Turing Machines

Thomas Wieting
Reed College, 2011

1 Verify/Prove

$1^\bullet$ $a^2 + b^2 = c^2$

$2^\bullet$ $2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, \ldots$

$3^\bullet$ $20216 = 2{\cdot}10108 = 2{\cdot}2{\cdot}5054 = 2{\cdot}2{\cdot}2{\cdot}2527 = 2{\cdot}2{\cdot}2{\cdot}7{\cdot}361 = 2{\cdot}2{\cdot}2{\cdot}7{\cdot}19{\cdot}19$
$= 2^3 3^0 5^0 7^1 11^0 13^0 17^0 19^2$

$4^\bullet$ $\sqrt{2}$

2 Turing Machines

5° $\mathbf{N},\ \mathbf{N} \times \mathbf{N},\ \mathbf{N} \times \mathbf{N} \times \mathbf{N},\ \ldots$

6° We define Turing Machines by means of Tables. The tables are Programs of Instructions. For examples:

(Z)

	$\circ$	$\bullet$
S_1	$\bullet S_2 R$	$\circ S_1 R$
S_2	$\circ S_2 L$	

(Σ)

	$\circ$	$\bullet$
S_1	$\bullet S_2 L$	$\bullet S_1 L$
S_2	$\circ S_2 R$	

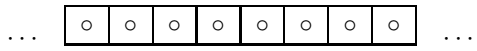
and:

$(P_{2,3})$

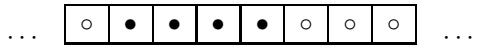
	$\circ$	$\bullet$
S_1	$\circ S_2 R$	$\circ S_1 R$
S_2	$\circ S_3 R$	$\bullet S_2 R$
S_3	$\circ S_4 L$	$\circ S_3 R$
S_4	$\circ S_4 L$	$\bullet S_5 L$
S_5	$\circ S_6 R$	$\bullet S_5 L$
S_6		

For each of these machines, the Symbols are $\circ$ and $\bullet$. Let us call them the Blank and the Mark. For the first two machines, the states are S_1 and S_2; for the third machine, S_1, S_2, S_3, S_4, S_5, and S_6. Let us call S_1 the Start State.

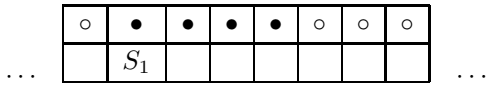
7° Turing Machines compute Functions. Let us describe, by examples, how they do it. We begin with the simple case of the Turing Machine Z. First, we introduce a Tape comprised of cells, extending indefinitely to the left and to the right. We fill the cells with blanks:



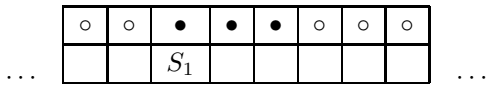
Second, we introduce a Nonnegative Integer x. For instance, let x be 3. We represent x on the tape by substituting $x + 1$ successive marks for blanks:



Third, we set the machine in the start state S_1. Fourth, we roll the machine into position so that it can Read the first of the marks:

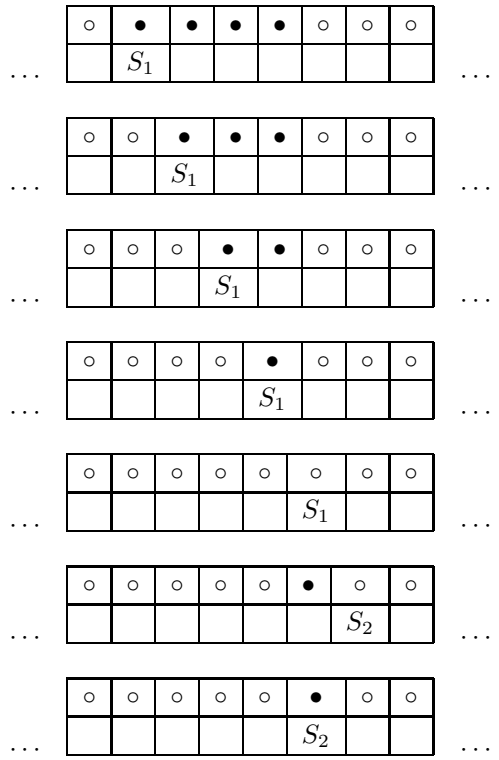


Now the Turing Machine Z is ready to compute. The machine stands in state S_1 and it reads the symbol $\bullet$. By its program of instructions, it (erases the $\bullet$ and) prints the symbol $\circ$; remains in state S_1; and moves one cell to the Right:



Again, the machine stands in state S_1 and it reads the symbol $\bullet$. Of course, it will repeat the preceding maneuvers. In fact, it will continue to do so until it stands in state S_1 and it reads the symbol $\circ$. By its program of instructions,

it then (erases the ∘ and) prints the symbol •; enters state S_2; and moves one cell to the right. Now the Turing Machine stands in state S_2 and it reads the symbol ∘. By its program of instructions, it leaves the symbol ∘ in place; remains in state S_2; and moves one cell to the Left. At this point, the machine stands in state S_2 and it reads the symbol •. By its program of instructions, it Halts, because the relevant place in the table is empty. Let us display the entire computation:



In the foregoing computation, the Turing Machine Z acted upon the input $x = 3$ to produce the output 0. In fact, for any nonnegative integer x, Z will act upon the input x to produce the output 0:

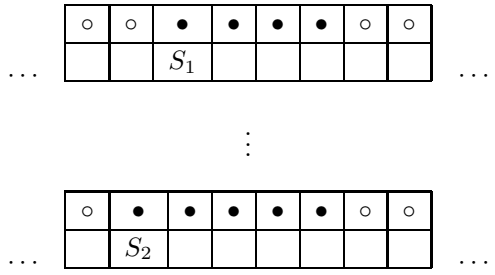
$$Z(x) = 0$$

We refer to Z as the Zero Turing Machine. It computes the Zero Function, which we denote by the same symbol Z.

8• Show that, for any nonnegative integer x, the Turing Machine Σ will act upon the input x to produce the output $x + 1$:

$$\Sigma(x) = x + 1$$

For $x = 3$, the first and last steps of the computation are the following:



We refer to Σ as the Successor Turing Machine. It computes the Successor Function Σ.

9° With reference to the foregoing computations, let us refer to the successive steps as Configurations. Obviously, each configuration displays both the current state of the machine and the symbol under review. Moreover, the Initial and Terminal Configurations have special forms. For the initial configuration, the state of the machine is the start state S_1 and the symbol under review is a mark, the first of a finite sequence of marks embedded in a stream of blanks. For the terminal configuration, the state of the machine is one of the states prescribed; the symbol under review is a mark, the first of a finite sequence of marks embedded in a stream of blanks; and the relevant place in the table is empty. One can observe the input and the output explicitly in these special configurations.

10° Now let T be any Turing Machine. The symbols are the blank $\circ$ and the mark $\bullet$ but the states may be legion:

$$S_1, S_2, S_3, \cdots, S_n$$

where n is any positive integer. Let x be any nonnegative integer. Let T begin work in the corresponding initial configuration. Of course, T may halt in a terminal configuration; it may halt in a configuration other than a terminal configuration; or it may never halt. In the first of these cases, we declare that x is a member of the Domain of T and we denote by:

$$T(x)$$

the nonnegative integer corresponding to the terminal configuration. In the second and third of these cases, we declare that x is not a member of the domain of T. For each x in the domain of T, we refer to $T(x)$ as the Value assigned to x by T. We may say that the Turing Machine T computes the Function T.

11° For later reference, let us denote the domain of T by:

$$dom_1(T)$$

We apply the subscript 1 to emphasize that the function T depends on just one variable x. See articles 15° and 16°.

