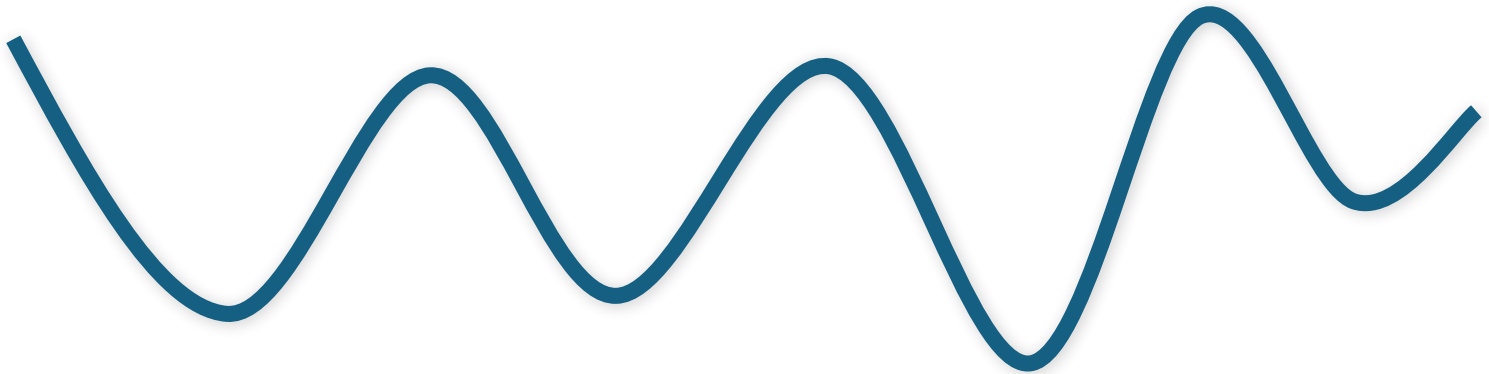
The Highs and Lows of a Language Researcher

Highs and Lows




OOPS, TIME FOR MY DAILY FIVE-MINUTE EXISTENTIAL CRISIS.
WHAT AM I DOING WITH MY LIFE??
WHAT'S THE POINT??
WHAT AM I GOING TO DO WHEN I GRADUATE?????
SIGH
BACK TO WORK!
JORGE CHAM © 2012
WWW.PHDCOMICS.COM

Thoughts I have…

- This will never impact the real world.
- This will never last.
- This isn't solving an important problem.
- No one will find this interesting.
- I'm bored with this.
- Oh sheesh, [SimonIXavierIDexterI…] has already solved this problem.
- I don't deserve to be here.

How to dig out?

- Value your own growth.
- Change directions or topics.
- Get your hands dirty.
- Explain your work to someone else.
- Have some patience.

Background

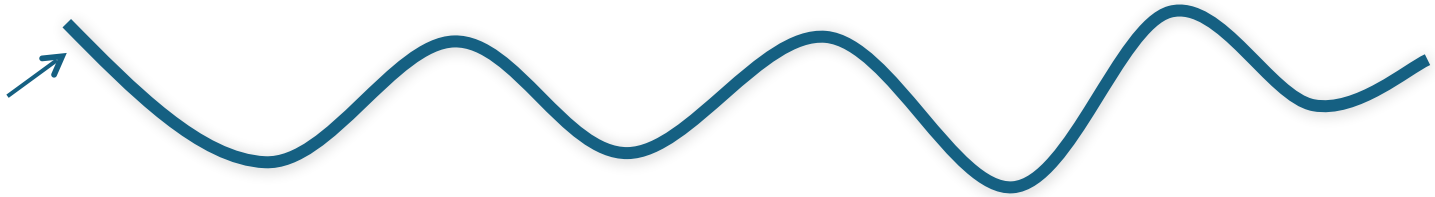
- Univ. of Richmond (B.S. in CS & Math, ‘89)
- Carnegie Mellon (PhD in CS, ‘95)
- Cornell (Asst. & Assoc. Prof., ’96-‘03 
- Harvard (Prof. & Assoc. Dean, ’04-’15)
- Cornell (Prof. & Dean, ’15-’19)
- Cornell Tech (Dean & VP, ’19-’28)

The Early Years (’90s)

- Context: Most programs written in C. Security was not an issue.
- My belief: All programs should be written in a *decent* language.

