Cyber-Physical Systems Education: Explorations and Dreams

Sanjit A. Seshia

Department of Electrical Engineering and Computer Sciences,
University of California, Berkeley, USA

Abstract. The field of cyber-physical systems (CPS), as a well-defined intellectual discipline, is entering its second decade. The past decade has seen several explorations in CPS education, accompanied by related research projects and technologies. This article reviews some of these explorations that the author has been involved with, and tries to extrapolate these to "dreams" for what the future may bring.

1 Prologue

In 2006, the term "cyber-physical systems" was coined by Helen Gill at the U.S. National Science Foundation to capture an emerging discipline concerned with the integrations of computation with physical processes. A nascent research community started to emerge, building on the momentum cutting across fields such as embedded systems, real-time systems, hybrid systems, control theory, sensor networks, and formal methods. Discussions began about developing curricula for training students interested in working in the broad area of cyber-physical systems.

During the 2006-07 academic year, a small group of UC Berkeley faculty in the Electrical Engineering and Computer Sciences (EECS) Department, including Edward Lee, Claire Tomlin, and myself, met to discuss the creation of an undergraduate curriculum in CPS. Berkeley had already been a pioneer in research and graduate education in CPS for several years, but there were still no undergraduate courses focusing on CPS. A major challenge was the breadth of topics needed to cover the area. A further challenge was to achieve a balance between theoretical content and practical, lab-based coursework. From the discussion, the basic contours of an undergraduate course emerged, and over the next decade it developed into a broader "expedition" in CPS education.

A major expedition to explore the unknown is best undertaken by a team. This has been true of some of the major expeditions in history, such as the Lewis and Clark expedition, and it is true of expeditions in research and teaching. In my case, I have been fortunate to undertake this expedition in CPS education with Edward Lee and several others. It has comprised several smaller explorations along "trails" in CPS education. This article is my attempt to report on these explorations, the results they obtained, and what we learned from them, and to extrapolate them to "dreams" for the future of CPS education.

2 Trail I: EECS 149

During the Spring 2008 semester, Edward Lee and I co-taught the first offering of *Introduction to Embedded Systems*, the undergraduate course in CPS we created at Berkeley [14].[1] I will refer to this course by its number in the Berkeley course catalog, EECS 149. We limited enrollment and advertised the class as being for "advanced and adventurous" undergraduates. A small class of about 20 students showed up. Over the course of the next 13 weeks, we explored a selection of topics in CPS together, blending theoretical topics with experimental work. It was a rewarding experience in many ways — perhaps most satisfyingly, two of the students in that class, Jeff Jensen and Trung Tran, went on to work closely with us on further developing the laboratory and online content for the course.

There were a few key decisions we faced in designing a new undergraduate course in CPS. At the time, we ended up making choices that just seemed natural to us. Ten years on, I believe these choices have proved crucial in developing a durable and unique CPS curriculum at Berkeley, one which is also starting to have a promising impact at institutions around the world.

Diversity of Topics and Backgrounds: The field of CPS draws from several areas in computer science, electrical engineering, and other engineering disciplines, including computer architecture, embedded systems programming languages, software engineering, real-time systems, operating systems and networking, formal methods, algorithms, theory of computation, control theory, signal processing, robotics, sensors and actuators, and computer security. How do we integrate this bewildering diversity of areas into a coherent whole?

One approach would be not to attempt such an integration. Instead, one could have a collection of courses that together cover all the key areas in CPS. However, we felt that this approach would have two major shortcomings. First, the reader can observe that the collection of areas could essentially end up covering a whole undergraduate program in electrical engineering and computer science! Second, in CPS there is a pressing need for people who understand the *intersection* between the various areas. We believed that the treatment of the subject of CPS would be best achieved by carefully selecting a collection of topics from the various areas and then presenting a unified treatment that emphasizes how they interact in the modeling, design, and analysis of CPS.

A related challenge was to deal with the diversity of backgrounds students bring to a course in CPS. Over the years, we have had students from computer science, electrical and computer engineering, mechanical engineering, civil engineering, and even bioengineering. Presenting a unified treatment of the various topics in CPS helps mitigate this challenge somewhat, by reducing the dependence on any specific collection of topics one might encounter in a specific engineering program.

Balancing Theory and Practice: Many courses on embedded systems focus on the collection of technologies needed to get computers to interact properly with the physical world, including sensor calibration, interfacing with sensors and other input/output

[1] At the time, the term "cyber-physical systems" was still in its infancy, so we decided to use "embedded systems" instead since, at least at Berkeley, we believed that they were essentially equivalent — two names for the same class of systems.

devices, programming in assembly or low-level languages, etc. Others focus more on applications, such as building a robotic system such as an autonomous vehicle or an Internet-of-Things (IoT) application. Still others focus entirely on theoretical topics, such as models of computation for CPS, or formal modeling and verification. With EECS 149, we decided very early on to blend theoretical topics with practical, lab-based work. Further we decided to build some flexibility into the lab work, splitting it into a 6-week structured lab sequence followed by a capstone design project whose topic students could choose for themselves. In particular, from the very beginning, a programmable, somewhat-customizable mobile robotic platform called the Cal Climber (see Fig. 1)[2] was chosen for the structured lab sequence, with the following assignment:

> *Design a controller to drive the Cal Climber. On level ground, your robot should drive straight. When an obstacle is encountered, such as a cliff or an object, your robot should navigate around the object and continue in its original orientation. On an incline, your robot should navigate uphill, while still avoiding obstacles. Use the accelerometer to detect an incline and as input to a control algorithm that maintains uphill orientation.*

Fig. 1. Cal Climber Laboratory Platform (early prototype).