12° Obviously, for the functions Z and Σ:

$$dom_1(Z) = \mathbf{N} = dom_1(\Sigma)$$

We say that such functions are Total Functions.

13• Show that the following Turing Machine never halts:

(Ω)

	○	•
S_1	$\circ S_1 R$	$\bullet S_1 R$

14° Turing Machines compute not only Functions of one variable but also Functions of several variables. Let us describe, by examples, how they do it. We begin with the simple case of the Turing Machine $P_{(2,3)}$. Let x, y, and z be any nonnegative integers. For instance, let x be 4, let y be 2, and z be 3. We set the initial configuration as follows:

○	•	•	•	•	•	○	•	•	•	○	•	•	•	•	○
	S_1														

We apply the program of instructions for $P_{(2,3)}$, step by step, to obtain the terminal configuration:

○	○	○	○	○	○	○	•	•	•	○	○	○	○	○	○
							S_6								

Hence:

$$P_{(2,3)}(4, 2, 3) = 2$$

In fact, for any nonnegative integers x, y, and z, $P_{(2,3)}$ will act upon the input (x, y, z) to produce the output y:

$$P_{(2,3)}(x, y, z) = y$$

We refer to $P_{(2,3)}$ as the $(2,3)$-Projection Turing Machine. It computes the $(2,3)$-Projection Function $P_{(2,3)}$.

15° Again let T be any Turing Machine. Let k be any positive integer. For instance, let k be 3. Let:

$$(x, y, z)$$

be any 3-tuple of nonnegative integers. Let T begin work in the corresponding initial configuration. Of course, T may halt in a terminal configuration; it may halt in a configuration other than a terminal configuration; or it may never halt. In the first of these cases, we declare that (x, y, z) is a member of the Domain of T and we denote by:

$$T(x, y, z)$$

the nonnegative integer corresponding to the terminal configuration. In the second and third of these cases, we declare that (x, y, z) is not a member of the domain of T. For each (x, y, z) in the domain of T, we refer to $T(x, y, z)$ as the Value assigned to (x, y, z) by T. We may say that the Turing Machine T computes the Function T.

16° For later reference, let us denote the domain of T by:

$$dom_3(T)$$

We apply the subscript 3 to emphasize that the function T depends on 3 variables x, y, and z.

17° Obviously, for the function $P_{(2,3)}$:

$$dom_3(P_{(2,3)}) = \mathbf{N} \times \mathbf{N} \times \mathbf{N}$$

We say that such a function is a Total Function.

3 Problems

18• With reference to article 14°, verify the computation by $P_{(2,3)}$.

19• Design the $(1,5)$-Projection Turing Machine $P_{(1,5)}$. Of course:

$$dom_5(P_{(1,5)}) = \mathbf{N} \times \mathbf{N} \times \mathbf{N} \times \mathbf{N} \times \mathbf{N}$$

and:

$$P_{(1,5)}(a, b, x, y, z) = a$$

where (a, b, x, y, z) is any 5-tuple of nonnegative integers.

20• Explain why the following correspondence between $\mathbf{N} \times \mathbf{N}$ and $\mathbf{N}$ is bijective:

$$(x, y) \Longleftrightarrow (2x + 1)2^y$$

where x and y are any nonnegative integers.

21• Design a Turing Machine $\bar{\Sigma}$ which computes the function:

$$\bar{\Sigma}(x) = x + 2$$

where x is any nonnegative integer. To do so, apply the Turing Machine Σ twice.

22• Design a Turing Machine A which computes the Addition Function:

$$A(x, y) = x + y$$

where x and y are any nonnegative integers.

23• Design a Turing Machine S which computes the Subtraction Function:

$$S(x, y) = \begin{cases} 0 & \text{if } y < x \\ y - x & \text{if } x \leq y \end{cases}$$

where x and y are any nonnegative integers.

24• Design a Turing Machine M which computes the Multiplication Function:

$$M(x, y) = xy$$

where x and y are any nonnegative integers.

