CS595: Topics and Applications in Programming Languages

Fall 2021

Instructor:	Stefan Muller	**Time:**	TBA
Email:	smuller2@iit.edu	**Place:**	TBA

Course Description: Type systems provide many programming languages with a certain guarantee of runtime safety, usually that type errors will not occur at runtime. More advanced and exotic type systems can guarantee more interesting properties of programs, and are used in active research to guarantee information security, resource (e.g., time, energy, space) usage bounds, safety properties of concurrent programs and even to prove complex mathematical theorems. This course will teach the formal theory underlying the study of programming languages and, in particular, type systems, and the techniques that are used to prove properties of programming languages. We will also explore interesting applications of type systems in modern research, including some drawn from those above and others based on students' interest and research.

Students will complete a final project in which they design a type system or language model (or extension of an existing one) to solve an interesting problem related to their own research or other interests.

Prerequisites: There are no formal prerequisites beyond mathematical sophistication and experience programming in any statically typed programming language (e.g., C/C++, Rust, Java, Haskell, ML). In addition, experience with functional programming will be helpful but not required.

Advanced undergraduate students are invited to take this class, but require permission from the instructor and department to register.

Textbook: There is **no required textbook** for the class.

For students wanting a reference and additional depth into some of the material, the following books may be useful:
Robert Harper, *Practical Foundations for Programming Languages (Second Edition)*, Cambridge, 2016.
Benjamin C. Pierce, *Types and Programming Languages*, MIT, 2002.

Course Objectives: After taking this class, students should be able to:

- Understand the models, notations and techniques used by Programming Languages researchers to formalize languages and systems
- Formally prove properties about programming languages and programs
- Use the above techniques to design new type systems and models to formalize and prove properties about their favorite language features and systems
- Read and understand a research paper in the field of Programming Languages

Detailed Course Topics: (*subject to slight changes*)

- Foundations of Programming Language Theory:
  - Defining the structure and evaluation of a programming language
  - Proving the safety of type systems
- Fundamental Programming Language Models:
  - The untyped λ-calculus
  - Simply-typed λ-calculus
- Modeling features of real programming languages:
  - Polymorphism and data abstraction
  - Recursion and recursive types
  - Stored state
  - Subtyping
- Modeling and predicting the evaluation of programs:
  - Stack machines and hardware models
  - Tracking and predicting resource usage
- Parallelism and concurrency:
  - Fork-join parallelism
  - Shared-memory and/or message passing concurrency
  - Predicting parallel running time
- Possible additional topics depending on time and student interest:
  - Connections to propositional logic
  - Connections to algebra
  - Type systems for modeling ownership and sharing of resources
  - Proving equivalence of programs
  - Refining types with additional information about values
- Applications of type systems (will cover some subset of these):
  - Resource usage analysis
  - Information security
  - Data race freedom
  - Absence of priority inversions
  - Memory safety of garbage collectors
  - Others based on suggestions from students!