TEACHING STATEMENT

# Building the educational foundations of Responsible Autonomy

Houssam Abbas
Assistant Professor, Oregon State University
December 2017

## 1 Why I teach

*My educational agenda is to build a comprehensive curriculum in Responsible Autonomy: the science and practice of building autonomous machines that are safe, affordable, re-usable, and subject to ethical and social considerations.* This agenda guides my course design and teaching approach, which appeals to a technically and socially diverse student body. Self-driving cars and drones raise many possibilities for improving our well-being, but also many social questions. Our students, be they engineers or from other majors, *will* deal with Autonomy, broadly construed: they might use it, build it, or even oppose it. Whatever they choose, our role is to give them the necessary tools to inform their thinking.

*These tools are technical, ethical, and historical.* In my teaching and interactions with mentees, I complement the technical depth of the material with an exposition of its broader context. For example, in my class Digital Twins, we learn how autonomous systems can be rigorously verified. Along the way, we survey the history of formal logic in Philosophy, and the ambiguities of using natural language for specifying what is 'correct' or 'true'. We conclude the course with a student seminar on the 'automation paradox': how automation causes rarer but more catastrophic accidents as the skills of human operators atrophy and they become unable to respond when automation fails. This in turn has generated a discussion of rare events and the role of probability in formal verification.

*I also teach because I love sharing the excitement of discovery.* This excitement is what led me to a career in engineering and what has sustained me in it. I want for my students to experience that. So they must be allowed to be lost in the material for a while, and struggle to find a way out: find that key argument in a proof, or key step in an experiment. This can be a bit disorienting: some students have indicated that some assignments were *too* open-ended. But that is why I dedicate two classes after every large assignment to a collective process of solution-finding, and I record these classes so students can re-live those arguments, and understand *why* an answer is a certain way.

*I believe that training a diverse group of students in the tools of engineering and computer science is necessary to develop ethical autonomous systems and an equitable world.* I work diligently to create an inclusive classroom where historically underrepresented student populations can thrive. This semester at the University of Pennsylvania, one third of my students identify as women. Given their marginalization in the field, I am particularly attentive to their needs, make sure that their questions are central to my lectures and explanations, and highlight their contributions. I structured the student seminar so the women in each group lead their group's presentation. I also offer additional office hours to students who might need extra help building their confidence to participate in class. Moreover, when we think - as we must - of how to engineer an ethics into autonomous systems, we must ask: Whose ethics? Is it an inclusive ethics, which recognizes the diversity of experiences in U.S. society? Training a diverse engineering student body and workforce, which includes historically under-represented and disenfranchised minorities, is a necessary step towards developing such an ethics. Otherwise, we risk replicating the biases of our society in the systems we build, as already evidenced by the use of machine learning in the banking industry. By increasing the representation of minorities in the sciences, we can help ensure that the discussion around tomorrow's technology and the research priorities we set ourselves

truly involve everyone with a stake in this future.

## 2 Experience in course development and teaching

At the University of Pennsylvania, I co-developed and taught *Digital Twins*, a course on Cyber-Physical Systems modeling and implementation; a module on *Autonomous Racing*; and I will be co-teaching a *Massive Online Open Course (MOOC) on Embedded Systems.*

**ESE 680. F1/10 Autonomous Racing (Fall 2018 - 80 contact hours)** This course teaches how to build and program a $1/10^{th}$-scale autonomous race car! Students learn the fundamental operations of Perception, Planning, and Control, and apply these lessons to make their car the fastest. Students build the car from scratch in the first week. Every subsequent week, they learn the fundamentals of a task, like localization, and implement it in simulation and on the car. Theoretical concepts include convex optimization, randomization, and parallel data manipulation.

It was important in this class to put the fundamentals of Perception, Planning, and Control at the center of my teaching. Because there are so many techniques and 'tricks' that people use to make the car work, I had to avoid giving the impression that anything could be handled by enough tweaking of parameters. Thus, for instance, I brought the discussion back to data structures when the code is slow, back to modularity when the team has trouble coordinating their code development, back to the mathematical formulation when the results are surprisingly bad. It was also fun to draw parallels between our own human driving styles, and some of the navigation and perception algorithms that we were studying: looking straight ahead (bad for humans and for robots), situational awareness (good), driving with your eyes closed (very bad). The students gained a more intuitive understanding of the algorithms via these analogies and used them as jump-off points to improve them.

