban challenge. J. Field Robot. **25**(8), 425–466 (2008)
53. C. Weaver, J. Emer, S.S. Mukherjee, S.K. Reinhardt, Techniques to reduce the soft error rate of a high-performance microprocessor, in *ACM SIGARCH Computer Architecture News*, vol. 32 (IEEE Computer Society, Washington, 2004), p. 264
54. M. Wolf, A. Weimerskirch, C. Paar, Security in automotive bus systems, in *Workshop on Embedded Security in Cars* (2004)
55. S.Ying et al., Foundations for innovation in cyber-physical systems, in *Workshop Report, Energetics Incorporated*, Columbia (2013)
56. W. Zheng, Q. Zhu, M. Di Natale, A.S. Vincentelli, Definition of task allocation and priority assignment in hard real-time distributed systems, in *2007. RTSS 2007. 28th IEEE International Real-Time Systems Symposium* (IEEE, Piscataway, 2007), pp. 161–170
57. B. Zheng, Y. Gao, Q. Zhu, S. Gupta, Analysis and optimization of soft error tolerance strategies for real-time systems, in *2015 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)* (2015), pp. 55–64
58. B. Zheng, P. Deng, R. Anguluri, Q. Zhu, F. Pasqualetti, Cross-layer codesign for secure cyber-physical systems. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **35**(5), 699–711 (2016)
59. B. Zheng, C.-W. Lin, H. Yu, H. Liang, Q. Zhu, CONVINCE: a cross-layer modeling, exploration and validation framework for next-generation connected vehicles, in *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)* (2016)
60. B. Zheng, C.W. Lin, H. Liang, S. Shiraishi, W. Li, Q. Zhu, Delay-aware design, analysis and verification of intelligent intersection management, in *2017 IEEE International Conference on Smart Computing (SMARTCOMP)* (2017), pp. 1–8

61. Q. Zhu, P. Deng, Design synthesis and optimization for automotive embedded systems, in *Proceedings of the 2014 on International Symposium on Physical Design* (ACM, New York, 2014), pp. 141–148
62. Q. Zhu, Y. Yang, M. Natale, E. Scholte, A. Sangiovanni-Vincentelli, Optimizing the software architecture for extensibility in hard real-time distributed systems. IEEE Trans. Ind. Inf. **6**(4), 621–636 (2010)
63. Q. Zhu, H. Zeng, W. Zheng, M.D. Natale, A. Sangiovanni-Vincentelli, Optimization of task allocation and priority assignment in hard real-time distributed systems. ACM Trans. Embed. Comput. Syst. **11**(4), 85 (2012)
64. Q. Zhu, P. Deng, M. Di Natale, H. Zeng, Robust and extensible task implementations of synchronous finite state machines, in *Design, Automation Test in Europe Conference Exhibition (DATE), 2013* (2013), pp. 1319–1324

Chapter 10
Compositional Verification for Autonomous Systems with Deep Learning Components

White Paper

Corina S. Păsăreanu, Divya Gopinath, and Huafeng Yu

10.1 Introduction

Autonomy is increasingly prevalent in many applications, ranging from recommendation systems to fully autonomous vehicles, that require strong safety assurance guarantees. However, this is difficult to achieve, since autonomous systems are large, complex systems, that operate in uncertain environment conditions and often use data-driven, machine-learning algorithms. Machine-learning techniques such as deep neural nets (DNN), widely used today, are inherently unpredictable and lack the theoretical foundations to provide the assurance guarantees needed by safety-critical applications. Current assurance approaches involve design and testing procedures that are expensive and inadequate, as they have been developed mostly for human-in-the-loop systems and do not apply to systems with advanced autonomy.

