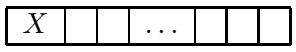
Turing Machines and Binary Addition

An Historical Project

The logic behind the modern programmable computer owes much to Turing's "computing machines," discussed in the first project, which the reader should review. Since the state of the machine, or m-configuration as called by Turing, can be altered according to the symbol being scanned, the operation of the machine can be changed depending on what symbols have been written on the tape, and affords the machine a degree of programmability. The program consists of the list of configurations of the machine and its behavior for each configuration. Turing's description of his machine, however, did not include memory in its modern usage for computers, and symbols read on the tape could not be stored in any separate devise. Using a brilliant design feature for the tape, Turing achieves a limited type of memory for the machine, which allows it to compute many arithmetic operations. The numbers needed for a calculation are printed on every other square of the tape, while the squares between these are used as "rough notes to 'assist the memory.' It will only be these rough notes which will be liable to erasure" [1, p. 232].

Turing continues: The convention of writing the figures only on alternate squares is very useful: I shall always make use of it. I shall call the one sequence of alternate squares F-squares, and the other sequence E-squares. The symbols on E-squares will be liable to erasure. The symbols on F-squares form a continuous sequence. ... There is no need to have more than one E-square between each pair of F-squares: an apparent need of more E-squares can be satisfied by having a sufficiently rich variety of symbols capable of being printed on E-squares [1, p. 235].

Let's examine the Englishman's use of these two types of squares. Determine the output of the following Turing machine, which begins with the tape



and the scanner at the far left, reading the symbol X.

Configuration		Behavior	
m-config.	symbol	operation	final m-config.
a	X	R	a
a	1	R, R	a
a	blank	P(1), R, R, P(1), R, R, P(0)	b
b	X	E, R	c
b	other	L	b
c	0	R, P(X), R	a
c	1	R, P(X), R	d
d	0	R, R	e
d	other	R, R	d
e	blank	P(1)	b
e	other	R, R	e

Recall the meaning of the following symbols used for operations.

- R: Move one position to the right.
- L: Move one position to the left.
- E: Erase the currently scanned square
- P( ): Print whatever is in parentheses in the current square.

(a) What is the precise output of the machine as it just finishes configuration a and enters configuration b for the first time? Justify your answer.

(b) What is the precise output of the machine as it just finishes configuration a and enters configuration b for the second time? Justify your answer.

(c) What is the precise output of the machine as it just finishes configuration a and enters configuration b for the third time? Justify your answer.

(d) Guess what the output of the machine is as it just finishes configuration a and enters configuration b for the n-th time. Use induction to prove that your guess is correct. Be sure to carefully write the details of this proof by induction.

(e) Design a Turing machine that computes the sum of two positive integers in base two. First consider the examples of binary addition afforded by

$$111 + 101 \quad \text{and} \quad 110 + 101\,.$$

Explain how each of these sums is computed by hand, without the use of a Turing machine. Pay careful attention to the operation of carrying.

Now consider positive integers A and B written in base 2 with digits

$$A = a_n\, a_{n-1}\, a_{n-2}\, \ldots\, a_2\, a_1\, a_0$$
$$B = b_n\ b_{n-1}\ b_{n-2}\, \ldots\, b_2\ b_1\ b_0\,.$$

Each $a_i \in \{0,\ 1\}$, each $b_i \in \{0,\ 1\}$, and the subscript n is the same for both A and B. Suppose that * appears on a tape followed by the a_i's on every other square, followed by &, followed by the b_i's on every other square, and terminated by #, as follows:

*		a_n		$\ldots$		a_1		a_0		&		b_n		$\ldots$		b_1		b_0		#

Design a Turing machine T that computes $A + B$ in base 2. Suppose that T begins reading the above tape at the far right, scanning the symbol # . Let d_{n+1}, d_n, ..., d_1, d_0 be the digits of $A + B$. The machine should place the d_i's to the left of * with final output

												%		d_{n+1}		d_n		$\ldots$		d_2		d_1		d_0	
*		a_n		$\ldots$		a_1		a_0		&		b_n		$\ldots$		b_1		b_0		#		

Before writing T, you may wish to consider the branching in logic necessary to account for the cases $a_i = 0$, 1, $b_i = 0$, 1, and whether there is a carried digit. Such branching can even be represented schematically by a binary tree. To realize the effect of a carried digit, one method "to remember" that a 1 needs to be carried after adding a_i and b_i is to print a "C" immediately to the left of a_i, or immediately to the left of d_i. To indicate that a 1 need not be carried, print "N" instead. Finally, design machine configurations with a specific purpose such as printing 1 in the space of d_i or printing 0 there. Numbering configurations may also be more efficient than lettering them. Briefly, state how the procedure for computing $A + B$ is recursive.

Extra Credit: Explain what difficulties are encountered when designing a Turing machine to perform the addition $A + B$ in base 10.

REFERENCES:

[1] Turing, A. M., "On Computable Numbers with An Application to the Entscheidungsproblem," *Proceedings of the London Mathematical Society*, 42, 1936, pp. 230–265.