CS 111 presentation on TCS

Mark C. Wilson

2007-10-05

What is (and isn't) Theoretical Computer Science?

- The abstract study of computational processes, independent of particular hardware (running on an idealized model of a computer).

What is (and isn't) Theoretical Computer Science?

- The abstract study of computational processes, independent of particular hardware (running on an idealized model of a computer).
- "[Theoretical] computer science is no more about computers than astronomy is about telescopes" (E. W. Dijkstra).

What is (and isn't) Theoretical Computer Science?

- The abstract study of computational processes, independent of particular hardware (running on an idealized model of a computer).
- "[Theoretical] computer science is no more about computers than astronomy is about telescopes" (E. W. Dijkstra).
- It has a strong mathematical flavour. Since it is not experimental, correctness of an idea is established by mathematical proof.

What is (and isn't) Theoretical Computer Science?

- The abstract study of computational processes, independent of particular hardware (running on an idealized model of a computer).
- "[Theoretical] computer science is no more about computers than astronomy is about telescopes" (E. W. Dijkstra).
- It has a strong mathematical flavour. Since it is not experimental, correctness of an idea is established by mathematical proof.
- Includes: algorithm analysis and design; computational geometry, information theory, cryptography, quantum computing, study of randomness, computational algebra, bioinformatics, formal languages.

What is (and isn't) Theoretical Computer Science?

- The abstract study of computational processes, independent of particular hardware (running on an idealized model of a computer).
- "[Theoretical] computer science is no more about computers than astronomy is about telescopes" (E. W. Dijkstra).
- It has a strong mathematical flavour. Since it is not experimental, correctness of an idea is established by mathematical proof.
- Includes: algorithm analysis and design; computational geometry, information theory, cryptography, quantum computing, study of randomness, computational algebra, bioinformatics, formal languages.
- Has many applications to, and is inspired by, concrete problems in science, engineering, etc.

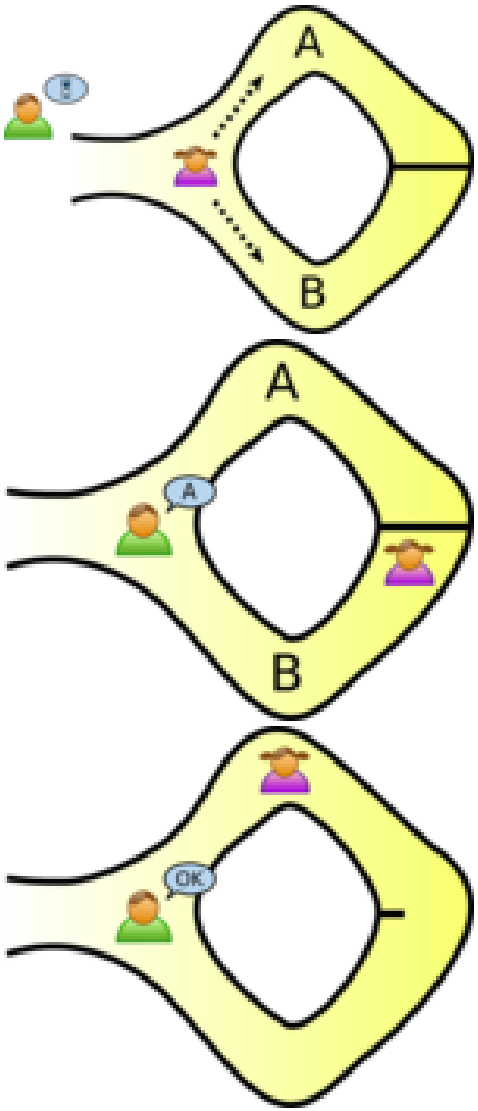
Some big ideas in TCS

- algorithm: systematic solution procedure for computational problem;
- uncomputability: not every problem can be solved by computation;
- recursion (self-reference): a program can use its own output as input;
- universality: anything a supercomputer can compute, an ancient laptop can too;
- tractability: some problems are intrinsically harder than others, no matter how clever we are;
- one-way functions: some functions are easy to compute but their inverse is very hard;
- zero-knowledge proofs (ZKP): Alice can convince Bob that she knows a secret without revealing it.

Algorithms

- A systematic procedure for solving a computational problem.
- Any programming task requires algorithmic thinking.
  Examples:
  - How does TeX decide how to insert spaces?
  - Sort a big database.
  - Find the shortest path between two points in a large network.
  - Find all matches for `*eju*` in an English dictionary.
  - Find the shortest tour for a travelling sales rep, visiting all towns.
  - Decompose a large integer into prime factors.
- Finding some algorithm is often easy, but finding an efficient one is often hard. See COMPSCI220, 320, ...

