american doctoral

(with apologies to philip roth)

in which a plan for my next few years is sketched.
in which i seek advisors of academic unorthodoxy.
in which we dream of new and exciting computer science.

“**What dire offence from am'rous causes springs,**
What mighty contests rise from trivial things.”
--Alexander Pope, *The Rape of the Lock*

"Young man, let me explain to you about something new and exciting in physics."
--Niels Bohr, whispering in the ear of Herbert Anderson, Pupin Hall, Columbia, 1939-01-05

Two-sentence dissertation

- “The chosen problems and architectures are subjected to extensive analysis and modeling, using group theory, combinatorics, number theory, operator theory and the like to prove many new theorems regarding instruction sets, microarchitectures, and memories.”
- **???**
- “We have constructed (“solved for” is asking too much) the supremum – absolutely optimal – set of 1 or more codes for these problems + architectures. In doing so, we've developed novel links between algorithmic information theory, compiler design, architecture, and high performance computing.
- **Profit!**

Say what?

- Any old naïve solution to the problem gives us a lower bound on the optimal code.
- Stupidly (exhaustively) solve for an optimal sort via:
  - Running qsort(),
  - Counting Q dynamic instructions at CPI *c*,
  - Simulating all strings over the instruction set having Q_1 static instructions at CPI c_1 s.t. Qc >= $Q_1C_{1,}$
  - (we cheat the *Entscheidungsproblem* its *zonbi* by always halting after Qc quanta~=cycles), and
  - Sorting correct configurations by time-to-halt.
- I will address well-defined but complex integer problems on streams and large problem sets. I'd today lean towards “crypto/search primitives on a SSSE3, a T4, and a GTX680.”

So hey chief what's “optimal”

- Whatever we want it to be – whatever yields the best theorems.
- A range of things. Indeed, the types of optimality are known research (see Rice's and Blum's theorems)
- Something useful: max throughput with min power draw and maximum amount of useful work that can be done by other threads...
- The Dr. Nate Clark Metric: minimize branch mispredictions in code written by Indonesian midgets

Long textual insert! Huzzah!

...from an email, obviously...

we can take a problem, solve it in some naive way, and thus get an upper bound on the time necessary to optimally solve the problem (and remember, we get to pick architecture, problem, and definition of "optimal", subject to the right-handed indonesian midget rule) on that architecture, right? So this means we only need test programs through that time -- we don't need deal with the halting problem. theoretically we could do exhaustive search of the strings over the { maximum throughput of dynamic instruction bits * upper bound }, and given perfect simulation, sort all the strings which left us in a correct configuration by time taken to halt. this is all algorithmic information theory in the tradition of kolmolgorov and chaitin plus algebra.

so that's great, but the GNU SuperOptimizer was doing this kind of thing in 1992. it's just intractable to search a large space, right? and thus optimal scheduling and the compilation problem of which the former is only a subset remain creepily non-polynomial time. motwani or someone proved this in 1995 iirc -- you surely know better than i do. i don't think it's known to be in EXP verification vs NP-hard but it's definitely EXP for construction. Feel free to correct me here.

so if it is NP-hard rather than EXP throughout we can verify a code as optimal, given a perfect machine description. seems dubious. i bet EXP. let's proceed under that assumption unless you know it to be wrong. if so, techniques for proving codes optimal are totally underdeveloped.

this is useful to anyone who needs to know "when to stop" when optimizing. multiple code sequences might be equally optimal for maximizing single-problem throughput, but some might e.g. require more power, or leave less resources for other threads in a SMT/HT situation. we explore and develop, explore and develop, and publish our findings.

for one thing, it'll be a huge ball of shit to *PROVE* that *NO OTHER SEQUENCE OF BITS EXISTS* that results in the same configuration in less time. you have to show that there's no way BCD addition and 80 bit floating point on the x87 stack followed by a page fault, LOCK prefix, NMI and EFLAGS set to the constant 0x8937893 happen to yield an ultra-fast AES round.

so immediately, you develop design guidelines for anyone who wants to make their architecture amenable to this kind of analysis. you relate them to the virtualization criteria of popek and goldberg, maybe, if there's a relation worth pointing out.

what i'm hoping for is that i'll find things like (pulled down from the air) "it's useless to provide more than a word per second of transcendental throughput per N words of integer unit". Surely such theorems exist, waiting to be discovered and proved.

