Computation

Extended Summary from B. Jack Copeland's Paper,
Chapter 1 in The Blackwell Guide to the Philosophy of Computing and Information

Turing Machine – *The abstract model of computation*

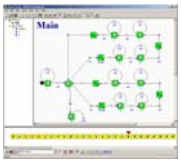
- **Turing machine (TM) consists of**
  Memory in the form of a tape, divided into squares each of which may hold a single symbol from a finite alphabet.
  A head (scanner), which moves back and forth through the tape to read or write symbols.

- **Basic operations of a TM**
  erase, *write*, *shift*, *halt*

- **Change of state**
  In addition to the basic operations, the scanner is able to change states. Each TM has a table of instructions (state transition table) showing what basic operations should be done in a state, and upon which conditions the state will be changed to which state.

Turing Machine – *A small demonstration*

- **Demonstration using Visual Turing**
  Free software by Christian Cheran. You can download it from http://www.cheransoft.com/vturing/index.html



- **There are many representations of TMs.**
  See http://www.igs.net/~tril/tm/ for another simulation.
- **Universal Turing Machine (UTM)**
  A UTM is a TM with a fixed table of instructions. It takes as input any TM and its input for execution, encoded in a specific way, and simulates the execution steps the encoded TM would go through.

Turing Machines – *Today's computing giants as mathematical relatives*

- **A TM is as powerful as any computer.**
  As of today, we know of no physically realizable computer hardware that can carry out a computation that cannot be carried out by a TM!
  Efficiency? Well, there must be a reason that today's computing machinery exists, right?
- **Not all computers were UTMs.**
  The first electronic computers were not UTMs: their programs were hard-wired (just like a calculator).
  Colossus (British, 1943)
  ENIAC (American, 1945)
- **There was a competition for building the first UTM in hardware.**
  John von Neumann (America)
  Max Newman (UK) → Manchester Baby
  Manchester University, UK
  June 21, 1948

Turing Machines – *The idea behind*

- **Father of TMs: Alan Turing**
  The first paper in Computation as a field is acknowledged to be Alan Turning's 1936 paper titled "On computable numbers, with an application to the Entscheidungs problem." In this paper he first defined the concept of TMs.
- **Father of λ-Computability: Alonzo Church**
  λ-Computability, which defines a mathematical class of functions equivalent to the class of functions defined through Turing Machines, was also published concurrently by Alonzo Church (1936).
- **Characterization of Human Computers**
  Both studies aim to characterize "human computers", and to relate this to Hilbert's program (more on both will come shortly).

Human Computers – *How the computations were carried out before computers*

- **A computing non-machine:**
  When Turing wrote "On computable numbers", a computer was not a machine at all, but a *human being*: a mathematical assistant who calculated by rote, in accordance with some *effective method* supplied by an overseer prior to the calculation.
- **An effective method**
  - Demands no ingenuity from the human carrying it out
  - Produces the correct answer in a finite number of steps
- **The concept of "effective method" is an informal one!**

Church-Turing Thesis – *TM is the ultimate model (important: to what?)*

- **The Church-Turing thesis is stated as:**
  The UTM is able to perform any calculation that any human computer can carry out.
  No proof!
  But no counterexamples yet (almost 70 years passed).

- **If the Church-Turing thesis is correct,**
  That is to say, if the TMs correctly model what is informally called an "effective method", then the existence of "effective method" discussions can be replaced by existence of UTMs for doing the task. This provides a mathematical basis to the concept of "effective method".
- **Note: Converse of the Church-Turing thesis** is obviously true that any problem-solving method that can be carried out by a UTM can be carried out by a human computer.

Limits of Computation – *What is a UTM not able to do?*

- **Computable number:**
  (A slightly Modified form of Turing's definition)
  A number is computable if and only if there's a TM capable of calculating the digits of its decimal representation in sequence.

- **How many are there of computable numbers?**
  There are countably infinite (which means you can count computable numbers with the natural numbers).

