The Theory of Computation

Bernard M.E. Moret

Addison-Wesley-Longman, 1998

Solution Manual

Chapter 4, Problem 5.10

The transition function of such a machine is almost identical to that of normal Turing machine: the only difference is that the head movement must be specified as one of four directions rather than just as one of two. We do need some conventions for input and output, however; we assume that each takes the form of a string stored "horizontally" on the tape (on the same row, with the head stationed on the first (leftmost) square used by the string.

To simulate such a machine on a conventional Turing machine. we need to develop an encoding for the 2-dimensional tape. There are many ways to do this: we can use pairing, we can use prime power encoding, and many others. All will require that our simulating Turing machine, before simulating a single step of the 2-dimensional one, do some decoding of the encoded 2-dimensional tape. The one step is then simulated, then the results re-encoded and rewritten on the tape. Thus the simulation will run much more slowly than the original machine.

For instance, if we use a pairing function, to simulate a step where the 2D machine moves up on its grid, we need to compute the index on our 1D tape of the new pair $\langle x, y+1\rangle$, then move right until this index is found, which may entail traversing a large fraction of the 1D tape several times, especially since we have to use some of that tape to carry out the calculations and since seeking an index is not that simple in a conventional Turing machine (easiest may be pushing a marker forward, coming back every time to decrement the index, an extremely costly way of doing the work).

Prime power encoding may be better: we encode a column of the 2D tape as the number $2^{i_1}3^{i_2}5^{i_3}\cdots$, where i_1, i_2, i_3 are the values in the range of rows from the first to the last nonzero entry in the column. Our 1D tape is then a list of these codes. Yet everything must be decoded and later re-encoded, which is quite time-consuming, especially as it must be done in some blank area of the tape, so that the value must first be copied beyond one end of the currently used tape.

Another, simpler solution is to appeal to Theorem 4.1 and simulate our 2-dimensional TM with a *k*-tape TM rather than just a single-tape machine. One approach is to consider at all times the smallest rectangle on the 3-dimensional tape that contains every square examined so far by the 2D machine. Consider this rectangle as an array of 1-dimensional rows; we just use one tape of a 2-tape TM to store this rectangle, row by row, while the second tape stores the current dimensions of the rectangle and the third tape is used for auxiliary computations. A move of the 2D machine that stays within the current rectangle is either a horizontal move, which is simply a move left or right by one square on the first tape of our 2-tape machine, or a vertical move, which corresponds to a *m* moves left or right on the first tape, where *m* is the horizontal size of the rectangle (stored on the second tape) and the counting is done on the third tape. If the move of the 2D machine takes it outside the current rectangle, we extend the rectangle. For a vertical move, this consists simply of adding a row of blanks either in front of or in back of the storage on the first tape, plus incrementing the number of rows on the second tape. For a horizontal move, every row must get a blank added in front or back of it, which we accomplish by our standard shifting mechanism, one at a time (again, we use the third tape for counting off rows), then we increase the number of columns on the second tape.