Chapter 1
Cyber-Physical Systems: Overview

Synopsis Cyber-physical systems combine cyber capabilities with physical capabilities to solve problems that neither part could solve alone. This chapter provides an informal introduction to cyber-physical systems, setting the stage for this textbook. The primary purpose is a lightweight overview of the technical and nontechnical characteristics of cyber-physical systems, an overview of some of their application domains, and a discussion of their prospects and challenges. The chapter also informally outlines and explains the approach taken in this book to address crucial safety challenges in cyber-physical systems.

1.1 Introduction

This chapter provides a lightweight introduction to *cyber-physical systems* (*CPS*), which combine cyber capabilities (computation and/or communication as well as control) with physical capabilities (motion or other physical processes).

Note 1 (CPS) *Cyber-physical systems* combine cyber capabilities with physical capabilities to solve problems that neither part could solve alone.

Cars, aircraft, and robots are prime examples, because they move physically in space in a way that is determined by discrete computerized control algorithms that adjust the actuators (e.g., brakes) based on sensor readings of the physical state. Designing these algorithms to control CPSs is challenging due to their tight coupling with physical behavior. At the same time, it is vital that these algorithms be correct, since we rely on CPSs for safety-critical tasks such as keeping aircraft from colliding.

How can we provide people with cyber-physical systems they can bet their lives on? – Jeannette Wing

© Springer International Publishing AG, part of Springer Nature 2018

A. Platzer, *Logical Foundations of Cyber-Physical Systems*,
https://doi.org/10.1007/978-3-319-63588-0_1

Since cyber-physical systems combine cyber and physical capabilities, we need to understand both to understand CPS. It is not enough to understand both capabilities only in isolation, though, because we also need to understand how the cyber and the physics elements work together, i.e., what happens when they interface and interact, because this is what CPSs are all about.

1.1.1 Cyber-Physical Systems Analysis by Example

Airplanes provide a rich source of canonical examples for cyber-physical systems analysis challenges. While they are certainly not the only source of examples, airplanes quickly convey both a spatial intuition for their motion and an appreciation for the resulting challenges of finding out where and how to fly an airplane.

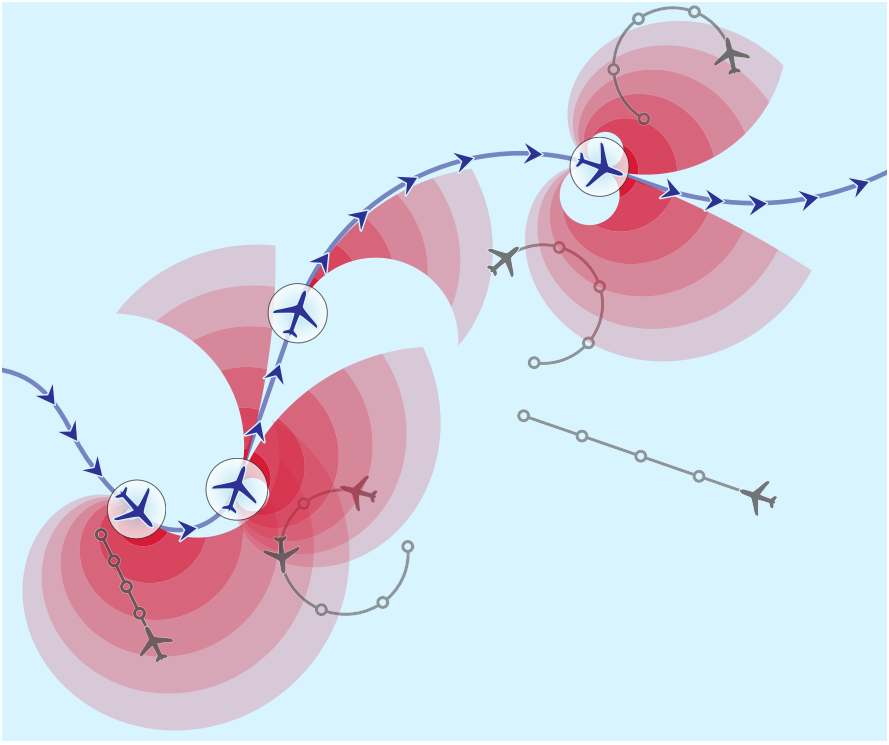


Fig. 1.1 Airplane example: Which control decisions are safe for aircraft collision avoidance?

If a pilot has gotten into a situation where her airplane is too close to other aircraft, see Fig. 1.1, then it would be immensely helpful to give the pilot good advice about how best to maneuver to resolve the situation. Of course, such advice needs

to be given quickly and safely. There is not enough time to carefully plan out every possible trajectory of the the pilot's *ownship* and of all other *intruder* aircraft, but a quick response is needed right away, which is what computers are good at. But the advice also has to be safe such that it reliably separates the aircraft always under all relevant scenarios of when and how exactly the pilots will respond to the advice. For the ownship (following the thick blue trajectory), Fig. 1.1 gives a schematic illustration of unsafe zones (in shades of red, darker with more imminent danger) resulting from the given intruder aircraft (gray).

More generally, this begs the question of which control decisions are safe for aircraft collision avoidance. How can one predict right away whether given control decisions for the airplane and intruder aircraft are guaranteed to be safe in the future or whether they could possibly lead to a collision? How can a computer control program be designed that reaches safe decisions and gives good advice to pilots sufficiently quickly? What would constitute a safety argument for such a pilot decision support system that justifies why the system always gives safe collision avoidance advice that the pilot can follow confidently?

1.1.2 Application Domains

Cyber-physical systems provide prospects of improved safety and efficiency in numerous application domains [2, 29, 30, 60]. Examples include both fully autonomous self-driving cars and improved driver assistance technology for cars such as lane-keeping assistants or distance-keeping assistants [1, 11, 31, 34], where computer control technology helps people drive cars more safely and more efficiently. Both pilot decision support systems [22, 23, 57, 66] and full autopilots for unmanned drone aircraft fall under this paradigm. In the former, the computer focuses on an advisory rôle where it gives decision support to pilots, who are in charge. But autopilots also automate the flight during certain well-defined phases of the flight, such as in normal cruise flight or during landing. The case of drones provides more comprehensive automation where the computer is in primary control of the drone for extended periods of time and remote pilots limit themselves to only providing certain control decisions every once in a while. Other applications include train protection systems [35, 58], power plants [13], medical devices [25, 30], mobile robots that operate in the vicinity of humans [36, 41], and robotic surgery systems [7, 26]. Autonomous underwater vehicles (AUVs) also need computer control for sustained operation since their operating conditions only provide infrequent opportunities for human intervention. Many other application domains are of relevance, though, because the principle of using computer control to help physical systems is quite general.

1.1.3 Significance

Cyber-physical systems can help in many ways. Computers support the human driver in a car by taking control of the car either in full or partially for a certain period of time. For example, the computer can help prevent accidents by keeping the car in the lane in case the human driver is inattentive and it decelerates when the driver fails to notice that the car in front is braking. Of course, the tricky bit is that the computer needs to be able to reliably detect the circumstances where a correction of the car's trajectory is in order. In addition to the nontrivial challenges of reliably sensing other cars and lanes, the computer needs to distinguish user-intended lane changing from accidental lane departures, for example based on whether the driver signaled a lane change by a turn signal, and apply steering corrections appropriately.