This simple assignment allowed us to interleave basic conceptual and theoretical topics with the lab sequence: for example, students learned about modeling and interfacing with sensors and actuators in class as they interfaced to an accelerometer in the lab; they learned about programming with interrupts in class as they worked with an interrupt-driven controller in the lab, and they learned about modeling with state machines, composition, and hierarchy in class as they programmed their controller with StateCharts in the lab. We have found this interleaved presentation to help students gain an appreciation for theory in lab work and for motivating theoretical topics in class. We have seen students continue making these connections in their capstone projects, with very satisfactory learning outcomes.

A further aspect of balancing theory with practice is the emphasis in the course on formal methods and model-based design. *Formal methods* is a field of computer science and engineering concerned with the rigorous mathematical specification, design,

[2] In the initial years, this platform was the iRobot Create, the programmable version of the Roomba vacuum cleaner. Later, we moved to the very similar Kobuki platform.

and verification of computational systems [33, 1]. We decided early on to make formal methods and model-based design a key component of the course. In part, this is due to our research backgrounds involving extensive work on these topics. However, we took this step with some trepidation — after all, it is known from earlier experiments in undergraduate education that an emphasis on formal modeling and proof can be a bit "dry" and difficult for students. However, our overall experience has been very promising. Students realized the value of formal modeling, e.g., in reducing the number of design iterations needed to succeed in their capstone projects. This positive experience has extended beyond the community of Berkeley students to include students in the online course we offered a few years ago (see Sec. 4). Similarly, the integration of formal methods with practical lab work forced us to prune down the subject to a core set of formal methods topics that we found to be most relevant to an introductory course in CPS.

A more detailed discussion of the philosophy underlying the course, especially on the lab component and its integration with theoretical content, appears in [4, 19].

3 Trail II: The Lee & Seshia Textbook

By the fall of 2009, we had offered EECS 149 twice already. The course, as noted earlier, was unique in its coverage of a broad set of topics and its integration of theory and practical content. Edward Lee and I could not find a single book that could cover all the content we wanted to teach. Therefore, we started to develop our own course notes. Gradually the notes grew into something more coherent, and we decided to put them together as a textbook. In 2010-11, this effort culminated in the publication of the first edition of *Introduction to Embedded Systems: A Cyber-Physical Systems Approach* [16].

In earlier articles and the book's preface [15, 19, 16, 17], we have discussed at length the various design decisions made in writing the textbook, and how this book differs from other CPS textbooks. Therefore, we discuss here aspects that have not been covered in depth elsewhere.

Definition of CPS: A textbook on cyber-physical systems must define what that class of systems is. CPS have been informally described as integrations of computation with physical processes. Some definitions emphasize the networked aspect of these systems. Still others make distinctions between CPS and other terms such as the Internet of Things (IoT), embedded systems, the Industrial Internet, Industry 4.0, etc.

We chose an inclusive approach, formulating the following definition:

> A *cyber-physical system* (CPS) is an integration of computation with physical processes whose behavior is defined by *both* cyber and physical parts of the system.

This definition defines CPS as being about the *intersection*, not the union, of the physical and the cyber. It is not sufficient to separately understand the physical components and the computational components; we must instead understand their interaction. Note that we do not define the CPS as being networked or having other specific characteristics. We believe that the terms CPS, embedded systems, IoT, Industry 4.0, etc. are

essentially equivalent, describing the same class of systems while emphasizing different characteristics of those systems.

Emphasis on Models and Software: Mirroring the EECS 149 course, we decided to focus the book on the interplay of software and hardware with the physical processes with which they interact. Most specifically, we chose not to focus on the design of hardware components of CPS. These aspects are clearly important in any actual CPS design. However, with increasing commoditization of hardware, including processors, sensors, and actuators, we believe that the major intellectual challenge in CPS lies in how we can most effectively design algorithms, models and software to harness a combination of hardware, sensors, actuators, and networking components to achieve a design objective. Moreover, industry trends (e.g., in automotive systems) clearly indicate an explosive growth in the software in CPS – both in terms of opportunities and complexity. For these reasons, our textbook places an emphasis on rigorous mathematical models coupled with software design and implementation for the design of CPS. Such modeling and programming must be informed by hardware, no doubt. To that end, we present ways to effectively model relevant properties of hardware components so as to use those properties in making higher-level design choices.

