## The Turing Machine

The term *Computer* suggests that their main use is for numerical calculation. But modern computers are versatile symbol processors. Their versatility can be traced back to the theory of computation as developed by mathematical logicians. In particular, the wonderful discovery made by Alan Turing of the universal (general purpose) digital computer as an abstract mathematical structure has had a decisive influence on the development of computers.

At the turn of the century, David Hilbert the famous German mathematician, proposed the *Entscheidungs problem*. The *Entscheidungs problem* asks for an algorithm which when presented with any formula in first order logic will decide (correctly!) whether or not it is a valid formula.(To be honest, we are giving here a conveniently modified version of the original formulation). Hilbert had shown that large parts of mathematics could be formalized within first order logic. Hence when Kurt Gödel -arguably the greatest logician of this century- came up with his famous *incompleteness theorems*, it was widely suspected that the *Entscheidungs problem* did not admit a solution.

However the trouble was that for showing that there existed *no* algorithm which would solve the *Entscheidungs problem* one had to first come up with a formal answer to the question: What is an algorithm? Gödel himself had proposed a notion called general recursive functions. This notion was a modified version of a proposal made by the French logician Hildebrand. Simultaneously, Alonzo Church at Princeton University had proposed a calculus of functions called the $\lambda$-calculus. He showed that his notion was mathematically equivalent to Gödel's notion. He advocated the thesis (Church's thesis) that a function is computable i.e. its calculation can be stated as an algorithm if and only if it is definable ( $\lambda$-definable) in his calculus. Finally, he showed that under the regime of his thesis the *Entscheidungs problem* was unsolvable. Gödel however was not convinced that Church's thesis was a natural one because both the $\lambda$-calculus and general recursive functions were far removed from the intuitive understanding of what an algorithm is.

At this stage, Alan Turing in a paper entitled 'On computable numbers with an application to the *Entscheidungs problem*' publised in *The London Math. Soc.* in 1936 formulated a model now called a Turing machine. He proposed that a function is computable if and only if it is computable by a Turing machine (Turing's thesis). He then proceeded to show that there was no Turing machine which could solve the *Entscheidungs problem*. He also showed that there was a universal Turing machine (which he was able to explicitly define!) which could carry out the operations of *any* Turing machine. The construction that he used to achieve this would be called in modern terminology, an interpretative program.

Thus, much of what Turing had accomplished regarding the *Entscheidungs problem* was a rediscovery of what was already done by Church, albeit without knowing it. However his analysis of what constitutes an effective computation and his discovery of the universal computing machine were stunningly original. The simplicity and the transparency of his analysis readily convinced Gödel and Church that Turing's formulation of what is a computable function was the most natural one.. Apparently Gödel said : "..with this concept one has for the first time succeeded in giving an absolute definition of an interesting epistemological notion, i.e.,

one not depending on the formalism chosen".

Turing's notion of what is computable turned out to be equivalent to Church's notion as shown by Turing himself. Indeed a number of other formulations (due to Post, Markov...) have all turned out to possess the same mathematical power.

So what is a Turing machine? The answer is easy to comprehend once Turing's analysis of what is an effective computation is understood. Turing's key insight was that if a function is to be computable then it should be *mechanically* computable. He consequently considered the behaviour of a human being 'blindly' carrying out a sequence of steps as dictated by some prespecified algorithm. He then proceeded, through a number of obvious simplifications, to arrive at the notion of a Turing machine.For instance, he pointed out it involves no loss of generality to assume that the computation is carried out on a one dimensional paper i.e. an infinite tape divided into cells. Actually it suffices to assume a finite tape which can be extended by a fixed number of cells on demand. Taking into account the limitations of our sensory and mental apparatus Turing also arrived at the following restrictions to be met by a computer. There is a fixed upper bound on the number of distinct symbols which can be *written on* and *read from* a single square. Among these, there is one special symbol called the blank symbol. There is a fixed upper bound to the number of contiguous cells whose contents the computer can take in at a glance in one step. At each step, the computer can alter the contents of just a fixed number of cells each of which is atmost a fixed distance away from the squares being currently scanned. There is a fixed upper bound to the distance the computer moves along the tape between each step in order to scan a fresh set of cells. There is a fixed upper bound on the number of 'states of mind' of the computer. Moreover at each step the action taken by the computer (what symbols to write on which squares, which squares to scan in the next step and which mental state to assume next) is unambiguously determined by the current mental state and the contents of the cells currently being scanned. Finally, all but a finite number of cells conta is the blank symbol to start with.

One crucial point to be emphasized is that the mathematical presentation of Turing is strictly finitary. One specifies a finite set of (mental) states, a finite alphabet set and a finite instruction set. Everything else follows from a prespecified list of conventions as to how the input is encoded and presented on the tape, which square(s) the machine is scanning when computation begins, how the end of a computation is signalled, how the result of the computation of the machine is to be decoded etc. A wide variety of choices are available concerning these conventions and they all lead to the same notion of computability.

As mentioned earlier, the related idea of a universal Turing machine seems to have played an indirect but substantial role in the work of John von Neumann and Turing himself in the development modern general purpose electronic computers. The Turing machine model has also provided the basis for measuring the inherent space and time complexity of algorithms. This has led to the development of the area known as *complexity theory* which is one of the corner stones of present day theoretical computer science.Equally important, the influence exerted by mathematical logic via Turing's work has played a decisive role in the development of the modern computer as a tool for information processing.

*P S Thiagarajan*