Teaching Cyber-Physical Systems with Logic

Sarah M. Loos
Computer Science Department
Carnegie Mellon University
sloos@cs.cmu.edu

André Platzer
Computer Science Department
Carnegie Mellon University
aplatzer@cs.cmu.edu

ABSTRACT

This paper reports on a course breaking with the myth that cyber-physical systems are too challenging to be taught at the undergraduate level. Cyber-physical systems (CPSs) such as computer-controlled cars, airplanes or robots play an increasingly crucial role in our daily lives. They are systems that we bet our lives on, so they need to be safe. Getting CPSs safe, however, is an intellectual challenge, because of their intricate interactions of complex control software with their physical behavior. Who can design these notoriously challenging systems with the scrutiny that is required to make sure they can be used safely? How can students, scientists, and practitioners acquire the required background in logic as well as in discrete and continuous mathematics and control in a single course in a way that meets the demands on rigor required in safe CPS design? This paper reports on the experience with a new undergraduate course, *Foundations of Cyber-Physical Systems*, that teaches students how to design a correct CPS, identify required safety properties, and justify that their designs meet these safety goals.

Categories and Subject Descriptors

K.3.2 [**Computers and Education**]: Computer and Information Science Education

1. INTRODUCTION

Cyber-physical systems (CPSs) combine cyber (computation and communication) with physics (motion or process control) and are prominent in, for example, aeronautics, automotive, and robotics industries and medical devices [1]. Despite the growing ubiquity of and reliance on software in safety-critical systems, the authors are not aware of any (undergraduate) course that provides students with the necessary theoretical foundations for analyzing safety in cyber-physical environments. This may stem from the fact that teaching CPS-related topics is notoriously challenging. A few students may have a strong background either in engineering physical systems, or in some areas of formal methods, but almost never in both. A sharp educational gap has been identified across the board at a workshop on CPS education [5]. This brings up the question of how to best teach the core aspects of CPS with the rigor that is required to prepare students and professionals for the challenges that lie ahead in enriching our world with safe and reliable cyber-physical systems. *Systems that we can bet our lives on.*

In Fall 2013, we introduced a new undergraduate course, *Foundations of Cyber-Physical Systems* (FCPS)[1] [12], at Carnegie Mellon University in which undergraduate students quickly advance from learning the basic concepts underlying CPS to being able to prove safety properties about complex CPS. A shorter version of the course was then offered as a research school at ENS Lyon and at University of Minho in 2014.

The primary insight behind the course design is that logical scrutiny, formalization, and correctness proofs are critical for CPS! Because CPSs are so easy to get wrong, these logical aspects are an integral part of CPS design and critical to the students' understanding of the intricate complexities of CPS. This paper advocates for a foundational approach that tames some of the complexities of CPS by focusing on a simple programming language to illustrate the core aspects of CPS. This language is introduced gradually in layers that the students master completely before moving on to the next challenge, like in approaches for introductory programming [6]. The attention is on the core elements of CPS hand-in-hand with their reasoning principles, because abstraction is a key factor for success in CPS and retrofitting safety is not possible in CPS. Overall, the FCPS course realizes computational thinking [13] for cyber-physical systems.

2. CPS EDUCATION OVERVIEW

How can we design a CPS course, which necessarily will cover topics spanning multiple disciplines, for students who likely will not have suitable background and prerequisites in all related areas? Nearly every paper included in the 2013 CPS-Education Workshop (CPS-Ed) [5] asks and attempts to address this question. The answer is not clear even at the graduate level, but it is an especially crucial issue to address for undergraduates. Some universities have gone so far as to create CPS concentrations or majors which require students to take years of coursework before getting into any core CPS materials. Because students often do not have the benefit of a breadth of prerequisites prior to taking a CPS course, it is hard to cover and balance both theoretical and practical approaches, as mentioned in several CPS-Ed papers [5].

The most common approach for teaching CPS is in a simulation environment, in which students can interact with their CPSs and update designs by trial and error [5]. One advantage of simulation is the ability to use an autograder, which can allow for the possibility of a massive open online course [4]. While students certainly have fun interacting with a virtual world, the findings we present in our analysis of lab submissions comparing across three offerings of the

[1]Materials for the Fall 2013 course offering can be found online: `http://symbolaris.com/course/fcps13.html`

course (see Section 6) and comments in our course evaluations actually indicate that simulations, while pretty, are not so helpful for ensuring correctness. With a physical environment, it is impossible to observe every scenario the system must be designed to safely handle. Moreover, seeing bad behavior only in a simulation may incentivize the student to create a patch for that specific scenario, while still allowing the root cause of the error to persist. Using trial and error to design CPS might still work in a controlled classroom environment, but this is a dangerous methodology to teach students in preparation for industry. Real software requires that people frequently bet their lives on its safe functioning in a wide range of circumstances.

In courses that cover background materials, theoretical principles of cyber-physical systems, and still provide hands-on practice, there is rarely any time leftover to properly cover verification and safety analysis. Verification frequently gets listed as future work in the CPS-Ed literature [5].