We propose a compositional approach for the scalable verification of learning-enabled autonomous systems to achieve design-time assurance guarantees. The approach is illustrated in Fig. 10.1. The input to the framework is the design model of an autonomous system (this could be given as, e.g., Simulink/Stateflow or prototype implementation). As the verification of the system as a whole is likely intractable we advocate the use of compositional assume-guarantee verification whereby formally defined *contracts* allow the designer to model and reason about learning-enabled components working side-by-side with the other components in the system. These contracts encode the properties *guaranteed* by the component and

C. S. Păsăreanu (✉) · D. Gopinath
Carnegie Mellon University and NASA Ames, Moffett Field, CA, USA
e-mail: corina.pasareanu@west.cmu.edu

H. Yu
Boeing, Chicago, IL, USA

H. Yu et al. (eds.), *Safe, Autonomous and Intelligent Vehicles*,
Unmanned System Technologies, https://doi.org/10.1007/978-3-319-97301-2_10

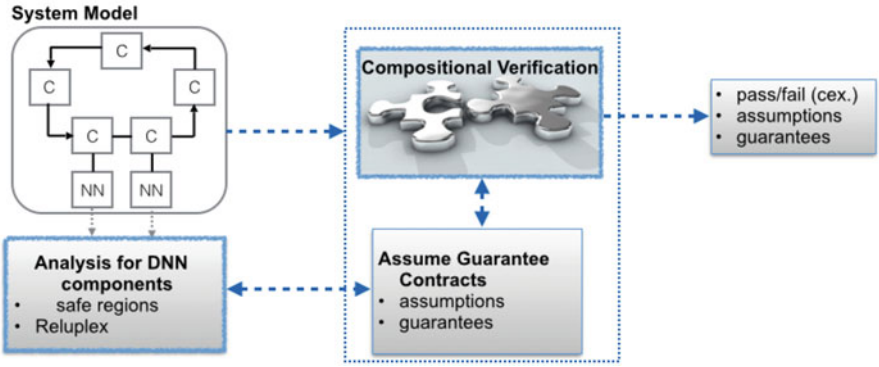


Fig. 10.1 Overview

the environment *assumptions* under which these guarantees hold. The framework will then use compositional reasoning to *decompose* the verification of large systems into the more manageable verification of individual components, which are formally checked against their respective assume-guarantee contracts. The approach enables separate component verification with specialized tools (e.g., one can use software model checking for a discrete-time controller but hybrid model checking for the plant component in an autonomous system) and seamless integration of DNN analysis results.

For DNN analysis, we propose to use clustering techniques to automatically discover *safe regions* where the networks behave in a *predictable* way. The *evidence* obtained from this analysis is *conditional*, subject to constraints defined by the safe regions, and is encoded in the assume-guarantee contracts. The contracts allow us to relate the DNN behavior to the validity of the system-level requirements, using compositional model checking. We illustrate the approach on an example of an autonomous vehicle that uses DNN in the perception module.

10.2 Compositional Verification

Formal methods provide a rigorous way of obtaining strong assurance guarantees of computing systems. There are several challenges to formally modeling and verifying autonomous systems. Firstly, such systems comprise of many *heterogeneous components*; each with different implementations and requirements, which can be addressed best with different verification models and techniques. Secondly, the *state space of such systems is very large*. Suppose we could model all the components of such a system as formally specified (hybrid) models; even ignoring the learning aspect, their composition would likely be intractable. The DNN components make the scalability problem even more serious: for example, the feature space of RGB

1000×600 px pictures for an image classifier used in the perception module of an autonomous vehicle contains $256^{1000\times600\times3}$ elements. Last but not the least, it is not clear how to formally reason about the DNN components as there is no clear consensus in the research community on a *formal definition of correctness for the underlying machine learning algorithms*.

We propose a compositional assume-guarantee verification approach for the scalable verification of autonomous systems where DNN components are working side-by-side with the other components. Compositional verification frameworks have been proposed before to improve the reliability and predictability of CPS [1, 4, 5, 18], but none of these works address systems that include DNN components. Recent work [6] proposes a compositional framework for the analysis of autonomous systems with DNN components. However, that approach addresses *falsification* in such systems and, while that is very useful for debugging, it is not clear how it can be used to provide assurance *guarantees*.

