What is software foundations?

Same thing as "theory of programming languages": the mathematical study of the *meaning* of programs.

What is the goal of software foundations?

Finding ways to describe program behaviors that are both precise and abstract.

Why study software foundations?

- To prove specific properties of a particular program.
- To develop intuitions for *informal* reasoning about programs.
- To prove general facts about all the programs of a given language (*e.g.*, "Java is secure").
- To deeply understand language features for the betterment of future languages.

Name some approaches to studying program meaning.

- Denotational semantics and domain theory.
- Program logics (*e.g.*, Hoare logic, dependent type theories).
- Operational semantics.
- Process calculi.
- Type systems.

What is denotational semantics?

An approach to formalizing the meanings of programs by constructing mathematical objects (called *denotations*) which describe the meanings of expressions from the languages. Flow of control is abstracted away, in preference of an input-output view.

What is domain theory?

A branch of mathematics that studies special kinds of posets commonly called *domains*. It is used to specify denotational semantics.

What are program logics?

Approaches to formalizing the meanings of programs that focus on logical rules for reasoning about programs.

What is operational semantics?

An approach to formalizing the meanings of programs by means of abstract machines. It is low-level but flexible.

What are process calculi?

Approaches to formalizing the meanings of programs that focus on communication and synchronization behaviors of complex concurrent systems.

What are type systems?

Tractable syntactic methods for proving the absence of certain program behaviors by classifying phrases according to the kinds of values they compute. They only describe *approximations* of program behaviors, concentrating on the shapes of the values passed between parts of the program.

What is a proof?

An indisputable argument for the truth of some mathematical assertion.

How might a computer assist in writing proofs?

- Automatic theorem proving. Successful in only limited domains. Not feasible in the general case.
- Proof checkers. Simply *verify* proofs that are given in extremely low-level forms.
- Proof assistants. A hybrid of the two. It fills in the "easy parts" while the "hard parts" are done by a human.