THE MISSING PART OF A PROOF
OF THE UNSOLVABILITY
OF THE WORD PROBLEM
FOR GROUPS

by

MARTIN SEBASTIAN GERSON

B.A., McGill University, 1967

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
In the Department
of
Mathematics

SIMON FRASER UNIVERSITY

JULY, 1970

Name: Martin Sebastian Gerson

Degree: Master of Science

Title of Thesis: The Missing Part of a Proof of the Unsolvability of the Word Problem for Groups

EXAMINING COMMITTEE APPROVAL

(T. C. Brown) Senior Supervisor

(B. Alspach) Examining Committee

(D. M. Eaves) Examining Committee

(K. E. Rieckhoff) Examining Committee
External Examiner
Department of Physics

Date Approved: July 28, 1970

Abstract

Joseph J. Rotman, in his group theory textbook, The Theory of Groups, An Introduction, gives a partial proof of the unsolvability of the word problem for groups. The complete proof can be broken into two parts. The first part is a proof of the unsolvability of the word problem for semigroups and the second part is a proof that the unsolvability of the word problem for semigroups yields the unsolvability of the word problem for groups. Rotman gives only the second part of the complete proof. This thesis, then, gives the first part of the proof.

Two problems are presented in the thesis, the weak and the strong word problems for finitely presented groups. In Chapter I the two problems and the use of Church's thesis to prove them both unsolvable is described. Chapter II deals with the concepts of Turing Machines and computable functions. Chapter III is devoted to defining recursive functions and proving them to be computable. In Chapter IV the unsolvability of the weak halting problem for Turing machines is proved and in Chapter V the unsolvability of the strong halting problem for Turing machines is proved. Chapter VI is devoted to showing how the results of Chapters IV and V yield the unsolvability of the weak and strong word problems for semigroups, respectively, and introducing Rotman's completion of the proof of the weak and strong word problems

for groups.

In Appendix A a few solutions for word problems for special classes of groups are mentioned, and in Appendix B we show the equivalence of recursive and computable functions and the existence of the universal Turing machine.

TABLE OF CONTENTS

ACKNOWLEDGMENT

I wish to take this opportunity to express my appreciation to Dr. T.C. Brown for his advice, suggestions, and encouragement during the preparation of this thesis. I would also like to thank Dr. H. Gerber for suggesting one of my major references, and Mrs. Margaret Kelley for doing the typing.

The support received from Dr. Brown's N.R.C. grant No. A-3983 is also appreciated.

Introduction

A. A Brief History

It is known that every group, G, can be defined by means of a presentation, (S;R), where S is a set of generators of the group, and R is a set of defining relations or relators. A relation is of the form $W_1 = W_2$, where W_1 and W_2 are words on the symbols in S and their inverses, and a relator is a single word W on the symbols in S and their inverses. The group G defined by the presentation (S;R) is obtained in the following manner. If we replace each defining relation, $W_1 = W_2$, in R by the relator $W_1W_2^{-1}$ to obtain the set R' of relators and then take the factor group, F/N, where F is the free group generated by S and N is the normal subgroup of F generated by R', we have G is isomorphic to F/N.

Speaking very imprecisely, then, G is (or is isomorphic to) the largest group generated by S in which all the defining relations in R hold and in which all the defining relators in R are equal to the identity. It is also known that anything which "looks like" a presentation for a group, no matter how large nor how strange the defining relations, always determines, in the manner described above, a unique group (up to isomorphism).

We shall employ what many algebraists consider to be an abuse of language, and shall say, if G is a group given by the presentation (S;R), that the elements of S are, in fact, elements of the group G, and that all the elements of G are words in the symbols of S and their inverses.

We said that every presentation defines a group, but there may arise great difficulties as soon as we wish more specific information about the group; e.g., is it Abelian? is it finite? is such-and-such a word in the group equal to such-and-such another word in the group? etc.

In 1911, Max Dehn[7] formulated the following three decision problems for a group, G, given in terms of a presentation, called the "word problem", the "conjugacy" or "transformation problem", and the "isomorphism problem" respectively.

(I) For an arbitrary word W in the generators of G, decide in a finite number of steps whether W is equal to the identity element of G, or not.

(II) For two arbitrary words W_1, W_2 in the generators of G, decide in a finite number of steps whether W_1 and W_2 are conjugate elements of G, or not.

(III) For an arbitrary group G' defined by means of another presentation, decide in a finite number of steps whether G is isomorphic to G', or not.

We say that one of these problems is solved when an "effective procedure" for making the required decision is found. It is the case, however, that none of these problems has been solved in general, and, in fact, none of these problems can be solved in general. In other words, there is a group, G, for which none of the above three problems can be solved.

