

Computer Science as a Liberal Art
the Convergence of Technology and Reason

Natural or Artificial Science?

"Computer Science is no more about computers than astronomy is about telescopes."
 -- E. W. Dijkstra

- Studying a particular computing technology makes it an artificial science.
- Building a computer to solve a specific task is engineering.
- Studying computation as a natural phenomenon (mechanism or mentalism) is a natural science.

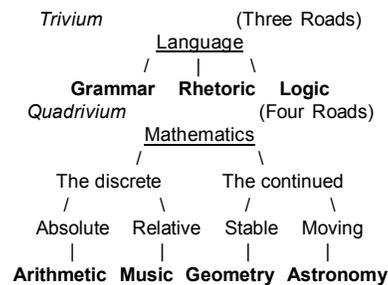
Reasoning & Reckoning

"REASON ... is nothing but Reckoning"
 -- Thomas Hobbes, Leviathan, 1651

Mechanization of Thought:

- Can human thinking be mechanized? (routine, automatic calculations)
Is all reckoning reasonable?
- Rationalization of Mechanical Processes:
- Can machine calculation be explained by logical reasoning?

The Liberal Arts



Historical Methods of Computing

- fingers; abacus
- *geometry*: rules for solving measurement problems
- *algebra*: rules for solving arithmetical problems
- adding machines ($\pm \times \div$)
- modern digital computers

We are in the *Dark Ages* of Computing:

- Even recent technology seems arcane!
- Are there limits to the potential of computers?

Information

- The science of transmitting data:
text: 100...10 ; or pictures: bits in 2-D

data transmits paper is:
 • **Storage:** *now to then* *permanent* in time
 • **Communication:** *here to there* *portable* in space
main issues
 • **Efficiency:** data compression
 • **Accuracy:** coding theory

Explicit Definability

Graph: set of nodes related by edges ($a \rightarrow b$)

Graph Simplicity (local property): $\bullet \longrightarrow \bullet$

- $(x \rightarrow y) \Rightarrow (y \rightarrow x)$
- $\neg (x \leftrightarrow x)$

Ordering (global property): $\bullet < \bullet < \bullet < \bullet$

- $(x \neq y) \Rightarrow [(x \rightarrow y) \Leftrightarrow \neg (y \rightarrow x)]$
- $(x \rightarrow y) \ \& \ (y \rightarrow z) \Rightarrow (x \rightarrow z)$

1/10/2010

PHAN talk

19

Recursive Definability

Reachability in a simple graph ($s \sim t$):

- Is there a *path* of edges from s to t ?

Not explicit, but still tractable. Idea (linear-time):

- Mark all nodes reachable from s , then see if t got marked

Define path in terms of edge and *itself*:

- $(x - y) \Rightarrow (x \sim y)$
- $(x \sim y \sim z) \Rightarrow (x \sim z)$

1/10/2010

PHAN talk

20

Implicit Definability

A definition in which one **conjectures** *any* (some / every) result which **substantiates** *the* (unique) answer.

GCD(a, b): guess any integers x and y which make a positive $d = a*x + b*y$ that divides both a and b .

Reach(s, t): guess any bunch of nodes which form *either* a finite chain between s and t ; *or* a closed set including s and excluding t .

Factor(n): guess any collection (will be unique) of prime factors whose product is n .

1/10/2010

PHAN talk

21

Logic as a Language

The grammatical **perspicacity** of a problem specification (*syntactic*) corresponds closely with (*captures*) the computational **efficiency** of an algorithmic solution (*semantic*).

This fits with Quine's epistemology that we only know the world (of computing) through (the formal) language (of logic).

Frontier of Knowledge: Is there a (mathematical) logic which captures the notion of (physical) computation that is both accurate and efficient?

1/10/2010

PHAN talk

22

Speculative Remarks

'**natural**' limitations to computing:

- *mass*: bounded information per node
- *energy*: self-powered automatus

universal data assumption: graphs of all shapes and sizes can be in memory

- data structure I/O: converting arbitrary graph to string
- many hard problems run in linear-time on tree-like graphs

Latest Research: on trees, a method to transform problem specifications into accurate & efficient programs:

logic = implicit definitions; *device* = self-powered automata

1/10/2010

PHAN talk

23

Conclusion

The "**Holy Grail**" of Computer Science would be some way to validate *Feynman's Method*, turning algorithmic design from an art into a *science*.

- to transform *feasible* specifications into correct and efficient algorithms (*automatic programming*)

How closely will abstract thinking (human reasoning) and machine computation (technological devices) converge?

1/10/2010

PHAN talk

24