25• Design a Turing Machine E which computes the Exponentiation Function:

$$E(x, y) = x^y$$

where x and y are any nonnegative integers.

26• Design Turing Machines Q and R such that:

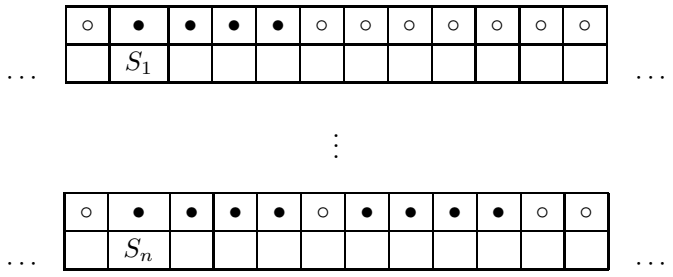
$$y = Q(x, y)x + R(x, y) \quad \text{and} \quad 0 \leq R(x, y) < x$$

where x and y are any nonnegative integers for which $0 < x$, while:

$$Q(0, y) = 0 \quad \text{and} \quad R(0, y) = y$$

where y is any nonnegative integer. We refer to Q and R as Quotient and Remainder Turing Machines. They compute the Quotient and Remainder Functions.

27• Design the Copy Turing Machine C. The following diagram suggests the purpose of this machine:



28• Design a Turing Machine P which computes the function:

$$P(x) = x^2$$

where x is any nonnegative integer. To that end, apply the Turing Machines C and M, described earlier.

29• For the Turing Machine J defined by the following table:

(J)

	$\circ$	$\bullet$
S_1	$\bullet S_3 R$	$\bullet S_2 R$
S_2	$\circ S_2 R$	$\bullet S_1 R$
S_3	$\circ S_3 L$	

describe the function J of one variable which J computes.

30• For the Turing Machine Π defined by the following table:

(Π)

	$\circ$	$\bullet$
S_1	$\bullet S_4 R$	$\circ S_2 R$
S_2	$\bullet S_3 R$	$\circ S_1 R$
S_3	$\circ S_3 L$	
S_4	$\bullet S_4 L$	

describe the function Π of one variable which Π computes. Explain how one may use this machine to "decide" whether a given nonnegative integer is odd or even.

31$^\bullet$ Design a Turing Machine Δ which computes the function:

$$\Delta(x, y) = \begin{cases} 0 & \text{if } x \neq y \\ 1 & \text{if } x = y \end{cases}$$

where x and y are any nonnegative integers. Explain how one may use the machine Δ to "decide" whether or not two given nonnegative integers are equal.

32$^\bullet$ Let T be any Turing Machine. For many purposes, it would be convenient if T carried out its computations of functions without ever moving to the left of its initial position. In fact, one can arrange for it to be so. Design a Turing Machine $\hat{T}$ such that $dom_1(\hat{T}) = dom_1(T)$; such that, for any nonnegative integer x, if x is contained in the common domain of $\hat{T}$ and T then $\hat{T}(x) = T(x)$; and such that, in the course of computation, $\hat{T}$ never moves to the left of its initial position. Do the same for the computation of functions of any number of variables.

4 Computable Functions

33° We know that the functions Z, Σ, the various $P_{(j,k)}$, A, S, M, E, Q, R, C, P, Π, and Δ are computable by Turing Machines. Of course, there are many others. In our Lecture/Discussion, we will suggest that any function which can be defined from computable functions by one of the following methods:

(1) *Composition*
(2) *Recursion*
(3) *Minimization*

is itself computable. We will describe these methods by means of examples. Conversely, we will suggest that any computable function can be defined in terms of Z, Σ, and the various $P_{(j,k)}$ by some combination of the foregoing methods.

34° In the last section, we will describe the Busy Beaver Function, which, while fully comprehensible to a rational Being, CANNOT be computed by a Turing Machine.