This thesis deals only with the first of the three problems, the word problem, and the only mention that will be made of the other two problems, the conjugacy and isomorphism problems, will be to show that if the word problem for G is unsolvable, then the conjugacy problem for G is also unsolvable. The isomorphism problem is considerably more difficult to deal with, and so no further mention will be made of it here.

Suppose that G has unsolvable word problem; i.e. suppose that there exists no decision process which will in a finite number of steps determine, for an arbitrary word W in the generators of G, whether or not W is equal to the identity element of G. If the conjugacy problem for G were solvable then there would be a decision process which would finitely determine whether or not W and any given word in G are conjugate in G. In particular we could determine whether W and the identity of G are conjugate. But, of course, any conjugate of the identity is equal to the identity, and so we could determine where or not W is equal to the identity in G, contradicting our supposition.

In 1937, A. M. Turing [20] showed the unsolvability of what we shall call "the strong halting problem for Turing machines." Emil L. Post [13], in 1947, used Turing's result to show the unsolvability of our "strong word problem for finitely presented semigroups." The unsolvability of the word problem for finitely presented groups was finally proved independently by W. W. Boone [26] and P. Novikov [12] in the mid 1950's. Boone's revised proof of 1959 [2] was considerably shortened by J. L. Britton in 1963 [5]. Both Boone's and Britton's proofs start from Post's semigroup result.

In Chapter 12 of his book The Theory of Groups: An Introduction, Rotman [15] gives an exposition of the unsolvability of the word problem for groups. The complete proof can be divided into two parts; the first part is the proof of the unsolvability of the word problem for finitely presented semigroups based on the work of Turing and Post, and the second is the proof of the construction, from the semigroup of the first part, of a finitely presented group with unsolvable word problem. While Rotman roughly describes some of the notions involved in the first part of the proof, he only rigorously gives the second part in his book.

The purpose of this thesis, then, is to give a complete, detailed, rigorous proof of the unsolvability of the word problem for semigroups required by Rotman, so that a student of group theory with a good knowledge of the important concepts of basic group theory, but with no advanced mathematical knowledge outside the area of group theory, except for a small amount of number theory, can read and follow a complete proof of the unsolvability of the word problem for groups by first reading this thesis, and then reading Rotman's proof of the second part, starting in the middle of page 265 of his book, without requiring outside reference material.

Turing's methods have been altered and revised considerably by a number of people, including Post, and the sections of this thesis on Turing machines and recursive functions (Chapters II, III, and V) although rearranged considerably and adapted more specifically for our own purposes, are based heavily on the exposition in Martin Davis's text, Computability and Unsolvability, [6]. The main theorem of Chapter VI is based directly on Post's original paper [13].

B. Mathematical Prerequisites

The only mathematical prerequisite for the understanding of this thesis, other than a basic knowledge of group theory, elementary set theory, and the definition of a semigroup, is some knowledge of the basic concepts and results of the Theory of Numbers; including the concept of divisibility, the definition of a prime number and the fact that there are infinitely many primes, the Fundamental Theorem of Arithmetic, and the Chinese Remainder Theorem.

C. Notation and Terminology

Numbers. The only numbers that will be used in this thesis will be the natural numbers. By "natural numbers" we will mean the positive integers including 0 (zero). Thus, unless it is clearly specified otherwise, the term "number" will always mean "natural number", as will the term "integer" (no negative integers will be used).

n-tuples. We will be frequently referring to n-tuples of numbers and will use the notational convention that the n-tuple $(x_1,x_2\cdots,x_n)$ may be denoted by $(X^{(n)})$ and the n-tuple $(y_1,y_2,\cdots y_n)$ may be denoted by $(Y^{(n)})$. This convention will only be employed for n-tuples of x's or y's. We will however often denote the (n+2)-tuple $(z,x_1,x_2,\cdots,x_n,y)$ by $(z,X^{(n)},y)$ or the (n+1)-tuple $(x,y_1,y_2,\cdots,y_n)$ by $(x,Y^{(n)})$, etc..

Functions. All functions used will be functions whose domains are contained in the natural numbers, or are contained in the set of n-tuples of natural numbers, for some n. A function whose domain is contained in the set of n-tuples will be called "n-ary" or "n-place". (A one-place function will be called singulary; a two-place, binary; and a three-place, ternary.) We shall usually employ the "abuse of language" whereby an n-ary function f..is written $f(x_1,x_2,\cdots x_n)$ or $f(X^{(n)})$, etc.. All functions shall be functions into the natural numbers. We shall adhere to the convention that a necessary condition for two functions to be equal is that they have the same domain, i.e. if one of the functions is not defined for a particular n-tuple of numbers, neither is the other. An n-ary function whose domain is the set of all n-tuples of numbers will be called "total".