In aerospace, computers can not only support pilots during figurative and literal fair-weather phases of the flight such as cruise flight, but can also help by providing pilots with quick collision avoidance advice whenever two or more aircraft get too close together. Since that is a very stressful situation for the pilots, good advice on how to get out of it again and avoid possible collisions is absolutely crucial. Likewise remote pilots cannot necessarily monitor all flight paths of drones closely all the time, such that computer assistance would help prevent collisions with commercial aircraft or other drones. Besides detection, the primary challenges are the uncertainties of when and how exactly the respective aircraft follow their trajectories and, of course, the need to prevent follow-on conflicts with other aircraft. While already quite challenging for two aircraft, this problem gets even more complicated in the presence of multiple aircraft, possibly with different flight characteristics.

For railway applications, technical safety controllers are also crucial, because the braking distances of trains[1] exceed the field of vision so that the brakes need to be applied long before another train is in sight. One challenge is to identify a safe braking distance that works reliably for the train and track conditions without reducing the expected overall performance by braking too early, which would limit operational suitability. Unlike a maximum use of the conventional service brake, full emergency brakes on a train may also damage the rails or wheels and are, thus, only used in the event of a true emergency.

1.1.4 The Importance of Safety

Wouldn't it be great if we could use computers to leverage the advances in safety and efficiency in the CPS application domains? Of course, the prerequisite is that the cyber-physical systems themselves need to be safe, otherwise the cure might be worse than the disease. Safety is paramount to ensure that the cyber-physical systems that are meant to improve safety and efficiency actually help. So, the key question is:

[1] Heavy freight trains as well as high-speed trains need a braking distance of 2.5 km to 3.3 km.

How do we make sure cyber-physical systems make the world a better place?

Because the world is a difficult place, this is rather a difficult question to answer. An answer needs enough understanding of the world (in a model of the relevant part of the world), the control principles (what control actions are available and what their effect is on the physical world) and their implementation in a computer controller, as well as the requisite safety objectives (what precisely discriminates safe from potentially unsafe behavior). This leads to the following rephrasing [53]:

How can we ensure that cyber-physical systems are guaranteed to meet their design goals?

Whether we can trust a computer to control physical processes depends on how it has been programmed and on what will happen if it malfunctions. When a lot is at stake, computers need to be guaranteed to interact correctly with the physical world.

The rationale pursued in this textbook argues that [53]:

1. Computers would perfectly earn our trust to control physical processes if only they came with suitable guarantees.
2. Safety guarantees require appropriate analytical foundations.
3. A foundational core that is common to *all* application domains is more useful than different mathematics for each area, e.g., a special mathematics for trains.
4. Foundations have already revolutionized the digital parts of computer science and, indirectly, the way our whole society works.
5. But we need even stronger foundations when software reaches out into our physical world, because they directly affect our physical environment.

These considerations lead to the following conclusion:

Because of the impact that they can have on the real world, cyber-physical systems deserve proofs as safety evidence.

As has already been argued on numerous other occasions [2–6, 9, 10, 12, 18, 19, 27, 28, 32, 33, 37–40, 42–45, 60, 63–65, 68], the correctness of such systems needs to be verified, because testing may miss bugs. This problem is confounded, though, because the behavior of a CPS under one circumstance can radically differ from the behavior under another, especially when complex computer decisions for different objectives interact. Of course, due to their involvement in models of reality, the safety evidence should not be limited to proofs alone either, but needs to include appropriate testing as well. But without the generality resulting from mathematical proofs, it is ultimately impossible to obtain strong safety evidence beyond the isolated experience with the particular situations covered by the test data [49, 56, 69]. Even statistical demonstrations of safety by test driving are nearly impossible [24].

1.2 Hybrid Systems Versus Cyber-Physical Systems

While the defining criterion that cyber-physical systems combine cyber capabilities with physical capabilities makes it easy to recognize them in practice, this is hardly a precise mathematical criterion. For the characteristic behavior of a system, it should be mostly irrelevant whether it happens to be built literally by combining an actual computer with a physical system, or whether it is built in another way, e.g., by combining the physical system with a small embedded controller achieving the same performance, or maybe by exploiting a biochemical reaction to control a process.

Indeed, cyber-physical systems share mathematical characteristics, too, which are in many ways more important for our endeavor than the fact that they happen to be built from cyber components and from physical components. While a full understanding of the mathematical characteristics of cyber-physical systems will keep us busy for the better part of this book, it is reasonably straightforward to arrive at what is at the core of all mathematical models of cyber-physical systems. From a mathematical perspective, cyber-physical systems are hybrid systems (or extensions thereof):

Note 2 (Hybrid systems) *Hybrid systems* are a mathematical model of dynamical systems that combine discrete dynamics with continuous dynamics. Their behavior includes both aspects that change discretely one step at a time and aspects that change continuously as continuous functions over time.

For example, the aircraft in Fig. 1.1 fly continuously along their trajectories as a continuous function of continuous time, because real aircraft do not jump around in space with discrete jumps. Every once in a while, though, the pilot and/or autopilot reaches a decision about turning in a different direction to avoid a possible collision with intruder aircraft. These discrete decisions are best understood as a discrete dynamics in discrete time, because they happen one step after another. The system reaches one discrete decision for a collision avoidance course, follows it continuously for a certain period of time, and then re-evaluates the resulting situation later to see whether a better decision becomes possible.

Similarly, a car controller decides to accelerate or brake, which is best understood as a discrete dynamics, because there is a discrete instant of time where that decision is reached and scheduled to take effect. The car's continuous motion down the road, instead, is best understood as a continuous dynamics, because it changes the position as a continuous function of time.

In the most naïve interpretation, the cyber components of cyber-physical systems directly correspond to the discrete dynamics of hybrid systems while the physical components of cyber-physical systems directly correspond to the continuous dynamics of hybrid systems. While possibly a good mental model initially, this view will ultimately turn out to be too simplistic. For example, there are events in physical models that are best described by a discrete dynamics even if they come from the physics. For instance, the touchdown of an airplane on the ground can be considered as causing a discrete state change by a discrete dynamics from flying to driving

even if the runway that the aircraft touches down on is quite physical and not a cyber construct at all. Conversely, for some purposes, some of the computations happen so frequently and so quickly that we best understand them as if they were running continuously even if that is not entirely true. For instance, a digital PID controller[2] for an inner-loop flight controller that quickly adjusts the ailerons, rudders, and elevators of an aircraft can sometimes be considered as having a continuous effect even if it is actually implemented as a digital device with a fast clock cycle.

In fact, this is one of the most liberating effects of understanding the world from a hybrid systems perspective [53]. Since the mathematical principles of hybrid systems accept both discrete and continuous dynamics, we do not have to either coerce all aspects of a system model into the discrete to understand it with discrete mathematics or force all system aspects into a continuous understanding to analyze it with continuous techniques. Instead, hybrid systems make it perfectly acceptable to have some aspects discrete (such as the decision steps of a digital controller) and others continuous (such as continuous-time motion), while allowing modeling decisions about ambivalent aspects. For some purposes, it might be better to model the touchdown of an aircraft as a discrete state change from in-the-air to on-the-ground. For other purposes, such as developing an autopilot for landing, it is important to take a more fine-grained view, because the aircraft will simply take off again if it is still going too fast even if its state changes to on-the-ground. Hybrid systems enable such tradeoffs.

Overall, hybrid systems are *not* the same as cyber-physical systems. Hybrid systems are mathematical models of complex (often physical) systems, while cyber-physical systems are defined by their technical characteristics. Nevertheless, exhibiting a hybrid systems dynamics is such a common feature of cyber-physical systems that we will take the liberty of using the notions cyber-physical system and hybrid system quite interchangeably in Parts I and II of this book.

