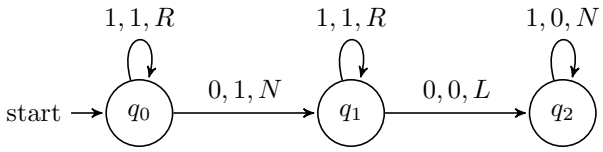

Figure 1: A machine computing $f(x, y) = x + y$

tur:mac:una:fig:adder

mac.1 Unary Representation of Numbers

tur:mac:una:sec

explanation

Turing machines work on sequences of symbols written on their tape. Depending on the alphabet a Turing machine uses, these sequences of symbols can represent various inputs and outputs. Of particular interest, of course, are Turing machines which compute *arithmetical* functions, i.e., functions of natural numbers. A simple way to represent positive integers is by coding them as sequences of a single symbol 1. If $n \in \mathbb{N}$, let 1^n be the empty sequence if $n = 0$, and otherwise the sequence consisting of exactly n 1's.

Definition mac.1 (Computation). A Turing machine M *computes* the function $f\colon \mathbb{N}^k \to \mathbb{N}$ iff M halts on input

$$1^{n_1}01^{n_2}0\ldots01^{n_k}$$

with output $1^{f(n_1,\ldots,n_k)}$.

Problem mac.1. Give a definition for when a Turing machine M computes the function $f\colon \mathbb{N}^k \to \mathbb{N}^m$.

tur:mac:una:ex:adder

Example mac.2. *Addition:* Let's build a machine that computes the function $f(n, m) = n + m$. This requires a machine that starts with two blocks of 1's of length n and m on the tape, and halts with one block consisting of $n + m$ 1's. The two input blocks of 1's are separated by a 0, so one method would be to write a stroke on the square containing the 0, and erase the last 1.

Problem mac.2. Trace through the configurations of the machine from Example mac.2 for input $\langle 3, 2\rangle$. What happens if the machine computes $0 + 0$?

explanation

In **??**, we gave an example of a Turing machine that takes as input a sequence of 1's and halts with a sequence of twice as many 1's on the tape—the doubler machine. However, because the output contains 0's to the left of the doubled block of 1's, it does not actually compute the function $f(x) = 2x$, as you might have assumed. We'll describe two ways of fixing that.

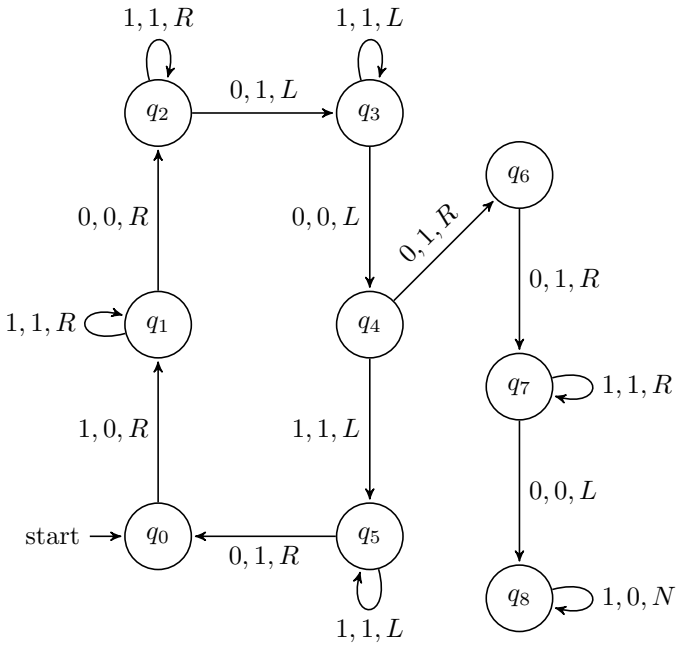

Figure 2: A machine computing $f(x) = 2x$

tur:mac:una:
fig:doubler-disc

Example mac.3. The machine in Figure 2 computes the function $f(x) = 2x$. Instead of erasing the input and writing two 1's at the far right for every 1 in the input as the machine from **??** does, this machine adds a single 1 to the right for every 1 in the input. It has to keep track of where the input ends, so it leaves a 0 between the input and the added strokes, which it fills with a 1 at the very end. And we have to "remember" where we are in the input, so we temporarily replace a 1 in the input block by a 0.