Assume-guarantee reasoning attempts to break up the verification of a large system into the local verification of individual components, using *assumptions* about the rest of the system. The simplest assume-guarantee rule first checks that a component M_1 satisfies a property P under an assumption A (this can be written as $M_1 \models A \rightarrow P$). If the "environment" M_2 of M_1 (i.e., the rest of the system in which M_1 operates) satisfies A (written as $M_2 \models true \rightarrow P$), then we can prove that the whole system composed of M_1 and M_2 satisfies P. Thus we can decompose the global property P into two local assume-guarantee properties (i.e., contracts) $A \rightarrow P$ and A that are expected to hold on M_1 and M_2, respectively. Other, more involved, rules allow reasoning about the circular dependencies between components, where the assumption for one component is used as the guarantee of the other component and vice versa; if the conjunction of the assumptions implies the specification then the overall system guarantees the system-level requirement. Rules that involve circular reasoning use inductive arguments, over time, formulas to be checked, or both, to ensure soundness. Furthermore, the rules can be naturally generalized to reasoning about more than two components and use different notions for property satisfaction such as trace inclusion or refinement checking.

The main challenge with assume-guarantee reasoning techniques is to come up with assumptions and guarantees that can be suitably used in the assume-guarantee rules. This is typically a difficult manual process. Progress has been made on automating assume-guarantee reasoning using learning and abstraction-refinement techniques for iterative building of the necessary assumptions [19]. The original work was done in the context of systems expressed as finite-state automata, but progress has been made in the automated compositional verification for probabilistic and hybrid systems [2, 14], which can be used to model autonomous systems.

Assume-guarantee reasoning can be used for the verification of autonomous systems either by replacing the component with its assume-guarantee specification in the compositional proofs or by using an assume-guarantee rule such as the above to decompose the verification of the systems into the verification of its components. Furthermore, the assume-guarantee specifications can be used to drive

component-based testing and run-time monitoring, in the cases where the design-time formal analysis is not possible, either because the components are too large or they are *adaptive*, i.e. the component behavior changes at run-time (using, e.g., reinforcement learning).

10.3 Analysis for Deep Neural Network Components

Deep neural networks (DNNs) are computing systems inspired by the biological neural networks that constitute animal brains. They consist of neurons (i.e., computational units) organized in many layers. These systems are capable of *learning* various tasks from *labeled examples* without requiring task-specific programming. DNNs have achieved impressive results in computer vision, autonomous transport, speech recognition, social network filtering, bioinformatics, and many other domains and there is increased interest in using them in safety-critical applications that require strong assurance guarantees. However, it is difficult to provide such guarantees since it is known that these networks can be easily fooled by adversarial perturbations: minimal changes to correctly-classified inputs, that cause the network to misclassify them. For instance, in image-recognition networks it is possible to add a small amount of noise (undetectable by the human eye) to an image and change how it is classified by the network.

This phenomenon represents a safety concern, but it is currently unclear how to measure a network's robustness against it. To date, researchers have mostly focused on efficiently finding adversarial perturbations around select individual input points. The goal is to find an input x' as close as possible to a known input x such that x' and x are labeled differently. Finding the optimal solution for this problem is computationally difficult, and so various approximation approaches have been proposed. Some approaches are *gradient based* [7, 8, 20], whereas others use optimization techniques [3]. These approaches have successfully demonstrated the weakness of many state-of-the-art networks; however, these approaches operate on individual input points, and it is unclear how to apply them to large input domains, unless one does a brute-force enumeration of all input values which is infeasible for most practical purposes. Furthermore, because they are inherently incomplete, these techniques cannot even provide any guarantees around the few selected individual points. Recent approaches tackle neural network verification [10, 13] by casting it as an SMT solving problem. Still, these techniques operate best when applied to individual points and further do not have a well-defined rationale to select meaningful regions around inputs within which the network is expected to behave consistently.

In [9], we developed a DNN analysis to automatically discover *input regions* that are likely to be robust to adversarial perturbations, i.e. to have the same true label, akin to finding likely invariants in program analysis. The technique takes inputs

with known true labels from the training set and it iteratively applies a clustering algorithm [12] to obtain small groups of inputs that are close to each other (with respect to different distance metrics) and share the same true label. Each cluster defines a *region* in the input space (characterized by the centroid and radius of the cluster). Our hypothesis is that for regions formed from dense clusters, the DNN is well-trained and we expect that all the other inputs in the region (not just the training inputs) should have the same true label. We formulate this as a safety check and we verify it using off-the-shelf solvers such as Reluplex [13]. If a region is found to be *safe*, we provide *guarantees w.r.t all points within that region*, not just for individual points as in previous techniques.