- A language was decent if it was:
- Higher-order
- Garbage collected
- Strongly, statically typed, ideally with good modules
- (i.e., SML)

First Projects



- Venari: SML/NJ + persistence + transactions
- Multi-core SML/NJ (Bell Labs)
- Hardware-based transactional memory (DEC Labs)
- Fox: an OS personality coded on top of Mach's micro-kernel

Frustrations using a decent language:

- SML/NJ was too slow.
- SML/NJ used too much memory.
- SML didn’t give control over data representations.

> *"Just as it's possible to write a TCP/IP protocol stack in some utterly inappropriate programming language like ML or Visual Basic, so, too, it's possible to implement TCP/IP over carrier pigeons, or paper tape, or daemons summoned from the vasty deep."* -- Charles Stross

The PhD Thesis

- Compile SML so that records, floats, etc. are unboxed.
- C & Pascal can be compiled this way.
- The culprit: What is the **`sizeof`**(α)?
- This really grew out of a flame war on netnews.
- An instance of getting my hands dirty.

Possible Fixes

- C++: give up on separate compilation
- Box everything: **`sizeof`**(α) = 1 word
- Leroy's boxing coercions (POPL '94).
  - Beeeeeautiful paper.
  - But no.
- Pass around types at run-time (POPL '95): TIL compiler
  - Seemed like a great idea at the time.

$[\ \text{int}\ ] = \text{int}$

$[\ \text{float}\ ] = \text{float}$

$[\ \tau_1 \rightarrow \tau_2\ ] = [\tau_1] \rightarrow [\tau_2]$

$[\ \alpha\ ] = \text{ptr}(\alpha)$

$[\ \forall\alpha.\tau\ ] = \forall\alpha.[\tau]$

$[i : \text{int}\ ] = i : [\text{int}]$

$[\ f : \text{float}\ ] = f : [\text{float}]$

$[\ \lambda x.e : \tau_1 \rightarrow \tau_2\ ] = \lambda x.[\ e\ ] : [\tau_1 \rightarrow \tau_2]$

$[\ e_1\ e_2 : \tau\ ] = [\ e_1\ ]\ [\ e_2\ ] : [\tau]$

$[\ \Lambda\alpha.e : \forall\alpha.\tau\ ] = \Lambda\alpha.[\ e\ ] : [\forall\alpha.\tau]$

$[\ e\ \tau_1 :\ \tau_2\{\tau_1/\alpha\}\ ] = ??? : [\tau_2\{\tau_1/\alpha\}]$

- $[(\Lambda\alpha.\lambda x{:}\alpha.x)\ \text{float}] : \text{ptr(float)} \rightarrow \text{ptr(float)}$ but it *should* have type $[(\alpha \rightarrow \alpha)\ \{\text{float}/\alpha\}] = \text{float} \rightarrow \text{float}$
- $[\tau_2\{\tau_1/\alpha\}] \neq [\tau_2]\{[\tau_1]/\alpha\}$ Translation does not respect substitution

But $[\tau_2\{\tau_1/\alpha\}]$ is *isomorphic* to $[\tau_2]\{[\tau_1]/\alpha\}$:

$S[\tau,\alpha] : [\tau]\{[\tau_1]/\alpha\} \to [\tau\{\tau_1/\alpha\}]$
$G[\tau,\alpha] : [\tau\{\tau_1/\alpha\}] \to [\tau]\{[\tau_1]/\alpha\}$

$S[\alpha,\alpha]$ (e : ptr $[\tau_1]$) := unbox e : $[\tau_1]$
$G[\alpha,\alpha]$ (e : $[\tau_1]$) := box e : ptr $[\tau_1]$