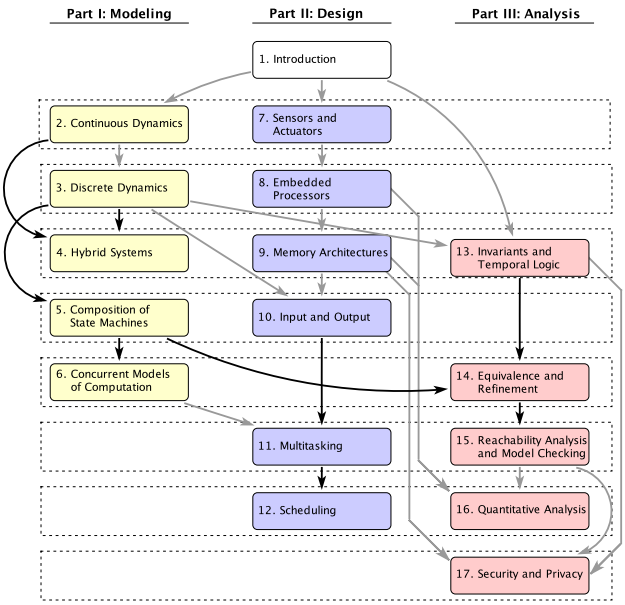

Fig. 2. Organization of the Lee & Seshia Textbook (2nd edition).

Organization: We made two key decisions in the organization of course content in the textbook. First, we decided to split the book into three parts, focusing respectively on *Modeling*, *Design*, and *Analysis*. Figure 2 gives the organization of the current edition

with the dependencies between chapters. The three parts are relatively independent of each other and are meant to be read concurrently. For example, in EECS 149, we interleave a discussion of programming with interrupts and threads (Part II on Design) with one on modeling with interacting state machines (Part I on Modeling). This enables students to see the connections between the more theoretical content on modeling with the more applied content on design and implementation. Moreover, it has enabled others to use the textbook for their own customized purposes — we are aware of more applied classes focusing mainly on Part II, whereas more theoretical classes on formal methods use portions of Parts I and III. Finally, we decided not to include details of laboratory exercises in this textbook. A separate lab book, co-authored with Jeff Jensen and in collaboration with National Instruments, was published online [5]. One reason for this separation is the difficulty in replicating a lab setup across institutions. We have found a far greater number of institutions using our textbook as compared to the lab book, and this is in part due to the various hardware and space dependencies, as well as support systems required to successfully replicate the lab content.

Publishing: We decided to try a publishing experiment with the first edition of this book. Rather than going with a traditional, established publisher, we decided to use a "print-on-demand" online publisher. This allowed us, amongst other things, to keep a free PDF version online while also providing readers with the option of purchasing a paper copy. Moreover, the paper copy was able to be sold at a much lower price that we believed would be the case with traditional publishers. Six years on, we believe this experiment made the right choices. As of this writing, our textbook has been adopted at around 300 institutions in over 50 countries — some of these, we are fairly sure, would not have been possible without the free PDF version being available online. Additionally, we found that many readers do purchase a paper copy even though a free PDF is available. Our most recent (second) edition is now published by MIT Press at what appears to be an affordable price and with a PDF available online for free. Having the PDF available online also made it easier for us to use the book in the online version of EECS 149 on edX, since enrollees around the world could consult the version they could most easily get their hands on.

4 Trail III: MOOCs and Exercise Generation

The advent of massive open online courses (MOOCs) [23] has promised to bring world-class education to anyone with Internet access. Additionally, it has placed a renewed focus on the development and use of computational aids for teaching and learning. MOOCs present a range of problems to which the field of formal methods has much to contribute. These include *automatic grading*, *automated exercise generation*, and *virtual laboratory environments*. In automatic grading, a computer program verifies that a candidate solution provided by a student is "correct", i.e., that it meets certain instructor-specified criteria (the specification). In addition, and particularly when the solution is incorrect, the automatic grader (henceforth, *auto-grader*) should provide feedback to the student as to where he/she went wrong. Automatic exercise generation is the process of synthesizing problems (with associated solutions) that test students' understanding of course material, often starting from instructor-provided sample problems. Finally, for

courses involving laboratory assignments, a virtual laboratory (henceforth, lab) seeks to provide the remote student with an experience similar to that provided in a real, on-campus lab.

In 2011-12, we started to brainstorm about an online version of EECS 149, and what technologies we could develop to aid in creating such an online course. We first looked at the task of automatic exercise generation. The term "automatic" may seem to indicate a goal of completely automating the process of creating problems and solutions. However, we felt that it would be unrealistic and also somewhat undesirable to completely remove the instructor from the problem generation process, since this is a creative process that requires the instructor's input to emphasize the right concepts. Automation is best employed in those aspects of problem generation that are tedious for an instructor. Additionally, in the MOOC setting, generating customized problems for students is impossible without some degree of automation. Finally, creating many different versions of a problem can help to reduce cheating by blind copying of solutions.

