A Study of Logic and Programming via Turing Machines

Jerry M. Lodder*

An Introduction to Turing Machines

During the International Congress of Mathematicians in Paris in 1900 David Hilbert (1862–1943), one of the leading mathematicians of the last century, proposed a list of problems for following generations to ponder [8, p. 290–329] [9]. On the list was whether the axioms of arithmetic are consistent, a question which would have profound consequences for the foundations of mathematics. Continuing in this direction, in 1928 Hilbert proposed the decision problem (das Entscheidungsproblem) [10, 11, 12], which asked whether there was a standard procedure that can be applied to decide whether a given mathematical statement is true. Both Alonzo Church (1903–1995) [2, 3] and Alan Turing (1912–1954) [13] published papers in 1936 demonstrating that the decision problem has no solution, although it is the algorithmic character of Turing's paper "On Computable Numbers, with an Application to the Entscheidungsproblem" [13] that forms the basis for the modern programmable computer. Today his construction is known as a *Turing machine*.

Let's first study a few excerpts from Turing's original paper [13, p. 231–234], and then design a few machines to perform certain tasks.

ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO
THE ENTSCHEIDUNGSPROBLEM

By A. M. Turing

1. *Computing Machines.*

> We have said that the computable numbers are those whose decimals are calculable by finite means. This requires more explicit definition. No real attempt will be made to justify the definitions given until we reach §9. For present I shall only say that the justification lies in the fact that the human memory is necessarily limited.
>
> We may compare a man in the process of computing a real number to a machine which is only capable of a finite number of conditions q_1, q_2, ..., q_R, which will be called the "m-configurations". The machine is supplied with a "tape" (the analogue of paper) running through it, and divided into sections (called "squares") each capable of bearing a "symbol". At any moment there is just one square, say the r-th, bearing the symbol $\mathcal{S}(r)$ which is "in the machine". We may call this square the "scanned square". The symbol on the scanned square may be called the "scanned symbol". The "scanned symbol" is the only one of which the machine is, so to speak, "directly aware". However, by altering its m-configuration the machine can effectively remember some of the symbols it has "seen" (scanned) previously. The possible behaviour of the machine at any moment is determined

*Mathematical Sciences; Dept. 3MB, Box 30001; New Mexico State University; Las Cruces, NM 88003; jlodder@nmsu.edu.

by the m-configuration q_n and the scanned symbol $\mathcal{S}(r)$. This pair q_n, $\mathcal{S}(r)$ will be called the "configuration"; thus the configuration determines the possible behaviour of the machine. In some of the configurations in which the scanned square is blank (i.e. bears no symbol) the machine writes down a new symbol on the scanned square; in other configurations it erases the scanned symbol. The machine may also change the square which is being scanned, but only by shifting it one place to right or left. In addition to any of these operations the m-configuration may be changed. Some of the symbols written down will form the sequence of figures which is the decimal of the real number which is being computed. The others are just rough notes to "assist the memory". It will only be these rough notes which will be liable to erasure.

It is my contention that these operations include all those which are used in the computation of a number. The defense of this contention will be easier when the theory of the machines is familiar to the reader. In the next section I therefore proceed with the development of the theory and assume that it is understood what is meant by "machine", "tape", "scanned", etc.

2. *Definitions.*

Automatic machines.

If at each stage the motion of a machine (in the sense of §1) is *completely* determined by the configuration, we shall call the machine an "automatic machine" (or a-machine).

For some purposes we might use machines (choice machines or c-machines) whose motion is only partially determined by the configuration (hence the use of the word "possible" in §1). When such a machine reaches one of these ambiguous configurations, it cannot go on until some arbitrary choice has been made by an external operator. This would be the case if we were using machines to deal with axiomatic systems. In this paper I deal only with automatic machines, and will therefore often omit the prefix a-.

Computing machines.