tur:mac:una:
ex:mover

Example mac.4. A second possibility for computing $f(x) = 2x$ is to keep the original doubler machine, but add states and instructions at the end which move the doubled block of strokes to the far left of the tape. The machine in Figure 3 does just this last part: started on a tape consisting of a block of 0's followed by a block of 1's (and the head positioned anywhere in the block of 0's), it erases the 1's one at a time and writes them at the beginning of the tape. In order to be able to tell when it is done, it first marks the end of the block of 1's with a ▷ symbol, which gets deleted at the end. We've started numbering the states at q_6, so they can be added to the doubler machine. All you'll need is an additional instruction $\delta(q_5, 0) = \langle q_6, 0, N\rangle$, i.e., an arrow from q_5 to q_6 labelled $0, 0, N$. (There is one subtle problem: the resulting machine does not work for input $x = 0$. We'll leave this as an exercise.)

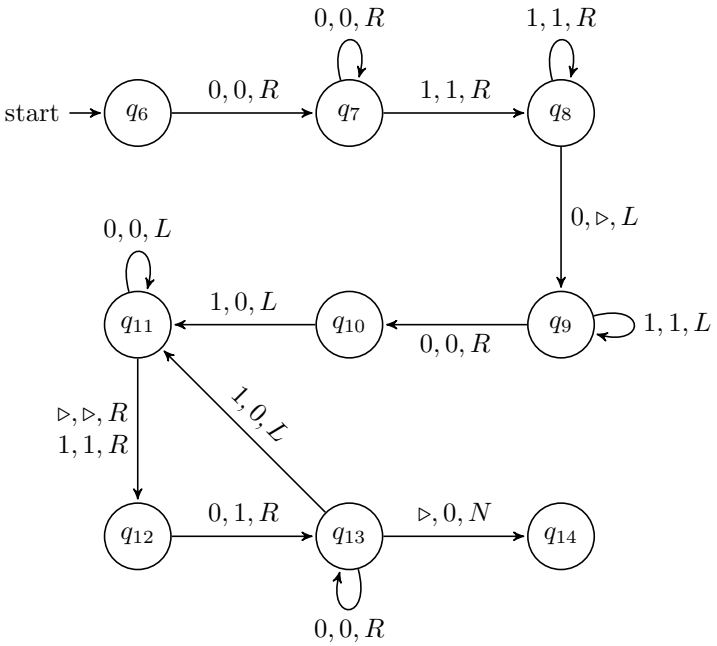

Figure 3: Moving a block of 1's to the left

tur:mac:una:fig:mover

Problem mac.3. In Example mac.4 we described a machine consisting of a combination of the doubler machine from Figure 2 and the mover machine from Figure 3. What happens if you start this combined machine on input $x = 0$, i.e., on an empty tape? How would you fix the machine so that in this case the machine halts with output $2x = 0$? (You should be able to do this by adding one state and one transition.)

Problem mac.4. *Subtraction:* Design a Turing machine that when given an input of two non-empty strings of strokes of length n and m, where $n > m$, computes the function $f(n, m) = n - m$.

Problem mac.5. *Equality:* Design a Turing machine to compute the following function:

$$\text{equality}(n, m) = \begin{cases} 1 & \text{if } n = m \\ 0 & \text{if } n \neq m \end{cases}$$

where n and $m \in \mathbb{Z}^+$.

Problem mac.6. Design a Turing machine to compute the function $\min(x, y)$ where x and y are positive integers represented on the tape by strings of 1's

separated by a 0. You may use additional symbols in the alphabet of the machine.

The function min selects the smallest value from its arguments, so $\min(3, 5) = 3$, $\min(20, 16) = 16$, and $\min(4, 4) = 4$, and so on.

Definition mac.5. A Turing machine M computes the partial function $f\colon \mathbb{N}^k \nrightarrow \mathbb{N}$ iff,

1. M halts on input $1^{n_1} \frown 0 \frown \ldots \frown 0 \frown 1^{n_k}$ with output 1^m if $f(n_1, \ldots, n_k) = m$.
2. M does not halt at all, or with an output that is not a single block of 1's if $f(n_1, \ldots, n_k)$ is undefined.

Photo Credits

Bibliography