- **Example of a non-computable number** is the Chaitin's constant (Ω), which gives the probability of halting of a TM constructed from a random binary sequence:

$$\Omega \equiv \sum_{p \text{ halts}} 2^{-|p|}$$

Limits of Computation – *What is a UTM not able to do?*

- **The Printing Problem:**
  Turing showed that there exists no UTM that takes an encoding of an arbitrary TM and an input of it, and decides whether this TM writes on its tape "0" at some point in its computations or not.
- **The Halting Problem:**
  (Davis 1958) Given any arbitrary TM, decide whether or not the machine will eventually halt when started on a blank tape.
  - You may find this problem stated in many forms.
  - If you remember that any computing machinery we have today is mathematically equivalent to a UTM, nonexistence of its solution means that no programs can take another program and its input, and decide whether it will stop or get stuck into an infinite loop. (Guess why there's an army of researchers studying what is called "Program Verification/Testing" ;))

Limits of Computation – *What is a UTM not able to do?*

- **Computable Functions**
  A function is a mapping from "arguments" (or inputs) to "values" (or outputs).
  A function is said to be *computable by a TM* if there exists a TM which, for all arguments of the function, would take in the arguments of the function, carry out some finite number of basic operations, and produce the corresponding value.
- **Halting Function**
  Assume an ordering of all possible TMs (a mapping from TMs to natural numbers). The value of the halting function for any argument n is 1 if the n^{th} TM in the ordering halts when started with a blank tape, and 0 if it runs forever (such as a TM writing the digits of π).
- **Halting Theorem**
  The halting function is not computable by a TM.

The Entscheidungsproblem

- **David Hilbert's 1900 Paris Lecture**
  In the year 1900, David Hilbert gave a speech in which he identified a set of problems that would set the agenda for mathematics for the following century. "Entscheidung" means "decision", and his tenth problem in his speech is about determination of solution to a diophantine equation and is titled "Entscheidung der Lösbarkeit einer Diophantischen Gleichung".
- **Turing's work is not about the tenth problem!**
  You might find in some places that Turing solved the tenth problem of Hilbert. It is not right.
  The entscheidungsproblem Turing worked on is part of what is called the Hilbert program in mathematics, which he expressed in the 1920s. Hilbert noted that mathematicians should seek to express mathematics in the form of a *complete*, *consistent*, and *decidable* formal system.

The Entscheidungsproblem – *Consistency, Completeness, and Decidability*

- **Consistency**
  A consistent system is one that contains no contradictions. In an inconsistent system –a system containing contradictions- *any* statement can be derived from a contradiction.
- **Completeness**
  A complete system is one in which every true statement is provable.
- **Decidability**
  A decidable system is one where there exists an *effective method* using which one can tell for any particular statement whether it is provable in the system or not.

- **Hilbert's program's aim (repeated):**
  Mathematics expressed as a consistent, complete and decidable formal system.

The Entscheidungsproblem – *Gödel and Turing's work*

- **Consistency and Completeness**
  In 1931, Gödel proved that the system called Peano arithmetic is, if consistent, incomplete (Gödel's *first incompleteness theorem*). Later he generalized this result, pointing out that "due to A.M. Turing's work, a precise and unquestionably adequate definition of the general concept of formal system can now be given," with the consequence that incompleteness can "be proved rigorously for *every* consistent formal system containing a certain amount of finitary number theory" (Copeland cites (Gödel, 1965)).
- **Decidability**
  Decidability was not addressed by Gödel, but by Turing and by Church independently. They both showed that the *first-order predicate calculus*, which is presupposed by any formal system of arithmetic, is not decidable.
- **Gödel, Turing and Church's results were a catastrophe for the Hilbert program.**

Misunderstandings of the Church-Turing Thesis

- **Generalization to all machines**
  A myth has arisen: The UTM can simulate the behavior of *any* machine, therefore Turing gave a treatment of the limits of mechanism.
  - **What is wrong** is that UTM can simulate any *Turing Machine*, not any machine whatsoever. Thus his results involve limits of TMs, not of mechanism. The Church-Turing thesis concerns *effective methods* only (methods human beings working by rote with pen and paper employ), and is still not proved. Among a machine's repertoire of basic operations, there may be those that no human working by rote with pen and paper can perform.
- **Here's a quotation showing the myth at work:**
  Turing's "results entail something remarkable, namely that a standard digital computer, given only the right program, a large enough memory and sufficient time, can compute *any* rule-governed input-output function. That is, it can display any systematic pattern of responses to the environment whatsoever." (Paul and Patricia Churchland, 1990)