If an a-machine prints two kinds of symbols, of which the first kind (called figures) consists entirely of 0 and 1 (the others being called symbols of the second kind), then the machine will be called a computing machine. If the machine is supplied with a blank tape and set in motion, starting from the correct initial m-configuration the subsequence of the symbols printed by it which are of the first kind will be called the *sequence computed by the machine*. The real number whose expression as a binary decimal is obtained by prefacing this sequence by a decimal point is called the *number printed by the machine*.

At any stage of the motion of the machine, the number of the scanned square, the complete sequence of all symbols on the tape, and the m-configuration will be said to describe the *complete configuration* at that stage. The changes of the machine and tape between successive complete configurations will be called the *moves* of the machine. ...

3. *Examples of computing machines.*

I. A machine can be constructed to compute the sequence $0\,1\,0\,1\,0\,1\ldots$. The machine is to have the four m-configurations "b", "c", "f", "e" and is capable of printing "0" and "1". The behaviour of the machine is described in the following table [Example 1] in which "R" means "the machine moves so that it scans the square immediately on the right of the one it was scanning previously". Similarly for "L". "E" means "the scanned symbol is erased

and "P" stands for "prints". This table (and all succeeding tables of the same kind) is to be understood to mean that for a configuration described in the first two columns the operations in the third column are carried out successively, and the machine then goes over into the m-configuration described in the last column. When the second column is blank, it is understood that the behaviour of the third and fourth columns applies for any symbol and for no symbol. The machine starts in the m-configuration b with a blank tape.

Configuration		Behaviour	
m-config.	symbol	operation	final m-config.
b	none	P(0), R	c
c	none	R	e
e	none	P(1), R	f
f	none	R	b

If (contrary to the description §1) we allow the letters L, R to appear more than once in the operations column we can simplify the table considerably.

Configuration		Behaviour	
m-config.	symbol	operation	final m-config.
b	none	P(0)	b
b	0	R, R, P(1)	b
b	1	R, R, P(0)	b

1.1. Describe the workings of a Turing machine (referred to as a "computing machine" in the original paper).

1.2. What is the precise output of the machine in Example 1? Certain squares may be left blank. Be sure to justify your answer.

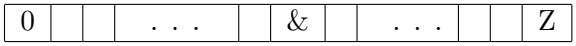
1.3. Design a Turing machine which generates the following output. Be sure to justify your answer.

$$010010100101001\ \ldots$$

1.4. Describe the behavior of the following machine, which begins with a blank tape, with the machine in configuration α.

Configuration		Behavior	
m-config.	symbol	operation	final m-config.
α	none	R P(1)	β
α	1	R P(0)	β
α	0	HALT	(none)
β	1	R P(1)	α
β	0	R P(0)	α

1.5. Given finite, non-empty, sets A and B, design a Turing machine which tests whether $A \subseteq B$. Suppose that the first character on the tape is a 0, simply to indicate the beginning of the tape. To the right of 0 follow the (distinct, non-blank) elements of A, listed in consecutive positions, followed by the symbol &. To the right of & follow the (distinct, non-blank) elements of B, listed in consecutive positions, followed by the symbol Z to indicate the end of the tape:



The symbols 0, &, Z are neither elements of A nor B. The machine starts reading the tape in the right most position, at Z. If $A \subseteq B$, have the machine erase all the elements of A and return a tape with blanks for every square which originally contained an element of A. You may use the following operations for the behavior of the machine:

- R: Move one position to the right.
- L: Move one position to the left.
- S: Store the scanned character in memory. Only one character can be stored at a time.
- C: Compare the currently scanned character with the character in memory. The only operation of C is to change the final configuration depending on whether the scanned square matches what is in memory.
- E: Erase the currently scanned square.
- P( ): Print whatever is in parentheses in the current square.

You may use multiple operations for the machine in response to a given configuration. Also, for a configuration q_n, you may use the word "other" to denote all symbols $\mathcal{S}(r)$ not specifically identified for the given q_n. Be sure that your machine halts.

Turing Machines, Induction and Recursion

The logic behind the modern programmable computer owes much to Turing's "computing machines," discussed in the first section of the project. 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 device. 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" [13, p. 232].