Despite this linguistic simplification, you should note that hybrid systems can be nontechnical. For example, certain biological mechanisms can be captured well with hybrid system models [65] or genetic networks [17] even if they have nothing to do with cyber-physical systems. Conversely, a number of cyber-physical systems feature additional aspects beyond hybrid systems, such as adversarial dynamics (studied in Part III), distributed dynamics [47], or stochastic dynamics [46].

1.3 Multi-dynamical Systems

Owing to the fact that cyber-physical systems can have more dynamical aspects than just those of hybrid systems, this book follows the more general multi-dynamical systems principle [48, 53] of understanding cyber-physical systems as a combination of multiple elementary dynamical aspects.

[2] Proportional-integral-derivative or PID controllers control a system by a linear combination of the error, the integral of the error over time, and the derivative of the error.

Note 3 (Multi-dynamical system) *Multi-dynamical systems* [48] are mathematical models of dynamical systems characterized by multiple facets of dynamical systems, schematically summarized in Fig. 1.2.

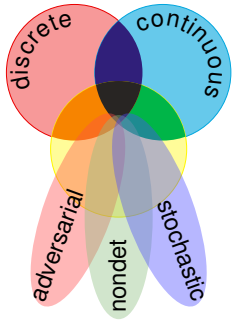


Fig. 1.2 Multi-dynamical systems aspects of CPS

CPSs involve computer control decisions and are, thus, *discrete*. CPSs are also *continuous*, because they evolve along differential equations of motion or of other physical processes. CPSs are often *uncertain*, because their behavior is subject to choices coming from either environmental variability or from intentional uncertainties that simplify their model. This uncertainty can manifest in different ways. Uncertainties make CPSs *stochastic* when good information about the distribution of choices is available [46]. Uncertainties make CPSs *nondeterministic* when no commitment about the resolution of choices is made. Uncertainties make CPSs *adversarial* when they involve multiple agents with potentially conflicting goals or even active competition in a game [52, 55]. Verifying that CPSs work correctly requires dealing with many of these dynamical features at the same time. Sometimes, CPSs require even more dynamical features, such as *distributed* dynamics [47].

Hybrid systems are the special case of multi-dynamical systems that combine discrete and continuous dynamics, and will be considered in Parts I and II. *Hybrid games* are multi-dynamical systems that combine discrete, continuous, and adversarial dynamics that will be studied in Part III. *Stochastic hybrid systems* are multi-dynamical systems that combine discrete, continuous, and stochastic dynamics, but are beyond the scope of this book [8, 46]. *Distributed hybrid systems* are multi-dynamical systems combining discrete, continuous, and distributed dynamics [47].

Multi-dynamical systems study complex CPSs as a combination of multiple elementary dynamical aspects. Throughout this textbook, we will come to appreciate how this approach helps to tame the complexity of CPSs by understanding that their complexity just comes from combining lots of simple dynamical effects with one another. The overall system is quite complex, but each of its pieces is better behaved, since it only has one dynamics as opposed to all of them at once. What miracle translates this *descriptive simplification* of a CPS described as a combination of multiple dynamical aspects into an *analytic simplification* of multiple dynamical systems that can be considered side by side during analysis? The descriptive simplification is a

helpful modeling advantage to disentangle different dynamical aspects of the system into separate aspects of a model. But the biggest impact of multi-dynamical systems is in how they enable an analytic simplification of studying and analyzing the individual dynamical aspects separately while still yielding results about the combined multi-dynamical system. How does the descriptive advantage of a multi-dynamical systems composition carry over to an analytic advantage?

The key to this mystery is to integrate the CPS dynamics all within a single, compositional logic [48, 53]. Compositionality means that the meaning of a construct is a simple function of the meaning of the pieces [61]. For example, the meaning of the logical conjunction operator $\wedge$ (read as "and") is a simple function of the meaning of its pieces. The formula $A \wedge B$ (read as "A and B") is true exactly if A is true and B is true, too. Another way to say this is that the set of states of a system in which formula $A \wedge B$ is true is exactly the intersection of the set of states in which A is true with the set of states in which B is true, because it is this intersection of states in which both A and B are true. This (simple) insight will already enable us to analyze a system separately for the questions of whether A is true and whether B is true in order to find out whether the conjunction $A \wedge B$ is true. Achieving compositionality for other CPS operators is more demanding but equally impactful.

Since compositionality is an intrinsic feature starting from the very semantics of logic [14, 16, 20, 21, 59, 62], logics naturally reason compositionally, too. For example, a proof of the formula $A \wedge B$ consists of a combination of a proof of A together with a proof of B, because the two of those proofs together justify that A and B are both true, which means that $A \wedge B$ is true. This makes it possible to take advantage of the compositionality in the formulas of a logic also when reasoning about formulas in the logic. A proof of $A \wedge B$ decomposes into a proof of the simpler subformula A together with a proof of the simpler subformula B.

With suitable generalizations of logics to embrace multi-dynamical systems [42, 44, 46, 47, 49, 52, 54, 55], this compositionality generalizes to CPS. We "just" need to make compositionality work for the CPS operators, which are, of course, more complicated than a mere logical $\wedge$ operator. Verification works by constructing a proof in such a multi-dynamical systems logic. The whole proof verifies a complex CPS. Yet, each proof step only reasons separately about one dynamical aspect at a time, for example, an isolated discrete assignment or the separate local dynamics of a differential equation, each captured in a separate, modular reasoning principle.

Multi-dynamical systems also impact and simplify the presentation of the Logical Foundations of Cyber-Physical Systems. The compositionality principles of logic and multi-dynamical systems considerably tame the conceptual complexity of CPS by making it possible to focus on one aspect at a time, one chapter after another, without losing the ability to combine the understanding attained for each aspect. This gradual approach effectively conveys the principles for a successful separation of concerns for CPS.

1.4 How to Learn About Cyber-Physical Systems

There are two primary ways of learning about cyber-physical systems.

Onion Model

The *Onion Model* follows the natural dependencies of the layers of mathematics going outside in, peeling off one layer at a time, and progressing to the next layer when all prerequisites have been covered. This would require the CPS student to first study all relevant parts of computer science, mathematics, and engineering, and then return to CPS in the big finale. That would require the first part of this book to cover real analysis, the second part differential equations, the third part conventional discrete programming, the fourth part classical discrete logic, the fifth part theorem proving, and finally the last part cyber-physical systems. In addition to the significant learning perseverance that the Onion Model requires, a downside is that it misses out on the integrative effects of cyber-physical systems that can bring different areas of science and engineering together, and which provide a unifying motivation for studying them in the first place.

Scenic Tour Model

This book follows the *Scenic Tour Model*, which starts at the heart of the matter, namely cyber-physical systems, going on scenic expeditions in various directions to explore the world around as we find the need to understand the respective subject matter. The textbook directly targets CPS right away, beginning with simpler layers that the reader can understand in full before moving on to the next challenge.

For example, the first layer comprises CPSs without feedback control, which allow simple finite open-loop controls to be designed, analyzed, and verified without the technical challenges considered in later layers of CPS. Likewise, the treatment of CPS is first limited to cases where the dynamics can be solved in closed form, such as straight-line accelerated motion of Newtonian dynamics, before generalizing to systems with more challenging differential equations that can no longer be solved explicitly. This gradual development where each level is mastered and understood and practiced in full before moving to the next level is helpful to tame complexity. The Scenic Tour Model has the advantage that we stay on cyber-physical systems the whole time, and leverage CPS as the guiding motivation for understanding more and more about the connected areas. It has the disadvantage that the resulting gradual development of CPS does not necessarily always present matters in the same way that an after-the-fact compendium would treat it. This textbook compensates by providing appropriate technical summaries and by highlighting important results for later reference in boxes, with a list of theorems and lemmas in the table of contents. A gradual development can also be more effective at conveying the ideas, reasons, and rationales behind the development compared to a final compendium.

