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 also 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 them into one machine that can solve the problem. This point is especially important when tackling the Halting Problem in the next section. explanation

How do we combine Turing machines $M = \langle Q, \Sigma, q_0, \delta\rangle$ and $M' = \langle Q', \Sigma', q_0', \delta'\rangle$? We now use the configuration of the tape after M has halted as the input configuration of a run of machine M'. To get a single Turing machine $M \frown M'$ that does this, do the following:

1. Renumber (or relabel) all the states Q' of M' so that M and M' have no states in common ($Q \cap Q' = \emptyset$).
2. The states of $M \frown M'$ are $Q \cup Q'$.
3. The tape alphabet is $\Sigma \cup \Sigma'$.
4. The start state is q_0.
5. The transition function is the function δ'' given by:

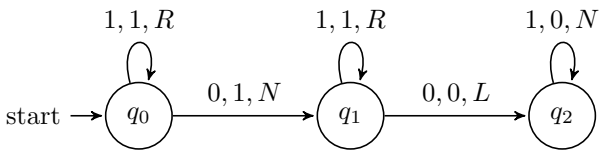
$$\delta''(q,\sigma) = \begin{cases} \delta(q,\sigma) & \text{if } q \in Q \\ \delta'(q,\sigma) & \text{if } q \in Q' \\ \langle q_0', \sigma, N\rangle & \text{if } q \in Q \text{ and } \delta(q,\sigma) \text{ is undefined} \end{cases}$$

The resulting machine uses the instructions of M when it is in a state $q \in Q$, the instructions of M' when it is in a state $q \in Q'$. When it is in a state $q \in Q$ and is scanning a symbol σ for which M has no transition (i.e., M would have halted), it enters the start state of M' (and leaves the tape contents and head position as it is).

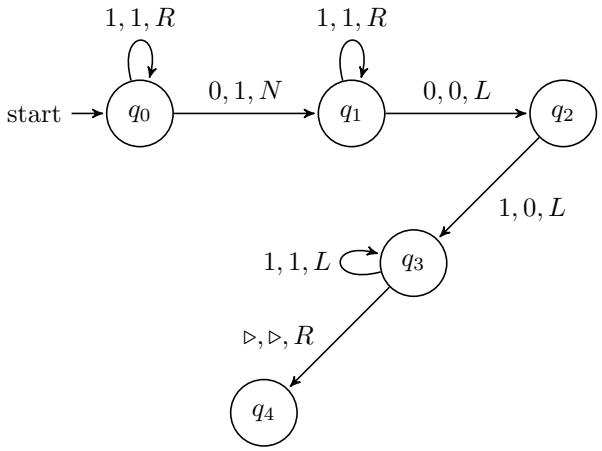
Note that unless the machine M is disciplined, we don't know where the tape head is when M halts, so the halting configuration of M need not have the head scanning square 1. When combining machines, it's important to keep this in mind.

Example mac.1. *Combining Machines:* We'll design a machine which, when started on input consisting of two blocks of 1's of length n and m, halts with a single block of $2(m+n)$ 1's on the tape. 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 in 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 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.

Proposition mac.2. *If M and M' are disciplined and compute the functions $f\colon \mathbb{N}^k \to \mathbb{N}$ and $f'\colon \mathbb{N} \to \mathbb{N}$, respectively, then $M \frown M'$ is disciplined and computes $f' \circ f$.*

Proof. Since M is disciplined, when it halts with output $f(n_1, \dots, n_k) = m$, the head is scanning square 1. If we now enter the start state of M', the machine will halt with output $f'(m)$, again scanning square 1. The other conditions of **??** are also satisfied. □

Problem mac.1. Give a disciplined Turing machine computing $f(x) = x + 2$ by taking the machine M from **??** and construct $M \frown M$.

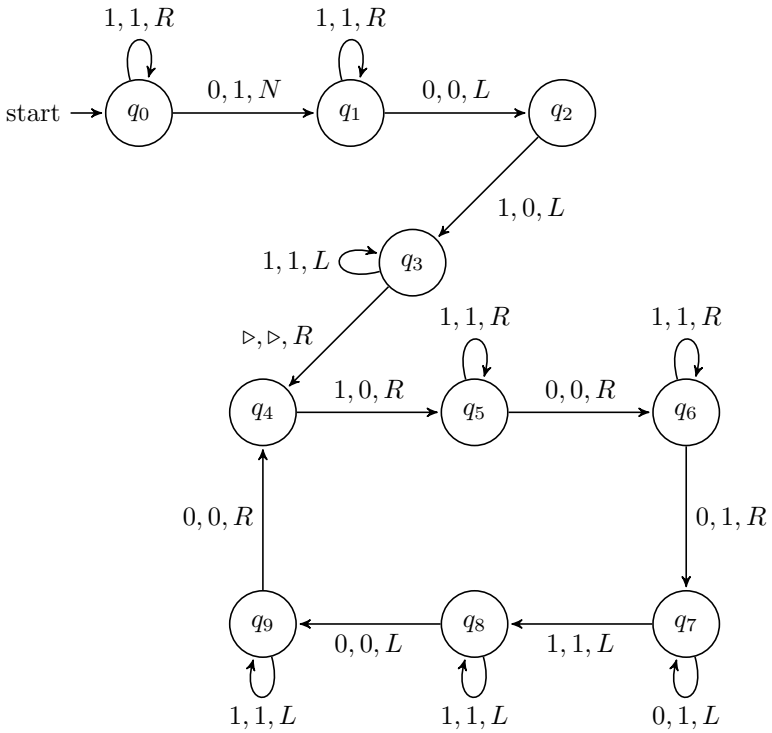


Figure 1: Combining adder and doubler machines

tur:mac:cmb:
fig:combined

Photo Credits

Bibliography