Even though teaching CPS is hard, getting it right is imperative as the reliance on software in safety-critical systems continues to increase. One message was clear from every paper in CPS-Ed [5]: *we need to train the future generation of scientists and practitioners to handle CPS correctly.*

Unlike other courses, we teach CPS through the lens of verification. The uniqueness of our "safety first" approach is not surprising, as many people still consider safety verification to be too hard for industry and too hard for undergraduates. We have observed exactly the opposite. By formalizing the specifications and proving they are satisfied from the very beginning, students gain an intimate understanding of the complexities of their systems from day one. This allows us to quickly advance to more challenging problems. We can also get into the details of CPS with challenging labs and course projects, rather than teaching a general appreciation or seminar course, as are often found at the graduate level.

In addition to this overview of the state of CPS education as a whole, we discuss courses based on two embedded systems textbooks [8, 7]. These embedded systems design courses focus primarily on system architecture or modeling and simulation. These courses touch on verification topics (e.g. specifications, model checking, reachability analysis) conveying a sense of the importance of verification for CPS; however, verification does not play an active role in the practical components of these courses. There are also plans for a CPS undergraduate course with real-time emphasis [3].

3. FOUNDATIONS OF CYBER-PHYSICAL SYSTEMS

The course *Foundations of Cyber-Physical Systems* (FCPS) is a newly developed[2] undergraduate course at CMU that has been offered first in Fall 2013. It starts without assuming prior knowledge in logic but with just enough prior exposure to differential equations that students have a minimal intuition for them. Students are also expected to have succeeded in a first course in computer science and have some background experience with mathematical proofs. The course has been offered as an elective at CMU and attracted students in their junior or senior year with a wide range of majors, including computer science, electrical engineering, robotics, mathematics, and network systems.

The first half of the course has been offered twice as a one week research school: first at ENS Lyon for first and second year master's students in computer science, and then at University of Minho for PhD students in computer science.

Learning Objectives.

Detailed learning objectives for the course design have been reported at a workshop [12] before the course was offered, and categorized with the goals of modeling/control, computational thinking, and CPS skills. To sum up, the most important learning objectives are that students:

- Understand the core principles behind CPS.
- Develop models and controls.
- Identify safety specifications and critical properties.
- Understand abstraction and system architectures.
- Learn how to design by invariant.
- Reason rigorously about CPS models.
- Verify CPS models of appropriate scale.
- Understand the semantics of a CPS model.
- Develop an intuition for operational effects.

Educational Benefits.

CPS topics provide a number of intrinsic advantages that are beneficial for broader education:

1. **Motivation.** Given the significance of CPSs in a wide range of practical applications and how much we trust them with our lives, it is spectacularly easy to motivate students, to get them excited about CPS, and to fascinate them with the alluring subtle challenges that CPSs provide. This is especially so since CPSs quickly reach the limits of what is easy and obvious to understand. It is straightforward to spark the student's interest with deceptively simple-looking dynamical systems that exhibit surprisingly subtle behavior. This true motivation by the students has served them well for the perseverance required in mastering the most challenging aspects of the course.
2. **Mathematical.** CPSs provide a framework within which vast mathematical skills can be acquired and used right away. They provide a scientific playground in which mathematical theories can be explored more playfully and always with a concrete motivating critical problem in mind, rather than in isolation.
3. **Logical.** The stringent correctness requirements on a CPS's functionality make CPS an ideal vehicle to convey how logic and reasoning help in making real systems safer for everybody. In its current instances, the course has also attracted a sizable student body with no prior affinity to formal methods, not just those with a prior background. A background quiz in the first week of class revealed that most students did not have a good grasp of basic logic when starting the course. The level of mastery of logic has improved significantly during the CPS course as demonstrated, e.g., by the students' performance on nontrivial proof calculus questions on the final exam.

An indication that students are aware of these benefits and appreciate them can be found in course evaluations:

[2] The development of the undergraduate course has been based on the experience with two prior instances of a corresponding course for PhD students that the second author offered at CMU in 2009 and 2011.

> "This course maxed out my cool-new-stuff-learned-per-hour meter. [...] Addicting labs and assignments. [...] It was very neat to see so many concepts (both new to me and previously encountered) from so many different fields of study come together to build this subject."

Educational Challenges.

The most challenging aspects of CPS education have been identified [12] with a consensus among CPS experts [5]:

1. **Mathematical.** The high demand for mathematical sophistication cutting across numerous areas of mathematics, which are traditionally less seamlessly integrated than they have to be for CPS;
2. **Diverse.** The diversity of backgrounds from people interested in CPS, which reflects the large number of disciplines which CPS builds upon and is related to;
3. **Intellectual.** The intellectual challenges of understanding and identifying the complex interactions that different aspects of a CPS design have on its behavior;
4. **Impact.** The difficulties with making CPS designs safe, which is still a grand challenge in research, but whose current solutions must transition into widespread practice regardless, as system designs need to be safe today and not just patched after a problem has already caused harm to people.

The FCPS course addresses these challenges as follows. Challenge 1, which is dual to Benefit 2, is addressed by introducing the mathematical background gradually alongside the practical application challenges throughout the course, with a focus on the most important guiding principles and intuitions. Advantages have been identified in other courses when students study the theoretical background in the context of actual applications [2], which CPSs naturally provide.

Challenge 2 is mitigated by the interdisciplinary nature of the course, which ensures that all students are more familiar with some background area, but more challenged by another. Some students have had prior exposure to logic, while most did not. The level of prior exposure to differential equations also varied more widely depending on the background.

Challenge 3 is simplified by the principles underlying the course design. Continuous dynamics, discrete dynamics, temporal effects, and adversarial dynamics are introduced in succession in the course, for example, but integrate easily into a single common concept for understanding CPSs [10]. It is significantly easier to first understand these dynamical aspects separately and then understand how they interact, which is in line with results from learning theory [2].

