15-424: Foundations of Cyber-Physical Systems

Course Syllabus

André Platzer

Computer Science Department, Carnegie Mellon University

Cyber-physical systems (CPSs) combine cyber effects (computation and/or communication) with physical effects (motion or other physical processes). Cars, aircraft, and robots are prime examples, because they move physically in space in a way that is determined by discrete computerized control algorithms. 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 like keeping aircraft from colliding. In this course we will strive to answer the fundamental question posed by Jeannette Wing:

> "How can we provide people with cyber-physical systems they can bet their lives on?"

The cornerstone of our course design are hybrid programs (HPs), which capture relevant dynamical aspects of CPSs in a simple programming language with a simple semantics. One important aspect of HPs is that they directly allow the programmer to refer to real-valued variables representing real quantities and specify their dynamics as part of the HP.

This course will give you the required skills to formally analyze the CPSs that are all around us – from power plants to pace makers 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. It will provide an excellent foundation for students who seek industry positions and for students interested in pursuing research.

Contents

1 Course Information 2

2 Learning Objectives 3
- 2.1 Modeling and Control . . . 3
- 2.2 Computational Thinking . . . 4
- 2.3 CPS Skills . . . 4

3 Programming Language 5

4 Course Project for the CPS V&V Grand Prix 5

5 FAQ 6
- 5.1 Who Should Take This Course? . . . 6
- 5.2 What are Students Expected to Know Before This Course? . . . 6
- 5.3 What Time Commitment Does This Course Need to Succeed? . . . 7

6 Schedule 7

7 Take Care of Yourself 8

8 Policies 8
- 8.1 Laptops in Lecture . . . 8
- 8.2 Due Dates and Late Days . . . 8
- 8.3 Grading . . . 9
- 8.4 Students with Disabilities . . . 9
- 8.5 Collaboration and Academic Integrity . . . 9
- 8.6 Proof Redo Principle . . . 10
- 8.7 Extra Points for Proof Exploits: KeYmaera X Integrity . . . 11

1 Course Information

Lectures Tue/Thu noon-1:20, GHC 4307

Recitations Fri 3:00-4:20, GHC 4307

Credit 12 units

Prerequisites As elaborated in Section 5, the course assumes prior exposure to basic computer programming, differentiation, and mathematical reasoning:

1. 15-122 Principles of Imperative Computation (**or** equivalent), **and**
2. 21-122 Integration, Differential Equations, and Approximation (**or** equivalent), **and**

3. (15-251 Great Theoretical Ideas in Computer Science **or** 21-241 Matrix algebra **or** 18-202 Mathematical Foundations of Electrical Engineering **or** equivalent)

Textbook You are expected to follow the lecture notes on the course web page

Grading 22% Homework, 51% labs, including 22% final course project, 11% Midterm exam, 11% Final exam, 5% active participation in class and Piazza

Homework Weekly, usually Thursdays. 6 late days total for Veribot labs.

Midterm I Thu 03/02, in class, closed book.

Final Thu 04/13, in class, closed book.

Grand Prix Thu 05/11 for presenting final course project to a panel of experts.

Home http://lfcps.org/course/fcps17.html

Piazza discussion board linked from course web page

Autolab homework submission and grade information web page

Tools we will use the hybrid systems verification tool KeYmaera X

The 15-424 course counts as a Logics/Languages elective in the Computer Science curriculum. The course 15-824 fulfills the Programming Languages star requirement.

2 Learning Objectives

The learning objectives of Foundations of Cyber-Physical Systems are organized along the dimensions: *modeling and control*, *computational thinking* [1], as well as *CPS skills*.

2.1 Modeling and Control

In the area of *modeling and control*, successful students will

- **understand the core principles behind CPS.** A solid understanding of these core principles is important for anyone who wants to integrate cyber and physical components to solve problems that no part could solve alone.
- **develop models and controls.** In order to understand, design, and analyze CPS, it is important to be able to develop models for 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, where it is important to manage adversarial effects, or where a nondeterministic model is sufficient.

2.2 Computational Thinking

In the area of *computational thinking*, successful students should be able 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.
- **understand abstraction in system designs.** The power of abstraction is essential for the modular organization of CPS, 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 post-conditions and invariants for CPS models.** Pre- and post-conditions allow us to capture under which circumstance it is safe to run a CPS or a part of a CPS 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. Invariants achieve 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. 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 course is not limited to covering the science of how to prove CPSs, but you will gain practical experience through appropriately scoped projects in the theorem prover KeYmaera X. This experience will help you learn how to best 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.
- **use formal methods tools for CPS.** Formal verification at nontrivial scale becomes more feasible with a good command of formal verification tools. While a full coverage of all aspects of, say, an aircraft is out of reach for this course, you will be exploring a series of safe designs for increasingly challenging tasks of a robot controller. You also have the opportunity to explore your favorite projects in the final course project.