Characteristic Functions. Given a set S of n-tuples we define the characteristic function of S, written $C_S(x_1,\cdots,x_n)$, by

$$C_S(a_1,a_2,\cdots a_n) = 0 \text{ if the n-tuple } (a_1,a_2,\cdots,a_n)\in S$$
$$\text{and } C_S(a_1,a_2,\cdots a_n) = 1 \text{ if } (a_1,a_2,\cdots,a_n)\notin S.$$

Ends of Proofs: The symbol □ will be used to denote the end of a proof.

CHAPTER I

THE WORD PROBLEM AND CHURCH'S THESIS.

Let G be a group with generators, S, and defining relations, R. The word problem for G is the problem of determining whether an arbitrary word in the generators of G is or is not equal to the identity in G. We say that the word problem for G is solvable if there exists an effective procedure for determining whether or not any given word in the generators of G is equal to the identity in G, and we say that the word problem for G is unsolvable if no such effective procedure exists. If U and V are words in the generators of G then clearly $U = V$ if and only if $UV^{-1} = 1$. Thus the word problem for G could also be said to be the problem of determining whether two arbitrary words in the generators of G are or are not equal. If H is a semigroup with generators T and defining relations E, then the word problem for H is the problem of determining whether two words in the generators of H are or are not equal.

There are two problems which we can call the word problem for finitely presented groups. The first, which we shall refer to as the weak word problem for finitely presented groups, or simply the weak word problem, is the problem of determining whether in an arbitrary group G with a finite set, S, of generators and a finite set, R, of defining relations, an arbitrary word in the generators of G is or is not equal to the identity in G, or whether two arbitrary words in G are or are not equal in G. Thus the weak word problem for finitely presented groups is the problem of solving the word problem for G for all finitely presented groups G simultaneously. We say that the _weak word problem is solvable_ if there exists an effective procedure for determining whether in an arbitrary finitely presented group G, two arbitrary words in the generators of G are or are not equal. We say that the _weak word problem is unsolvable_ if no such effective procedure exists. The important thing to note here is that we are looking for _one_ single effective procedure that can be applied to _any_ finitely presented group to solve the word problem for that group.

The other problem, which we shall call the _strong word problem for finitely presented groups_, is the problem of finding, for each finitely presented group, G, an effective procedure which will solve the word problem for G; i.e. which can be used to determine whether or not two arbitrary words in the generators of G are equal. The important thing to note here is that we are _not_ looking for one single procedure that can be applied to any finitely presented group, but are asking whether or not for each group we can find a procedure to fit that group. The strong word problem would be unsolvable, then, if we can find a group, G, such that G has unsolvable word problem.

It is immediately clear that a solution to the weak problem yields or includes a solution to the strong problem, and so it might at first seem that the terms "weak" and "strong" should be applied the other way around.

Now suppose that there were no solution to the first, i.e. weak, word problem. Then there is no uniform effective procedure which can be applied to any finitely presented group to solve the word problem for that group. But there are infinitely many finitely presented groups, and it is conceivable that for each one there is an effective procedure which will solve its word problem, but that these procedures are all different and there is no way of determining which procedure to use for an arbitrary group so that there is no single procedure which can be applied to all finitely presented groups. Thus the unsolvability of the second problem is a stronger result that the unsolvability of the first; and since we will show that both problems are, in fact, unsolvable, this is why the terms "weak" and "strong" are applied as they are here.

Similarly, we define the weak and strong word problems for finitely presented semigroups. The problem which most authors refer to as simply <u>the word problem</u> is our strong word problem for finitely presented groups.

Since both word problems are unsolvable, and since the unsolvability of the strong implies the unsolvability of the weak, and since we shall be proving the unsolvability of the strong word problem, there might seem to be no reason to include a separate proof of the unsolvability of the weak word problem. However, a major portion of the proof of the unsolvability of the weak problem is so much simpler than that part of the proof for the strong problem that a separate portion giving the weak result is included in this thesis, if only for the reader who wishes a proof of the weak result without having to read the considerably longer portion for the strong result.

There is one basic difficulty in proving the word problems, both weak and strong, unsolvable. We have only an intuitive idea of what an effective procedure is. The notion of an "effective procedure to solve a problem" is described by Boone[1] as "a uniform set of directions which, when applied to any one of the questions constituting the problem[2], produces the correct answer after a finite number of steps, never at any stage of the process leaving the user in doubt as to what to do next." This seems to be a precise enough description of what we would all expect an effective procedure to be, and, in fact, if we were trying to prove the solvability of the word problem and someone were to propose a certain procedure and argue reasonably that the procedure was effective, there would be no problem in accepting that. The difficulty arises when we are trying to prove that no effective procedure exists. Our description of an effective procedure is sufficient to let us recognize that we have an effective procedure when we do have one, but it doesn't tell us anything about what kind of situation must arise when we don't have one.