Cave story for ZKP



Some recent applications of CS Theory

- Engineering: Fast Fourier Transform revolutionized signal processing.

Some recent applications of CS Theory

- Engineering: Fast Fourier Transform revolutionized signal processing.
- Biology: algorithmic approach vastly speeded up work on the Human Genome Project.

Some recent applications of CS Theory

- Engineering: Fast Fourier Transform revolutionized signal processing.
- Biology: algorithmic approach vastly speeded up work on the Human Genome Project.
- Programming languages: fast efficient algorithms for searching, sorting, string matching, . . . now implemented.

Some recent applications of CS Theory

- Engineering: Fast Fourier Transform revolutionized signal processing.
- Biology: algorithmic approach vastly speeded up work on the Human Genome Project.
- Programming languages: fast efficient algorithms for searching, sorting, string matching, . . . now implemented.
- Computer engineering: chip layout and parallel machine architecture improved by graph algorithms.

Some recent applications of CS Theory

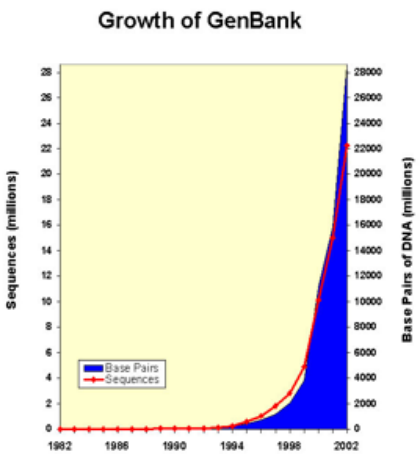
- Engineering: Fast Fourier Transform revolutionized signal processing.
- Biology: algorithmic approach vastly speeded up work on the Human Genome Project.
- Programming languages: fast efficient algorithms for searching, sorting, string matching, . . . now implemented.
- Computer engineering: chip layout and parallel machine architecture improved by graph algorithms.
- E-commerce: public key cryptography enables you to buy things on the web; ZKP allows secure elections, auctions, etc.

Some recent applications of CS Theory

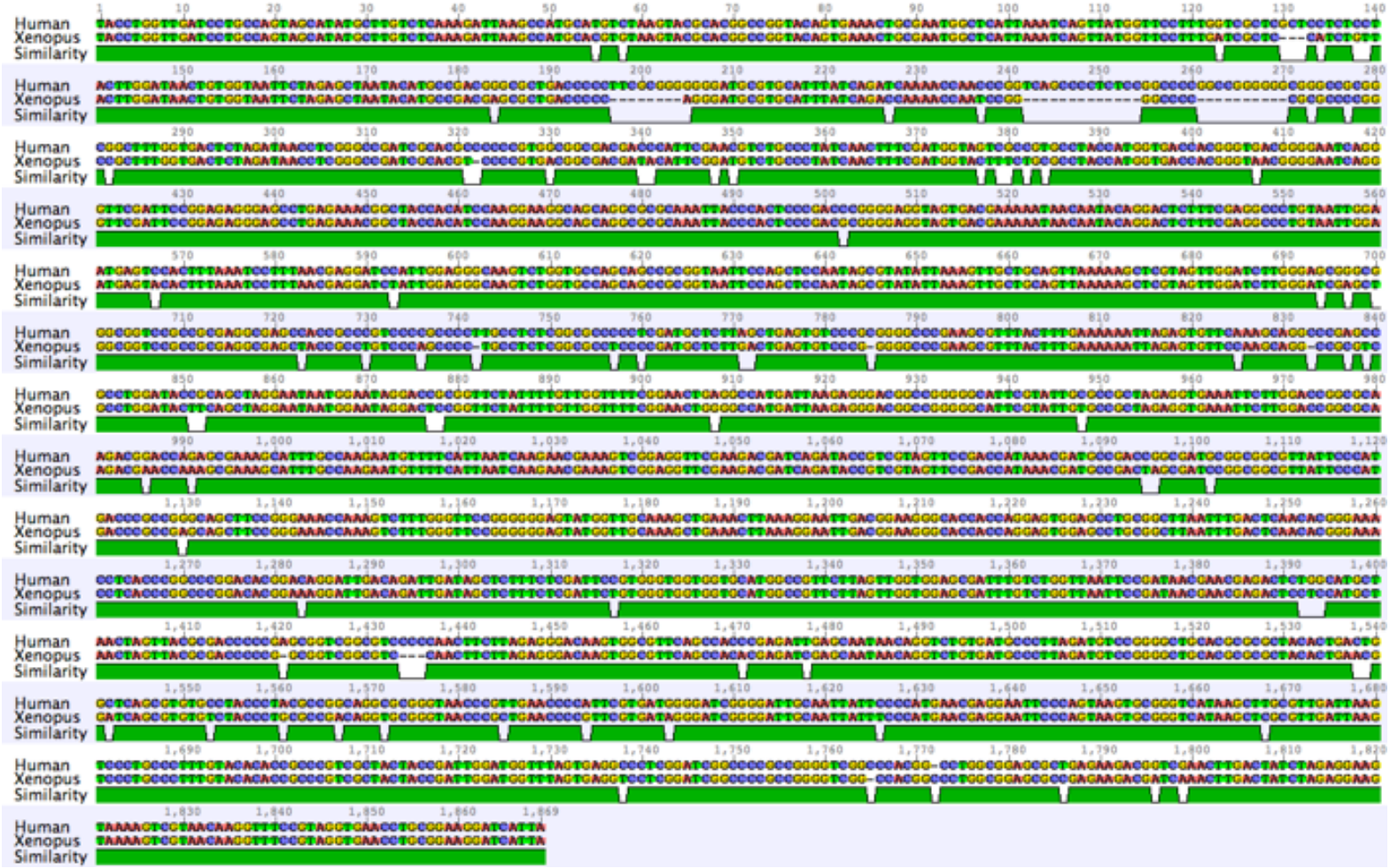
- Engineering: Fast Fourier Transform revolutionized signal processing.
- Biology: algorithmic approach vastly speeded up work on the Human Genome Project.
- Programming languages: fast efficient algorithms for searching, sorting, string matching, . . . now implemented.
- Computer engineering: chip layout and parallel machine architecture improved by graph algorithms.
- E-commerce: public key cryptography enables you to buy things on the web; ZKP allows secure elections, auctions, etc.
- Internet: Google's success is largely due to its PageRank algorithm; routing protocols have been improved by graph algorithms.