Besides the substantial organizational impact that this "CPS first" approach has throughout the presentation of this book, the Scenic Tour Model is most easily noticeable in the Expedition boxes that this textbook provides. Every part of this textbook is written in a simple style bringing mathematical results in as needed, and with an emphasis on intuition. The Expedition boxes invite the reader to additionally connect to other areas of science that are of no crucial relevance for the immediate study of CPS nor the remainder of the textbook, but still provide a link to another area, in case the reader happens to be familiar with it or takes this link as an inspiration to explore that other area of science further.

Prerequisites

Even if deliberately light on prerequisites, this textbook cannot start from zero either. Its primary assumptions are some prior exposure to basic programming and elementary mathematics. Specifically, the textbook assumes that the reader has had some prior experience with computer programming (such as what is covered in a first semester undergraduate course taught in any programming language covering if-then-else conditionals and loops).

While Chap. 2 starts out with an intuitive and a rigorous treatment of differential equations and provides a few conceptually important meta-results in its appendix, this book is no replacement for a differential equations course. But it also does not have to be. The concepts required for CPS from differential equations will be picked up and expanded upon at a light pace throughout this textbook. The textbook does, however, assume that the reader is comfortable with simple derivative and differential equation notation. For example, Chap. 2 will discuss how $x' = v, v' = a$ is a differential equation, in which the time-derivative x' of position x equals velocity v, whose time-derivative v' in turn equals the acceleration a. This differential equation characterizes accelerated motion of a point x with velocity v and acceleration a along a straight line.

While a good deal of the interest in this textbook comes from its general applicability, it is also structured to minimize dependency on prerequisites. In particular, Part I of this book can already be understood if the reader is familiar with the differential equation $x' = v, v' = a$ for accelerated motion of point x along a straight line. While Part II provides analytic tools for studying systems with significantly more general differential equations, it is enough to have an intuition for the differential equation $x' = y, y' = -x$ characterizing rotation of the point (x, y) around the origin. Of course, this textbook studies other differential equations in some illustrative examples as well, but those are not on the critical path to understanding the rest of this book.

Most crucially, however, the textbook assumes that the reader has been exposed to some form of mathematical reasoning before (such as *either* in a calculus or analysis course *or* in a matrix or linear algebra course *or* a mathematics course for computer scientists or engineers). The particular contents covered in such a prior course are not at all as important as the mathematical experience itself with mathematical

concept developments and proofs. This textbook develops a fair amount of logic on its own as part of the way of understanding cyber-physical systems. A prior understanding of logic is, thus, not necessary for the study of this book. And, in fact, the *Foundations of Cyber-Physical Systems* undergraduate course that the author teaches at Carnegie Mellon University and on which this textbook is based counts as fulfilling a Logics/Languages elective or Programming Languages requirement without prior background in either.

1.5 Computational Thinking for Cyber-Physical Systems

The approach that this book follows takes advantage of Computational Thinking [67] for cyber-physical systems [50]. Due to their subtleties and the intricate interactions of complex control software with the physical world, cyber-physical systems are notoriously challenging. Logical scrutiny, formalization, and thorough safety and correctness arguments are, thus, critical for cyber-physical systems. Because cyber-physical system designs are so easy to get wrong, these logical aspects are an integral part of CPS design and critical to understanding their complexities.

The primary attention of this book, thus, is on the foundations and core principles of cyber-physical systems. The book tames some of the complexities of cyber-physical systems by focusing on a simple core programming language for CPS. The elements of the programming language are introduced hand in hand with their reasoning principles, which makes it possible to combine CPS program design with their safety arguments. This is important, not just because abstraction is a key factor for success in CPS, but also because retrofitting safety is not possible in CPS.

To simplify matters, the chapters in this textbook are also organized to carefully reveal the complexities of cyber-physical systems in layers. Each layer will be covered in full, including its programmatic, semantic, and logical treatment, before proceeding to the next level of complexity. For example, the book first studies single-shot control before considering control loops, and only then proceeds to systems with differential equations that cannot be solved in closed form. Adversarial aspects are covered subsequently.

1.6 Learning Objectives

The respective learning objectives are identified at the beginning of each chapter, both textually and with a schematic diagram. They are organized along the three dimensions *modeling and control*, *computational thinking*, and *CPS skills*. The most important overall learning objectives throughout this textbook are the following.

Modeling and Control: In the area of *Modeling and Control* (MC), the most important goals are to

- *understand the core principles behind CPS.* The core principles are important for effectively recognizing how the integration of cyber and physical aspects can solve problems that no part could solve alone.
- *develop models and controls.* In order to understand, design, and analyze CPSs, it is important to be able to develop models of the various relevant aspects of a CPS design and to design controllers for the intended functionalities based on appropriate specifications.
- *identify the relevant dynamical aspects.* It is important to be able to identify which types of phenomena of a CPS have a relevant influence for the purpose of understanding a particular property of a particular system. These allow us to judge, for example, when it is important to manage adversarial effects, and when a nondeterministic model is sufficient.

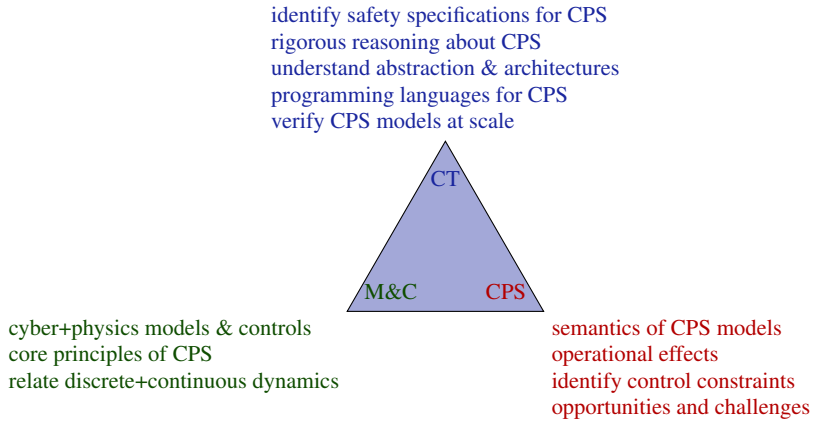
Computational Thinking: In the area of *Computational Thinking* (CT), the most important goals are to

- *identify safety specifications and critical properties.* In order to develop correct CPS designs, it is important to identify what "correctness" means, how a design may fail to be correct, and how to make it correct if it is not correct yet.
- *understand abstraction in system designs.* The power of abstraction is essential for the modular organization of CPSs, and for the ability to reason about separate parts of a system independently. Because of the overwhelming practical challenges and numerous levels of detail, abstraction is even more critical than it already is in conventional software design.
- *express pre- and postconditions and invariants for CPS models.* Pre- and postconditions allow us to capture under which circumstance it is safe to run a CPS or a part of its design, and what safety entails. They allow us to achieve what abstraction and hierarchies achieve at the system level: decompose correctness of a full CPS into correctness of smaller pieces. The fundamental notion of invariants achieves a similar decomposition by establishing which relations of variables remain true no matter how long and how often the CPS runs.
- *use design-by-invariant.* In order to develop correct CPS designs, invariants are an important structuring principle guiding what the control has to maintain in order to preserve the invariant and, thereby, safety. This guidance simplifies the design process, because it applies locally at the level of individual localized control decisions that preserve invariants without explicitly having to take system-level closed-loop properties into account.
- *reason rigorously about CPS models.* Reasoning is required to ensure correctness and find flaws in a CPS design. Both informal reasoning and formal reasoning in a logic are important objectives for being able to establish correctness.
- *verify CPS models of appropriate scale.* This textbook covers the science of how to prove CPSs. You can gain practical experience through its exercises and appropriately scoped projects in the theorem prover KeYmaera X. This experience will help you learn how best to select the most interesting questions in formal verification and validation. Formal verification is not only critical but, given the right abstractions, quite feasible in high-level CPS control designs.

