CS152: Programming Languages

Lecture 25 — Course Victory Lap

Dan Grossman
Spring 2011

Victory Lap

- A victory lap is an extra trip around the track by the exhausted victors
  - A great way to soak in the highlights of the race
  - Very different from final-exam review (do that!)
  - Very different from course evaluations (do that too – for me and Harvard!)
- Review course goals and themes
  - Did we succeed according to lecture 1 and the syllabus?
- What didn't we have time for?
- What is any of this good for?

Here are three slides verbatim from lecture 1...

Programming-language concepts

Focus on *semantic* concepts:

What do programs mean (do/compute/produce/represent)?

How to define a language *precisely*?

English is a poor *metalanguage*

Aspects of meaning:

equivalence, termination, determinism, type, . . .

This course does *not* gives superficial exposure to N weird PLs

- More like CS121 than CS51, but not really like either
- But it will help you learn new languages via foundations

Is this Really about PL?

Building a rigorous and precise model is a hallmark of deep understanding.

The value of a model is in its:

- Fidelity
- Convenience for establishing (proving) properties
- Revealing alternatives and design decisions
- Ability to communicate ideas concisely

Why we mostly do it for programming languages:

- Elegant things we all use
- Remarkably complicated (need rigor)

I believe this "theory" makes you a better computer scientist

- Focus on the model-building, not just the PL features

Course goals

1. Learn intellectual tools for describing program behavior
2. Investigate concepts essential to most languages
   - mutation and iteration
   - scope and functions
   - types
   - objects
   - threads
3. Write programs to "connect theory with the code"
4. Sketch applicability to "real" languages
5. Provide background for current PL research
   (less important for most of you)

Some common themes

- Interpretation versus compilation
- Preservation and progress
- Determinism or lack thereof (particularly with inference rules)
- Encodings (to show expressive power)
- Functions (a great operational foundation)
- Types (a logical foundation for enforcing structured invariants)

Tons more to learn

Off the top of my head...

- Macros
- Process calculi for concurrency
- Languages for distributed computing
- "Real" denotational semantics
- Formal verification of full correctness
- Abstract interpretation / dataflow analysis
- Monads (beyond just the IO Monad)
- Type classes
- Module systems and foundations thereof
- and more

... but at leat you'll always remember Curry-Howard, right? :-)

Unsolicited testimonials (April 2011)

```
Hi Dan,

Long time, no see ;) I figured I'd drop you a line
about the latest project I've been working on for a
few months:  [snip] Finally, a chance to apply my
hard-won 505 knowledge to something out here in the
so-called "real world." I even had to pull out the
Pierce book at one point.
```

(A language for querying streams of real-time data on top of Hadoop)

Unsolicited testimonials (March 2011)

```
I'm writing up the proof that my
selector-intersection function is Correct and Total:

...

And lo and behold, my algorithm isn't total.  Two of
the cases fell through--once I rewrote the cases to
be in the form needed for the induction, it became
obvious that they were false.  Fortunately, it's
easily fixable...  but I'll have to go redo the
performance calculations now (there are more cases
than there were before...)
```

(An algorithm for deciding if there exist HTML trees for which two CSS selectors apply to the same nodes)

Unsolicited testimonials (July 2010)

```
I'm not sure if you remember me, but I took your
programming languages course a year or two ago...

Today I had to do some work with a minimal browser
shell around [snip], and found that I didn't have my
usual Javascript debugging tools.  So I tried to
write a small "immediate window" for Javascript so I
could conveniently execute commands.  I started off
knowing I'd probably use some eval(), but only a
little while in, I realized the naive approach wasn't
going to work because eval() does its evaluation in
the current context...

I eventually got it to work using some eval tricks
and some closure tricks.  I am 100% sure that if I
had not taken your mind-bending class, there's no way
I could have figured this out...
```

Unsolicited testimonials (May 2010)

Hi Dan,

I just came across continuations by accident while I
was looking at comparisons of lua with other
languages. I completely forgot we had gone over
those in your class, and am beating myself up for not
using them *ALL THE TIME* in my code - they are
awesome! Why are languages the coolest?!

Unsolicited testimonials (July 2006)

```
This class has changed the way I think about
programming - even if I don't get to use all of the
concepts we explored in OCaml (I work in C++ most of
the time), understanding more of the theory makes a
tremendous difference to how I go about solving a
problem.
```

My interpretation

What I think these testimonials are hinting at:

- Languages follow guiding principles you have now seen
- You can use these principles to make software better
- An education gives you “muscle memory” in surprising ways
- Computer science is fun (better than being a dog)

A personal thanks

- I have had a great time visiting
- And I appreciate you taking the risk of a “weird” class from “someone not from around here”
  - Even if this room was *sooooo* far away
- Come to UW for graduate school!
  - Or Seattle for a software career — Microsoft, Google, Amazon, Facebook, and many, many, more startups, etc.
- Stay in touch!