mac.1 Unary Representation of Numbers

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 \( f : \mathbb{N}^n \rightarrow \mathbb{N} \) iff \( M \) halts on input

\[
1^{k_1}01^{k_2}0 \ldots 01^{k_n}
\]

with output \( f(k_1, \ldots, k_n) \).

**Example mac.2.** *Addition:* Build a machine that, when given an input of two non-empty strings of 1’s of length \( n \) and \( m \), computes the function \( f(n, m) = n + m \).

We want to come up with a machine that starts with two blocks of strokes on the tape and halts with one block of strokes. We first need a method to carry out. The input strokes are separated by a blank, so one method would be to write a stroke on the square containing the blank, and erase the first (or last) stroke. This would result in a block of \( n + m \) 1’s. Alternatively, we could proceed in a similar way to the doubler machine, by erasing a stroke from the first block, and adding one to the second block of strokes until the first block has been removed completely. We will proceed with the former example.

**Problem mac.1.** Trace through the configurations of the machine for input \( \langle 3, 5 \rangle \).

**Problem mac.2.** *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.3.** *Equality:* Design a Turing machine to compute the following function:

\[
\text{equality}(x, y) = \begin{cases} 
1 & \text{if } x = y \\
0 & \text{if } x \neq y 
\end{cases}
\]

where \( x \) and \( y \) are integers greater than 0.
Problem mac.4. 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.

Photo Credits

Bibliography