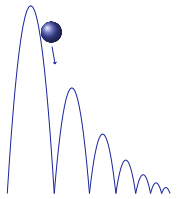
Challenge 4 is addressed as follows. The course design follows the realization that it is difficult, maybe impossible, to fully comprehend the subtleties and intricacies of CPS designs without the help of a technique that tells right and wrong designs apart. CPS designs can be flawed for very subtle reasons. Without sufficient rigor in their analysis it can be impossible to spot the flaws, and even more challenging to say for sure whether and why a design is no longer faulty. Analytic results about CPSs, even in cases where they only exist for parts of their operating conditions, help the CPS expert gain confidence in their design.

4. COURSE APPROACH

The CPS programming language that the course uses comes from *differential dynamic logic* (d$\mathcal{L}$) [10, 11], a specification and verification logic for hybrid systems. This logic and its programming language are used in the course for expressing CPS models and their desired correctness properties, and for proving them using the proof calculus of d$\mathcal{L}$. This calculus has separate simple proof rules for each of the operators, which makes it possible to understand CPSs one aspect at a time, rather than having to understand everything at once.

The following simple example shows how CPS programs combine differential equations and discrete programs. This program models the behavior of a bouncing ball:

```
{
  {x'=v, v'=-g, x ≥ 0};
  if (x = 0) {
      v := -c*v;
  }
}*
```



The differential equation system models how the ball is falling according to gravity. Its height x changes with velocity v, which, in turn, changes according to gravity g. The additional evolution domain constraint $x \geq 0$ represents the fact that the ball always stays above ground. The subsequent conditional statement tests if the ball is on the ground ($x=0$) and then changes v to $-c*v$ by an assignment. This models a bounce by changing the direction the ball is moving and decreasing the ball's energy according to a damping factor c. The continuous and discrete operations of the bouncing ball repeat, as indicated by the repetition $*$ at the end. One simple property to prove about the bouncing ball in d$\mathcal{L}$ is that x never exceeds its initial height. But it takes a surprising amount of thought to identify all the subtle assumptions that are required to even make this true. For example, the ball should not initially be thrown upwards, so its initial velocity must be ≤ 0. But it must not be thrown down too hard either, for then it would bounce back up higher than before, like a dribbled basket ball. These requirements, among others, quickly become evident when students attempt to prove that the ball does not exceed its initial height. Variations of the bouncing ball example are ideal subjects for the first lectures, because everyone can relate intuitively but they still convey quickly how subtle the interaction of discrete change and physics can become.

Even this simple system model already requires quite some thought to get right. Logical reasoning, thus, needs to go hand-in-hand with the system design and provides helpful assurance that the system design does not have subtle bugs.

In order to give the students the opportunity to tackle more interesting and more challenging system designs, this course does not limit specification & verification in differential dynamic logic to pen-and-paper proofs on the theory assignments. Rather, students interact with the implementation of d$\mathcal{L}$ in the theorem prover KeYmaera.

Course Format and Material.

The course features chalkboard lectures in which students are encouraged to contribute to the active development of all important concepts. Detailed lecture notes are released for later reference and independent study by the students. Lectures and concepts are listed in Appendix A. The lectures are complemented by recitations with a direct involve-

ment of the students and a focus on the practical aspects of KeYmaera as well as the study of examples.

Students continually practice and actively internalize the material on biweekly written theory assignments and biweekly modeling and verification labs of increasing conceptual complexity. They show mastery of the material in a midterm and final exam and in a capstone project and term paper at the end of the course. This allows students to learn skills individually and subsequently intensify their combined use in increasingly complex real-world applications [2].

The course material is augmented with short YouTube tutorial videos explaining how to prove properties in KeYmaera. Interacting with a prover can challenge students in ways that stretch their abilities and give them a deeper understanding of complicated CPSs. In our experience, students often do not realize the significance of all details in tutorials until they experience that their proof is stuck without it. This is the analogue of the well-studied phenomenon that active problem solving has a longer retention rate than topics merely stated in lecture [2]. The videos enable the students to rewatch user instruction as their understanding of CPS verification challenges matures over the course of the semester. The students responded positively to the video tutorials in the course evaluations.

The Role of Theorem Proving in CPS Education.

Using a theorem prover such as KeYmaera in an undergraduate CPS course has advantages and disadvantages:

- **More realistic models.** A theorem prover makes it possible for the students to work on system designs whose complexity vastly exceeds their capability of proving properties manually and, in fact, probably exceeds the capability and diligence of most people. This effect is partially due to the branching factor and depth of the proof and partially due to the overwhelming complexities of the resulting arithmetic. In the lab assignments, the students managed to verify extremely challenging robot control systems that, until recently, have been unsolved challenges in CPS verification [9].
- **More rigorous proofs.** Theorem provers do not accidentally let the user get away with hand-waving or subtly flawed arguments in support of a system design but require rigorous correctness proofs.
- **Guidance in conducting proofs.** The use of KeYmaera has the beneficial effect of offering students more guidance in how to conduct a proof. Rather than the blank page syndrome where students do not know how to begin a proof, KeYmaera helps students make some progress by offering applicable proof rules and structuring the proof for them.
- **Learning curve.** All verification tools have nontrivial learning curves and KeYmaera is no exception. Tools such as KeYmaera do, however, also guide the user in their understanding of the material by making it possible to learn how to prove a CPS by playing with the proof rules that the prover offers. For this effect to work, it is critical that KeYmaera only offers applicable proof rules, although an even more narrow focus on the most important proof rules might benefit the novice.
- **Automation surprise and loss of control.** Compared to manual proofs, the automation in KeYmaera bears a higher risk of making the user lose control over what the proof is doing and how it works. It has happened that some students got carried away in the proving excitement in KeYmaera without paying attention to the overall proof structure. Occasionally students will try to take a shortcut around fully understanding the property and proof that they are working on and attempt random clicking. Once control of the proof has been lost, some students realized too late that a case distinction (that KeYmaera's default proof strategies may follow automatically) had the adverse effect of branching into a huge number of cases. The right approach would then have been to scrutinize whether an assumption or invariant has been misplaced and exercise branching control techniques.