CPS Skills: In the area of *CPS skills*, the most important goals are to

- *understand the semantics of a CPS model.* What may be easy in a classical isolated program becomes very demanding when that program interfaces with effects in the physical world. A precise understanding of the nuanced meaning of a CPS model is fundamental to reasoning, along with an understanding of how it will execute. A deep understanding of the semantics of CPS models is also obtained by carefully relating their semantics to their reasoning principles and aligning them in perfect unison.
- *develop an intuition for operational effects.* Intuition for the joint operational effect of a CPS is crucial. For example, it is crucial to understand what the effect of a particular discrete computer control algorithm will be on a continuous plant.
- *identify control constraints.* An operational intuition guides our understanding of the operational effects and, along with their precise logical rendition, their impact on finding correct control constraints that make a CPS controller safe.
- *understand opportunities and challenges in CPS and verification.* While the beneficial prospects of CPS for society are substantial, it is crucial to also develop an understanding of their inherent challenges and of approaches to minimize the impact of potential safety hazards. Likewise, it is important to understand the ways in which formal verification can best help improve the safety of system designs.

This textbook will give the reader the required skills to formally analyze the cyber-physical systems that are all around us – from power plants to pacemakers and everything in between – so that when you contribute to the design of a CPS, you are able to understand important safety-critical aspects and feel confident designing and analyzing system models. Other beneficial by-products include that cyber-physical systems provide a well-motivated exposure to numerous other areas of mathematics and science in action.



1.7 Structure of This Textbook

This textbook consists of four main parts, which develop different levels of the logical foundations of cyber-physical systems. You are now reading the introduction.

Elementary Cyber-Physical Systems

Part I studies elementary cyber-physical systems characterized by a hybrid system dynamics whose continuous dynamics can still be solved in closed form. Differential equations are studied as models of continuous dynamics, while control programs are considered for the discrete dynamics. Part I investigates differential dynamic logic for specifying properties and axioms for reasoning about CPS. It further investigates appropriate structuring principles for proofs and the handling of control loops via loop invariants, and discusses both event-triggered and time-triggered control. This part provides an extensive introduction to the wonders and challenges of cyber-physical systems, but still isolates most of the reasoning challenges in the search for discrete loop invariants since their differential equations can still be solved explicitly. While enabling interesting and challenging considerations about CPSs, Part I limits the level of interaction and subtlety in their safety arguments. The insights from Part I already enable, for example, a comprehensive study of controllers for safe acceleration and braking of a car along a straight lane.

Differential Equations Analysis

Part II considers advanced cyber-physical systems whose dynamics cannot be solved in explicit closed form. Most crucially, this necessitates indirect forms for analyzing the safety of a CPS, because solutions are no longer available. Based on the understanding of discrete induction for control loops from Part I, Part II develops induction techniques for differential equations. In addition to developing differential invariants as induction techniques for differential equations, this part studies differential cuts that make it possible to prove and then use lemmas about differential equations. It also considers so-called differential ghosts, which can simplify safety arguments by adding extra variables (ghost variables or auxiliary variables) with additional differential equations into the dynamics to balance out generalized energy invariants. Part II is required for handling safety arguments for CPS with nonsolvable dynamics such as robots racing on a circular race track or driving along curves in the plane or for aircraft flying along three-dimensional curves.

Adversarial Cyber-Physical Systems

Part III advances the understanding of cyber-physical systems to cover hybrid games mixing discrete dynamics, continuous dynamics, and adversarial dynamics. Based

on the understanding of hybrid systems models for CPSs from Part I and invariants for differential equations from Part II, Part III shifts the focus to an exploration of hybrid games, in which the interaction of different players with different objectives is a dominant aspect. Unlike hybrid systems, in which all choices are nondeterministic, hybrid games give different choices to different players at different times. Part III is required for handling safety arguments for CPSs in which multiple agents interact with possibly conflicting goals, or with the same goals but possibly conflicting actions resulting from different perceptions of the world.

Comprehensive CPS Correctness

Part IV complements the CPS foundations from the previous parts with an account of what it takes to round out a comprehensive correctness argument for a cyber-physical system. Part IV condenses the logical reasoning principles of CPS from Parts I and II into a completely axiomatic style that makes it easy to implement logical reasoning with an extremely parsimonious logical framework based solely on uniform substitutions. Since the nuances of cyber-physical systems provide ample opportunity for subtle discrepancies, Part IV also investigates a logical way to tame the subtle relationship of CPS models to CPS implementations. The logical foundations of model safety transfer can synthesize provably correct monitor conditions that, if checked to hold at runtime, are guaranteed to imply that offline safety verification results about CPS models apply to the present run of the actual CPS implementation. Finally, this part considers logical elements of reasoning techniques for the real arithmetic that is used in CPS verification.

Online Material

The theory exercises provided at the end of the chapters are designed to actively check the understanding of the material and provide routes for further developments. In addition, the reader is invited to advance his or her understanding of the material by practicing CPS proving in the KeYmaera X verification tool [15], which is an aXiomatic Tactical Theorem Prover for Hybrid Systems that implements differential dynamic logic [48, 49, 51, 54]. For technical reasons, the concrete syntax in KeYmaera X has a slightly different ASCII notation, but, other than that, KeYmaera X closely follows the theory of differential dynamic logic as presented in this book. For educational purposes, this textbook also focuses on a series of instructive simpler examples instead of the technical complexities of full-blown applications that are reported elsewhere [22, 26, 31, 35, 36, 57, 58].

The Web page for this book is at the following URL:

http://www.lfcps.org/lfcps/

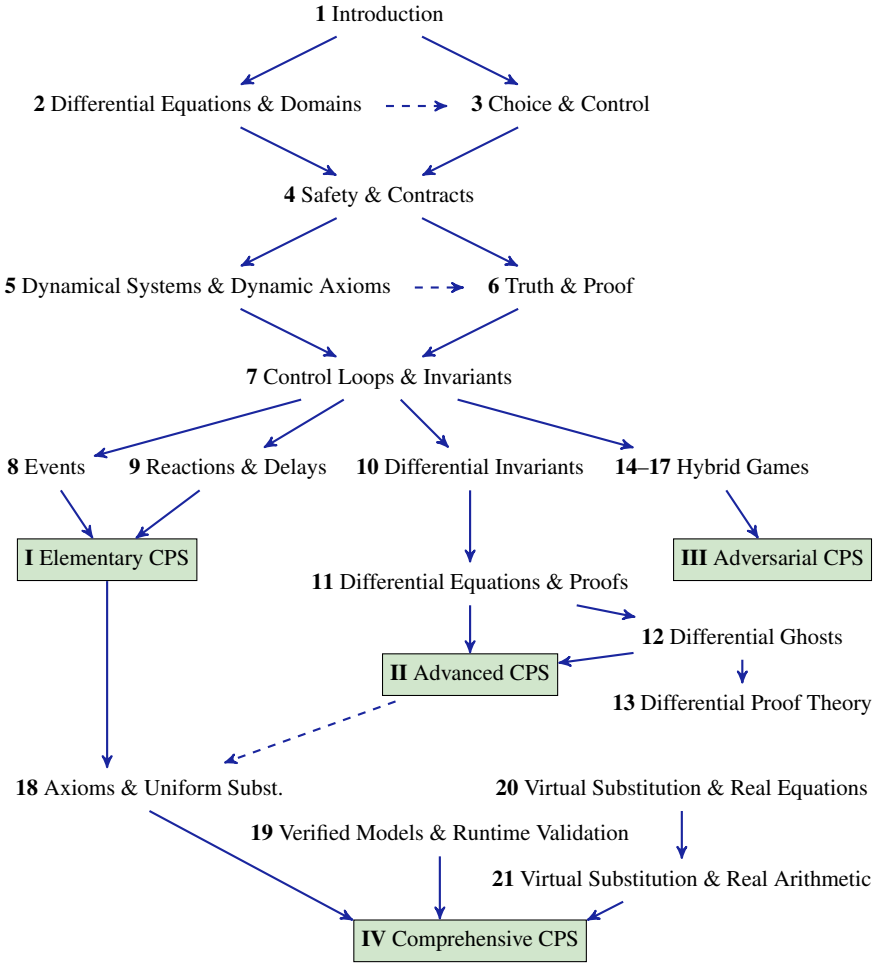


Fig. 1.3 Dependencies and suggested reading sequences of the chapters

Suggested Reading Sequence

Even if the basic suggested reading order in this book is linear, this textbook can be read in many different ways. Except for most of the foundation developed in Part I, the other parts of this book are independent and can be read in any order. The dependencies among the topics in the chapters are shown in Fig. 1.3. Weak dependencies on some small number of concepts are indicated as dashed lines, as these topics might be presented in a different order. The core of the textbook is the chapters that lead to Elementary CPS (Part I) in Fig. 1.3, including either Chaps. 8 or 9 or both. An integral part for Advanced CPS is Chaps. 10 and 11, along with an

optional study of the topic of differential ghosts for advanced differential equations in Chap. 12.

Different reading sequences are possible for this textbook. The minimal core for an understanding of elementary cyber-physical systems consists of Chaps. 1–7 from Part I. A minimal course emphasizing experience with system modeling covers the Chaps. 1–9 that lead to Part I on Elementary CPS in Fig. 1.3. For a minimal course emphasizing CPS reasoning Chaps. 1–7 would be followed by Chaps. 10–11 from Part II, possibly including Chap. 12 for advanced reasoning techniques. The other chapter sequences are independent. After Chaps. 1–9, any sequence of the other topics following the reader's interest is possible since the hybrid game chapters Chaps. 14–17 in Part III are independent of the subsequent topics in Part IV, which are, in turn, mostly independent of one another.

This textbook features an active development leading the reader through a critical and self-propelled development of the core aspects of cyber-physical systems. Especially at places marked as follows ...

Before you read on, see if you can find the answer for yourself.

... the reader is advised to work toward an answer before comparing it with the development pursued in the textbook. Of course, when comparing answers, the reader should keep in mind that there is more than one correct way of developing the material. The reader may have found a perfectly correct answer that just was not anticipated in the writing of the textbook. That represents a great opportunity to investigate advantages and downsides of the respective approaches.

1.8 Summary

This chapter gave an informal overview of application domains for cyber-physical systems, which combine cyber capabilities such as communication, computation, and control with physical capabilities such as motion or chemical process control. It motivated the need for careful designs and comprehensive safety analyses, which will be developed in this book. Closely related is the mathematical notion of hybrid systems, which are dynamical systems that combine discrete dynamics with continuous dynamics. Despite the fact that they are different notions (cyber-physical systems are based on the technical characteristics, while hybrid systems are a mathematical model), this textbook simplifies matters by using the two notions interchangeably in Parts I and II. More advanced models of cyber-physical systems will be deferred to Part III after the hybrid systems model has been understood well in Part I and Part II.

This chapter set the stage for the multi-dynamical systems approach that this textbook follows. Multi-dynamical systems are characterized by multiple facets of dynamical systems whose compositionality in a logic of dynamical systems enables a separation of concerns for CPS. The multi-dynamical systems view directly benefits the presentation in this book as well, by making it possible to focus on one

aspect at a time without losing the ability to combine the understanding attained for each aspect.

References

[1] Matthias Althoff and John M. Dolan. Online verification of automated road vehicles using reachability analysis. *IEEE Trans. on Robotics* **30**(4) (2014), 903–918. DOI: 10.1109/TRO.2014.2312453.

[2] Rajeev Alur. Formal verification of hybrid systems. In: *EMSOFT*. Ed. by Samarjit Chakraborty, Ahmed Jerraya, Sanjoy K. Baruah, and Sebastian Fischmeister. New York: ACM, 2011, 273–278. DOI: 10.1145/2038642.2038685.

[3] Rajeev Alur. *Principles of Cyber-Physical Systems*. Cambridge: MIT Press, 2015.

[4] Rajeev Alur, Costas Courcoubetis, Nicolas Halbwachs, Thomas A. Henzinger, Pei-Hsin Ho, Xavier Nicollin, Alfredo Olivero, Joseph Sifakis, and Sergio Yovine. The algorithmic analysis of hybrid systems. *Theor. Comput. Sci.* **138**(1) (1995), 3–34. DOI: 10.1016/0304-3975(94)00202-T.

[5] Rajeev Alur, Thomas Henzinger, Gerardo Lafferriere, and George J. Pappas. Discrete abstractions of hybrid systems. *Proc. IEEE* **88**(7) (2000), 971–984.

[6] Michael S. Branicky. General hybrid dynamical systems: modeling, analysis, and control. In: *Hybrid Systems*. Ed. by Rajeev Alur, Thomas A. Henzinger, and Eduardo D. Sontag. Vol. 1066. LNCS. Berlin: Springer, 1995, 186–200. DOI: 10.1007/BFb0020945.

[7] Davide Bresolin, Luca Geretti, Riccardo Muradore, Paolo Fiorini, and Tiziano Villa. Formal verification applied to robotic surgery. In: *Coordination Control of Distributed Systems*. Ed. by Jan H. van Schuppen and Tiziano Villa. Vol. 456. Lecture Notes in Control and Information Sciences. Berlin: Springer, 2015, 347–355. DOI: 10.1007/978-3-319-10407-2_40.

[8] Luminita Manuela Bujorianu. *Stochastic Reachability Analysis of Hybrid Systems*. Berlin: Springer, 2012. DOI: 10.1007/978-1-4471-2795-6.

[9] Edmund M. Clarke, E. Allen Emerson, and Joseph Sifakis. Model checking: algorithmic verification and debugging. *Commun. ACM* **52**(11) (2009), 74–84. DOI: 10.1145/1592761.1592781.

[10] Jennifer M. Davoren and Anil Nerode. Logics for hybrid systems. *IEEE* **88**(7) (2000), 985–1010. DOI: 10.1109/5.871305.

[11] Akash Deshpande, Aleks Göllü, and Pravin Varaiya. SHIFT: a formalism and a programming language for dynamic networks of hybrid automata. In: *Hybrid Systems*. Ed. by Panos J. Antsaklis, Wolf Kohn, Anil Nerode, and Shankar Sastry. Vol. 1273. LNCS. Springer, 1996, 113–133. DOI: 10.1007/BFb0031558.