5 Gödel Numbers

35• Let us assign to each Turing Machine T a positive integer $|T|$. We will call $|T|$ the Gödel Number of T. We will make sure that distinct Turing Machines receive distinct Gödel Numbers. Moreover, for any positive integer y, we will make sure that we can decide by straightforward computation whether or not y is the Gödel Number of a Turing Machine and, if it is so, that we can recover from y the Program of Instructions for the machine. We represent the assignment of Gödel Numbers to Turing Machines schematically as follows:

$$T \longrightarrow y = |T|, \qquad\qquad y \longrightarrow T_y$$

Specifically, we assign Gödel Numbers to the basic symbols as follows:

$$\begin{array}{ccc} \circ & \longleftarrow & 1 \\ \bullet & \longleftarrow & 2 \\ S_j & \longleftarrow & j \\ L & \longleftarrow & 1 \\ R & \longleftarrow & 2 \end{array}$$

where j is any positive integer. We assign a Gödel Number to the Turing Machine Z as follows:

$$\begin{aligned} |Z| &= 2^0\ \ 3^2\ \ 5^2 7^2 11^2\ \ 13^1 17^1 19^2\ \ 23^1 29^2 31^1\ \ 37^0 41^0 43^0 \\ &= 63818849360591325 \end{aligned}$$

We assign a Gödel Number to the Turing Machine Σ as follows:

$$\begin{aligned} |\Sigma| &= 2^0\ \ 3^2\ \ 5^2 7^2 11^1\ \ 13^2 17^1 19^1\ \ 23^1 29^2 31^2\ \ 37^0 41^0 43^0 \\ &= 3969593500898025 \end{aligned}$$

36° We assign a Gödel Number to the Turing Machine $P_{(2,3)}$ as follows:

$$\begin{aligned} &|P_{(2,3)}| \\ &= 2^0\ \ 3^6\ \ 5^1 7^2 11^2\ \ 13^1 17^1 19^2\ \ 23^1 29^3 31^2\ \ 37^2 41^2 43^2\ \ 47^1 53^4 59^1\ \ 61^1 67^3 71^2 \ldots \end{aligned}$$

In fact:

$$\begin{aligned} &|P_{(2,3)}| \\ &= 131693726636806447719355735660220003386094388487409924790188883405142193490123529134180517307401503932120695028104 5 \end{aligned}$$

37• Explain, step by step, the General Rule for assigning Gödel Numbers to Turing Machines. For any positive integer y, explain how to decide by straightforward computation whether or not y is the Gödel Number of a Turing Machine and, if so, how to recover the Program of Instructions for the machine.

6 Universal Turing Machines

38° In 1936, A. Turing designed a Turing Machine U such that, for any pair (x, y) of nonnegative integers, (x, y) is contained in $dom_2(U)$ if and only if y is the Gödel Number of a Turing Machine T_y and x is contained in $dom_1(T_y)$, in which case:

$$T_y(x) = U(x, y)$$

We refer to U as a Universal Turing Machine, because U can compute any computable function (of one variable).

39° In general, Turing designed Universal Turing Machines which will compute any computable function of any specified number of variables. These theoretical designs inspired confidence in the development of modern Computers.

7 The Halting Problem

40° Let U be a Universal Turing Machine, as described in article 30°. Let $\Delta = dom_2(U)$. Let us imagine a Turing Machine H such that $dom_2(H) = \mathbf{N} \times \mathbf{N}$ and such that:

$$H(x, y) = \begin{cases} 0 & \text{if } (x, y) \notin \Delta \\ 1 & \text{if } (x, y) \in \Delta \end{cases}$$

where (x, y) is any pair of nonnegative integers. Given such a machine, we could decide, for any Turing Machine T and for any input x, whether or not T will halt, that is, whether or not x lies in $dom_1(T)$. We would simply compute the Gödel Number y for T, apply H to the input (x, y), and observe whether or not $H(x, y)$ equals 1. In this way, we would solve the Halting Problem.