Trendy example — bioinformatics

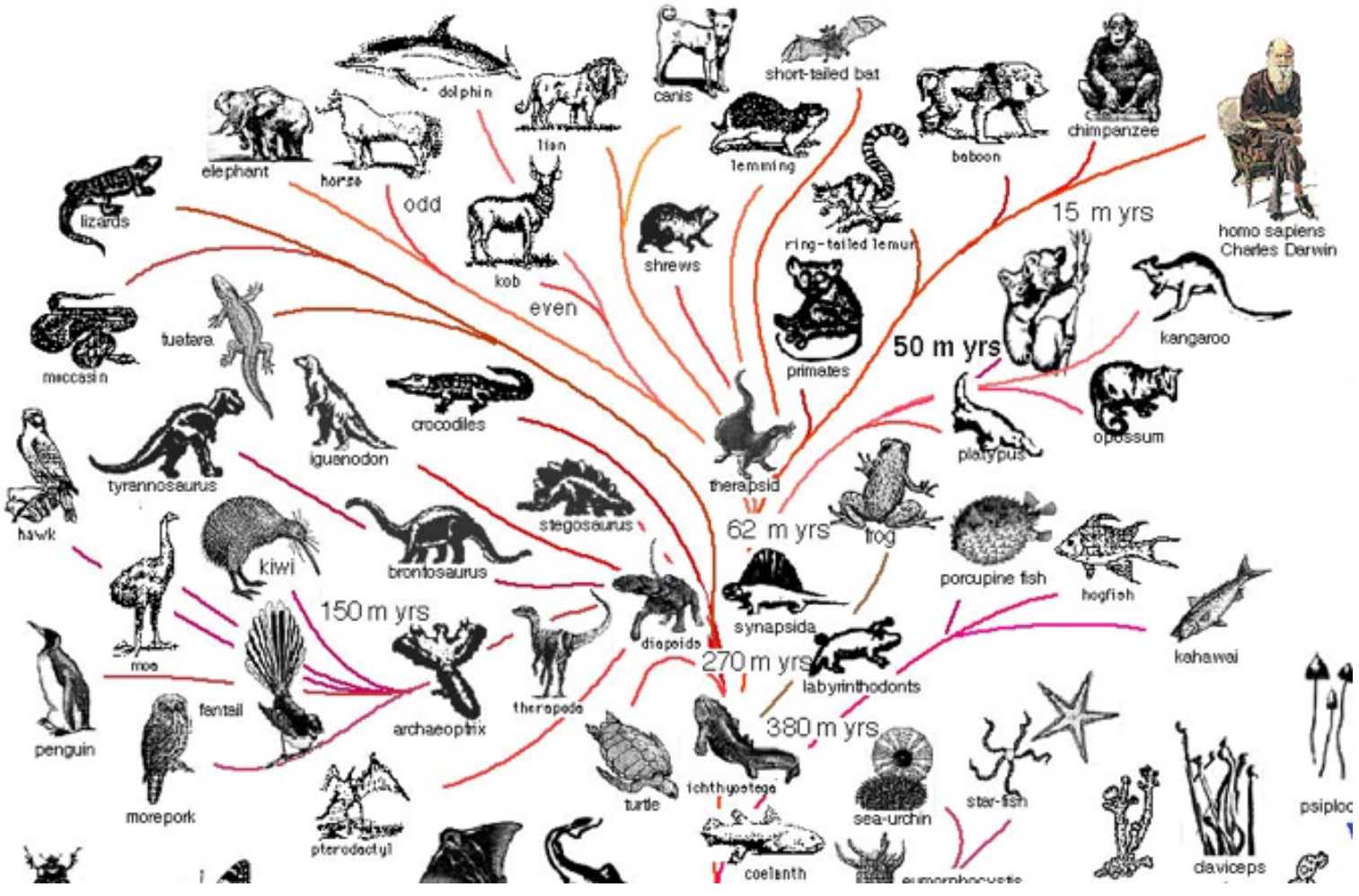
- Has become a central part of biological sciences with many applications in ecology, evolutionary biology, diseases, molecular and cell biology.
- Involves using mathematics, statistics and computer science tools to analyse rapidly growing databases of biological data.
- New algorithms are the biggest requirement right now, because we are being buried under an avalanche of data.