If we are trying to show that there is no effective procedure for solving a problem P_1 we might start by showing that

[1] See item [2] of Bibliography, p. 211.

[2] In our case the questions are those of the form, "Is the word W in the group G with generators S and defining relations R equal to 1?"

an effective procedure for solving the problem P_1 can be transformed into an effective procedure for solving some problem P_2 and that any effective procedure for solving the problem P_2 can be transformed into an effective procedure for solving some problem P_3, and so on. But clearly any proof must be finite in length and so we must eventually come to some problem P_n for which we must show directly that no effective procedure exists. So we can transfer the difficulty of showing lack of an effective procedure from one problem to another, but eventually we have to deal with this difficulty directly.

Let us consider the following proposal for handling this difficulty. Suppose we have a problem whose solution consists of finding an effective procedure for answering a countable set of questions, each of which admits only a yes or no answer. Since the number of questions is countable, we can list the questions in a sequence: $Q_1, Q_2, Q_3, \cdots$. We define a function, f, on the natural numbers by $f(n) = 0$ if the answer to question Q_n is yes and $f(n) = 1$ if the answer to question Q_n is no. We say that a function is calculable if an effective procedure exists for determining the value of that function at any given point. It is immediate, then, that our function f is calculable if and only if our problem is solvable. We don't seem to be any further ahead, however, because if we are to show the problem unsolvable we must show the function f uncalculable and this again means we must show that no effective procedure exists. But this difficulty can be taken care of by accepting a proposal generally accepted by logicians and known as

"Church's Thesis". Church's thesis, in the form that we shall employ, is the claim that *every calculable function is recursive*. The converse of this claim will be immediately obvious upon the definition of a recursive function because, as with effective procedure, there is no problem in recognizing a calculable function when faced with one. Church's thesis has, of course, no proof since there is no way that one could formally prove anything about a notion as loosely described as that of a calculable function. However, if we accept Church's thesis, and we shall, we find that we have dealt successfully with our difficulty, for the notion of a recursive function is a well defined one, and there exist methods, sufficient for our purposes, of determining whether or not a function is recursive.

We shall now define a device, called a Turing machine, which can be used to compute certain numerical functions. If a function f can be computed by some Turing machine we call it computable. We will go on to show that a function is recursive only if it is computable, and then produce a function which is not computable, hence not recursive, hence uncalculable. Thus, we will have a problem (the calculation of this function) for which no effective procedure exists. Since, as was said before the difficulty of showing no effective procedure exists can be transferred from one problem to another we will be well on our way to a proof of the unsolvability of the word problem.

The initial problem, or problems, whose unsolvability we shall prove by this method are called the *weak* and *strong* halting problems for Turing machines. We then snow that the solvability of the weak or strong word problem for finitely presented semi-groups would imply the solvability of the weak or strong, respectively, halting problem for Turing machines, showing that the weak and strong word problems for finitely presented semigroups must be unsolvable. Rotman shows that the unsolvability of the strong word problem for finitely presented semigroups yields the unsolvability of the strong word problem for finitely presented groups, and we shall show how Rotman's proof can be adapted to do the same thing for the weak problems.

CHAPTER II

TURING MACHINES.

This chapter will deal with "calculable functions". As we noted in Chapter I, the term "calculable function" is not really properly defined and we will first make an appeal to intuition and say simply that what we mean by a calculable function is a function on a subset of the natural numbers or a set of n-tuples of natural numbers for which an algorithm exists that can be used to determine the value of the function at any given point in its domain. Without attempting to define "algorithm" we will note that it seems intuitively clear that if an algorithm exists for doing a certain job then that job can be done by some kind of "calculating machine".

Our plan of attack will be as follows. We will define a certain kind of "machine", specifically a Turing machine, and will show how this machine can be used to calculate certain functions. We will then define a "computable function" to be any function on a subset of the natural numbers, or a set of n-tuples of natural numbers, for which there exists a Turing machine that will compute the value of this function at any given point in its domain.

Informally, a Turing machine may be thought of as a box that looks something like a tape recorder. The machine can be loaded with a tape, infinitely long in both directions, which is actually a discrete sequence of squares. A finite number of these squares may have symbols printed on them, one symbol to a square, the symbols coming from the list s_1, s_2, s_3, $\cdots$. At any given moment during its operation the machine is in one of a finite number of states or "internal configurations" (the possible internal configurations being denoted by symbols from the list q_1, q_2, q_3, $\cdots$) and is "scanning" one of the squares on the tape. The action of the machine is completely determined by its internal configuration, the symbol, if any, printed on the scanned square, and a finite set of built-in instructions, each instruction being of one of the following forms:

i) When in internal configuration q_i scanning a square with symbol s_j(or scanning a blank square) change the symbol on the scanned square to s_k and change to internal configuration q_l.

ii) When in internal configuration q_i scanning a square with symbol s_j (or a blank square) move the tape so that scanned square is now next the one to the right and change to internal configuration q_l.

iii) When in internal configuration q_i scanning a square with symbol s_j (or a blank square) move the tape so that the scanned square is now the next one to the left and change to internal configuration q_l.