The permanent improvements that KeYmaera is undergoing, including those inspired by the experience with the FCPS course, will alleviate some of the concerns with the learning curve of verification tools. Even without that, though, we experience that the benefits of using KeYmaera in CPS education by far outweigh the downsides. Students across the board seem to have appreciated the opportunity for handling challenging applications. Moreover, students have appreciated the level of correctness that can be achieved for these complicated systems through formalization and proofs.

5. LAB DESIGN

The complexity of the model, control, physics, and verification in the labs increase simultaneously over the course of the semester. Each of the designed labs builds upon and concretizes the concepts the students learn in the lectures and in the theoretical assignments, a method which the students appreciated in their evaluations:

> "The structure of [the FCPS course] is good: theory, then the intuition, then example."
> "I think actually finishing proofs of the labs was really helpful. It was a great feeling to see it work, and required a few tricks and a solid understanding of proof rules and proof strategies."

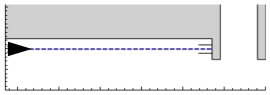
In each lab, the students must design a controller for a single robot that can interact with an unknown environment. The students are also required to design an appropriate model for the continuous behavior that their controlled robot would exhibit given the discrete control inputs. They must decide on an appropriate model for the robot's environment, including using nondeterminism to capture unknown behaviors in the environment. And finally, the students must formalize a safety property as a logical formula and prove that their controller never violates it.

The labs are all related and build on each other, with the ultimate goal that the students design and prove safety for a robot that can avoid moving obstacles (inspired by [9]). We give a brief description of these labs in Fig. 1, but believe this is only one example of a series of labs that could teach the same foundational elements of CPS and verification.

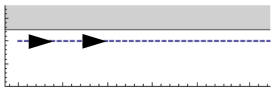
5.1 Beta Submissions

In order to ensure that students are on the right track, they submit a *betabot*[3] one week before the final deadline for each of the labs. Similar to a beta release of a software application, this betabot is not guaranteed to be error-free, as the students have not yet produced a proof of correctness

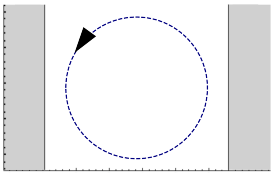
[3]In the CMU full semester offering of the course only.



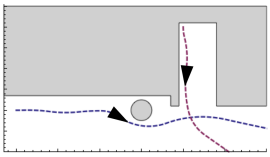
Lab 1: Charging Station



Lab 2: Follow the Leader



Lab 3: Racetrack



Labs 4&5: Static and Dynamic Obstacles

Lab 1. Students design a robot to stop on a charging station. If the robot overshoots, it hits the wall - a safety violation; if it stops short, it is inefficient.

Lab 2. Now their robot must follow a leader along a straight line. The leader may accelerate or brake at any time. By designing both event-driven and time-triggered solutions, they learn the difference between continuous and discrete sensing.

Lab 3. Students make the huge conceptual leap from straight-line dynamics to dynamics on a curve. Robots must drive on a circular track and are required to stop before reaching a stationary obstacle on the track.

Labs 4 & 5. Students design robots that can roam freely in two dimensions, starting by avoiding static obstacles in Lab 4, and then extending their solution to safely avoid dynamic obstacles in Lab 5.

Figure 1: Visual representation of one example scenario for each lab assignment. For each lab, students design a controller and physical model, then give a proof that the safety specifications are satisfied for all such environments.

for it. This gives the instructors a chance to return notes and generate a simulation (often of an unsafe behavior) of the betabot submissions.

It is impossible to prove safety for a system which is not in fact safe. And in CPS, it's especially challenging to write a controller that is correct due to the many lasting physical consequences of a single, seemingly simple, control choice. By giving this quick feedback to alert students to possible problem areas in their programs without giving them solutions outright, we remind them that difficulties in proving a property may be because the property is actually false. From the course evaluations, many students found the double submission policy to be helpful for solving the challenging labs:

> "[Beta] tests before labs are crucial! I would have failed the labs otherwise."

Other students, however, have also indicated that, if something needs to be cut from the course, it should be the simulations. This is a positive sign that students realize that simulations, although pretty, do not lead to the same level of understanding of a CPS or give the required level of safety guarantees that a proof does.