Alignments picture



Tree of life picture



The biggest open problem in Computer Science

- There is a class P of problems that can all be solved in "polynomial time" (roughly speaking, quickly). There is another class NP of problems for which a solution can be verified in polynomial time provided you can guess it.
- Obviously P is contained in NP, but "obviously" NP must be much bigger. No one has been able to prove this.
- The Clay Mathematics Institute has a 1 million USD prize.
- If $P = NP$ then most encryption methods will fail and most mathematics will be done by machine. If $P \neq NP$ then many important practical computational problems will never be solved quickly, and randomization does not really help.
- Some problems thought to be in NP but not P: travelling salesman, subset sum, boolean satisfiability, graph colouring.
- It is known that if a quantum computer could be built, it could solve the above problems in polynomial time.

TCS group at UoA

- 6 permanent staff, active in research, with research interests from fundamental to trendy.
- 2 postdoctoral research fellows
- 10(?) PhD and Masters students
- The CDMTCS (http://www.cs.auckland.ac.nz/CDMTCS/) provides a framework for seminars, conferences, student prizes, etc.
- In 2008 there will be a major programme Algorithms: New Directions and Applications that will involve everyone from staff to the general public. Watch out for it!

CS theory staff

- Cris Calude. Professor. Originally from Romania. Research interests: unconventional models of computation.
- Michael Dinneen. Senior Lecturer. Originally from USA. Research interests: graph algorithms.
- Alexei Drummond. Senior Lecturer. Research interests: bioinformatics.
- Bakhadyr (Bakh) Khoussainov. Professor. Originally from Uzbekistan. Research interests: automata, logic.
- Andre Nies. Senior Lecturer. Originally from Germany. Research interests: computability theory (logic), randomness.
- Mark Wilson. Senior Lecturer. Research interests: analysis of algorithms, discrete mathematics.

CS Theory undergraduate courses

- COMPSCI 105 (principles of computer science)

CS Theory undergraduate courses

- COMPSCI 105 (principles of computer science)
- COMPSCI 220 (algorithms and data structures), COMPSCI 225 (discrete mathematics)

CS Theory undergraduate courses

- COMPSCI 105 (principles of computer science)
- COMPSCI 220 (algorithms and data structures), COMPSCI 225 (discrete mathematics)
- COMPSCI 320 (algorithm design), COMPSCI 350 (computability), COMPSCI 369 (bioinformatics)

CS Theory undergraduate courses

- COMPSCI 105 (principles of computer science)
- COMPSCI 220 (algorithms and data structures), COMPSCI 225 (discrete mathematics)
- COMPSCI 320 (algorithm design), COMPSCI 350 (computability), COMPSCI 369 (bioinformatics)
- COMPSCI 720 (advanced algorithms), COMPSCI 750 (complexity theory), and more.