Turing continues [13, p. 235]:

> 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.

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

X			...			

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

2.1. 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.

2.2 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.

2.3. 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.

2.4. 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 write carefully the details of this proof by induction.

2.5. Design a Turing machine, which when given two arbitrary natural numbers, n and m, will compute the product $n \cdot m$. Suppose that the machine begins with the tape

A	1		1		...		1		1	B	1		1		...		1		1	C

where the number of ones between A and B is n, the number of ones between B and C is m, and the machine begins scanning the tape at the far left, reading the symbol A. The output of the machine should be:

A	1		...	1	B	1		...	1	C	1		1		...		1	D

where the number of ones between C and D is $n \cdot m$. Use induction to verify that the machine produces the correct output.

Letting T denote the Turing machine which multiplies n and m together, so that the value of $T(n, m)$ is $n \cdot m$, design T so that for $n \in \mathbf{N}$,

$$T(n, 1) = n$$

and for $m \in \mathbf{N}$, $m \geq 2$, we have

$$T(n, m) = T(n, m - 1) + n.$$

Such an equation provides an example of a recursively defined function, an important topic in computer science. In our case, the algorithm for multiplication, T, is defined in terms of addition, a more elementary operation.

The Universal Computing Machine

The decision problem of Hilbert asked whether there is a standard procedure, an algorithm in modern terminology, which can be invoked to decide whether an arbitrary statement (within some system of logic) is valid. In answering this question, Turing introduced several fundamental concepts, certainly of importance to logic, but also pivotal to the development of the modern programmable computer. The first of these is a "computing machine," called a "Turing machine" today, which is the forerunner of a modern computer program. The next concept is the "universal computing machine," which is in fact a particular type of Turing machine that accepts the instructions of some other machine M in standard form, and the outputs the same sequence as M. Turing writes [13, p. 241–242]:

> It is possible to invent a single machine which can be used to compute any computable sequence. If this machine U is supplied with a tape on the beginning of which is written the $S.D$ [standard description] of some computing machine M, then U will compute the same sequence as M.

The reader should review §3 *Examples of computing machines* reprinted above under "An Introduction to Turing Machines," and have the table from Example 1 at hand before reading the following excerpts [13, p. 239–241]:

5. *Enumeration of computable sequences.*

> A computable sequence γ is determined by a description of a machine which computes γ. Thus the sequence 001011011101111... is determined by the table on p. 234, and, in fact, any computable sequence is capable of being described in terms of such a table.
>
> It will be useful to put these tables into a kind of standard form. In the first place let us suppose that the table is given in the same form as the first table, for example, 1 on p. 233. That is to say, that the entry in the operations column is always one of the form E: E, R: E, L: $P\alpha$: $P\alpha, R$: $P\alpha, L$: R: L: or no entry at all. The table can always be put into this form by introducing more m-configurations. Now let us give numbers to the m-configurations, calling them $q_1, \ldots, q_R$ as in §1. The initial m-configuration is always to be called q_1. We also give numbers to the symbols $S_1, \ldots, S_m$ and, in particular, blank $= S_0$, $0 = S_1$, $1 = S_2$. The lines of the table are now of form

m-config.	*Symbol*	*Operations*	*Final m-config.*	
q_i	S_j	PS_k, L	q_m	(N_1)
q_i	S_j	PS_k, R	q_m	(N_2)
q_i	S_j	PS_k	q_m	(N_3)

Lines such as

$$q_i \qquad S_j \qquad E,\, R \qquad q_m$$

are to be written as

$$q_i \qquad S_j \qquad PS_0,\, R \qquad q_m$$

and lines such as

$$q_i \qquad S_j \qquad R \qquad q_m$$

to be written as

$$q_i \qquad S_j \qquad PS_j,\, R \qquad q_m$$

In this way we reduce each line of the table to a line of one of the forms (N_1), (N_2), (N_3).

From each line of form (N_1) let us form an expression $q_i\, S_j\, S_k\, L\, q_m$; from each line of form (N_2) we form an expression $q_i\, S_j\, S_k\, R\, q_m$; and from each line of form (N_3) we form an expression $q_i\, S_j\, S_k\, N\, q_m$.

