The Theory of Computation

Bernard M.E. Moret

Addison-Wesley-Longman, 1998

Solution Manual
Chapter 6, Problem 6.11

Transitivity is trivial for polynomial-time reductions, but more difficult for logspace reductions, simply because there is no easy way to store the intermediate instance. If A reduces to B and B reduces to C, the output of the first reduction serves as input to the second in the compound reduction from A to C—but we cannot afford to store more than a few bits of that output, since now that is only an intermediate result and so must find place on our scratch tape.

On the other hand, we have no direct bound on time, so we can trade time for space as follows. We run the second reduction normally, except that it does not have an input tape; instead, whenever it needs to read a bit of its input, it calls upon (a slightly modified version of) the first reduction to generate that bit of its output. In order to identify the needed bit, the second reduction keeps a counter which identifies the position of the read-only head it would have on its input tape and passes that counter to the (modified) first reduction. The (modified) first reduction generates the bit at that position on its (non-existent) output tape by restarting from scratch every time, using its input tape, but not writing any output—instead simply counting how many bits it would have produced had it been writing, stopping when the proper count is reached, and delivering that last bit to the second reduction. The first reduction runs in logarithmic space by hypothesis and so the modified version never exceeds that bound whenever called upon to produce a bit of its output. Moreover, the size of its output is at most polynomial, since it cannot run for more than polynomial time within its logspace bound. The second reduction also runs in logarithmic space on any input; however, it now runs on an intermediate result—the input to the first reduction is an instance of A of size n, but the input to the second reduction is an instance of B is size $O(p(n))$ for some polynomial $p()$. Fortunately, $\log p(n)$ is $O(\log n)$, so that our second reduction still runs in logarithmic space as measured against the instance of A. Thus the total process never uses more than logarithmic space and we are done.