Formal (?) Education in PL at CMU

Robert Harper
Carnegie Mellon University

Formal Education Workshop
Newton Institute
July 20, 2022

Core curriculum

- Mathematical Foundations for CS [No λ, no $\leq$, no μ!]
- Computer Systems and Architecture [Not relevant]
- Functional Programming [In SML]
- Imperative Programming [In C_0, ie Pascal with $\{\}$]
- Parallel Algorithms [In Parallel SML]

Foundations electives

- Principles of PL [This talk]
- Constructive Logic [Relevant]
- Formal methods for SE [Tangential]

Principles of PL

Structure

- Text: PFPL + Supplements.
- HW's: 6 assignments x 2 weeks. Theory + Implementation.
- Recitation: 1 hour per week.

Objectives

- Broad foundation: functional, imperative, parallel, concurrency, abstraction, modularity.
- Persistent concepts, not ephemeral trends.

Has a reputation of being demanding, but rewarding.

Formal Foundations

Abstract Binding Trees (ABT's)

- Hierarchy, binding + scope: eg, $\mathsf{let}(e_1; x.e_2)$, $\mathsf{dcl}(e_1; a.m_2)$.
- Substitution, α-equivalence / freshness.
- Variables are placeholders, symbols indices, eg $\mathsf{get}[a]$, $\mathsf{set}[a](e)$.

Statics (Typing Derivations)

- Basic judgments: eg, $e : \tau$.
- Hypothetical/general/parametric judgments: $\cdots x : \tau \cdots \vdash_{\ldots a \sim \tau \ldots} \mathsf{set}[a](x) \div \tau$.
- Structural properties of entailment.

Dynamics (Transition Systems)

- Transition systems: s state, $s \mapsto s'$, s intial, s final.
- Plotkin's SOS: $e \mapsto e'$ inductively defined by rules.

Typical Assignments

First, establish the tools and methods:

- Structural induction mod α.
- "The Wizard of TILT": locally nameless form, hash-consing, pattern matching.
- Inductive definitions, proof by rule induction.
- Preservation + Progress.

Second, explore language concepts:

- FPC: functional programming with recursive types.
- PyCF: a "dynamic language" [afer you-know-what].
- MA: Modernized Algol [after Reynolds].
- CA: Concurrent Algol [after CML].

Each assignment requires both proof and implementation!

Some Issues

Steep on-ramp.

- No prior exposure to rigorous proof. [Math foundations emphasizes cleverness.]
- Not clear where I am heading: must have faith!
- Aha moment at week 3, typically.

Abstract

- Concepts are considered largely in isolation.
- Assignments develop "real world" connections, but to a limited extent.

Preparation

- Relies on core curriculum, eg all students know ML and C.
- "Mathematical maturity": mindset, not skills.

Nevertheless, often cited as a "most important class" ten years out.

Whither Mechanization?

The entire course is done within a second-order logical framework.

- Syntactic: binding and scope of variables and symbols.
- Deductive: structural entailment and generality, parameterized by symbols/indices.

Easily mechanizable in Twelf, with some tricks.

- Binding, scope, entailment, generality, [indexing].
- Totality checker: $\forall\exists$ theorems, including type safety.

But that by far does not include all of the mathematics required in assignments!

- Failures of safety.
- Logics of programs, eg equations.

Nor does it encompass implementation!

Whither Mechanization?

Should Twelf (or any other prover) be used for assignments? NO!

- Requires even deeper preparation and groundwork.
- Privileges methods over content.
- Video game effect: get to level 7, regardless. [cf Constructive Logic course]
- Down-and-dirty: must understand details of theory and its relation to practice.

Should there be a course on mechanized metatheory? YES!

- Twelf is the tool of choice! [cf Mechanization of SML]
- Unfortunately we at CMU are too short-handed at the moment.

Should there be a course on verified compilation? YES!

- Crary's Higher-Order Typed Compilation class offered occasionally.
- I should write Principled Implementation of PL's as companion to Practical Foundations book.

Questions?

Thank you for the invitation!