Let us write down all expressions so formed from the table for the machine and separate them by semi-colons. In this way we obtain a complete description of the machine. In this description we shall replace q_i by the letter "D" followed by the letter "A" repeated i times, and S_j by "D" followed by "C" repeated j times. This new description of the machine may be called the *standard description* (S.D). It is made up entirely from the letters "A", "C", "D", "L", "R", "N", and from ";". ...

Let us find a description number for the machine I of §3. When we rename the m-configurations its table becomes:

m-config.	Symbol	Operations	Final m-config.
q_1	S_0	$PS_1,\, R$	q_2
q_2	S_0	$PS_0,\, R$	q_3
q_3	S_0	$PS_2,\, R$	q_4
q_4	S_0	$PS_0,\, R$	q_1

Other tables could be obtained by adding irrelevant lines such as

$$q_1 \qquad S_1 \qquad PS_1,\, R \qquad q_2$$

Our first standard form would be

$$q_1\, S_0\, S_1\, R\, q_2;\ q_2\, S_0\, S_0\, R\, q_3;\ q_3\, S_0\, S_2\, R\, q_4;\ q_4\, S_0\, S_0\, R\, q_1;\ .$$

The standard description is

$$DADDCRDAA; DAADDRDAAA; DAAADDCCRDAAAA; DAAAADDRDA;$$

Continuing from [13, p. 243], we read:

Each instruction consists of five consecutive parts

(i) "D" followed by a sequence of letters "A". This describes the relevant m-configuration.

(ii) "D" followed by a sequence of letters "C". This describes the scanned symbol.

(iii) "D" followed by another sequence of letters "C". This describes the symbol into which the scanned symbol is to be changed.

(iv) "L", "R", or "N", describing whether the machine is to move to left, right, or not at all.

(v) "D" followed by a sequence of letters "A". This describes the final m-configuration.

3.1. What is the output of the following machine, T, if T begins in configuration a with a blank tape, scanning the blank at the far left?

Configuration		Behavior	
m-config.	symbol	operation	final m-config.
a	1	R	c
a	blank	P(1), R	b
b	0	R	a
b	blank	P(1)	a
c	blank	P(0)	b

3.2. Rewrite the output of machine T using the "standard description" for the output symbols, i.e., $S_0 =$ blank, $S_1 = 0$, $S_2 = 1$, and then replace each S_j with D followed by C repeated j times.

3.3. What is the standard description ($S.D$) of machine T? Be sure that every instruction, including the last one, is followed by a semi-colon. Make sure that you have the correct answer to this before continuing.

3.4. Suppose that the number of configurations for a given machine M is limited to nine, while the number of symbols which M can recognize or write is limited to four. The machine M begins in its first listed configuration (a) with a blank tape, reading the blank at the far left. Now suppose that the standard description of M is written on every other square (in fact the F-squares) of a second tape, with each instruction followed by a semi-colon (on an F-square as well), with the last semi-colon followed by the symbol "::" on an F-square. Initially all other squares on this second tape are blank. If the standard description for machine I of §3 in Turing's paper is entered on tape in this way, what is the output of the following machine U, which begins in configuration one reading the tape at the far left? Note that 20R is shorthand for move 20 squares to the right, and similarly for 10R. How does this compare with the actual output of machine I, §3?

3.5. If the standard description of machine T from question 3.1 is written on tape as described in 3.4, what is the output of U when applied to this tape? How does this compare to the actual output of T?

3.6. Describe in words the operation of configuration one from machine U. Describe separately the operations of configurations two, three, and four.

3.7. Note that only the first 16 configurations of U are listed. From this point, if U was originally supplied with the standard description of a machine M (as specified in 3.4), then U should output the same sequence as M, except coded according to the standard description of the output symbols. Moreover, U keeps track of the current configuration of M in the first 18 squares immediately following "::". The position of the scanner for M is recorded via the symbol "T" on the tape which U processes. Beginning with configuration 17, outline the remaining operations of U. You may use phrases such as "match m-configuration," "match scanned symbol," or " move scanner for M," etc.

