Madison McClellan

Model-Based Design for Cyber-Physical Systems with Learning-Enabled Components

Mentor: Dr. Gabor Karsai

1. General Problem and Context

Cyber-physical systems (CPSs) are systems comprised of interacting physical and computational components. In recent years, there has been an increased use of CPSs in high risk and uncertain environments (Hartsell et al., 2019). For example, consider the operation of an autonomous vehicle in a crowded city. To operate safely, the vehicle must be able to react to many different events in an appropriate manner. However, it is not realistic nor feasible to explicitly program a response into the vehicle's software for each one of these events. To overcome this challenge, engineers have begun using data-driven techniques such as machine learning. This change led to the emergence of learning-enabled components (LECs) in CPSs. LECs use existing data to learn how to respond to different types of events. They can be used for a variety of different functions in CPSs, including perception and control.

The traditional principle of "separation of concerns" has caused differences to emerge across engineering disciplines in terms of modeling languages and software tools used (Ghezzi, Jazayeri, & Mandrioli, 1991). However, the design of CPSs with LECs is complex and requires a greater amount of coordination between disciplines. The main problem is that there is a current lack of tool support for the design of CPSs with LECs on a central and consolidated platform that supports all of the necessary functions: architectural modeling, data collection, LEC training, LEC evaluation and verification, modeling and analysis of safety cases, and system software deployment.

2. Description of the Specific Human/Cyber-Physical System Problem

The use of CPSs in safety-critical or mission-critical applications requires establishing that the system will operate correctly in response to a variety of different situations with a fairly high level of confidence. This practice of establishing an acceptable level of safety is called safety assurance, and it is one of the most important factors to consider in the design of CPSs with LECs. One of the most common techniques used for safety assurance is the development of safety cases, which are structured arguments that represent the safety of a particular system. Safety cases are typically structured around a top-level claim (i.e. "The system is safe because…"), and followed by evidence to support that claim. Goal Structuring Notation (GSN) is a type of graphical notation that is often used to write safety cases in a readable and standardized format. Ensuring the safety of autonomous systems is critical for gaining public trust. If the public lacks confidence that autonomous systems will perform at least as well as human-operated systems, it is unlikely that autonomous systems will ever become widespread.

Developing adequate safety cases, and thus establishing a high level of safety assurance, requires an understanding of how the particular system will interact with its environment. This includes both the interactions with the humans that are using the system, and if it operates in a public space, the humans in the surrounding area. Thus, the primary human/cyber-physical system problem in this project is developing safety cases that adequately address the ways in which a system will be interacting with the humans in its environment.

3. The Challenges of Reaching a Functional System

Because LECs rely on learning relationships from data, the primary challenge of reaching a functional CPS with LECs is acquiring a strong data set to use for training. For the example of autonomous vehicles, this training data would likely be acquired by attaching sensors to a human-driven car and driving that car in a variety of settings. The sensors would collect data about the car's surroundings and the human driver's reaction to different events. When translated into a training set, this would likely result in thousands, or even millions, of data points with many features (a measurable property of the event being observed).

However, gathering a large quantity of data is not enough; the data must also be high quality. This means that it should be representative of new cases that the system should be able to generalize to after learning. To achieve this, relevant features must be chosen carefully and high quality measurements must be taken to avoid errors and outliers.

4. The Technical Problem and the Research Setting

The research took place at the Institute for Software Integrated Systems at Vanderbilt University under Dr. Gabor Karsai. The project, called Assured Autonomy, has several teams working on different aspects relating to the engineering of autonomous systems. The team that I was involved in is developing the assurance-based learning-enabled CPS (ALC) toolchain. The ALC toolchain is a development environment that addresses the lack of tool support for the end-to-end design of CPSs with LECs. The toolchain supports all of the necessary tasks including architectural modeling, data collection, LEC training, LEC evaluation and verification, modeling and analysis of safety cases, and system software deployment. Additionally, the output from each of these tasks can be easily stored and accessed later for traceability and reproducibility.

My specific focus within the ALC toolchain was on the data collection method, and I worked directly with graduate student Charles Hartsell. We were focused on the use of the toolchain for designing an unmanned underwater vehicle (UUV) tasked with following a pipeline along the seafloor. This application provides a simple use case to develop the toolchain and demonstrate its numerous functions. The task involves following the pipe at an appropriate distance in varying conditions, including obstruction and water current. The control and perception of the UUV is controlled by LECs, and thus requires data collection and training. Prior to my work this summer, training set data was generated by explicitly specifying each desired variation of a given feature, then performing a cross product to return each unique combination of feature values. For example, if the desired features to vary were pipe bend angle and water current, values for each feature would be specified (e.g. [15 degree bend, 30 degree bend], [water current, no water current]) and a cross product would be performed (e.g. [15 degree bend, water current], [15 degree bend, no water current], [30 degree bend, water current], [30 degree bend, no water current]). However, the problem with this method is that more than just two features with many variations will need to be considered, resulting in thousands of unique experimental conditions to generate an adequate training set. Collecting this much real-world data is time-consuming, expensive, and not realistic.

My task for the summer was to improve the data collection method to allow for collection of large and high quality training sets in an efficient manner. To achieve this, I integrated Scenic into the toolchain to support generation of synthetic data. Scenic is a domainspecific scenario description language that allows the programmer to specify “scenes” of interest (Fremont et al., 2019). Scenes consist of physical objects and their properties, geometric relationships between those objects, and constraints and probability distributions on those objects. The Scenic syntax makes specification of scenes achievable with only a few lines of simple code. Furthermore, the ability to specify probability distributions over objects allows for the generation of thousands of iterations of a single experimental configuration using the same code. The second part of this task was to interface with the Gazebo simulator to provide functionality for generating data synthetically rather than in a real-world experiment. Once the Scenic scenes were created, they could be loaded into Gazebo and experiments could be run in the simulation environment. In the context of gathering training data, experiments consist of operating the UUV in the environment generated by Scenic while it collects information about its surroundings (e.g. distance to pipe, distance to seafloor, presence of obstacles) through sensors. Each experiment can be run with a different iteration of the Scenic scene, where each feature of the scene is varied slightly in each iteration. Thousands of experiments can be run to generate a large data set to train the LECs on. This new method for data collection addresses the primary challenge of reaching a functional system: the collection of large amounts of meaningful training data.

5. Future Research

The most pressing area for future research is the development of improved safety assurance methods. This would likely take form in improved techniques for quantitative evaluation of safety case GSN arguments. Improved confidence in the safety of CPSs with LECs is necessary for increasing the public's trust in such systems. Without public trust, there will never be widespread use of these systems and society may never be able to benefit from the technology.

A second area for future research is using Scenic with simulators other than Gazebo, such as Carla. Other applications of the toolchain may be more suitable for different simulators, so providing functionality for a wide variety of simulators would lead to greater flexibility.

References

Fremont et al. (2019). Scenic: Language-based scene generation. *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*.

Ghezzi, C., Jazayeri, M., & Mandrioli, D. (1991). *Fundamentals of software engineering.* Upper Saddle River, NJ: Prentice Hall.

Hartsell et al. (2019). Model-based design for CPS with learning-enabled components. *DESTION '19: Proceedings of the Workshop on Design Automation for CPS and IoT,* 1-9.