As the usual notion of safety might be too strong for many DNNs, we introduce the concept of *targeted safety*, analogous to targeted adversarial perturbations [7, 8, 20]. The verification checks *targeted safety* which, given a specific incorrect label, guarantees that no input in the region is mapped by the DNN to that label. Therefore, even if in that region the DNN is not completely robust against adversarial perturbations, we give guarantees that it is safe against specific targeted attacks.

As an example, consider a DNN used for perception in an autonomous car that classifies the images of a semaphore as red, green, or yellow. We may want to guarantee that the DNN will never classify the image of a green light as a red light and vice versa but it may be tolerable to misclassify a green light as yellow, while still avoiding traffic violations.

The safe regions discovered by our technique enable characterizing the input–output behavior of the network over partitions of the input space, which can be encoded in the assume-guarantee specifications for the DNN components. The regions will define the conditions (assumptions), and the guarantees will be that all the points within the region will be assigned the same labels. The regions could be characterized as geometric shapes in Euclidean space with centroids and radii. The conditions would then be in terms of standard distance metric constraints on the input attributes. For instance, all inputs within a Euclidean distance r from the centroid cen of the region would be labeled l by the network.

Note that the verification of even simple neural networks is an NP-complete problem and is very difficult in practice. Focusing on clusters means that verification can be applied to small input domains, making it more feasible and rendering the approach as a whole more scalable. Further, the verification of separate clusters can be done in parallel, increasing scalability even further.

In [9] we applied the technique on the MNIST dataset [16] and on a neural network implementation of a controller for the next-generation Airborne Collision Avoidance System for unmanned aircraft (ACAS Xu) [11], where we used Reluplex for the safety checks. For these networks, our approach identified multiple regions which were completely safe as well as some which were only safe for specific labels. It also discovered adversarial examples which were confirmed by domain experts. We discuss the ACAS Xu experiments in more detail below.

10.3.1 ACAS Xu Case Study

ACAS X is a family of collision avoidance systems for aircraft which is currently under development by the Federal Aviation Administration (FAA) [11]. ACAS Xu is the version for unmanned aircraft control. It is intended to be airborne and receive sensor information regarding the drone (the

ownship) and any nearby intruder drones, and then issue horizontal turning advisories aimed at preventing collisions. The input sensor data includes:

- ρ: distance from ownship to intruder;
- θ: angle of intruder relative to ownship heading direction;
- ψ: heading angle of intruder relative to ownship heading direction;
- v_{own}: speed of ownship;
- v_{int}: speed of intruder;
- τ: time until loss of vertical separation; and
- a_{prev}: previous advisory.

The five possible output actions are as follows: Clear-of-Conflict (COC), Weak Right, Weak Left, Strong Right, and Strong Left. Each advisory is assigned a score, with the lowest score corresponding to the best action. The FAA is currently exploring an implementation of ACAS Xu that uses an array of 45 deep neural networks. These networks were obtained by discretizing the two parameters, τ and a_{prev}, and so each network contains five input dimensions and treats τ and a_{prev} as constants. Each network has 6 hidden layers and a total of 300 hidden ReLU activation nodes. We were supplied a set of cut-points, representing valid important values for each dimension, by the domain experts [11]. We generated a set of 2,662,704 inputs (cartesian product of the values for all the dimensions). The network was executed on these inputs and the output advisories (labels) were verified. These were considered as the inputs with known labels for our experiments.

We were able to prove safety for 177 regions in total (125 regions where the network was completely safe against mis-classification to any label and 52 regions where the network was safe against specific target labels). An example of the safety guarantee is as follows:

$$\forall\, x \in |x - \{0.19, 0.31, 0.28, 0.33, 0.33\}|_{L_1} \leq 0.28 \quad \Rightarrow \quad label(x) = COC \tag{10.1}$$

Here {0.19,0.31,0.28,0.33,0.33} are the normalized values for the five input attributes (ρ,θ,ψ,v_{own},v_{int}) corresponding to the centroid of the region and 0.28 is the radius. The distance is in the Manhattan distance metric (L1). The contract states that under the condition that an input lies within 0.28 distance from the input vector {0.19,0.31,0.28,0.33,0.33}, the network is guaranteed to mark the action for it as COC which is the desired output.