2.3 CPS Skills

In the area of *CPS skills*, successful students will be able to

- **understand the semantics of a CPS model.** What may be easy in a classi-

cal isolated program becomes very demanding when that program interfaces with effects in the physical world. A full treatment of, e.g., the semantics of stochastic CPS effects is better placed in a specialized course. But understanding the meaning of a CPS model with fewer dynamical aspects and know how it will execute is fundamental to reasoning.

- **develop an intuition for operational effects.** Intuition for the joint operational effect of a CPS is crucial, e.g., about what the effect of a particular discrete computer control algorithm on a continuous plant will be.
- **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 for minimizing 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.

3 Programming Language

With a suitably generalized programming language, the behavior of a CPS can be described by a program. This course develops the programming language of *hybrid programs* (HPs) to capture relevant dynamical aspects of cyber-physical systems in a simple programming language with a simple semantics. The most distinctive features of HPs are that they prominently feature differential equations and nondeterminism. HPs support differential equations as continuous models of the physical system dynamics so that we can directly write down a differential equation in the middle of a program to describe the behavior of physics. Nondeterminism is another feature required for the adequacy of CPS models, e.g. for capturing choices in the system execution or uncertainty about the environment. When describing a robot controller, for example, we cannot know for sure what decisions other agents in the environment reach and need to be prepared to handle multiple choices in the execution. The course leverages differential dynamic logic as a specification and verification language for rigorous reasoning about hybrid programs that makes program expectations explicit and localizes reasoning about their correctness.

4 Course Project for the CPS V&V Grand Prix

The final course project gives you an opportunity for you to creatively use what you've learned throughout the course and dive deeply into a CPS problem of your choosing. It is your big chance to achieve fame, glory, and prizes at the CPS Verification and Validation final project competition (CPS V&V Grand Prix). What you attempt for your project is completely up to you. There are only two requirements: (1) We want your project to be challenging (you should learn something relevant to the themes of this class) and (2) we want your project to be fun (you should be excited to work on it)!

5 FAQ

This section elaborates the expected background and purpose of this course.

5.1 Who Should Take This Course?

You should definitely take this course if

- you ever want to program robots that operate near humans so that you need to understand how to do that safely, or
- you ever want to develop computer control systems for cars, or
- you ever want to write programs that control aircraft or drones, or
- you ever want to help computers control power plants or the smart grid, or
- you want to do embedded systems or cyber-physical systems, or
- you are interested in learning how computation interfaces with the real world, or
- you are simply fascinated by combining mathematics and computer science, or
- you want to see logic matter in reality.

Building robots is what other courses will teach you. But how you program them safely is what this course covers.

5.2 What are Students Expected to Know Before This Course?

The formal requirements for the course are listed in Section 1. The course assumes prior exposure to basic computer programming such as 15-122, that you have seen basic differentiation and differential equations such as 21-122, and that you have had prior exposure to some form of mathematical reasoning such as in 15-251, 21-241 or 18-202 where you learned about inductive proofs. The course covers the basic required mathematical and logical background of cyber-physical systems but you will be expected to follow the lecture notes as needed.

If you are afraid of programming or afraid of mathematics, then you will find this course more challenging. The course is specifically designed not to require particularly advanced background, but you should feel comfortable picking the required concepts up as we go. We will explain what you need to know in the course and provide pointers to further reading material. Coming into this course, you should definitely already know what a derivative is and be comfortable using derivatives in mathematical arguments. You should have an intuitive understanding of differential equations for modeling common physical processes. We will frequently need this ordinary differential equation system (ODE)

$$x' = v, v' = a \tag{1}$$

which can be understood as saying that the time-derivative of x is v, and the time-derivative of v is a. In other words, this differential equation means that the derivative of the position x is the velocity v, and the derivative of the velocity v is the acceleration a. Understanding ODE (1) will suffice for the first part of the course. As the course progresses, we will learn how to do elegant reasoning about even ODEs whose solutions are nasty, which provides a good opportunity to reinforce your understanding of ODEs.

5.3 What Time Commitment Does This Course Need to Succeed?

The course is a 12 unit course and includes both lectures and recitations. Assignments alternate between labs where you model and formally verify systems in a theorem prover and written assignments which exercise the underlying logical and mathematical theory. Before you submit your final robot (called *Veribot*), you will also submit a *Betabot*, which is a beta-version of your robot controller that you conjecture to be safe and submit for feedback. Unlike your final robot submission (the Veribot), your Betabot does not yet need to be verified. Keep in mind that most CPS designs are more challenging than it appears at first glance. You should, thus, start your assignments early. The more thorough your early designs are, the better and more informative our feedback to you can be.