Examining problems from all three parts of the Lee and Seshia textbook [16], Sadigh et al. [26] take a *template-based approach* to automatic problem generation. Specifically, several existing exercises in the book are shown to conform to a template. The template identifies common elements of these problems while representing the differentiating elements as parameters or "holes". In order to create a new problem, the template essentially must be instantiated with new parameter values. However, it is often useful to create new problems that are "similar" in difficulty to existing hand-crafted problems. To facilitate this, new problems are generated using a bounded number of *mutations* to an existing problem, under suitable constraints and pruning to ensure well-defined results. An instructor can then select results that look reasonable to him or her.

For brevity, we outline some of the main insights reported in [26] as they relate to the application of formal methods. The first insight relates to the structure of exercises After investigating the exercises from certain relevant chapters (Ch. 3,4,9,12,13) of Lee and Seshia [16], we found that more than 60% of problems fit into the model-based category, where the problem tests concepts involving relationships between models, properties and traces. Figure 3 is an illustration of the three entities, and their characteristics. At any point, given one or two of these entities, we can ask about instances of the unknown entity. Table 1 groups exercises into different classes based on what is given and what is to be found. Each group represents an interaction between models, properties and traces. The first column shows the given entity, and the second column is the unknown entity. The third column shows some of the variations of the same class of problem.

Table 2 states a solution technique for each problem category listed in Table 1. Note that major topics investigated in formal methods such as model checking, specification mining, and synthesis can be applied to various tasks in exercise generation. Moreover, since textbook problems are typically smaller than those arising in industrial use, their size is within the capacity of existing tools for synthesis and verification.

5 Trail IV: EECS 149.1x and CPSGrader

During 2012-13, we began a concerted effort to develop a MOOC version of EECS 149. Berkeley had joined edX as a university partner, and a large campus effort was under-

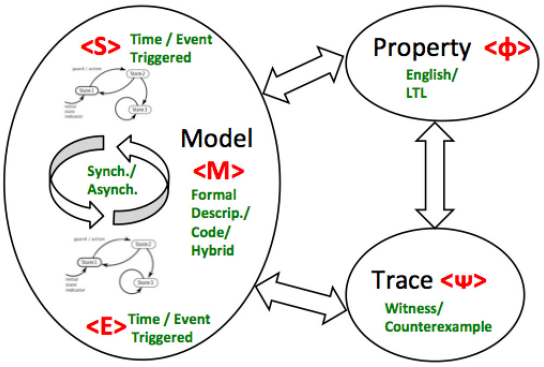

Fig. 3. Models, Properties and Traces: Entities in exercises in Lee and Seshia textbook [16] (reproduced from [26]).

Given	Find	Variations	Exercise #
$\langle\phi\rangle$	$\langle M\rangle$	(i)$\phi \in$ English or LTL (ii)use hybrid systems for M (iii)Modify pre-existing M	3.1, 3.2, 3.3, 4.1, 4.2, 4.3, 4.4, 4.5, 4.6, 4.8, 9.4, 9.6, 13.2, 13.3
$\langle M\rangle$	$\langle\psi\rangle$	(i)reachable trace (ii)describe output	3.3, 3.5, 4.2
$\langle M\rangle$	$\langle\phi\rangle$	Models can be given in code or formal description	3.2, 12.3
$\langle M\rangle$ & $\langle\psi\rangle$	$\langle\psi\rangle$	Given input trace $\rightarrow$ find output trace	9.5
$\langle M\rangle$ & $\langle\phi\rangle$	$\langle\psi\rangle$	Find counterexample or witness trace	3.4, 4.3, 12.1

Table 1. Classification of Model-Based Problems in Lee and Seshia [16], First Edition, Version 1.06 (reproduced from [26])

Given	Find	Solution Technique
$\langle\phi\rangle$	$\langle M\rangle$	Constrained Synthesis or Repair
$\langle M\rangle$	$\langle\psi\rangle$	Simulation of Model
$\langle M\rangle$	$\langle\phi\rangle$	Specification Mining
$\langle M\rangle$ & $\langle\psi\rangle$	$\langle\psi\rangle$	Simulation with Guidance
$\langle M\rangle$ & $\langle\phi\rangle$	$\langle\psi\rangle$	Model Checking

Table 2. Techniques to Find Solutions for Model-Based Problems (reproduced from [26])

way to engage with the emerging landscape on large-scale online education. However, taking EECS 149 online was not going to be easy.

One the one hand, with the growing interest in CPS from academia and industry, there was a clear demand for making educational resources in CPS more widely accessible. Having materials from EECS 149 and the Lee-Seshia textbook available freely online made it easier to offer a free MOOC to the broader community. On the other hand, a major challenge was posed by the lab component of the course. Lab-based courses that are not software-only, such as EECS 149, pose a particular technical challenge for MOOCs. A key component of learning in lab-based courses is working with hardware, getting "one's hands dirty." It appears extremely difficult, if not impossible, to provide that experience online. And yet, it is undeniably useful to provide a learning experience that approximates the real lab as well as possible. Indeed, in industrial design one often prototypes a design in a simulated environment before building the real artifact. Thus, we decided to build a virtual laboratory environment for EECS 149, and blend that with suitable theoretical content to create the MOOC version.

