Computability and Complexity

Wolfgang Schreiner
Wolfgang.Schreiner@risc.jku.at

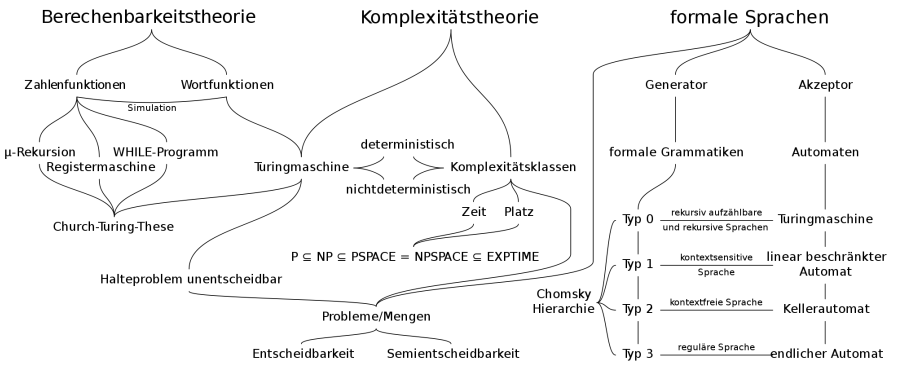
Research Institute for Symbolic Computation (RISC)
Johannes Kepler University, Linz, Austria
http://www.risc.jku.at

Computability and Complexity

- Theoretical computer science:
  - Eternal truths established by means of mathematics/logic.
  - Independent of future technological advances.
- Core questions:
  - Computability: what is computable and what not?
  - Complexity: how efficiently can something be computed?
- Clarifies limits of the field.
  - Physics: build a perpetual motion machine.
    - Impossible: violates second law of thermodynamics.
  - Mathematics: square a circle with compass and straightedge.
    - Impossible: π is not an algebraic number.
  - Computer science: write a reliable halting checker.
    - Impossible: the language of the halting problem is not recursive.

Fundamental knowledge for every decent computer scientist.

The Big Picture



(Wikipedia)

We will cover selected points of this picture.

Theoretical Skeleton

Die Theoretische Informatik . . . bietet Grundlagen für die Definition, Verifikation und Ausführung der Programme von Programmiersprachen, den Bau der Compiler von Programmiersprachen – den Compilerbau – und die mathematische Formalisierung und Untersuchung von meist diskreten Problemstellungen und deren Modellen. Mit Hilfe mathematischer Abstraktion der Eigenschaften von gewonnenen Modellen ergaben sich nützliche Definitionen, Sätze, Beweise, Algorithmen, Anwendungen und Lösungen von bzw. zu Problemen. Die Theoretische Informatik bildet mit ihren zeitlosen, mathematischen Wahrheiten und Methoden ein formales Skelett, das die Informatik in der Praxis mit konkreten Implementierungen durchdringt. Die Theoretische Informatik identifizierte viele unlösbare Problemstellungen mittels der Berechenbarkeitstheorie und erlaubt, häufig mit konstruktiver Beweisführung der Komplexitätstheorie, die Abgrenzung der praktisch effizient lösbaren Probleme von denen, für die das Gegenteil gilt. (Wikipedia)

A map for the landscape of computing.

Practical Application

Zu den konstruktiven Methoden der Theoretischen Informatik zählt auch das Entwerfen von formalen Systemen, Automaten, Graphen und Syntaxdiagrammen sowie das Festlegen von Grammatiken und Semantiken, um eine Problemstellung mit mathematischen Ausdrücken formal zu fassen und von der informellen Ebene abzuheben. Die Konstrukte beschreiben so die innere Logik eines Problems mit mathematisch-logischen Aussagen, was im Weiteren eine formale Untersuchung erlaubt und potenziell neue – durch Beweise gestützte – Aussagen und Algorithmen der formalen Modelle als Resultate erschließbar macht. Neben dem mathematischen Erkenntnisgewinn lassen sich manche der gefundenen Lösungen praktisch implementieren, um Menschen durch Maschinensemantik automatisierte Vorteile der Mathematik- und Computer-Nutzung zu verschaffen. (Wikipedia)

- Compiler construction and language processing.
- System specification and modeling.
- System verification.

A competitive advantage in modern information technology.