Early versions of the course material have been used for classes and projects at UT Austin and Clemson University. High schoolers in Portland, OR, are using the F1/10 autonomous race car in an intensive 3-week Advanced Programming Workshop: see `https://www.youtube.com/watch?v=vWb3oF7GD2E`. A reduced version of the material is available free online at `http://f1tenth.org`.

**ESE680 Digital Twins: Model-Based Embedded Systems (Fall 2017 and Spring 2018)** Digital Twins teaches students how to *model, verify, implement and test a non-trivial cyber-physical system via three cutting-edge applications*: implantable cardiac devices, energy-efficient buildings, and advanced driver assistance systems. This course is noteworthy for the way it synthesizes research results from the last 10 years into material accessible to beginning Master's students.

## 3 Teaching Strategies

When preparing and teaching a class, I also use the following strategies.

- **'See one, do one, teach one'**: this principle originated in surgical training: watch a procedure, do one yourself, then teach it. I apply it in class. For example, when teaching temporal logics, I first formalize several requirements in LTL. Then I ask 2-3 students to formalize one requirement each in front of the class, then ask everyone to do the same on a piece of paper. They then go over each other's responses. This process is time-consuming, but it always brings up questions that the students would not even know they had otherwise.

- **Adjust the abstraction level to students' learning style**: In my class, I noticed that not everyone learns at the same level of abstraction. Some students require several examples, while others

prefer things to remain at the level of theorem and proof. So when I taught the model checking algorithm for example, I gave several judiciously-chosen examples *and counter-examples* to highlight the possibilities *not covered* by the algorithm, which caters to the students who require more ample illustration of the ideas. When discussing cardiac modeling choices, I stressed the differences in model expressivity and in what they allow you to prove, which caters to the students who are more comfortable with, or find more enjoyable, theoretical constructions.

- **Teach CS as part of the broader human intellectual endeavor**: When we think of how to program an ethics into autonomous systems, and algorithms more generally, we must ask: Whose ethics? Is it an inclusive ethics, which recognizes the diversity of experiences in U.S. and global society? In my Autonomous Racing course, we read three scholarly articles about autonomous weapons from a technical, philosophical, and judicial perspective. We then took a week to discuss and question the very possibility of encoding ethics in such systems. Several students told me they were happy to be debating these questions *in class, as part of their education.* One student felt free to share her experience working in a military research lab, and how she had to navigate the ethical dilemmas this posed her.

# 4 A curriculum in Responsible Autonomy

I have a Bachelor's in Computer Engineering, a minor in Mathematics, a Master's thesis in Image Processing, eight years of experience in Intel's Verification CAD group, and a PhD in Electrical Engineering. This diverse technical background allows me to teach core undergraduate and graduate courses in the areas of Signals and Systems, Controls, Digital Logic and Software Engineering.

In addition to the **Digital Twins** and **F1/10 Autonomous Racing** courses described above, I look forward to developing the following courses. Together, they form a basis for a curriculum in Responsible Autonomy. They align with the objectives of NSF programs like Research Experience for Undergraduates (REU) and Transforming Undergraduate Education in Science (TUES).

**1) Introduction to Cyber-Physical Systems**: In this lecture-driven course, the students will learn about modeling and verification techniques suitable for CPS. Topics will be selected from: reachability analysis, barrier certificates, statistical model checking, temporal logics, specification-guided testing, runtime verification, property-preserving abstractions and theorem proving. Equal emphasis is placed on proofs and gaining familiarity with the tools.

**2) Control theory and logic**: How do you design an autonomous Air Traffic Controller that can land hundreds of planes safely? As the systems we build are tasked with ever more complex tasks which they must perform autonomously, it is important to adopt a formal language for describing their objectives, and to develop control algorithms that are guaranteed to achieve said objectives. In this class, we look at temporal logic as a specification language, and at a rich variety of control algorithms that draw on methods from graph theory, linear algebra, finite automata, and of course, logic. We will emphasize the building of software tools that are maintainable and re-usable, and open-source all code to demystify the methods underlying autonomous systems.