Be sure to include a written explanation of these and any other operations you decide to use in your outline. For this part, you do not need to design an actual Turing machine to perform these tasks. Does recursion occur in your outline? How?

Configuration		Behavior	
m-config.	symbol	operation	final m-config.
1	D	R	1
1	A	R	2
1	::	none	none
1	other	R	1
2	blank	R	2
2	A	R	4
2	D	R, R	3
3	D	R, P(X)	5
3	C	R	4
4	;	R	1
4	::	none	none
4	other	R	4
5	::	20R, P(Y), R, R, P(D), 10R, P(Z)	6
5	other	R	5
6	X	E, R	7
6	other	L	6
7	C	R, P(X)	8
7	R	R, P(X)	10
7	L	none	none
7	N	R, P(X)	11
8	Y	4R	9
8	other	R	8
9	blank	P(C)	6
9	C	R, R	9
10	Z	E, P(T)	12
10	other	R	10
11	Y	R, P(T)	12
11	other	R	11
12	X	E, 4R, P(X)	13
12	other	L	12
13	::	R, R	14
13	other	R	13
14	blank	P(A)	15
14	Y	none	none
14	other	R, R	14
15	X	E, R	16
15	other	L	15
16	;	none	17
16	A	R, P(X)	13

3.8. Beginning with configuration 17, find the actual machine instructions of U so that U finds a match between an arbitrary configuration stored on the 18 squares to the right of "::" and

a configuration at the beginning of a coded instruction to the left of "::". Suppose that the standard description entered on tape is that of a machine M for which there is always a well-defined configuration to follow every move of M. Use only the operations "R," "L," "E," "P( )," "none," and be sure to explain the new steps of U. What is the present-day terminology used to describe U?

Extra Credit: Write a computer program for a universal computing machine (in the language of your choice). Demonstrate with several examples that your universal machine functions properly.

The Decision Problem, *Das Entscheidungsproblem*

Turing's paper "On Computable Numbers with an Application to the Entscheidungsproblem" proved most influential not only for mathematical logic, but also for the development of the programmable computer, and together with work of Alonzo Church (1903–1995) [2, 3] and others [7], inaugurated a new field of study known today as computability. Recall that Turing's original motivation for writing the paper was to answer the decision problem of David Hilbert, posed in 1928 along with a list of other problems [10, 11] dealing with the consistency, completeness and independence of the axioms of a logical system in general. The solutions to these problems, in particular Kurt Gödel's (1906–1978) demonstration of the incompleteness of arithmetic with the existence of statements that are not provable (as true or false) [6], had profound consequences for mathematics, and brought mathematical logic to the fore as a separate field of study [7]. In this section, however, we deal primarily with the problem of deciding whether a given statement is valid within a logical system, the decision problem, which Hilbert expressed as [12, p. 112–113]:

> ... [T]here emerges the fundamental importance of determining whether or not a given formula of the predicate calculus is universally valid. ... A formula ... is called *satisfiable* if the sentential variables can be replaced with the values truth and falsehood ... in such a way that the formula [becomes] a true sentence. ... It is customary to refer to the equivalent problems of *universal validity* and *satisfiability* by the common name of the *decision problem*.

Following Gödel's results, the decision problem remained, although it must be reinterpreted as meaning whether there is a procedure by which a given proposition can be determined to be either "provable" or "unprovable". In the text *Introduction to Mathematical Logic* [4, p. 99], Church formulated this problem as:

> *The decision problem* of a logistic system is the problem to find an effective procedure or algorithm, a *decision procedure*, by which for an arbitrary well-formed formula of the system, it is possible to determine whether or not it is a theorem ... .

To be sure, Church proves that the decision problem has no solution [2, 3], although it is the algorithmic character of Turing's solution that is pivotal to the logical underpinnings of the programmable computer. Moreover, the simplicity of a Turing machine provides a degree of accessibility to logic and computability ideal for readers new to this material.

