The Theory of Computation

Bernard M.E. Moret

Addison-Wesley-Longman, 1998

Solution Manual
Chapter 7, Problem 7.37

That *Digraph Reachability* is in NL follows from the following nondeterministic solution algorithm: start at vertex 1, with counter k set to 0; at each step, guess the next vertex on the solution path, increment the counter k, and verify that there exists an arc from the current vertex to the next vertex. Stop and accept as soon as the destination vertex is reached; stop and reject as soon as the counter reaches n or no arc is found between the current vertex and the guessed vertex. The storage requires is the current vertex, the guess vertex, and the counter; each takes logarithmic space.

To show that *Digraph Reachability* is NL-complete, we reduce a generic problem in NL to it, using much the same style of reduction as of *PSA*. We shall represent the state of the NL machine by full IDs consisting, as in the proof of Theorem 6.8, of a k-tuple. (But this time we can afford full IDs, because the machine we are encoding uses only logarithmic space.) Since an NL machine has three tapes, we need to indicate the head position on both tapes and the contents of the scratch tape. (The output tape is irrelevant in a language acceptor, since the output is either 0 or 1; and the contents of the input tape is just x, unchanging, so we do not encode it—our simulation can always recover the value of a particular tape square by reading the input from tape square 1 to the desired tape square.) Thus an ID will contain the current control state, the current step number, the current head position on each tape, and the current contents of each square on the scratch tape. The current control state and the current contents of one tape square require $O(1)$ space only; the current head position on the input tape requires $O(\log n)$ space and that on the scratch tape $O(\log \log n)$ space; the step counter may have to count a polynomial number of steps and thus also takes logarithmic space; and finally, we need to encode a logarithmic number of tape squares on the scratch tape since we are dealing with an NL machine. Thus a full ID of the NL machine takes logarithmic space. We can thus generate all full IDs (the nodes of the graph) using just logarithmic space (we just need suitably nested loops) and then generate any arcs as needed on the fly by testing the two IDs for the possibility of a single transition (at which time we need to consult the input tape to extract the tape symbol under the head—a rather slow process, since we can only do it by reading from the start of the input to the desired symbol, but one that does not use storage beyond what is needed for a counter). If the NL machine has multiple accepting configurations, we add a single target node to the graph and add an (implicit) arc from each accepting configuration to the target node.