Meeting 16: Type Checking

Quiz 3 Grades: 14 mean, 3.1 stdev, 14.25 median, 16.625 third quartile, 20 max
Quiz 2 Grades: 13.5 mean, 2.5 stdev, 14 median, 15.25 third quartile, 17 max

Lab 3 Grades: 85 mean, 9.1 stdev, 87 median, 89.75 third quartile (excluding no submissions)

Lab 3 Time Spent: 13.45 mean, 5 stdev, 5-20 range, 14 median
Lab 3 Difficulty: 4.85 mean, 1 stdev, 3-6 range, 5 median

Lab 3 Comments (my responses italicized):
- Auto-tester down painful.** I hear you!
- Difficult learning mystery. I hear you!
- Difficulty “determining if I had included everything that I needed to.” Formalizing semantics helps but interpreters are very hard to get bug-free.
- Figuring out how to correctly do the small-step interpreter and substitution were difficult.
- The implementation of the theory generally seems to be more difficult than the theory itself. I think it is the connection that is particularly difficult. The theory can seem deceivingly easy until you realize it is harder to implement than originally thought.
- I’m beginning to come around to Scala. Cool! For the skeptics, I guarantee it would not be any easier in Java with at least twice as much code for the same thing.
- We spent many hours just trying to comprehend what it was we were supposed to answer (i.e. interpreting the language specs) then trying to apply that comprehension to scala. Please, please ask as many questions as you can. This is what was hard about the lab and to some extent PL in general. There are so many cases in a language spec that a formal semantics is the most detailed way one can explain what to do. You cannot “power through” this kind of programming. The actual writing down code would only take 1-2 hours if all of the issues were internalized.
- Very difficult to start. Too much in the second week. Probably about half the time I spent on this lab didn't get me anywhere. Though I think about the scheduling carefully, I agree that got to small-step a little later than ideal. There are parts that can be started earlier to start the thinking, though things may just come together in week 2.
- Overall, this assignment really forced me to understand formal semantics. Implementation of the small step function became much easier once the semantics finally clicked. Yes!

Midterm: 67.2 mean, 15.9 stdev, 67.5 median, 79.5 third quartile, 91 max

Offer: Oral re-examination

Compress

\( \text{List}(1, 2, 3, 2) \Rightarrow \text{List}(1, 2, 3, 2) \)
Type Checking

\[
\text{call back}
\]

\[
\ell \cdot \text{foldRight}[^A,B]: B \Rightarrow [((A,B) \Rightarrow B) \Rightarrow B] \\
\uparrow \\
\subseteq^B + [A]
\]

Do Call

\[
\text{function } (x) e_1 (v_2) \Rightarrow e_1 [v_2/x]
\]

3(4) ≤ Yucky → want to throw out

we would like to check for an expression e, that e never results in dynamic type error

⇒ write this checken

⇒ static type checker
Type system

(1) language of types:
   a type classifies values
   (e.g., what operations can be performed on what "type"
of values)

\[ \exists ::= \text{number} \mid \text{string} \mid \text{bool} \mid \text{undefined} \]

(2) typing rules: rules govern whether a program respects types \( \Rightarrow \) checken
\( \Gamma \vdash e : \tau \)

\[\text{Gamma} \quad \text{type} \quad \text{environment} \quad \text{type} \quad \text{tau} \]

\(\text{const } x = \text{true};\)

\(x\)

\[E \vdash e \Downarrow v\]

\text{big-step interpreter}
\[ \Gamma \vdash e : \tau \]

\[ \Gamma \vdash x : \Gamma(x) \]

\[ \Gamma \vdash n : \text{number} \]

\[ \Gamma \vdash e_1 : \text{number} \quad \Gamma \vdash e_2 : \text{number} \]

\[ \Gamma \vdash e_1 - e_2 : \text{number} \]

\[ \Gamma \vdash b : \text{bool} \]

\[ \Gamma \vdash \text{str} : \text{string} \]

\[ \Gamma \vdash [x : \tau] e_1 : \tau \]

\[ \Gamma \vdash (\text{function} (x : \tau) e_1) : (x : \tau) \Rightarrow \tau \]
\[ \Gamma \vdash e_1 : \alpha \quad \Gamma \vdash e_2 : \alpha \]

\[ \frac{\Gamma \vdash e_1 : \alpha}{\Gamma \vdash e_1(e_2) : \alpha'} \]