The Theory of Computation

Bernard M.E. Moret

Addison-Wesley-Longman, 1998

Solution Manual

Chapter 4, Problem 5.12

We need to charge more than unit time for some of the operations. The culprit, from what we have seen, is not the addition (which can only increase storage by one bit at a time, at least in asymptotic terms) nor the register transfer (which cannot increase storage at all, again in asymptotic terms), but the indirect addressing. In fact, if indirect addressing can be used with addition, we have enough power to recreate the situation of Exercise 4.11: it is enough to have the loop carry out the instructions

```
R[R1] <- R[R2] + R[R2]
inc R2
inc R1
```

With `R1` exactly one larger than `R2`.

Thus we need a charging policy that penalizes the use of indirect addressing heavily enough to bring space use to a level commensurate with time use. The easiest way to do this is specify that the cost of indirect addressing is directly proportional to the space used in the register that is addressed (not the index register itself, but the register addressed through the index register). It seems somewhat of an arbitrary decision; more importantly, it is not always intuitive: for instance, why should it cost constant time to increment a named register, but logarithmic time to do the same on an indirectly addressed register?

Another solution is to impose a policy similar to that used for Turing machines: every bit costs, so we make the cost of every instruction be proportional to the length of its arguments and results (if any). This, of course, returns us to our earlier discussion and the apparently exaggerated cost of basic arithmetic in such a model.