lan Mathison Turing conceived of the modern computer in 1935. Today all digital computers are, in essence, Turing machines. The

British mathematician also pioneered the field of artificial intelligence, or AI, proposing the famous and widely debated Turing test as a way of determining whether a suitably programmed computer can think. During World War II, Turing was instrumental in breaking the German Enigma code in part of a topsecret British operation that historians say shortened the war in Europe by two years. When he died at the age of 41, Turing was doing the earliest work on what would now be called artificial life, simulating the chemistry of biological growth.

Throughout his remarkable career, Turing had no great interest in publicizing his ideas. Consequently, important aspects of his work have been neglected or forgotten over the years. In particular, few people even those knowledgeable about computer science are familiar with Turings fascinating anticipation of connectionism, or neuronlike computing. Also neglected are his groundbreaking theoretical concepts in the exciting area of hypercomputation. According to some experts, hypercomputers might one day solve problems heretofore deemed intractable.

#### The Turing Connection

igital computers are superb number crunchers. Ask them to predict a rockets trajectory or calcu

late the financial figures for a large multinational corporation, and they can churn out the answers in seconds. But seemingly simple actions that people routinely perform, such as recognizing a face or reading handwriting, have been devilishy tricky to program. Perhaps the networks of neurons that make up the brain have a natural facility for such tasks that standard computers lack. Scientists have thus been investigating computers modeled more closely on the human brain.

Connectionism is the emerging science of computing with networks of artificial neurons. Currently researchers usually simulate the neurons and their interconnections within an ordinary digital computer (just as engineers create virtual models of aircraft wings and skyscrapers). A training algorithm that runs on the computer adjusts the connections between the neurons, honing the network into a specialpurpose machine dedicated to some particular function, such as forecasting international currency markets.

Modern connectionists look back to Frank Rosenblatt, who published the first of many papers on the topic in 1957, as the founder of their approach. Few realize that Turing had already investigated connectionist networks as early as 1948, in a littleknown paper entitled Intelligent Machinery.

Written while Turing was working for the National Physical Laboratory in London, the manuscript did not meet with his employers approval. Sir Charles Darwin, the rather headmasterly director of the laboratory and grandson of the great English naturalist, dismissed it as a schoolboy essay. In reality, this farsighted paper was the first manifesto of the field of artificial intelli

gence. In the work which remained unpublished until 1968, 14 years after Turings death the British mathematician not only set out the fundamentals of connectionism but also brilliantly introduced many of the concepts that were later to become central to AI, in some cases after reinvention by others.

In the paper, Turing invented a kind of neural network that he called a Btype

unorganized machine, which consists of artificial neurons and devices that modify the connections between them. Btype machines may contain any number of neurons connected in any pattern but are always subject to the restriction that each neurontoneuron connection must pass through a modifier device.

All connection modifiers have two training fibers. Applying a pulse to one of them sets the modifier to pass mode, in which an input either 0 or 1 passes through unchanged and becomes the output. A pulse on the other fiber places the modifier in interrupt mode, in which the output is always 1, no matter what the input is. In this state the modifier destroys all information attempting to pass along the connection to which it is attached.

Once set, a modifier will maintain its function (either pass or interrupt) unless it receives a pulse on the other training fiber. The presence of these ingenious connection modifiers enables the training of a Btype unorganized machine by means of what Turing called appropriate interference, mimicking education. Actually, Turing theorized that the cortex of an infant is an unorganized machine, which can be organized by suitable interfering training.

Each of Turings model neurons has two input fibers, and the output of a neuron is a simple logical function of its two inputs. Every neuron in the network executes the same logical operation of not and (or NAND): the output is 1 if either of the inputs is 0. If both inputs are 1, then the output is 0.

Turing selected NAND because every other logical (or Boolean) operation can

be accomplished by groups of NAND neurons. Furthermore, he showed that even the connection modifiers themselves can be built out of NAND neurons. Thus, Turing specified a network made up of nothing more than NAND neurons and their connecting fibers about the simplest possible model of the cortex. In 1958 Rosenblatt defined the theoretical basis of connectionism in one suc

cinct statement: Stored information takes the form of new connections, or transmission channels in the nervous system (or the creation of conditions which are functionally equivalent to new connections). Because the destruction of existing connections can be func

tionally equivalent to the creation of new ones, researchers can build a network for accomplishing a specific task by taking one with an excess of connections and selectively destroying some of them. Both actions destruction and creation are employed in the training of Turings Btypes.

At the outset, Btypes contain random interneural connections whose modifiers have been set by chance to either pass or interrupt. During training, unwanted connections are destroyed by switching their attached modifiers to interrupt mode. Conversely, changing a modifier from interrupt to pass in effect creates a connection. This selective culling and enlivening of connections hones the initially random network into one organized for a given job.