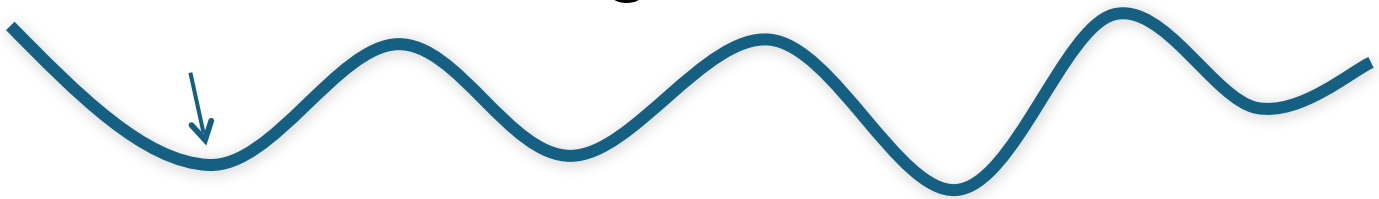
$S[\tau_2 \to \tau_3, \alpha]$ (e : $[\tau_2]\{[\tau_1]/\alpha\} \to [\tau_3]\{[\tau_1]/\alpha\}$) := $\lambda x{:}[\tau_2\{\tau_1/\alpha\}].S$ (e (G x))
$G[\tau_2 \to \tau_3, \alpha]$ (e : $[\tau_2\{\tau_1/\alpha\}] \to [\tau_3\{\tau_1/\alpha\}]$) := $\lambda x{:}[\tau_2]\ \{[\tau_1]/\alpha\}].G$ (e (S x))

$[e\,\tau_1 : \tau\{\tau_1/\alpha\}] = S\,[e]\,[\tau_1] : [\tau\{\tau_1/\alpha\}]$ when e : $\forall\alpha.\tau$

Back to TIL: A Challenge!

- We had to carry types through compilation in TIL.
- “Yeah, but you’ll never carry them all the way down to machine code.” – anonymous reviewer
  - That got me steamed!!!
- Hence: Typed Assembly Language (POPL ‘98)

Lots of Interesting Ideas Generated



- Type-preserving CPS, closure, object conversion.
- Types capturing calling conventions.
- Type states for the stack.
- Alias types.
- Modules at the assembly level.
- Typing run-time code generation.

Research

TAL didn’t solve all the problems…

- Still relied upon a garbage collector.
- Didn’t help with things like bounds-checking.
- And I started getting sick of assembly code.

Popcorn to Cyclone

- Simple, C-like language called Popcorn.
  - C.Hawblitzel coded up over holiday break.
- Software security started to take off.
  - Code Red, Nimda, …
- Could we turn Popcorn (now Cyclone) into something systems people would want to use?

Regions + SEXY

- The two big research questions we focused on:
- User-controlled memory management.
  - TT-style regions, unique pointers, etc.
  - See M.Hicks,D.Grossman, A.Ahmed, M.Fluet
- Static Extended Checking
  - Pre/post-conditions, aimed at getting rid of null-pointer and bounds checks.

Failure or Success?

- Two different goals:
  - Get C programmers to use it vs. explore research questions.
  - Failed miserably in the first goal.
  - Feel good that it's had some later influence on Rust. (Patience)
- Lots and lots of hacking (which was fun).
  - But some of this stuff never got written up.
  - And is now very obsolete.
- Also, C is just a horrible language.
  - After a while, I got tired of trying to "fix" it and was itching to get back to a decent language.

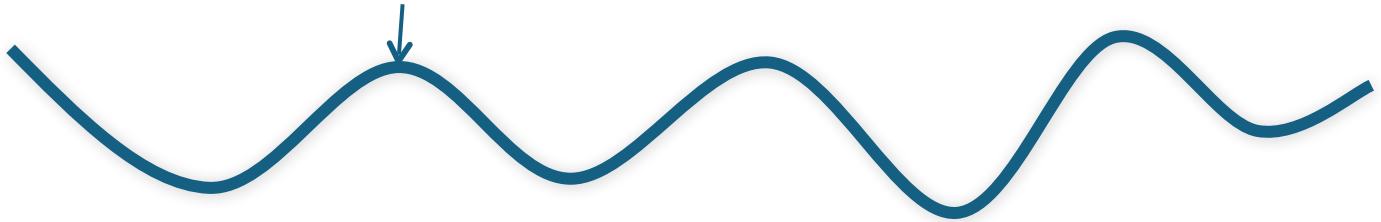
On Infrastructure

- The right infrastructure makes it easy to explore lots of questions.
- But building usable infrastructure is *much harder* than traditional research-level code.
- Plus, there's a danger that you'll spend all of your time polishing the hub caps.

