Incidental (re)discoveries about the basics of science

kena

30 jan. 2013

Universiteit van Amsterdam

Purpose of this talk

- I feel I am a bad scientist, and I want to tell you why
  - And maybe share some basics of epistemology in the process...

Just a reminder: Falsifiability

Falsifiable knowledge

- Reminder: a theory is **falsifiable** if it is possible to empirically demonstrate it is false
  - Example:
    "All swans are white" is falsifiable, it suffices to observe one non-white swan

- A falsifiable theory becomes stronger as we try to prove it wrong, and fail while trying

Example
Unfalsifiable theories

- Simple:

  "There are no black swans"

- More tricky:

  "White swans do exist"

Weak research questions

Weak questions

- Example:
  "What micro-architecture is better than OoOE?"
- Problems:
  - Open to dispute as to what "better" means
  - Open-ended search space: OK if I do find such an architecture, but when do I stop if I don't find one?

Weak questions

- Better:
  "What micro-architectures provide better performance/watt at fixed silicon budget than OoOE?"
- Problems:
  - Solved the metric problem
  - But the search space is still open

The problem with open-ended search spaces

- We can't just say
  "Here is my search space; if I can talk about it with words, there must be at least one solution" and expect peers to believe/like it
  - We actually have to **show/construct** the solution
- If we don't find a solution, what then?

The problem with open-ended search spaces

- *"there exists a solution in my open-ended search space but I don't know it yet"*
  is a **non-falsifiable theory**
- Conversely *"is there a solution in my open-ended search space?"*
  is not answerable scientifically
  - Unless the solution is known in advance
  - or one take formal precautions to enable negative proofs

Weak questions

- *"Can hardware multithreading be made cheaper and at least as effective than OoOE at tolerating fine-grained latencies on single cores, including memory loads and FPUs?"*
  (from previous talk)

WEAK SCIENCE

Your turn now

- What is your most recent research question?
- What is the most important question in your field?
- **What avenue(s) do you leave to someone else to prove you are wrong?**

The mirage of generality

Language matters

- Consider: *"A cow is an animal"*

- The word "is" introduces a vague notion of equivalence - but which one?
  - Hint: *"all cows are animals, but there are other animals that are not cows"*

Language matters

- What about:
  "a 21064 chip is an Alpha chip"
- *"all 21064 chips are Alpha chips"* is true, but there were no other Alpha chips that were not 21064's at the time
- The concept "Alpha chip" is derived from the concrete existence of at least one 21064 chip

Early 20th century

- 20th century: Russel, Hilbert
  "Without pre-agreed definitions for words, we can't talk about truth in mathematics and logic"

- Gödel, Boole:
  "If we can talk about truth without being vague, we can encode it too with numbers"

- Church, Turing:
  "If we can talk about symbolic computations without being vague, then we can also build programmable theoretical machines to do that for us"

Late 20th century

- Turing, and all computer architects afterwards:
  "Here is a real-world machine I built, and it can run these specific programs I wrote and it does stuff (to/with the real world)"
  "Oh, by the way, it fails sometimes and we don't know why"
  "oh, by the way, we haven't tried it with other programs yet"
- Computing is mostly the world of real-world "things" that escape pure theoretical reasoning

Dijkstra's cry (since the 1970's)

- Logicians, and later pure functional language designers; spearhead Dijkstra:
  "We want to talk to your machine in a fully general language without bothering with the implementation specifics"
- Engineers:
  "Nobody cares about generality. We make systems that do the particular jobs as cheap and fast as possible."
- Check out: http://www.dijkstrascry.com/

A surprising find

- J. Voeten, TU Eindhoven, 2001, "On the fundamental limitations of transformational design" (ACM Trans. DAES)
- Short version: "Engineers win."
- Long version:
  It is <u>not possible</u> to design a fully general language that programmers can use to achieve particular tasks <u>in the real world</u> without using non-formalizable knowledge about the real world machine.

What this means to me

- Can't assume/use *"it is good enough if it looks general"* when evaluating contributions
- **Generality is not intrinsically valuable for <u>designing</u> computing <u>systems</u>**
- Generality can be used to **describe** concrete implementations, not the other way around (certainly not to **specify** them!)

Why do I care?

- Consider: *"the Microgrid implements the SVP model"*
- Two problems:
  - SVP did not exist before the Microgrid (remember: Alpha vs. 21064)
  - Generality describes the concrete specific cases, not the other way around
- Other phrasing:
  "SVP is a general model that describes the behavior of Microgrids, for example as implemented by MGSim"

Why do you care?

- Consider: *"in this paper, we present method X; we then demonstrate our method is effective on example Y"*
  - Did you not develop method X after you found a solution to Y, by any chance?
    - **If so, your paper is <u>weak science</u> too**

- Consider instead: *"In this paper, we show how we solved problem Y in a specific way; we then propose to generalize our solution into a general method X"*
  - Harder, better, faster, stronger!

Description
vs.
Specification

Why do I care?

- Statement from our logician colleages:
  "Place(Component) = CoreNum"

- Is this a **description** or a **specification**?

Description vs. specification

- A **description** does not say how to actually place components at run-time
  - **It's useless to guide design**

- A **specification** must first tell me how to compute the *Place* function
  - **Can guide design, assuming proper engineering procedure**
- Example: saying that *"the Place function is a statistical distribution"* (like in ADVANCE) doesn't make it computable; it's merely a description, not a specification
  - **It cannot serve to build a system**

Suprise, surprise!
.... not.

- **Models** are descriptions, not specifications
  - Models are necessary to understand existing systems
  - But they are not sufficient to design and implement (new) systems
- Radically different approach between **people who describe** (and analyze, and predict) and **people who specify** (and program, and build)

The mirage of equational systems

- Languages and notations exist that allow us to express equations between the observed and the desired
  - e.g. functional languages, VHDL
- Pure equational statements **can be either descriptions or specifications, depending on P.O.V**
- **BUT:** we cannot **derive knowledge** from them before choosing a position first.

What is computer architecture really?

(As a human activity)

Why should you care?

- Consider: *"Based on my model of application behavior, I explored a design space using simulations and found this design point with some interesting properties"*
  - NB: Simulations are equivalent to automatic model derivation
  - Oops? **models can't specify designs.**
  - What's missing: empirical validation! By constructing and testing the real-world systems, of course.

We are doomed! ... or not?

- Most architecture research groups **can't afford to build** artifacts for every design point proposed
  - What then of the scientific value of our statements about design based on models?

We are doomed!
... or not?

- *Empirical observation: our peers let us publish.*
- My first 3 hypotheses:
  1. They just like us and don't care we do weak science
  2. They are all weak too, and our entire "scientific field" is a massive fraud
  3. Every result was ultimately accepted by an engineer who actually tried the idea out

We are doomed!
… or not?

- Hypothesis 4, my favorite:

  4. What is valuable in our work (and what we are expected to do by our peers) is not the science, but instead something else.

- What then?
  - My take: vision, inspiration, guidance, engineering support, "innovation"
- I don't mind <u>not being a strong scientist most of the time</u>, do you?

Wrapping up

What I took away from these thoughts

- Try to sound more scientific by **caring for falsifiability** and showing how others could prove you wrong if you are
- Don't confuse **description** and **specification**
- Don't abuse the word "**model**"
- **Build things that work** and show them around, this is **what your peers secretly want** behind the façade of science.

Thank you.