FUNDAMENTALLY, i want to bridge algorithmic information theory and architecture, and apply the result to compilers.

So build a compiler from it!

- Such an exhaustive search is of course intractable for any real problem or problem set.
- That doesn't stop, say, the GNU SuperOptimizer from using exactly this method.
- We are not compilers, but rather the masters of compilers! We seek similar results, but with flair, with panache! We trade elegance for generality; time for size; program depth for program shape.

Let's seed ten thousand compilers

- This is an intensely technical, detail-driven research agenda, yet it seems as close to pure science as one can come in arch/HPC.

- I expect to find exciting, wonderful new theorems in the landscape of instruction set architectures, microarchitectures, and program characterization. In order to pull this off, they'll have to be novel and unexpected, which effectively means they'll need be interdisciplinary.

- In my dreams, compiler PhD students and interns with superior academic credentials will mine this dissertation for ideas for two decades. And microarchitects. And HPC junkies. And anyone who reads Knuth, SICP, or Warren's *Hacker's Delight.*

- It is truly the journey and absolutely not the destination: the dissertation's nominal goal will be obsolete before it's even achieved. The methods, I think, might with luck stand.

Let's get down to brass tacks

- The author is 32 (two to the fighting fifth) years old -- about a decade older and ~50 lbs fatter than a typical newly-minted, fresh-faced GRA. He is a Senior Member of the ACM, has brought numerous successful products to market, and vibes hardcore freak on all axes.

- This will not be the typical PhD adventure.

What I can provide **you**

- One hell of a dissertation, if I pull it off.

“It appears to me that it is a bit of a gamble as to whether Oppenheimer will ever make any real contributions of an important character, but if he does make good at all, I believe that he will be a very unusual success...”

--Percy Bridgman to Ernest Rutherford regarding J. Robert Oppenheimer

- A contrarian, unorthodox, and fearless voice among your students.

- Deep architectural / HPC / compilers / PLA knowledge, via self-study, classes, and work. Insight on your mailing lists; mentoring in your lab (by “mentoring” I mainly mean telling people to read the damn `man` pages, but someone's go to do so).

- 15 years professional programming in leadership/research capacities

- Breadth of knowledge at the edge of testing's ability to discriminate :D
  - 99th percentile, CS GRE Subject Exam
  - Perfect GRE General Exam (and I was hungover!)

What I cannot provide **you**

- Much of an interest in publishing in this or that hot conference, the bread and butter of competitive academia. Mine is a more genteel effort -- leave the ISCA credits for the young bucks who need them.

- More than a max of 15 or 20 hours per week in a lab getting asked by masters students how `printf()` works or “what's a ping nicholas” or being told “i don't have time to make user accounts thus I read USENET as root” &c.

What you need provide **me**

- Access to machines, especially novel and high-performence architectures.

  Alert me to when I'm moving by slogging instead of thinking. I get easily sucked into the former.

- Convince the CoC that this is a valid way to do a PhD, since they want all that three top-tier papers crap that I couldn't care less about.

- References to things I'd otherwise have missed, when you can, and introductions to people who can provide further such references.

- I'm happy to take any excess grant money you have, but I can also fund myself, so long as you don't mind me ducking out for a month here and there to chase down consulting work.

Case Study: David Dagon

- Longtime student of Professor Wenke Lee.
- Go talk to Professor Lee.
- Go talk to David, if he can be cornered.

- He pretty much pioneered this model in the CoC, so far as I know (he's also just an absolutely stand-up guy, well worth knowing.)

- Upshot: GT, CoC, Professor Lee, and Dr. Dagon all mutually benefited from their unorthodox arrangement.

Advisory Committee, as dreamed

(with apologies to Mirah)

THIS LIST DOES NOT INDICATE AGREED AFFILIATION, SUPPORT IN WORD OR DEED, OR THAT THOSE LISTED HAVE ANY IDEA WHAT THIS IS, OR WHO I AM.

- Co-advisors:
  - Prof. Tom Conte and Prof. Richard Vuduc
- Advisor from the College:
  - Prof. David Bader or Prof. Michael Wolfe
- Extratopical advisor:
  - Prof. Dana Randall, Prof. Vijay Vazirani, or Prof. Richard Lipton

Transinstitutional advisor:

- Prof. Scott Aaronson (MIT)

let's drop a gem on 'em

thank you for your time