Working with Edward Lee and Jeff Jensen, and a team at National Instruments, my research group and I developed courseware and technologies for an online course in CPS. In Spring 2013, we presented a paper sketching out our main ideas for a virtual lab in CPS [6]. In 2013-14, we began an effort that culminated in two key contributions: EECS 149.1x [18], the online version of EECS 149 offered on edX in 2014, and CPSGrader, an automatic grading and feedback system for virtual laboratory environments [9]. We describe both these efforts in more detail below.

5.1 CPSGrader

In an ideal world, we would provide an infrastructure where students can log in remotely to a computer which has been preconfigured with all development tools and laboratory exercises and gives the students a view into how their solution is executing in the real lab setting; in fact, pilot projects exploring this approach have already been undertaken (e.g., see [22]). However, in the MOOC setting, the large numbers of students makes such a remotely-accessible physical lab expensive and impractical. A virtual lab environment, driven by simulation of real-world environments, appears to be the only solution at present.

To this end, we developed CPSGrader, which combines virtual lab software with automatic grading and feedback for courses in the areas of cyber-physical systems and robotics [10, 8, 9]. CPSGrader has been successfully used in both the on-campus *Introduction to Embedded Systems* at UC Berkeley [14] and its online counterpart on

edX [18]. Recall that in the lab component of this course, students program the Cal Climber [5] robot (see Fig. 1) to perform certain navigation tasks like obstacle avoidance and hill climbing. Students can prototype their controller to work within a simulated environment based on the LabVIEW Robotics Environment Simulator by National Instruments (see Figure 4 and 5).

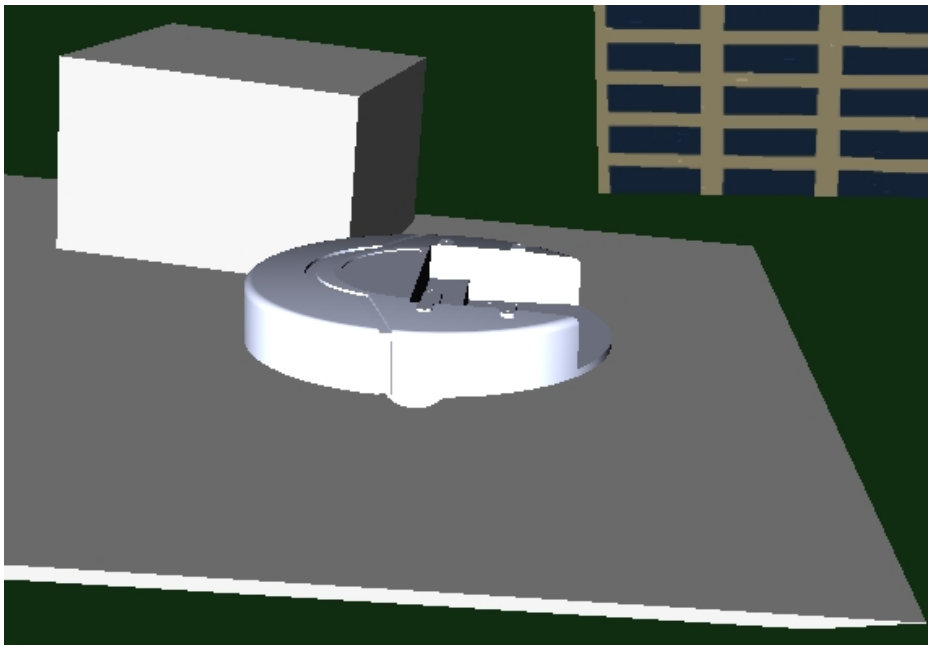

Fig. 4. Cal Climber in the LabVIEW Robotics Environment Simulator.

The virtual lab dynamical model is a complex, physics-based one, which, due to its complexity and dependence on third party components, we decided to treat as a black box. CPSGrader employs simulation-based verification, arguably the main scalable formal approach in this setting. Correctness and the presence of certain classes of mistakes are both checked using *test benches* formalized in *Signal Temporal Logic* (STL) [21]. However, coming up with these STL properties can be tedious and error-prone, even for instructors well-versed in formal methods. Therefore, in Juniwal et al. [10], we showed how these temporal logic testers can be synthesized from examples. Our approach can be viewed as an instance of machine learning from student solutions that have the fault (positive examples) and those that do not (negative examples). An active learning framework has also been developed to ease the burden of labeling solutions as positive or negative [8]. In machine learning terminology, this can be thought of as the *training* phase. The resulting test bench then becomes the *classifier* that determines whether a student solution is correct, and, if not, which fault is present. CPSGrader was

used successfully in the edX course EECS149.1x offered in May-June 2014 [18], an experiment we describe in more detail in Sec. 5.2.