41° A. Turing proved that no such Turing Machine H exists. To prove this remarkable result, one can argue by contradiction, as follows. Suppose that such a machine H exists. Then one could design a Turing Machine T such that $dom_1(T) = \mathbf{N}$ and such that:

$$T(x) = \begin{cases} 0 & \text{if } H(x, x) = 0 \\ U(x, x) + 1 & \text{if } H(x, x) = 1 \end{cases}$$

where x is any nonnegative integer. Let y be the Gödel Number of T. Of course, y is in $dom_1(T)$ and:

$$T(y) = U(y, y)$$

However:

$$\begin{aligned} y \in dom_1(T) &\Longrightarrow (y, y) \in dom_2(U) \\ &\Longrightarrow H(y, y) = 1 \\ &\Longrightarrow T(y) = U(y, y) + 1 \end{aligned}$$

This bald contradiction shows that such a machine cannot exist.

8 The Busy Beaver

42° Let n be any positive integer. Let us consider all Turing Machines T for which there are n states:

$$S_1, S_2, \ \ldots \ , S_n$$

and for which 0 lies in $dom_1(T)$. Let m stand for the largest of all the numbers:

$$T(0)$$

where T runs through all such machines. By definition, the number m is the value of the Busy Beaver Function B at n:

$$B(n) = m$$

Just to be complete, we define:

$$B(0) = 0$$

Let us show that B is NOT computable by a Turing Machine.

43° First, we contend that, for any positive integers m and n:

$$m + 1 < n \implies B(m) < B(n)$$

To prove the contention, we introduce a Turing Machine T such that the number of states of T is m, 0 lies in $dom_1(T)$, and $T(0) = B(m)$. Obviously, we can augment T by 2 additional states and modify the table to produce a new Turing Machine $\hat{T}$ such that 0 lies in $dom_1(\hat{T})$ and:

$$\hat{T}(0) = T(0) + 1$$

Hence:

$$B(m) = T(0) < \hat{T}(0) \leq B(m + 2) \leq B(n)$$

44° Second, we contend that there is a positive integer d such that, for any positive integer n:

$$2B(n) \leq B(n + d + 1)$$

To prove the contention, we introduce a Turing Machine D such that, for each nonnegative integer n, n lies in the $dom_1(D)$ and $D(n) = 2n$. Let d be the number of states of D. Let T be a Turing Machine such that the number of states of T is n, 0 lies in $dom_1(T)$, and $T(0) = B(n)$. Let DT be the composition of D and T. The number of states of DT is $n + d + 1$. The extra state figures in forming the composition of the actions of the machines. We find that:

$$2B(n) = 2T(0) = (DT)(0) \leq B(n + d + 1)$$

45° It follows that, for any positive integer n:

$$4B(n) \leq 2B(n + d + 1) \leq B(n + 2d + 2)$$

By induction, we find that, for any positive integer j:

$$2^j B(n) \leq B(n + jd + j)$$

46° Finally, let us SUPPOSE that there is a Turing Machine B which computes the Busy Beaver Function B. Let b be the number of states of B. Let n be any positive integer. Let $T = B^2\Sigma^n$ be the Turing Machine produced by composition of the 2-fold composition of B with itself and the n-fold composition of Σ with itself. The number of states of T is $2b + 2n + n + 1$. Obviously:

$$B(B(n)) = (B^2\Sigma^n)(0) = T(0) \leq B(2b + 3n + 1)$$

By article 43°, we infer that:

$$B(n) \leq 2b + 3n + 2$$

Indeed, if it were not so then $(2b+3n+1)+1 < B(n)$, so that $B(2b+3n+1) < B(B(n))$, which is false. By article 44°:

$$2^j B(n) \leq B(n + jd + j) \leq 2b + 3(n + jd + j) + 2$$

where j and n are any positive integers. We infer that B is the zero function, which is absurd.

47• Let n be any positive integer. Show that there are:

$$(4n + 1)^{2n}$$

distinct Turing Machines with n states.