CS Theory undergraduate courses

- COMPSCI 105 (principles of computer science)
- COMPSCI 220 (algorithms and data structures), COMPSCI 225 (discrete mathematics)
- COMPSCI 320 (algorithm design), COMPSCI 350 (computability), COMPSCI 369 (bioinformatics)
- COMPSCI 720 (advanced algorithms), COMPSCI 750 (complexity theory), and more.
- It is important to take a good amount of mathematics also.

Some benefits of CS Theory training

- General skills that remain relevant as technology changes.

Some benefits of CS Theory training

- General skills that remain relevant as technology changes.
- Ability to think abstractly and extract the essence of a messy real-world problem, then solve it.

Some benefits of CS Theory training

- General skills that remain relevant as technology changes.
- Ability to think abstractly and extract the essence of a messy real-world problem, then solve it.
- Win programming competitions! — it is impossible to win without knowing the material in CS320. The UoA team were 12th in the world last year.

Some benefits of CS Theory training

- General skills that remain relevant as technology changes.
- Ability to think abstractly and extract the essence of a messy real-world problem, then solve it.
- Win programming competitions! — it is impossible to win without knowing the material in CS320. The UoA team were 12th in the world last year.
- "Part of Microsoft's interview process is answering tough questions like the ones we put on our CS320 exams."

Some benefits of CS Theory training

- General skills that remain relevant as technology changes.
- Ability to think abstractly and extract the essence of a messy real-world problem, then solve it.
- Win programming competitions! — it is impossible to win without knowing the material in CS320. The UoA team were 12th in the world last year.
- "Part of Microsoft's interview process is answering tough questions like the ones we put on our CS320 exams."
- "The software company that I am involved with has 7 fulltime software developers and they are basically all theory guys – i.e. know the hard stuff . . . We basically only hire people that know how to think. Who cares whether or not they have done some specific applied paper. They are going to have to constantly learn on the job anyway . . ."

Some of our recent graduates

- Liu Xiong, developer for Microsoft (Redmond, Washington, USA).
- Tiki Wong, works for Microsoft (Redmond, Washington, USA).
- Nodira Khoussainova, PhD student at University of Washington (Seattle, USA). Had research internship at Microsoft Research.
- Jiamou Liu, PhD student with Bakh Khoussainov, spends half the year at Cornell University (USA).
- Owen Auger, PhD student in UoA Engineering Science Dept, working on electricity network modelling.
- Sasha Rubin, postdoctoral fellow supported by NZ government fellowship, visiting University of Wisconsin (USA) and Aachen (Germany).

The future: bright

- TCS provides the theoretical foundation for scientific computing.

The future: bright

- TCS provides the theoretical foundation for scientific computing.
- The older sciences have used mathematical formulae to describe phenomena.

The future: bright

- TCS provides the theoretical foundation for scientific computing.
- The older sciences have used mathematical formulae to describe phenomena.
- The newer sciences (neurophysiology, modern biology, realistic economics, ...) have found that formulae are less useful than algorithms.

The future: bright

- TCS provides the theoretical foundation for scientific computing.
- The older sciences have used mathematical formulae to describe phenomena.
- The newer sciences (neurophysiology, modern biology, realistic economics, ...) have found that formulae are less useful than algorithms.
- "The Algorithm's coming-of-age as the new language of science promises to be the most disruptive scientific development since quantum mechanics." (Bernard Chazelle)

The future: bright

- TCS provides the theoretical foundation for scientific computing.
- The older sciences have used mathematical formulae to describe phenomena.
- The newer sciences (neurophysiology, modern biology, realistic economics, ...) have found that formulae are less useful than algorithms.
- "The Algorithm's coming-of-age as the new language of science promises to be the most disruptive scientific development since quantum mechanics." (Bernard Chazelle)
- There are many exciting research challenges ahead!

References

- UoA CS Theory group, http://www.cs.auckland.ac.nz/research/groups/theory/
- The Algorithm: Idiom of Modern Science, by Bernard Chazelle, http://www.cs.princeton.edu/~chazelle/.
- Is The Thrill Gone? by Sanjeev Arora and Bernard Chazelle, www.cs.princeton.edu/ chazelle/pubs/cacm05.pdf.
- Theory Matters Wiki, http://theorymatters.org/pmwiki/pmwiki.php