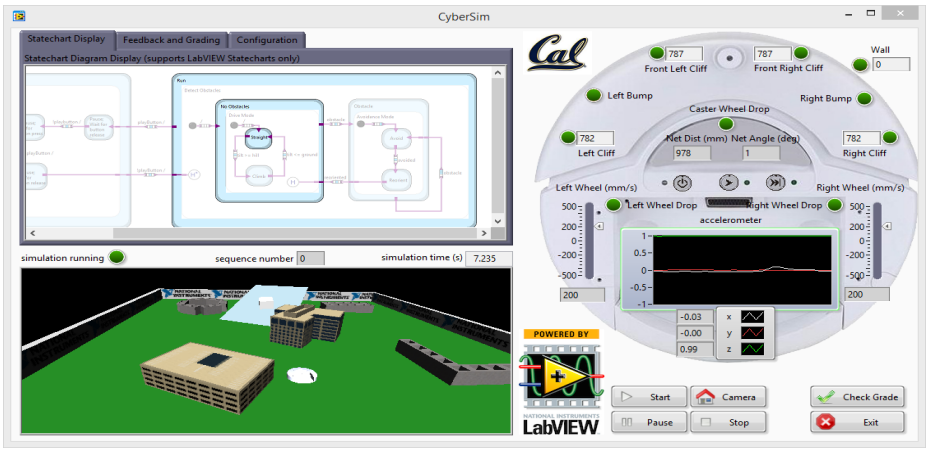

Fig. 5. Simulator with auto-grading functionality used in EECS 149.1x.

There are several interesting directions for future work, including developing quantitative methods for assigning partial credit, mining temporal logic testers to capture new classes of student mistakes, and online monitoring of these testers to improve responsiveness.

5.2 EECS 149.1x

Over a 7-week period in May-June 2014, we offered *EECS 149.1x: Cyber-Physical Systems* free to the public on the edX platform. To our knowledge, this was the first MOOC covering a breadth of topics in CPS offered on any of the major platforms. We simplified some of the course content from the material in EECS 149, since we could not rely on students in the MOOC having the same pre-requisite background that UC Berkeley students possess. The course included 49 lectures comprising nearly 11 hours of video content. It also included 6 weekly lab assignments that somewhat mirrored the 6-week structured lab sequence in EECS 149. The Cal Climber lab was turned into an entirely online lab to be performed using the virtual lab software we created — CyberSim and CPSGrader — described in the preceding section. In addition, we had an optional "hardware track" for those students who were open to purchasing and assembling the hardware components themselves. The theoretical course topics covered in the lectures included inline quizzes but no separate homework assignments — the lab sequence was integrated with the theoretical course content. We organized the lectures into the following ten modules: Introduction to CPS; Memory Architectures; Interrupts; Modeling Continuous Dynamics; Sensors and Actuators; Modeling Discrete Dynamics;

Extended and Hybrid Automata; Composition of State Machines; Hierarchical State Machines, and Specification & Temporal Logic.

The impact in MOOCs can be difficult to quantify, but here are some numbers from EECS149.1x. The course attracted a peak enrollment of 8767, of which 2213 ended up submitting at least one lab assignment. Of these, the number who passed the course was 342 (4% of peak enrollment). Around the 6th week of the course, we conducted an anonymized survey of the students still engaged in the MOOC. This produced some very interesting and encouraging data, which I summarize here:

- We seemed to attract a population of students who had already taken other MOOCs – 54% of those who stayed until the 6th week had taken 3 or more MOOCs.
- Of those who had taken at least one other MOOC, over 80% of the students rated our course as good or better than the one(s) they had previously taken.
- A majority of the students were new to model-based design with a language like LabVIEW, but even so, 73% found it to be a useful experience to do the lab assignment in two different languages, C and LabVIEW.
- 86% of the students rated CPSGrader as being useful in their lab assignments – an encouraging sign for the use of such tools for personalized education.
- A small fraction of students did the optional hardware track. Of these, over 90% found that if their solution passed CPSGrader in the virtual lab, it worked on the real hardware! This statistic was a really encouraging piece of data for the CPSGrader project. We had been using CyberSim in the on-campus lab for a couple of years, but saw no such correlation. However, using CPSGrader seems to have pushed students to debug the corner cases that also improved the reliability of their solution on the real hardware.
- We polled the students on the lecture modules they liked most, both from the viewpoint of relevance to lab assignments and from a theoretical standpoint. This poll was motivated in part for us to learn whether a general, non-Berkeley student population, including several students working in industry, would be receptive to the large formal methods content in the MOOC. To our pleasant surprise, we found that the modules on Hierarchical State Machines and Composition of State Machines were in the top three topics found relevant to the lab, while the module on Formal Specification and Temporal Logic ranked in the top three theoretical topics. This provides some real-world validation that relevance of and receptiveness to formal methods in a general student population.