The concept of a universal computing machine, studied above, has evolved into what now is known as a compiler or interpreter in computer science, and is indispensable for the processing of any programming language. The question then arises, does the universal computing machine provide a solution to the decision problem? The universal machine is the standard procedure for answering all questions that can in turn be phrased in terms of a computer program.

First, study the following excerpts from Turing's paper [13, p. 232–233]:

Automatic machines.

If at each stage the motion of a machine is *completely* determined by the configuration, we shall call the machine an "automatic machine" (or a-machine). ...

Computing machines.

If an a-machine prints two kinds of symbols, of which the first kind (called figures) consists entirely of 0 and 1 (the others being called symbols of the second kind), then the machine will be called a computing machine. If the machine is supplied with a blank tape and set in motion, starting from the correct initial m-configuration, the subsequence of symbols printed by it which are of the first kind will called the *sequence computed by the machine*. ...

Circular and circle-free machines.

If a computing machine never writes down more than a finite number of symbols of the first kind, it will be called *circular*. Otherwise it is said to be *circle-free*. ...

A machine will be circular if it reaches a configuration from which there is no possible move, or if it goes on moving and possibly printing symbols of the second kind, but cannot print any more symbols of the first kind.

Computable sequences and numbers.

A sequence is said to be computable if it can be computed by a circle-free machine. A number is computable if it differs by an integer from the number computed by a circle-free machine. ...

4.1. Consider the following machine, T_1, which begins in m-configuration a with a blank tape, reading the blank at the far left. Is T_1 circle-free? Justify your answer.

T_1 :

Configuration		Behavior	
m-config.	symbol	operation	final m-config.
a	blank	$R,\ P(1)$	b
a	0	R	b
b	1	$R,\ R,\ P(0)$	a
b	blank	(none)	a

4.2. Consider the following machine, T_2, which begins in m-configuration a with a blank tape, reading the blank at the far left. Is T_2 circle-free? Justify your answer.

T_2 :

Configuration		Behavior	
m-config.	symbol	operation	final m-config.
a	blank	$R,\ P(1)$	b
a	0	R	b
b	1	$R,\ R,\ P(0)$	a
b	0	R	a

4.3. Describe in your own words the key feature which distinguishes a circle-free machine from a circular machine.

4.4. Is the sequence 101001000100001 ... computable? If so, find a circle-free machine (with a finite number of m-configurations) that computes this sequence on every other square (the F-squares) of

a tape which is originally blank. If not, prove that there is no circle-free machine that computes the above sequence.

Turing's insight into the decision problem begins by listing all computable sequences in some order:

$$\phi_1,\ \phi_2,\ \phi_3,\ \ldots,\ \phi_n,\ \ldots,$$

where ϕ_n is the n-th computable sequence. Moreover, let $\phi_n(k)$ denote the k-th figure (0 or 1) of ϕ_n. For example, if

$$\phi_2 = 101010\ldots,$$

then $\phi_2(1) = 1$, $\phi_2(2) = 0$, $\phi_2(3) = 1$, etc. Turing then considers the sequence β' defined by $\beta'(n) = \phi_n(n)$. If the decision problem has a solution, then [13, p. 247]:

> We can invent a machine D which, when supplied with the $S.D$ [standard description] of any computing machine M will test this $S.D$ and if M is circular will mark the $S.D$ with the symbol "u" [unsatisfactory] and if it is circle-free will mark it with "s" [satisfactory]. By combining the machines D and U [the universal computing machine] we could construct a machine H to compute the sequence β'.

4.5. Is the number of computable sequences finite or infinite? If finite, list the computable sequences. If infinite, find a one-to-one correspondence between the natural numbers, $\mathbf{N}$, and a subset of the computable sequences. Use the result of this question to carefully explain why H must be circle-free.

