Introduction to Computer Science

Thomas Schwarz, SJ
Marquette University

Great Ideas I: Computability

- Hilbert's Program
  - *Grundlagenkrise* in Mathematics (~ 1900):
    - How to be sure that Mathematics is true
      - Attempts suffer from paradoxes
        - Example Naïve Set Theory: Russel's set of all sets that do not contain themselves as an element
  - Answers to the Grundlagenkrise
    - Intuitionism:
      - Mathematics is a human activity, it does not discover universal truth
    - Logicism:
      - All mathematics derives from logic
    - Formalism:
      - Mathematics is a game with certain rules that conform to our thinking processes

Great Ideas I: Computability

- A formulation of all mathematics
- Completeness:
  - Proof that all true mathematical statements can be proved in the formalism.
- Consistency:
  - Proof that no contradiction can be obtained in the formalism of mathematics.
- Conservation:
  - Proof that any result about "real objects" obtained using reasoning about "ideal objects" (such as uncountable sets) can be proved without using ideal objects.
- Decidability
  - There is an algorithm for deciding the truth or falsity of any mathematical statement.

Great Ideas I: Computability

- Hilbert's program:
  - Find an algorithm that can decide the truth or falsity of an arbitrary statement in first-order predicate calculus applied to integers

- Gödel's incompleteness result (1931)
  - No such effective procedure can exist

Great Ideas I: Computability

- Formalization of “effective procedure”
  - Each procedure should be described finitely
  - Each procedure should consist of discrete steps, each of which can be carried out mechanically

  - Number of proposals
    - λ-calculus
    - Turing machines (in different versions)
    - RAM machines (computers with infinite memory)

Great Ideas I: Computability

- Church Turing Result:
  - λ-calculus and Turing machines have the same computational power

- Church Hypothesis
  - Turing machines are equivalent to our intuitive notion of a computer
  - What is computable by a human is what is computable by a computer which is what is computable by a Turing machine

Great Ideas I: Computability

- Early career is as a Mathematical Logician
  - Idea: What is computable
  - Proposes the Turing machine as a simple example of what a Mathematician can calculate (without the brilliance)
    - I.e.: A very simple formal way to compute
    - Idea: If something is possible in that simple system then a human Mathematician can do it as well

Great Ideas I: Computability

- *Entscheidungsproblem:* Can every true statement in first order logic (with quantifiers) be derived in first order logic
  - Example for first order logic:
    - There are only n prime numbers.
  - Is equivalent to:
    - There exists n, there exists $p_1, p_2, p_3, \ldots, p_n$ such that if p is a prime, then there exists an index i with $1 \leq i \leq n$ such that $p_i = p$.
- Answers a dream of *Gottfried Leibniz*: Build a machine that could manipulate symbols in order to determine the truth values of mathematical statements.

Great Ideas I: Computability

- Made it plausible that a Mathematician is not more powerful than the Turing calculus
- Proved limitations on what a Turing calculus can achieve

Post-Turing Machine

- A Turing machine consists of
  - An infinitely-long tape divided into squares that are initially blank (denoted by a symbol ‘b’)
  - A read-write head that can read and write symbols
  - A control unit that consists of a state machine
    - In a given state and when reading a given symbol:
      - The machine goes to a new state
      - The machine writes a new symbol
      - The machine moves to the left or the right by one step.

Post-Turing Machines

- Turing machine input
  - A string on the tape, with all other symbols being blanks.

- Turing machine output
  - Turing machines can make decisions:
    - By writing them on the tape
    - By entering an “accepting” or a “rejecting” state
    - These possibilities are actually equivalent

http://morphett.info/turing/turing.html

Post-Turing Machines

- Turing machine programs:
  - A program consists of a set of transition rules:
    - Current state, Current Symbol —> New State, New Symbol, Move

- Note: All Turing machine programs are finite

Post-Turing Machine

- Despite its simplicity, a Turing machine can imitate any computer (known today)

Post-Turing Machine

- Turing machine programs
  - consists of lines

<curr. state> <curr. symb> <new symb> <dir> <new state>

Post-Turing Machine

- People have build Turing machines
  - For Fun
    - Because we can emulate Turing machines much faster
  - http://aturingmachine.com
  - https://www.youtube.com/watch?v=2PjU6DJyBpw
  - https://www.youtube.com/watch?v=E3keLeMwfHY&t=75s
  - https://www.youtube.com/watch?v=vo8izCKHiF0

Great Ideas I: Computability

- Results:
    - There are problems that cannot be computed
- But the problem is in principle solvable
    - Halting problem: Will a Turing machine on a given input halt or will it continue for ever?
    - In many concrete cases, can be solved.