In summary, creating the online version of EECS 149 was a lot of fun, and a tremondously rewarding experience. It is a first step towards creating a strong learning experience for "lab-based MOOCs" in science and engineering. The CPSGrader software is available open source and is architected to work with any simulator. Moreover, its success points the way to a broader application of formal methods for enhancing science and engineering education.

6 Dreams for the Future

As I think about the future of CPS education, and of engineering education in general, two sets of articles come to mind. The first is a pair of thought-provoking articles written

by Lee and Messerschmitt as the twentieth century drew to a close. One article discussed the future of engineering education, focusing, in particular, on electrical and computer engineering [12]. The other article presented an innovative viewpoint on what higher education might look like in the year 2049 [13]. The other set of articles has to do with the recent studies and opinions about the impact of automation and information technology (IT) on jobs (e.g., [2, 31, 32]).

A few threads emerge. First, with rapid technological change, and increasing automation, the need for lifelong learning becomes ever more important. Second, humans will increasingly need to collaborate with intelligent machines in their jobs. How should we design engineering and CPS education for such a future?

I will approach this question from the viewpoint of leveraging the work described in the preceding four sections.

The landscape for CPS education in the near future looks very exciting, with opportunities for further innovation. For inspiration, we can look to the emerging areas for research and industrial practice in CPS. There are at least three such areas that are not adequately covered by EECS 149 and the textbook: (i) networking and distributed CPS; (ii) human-CPS (CPS that work in concert with humans), and (iii) CPS that make extensive use of machine learning. All three are active areas of research by the CPS community, and are finding broad applications in the real world. For example, the TerraSwarm research center has developed a number of innovations in the area of networked, distributed CPS (see, e.g., [3, 11, 28]), and some of these ideas have had a direct impact on the material in EECS 149. Similarly, the CPS community is starting to bring the strong formal approach to modeling, design and analysis that underlies EECS 149 to the design of human-CPS, including modeling human cognition, perception, situational awareness, and action (see, e.g., [29, 24, 20, 25, 28]). Additionally, even as machine learning becomes more pervasive in CPS, recent advances in formal design and analysis of learning-based CPS (e.g., [27, 30, 7]) point the way to teaching a principled approach to designing such systems to students and practitioners. Currently, in EECS 149, we cover these topics mainly through the capstone design projects. Over the next decade, we see them making their way into the core curriculum, although much research remains to be done.

Some institutions around the world are seeing rapid growth in the number of students who want to take classes and major in computer science and related areas. CPS/IoT is already starting to be one of these areas. These institutions are seeing pressure on campus teaching resources due to burgeoning enrollments. Technologies for personalized education can play a role in reducing this pressure while ensuring that instructor attention is used more effectively. They can also play a role in broadening access to CPS education.

It is much harder to predict what role CPS education can play to mitigate the impact of automation and IT on jobs. It is clear that technology is only part of the solution (e.g., see [32, 2]). Even so, technologies for personalized education, such as CPSGrader, can help by providing students with personalized feedback on their work, even in online courses. Virtual lab technologies can enable students to build up expertise in a vocational topic which can then make them more attractive to prospective employers. However, the experience with several MOOCs so far has shown that the students who

benefit the most are the ones who were most motivated and best prepared in the first place. How can one ensure that students who need more help can benefit as well? For this, we need more work to develop structures such as the virtual "village" discussed by Lee and Messerschmitt [13], where technologies like CPSGrader are integrated with a community of mentors and peers forming a support system for the students.

Acknowledgments

I thank the anonymous reviewers for their feedback. The work described in this article was supported in part by the National Science Foundation, by a gift from National Instruments, and by TerraSwarm, one of six centers of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA.

References

1. Edmund M Clarke and Jeannette M Wing. Formal methods: State of the art and future directions. *ACM Computing Surveys (CSUR)*, 28(4):626–643, 1996.
2. Committee on Information Technology, Automation, and the U.S. Workforce. Information technology and the U.S. workforce: Where are we and where do we go from here? http://www.nap.edu/24649.
3. Edward A. Lee *et al.* The swarm at the edge of the cloud. *IEEE Design & Test*, 31(3):8–20, 2014.
4. Jeff C. Jensen, Edward A. Lee, and Sanjit A. Seshia. An introductory capstone design course on embedded systems. In *Proc. International Symposium on Circuits and Systems (ISCAS)*, pages 1199–1202, May 2011.
5. Jeff C. Jensen, Edward A. Lee, and Sanjit A. Seshia. *An Introductory Lab in Embedded and Cyber-Physical Systems*. LeeSeshia.org, Berkeley, CA, 2012.
6. Jeff C. Jensen, Edward A. Lee, and Sanjit A. Seshia. Virtualizing cyber-physical systems: Bringing CPS to online education. In *Proc. First Workshop on CPS Education (CPS-Ed)*, April 2013.
7. Susmit Jha and Sanjit A. Seshia. A Theory of Formal Synthesis via Inductive Learning. *Acta Informatica*, 2017.
8. Garvit Juniwal. CPSGrader: Auto-grading and feedback generation for cyber-physical systems education. Master's thesis, EECS Department, University of California, Berkeley, Dec 2014.
9. Garvit Juniwal, Alexandre Donzé, Jeff C. Jensen, and Sanjit A. Seshia. CPSGrader website. `http://www.cpsgrader.org`.
10. Garvit Juniwal, Alexandre Donzé, Jeff C. Jensen, and Sanjit A. Seshia. CPSGrader: Synthesizing temporal logic testers for auto-grading an embedded systems laboratory. In *Proceedings of the 14th International Conference on Embedded Software (EMSOFT)*, October 2014.
11. Elizabeth Latronico, Edward A Lee, Marten Lohstroh, Chris Shaver, Armin Wasicek, and Matthew Weber. A vision of swarmlets. *IEEE Internet Computing*, 19(2):20–28, 2015.
12. Edward A. Lee and David G. Messerschmitt. Engineering and education for the future. *IEEE Computer*, 31:77–85, 1998.
13. Edward A. Lee and David G. Messerschmitt. A highest education in the year 2049. *Proceedings of the IEEE*, 87(9):1685–1691, 1999.