5.2 Star Lab: Students get creative

In the Star Lab[3], students submit a whitepaper proposing what system they would like to verify along with a preliminary model and partial proof for feedback. Surprisingly, students had a good grasp of the type and difficulty level of problems that would be appropriate for proving, and what specifications they might be able to verify about them. While some students wished they could have verified more challenging systems or more properties in the end, all students had a good prior grasp of which verification tasks they would be able to tackle and which extensions they would pursue if possible. Most students achieved in their final project submissions what they originally proposed to accomplish. Moreover many students delivered on the challenging extensions that they listed as stretch goals in their proposals. They also successfully identified the right level of abstraction for the models, which is a nontrivial but ubiquitous challenge in CPS. The students implemented a fascinating multitude of different verification projects for Star Lab (Appendix B).

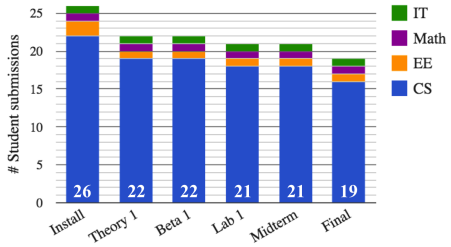


Figure 2: Attrition at CMU by assignments submitted (x-axis) broken down by student's background area.

6. ANALYSIS

In addition to collecting early feedback evaluations and final course evaluations, we also analyze the correctness of lab submissions, as well as some data on attrition.

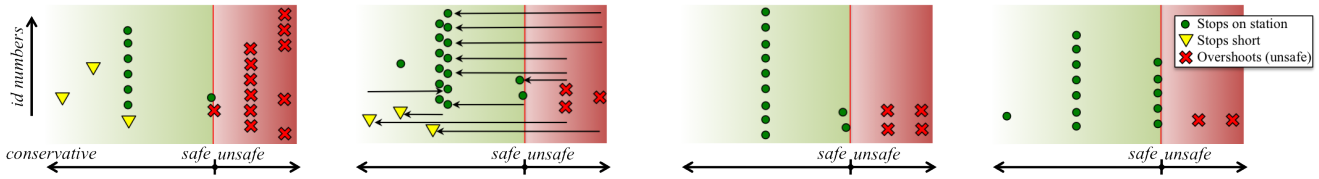
We include in Fig. 2 complete attrition data for the course at CMU. We take the low rate of attrition after the submission of the first written assignment (Theory 1) as an indicator that the course was successful and its topics were considered as significant by the students.

We measured the correctness of the beta submissions and of the final submissions (submitted one week later) for the first lab to see how well the CMU students' understanding of the material improved. Between these two submissions, the students were given written feedback, a simulation of the beta submission (often an example of unsafe behavior), and the students interacted with the theorem prover. A completed proof of correctness was due with the final submission of the lab.

Recall that in lab 1 the students were required to control acceleration and deceleration of a robot so that it would stop exactly on a charging station, but without overshooting and running into the wall behind (see Fig. 1). In Fig. 3 the submitted solutions plotted farther right are more aggressive in accelerating toward their goal. However, those that overshoot the charging station are plotted with a red X, and those that brake to a stop before they reach the station are plotted with a yellow triangle.

Comparing Fig. 3a with Fig. 3b, we can see the vast improvement that students made between their betabot submission and their final submission. This indicates the strong influence of written feedback, simulations, and requiring students to formally prove safety properties.

Surprisingly, the single-shot submissions from ENS Lyon and University of Minho (in Fig. 3c and 3d) are similar in correctness to the final submissions from CMU, despite not benefiting from getting feedback or a simulation from a beta submission. This demonstrates the power and impact that proofs of a CPS have on its ultimate correctness, as interacting with KeYmaera was the only method the students at



(a) CMU Lab 1 - Beta (b) CMU Lab 1 - Final (c) ENS Lyon Lab 1 (d) University of Minho Lab1

Figure 3: Illustrations of correctness of lab 1 submissions. Submissions plotted farther right are more aggressive in accelerating quickly to their goal. Fig. 3b also shows students' progress between the beta and final submissions, indicated by arrows.

ENS Lyon and University of Minho had for ensuring safety.

Similar observations hold for our analysis of the submissions to lab 2 (Appendix C).

7. CONCLUSIONS AND FUTURE WORK

The Foundations of Cyber-Physical Systems course is one of a kind. It is truly challenging and demanding, but for a higher value that students appreciate: making CPSs safe to entrust our lives. The course provides a framework for exciting expeditions into science at work in real applications.

In anonymous course evaluations, the students have widely expressed enthusiasm about how well the successive lab designs and lecture notes contributed to their learning. We take this as an indication that the course design strikes a good balance of challenge versus feasibility while capturing student motivation. Performance indicators in exams and final projects have clearly demonstrated successful and very impressive learning achievements by the students.

At the same time, the distribution of grades has been bimodal, indicating that a smoother introduction and more but smaller challenges might improve student learning. The experience with the use of the theorem prover KeYmaera in this course also highlights the importance of conceptual and technical advances in managing and understanding proof context and proof metastrategies.

Student evaluations expressed appreciation for how the lectures led the students to develop proof rules and conjectures and scrutinize all their required conditions and constraints, as opposed to just stating the final sound proof rule without critical reflection. We take this as an indication that the course setup has been successful at communicating the most fundamental aspect of them all: that it is more important to understand *why* something works (and why it could not work any other way) rather than just remembering *how* it works. This is also a positive indicator that the students have learned to critically reflect on whether they truly understand CPS and logic-related material.