When the machine changes a symbol s_j or a blank to a symbol s_k we say that the machine "prints the symbol s_k". At the beginning of a "computation" a tape, which may be blank or may have a finite number of s_i's printed on it, is placed in the machine so that the machine is scanning one of the squares on the tape, and the machine is set with an internal configuration, q_j. The machine begins operating by itself following its built-in instructions. Of course, *the built-in instructions must be such that no two are contradictory*. When the machine finds itself in a certain internal configuration scanning a square with a symbol (or no symbol) such that none of its instructions correspond to that configuration and symbol (or blank), the machine stops and we say that whatever is printed on the tape when the machine stops is the "resultant" in that machine of whatever was on the tape in the beginning. Notice that in some cases the machine may start and never stop.

We are now ready to formalize our description of the Turing machine.

DEFINITION 2.1. An *expression* is a finite sequence (possibly empty) of symbols chosen from the list: $q_1, q_2, q_3 \cdots$; $s_0, s_1, s_2, \cdots$; R, L.

We will often write B instead of s_0 and the symbol s_0 (or B) will be used to denote what we intuitively think of as a blank square. Because s_1 will be a commonly used symbol which will take on a special significance, we will often simply write 1 for s_1.

DEFINITION 2.2. A *quadruple* is an expression having one of the following forms:

(1) $q_i s_j s_k q_l$

(2) $q_i s_j R q_l$

(3) $q_i s_j L q_l$

DEFINITION 2.3. A *Turing Machine* is a finite, non-empty, set of quadruples that contains no two distinct quadruples beginning with the same q_i and s_j. The q_i's and s_j's that occur in the quadruples of a Turing Machine are called its *internal configurations* and its *alphabet* respectively.

The reader will note that the quadruples of types (1), (2) and (3) correspond to the instructions of types i), ii), and iii) respectively mentioned in the informal description. The condition that no two quadruples may have the same first two symbols corresponds to the informal condition that no two instructions are contradictory. If it is not clear now, this remark will become clear when the reader has reached DEFN 2.7.

DEFINITION 2.4. An *instantaneous description* is an expression that contains exactly one q_i, neither R nor L, and is such that the q_i is not the right-most symbol. If Z is a Turing machine and α an instantaneous description, then we say that *α is an instantaneous description of Z* if the q_i that occurs in α is an internal configuration of Z and the s_i's that occur in α occur in the alphabet of Z.

DEFINITION 2.5. An expression that consists entirely of the symbols s_i is called a *tape expression*. A tape expression may also be called a *word in the s_i's.*

In what follows, P and Q are usually tape expressions.

DEFINITION 2.6. Let Z be a Turing machine, and let α be an inst. des. of Z, where q_i is the internal configuration that occurs in α and where s_j is the symbol immediately to the right of q_i. Then we call q_i the internal configuration of Z at α, and we call s_j the symbol scanned by Z at α. The tape expression obtained by removing q_i from α is called the expression on the tape of Z at α.

DEFINITION 2.7. Let Z be a Turing machine, and let α, β be instantaneous descriptions. Then we write "$\alpha \rightarrow \beta$ (Z)", or (where no ambiguity can result) simply "$\alpha \rightarrow \beta$", to mean that one[1] of the following alternatives holds:

(1) There exist expressions P and Q (possibly empty) such that

$$\alpha \text{ is } Pq_is_jQ$$

$$\beta \text{ is } Pq_ls_kQ$$

and $q_is_js_kq_l \in Z$.

(2) There exist expressions P and Q (possibly empty) such that

$$\alpha \text{ is } Pq_is_js_kQ$$

$$\beta \text{ is } Ps_jq_ls_kQ$$

and $q_is_jRq_l \in Z$

(3) There exists an expression P (possibly empty) such that

$$\alpha \text{ is } Pq_is_j$$

$$\beta \text{ is } Ps_jq_ls_0$$

and $q_is_jRq_l \in Z$

[1] By the definition of a Turing machine, at most one of the alternatives can hold.

(4) There exist expressions P and Q (possible empty) such that

$$\alpha \text{ is } Ps_k q_i s_j Q$$

$$\beta \text{ is } Pq_1 s_k s_j Q$$

and $q_i s_j L q_1 \in Z$

(5) There exists an expression Q (possibly empty) such that

$$\alpha \text{ is } q_i s_j Q$$

$$\beta \text{ is } q_1 s_0 s_j Q$$

and $q_i s_j L q_1 \in Z$.

The following theorems follow immediately from this definition.