Our analysis also discovered adversarial examples of interest, which were validated by the developers. Figure 10.2 illustrates such an example for ACAS Xu.

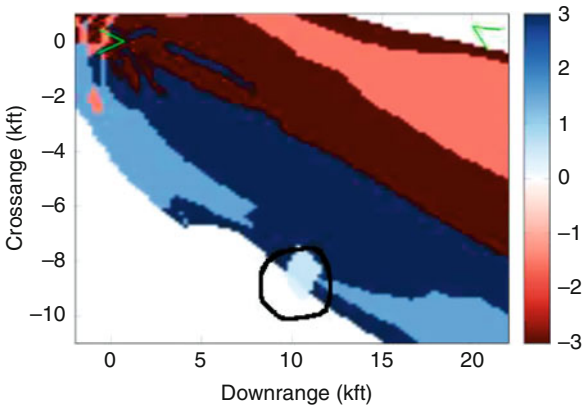


Fig. 10.2 Inputs highlighted in light blue are mis-classified as strong right instead of COC. Crossrange $= \rho \cdot \sin(\theta)$, Downrange $= \rho \cdot \cos(\theta)$

The safety contracts obtained with the region analysis can be used in the compositional verification of the overall autonomous systems, which can be performed with standard model checkers.

10.4 Example

We illustrate our compositional approach on an example of an autonomous vehicle. The platform includes learning components that allow it to detect other vehicles and drive according to traffic regulations; the platform also includes reinforcement learning components to evolve and refine its behavior in order to learn how to avoid obstacles in a new environment.

We focus on a subsystem, namely an automatic emergency breaking system, illustrated in Fig. 10.3. It has three components: the *BreakingSystem*, the *Vehicle* (which, to simplify the presentation, we assume it includes both the autonomous vehicle and the environment), and a *perception module* implemented with a DNN; there may be other sensors (radar, LIDAR, GPS) that we abstract away here for simplicity. The breaking system sends signals to the vehicle to regulate the acceleration and breaking, based on vehicle velocity, distance to obstacles and traffic signals. The velocity information is provided as a feedback from the plant, the distance information is obtained from sensors, while the information about traffic lights is obtained from the perception module. The perception module acts as a classifier over images captured with a camera. Such systems are already employed today in semi-autonomous vehicles where adaptive cruise controllers or lane keeping assist systems rely on image classifiers providing input to the software controlling electrical and mechanical subsystems [6]. Suppose we want to check that the system satisfies the following *safety property*: the vehicle will not enter an intersection if the traffic light at the intersection turns red.

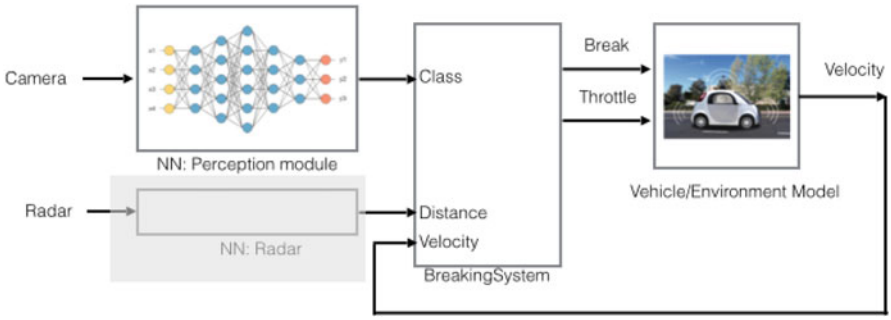


Fig. 10.3 Example

We write the system as the composition: *BreakingSystem||Vehicle||NN*. Each component has an interface that specifies its input and output variables (ports), and their parallel composition is formed by connecting components via ports. We write the property as follows (using Linear Temporal Logic, LTL, assuming discrete time): globally (G) if the semaphore (input image x) is red, then eventually (F), within 3 s, the velocity becomes 0:

$$P :: \quad G((x = red) \quad \Rightarrow \quad F_{T<3s}(velocity = 0))$$

In practice, we would also need to encode in P the assumption that the distance to traffic light is less than some threshold, but we simplify here to ease the presentation. We are thus interested in checking that the system satisfies property P, written as $S \models P$. We decompose the system into two subsystems: $M_1 = BreakingSystem||Vehicle$ and $M_2 = NN$ and define two assume-guarantee contracts C_1 and C_2 for the two subsystems. Suppose (part of) the contract for M_1 is:

$$C_1 :: \quad G((Class = red) \quad \Rightarrow \quad F_{T<3s}(velocity = 0))$$

The contract states that *assuming* the input (Class) to the subsystem M_1 is red then the vehicle is *guaranteed* to stop in at most three time units. We can further decompose the verification of M_1 into the separate verification of its components using additional contracts and perform component-wise verification. It remains to formally characterize the input–output behavior of the DNN in a contract that can be used in the compositional proofs. This is a difficult problem because DNN are known to be vulnerable to adversarial perturbations [15, 20]: a small perturbation added to an image that shows a red semaphore might lead the NN misclassifying it as having $Class = green$.

To address the problem, we use clustering over the training set (see Sect. 10.3) to automatically find regions where the network is likely to be robust to adversarial perturbations. The result is a finite set $\mathcal{R}$ of well-defined regions, where a region $\rho \in \mathcal{R}$ is characterized by a pair (c, r); c is the centroid and r is the radius of the region. We then use a verification tool (such as Reluplex) to check that, *for all*

inputs x within each region, the NN classifies them to the same label as that of known inputs (and of c):

$$C_\rho :: \quad |x - c| < r \quad \Rightarrow \quad Label(x) = Label(c)$$

The training data available and the amount of noise could impact the validity of the check. In such cases we may need to refine the contracts to include Bayesian estimates of uncertainty [17]. Let $Uncert(x)$ denote the uncertainty in the output of the NN for an input x. We can then refine the contract to check that the label is as expected *and* the uncertainty level is below a threshold. The DNN's *safety contract* C_2 could then be the union of all the constraints of the form C_ρ that are proved valid.

We are now ready to perform the compositional proof: if $M_1 \models C_1$ and $M_2 \models C_2$ and furthermore $C_1 \wedge C_2 \Rightarrow P$, it follows that $M_1 || M_2 \models P$; thus, we prove that the whole system satisfies the property, without composing its (large) state space. This proof can be performed with standard model checkers.

10.4.1 Run-Time Monitoring and Control

We note that the evidence we obtain from the analysis is *conditional*; we can only prove that the property holds for the region contracts that we found to be safe. The information encoded in the contract assumptions will need to be used to synthesize *run-time guards* that monitor inputs that fall outside the conditions and instruct the system to take appropriate, fail-safe actions. Note also that this compositional approach enables separate verification of individual components: we can thus replace some of the verification tasks for individual components with testing or simulation, which will increase scalability but will give only empirical guarantees.

Furthermore, if the system contains *adaptive* components, the verification of those components can be done at runtime, whereas the static components only need to be checked once, at design time. Adaptive learning-enabled components pose additional challenges over time. We can again use model uncertainty to identify situations in which the adaptive learning-enabled system is not confident about its decisions, and take appropriate actions in such cases.

10.5 Conclusion

We presented a compositional approach for the verification of autonomous systems. The approach uses assume-guarantee reasoning for scalable verification and can naturally integrate reasoning about the learning-enabled components in the system. We are working on evaluating the proposed approach on various simulation and real

autonomous platforms, including self-driving cars (discussed briefly in Sect. 10.4), autonomous quadcopters, and airplanes. These case studies cover perception, decision making, control and actuation of autonomous systems, and they include safety-critical cyber-physical components as well as DNN components.