How much time you need to complete this course depends on how easy the material comes to you. The course will certainly be challenging. It will not be challenging because of sheer volume of things that we demand you do. Instead, the challenges will be of a more conceptual nature. Your final safety arguments for a CPS design may be easy, but it takes time to develop a safe design in the first place and then build a safety argument for it. We structure the labs and assignments in a way that carefully builds things up layer by layer, so that you will learn about cyber-physical systems with a well-structured gradual approach. You will gradually learn about one layer of CPS challenges at a time and we will proceed to the next challenges once we have mastered the previous ones.

This is an interdisciplinary course. Every student will come in with substantial background in some but certainly not all areas. The course gives you time to play catchup on the background, including simple physics, differential equations, and logic, but you should expect to spend some time getting up to speed.

6 Schedule

The tentative schedule of lectures is as follows:

1. Cyber-Physical Systems: Introduction
2. Differential Equations & Domains
3. Choice & Control
4. Safety & Contracts
5. Dynamical Systems & Dynamic Axioms
6. Truth & Proof
7. Control Loops & Invariants
8. Events & Responses
9. Reactions & Delays
10. Differential Equations & Differential Invariants
11. Differential Equations & Proofs
12. Ghosts & Differential Ghosts
13. Differential Invariants & Proof Theory
14. Logical Foundations & CPS
15. Verified Models & Verified Runtime Validation

16. Hybrid Systems & Games
17. Winning Strategies & Regions
18. Winning & Proving Hybrid Games
19. Game Proofs & Separations
20. Axioms & Uniform Substitutions
21. Differential Axioms & Uniform Substitutions
22. Distributed Systems & Hybrid Systems
23. Virtual Substitution & Real Equations
24. Virtual Substitution & Real Arithmetic

7 Take Care of Yourself

Cyber-physical systems are crucially important for our society, but so are you! When you are facing CPS challenges or any others, please keep in mind that you can only help our society design better and safer systems if you also watch out for yourself. Do take some time to relax, which often helps you approach questions with a fresh perspective next morning.

All of us benefit from support during times of struggle. You are not alone. There are many helpful resources available on campus and an important part of the college experience is learning how to ask for help. You should ask sooner rather than later.

Should you find yourself or a friend in serious trouble, take it seriously: your classes can wait. For emergencies call UPMC's re:solve Crisis Network at `1-888-796-8226`. Counseling and Psychological Services (CaPS) is here to help: call `412-268-2922` and visit their website at `http://www.cmu.edu/counseling/`.

Also consider reaching out to a friend, faculty or family member you trust for help getting connected to the support that can help.

8 Policies

8.1 Laptops in Lecture

As research on learning shows, unexpected noises and movement automatically divert and capture people's attention, which means you are affecting everyone's learning experience if your phone, laptop, etc. makes noise or is visually distracting during class.

Therefore, please silence all mobile devices during class. You are welcome to use laptops for note-taking, but if you are able, please do so from the back of the classroom. If you are not in the back of the classroom, it is especially important that you do not use your laptop for anything besides note-taking. Do not work on assignments for this or any other class while attending lecture or recitation.

8.2 Due Dates and Late Days

Labs and written assignments will be submitted electronically via Autolab. **Labs have two different due dates**, the Betabots and the Veribots. Betabots are due at the

beginning of class on the Betabot due date. Veribots are due by midnight on the day the lab is due. You have a total of 6 late days for the semester of which you can use at most 2 per Veribot assignment (no late days can be used for Betabots nor for the final project nor on written assignments for reasons of course logistics). If you submit beyond those late days, 25% will be docked on an assignment for each additional late day.

Written assignments are due by midnight on the date they are due. There are no late days for written assignments. If you submit after a written assignment is due, 25% will be docked on a written assignment for each late day.

Labs 0 and 1 must be done individually. Later labs can be done *individually or in pairs*. You must choose a partner by the respective due date of the Betabot. If you have difficulty finding a partner, or if problems in your working relationship arise during the semester, please get in touch with the instructor as soon as possible. Written assignments must be done *individually*.

8.3 Grading

The most important criterion is always correctness. Buggy code is useless, and is likely to get a low score, because the corresponding CPSs are likely to do serious damage. A secondary criterion is the performance of your robot controller in terms of reaching its goal and interacting with its environment.

Grading for written assignments is based on the correctness of the answer and the presentation of your reasoning. Strive for clarity and conciseness, but show how you arrived at the answer. Stating an answer without explanation does not count as an answer. If you cannot solve a problem, explaining your approach and why you failed is encouraged. Such answers will be given partial credit.

Absent exceptional circumstances, students with a total score of 90% and above will get an A, 80% and above will get a B, 70% and above a C, and 60% and above a D.