THEOREM 2.1. If $\alpha \rightarrow \beta$ (Z) and $\alpha \rightarrow \gamma$ (Z), then $\beta = \gamma$.

THEOREM 2.2. If $\alpha \rightarrow \beta$ (Z), and $Z \subset Z'$, then $\alpha \rightarrow \beta$ (Z').

DEFINITION 2.8. An instantaneous description α is called terminal with respect to Z, if for no β do we have $\alpha \rightarrow \beta(Z)$.

DEFINITION 2.9. By a computation of a Turing machine Z is meant a finite sequence $\alpha_1, \alpha_2, \cdots \alpha_p$ of instantaneous descriptions such that $\alpha_i \rightarrow \alpha_{i+1}$ (Z) for $1 \leqq i \leqq p-1$ and such that α_p is terminal with respect to Z. In such a case we write "$\alpha_p = \text{Res}_Z(\alpha_1)$" and we call α_p the resultant of α_1 with respect to Z.

EXAMPLE 2.1. The reader will recall that we may write B for s_0 and 1 for s_1. Let Z be the Turing machine consisting of the following quadruples. (Formally, Z is the set whose elements are the following quadruples.)

$$q_1 1 B q_1$$
$$q_1 B R q_2$$
$$q_2 1 R q_2$$
$$q_2 B R q_3$$
$$q_3 1 B q_3$$

Let $\alpha_1 = q_1 11B111$. Then the following is a computation of Z.

$$\begin{aligned} \alpha_1 &= q_1 11B111 \\ &\rightarrow q_1 B1B111 \\ &\rightarrow Bq_2 1B111 \\ &\rightarrow B1q_2 B111 \\ &\rightarrow B1Bq_3 111 \\ &\rightarrow B1Bq_3 B11 \end{aligned}$$

Note that $B1Bq_3B11$ is terminal. Hence $\mathrm{Res}_Z(q_1 11B111) = B1Bq_3B11$. Also note that Z has internal configurations q_1, q_2 and q_3 and has alphabet B,1 (or s_0, s_1). The reader will note that this particular machine will always perform a computation, i.e. will always stop after a finite number of steps, no matter what instantaneous description it starts with.

EXAMPLE .2. Let Z be the Turing machine consisting of the following quadruples.

$$q_1 s_0 R q_1$$
$$q_1 s_1 R q_1$$
$$q_1 s_2 R q_1$$
$$q_1 s_3 s_0 q_2$$

Then the following is a computation of Z.

$$\begin{aligned} s_2q_1s_2s_1s_3s_2 &\rightarrow s_2s_2q_1s_1s_3s_2 \\ &\rightarrow s_2s_2s_1q_1s_3s_2 \\ &\rightarrow s_2s_2s_1q_2s_0s_2 \end{aligned}$$

which is terminal since no quadruple of Z begins with q_2s^0. Note, however, that if we begin with the instantaneous description $\alpha = s_2q_1s_2s_1s_2s_1$, we get the following

$$\begin{aligned} \alpha &= s_2q_1s_2s_1s_2s_1 \\ &\rightarrow s_2s_2q_1s_1s_2s_1 \\ &\rightarrow s_2s_2s_1q_1s_2s_1 \\ &\rightarrow s_2s_2s_1s_2q_1s_1 \\ &\rightarrow s_2s_2s_1s_2s_1q_1s_0 \\ &\rightarrow s_2s_2s_1s_2s_1s_0q_1s_0 \\ &\rightarrow s_2s_2s_1s_2s_1s_0s_0q_1s_0 \\ &\rightarrow s_2s_2s_1s_2s_1s_0s_0q_1s_0 \quad \cdots \end{aligned}$$

and the machine "will go on forever". Since no terminal instantaneous description is reached there is no computation of Z beginning with α and we say "$\text{Res}_Z(\alpha)$ is undefined".

We now show how Turing machines can be used to perform certain numerical computations. If n is a positive integer we write s_i^n to denote $s_is_i\cdots s_i$ (n occurrences of s_i) and we let s_i^0 denote the empty expression.

DEFINITION 2.10. With each number n we associate the tape expression $\bar{n}$ where $\bar{n} = 1^{n+1}$.

DEFINITION 2.11. With each k-tuple of integers, $(n_1,n_2,\cdots n_k)$, we associate the tape expression $(\overline{n_1,n_2,\cdots n_k})= \bar{n}_1 B\bar{n}_2 B\cdots B\bar{n}_k$.

Thus $\bar{3}$ = 1111 and $(\overline{2,3,0})$ = 111B1111B1

DEFINITION 2.12. Let M be any expression. Then $\|M\|$ is the number of occurrences of 1 (or s_1) in M.

Thus, $\|11q_4s_2s_41B1\| = 4$ and $\|s_2s_5q_4s_0\| = 0$.

