Type Systems for Programming Languages

Eduardo Bonelli

January 24, 2019

About the course

- Introduction to notion of types in programming languages
- Highlights of the approach
  - Rigour
  - Precision
  - Minimality

Organization

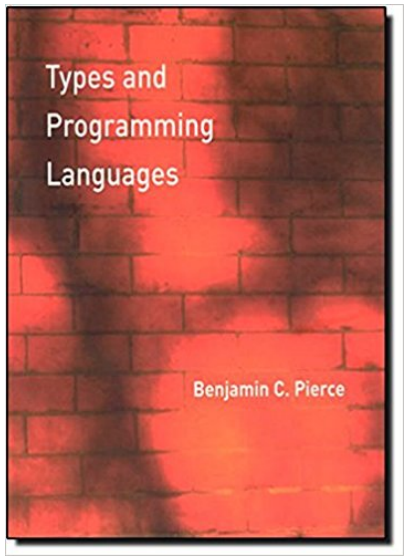
- TA: Ramana Nagasamudram
- Classes
  - Thursdays 6:15-8:45PM
- Pencil and paper exercises
  - Exercise booklets
- Some coding in OCaml
  - Bidirectional type checking
  - Type inference
- Resources: Canvas

Evaluation

- Midterm
- No final
- Student group presentations
  - Groups of at most two
  - 2 May 2018
  - 20 minute presentations
  - Possible topics: Rust, Idris, Agda, F#, Linear Types, Types in concurrency, Bi-directional typing, Gradual typing, GADTS/Phantom types, CIC and Coq, etc.
  - You can also propose your own (if relevant to this course; must check with me first)

Bibliography

- Benjamin C. Pierce. Types and Programming Languages, MIT Press, 2002.
  - We (mostly) follow this book



Bibliography

- Benjamin C. Pierce. Advanced Topics in Types and Programming Languages, MIT Press, 2004.
- John Mitchell. Foundations for Programming Languages, MIT Press, 1996.
- Henk Barendregt. The Lambda Calculus, its Syntax and Semantics. North-Holland, Amsterdam, second edition, 1984.
- Henk Barendregt, Will Dekkers and Richard Statman: Lambda Calculus with Types, Perspectives in Logic, Cambridge University Press, 2013.
- Robert Harper. Practical Foundations of Programming Languages, Second edition, Draft of Feb/2015 (`http://www.cs.cmu.edu/~rwh/plbook/2nded.pdf`)
- Rob Nederpelt and Herman Geuvers, Type Theory and Formal Proof: An Introduction, Cambridge University Press, 2014

Topics

1. Simply typed lambda calculus
2. Reference types
3. Polymorphism
   - let
   - parametric
4. Type inference
5. Subtyping
   - Properties
   - Use in object-oriented programming
6. Guarded subtyping and generics
7. Existential types
8. Recursive types
9. Dependent types

Motivation

- Type checking as static analysis
- Types in PL
  1. Standard
  2. Security
  3. Typed Assembly Language
  4. Concurrency
  5. Dependent types