mac.1 Combining Turing Machines

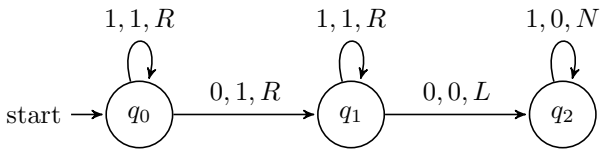
tur:mac:cmb:sec

The examples of Turing machines we have seen so far have been fairly simple in nature. But in fact, any problem that can be solved with any modern programming language can als o be solved with Turing machines. To build more complex Turing machines, it is important to convince ourselves that we can combine them, so we can build machines to solve more complex problems by breaking the procedure into simpler parts. If we can find a natural way to break a complex problem down into constituent parts, we can tackle the problem in several stages, creating several simple Turing machines and combining then into one machine that can solve the problem. This point is especially important when tackling the Halting Problem in the next section.

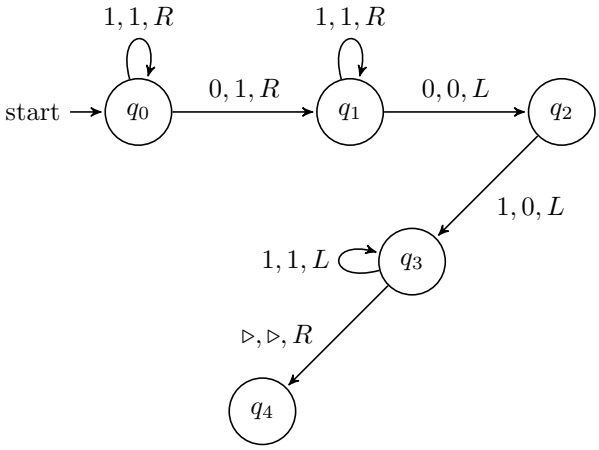
explanation

Example mac.1. *Combining Machines:* Design a machine that computes the function $f(m, n) = 2(m + n)$.

In order to build this machine, we can combine two machines we are already familiar with: the addition machine, and the doubler. We begin by drawing a state diagram for the addition machine.



Instead of halting at state q_2, we want to continue operation in order to double the output. Recall that the doubler machine erases the first stroke in the input and writes two strokes in a separate output. Let's add an instruction to make sure the tape head is reading the first stroke of the output of the addition machine.



It is now easy to double the input—all we have to do is connect the doubler machine onto state q_4. This requires renaming the states of the doubler machine

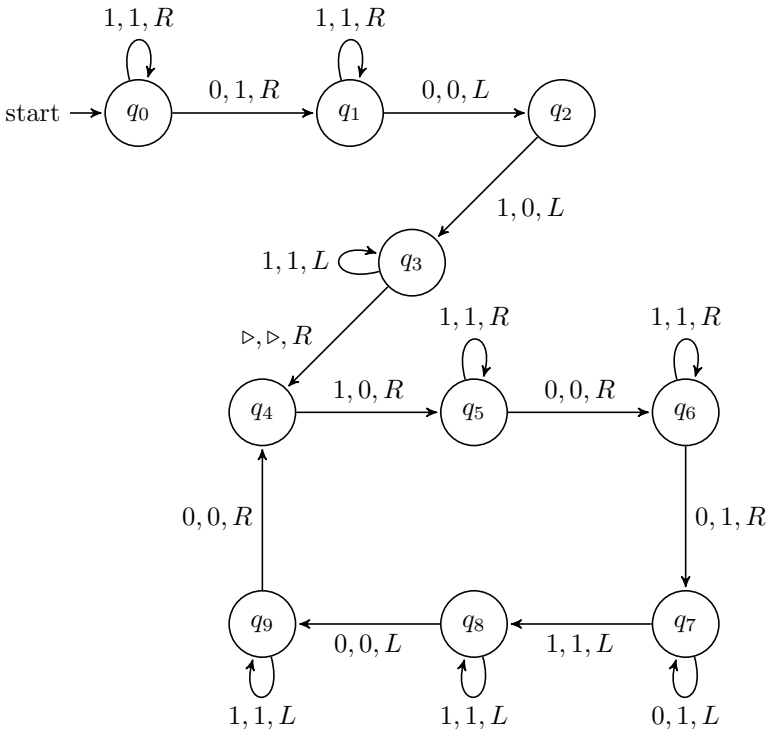


Figure 1: Combining adder and doubler machines

tur:mac:cmb:fig:combined

so that they start at q_4 instead of q_0—this way we don't end up with two starting states. The final diagram should look as in Figure 1.

Photo Credits

Bibliography