DEFINITION 2.13. Let Z be a Turing machine. Then, for each n, we associate with Z an n-ary function

$$\Psi_Z^{(n)}(x_1,x_2,\cdots x_n)$$

as follows. For each n-tuple $(m_1,m_2,\cdots m_n)$, we set $\alpha = q_1(\overline{m_1,m_2,\cdots m_n})$. Then we let $\Psi_Z^{(n)}(m_1,m_2,\cdots,m_n) = \|\text{Res}_Z(\alpha)\|$ if $\text{Res}_Z(\alpha)$ is defined, i.e. if there is a computation of Z beginning with α, and we leave $\Psi_Z^{(n)}(m_1,m_2,\cdots,m_n)$ undefined if $\text{Res}_Z(\alpha)$ is undefined. We write $\Psi_Z(x)$ for $\Psi_Z^{(1)}(x)$.

DEFINITION 2.14. An n-ary function $f(x_1,\cdots,x_n)$ is partially computable if there exists a Turing machine Z such that

$$f(x_1,x_2,\cdots,x_n) = \Psi_Z^{(n)}(x_1,x_2,\cdots,x_n)$$ [2]

In this case we say that Z computes f. If in addition, $f(x_1,x_2,\cdots,x_n)$ is a total function, i.e. has as its domain the set of all n-tuples of positive integers, then it is called computable.

CHAPTER III

COMPUTABLE AND RECURSIVE FUNCTIONS.

DEFINITION 3.1. The operation of composition associates with the functions $f(Y^{(m)})$, $g_1(X^{(n)})$, $g_2(X^{(n)}),\ldots,$ $g_m(X^{(n)})$, the function $h(X^{(n)}) = f(g_1(X^{(n)}),\ g_2(X^{(n)}),\ldots,$ $g_m(X^{(n)}))$. This function is defined for precisely those n-tuples $(a_1,\ldots,a_n)$ for which $(a_1,\ldots,a_n)$ is in the domain of each of the functions $g_i(X^{(n)})$, $i = 1,2,\ldots,m$, and for which the m-tuple

$$(g_1(a_1,\ldots,a_n),g_2(a_1,\ldots,a_n),\ldots,g_m(a_1,\ldots,a_n))$$

is in the domain of $f(Y^{(m)})$. Its value at $(a_1,\ldots,a_n)$ is $f(g_1(a_1,a_2,\ldots,a_n),\ g_2(a_1,a_2,\ldots,a_n),\ldots,g_m(a_1,a_2,\ldots,a_n))$.

DEFINITION 3.2. The operation of minimalization associates with each total function $f(y,X^{(n)})$ the function $h(X^{(n)})$, whose value for given $X^{(n)}$ is the least value of y, if one such exists, for which $f(y,X^{(n)}) = 0$, and which is undefined if no such y exists.

We write $h(X^{(n)}) = \min_y[f(y,X^{(n)}) = 0]$

EXAMPLE 3.1. Consider the function $x/2 = \min_y[(y+y)-x = 0]$. $x/2$ is a partial function defined only when $x/2$ is even.

DEFINITION 3.3. The total function $f(y,X^{(n)})$ is called regular if $\min_y[f(y,X^{(n)} = 0]$ is total.

DEFINITION 3.4. A function is partial recursive if it can be obtained by a finite number of applications of composition and minimalization beginning with the functions of the following list:

(1) $S(x) = x+1$

(2) $U_i^n(x_1,\cdots,x_n) = x_i, 1 \leq i \leq n$

(3) $x + y$

(4) $x \dot{-} y \quad \begin{array}{ll} = x - y & \text{if } x \geq y. \\ = 0 & \text{if } y > x. \end{array}$

(5) xy.

DEFINITION 3.5. A function is recursive if it can be obtained by a finite number of applications of composition and minimalization of regular functions, beginning with the functions of the list of definition 3.4.

The following corollary is immediate from the preceding definitions.

COROLLARY 3.1. Every recursive function is total and is partial recursive.

As will be shown in the appendix, the converse of this is also true.

The remainder of this chapter will be devoted primarily to showing that every recursive function is computable. (We will show in the appendix that every computable function is recursive.)

To show that every recursive function is computable we must construct Turing machines to compute each of the functions in the list of definition 3.4, and for each appropriate set of computable functions a machine that will compose them, and for each computable function a machine that will minimize it.

THEOREM 3.2. The addition function f(x,y) = x + y is computable.

