Maggie Johnson
CS109

Background and History

Key Topics

* Computer Science as The Mechanization of Abstraction
* Areas of Computer Science
* A Brief History of Computers and Computer Science

• Computer Science: The Mechanization of Abstraction

Abstraction: the replacement of a complex, detailed real-world situation with a simple, understandable model within which we can solve a particular problem.

Computer Science is a science of abstraction: creating the right model for a problem, and devising appropriate mechanizable techniques to solve it. We create abstractions of real-world problems that can be represented and manipulated in a computer.

>>> Example: You are hired to implement a computer system for a travel agency. Your system must determine the best route for a traveler to get from location A to location B ("best" means shortest distance traveled). How would you do it?

1) Specify the locations that the system can handle.
2) Create a database of distances between all the locations. Such a database must have a structure that allows for easy updates and quick searches.
3) Create algorithms that will operate on the database.
4) Create an algorithm that finds the shortest route between two locations.
5) Implement a program that takes as input location A and B and outputs the shortest route between the two.

There are several examples of abstraction in this example:

- representing the locations and distances as names and numbers
- placing this data in some kind of abstract data structure in a computer
- creating abstract operations on this data structure such as "Find" and "Find Shortest Distance"

Thus, computer science is the **mechanization of abstraction**. We created an abstract model for the data and the application, both of which can be represented and implemented on a computer.

• The Areas of Computer Science

The two basic components of computer science are **hardware** (the computer itself) and **software** (the abstractions we create to solve problems on a computer). Each of these components breaks down into several areas of study in computer science.

Hardware		*Software*
Finite State Machines	Machine and Assembly Language	Problem Solving and Algorithms
Digital Logic	Operating System Theory	Formal Languages
Digital Circuits	Network Theory	Programming Languages
Computer Architecture	Graphics	Compiler Theory
Computer Engineering		Data Structures
		Software Engineering
		File & Database Theory
		Complexity Theory

Finite State Machines: theoretical models of how a computer works
Digital Logic: application of Boolean Algebra to digital logic design
Digital Circuits: combination of logic gates into circuits
Computer Architecture: overall structure and function of hardware systems
Computer Engineering: engineering aspects of building hardware

Machine and Assembly Language: languages the computer itself understands
Operating System Theory: study of the system that sits between hardware and software applications which makes the hardware accessible to applications.
Data Communication: hardware and software that allow for data to be transported
Network Theory: hardware and software that allows for connection of 2+ computers so they can share data and resources
Graphics: hardware and software for creating graphic images

Problem Solving and Algorithms: methodologies for solving abstract problems
Formal Languages: theoretical models of programming language construction
Programming Languages: syntax and semantics of programming languages
Compiler Theory: translation of programming languages to assembly/machine language
Data Structures: models for the storage and manipulation of data
Software Engineering: study of the process of creating software
File and Database Theory: special ways of organizing data for quick, reliable access
Complexity Theory: study of the efficiency of running programs

And some other miscellaneous topics:

Computability: what computers can and cannot do
Artificial Intelligence: simulation of human reasoning, vision, movement, speech, etc.
Human Computer Interaction: interface designs
Ethics and Social Responsibility: privacy, reliability and risk, responsibility of professionals, etc.
Medical Informatics: use of computers in health care
Computer Music: use of computers to create, perform, analyze, notate, (etc.) music

• A Brief History of Computers and Computer Science

Computing (i.e., the processing of information) is an ancient discipline with roots that can be traced to the Greek, Babylonian and Egyptian civilizations. It is rooted in two quests that have motivated innovation for thousands of years:

1) the quest to systematize reasoning
2) the quest to develop means to make computations accurate and efficient

Systematization of Reasoning	*Accurate, Efficient Computational Methods*
Greeks (Aristotle): axiomatic method & foundation of formal logic	Babylonians and Egyptians: invention of abacus multiplication, sqr, sqrt, etc. tables
825 al-Khowârizmî: algebra	
1580 François Vieta: formalized algebra	1600 John Napier: logarithms
1620 Galileo: mathematical formulation of physical sciences	1622 Oughtred: slide rule 1623 Schickard calculator
1640 René Descartes: analytic geometry	1643 Pascal calculator
1700 Leibniz & Newton: calculus Leibniz: binary arithmetic symbolic logic	1700 Leibniz wheel 1805 Jacquard loom 1830 Babbage difference engine Babbage analytic engine
1850 Boole: Boolean algebra 1880 Hilbert's single axiomatic system 1890 Cantor set theory 1930 Gödel Incompleteness Proof 1936 Turing: TM's, Halting Problem 1940 Shannon: digital logic	Ada Byron: first "programmer" 1890 Hollerith: Census tabulation & IBM 1940 ABC, Mark I 1944 ENIAC, EDVAC
1950 Assembly language; Compilers	1950 UNIVAC Transistors
1954 FORTRAN (Backus) 1956 Dartmouth AI Conference 1959 COBOL (Hopper), LISP (McCarthy) 1972 Pascal (Wirth), C (K & R), Smalltalk (Kay) 1979: C with Classes... C++ (Stroustrup) 1996: Java (Sun)	1965 integrated circuits 1975 MITS Altair 1977 Apple and so on.....

Bibliography

* A good general resource on the History of Computing is the journal, *Annals of the History of Computing*. In addition:

E. Braun, S. MacDonald, *Revolution in Miniature: The History and Impact of Semiconductor Electronics*, Cambridge: Cambridge University Press, 1978.

H. Goldstine, *The Computer from Pascal to von Neumann*, Princeton: Princeton University Press.

B. Randell (ed.), *The Origins of the Digital Computer: Selected Papers, 3rd ed*., Berlin: Springer-Verlag, 1982.

* See also the published papers and other biographical resources on some of the "founding fathers" of CS:

V. Hollerith, "Biographical Sketch of Herman Hollerith," *ISIS*, Vol. 62, No. 210, 69-78.

P. Morrison, E. Morrison (eds), *Charles Babbage and His Calculating Engines*, New York: Dover, 1961.

S. Turing, *A.M. Turing*, Cambridge: Heffer, 1959.

A. Taub, *John von Neumann, Collected Works, in 6 Volumes*, New York: Pergamon, 1963.

* For lots of interesting little footnotes on who invented what when, see the History and Bibliography sections of:

D. Knuth, *Fundamental Algorithms, volume 1 of The Art of Programming, 2nd ed.*, Menlo Park, CA: Addison-Wesley, 1973.

* On programming languages:

J. Sammet, *Programming Languages: History and Fundamentals*, Englewood Cliffs, NJ: Prentice Hall, 1969.

R. Wexelblat (ed.), *History of Programming Languages*, New York: Academic Press, 1981.

* For more on computer science as a discipline:

P. Denning, "The Science of Computing: What is Computer Science?" *American Scientist* 73 (1985), 16-19.

P. Denning, et al, "Computing as a Discipline," *Communications of the ACM* 32, 1 (1989), 9-15.