Grade cutoffs may be lowered based on the difficulty of exams and assignments. Precise grade cutoffs will not be discussed at any point during or after the semester. For students very close to grade boundaries, instructors may, at their discretion, consider participation in lecture and recitation or exam and final project performance for the final grade.

8.4 Students with Disabilities

Carnegie Mellon University makes every effort to provide accessible facilities and programs for individuals with disabilities. If you have a disability and require accommodations, contact the Office of Disability Resources at `access@andrew.cmu.edu`. Please let the instructors know early in the semester so that your needs may be appropriately met. Special accommodation for exams must be requested for each exam separately one week in advance.

8.5 Collaboration and Academic Integrity

You are expected to comply with the University Policy on Academic Integrity, which will be applied rigorously.

The value of your degree depends on the academic integrity of yourself and your peers in each of your classes. It is expected that, unless otherwise instructed, the work you submit as your own is your own work and not someone else's work or a collaboration between yourself and other(s).

Please read the University Policy on Academic Integrity carefully to understand the penalties associated with academic dishonesty at Carnegie Mellon. In this class, cheating/copying/plagiarism means copying all or part of a program or homework solution, model, or proof from another student or unauthorized source such as the Internet, knowingly giving such information to another student, or giving or receiving unauthorized information during an examination. In general, *each solution you submit (written assignment, lab assignment, model, proof, or exam) must be your own work.* Some labs expressly indicate that they can be done by a single student or by a pair of students, at your discretion. But all written assignments must be your own. In the event that you use information by another person in your solution, you must clearly cite the source of this information (and receive prior permission if unsure whether this is permitted). It is considered cheating to compare or discuss complete or partial solutions.

It is *not* considered cheating to clarify vague points in the labs, assignments, or lecture material, or to give help or receive help in *general use* of the computer systems or tools such as KeYmaera X, or other facilities. It is permitted and encouraged to share general advice on how to use KeYmaera X or general discussions about course assignments. Any assistance, though, must be limited to discussion of the *problems in general*, and cannot be about the solutions of the assignments. You must also refrain from looking at other students' models and proofs while you are getting or receiving help for these tools. It is not cheating to review graded assignments or exams with students in the same class as you, but it is considered unauthorized assistance to share these materials between different iterations of the course. Do not post code from this course publicly, e.g. to GitHub or BitBucket.

When you are having difficulties designing safe controllers or conducting safety analyses for them, also keep in mind that this is quite a universal challenge. But keep in mind that a good strategy to overcome such obstacles is to consider simplified scenarios with simplifying assumptions first. Correctness is crucial, and a correct safety result for a simpler safe controller is more valuable than a more general and more complex controller that fails to be analyzable.

8.6 Proof Redo Principle

Proofs can be difficult. If you did not find a full proof for your CPS controller by the Veribot deadline, not all is lost, since you will not have activated a possibly unsafe CPS. As an opportunity to earn back some missed points on the labs, you can resubmit each of your Veribot labs *once* up to 3 days after the due date of the subsequent Veribot lab. Your model must be completely correct without modeling errors. Your proof must be complete. You will only earn back points on the proof portion of the labs, not points that you lost due to modeling errors.

8.7 Extra Points for Proof Exploits: KeYmaera X Integrity

All feedback about how to improve the lecture notes and KeYmaera X is always very welcome and is part of your participation grade. There is one form of feedback that is particularly helpful: feedback that concerns soundness.

Soundness is crucial and fundamental, but of special significance for the high stakes of cyber-physical systems. What good would a safety analysis of a broken cyber-physical system do if the analysis procedure itself is broken?

To reflect that, we are soliciting *Proof Exploits*. By which we mean proofs that exploit soundness-critical flaws in the lecture notes or soundness-critical bugs in KeYmaera X. Each new soundness-critical bug that you are the first person to report is worth 20 points of extra credit. For full credit you should also demonstrate with a proof exploit how that bug can be exploited to produce a proof of `false` or of `1=0`. A proof exploit is a formal proof on paper or a test case for KeYmaera X demonstrating how the flaw can be exploited to exhibit a proof of `false`, which, since `false` is rarely true, cannot have a proof in any sound verification procedure.

Needless to say that this is not just a great way for you to earn extra credit but also a really solid preparation for questions scrutinizing what rules and axioms and proof attempts are sound and which ones aren't. Which is an invaluable skill when it comes down to analyzing CPSs.

We will award a special prize during the CPS V&V Grand Prix to the person achieving the most extra credit via proof exploits.

Hint: You are allowed to be arbitrarily creative in your proof exploits.

References

[1] J. M. Wing. Computational thinking. *Commun. ACM*, 49(3):33–35, 2006. doi:10.1145/1118178.1118215.