14. Edward A. Lee and Sanjit A. Seshia. EECS 149 course website. http://chess.eecs.berkeley.edu/eecs149.
15. Edward A. Lee and Sanjit A. Seshia. An introductory textbook on cyber-physical systems. In *Proc. Workshop on Embedded Systems Education (WESE)*, October 2010.
16. Edward A. Lee and Sanjit A. Seshia. *Introduction to Embedded Systems - A Cyber-Physical Systems Approach*. LeeSeshia.org, Berkeley, CA, first edition, 2011.
17. Edward A. Lee and Sanjit A. Seshia. *Introduction to Embedded Systems - A Cyber-Physical Systems Approach*. MIT Press, second edition, 2016.
18. Edward A. Lee, Sanjit A. Seshia, and Jeff C. Jensen. EECS149.1x Course Website on edX. https://www.edx.org/course/uc-berkeleyx/uc-berkeleyx-eecs149-1x-cyber-physical-1629.
19. Edward A. Lee, Sanjit A. Seshia, and Jeff C. Jensen. Teaching embedded systems the berkeley way. In *Proceedings of the Workshop on Embedded and Cyber-Physical Systems Education, WESE 2012, Tampere, Finland, October 12, 2012*, page 1, 2012.
20. Wenchao Li, Dorsa Sadigh, S. Shankar Sastry, and Sanjit A. Seshia. Synthesis for human-in-the-loop control systems. In *Proceedings of the 20th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, pages 470–484, April 2014.
21. Oded Maler and Dejan Nickovic. Monitoring temporal properties of continuous signals. In *FORMATS/FTRTFT*, pages 152–166, 2004.
22. Massachusetts Institute of Technology (MIT). The iLab Project. https://wikis.mit.edu/confluence/display/ILAB2/Home, Last accessed: February 2014.
23. Laura Pappano. The Year of the MOOC. http://www.nytimes.com/2012/11/04/education/edlife/massive-open-online-courses-are-multiplying-at-a-rapid-pace.html, November 2012.
24. Dorsa Sadigh. *Safe and Interactive Autonomy: Control, Learning, and Verification*. PhD thesis, EECS Department, University of California, Berkeley, Aug 2017.
25. Dorsa Sadigh, Shankar Sastry, Sanjit A. Seshia, and Anca D. Dragan. Information gathering actions over human internal state. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 66–73, October 2016.
26. Dorsa Sadigh, Sanjit A. Seshia, and Mona Gupta. Automating exercise generation: A step towards meeting the MOOC challenge for embedded systems. In *Proc. Workshop on Embedded Systems Education (WESE)*, October 2012.
27. Sanjit A. Seshia. Combining induction, deduction, and structure for verification and synthesis. *Proceedings of the IEEE*, 103(11):2036–2051, 2015.
28. Sanjit A. Seshia, Shiyan Hu, Wenchao Li, and Qi Zhu. Design automation of cyber-physical systems: Challenges, advances, and opportunities. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 36(9):1421–1434, 2017.
29. Sanjit A. Seshia, Dorsa Sadigh, and S. Shankar Sastry. Formal methods for semi-autonomous driving. In *Proceedings of the Design Automation Conference (DAC)*, pages 148:1–148:5, June 2015.
30. Sanjit A. Seshia, Dorsa Sadigh, and S. Shankar Sastry. Towards Verified Artificial Intelligence. *ArXiv e-prints*, July 2016.
31. Moshe Y. Vardi. Humans, machines, and the future of work. In *Ada Lovelace Symposium 2015 - Celebrating 200 Years of a Computer Visionary, Ada Lovelace Symposium 2015, Oxford, UK, December 10, 2015*, page 2, 2015.
32. Moshe Y. Vardi. The moral imperative of artificial intelligence. *Commun. ACM*, 59(5):5, 2016.
33. Jeannette M Wing. A specifier's introduction to formal methods. *IEEE Computer*, 23(9):8–24, September 1990.