[12] Laurent Doyen, Goran Frehse, George J. Pappas, and André Platzer. Verification of hybrid systems. In: *Handbook of Model Checking*. Ed. by Edmund M.

Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem. Springer, 2018. Chap. 30. DOI: 10.1007/978-3-319-10575-8_30.

[13] G. K. Fourlas, K. J. Kyriakopoulos, and C. D. Vournas. Hybrid systems modeling for power systems. *Circuits and Systems Magazine, IEEE* **4**(3) (2004), 16–23. DOI: 10.1109/MCAS.2004.1337806.

[14] Gottlob Frege. *Begriffsschrift, eine der arithmetischen nachgebildete Formelsprache des reinen Denkens*. Halle: Verlag von Louis Nebert, 1879.

[15] Nathan Fulton, Stefan Mitsch, Jan-David Quesel, Marcus Völp, and André Platzer. KeYmaera X: an axiomatic tactical theorem prover for hybrid systems. In: *CADE*. Ed. by Amy Felty and Aart Middeldorp. Vol. 9195. LNCS. Berlin: Springer, 2015, 527–538. DOI: 10.1007/978-3-319-21401-6_36.

[16] Gerhard Gentzen. Untersuchungen über das logische Schließen I. *Math. Zeit.* **39**(2) (1935), 176–210. DOI: 10.1007/BF01201353.

[17] Radu Grosu, Grégory Batt, Flavio H. Fenton, James Glimm, Colas Le Guernic, Scott A. Smolka, and Ezio Bartocci. From cardiac cells to genetic regulatory networks. In: *CAV*. Ed. by Ganesh Gopalakrishnan and Shaz Qadeer. Vol. 6806. LNCS. Berlin: Springer, 2011, 396–411. DOI: 10.1007/978-3-642-22110-1_31.

[18] Thomas A. Henzinger. The theory of hybrid automata. In: *LICS*. Los Alamitos: IEEE Computer Society, 1996, 278–292. DOI: 10.1109/LICS.1996.561342.

[19] Thomas A. Henzinger and Joseph Sifakis. The discipline of embedded systems design. *Computer* **40**(10) (Oct. 2007), 32–40. DOI: 10.1109/MC.2007.364.

[20] David Hilbert. Die Grundlagen der Mathematik. *Abhandlungen aus dem Seminar der Hamburgischen Universität* **6**(1) (1928), 65–85. DOI: 10.1007/BF02940602.

[21] Charles Antony Richard Hoare. An axiomatic basis for computer programming. *Commun. ACM* **12**(10) (1969), 576–580. DOI: 10.1145/363235.363259.

[22] Jean-Baptiste Jeannin, Khalil Ghorbal, Yanni Kouskoulas, Aurora Schmidt, Ryan Gardner, Stefan Mitsch, and André Platzer. A formally verified hybrid system for safe advisories in the next-generation airborne collision avoidance system. *STTT* **19**(6) (2017), 717–741. DOI: 10.1007/s10009-016-0434-1.

[23] Taylor T. Johnson and Sayan Mitra. Parametrized verification of distributed cyber-physical systems: an aircraft landing protocol case study. In: *ICCPS*. Los Alamitos: IEEE, 2012, 161–170. DOI: 10.1109/ICCPS.2012.24.

[24] Nidhi Kalra and Susan M. Paddock. *Driving to Safety – How Many Miles of Driving Would It Take to Demonstrate Autonomous Vehicle Reliability?* Tech. rep. RAND Corporation, 2016. DOI: 10.7249/RR1478.

[25] BaekGyu Kim, Anaheed Ayoub, Oleg Sokolsky, Insup Lee, Paul L. Jones, Yi Zhang, and Raoul Praful Jetley. Safety-assured development of the GPCA infusion pump software. In: *EMSOFT*. Ed. by Samarjit Chakraborty, Ahmed

Jerraya, Sanjoy K. Baruah, and Sebastian Fischmeister. New York: ACM, 2011, 155–164. DOI: 10.1145/2038642.2038667.

[26] Yanni Kouskoulas, David W. Renshaw, André Platzer, and Peter Kazanzides. Certifying the safe design of a virtual fixture control algorithm for a surgical robot. In: *HSCC*. Ed. by Calin Belta and Franjo Ivancic. ACM, 2013, 263–272. DOI: 10.1145/2461328.2461369.

[27] Kim Guldstrand Larsen. Verification and performance analysis for embedded systems. In: *TASE 2009, Third IEEE International Symposium on Theoretical Aspects of Software Engineering, 29-31 July 2009, Tianjin, China*. Ed. by Wei-Ngan Chin and Shengchao Qin. IEEE Computer Society, 2009, 3–4. DOI: 10.1109/TASE.2009.66.

[28] Edward Ashford Lee and Sanjit Arunjumar Seshia. *Introduction to Embedded Systems — A Cyber-Physical Systems Approach*. Lulu.com, 2013.

[29] Insup Lee and Oleg Sokolsky. Medical cyber physical systems. In: *DAC*. Ed. by Sachin S. Sapatnekar. New York: ACM, 2010, 743–748.

[30] Insup Lee, Oleg Sokolsky, Sanjian Chen, John Hatcliff, Eunkyoung Jee, BaekGyu Kim, Andrew L. King, Margaret Mullen-Fortino, Soojin Park, Alex Roederer, and Krishna K. Venkatasubramanian. Challenges and research directions in medical cyber-physical systems. *Proc. IEEE* **100**(1) (2012), 75–90. DOI: 10.1109/JPROC.2011.2165270.

[31] Sarah M. Loos, André Platzer, and Ligia Nistor. Adaptive cruise control: hybrid, distributed, and now formally verified. In: *FM*. Ed. by Michael Butler and Wolfram Schulte. Vol. 6664. LNCS. Berlin: Springer, 2011, 42–56. DOI: 10.1007/978-3-642-21437-0_6.

[32] Jan Lunze and Françoise Lamnabhi-Lagarrigue, eds. *Handbook of Hybrid Systems Control: Theory, Tools, Applications*. Cambridge: Cambridge Univ. Press, 2009. DOI: 10.1017/CBO9780511807930.

[33] Oded Maler. Control from computer science. *Annual Reviews in Control* **26**(2) (2002), 175–187. DOI: 10.1016/S1367-5788(02)00030-5.

[34] Sayan Mitra, Tichakorn Wongpiromsarn, and Richard M. Murray. Verifying cyber-physical interactions in safety-critical systems. *IEEE Security & Privacy* **11**(4) (2013), 28–37. DOI: 10.1109/MSP.2013.77.

[35] Stefan Mitsch, Marco Gario, Christof J. Budnik, Michael Golm, and André Platzer. Formal verification of train control with air pressure brakes. In: *Reliability, Safety, and Security of Railway Systems. Modelling, Analysis, Verification, and Certification - Second International Conference, RSSRail 2017, Pistoia, Italy, November 14-16, 2017, Proceedings*. Ed. by Alessandro Fantechi, Thierry Lecomte, and Alexander Romanovsky. Vol. 10598. LNCS. Springer, 2017, 173–191. DOI: 10.1007/978-3-319-68499-4_12.

[36] Stefan Mitsch, Khalil Ghorbal, David Vogelbacher, and André Platzer. Formal verification of obstacle avoidance and navigation of ground robots. *I. J. Robotics Res.* **36**(12) (2017), 1312–1340. DOI: 10.1177/0278364917733549.