Misunderstandings of the Church-Turing Thesis
– *Distant Cousins*

- "Church-Turing Thesis: If there is a well defined procedure for manipulating symbols, then a Turing machine can be designed to do the procedure." (Henry, 1993)
- "It is difficult to see how any language that could actually be run on a physical computer could do more than Fortran can do. The idea that there is no such language is called Church's thesis." (Geroch & Hartle, 1986)
- "That there exists a most general formulation of machine and that it leads to a unique set of input-output functions has come to be called *Church's thesis*." (Newell, 1980)

- **Each of these say more or something different than the original form of Church-Turing Thesis.**

The Maximality Thesis

- **Generating a function**
  A machine m is said to be able to generate a certain function if m can be set up so that presented with any valid argument to the function, it will come up with the corresponding value of the function after some finite number of steps.
- **Maximality Thesis**
  All functions that can be generated by machines (working on finite input in accordance with a finite program of instructions) are computable by Turing machine.

- **Does Maximality Thesis hold?**
  Depends on what is meant by machine. If machine is an abstraction which covers what cannot exist in our world (physically), then it is false and counterexamples exist. Under a physical (this worldly) interpretation, we don't know if it holds.

Computational Form of Church-Turing Thesis

- **"Computation" as a term**
  Definition of computation is tied to effectiveness:
  A function is said to be computable if and only if there is an effective method for determining its values.

- **Computational Form of Church-Turing Thesis**
  Every computable function can be computed by a TM.

- **Beware of the corollaries, sometimes they are confusing taken out of their context:**
  "Certain functions are uncomputable in an absolute sense: uncomputable even by [Turing machine], and, therefore, uncomputable by any past, present, or future real machine. (Boolos & Jeffrey 1980).

Psychology, Philosophy of Mind, and Church-Turing Thesis

- **Application of maximality thesis**
  If the mind-brain is a machine, then the TM computable functions provide sufficient mathematical resources for a full account of human cognition.

  Corollary (Information processing view): Thus psychology must be capable of being expressed ultimately in terms of the TM.

- **The simulation fallacy**
  It's the belief that any biological or physical system (incl. the brain) can be simulated by a TM.

Psychology, Philosophy of Mind, and Church-Turing Thesis
– *The Simulation Fallacy*

- **Simulation:**
  (Dict. Def.) The act of imitating the behavior of some situation or some process by means of something suitably analogous.
  Simulation does not include a complete model of the original. Simulations does not constitute descriptive models by themselves.
- **Emulation:**
  (Dict. Def.)1.Ambition to equal or excel, 2. technique of one machine obtaining the same results as another
  Emulation involves a complete model.

- **The simulation fallacy**
  It's the belief that any biological or physical system (incl. the brain) can be *simulated* by a TM.
- **Thesis S:**
  Any process that can be given a mathematical description (or a "precise enough characterization of a set of steps" or that is scientifically describable or scientifically explicable) can be simulated by a Turing machine.

Psychology, Philosophy of Mind, and Church-Turing Thesis
– *The Simulation Fallacy*

- **Thesis S is widely taken to be correct in philosophy of mind and psychology.**
  "If the question ["Is consciousness computable?"] asks "Is there some level of description at which conscious processes and their correlated brain processes can be simulated [by a TM]?" the answer is trivially yes. Anything that can be described as a precise series of steps can be simulated [by a TM]." (Searle, 1997)

  "Church's Thesis says that whatever is computable is Turing computable. Assuming, with some safety, that what the mind-brain does is computable, then it can in principle be simulated by a computer." (Churchland & Churchland, 1983)

Computation (Ch. 1) – *End of the road?*

ANY QUESTIONS?