Soft Concepts vs Hard Facts: software models in the social sciences

1 Program correctness

- throughout the years, programming language have become increasingly more "high-level"
- a large part of this evolution involved identifying repeating patterns in the lower levels, that were named, abstracted out, and made into building blocks of the higher levels
  - example:
    * subroutines with goto
    * procedure calls
    * objects
    * arrays
    * lists
    * trees
    * graphs
- example: traversals of data structures
  - traverse a list, a tree, a graph
- even larger patterns: systems, frameworks, stacks
- the fact that repetition is reduced makes it easier to understand and verify programs
  - but we still have about the same number of bugs
    * law of conservation of programming errors: program complexity increases faster than our ability to control complexity
- the high-level abstractions hide massive amounts of detail, ideally in a controlled fashion
  - tower of abstractions; correctness must be ensured at **each** level
- the abstractions had in the beginning an "ad-hoc" character
  - example: object orientation
    * traversal: object-oriented solution with iterators unsatisfactory
- ultimate proof of correctness: **mathematical**
  - difficult, but necessary
- this forced us to introduce less "ad-hoc" and more mathematical abstractions
  - e.g., computing has imported a lot from category theory and type theory
  - category theory deals with very abstract forms of putting together simple things to make complex things (it is a part of algebra)
  - e.g., generic programming gives a much more satisfactory solution to traversals
- it is now a bit easier to ensure program correctness
  - but we need **formal specifications**
  - sometimes, the requirements do no seem amenable to formalisation

2 Vulnerability

- PIK
  - mission statement
  - slamming down the hard facts
  - very interdisciplinary research
    - requires **focal concepts**
- vulnerability
  - everybody understands it
  - separate department (currently "Climate Impacts and Vulnerabilities")
    - separate department "Sustainable Solutions"
  - foundation of climate change!
    - strange coincidence: these realisations of the importance of vulnerability came shortly after the World Bank announced a fund
  - definitions of vulnerability
    - many!
    - OED
    - tantalizingly similar (similar to traversals?)
      - also, there are programs we can look at
      - practice of vulnerabilitty

3 Formalising vulnerability

-vulnerability as measure of possible future harm

- example: small fish are vulnerable to predators
- a simplified representation:
  - small fish: "."
  - big fish: ":"
  - predator: "G"
  - initial situation: "..:.:. . . :..:.." s=10, b=4, p=0
    - potential futures: "..:.:. . . :..:.." —> "..:.:. . . :..:..G" —> "G::.:.:"
  - harm: decline in small fish population
  - vulnerability: some measure of possible future harm
- challenge: describe this in a very general way
  - category theory, haskell
  - contrast this algebraic approach with the Lotka-Volterra models!
- is it useful?
  - measure condition
    - violated in practice!
  - compatibility: average and expected value
  - vulnerability to cause vs vulnerability to harm!
    - small fish are vulnerable to predators vs small fish are vulnerable to being eaten (by predators)!

4 Conclusions

- "normal models": Amy Luers
  - rely essentially on **analogy**
- formalisation: **ground rules for analogy, distilled from common practice**
- opportunity for using program correctness tools for high-level formalisation of "soft concepts"
  - Axelrod: simulation is a third way of doing science
  - examples