Our overall takeaway message from the positive student feedback is that the advantages of the exciting and highly motivating topics that CPSs provide by far outweigh the risks and glitches, even in the first instance of the course. The benefits of using KeYmaera in an educational setting far outweighed the costs of technical challenges for both instructors and students. As both KeYmaera and course curricula evolve and improve, this tradeoff will balance ever more in favor of using logic and theorem provers in course curricula, particularly for topics as notoriously challenging as CPS.

The industrial significance of such additions to curricula are reflected in their desire to be involved in the course. The next instance of the course features a competition to which expert judges from at least 8 companies and organizations have already signed up to come to give feedback on the final project presentations of the students in the CPS V&V Grand Prix and to award prizes. While hard to measure, it would be interesting to get anecdotal evidence for how such a competition influences student motivation.

FCPS has been offered in quite different forms at three universities. By way of its development in gradual layers, the FCPS course has a modular design that allows for different ways of slicing the material for different needs (Appendix A). It would be interesting to get a broader experience with different configurations of the FCPS course at other places.

Acknowledgements

This material is based upon work supported by the National Science Foundation under NSF CAREER Award CNS-1054246 and NSF EXPEDITION CNS-0926181.

8. REFERENCES

[1] R. Alur. Formal verification of hybrid systems. In *EMSOFT*, pages 273–278, 2011.

[2] S. A. Ambrose, M. W. Bridges, M. DiPietro, M. C. Lovett, and M. K. Norman. *How Learning Works: Seven Research-Based Principles for Smart Teaching*. Jossey-Bass, 2010.

[3] A. M. Cheng. An undergraduate cyber-physical systems course. In *CyPhy Wksp*, pages 31–34, 2014.

[4] J.-P. de la Croix and M. Egerstedt. Flipping the controls classroom around a MOOC. In *ACC*, 2014.

[5] M. Egerstedt, R. Gupta, J. C. Jensen, and E. A. Lee, editors. *First Workshop on Cyber-Physical Systems Education*, 2013.

[6] M. Felleisen. TeachScheme! In T. J. Cortina, E. L. Walker, L. A. S. King, and D. R. Musicant, editors, *SIGCSE*, pages 1–2. ACM, 2011.

[7] E. A. Lee and S. A. Seshia. *Introduction to Embedded Systems*. Lulu.com, 2013.

[8] P. Marwedel. *Embedded System Design*. Springer, 2010.

[9] S. Mitsch, K. Ghorbal, and A. Platzer. On provably safe obstacle avoidance for autonomous robotic ground vehicles. In *Robotics: Science and Systems*, 2013.

[10] A. Platzer. *Logical Analysis of Hybrid Systems*. Springer, Heidelberg, 2010.

[11] A. Platzer. Logics of dynamical systems. In *LICS*, pages 13–24, 2012.

[12] A. Platzer. Teaching CPS foundations with contracts. In *CPS-Ed* [5], pages 7–10.

[13] J. M. Wing. Computational thinking. *CACM*, 2006.

APPENDIX

A. MODULAR LECTURE SEQUENCE

The succession of the most important lectures of the course is shown in Table 1 along with their primary educational concepts. The latter optional topics span more than one lecture but are condensed here for the sake of presentation. The dependencies of the labs on the lectures are indicated in the first column of Table 1, which allows for the modular selection of material and adaptation to different needs. This allows for other offerings of the course to change the duration or to adapt to audiences with different backgrounds.

Table 1: Modular lecture sequence

Lab	Lecture	Primary Purpose
1	CPS introduction	applications, criticality
	Differential equations & domains	continuous models
	Choice & control	discrete & hybrid system models
	Safety & contracts	CPS specifications
2	Dynamical systems & dynamic axioms	CPS verification & proof of dynamics
	Truth & proof	sequent proofs
	Control loops & invariants	loop unrolling & invariants
2a	Events & responses	event-driven control
2b	Reactions & delays	time-triggered control
3+	Differential equations & invariants	invariant functions
	Differential equations & proofs	induction for differential equations
optional	Ghosts & differential ghosts	auxiliary variables in proofs
	Differential & temporal logic	temporal behavior of CPS
	Virtual substitution & real arithmetic	proving real arithmetic
	Hybrid systems & games	adversarial dynamics in CPS

B. LIST OF SELF-DEFINED STAR LABS

The students at CMU in Fall 2013 implemented a fascinating multitude of different self-defined CPS verification projects for Star Lab:

Robot Projects

- Safe Ball Passing in RoboCup
- Modeling Wall-E and Eve: Verified robot dance choreography
- Robots on Treadmills

Car Projects

- Verified Centralized Automated Traffic Control for Cars

Air and Space Projects

- Verified Lunar Lander
- Verified Landing on an Aircraft Carrier
- Verified Hovercrafts and Helicopters Game

HVAC Projects

- Verified Space Heater
- Thermostat Modeling and Verification
- Modeling a Flow-modulated Radiator Heating System
- Modeling, Analysis and Verification of a Temperature Control System

Chemical/Biological/Circuit Projects

- Design and Safety of Insulin Pumps: Proving safety properties of a proposed controller
- Modeling Chemical Reactions as Hybrid Systems
- Fox and Duck Game
- Verified Electrical Circuits

These achievements of a wide range of verified CPSs are all the more remarkable as the students had only two weeks of class for the final project, due to delays in developing the assignments for the first instance of the course.

