CS 395T: Program Synthesis

Spring 2021

Instructor: Swarat Chaudhuri

URL: http://www.cs.utexas.edu/~swarat
Email: swarat@cs.utexas.edu

When: Tuesdays and Thursdays, 3:30-5:00pm
Where: GDC 1.406
Course website: http://www.cs.utexas.edu/~swarat/Synthesis-Course

Program synthesis — the problem of automatically discovering a program that fits a given specification — is a classic problem in computer science. In the traditional formulation of the problem, the specification is a formal constraint, and the synthesizer's objective is to search for a program that satisfies this constraint. More recent research has extended this problem statement, requiring the synthesizer to *optimize* quantitative objective functions over programs. This line of work is closely related to machine learning. Indeed, recent work on *program learning* has explicitly formulated machine learning as a form of program synthesis, considered program representations that combine neural and symbolic elements, and given "neurosymbolic" learning algorithms that combine symbolic program synthesis and gradient-based learning.

This course will offer a broad study of the theory and practice of program synthesis and program learning. The course is divided into three modules:

- Functional Synthesis, the problem of synthesizing programs that satisfy a formal constraint over inputs and outputs.
- Reactive Synthesis, the problem of synthesizing reactive programs that interact with an adversarial environment in an ongoing manner. The problem was originally proposed by Church and has deep roots in logic, automata theory, and game theory.
- Program Learning and Neurosymbolic Program Synthesis, i.e., state-of-the-art mechanisms that bring together symbolic program synthesis and machine learning (in particular, deep learning).

Our discussion of each module will cover theory, implementation issues, and practical applications. As we go through these topics, we will introduce and utilize concepts from logic and automata theory; automated reasoning; programming languages; and optimization and

machine learning. A significant part of the credit will come from a course project. The only necessary prerequisites are undergrad-level expertise in algorithms and discrete mathematics and the ability to program.

Evaluation

The credit for the course is distributed as follows:

- 30%: Homework assignments
- 30%: Summaries of papers presented in class
- 40%: Course project

Sequence of Topics

- Overview: What is program synthesis?

- Unit 1: Functional program synthesis
    - Counterexample-guided synthesis
    - Learning strategies:
        - Top-down and bottom-up enumeration
        - Stochastic search
        - Symbolic search
    - Type-directed synthesis
    - Version space algebras
    - Synthesis through quantifier elimination
    - Invariant synthesis
    - Synthesis with abstract interpretation

- Unit 2: Reactive program synthesis
    - Reactive synthesis as an infinite game
    - Solving LTL and GR(1) games
    - Symbolic reactive synthesis
    - Applications in robotics and control

- Unit 3: Quantitative and statistical program synthesis
    - Quantitative reactive synthesis
    - Synthesis through continuous optimization
    - Bayesian program learning
    - Neural program induction
    - Synthesis of programs with neural modules

- Program synthesis with blended program representations and mirror descent
- Learning to synthesize programs

Format

Read before class: One paper per class
Reading quiz: Fill out on Canvas ahead of time
Student-led presentation (in some classes): 25-minute presentation + 5-minute Q&A
Lecture: 30-35 minutes.

Calendar

<To be filled out as the semester progresses>