[37] Anil Nerode. Logic and control. In: *CiE*. Ed. by S. Barry Cooper, Benedikt Löwe, and Andrea Sorbi. Vol. 4497. LNCS. Berlin: Springer, 2007, 585–597. DOI: 10.1007/978-3-540-73001-9_61.

[38] Anil Nerode and Wolf Kohn. Models for hybrid systems: automata, topologies, controllability, observability. In: *Hybrid Systems*. Ed. by Robert L. Grossman, Anil Nerode, Anders P. Ravn, and Hans Rischel. Vol. 736. LNCS. Berlin: Springer, 1992, 317–356.

[39] NITRD CPS Senior Steering Group. *CPS vision statement*. NITRD. 2012.

[40] George J. Pappas. Wireless control networks: modeling, synthesis, robustness, security. In: *Proceedings of the 14th ACM International Conference on Hybrid Systems: Computation and Control, HSCC 2011, Chicago, IL, USA, April 12-14, 2011*. Ed. by Marco Caccamo, Emilio Frazzoli, and Radu Grosu. New York: ACM, 2011, 1–2. DOI: 10.1145/1967701.1967703.

[41] Erion Plaku, Lydia E. Kavraki, and Moshe Y. Vardi. Hybrid systems: from verification to falsification by combining motion planning and discrete search. *Form. Methods Syst. Des.* **34**(2) (2009), 157–182. DOI: 10.1007/s10703-008-0058-5.

[42] André Platzer. Differential dynamic logic for hybrid systems. *J. Autom. Reas.* **41**(2) (2008), 143–189. DOI: 10.1007/s10817-008-9103-8.

[43] André Platzer. Differential Dynamic Logics: Automated Theorem Proving for Hybrid Systems. PhD thesis. Department of Computing Science, University of Oldenburg, 2008.

[44] André Platzer. Differential-algebraic dynamic logic for differential-algebraic programs. *J. Log. Comput.* **20**(1) (2010), 309–352. DOI: 10.1093/logcom/exn070.

[45] André Platzer. *Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics*. Heidelberg: Springer, 2010. DOI: 10.1007/978-3-642-14509-4.

[46] André Platzer. Stochastic differential dynamic logic for stochastic hybrid programs. In: *CADE*. Ed. by Nikolaj Bjørner and Viorica Sofronie-Stokkermans. Vol. 6803. LNCS. Berlin: Springer, 2011, 446–460. DOI: 10.1007/978-3-642-22438-6_34.

[47] André Platzer. A complete axiomatization of quantified differential dynamic logic for distributed hybrid systems. *Log. Meth. Comput. Sci.* **8**(4:17) (2012). Special issue for selected papers from CSL'10, 1–44. DOI: 10.2168/LMCS-8(4:17)2012.

[48] André Platzer. Logics of dynamical systems. In: *LICS*. Los Alamitos: IEEE, 2012, 13–24. DOI: 10.1109/LICS.2012.13.

[49] André Platzer. The complete proof theory of hybrid systems. In: *LICS*. Los Alamitos: IEEE, 2012, 541–550. DOI: 10.1109/LICS.2012.64.

[50] André Platzer. Teaching CPS foundations with contracts. In: *CPS-Ed*. 2013, 7–10.

[51] André Platzer. A uniform substitution calculus for differential dynamic logic. In: *CADE*. Ed. by Amy Felty and Aart Middeldorp. Vol. 9195. LNCS. Berlin: Springer, 2015, 467–481. DOI: 10.1007/978-3-319-21401-6_32.

[52] André Platzer. Differential game logic. *ACM Trans. Comput. Log.* **17**(1) (2015), 1:1–1:51. DOI: 10.1145/2817824.

[53] André Platzer. Logic & proofs for cyber-physical systems. In: *IJCAR*. Ed. by Nicola Olivetti and Ashish Tiwari. Vol. 9706. LNCS. Berlin: Springer, 2016, 15–21. DOI: 10.1007/978-3-319-40229-1_3.

[54] André Platzer. A complete uniform substitution calculus for differential dynamic logic. *J. Autom. Reas.* **59**(2) (2017), 219–265. DOI: 10.1007/s10817-016-9385-1.

[55] André Platzer. Differential hybrid games. *ACM Trans. Comput. Log.* **18**(3) (2017), 19:1–19:44. DOI: 10.1145/3091123.

[56] André Platzer and Edmund M. Clarke. The image computation problem in hybrid systems model checking. In: *HSCC*. Ed. by Alberto Bemporad, Antonio Bicchi, and Giorgio C. Buttazzo. Vol. 4416. LNCS. Springer, 2007, 473–486. DOI: 10.1007/978-3-540-71493-4_37.

[57] André Platzer and Edmund M. Clarke. Formal verification of curved flight collision avoidance maneuvers: a case study. In: *FM*. Ed. by Ana Cavalcanti and Dennis Dams. Vol. 5850. LNCS. Berlin: Springer, 2009, 547–562. DOI: 10.1007/978-3-642-05089-3_35.

[58] André Platzer and Jan-David Quesel. European Train Control System: a case study in formal verification. In: *ICFEM*. Ed. by Karin Breitman and Ana Cavalcanti. Vol. 5885. LNCS. Berlin: Springer, 2009, 246–265. DOI: 10.1007/978-3-642-10373-5_13.

[59] Vaughan R. Pratt. Semantical considerations on Floyd-Hoare logic. In: *17th Annual Symposium on Foundations of Computer Science, 25-27 October 1976, Houston, Texas, USA*. Los Alamitos: IEEE, 1976, 109–121. DOI: 10.1109/SFCS.1976.27.

[60] President's Council of Advisors on Science and Technology. *Leadership under challenge: information technology R&D in a competitive world.* An Assessment of the Federal Networking and Information Technology R&D Program. Aug. 2007.

[61] Dana Scott and Christopher Strachey. *Towards a mathematical semantics for computer languages*. Tech. rep. PRG-6. Oxford Programming Research Group, 1971.

[62] Raymond M. Smullyan. *First-Order Logic*. Mineola: Dover, 1968. DOI: 10.1007/978-3-642-86718-7.

[63] Paulo Tabuada. *Verification and Control of Hybrid Systems: A Symbolic Approach*. Berlin: Springer, 2009. DOI: 10.1007/978-1-4419-0224-5.

[64] Ashish Tiwari. Abstractions for hybrid systems. *Form. Methods Syst. Des.* **32**(1) (2008), 57–83. DOI: 10.1007/s10703-007-0044-3.

[65] Ashish Tiwari. Logic in software, dynamical and biological systems. In: *LICS*. IEEE Computer Society, 2011, 9–10. DOI: 10.1109/LICS.2011.20.

[66] Claire Tomlin, George J. Pappas, and Shankar Sastry. Conflict resolution for air traffic management: a study in multi-agent hybrid systems. *IEEE T. Automat. Contr.* **43**(4) (1998), 509–521. DOI: 10.1109/9.664154.

[67] Jeannette M. Wing. Computational thinking. *Commun. ACM* **49**(3) (2006), 33–35. DOI: 10.1145/1118178.1118215.

[68] Jeannette M. Wing. Five deep questions in computing. *Commun. ACM* **51**(1) (2008), 58–60. DOI: 10.1145/1327452.1327479.

[69] Paolo Zuliani, André Platzer, and Edmund M. Clarke. Bayesian statistical model checking with application to Simulink/Stateflow verification. *Form. Methods Syst. Des.* **43**(2) (2013), 338–367. DOI: 10.1007/s10703-013-0195-3.