C. INVARIANT ANALYSIS

In cyber-physical systems, an *invariant* is a formula that is always satisfied by the system. An invariant is *inductive* if it is a strong enough formula that no additional assumptions besides the inductive invariant need to be added in order for the inductive invariant to hold always (also called a *loop invariant*, similar in concept to an *induction hypothesis*).

Inductive invariants are arguably the most crucial and challenging part of the core operating principles of a cyber-physical system. Designing an invariant for a CPS requires a deep understanding of how the system should behave now and at all times in the future. An invariant also needs to remain true for all possible future behavior of the environment, and, thus, requires considerable foresight. Moreover, the invariant must relate the system behavior to the desired safety property. And finally, the invariant must be strong enough that it is inductive, but not so strong that it restricts systems to be overly conservative. When a student balances all of these factors and designs a good inductive invariant, it demonstrates that the student has a solid grasp on the intricacies of the system.

C.1 Car Following Labs

Recall that in lab 2 the students were required to follow a leader robot along a straight line (see Fig. 1). Students at CMU were required to provide both event-driven and time-triggered solutions. In the later offerings of the course at ENS Lyon and University of Minho, we split these two problems into separate labs. In this section, we present and analyze student submissions from CMU, ENS Lyon, and University of Minho for both the event-driven and the time-triggered components of the lab.

If a student chooses an invariant that is *too weak* so that it allows too much behavior, then even unsafe systems may be able to satisfy such an invariant. In these cases, it is not possible to prove that the invariant is inductive, and therefore students will not be able to prove safety for any systems (safe or unsafe) using these invariants. This corresponds to the red regions with the yellow triangles on the right side of the subfigures in Fig. 4.

With the *optimal* invariant, the students can prove safety for any controller that is actually safe without imposing any restrictions on the system behavior that are not necessary for its safety. Note that an optimal invariant is guaranteed to exist in $\mathsf{d}\mathcal{L}$ [11] for every system, but may not be first-order. It is first-order for all labs in this course.

Moving farther to the left, if a student chooses an invariant that is *a bit too restrictive*, he/she may only be able to verify system controllers that are unnecessarily conservative. This means that there may be several safe controllers which are more efficient (e.g. a robot controller that can follow the leader more closely) and, even though they are still safe, the student has no hope to prove safety using this invariant. This corresponds to the green area to the left of the optimal invariant choice in Fig. 4. The farther left, the more restrictive the inductive invariant. For example, the most restrictive invariant that a student submitted (at the far left in Fig. 5c) requires that the velocity of the follow car start and remain zero. While this technically can guarantee safety, it is far too restrictive.

For the expert reader, the invariant formulas are plotted with respect to the partial order induced by implication, with stronger formulas plotted farther left. The few invariants which were incomparable under that partial order were ordered at the discretion of the authors, who plotted invariants that indicated a better understanding of the system dynamics closer to the optimal invariant.

C.2 Analysis of Event-Driven Submissions

Comparing Fig. 4a with Figs. 4b - 4d, where students had the benefit of interacting with a theorem prover, suggests that interacting with proofs enhances students' understanding of CPS. This conclusion reinforces our observations in Section 6.

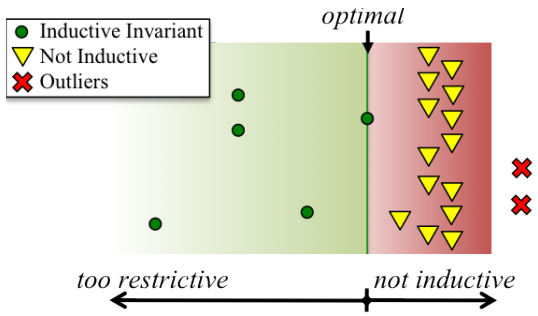
The submissions in Fig. 4c and Fig. 4d benefitted only from interacting with a theorem prover, while the CMU students were exposed to simulations, feedback, and theorem prover interactions between the beta submission (Fig. 4a) and the final submission (Fig. 4b). The data indicates that students who only had the interaction with KeYmaera did not come up with worse designs than those that also received feedback and simulations on a beta submission. While it may be tempting to conclude that the concrete examples that simulations depict actually hurt students' ability to design general inductive invariants, the authors are inclined to attribute this difference primarily to a lack of previous experience with challenging inductive proofs amongst the (more junior) students in the CMU course. This did not play a role in the analysis of the controller submissions for lab1, as that lab did not yet need inductive invariants. The CMU students worked individually, while the students in ENS Lyon and University of Minho were allowed to work in pairs. Additionally, as the CMU offering was the first of the three courses, the ENS Lyon and University of Minho students benefitted from updates to the lab writeup and lessons learned on how to teach related concepts, which affected lab 2 much more than lab 1.

C.3 Analysis of Time-Triggered Submissions

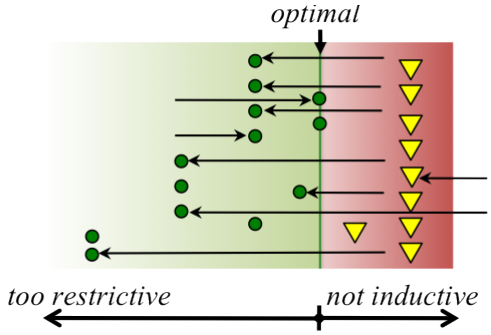
While both tasks are challenging, we assign the time-triggered lab second, since it requires students to design controllers that plan ahead for scenarios that the controller will not be able to react to instantaneously. The loop invariant for both systems, however, must capture some future behavior of the cars, or else it will not be strong enough to be invariant. In fact, the set of possible loop invariants would be the same for both labs, except that we make the event-triggered lab slightly easier by guaranteeing that the lead car maintains a constant velocity. This means that the optimal loop invariant for the event-triggered lab is not inductive for the time-triggered lab. We notice that CMU students, who were presented both labs in the same assignment, tended to reuse the same invariant for both (particularly noticeable in the Beta submission, shown in Figs. 4a and 5a). However, ENS Lyon and University of Minho students, who were presented each lab separately, were more likely to use different invariants for each.