4.6. Since H is circle-free, the sequence computed by H must be listed among the ϕ_n's. Suppose this occurs for $n = N_0$. In a written paragraph, explain how $\beta'(N_0)$ should be computed. Is it possible to construct a machine H that computes β'? If so, find the configuration table for H. If not, what part of H, i.e., D or U, cannot be constructed? Justify your answer.

4.7. Does the universal computing machine solve the decision problem? Explain.

4.8. By what name is the decision problem known today in computer science? Support your answer with excerpts from outside sources.

Notes to the Instructor

The project contains four sub-projects "An Introduction to Turing Machines," "Turing Machines, Induction and Recursion," "The Universal Computing Machine," and "The Decision Problem," with the first two of these ideal for an introductory undergraduate course in discrete mathematics or computer programming. The first could be assigned as a separate two-week project while covering naive set theory, while the second could be assigned while the class studies inductive or recursive constructions. A variant of this second sub-project used recently in a beginning programming course requires the construction of a Turing machine to compute the sum of two positive binary integers. This version of the project, "Turing Machines and Binary Addition," is available from the web resource [1].

The latter two sub-projects, which sketch the main results of Turing's paper, are well suited for an advanced or intermediate course in discrete mathematics, logic, or programming. The universal machine is a Turing machine that accepts as its input any other machine, T, and computes the same output as T. This foreshadows the development of a compiler or interpreter in computer science. The decision problem asks whether there is a decision procedure that can be applied to any well-formulated mathematical statement and determine whether the statement is true or false. Turing's negative solution to this problem forms the historical legacy of his paper, not to mention the notion of "Turing computable" to refer to that which can be computed via a Turing machine.

References

[1] Bezhanishvili, G., Leung, H., Lodder, J., Pengelley, D., Ranjan, D., "Teaching Discrete Mathematics via Primary Historical Sources," `www.math.nmsu.edu/hist_projects/`

[2] Church, A., "An Unsolvable Problem of Elementary Number Theory," *Amer. Journal Math.*, **58** (1936), 345–363. This paper with a short foreword by Davis was reprinted on pages 88–107 of [5].

[3] Church, A., "A Note on the Entscheidungsproblem," *Journal of Symbolic Logic*, **1** (1936), 40–41.

[4] Church, A., *Introduction to Mathematical Logic*, Princeton University Press, Princeton, New Jersey, 1996.

[5] Davis, M., *The Undecidable. Basic Papers on Undecidable Propositions, Unsolvable Problems and Computable Functions*, Martin Davis (editor), Raven Press, Hewlett, N.Y., 1965.

[6] Gödel, K., "Über Formal Unentscheidbare Sätze der Principia Mathematica und Verwandter Systeme I," *Monatsh. Math. Phys.*, **38** (1931), 173–198. The English translation of this paper by Mendelson with foreword by Davis was reprinted on pages 4–38 of [5].

[7] Grattan-Guiness, I., *The Search for Mathematical Roots, 1870–1940: Logics, Set Theories and the Foundations of Mathematics from Cantor through Russell to Gödel*, Princeton University Press, Princeton, New Jersey, 2000.

[8] Hilbert, D., *Gesammelte Abhandlungen*, Vol. III, Chelsea Publishing Co., New York, 1965.

[9] Hilbert, D., "Mathematical Problems," Newson M., (translator) *Bulletin of the American Mathematical Society*, **8** (1902), 437–439.

[10] Hilbert, D., "Probleme der Grundlegung der Mathematik," *Mathematische Annalen*, **102**, (1930), 1–9.

[11] Hilbert, D., Ackermann, W., *Grundzüge der Theoretischen Logik*, Dover Publications, New York, 1946.

[12] Hilbert, D., Ackermann, W., *Principles of Mathematical Logic*, L. Hammond, G. Leckie, F. Steinhardt, translators, Chelsea Publishing Co., New York, 1950.

[13] Turing, A. M., "On Computable Numbers with an Application to the Entscheidungsproblem," *Proceedings of the London Mathematical Society* **42** (1936), 230–265. A correction, **43** (1937), 544–546. This paper with a short foreword by Davis was reprinted on pages 115–154 of [5].