Cyclone to Ynot

- Our static extended checking didn't work.
  - Could handle Pressburger arithmetic and that's about it.
  - How to eliminate the "key not found" exception on a hash-table lookup?
- We need a real program logic!
  - At the same time, separation logic, CompCert, and a host of other cool things going on.
- We'll just prototype something…

Ynot



- Originally intended to build SML + dependent types (like F*) for describing state changes.
- Prototyped in Coq and realized it already had everything we could hope to do.
  - Wasn't going to fall into the infrastructure trap again.
- So focus shifted to treating Coq as an imperative programming language with an integrated logic.
- Never really connected it back to Cyclone…

Freedom

- One of the great things about being an academic(y) researcher is that you get to go where the research takes you.

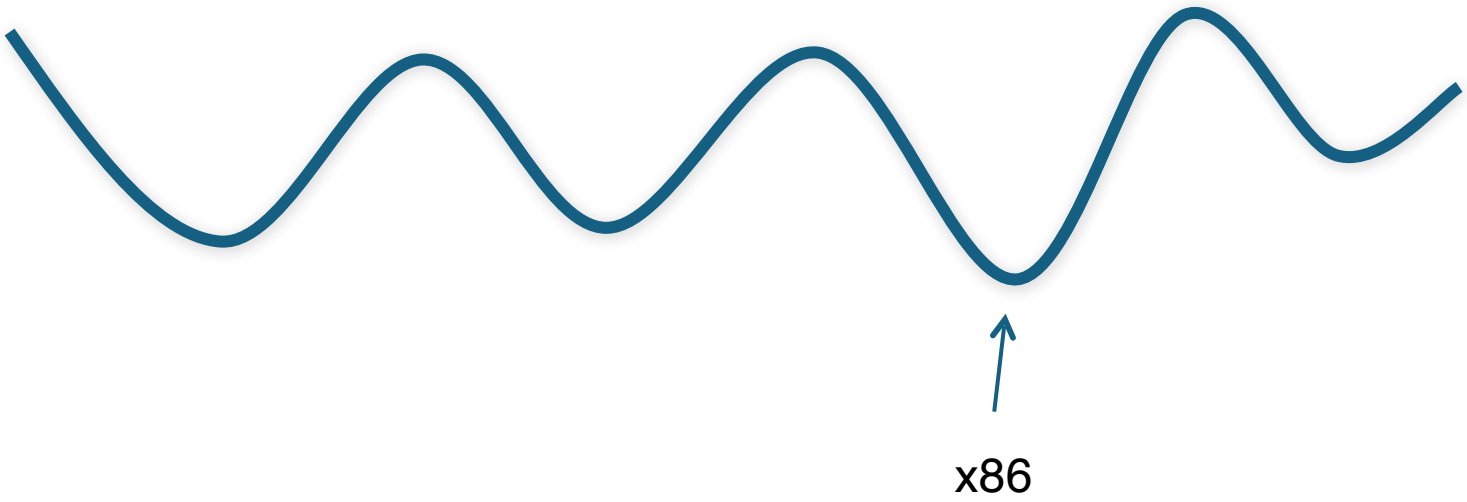
I got more interested in security…

- I'd been serving on Microsoft's Trustworthy Computing Advisory Board for many years.
  - Gave up on the "decent language" approach --- at least for the short run.
  - Too much like trying to convince NRA that gun control might be a good step.
- So… let's try to do something really, really basic.

Software Fault Isolation

- Insert checks into machine code to make sure the code doesn't read/write/jump outside of specified segments.
  - Basic integrity policy that we should be able to impose, right?
- Client wants to check that binary has been properly rewritten before executing it.
- (World's dumbest type system.)
- Wanted to do this on x86 code…

You are here…again.



RockSalt

- Built model of x86 binary execution in Coq.
- Built SFI checker in Coq.
- Proved SFI checker was correct.

