The Theory of Computation

Bernard M.E. Moret

Addison-Wesley-Longman, 1998

Solution Manual
Chapter 3, Problem 3.31

There is no particular difficulty in solving this problem other perhaps than notation. The solution follows the approach given in the text: we build a machine with states denoted by tuples that encode information about various guesses and states of simulated runs of the original machine. If we want to accept the ith out of k pieces of equal length from strings of L (the original language), we shall have to reconstitute (guess) the first $i-1$ pieces as well as the last $k-i$ pieces and process them concurrently—even though the machine for L would have processed them and the piece given in the input in serial order. Thus we need to guess, not just the $k-1$ missing pieces, but also the state in which the machine for L would have found itself as it started processing each piece. We then need to verify that the pieces can be "stitched together" properly, i.e., that we can end the processing of each piece (guessed or not) in the state used to start the processing of the next piece. Thus, given a DFA M for the original language L, we construct an NFA M' for our language as follows: except for a new start state, each state of M' will be represented as a $(2k-1)$-tuple, encoding k current states (one for each piece) and $k-1$ guessed initial processing states (one for every piece except the first, since that piece is processed from the start state); denote such a state

$$(q_1, g_2, q_2, g_3, q_3, \ldots, g_k, q_k)$$

where q_i and g_j are states of M and q_i denotes the current state in the processing of the i piece and g_i denotes the guessed initial processing state for the ith piece. The new start state, call it S', has ε-transitions to all states of the form

$$(S, g_2, g_2, g_3, g_3, \ldots, g_k, g_k)$$

where S is the start state of M; and accepting states are any states of the form

$$(g_2, g_2, g_3, g_3, g_4, \ldots, g_k, f)$$

where f is any accepting state of M. Under input a, if machine M in state q_i has transition $\delta(q_i, a) = q_i'$, then our new machine can make a transition from state

$$(q_1, g_2, q_2, , \ldots , g_i, q_i, \ldots , g_k, q_k)$$

to any state of the form

$$(q_1', g_2, q_2', , \ldots , g_i, q_i', \ldots , g_k, q_k')$$

where for each state q_j $(j \neq i)$ in the tuple, there exists some alphabet character c_j such that $\delta(q_j, c_j) = q_j'$.

Thus machine M' guesses all missing $k - 1$ pieces (all of length equal to the input piece), guesses the state which M would have reached when beginning to process the jth piece for each j, and verifies at the end that its guesses are all consistent—that the k pieces fit together and that each simulation of M ends in the same state that was guessed as the starting point for the simulation of the next piece. The checking prevents M' from accepting strings not in our target language; the guessing allows it to accept any string in our target language.