References

1. S. Bak, S. Chaki, Verifying cyber-physical systems by combining software model checking with hybrid systems reachability, in *2016 International Conference on Embedded Software, EMSOFT 2016*, Pittsburgh, Pennsylvania, USA, October 1–7, 2016 (2016), pp. 10:1–10:10
2. S. Bogomolov, G. Frehse, M. Greitschus, R. Grosu, C.S. Pasareanu, A. Podelski, T. Strump, Assume-guarantee abstraction refinement meets hybrid systems, in *Proceedings of Hardware and Software: Verification and Testing - 10th International Haifa Verification Conference, HVC 2014*, Haifa, Israel, November 18–20, 2014 (2014), pp. 116–131
3. N. Carlini, D. Wagner, Towards evaluating the robustness of neural networks, in *Proceedings of 38th IEEE Symposium on Security and Privacy* (2017)
4. C. Chilton, B. Jonsson, M.Z. Kwiatkowska, An algebraic theory of interface automata. Theor. Comput. Sci. **549**, 146–174 (2014)
5. C. Chilton, B. Jonsson, M.Z. Kwiatkowska, Compositional assume-guarantee reasoning for input/output component theories. Sci. Comput. Program. **91**, 115–137 (2014)
6. T. Dreossi, A. Donzé, S.A. Seshia, Compositional falsification of cyber-physical systems with machine learning components, in *Proceedings of NASA Formal Methods - 9th International Symposium, NFM 2017*, Moffett Field, CA, USA, May 16–18, 2017 (2017), pp. 357–372
7. R. Feinman, R.R. Curtin, S. Shintre, A.B. Gardner, Detecting adversarial samples from artifacts. Technical Report (2017). http://arxiv.org/abs/1703.00410
8. I.J. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples. Technical Report (2014). http://arxiv.org/abs/1412.6572
9. D. Gopinath, G. Katz, C.S. Pasareanu, C. Barrett, Deepsafe: a data-driven approach for checking adversarial robustness in neural networks, in *Proc. ATVA'18* (2017). https://arxiv.org/abs/1710.00486
10. X. Huang, M. Kwiatkowska, S. Wang, M. Wu, Safety verification of deep neural networks, in *Proceedings of 29th International Conference on Computer Aided Verification (CAV)* (2017), pp. 3–29
11. K. Julian, J. Lopez, J. Brush, M. Owen, M. Kochenderfer, Policy compression for aircraft collision avoidance systems, in *Digital Avionics Systems Conference (DASC)* (2016), pp. 1–10
12. T. Kanungo, D.M. Mount, N.S. Netanyahu, C.D. Piatko, R. Silverman, A.Y. Wu, An efficient k-means clustering algorithm: analysis and implementation. IEEE Trans. Pattern Anal. Mach. Intell. **24**(7), 881–892 (2002)
13. G. Katz, C. Barrett, D. Dill, K. Julian, M. Kochenderfer, Reluplex: an efficient SMT solver for verifying deep neural networks, in *Proceedings of the 29th International Conference on Computer Aided Verification (CAV)* (2017), pp. 97–117
14. A. Komuravelli, C.S. Pasareanu, E.M. Clarke, Assume-guarantee abstraction refinement for probabilistic systems, in *Proceedings of Computer Aided Verification - 24th International Conference, CAV 2012*, Berkeley, CA, USA, July 7–13, 2012 (2012), pp. 310–326
15. A. Kurakin, I. Goodfellow, S. Bengio, Adversarial examples in the physical world, 2016. Technical Report. http://arxiv.org/abs/1607.02533
16. Y. LeCun, C. Cortes, C.J.C. Burges, The MNIST database of handwritten digits. http://yann.lecun.com/exdb/mnist/
17. Y. Li, Y. Gal, Dropout inference in Bayesian neural networks with alpha-divergences, in *International Conference on Machine Learning* (2017), pp. 2052–2061

18. J. Li, P. Nuzzo, A.L. Sangiovanni-Vincentelli, Y. Xi, D. Li, Stochastic assume-guarantee contracts for cyber-physical system design under probabilistic requirements. CoRR (2017). http://arxiv.org/abs/1705.09316
19. C.S. Pasareanu, D. Giannakopoulou, M.G. Bobaru, J.M. Cobleigh, H. Barringer, Learning to divide and conquer: applying the l* algorithm to automate assume-guarantee reasoning. Formal Methods Syst. Des. **32**(3), 175–205 (2008)
20. C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, R. Fergus, Intriguing properties of neural networks, 2013. Technical Report. http://arxiv.org/abs/1312.6199

Index

H. Yu et al. (eds.), *Safe, Autonomous and Intelligent Vehicles*,
Unmanned System Technologies, https://doi.org/10.1007/978-3-319-97301-2