When students are first required to add a time delay to the controller of their system, they are sometimes inclined to add the delay everywhere, including erroneously adding it into the loop invariant. We use an orange hour glass icon to identify students who made this common mistake in the time-triggered lab in Fig. 5.

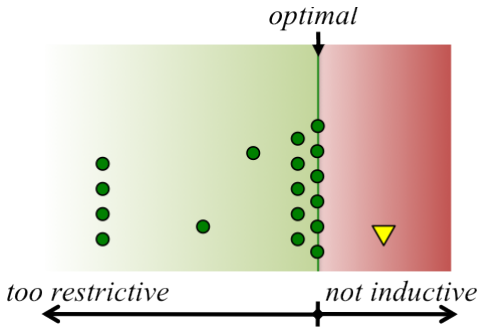
(a) CMU Lab 2a - Beta



(b) CMU Lab 2a - Final



(c) ENS Lyon Lab 2



(d) University of Minho Lab 2

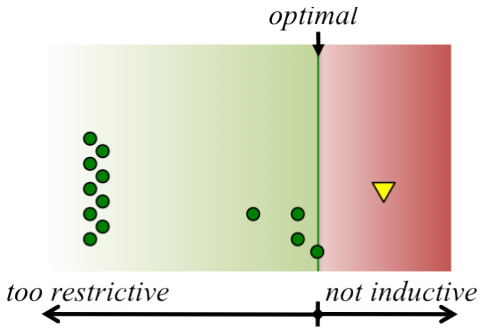
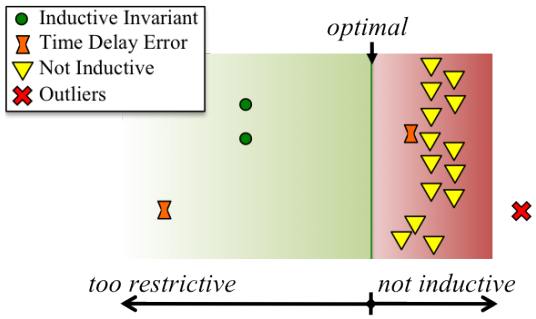
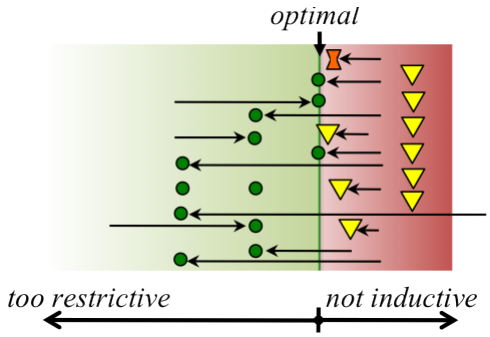


Figure 4: Illustrations of invariants for lab 2 (event-driven) submissions. Invariants plotted farther right allow controllers to exhibit a wider range of behavior. Invariants farther left restrict the system behavior more. Farther right is therefore generally better, until the invariants become too weak to be inductive. Fig. 4b also shows students' progress between the beta and final submissions, indicated by arrows.

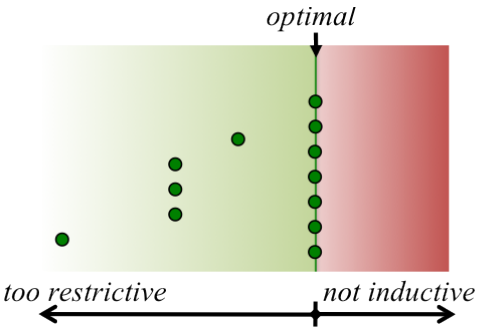
(a) CMU Lab 2b - Beta



(b) CMU Lab 2b - Final



(c) ENS Lyon Lab 3



(d) University of Minho Lab 3

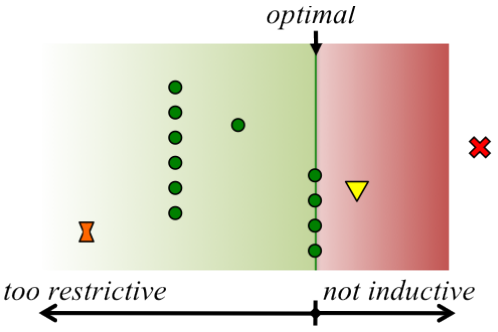


Figure 5: Illustrations of invariants for lab 2 (time-triggered) submissions. Invariants plotted farther right allow controllers to exhibit a wider range of behavior. Invariants farther left restrict the system behavior more. Farther right is therefore generally better, until the invariants become too weak to be inductive. Fig. 5b also shows students' progress between the beta and final submissions, indicated by arrows. The orange hour glass shape denotes an invariant that depends unnecessarily on the time delay. This error is common for beginners and is specific to time-triggered invariants.