Getting It Taped

The Turing machine is a purely theoretical device, used for deciding whether a problem is computable or not

So far in THE HOME COMPUTER COURSE, we have tended to emphasise practical subjects, and things to do on your home computer. In this article, however, we're going to take a look at the theoretical side of computers: the field that is called 'computer science'. This is to computing what pure mathematics is to engineering — a highly theoretical subject, but one from which the practical ideas ultimately derive.

The Turing machine, for example, is a purely theoretical idea, developed by Alan Turing (see page 200) to assist in the study of algorithms and computability. It is really the minimal possible computer, so that if it is possible to prove that a particular problem could not be solved using a Turing machine, then that problem could be said to be 'non-computable'. Turing decided that such a minimal computer would need three facilities: an external storage for recording and storing input and output information; a means for reading from and writing to that storage; and a control unit to determine the actions to be undertaken.

A Turing machine is therefore usually defined as having a tape (if it helps, think of it as a magnetic tape), which is infinite in length (that is to say: however much tape is needed to solve a problem, there will always be enough). The tape is divided up into squares, which will either be blank or contain a symbol. A tape head mechanism that can read or write the symbols in the squares moves along the tape, receiving its instructions from a control unit that tells it what symbols to write and the direction in which to move next.

The control unit contains an execution program, and in this respect a Turing machine can be considered to have been 'built' specifically to perform one application, since there is no provision in the specification for loading or altering a program. We use the term 'built' advisedly, since the only Turing machines that have ever been physically constructed have been purely for educational purposes. However, it is a relatively simple exercise to write a BASIC program that will simulate the operation of a Turing machine on a home computer.

The control program in a Turing machine is made up of a collection of 'quintuples', or statements that contain five elements. Which quintuple is executed at any stage depends on two factors: the symbol in the square currently underneath the tape head, and the 'state' or 'condition' of the machine. This state is a purely arbitrary quality: we can specify that the machine starts off in state S_A, and when it reaches the special state H then it halts, the computation being finished. In between, the state will change many times according to instructions from the quintuples. The state merely reflects what has happened in the computation so far, and serves to select which quintuple is executed next (again, if it helps, think of it as a flag variable in BASIC programming).

The five elements of each quintuple are:

1) The current state of the machine;
2) The symbol in the square of tape underneath the head;
3) The symbol to be written in that square (this is the same as 2 if no change to the data is required);
4) The state that the machine should now go into; and
5) The direction in which the tape head should move — left or right.

The quintuple (S_A,5,3,S_B,R), for example, will be executed whenever the machine is in state S_A and the tape head reads a 5. The 5 will then be replaced by a 3, the machine changed from state S_A to S_B, and the tape head moved one square to the right.

Designing a theoretical Turing machine to perform a particular task involves specifying the format in which your input data will be presented to the machine on tape, the format of the output data on tape when the computation is finished (i.e. the machine is in state H), and the set of quintuples needed to execute the algorithm.

In our panel, we have designed a Turing machine to perform the AND function. We will set up the two input bits (each a 1 or a 0) in adjacent squares, followed by a question mark symbol, which is to be replaced by the answer (again a 1 or a 0, depending on the two inputs). For decorum, we have added an asterisk symbol at either end of the data area, and will start the machine going in state S_A on the left-most asterisk, finishing on the right-hand one.

A total of ten quintuples are needed to specify this machine, though as you can see from the worked example (1 AND 1 = 1), only five are used for any run. If you try the same machine out for, say, 0 AND 1, you will find that a different set of quintuples will be selected from the ten.