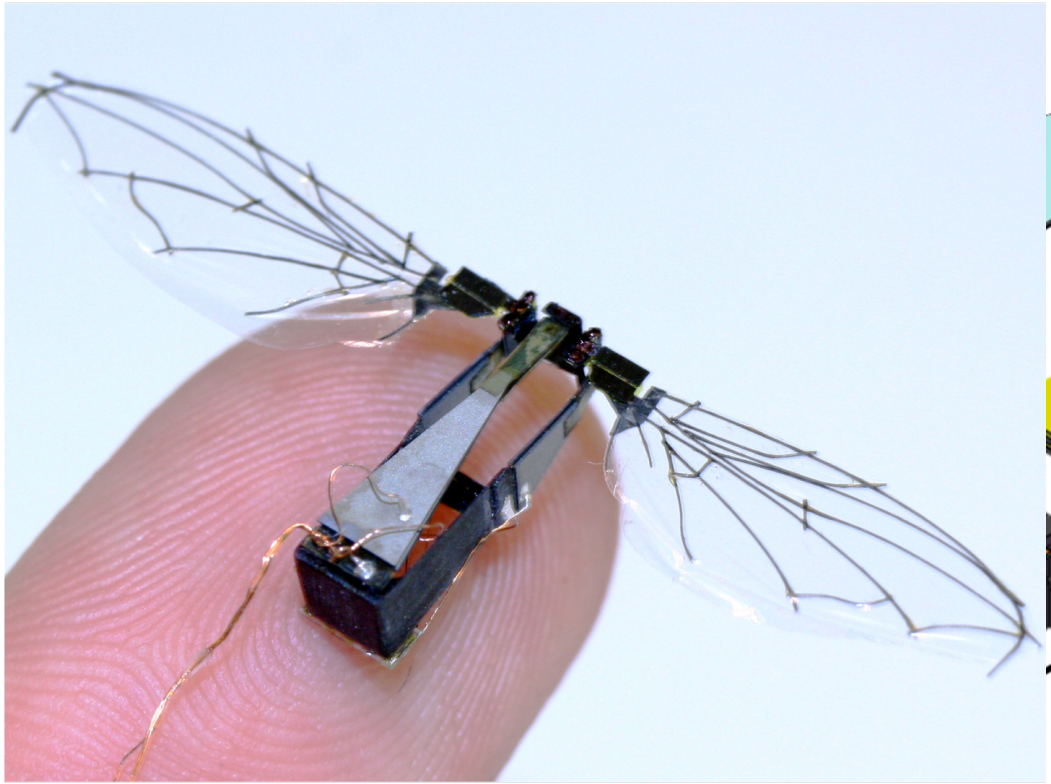
- 99% of effort went into building x86 model.
- But at least I was hacking in Coq!

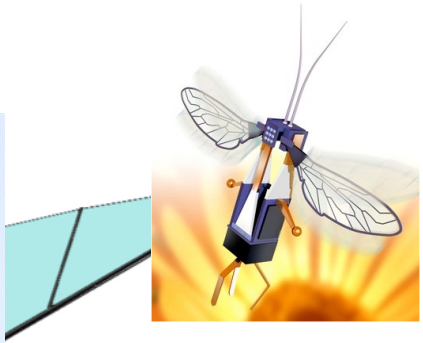
I got tired of x86.

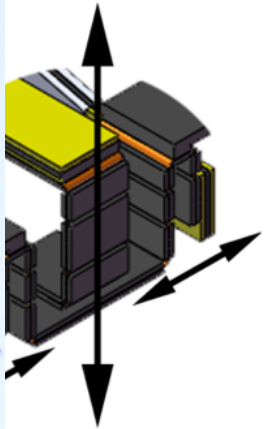
NSF had a new program.

My friends had heard about something called “colony collapse disorder.”

Aha! Robotics Bees!







Good Areas for the Future

- How to program a swarm of thousands of robots?
  - I still have no idea.
  - But I learned a *lot* working on that project.
- How to get safety guarantees out of cyber-physical systems, especially when they use “deep learning”?
- Or how to coordinate thousands of sensors so that scientists can get better spatial resolution for monitoring?
- Or how to program a network?
- Or how to program DNA hairpin assemblies?

A note on work/life

- We had our kids when I started my career.
- I'm *very* happy that I didn't put my life on hold for tenure.
- It's not like you'll have more energy when you're older.
- Your habits form early --- make time for your life now.

To try summarizing…

- Research is loads of fun and horribly frustrating.
- PL is one of the most flexible fields you can enter.