Historical Development

- David Hilbert: mechanization of mathematics (1920s)
  - Formal logical system for all of mathematics.
    - Consistent: no contradictions can be derived.
    - Complete: all true statements can be derived.
  - Entscheidungsproblem: devise a mechanical procedure that decides whether a given statement can be derived or not (i.e., is true or not).
- Target: first order predicate logic.
  - John v. Neumann: predicate logic is consistent (1925).
  - Kurt Gödel: predicate logic is complete (1929).
- Kurt Gödel: incompleteness theorem (1931).
  - Arithmetic cannot be captured by any logic system that is both consistent and complete.
  - Full mechanization of mathematics is not possible.

Roots in mathematical logic.

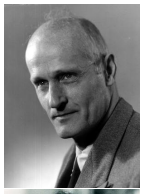
Historical Development

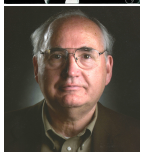
- Alonzo Church/Alan Turing: Entscheidungsproblem is unsolvable (1936/1937).
  - λ-calculus/Turing machine.
  - Both computational models have same expressiveness.
  - Church/Turing Thesis: these models already cover everything that is "computable" in an intuitive sense.
- Turing: also problems in computing are unsolvable.
  - Halting problem is undecidable.
  - Acceptance problem is undecidable.
  - ...
- Alan Turing: 1912-1954.
  - Theoretical computer science, cryptanalysis, artificial intelligence, computational biology.
  - Turing machine, Turing test, ACM Turing award.

Link between mathematics/logic and computer science.

Historical Development

- Finite State Machines (1940s/1950s).
  - Stephen Kleene, George Mealy, Edward Moore, Michael Rabin, Dana Scott.
  - Inspired by formal modeling of neural networks.
  - More limited than Turing machines.
  - But with decidable properties.
- Non-deterministic finite state machines (1959).
  - Machine has multiple choices of possible actions.
  - Simpler construction and manipulation.
  - Equivalent to deterministic machines.
  - Rabin/Scott: 1976 ACM Turing Award.



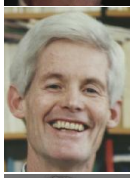


Work horse of (also practical) computer science.

Historical Development

- Complexity theory (since 1960s).
  - Alan Cobhman and Jack Edmonds: complexity class $\mathcal{P}$.
  - Core question $\mathcal{P} \stackrel{?}{=} \mathcal{NP}$.
  - Stephen Cook: SAT-problem is $\mathcal{NP}$-complete (1971).
  - Richard Karp: also other problems are (1972).
  - Cook: 1982 Turing Award, Karp: 1985 Turing Award.
- Active area of research.
  - Till 2002, it was conjectured, primality testing is not in $\mathcal{P}$.
  - Manindra Agrawal et al: new polynomial time primality test (AKS, 2002).

Emerging complexity theory of quantum computing.



Some Seminal Papers

- Alan M. Turing. *On Computable Numbers, with an Application to the Entscheidungsproblem*. Proceedings of the London Mathematical Society 2(42), p. 230–265, July–September, 1936.
- Alonzo Church. *An Unsolvable Problem of Elementary Number Theory*. American Journal of Mathematics 58(2), p. 345–263, 1936.
- Michael O. Rabin and Dana Scott. *Finite Automata and Their Decision Problems*. IBM Journal of Research and Development 3, p. 114–125, 1959.
- Juris Hartmanis and Richard E. Stearns *On the Computational Complexity of Algorithms*. Transactions of the American Mathematical Society 117, p. 285–306, 1965.
- Stephen A. Cook. *The Complexity of Theorem-Proving Procedures*. Proceedings of the 3rd Annual ACM Symposium on the Theory of Computing, p. 151–158, ACM Press, New York, 1971.
- Stephen A. Cook and Robert A. Reckhow. *Time Bounded Random Access Machines*. Journal of Computer and System Sciences 7, p. 354–375, 1973.

Course Organization

- Moodle course:
  `http://www.risc.jku.at/people/schreine/courses/ws2012/bekomp`
  - Register as course member: questions in forum, answers and announcements forwarded per email.
- Lectures: Wolfgang Schreiner.
  - Lecture notes and slides.
  - Exam: February 8, 2012 (KUSSS registration, materials allowed).
- Exercises: Ralf Hemmecke and Burkhard Zimmermann (3 groups).
  - Weekly assignments.
  - 2 exams (no registration, no materials allowed).

Questions preferably in forum or per email or after lecture (for other appointments, send an email).