Turing wished to investigate other kinds of unorganized machines, and he longed to simulate a neural network and its training regimen using an ordinary digital computer. He would, he said, allow the whole system to run for an appreciable period, and then break in as a kind of inspector of schools and see what progress had been made. But his own work on neural networks was carried out shortly before the first generalpurpose electronic computers became available. (It was not until 1954, the year of Turings death, that Belmont G. Farley and Wesley A. Clark succeeded at the Massachusetts Institute of Technology in running the first computer simulation of a small neural network.)

Paper and pencil were enough, though, for Turing to show that a sufficiently large Btype neural network can be configured (via its connection modifiers)

in such a way that it becomes a generalpurpose computer. This discovery illuminates one of the most fundamental problems concerning human cognition.

From a topdown perspective, cognition includes complex sequential processes, often involving language or other forms of symbolic representation, as in mathematical calculation. Yet from a bottomup view, cognition is nothing but the simple firings of neurons. Cognitive scientists face the problem of how to reconcile these very different perspectives.

Turings discovery offers a possible solution: the cortex, by virtue of being a neural network acting as a generalpurpose computer, is able to carry out the sequential, symbolrich processing discerned in the view from the top. In 1948 this hypothesis was well ahead of its time, and today it remains among the best guesses concerning one of cognitive sciences hardest problems.

#### Computing the Uncomputable

n 1935 Turing thought up the abstract device that has since become known as the universal Turing machine. It consists of a limitless memory

I

100



**Turings Anticipation of Connectionism**

Iuntil 14 years after his death (*top*),

n a paper that went unpublished

Alan Turing described a network of

artificial neurons connected in a random manner. In this Btype unorganized machine (*bottom left*), each connection passes through a modifier that is set either to allow data to pass unchanged (*green fiber*) or to destroy the transmitted information (*red fiber*). Switching the modifiers from one mode to the other enables the network to be trained. Note that each neuron has two inputs (*bottom left, inset*) and executes the simple logical operation of not and, or NAND: if both inputs are 1, then the output is 0; otherwise the output is 1.

In Turings network the neurons interconnect freely. In contrast, modern networks (*bottom center*) restrict the flow of information from layer to layer of neurons. Connectionists aim to simulate the neural networks of the brain (*bottom right*).

Scientific American

April 1999

*Alan Turings Forgotten Ideas in Computer Science*

that stores both program and data and a scanner that moves back and forth through the memory, symbol by symbol, reading the information and writing additional symbols. Each of the machines basic actions is very simple such as identify the symbol on which the scanner is positioned, write 1 and move one position to the left. Complexity is achieved by chaining together large numbers of these basic actions. Despite its simplicity, a universal Turing machine can execute any task that can be done by the most powerful of todays computers. In fact, all modern digital computers are in essence universal Turing machines [see Turing Machines, by John E. Hopcroft; Scientific American, May 1984].

Turings aim in 1935 was to devise a machine one as simple as possible capable of any calculation that a human mathematician working in accordance with some algorithmic method could perform, given unlimited time, energy, paper and pencils, and perfect concentration. Calling a machine universal merely signifies that it is capable of all such calculations. As Turing himself wrote, Electronic computers are in

tended to carry out any definite ruleofthumb process which could have been done by a human operator working in a disciplined but unintelligent manner.

Such powerful computing devices notwithstanding, an intriguing question arises: Can machines be devised that are capable of accomplishing even more? The answer is that these hypermachines can be described on paper, but no one as yet knows whether it will be possible to build one. The field of hypercomputation is currently attracting a growing number of scientists. Some speculate that the human brain itself the most complex information processor known is actually a naturally occurring example of a hypercomputer.

Before the recent surge of interest in hypercomputation, any informationprocessing job that was known to be too difficult for universal Turing machines was written off as uncomputable. In this sense, a hypermachine computes the uncomputable.

Examples of such tasks can be found in even the most straightforward areas of mathematics. For instance, given arithmetical statements picked at random, a universal Turing machine may

not always be able to tell which are theorems (such as 7 + 5 = 12) and which are nontheorems (such as every number is the sum of two even numbers). Another type of uncomputable problem comes from geometry. A set of tiles variously sized squares with different colored edges tiles the plane if the Euclidean plane can be covered by copies of the tiles with no gaps or overlaps and with adjacent edges always the same color. Logicians William Hanf and Dale Myers of the University of Hawaii have discovered a tile set that tiles the plane only in patterns too complicated for a universal Turing machine to calculate. In the field of computer science, a universal Turing machine cannot always predict whether a given program will terminate or continue running forever. This is sometimes expressed by saying that no generalpurpose programming language (Pascal, BASIC, Prolog, C and so on) can have a foolproof crash debugger: a tool that detects all bugs that could lead to crashes, including errors that result in infinite processing loops.

Turing himself was the first to investigate the idea of machines that can perform mathematical tasks too difficult