PROOF: We must construct a Turing machine Z such that $\Psi_Z^{(2)}(x,y) = x + y$. Consider the machine of example 2.1 consisting of the quadruples

$$q_1 1 B q_1$$
$$q_1 B R q_2$$
$$q_2 1 R q_2$$
$$q_2 B R q_3$$
$$q_3 1 B q_3$$

Let $\alpha_1 = q_1(\overline{m_1, m_2}) = q_1\overline{m_1}B\overline{m_2}$. Then

$$\begin{aligned} \alpha_1 &= q_1 1 1^{m_1} B 1 1^{m_2} \\ &\rightarrow q_1 B 1^{m_1} B 1 1^{m_2} \\ &\rightarrow B q_2 1^{m_1} B 1 1^{m_2} \\ &\rightarrow \ldots \\ &\rightarrow B 1^{m_1} q_2 B 1 1^{m_2} \\ &\rightarrow B 1^{m_1} B q_3 1 1^{m_2} \\ &\rightarrow B 1^{m_1} B q_3 B 1^{m_2} \end{aligned}$$

which is terminal. Thus,

$$\begin{aligned}\Psi_Z^{(2)}(m_1,m_2) &= \|\mathrm{Res}_Z(\alpha_1)\| \\ &= \|B1^{m_1}Bq_3B1^{m_2}\| \\ &= m_1 + m_2. \qquad \Box\end{aligned}$$

THEOREM 3.3. The successor function, $S(x) = x + 1$, is computable.

PROOF: Let Z be any Turing machine with respect to which $q_1\overline{m}$ is terminal for all m, e.g. let Z consist of the single quadruple q_1BBq_1. Then $\Psi_Z(m) = \|q_1\overline{m}\| = m + 1$. □

THEOREM 3.4. The proper subtraction function, $f(x,y)= x \dot{-} y$, is computable.

Let Z be the Turing machine consisting of the following quadruples:

$q_1 1Bq_1$

$q_1 BRq_2$

$q_2 1Rq_2$

$q_2 BRq_3$

$q_3 1Rq_3$

$q_3 BLq_4$

$q_4 1Bq_4$

$q_4 BLq_5$

$q_5 1Lq_6$

$q_6 1Lq_6$

$q_6 BLq_7$

$q_7 1Lq_8$

$$q_7 B R q_9$$

$$q_8 1 L q_8$$

$$q_8 B R q_1$$

$$q_9 B R q_{10}$$

$$q_{10} 1 B q_9$$

We must show that $\Psi_Z^{(2)}(m_1,m_2) = m_1 \dot{-} m_2$ and we consider two cases. Let $\alpha_1 = q_1(\overline{m_1,m_2}) = q_1\overline{m_1}B\overline{m_2}$. First suppose that $m_1 \geqq m_2$, and let $k = m_1 - m_2$. Then

$$\begin{aligned}
\alpha_1 &= q_1 1^{m_1+1} B 1^{m_2+1} \\
&= q_1 1 1^{m_2} 1^k B 1^{m_2} 1 \\
&\to q_1 B 1^{m_2} 1^k B 1^{m_2} 1 \\
&\to B q_2 1^{m_2} 1^k B 1^{m_2} 1 \\
&\to \ldots \\
&\to B 1^{m_2} 1^k q_2 B 1^{m_2} 1 \\
&\to B 1^{m_2} 1^k B q_3 1^{m_2} 1 \\
&\to \ldots \\
&\to B 1^{m_2} 1^k B 1^{m_2} 1 q_3 B \\
&\to B 1^{m_2} 1^k B 1^{m_2} q_4 1 B \\
&\to B 1^{m_2} 1^k B 1^{m_2} q_4 B B \\
&\to \ldots \\
&\to B 1^{m_2} 1^k B q_6 1^{m_2} B B \\
&\to B 1^{m_2} 1^k q_7 B 1^{m_2} B B
\end{aligned}$$

$$
\begin{aligned}
&\to \ldots \\
&\to q_8B1^{m_2}1^kB1^{m_2}BB \\
&\to Bq_1 1^{m_2}1^kB1^{m_2}BB = \alpha_3
\end{aligned}
$$

Now, except for initial and final B's, α_3 is like α_1 with a pair of 1's cancelled. The process is now repeated. Eventually

$$
\begin{aligned}
\alpha_1 &\to \ldots \\
&\to \alpha_3 \\
&\to \ldots \\
&\to \ldots \\
&\to B^{m_2}q_1 11^kB1B^{m_2+1} \\
&\to \ldots \\
&\to B^{m_2+1}1^kq_2B1B^{m_2+1} \\
&\to \ldots \\
&\to B^{m_2+1}1^kBq_4 1B^{m_2+1} \\
&\to B^{m_2+1}1^kBq_4BB^{m_2+1} \\
&\to B^{m_2+1}1^kq_5BBB^{m_2+1} = \alpha_p
\end{aligned}
$$

which is terminal. But $\|\alpha_p\| = k = m_1 - m_2$. Hence, if $m_1 \geq m_2$,

$$\Psi_Z^{(2)}(m_1,m_2) = m_1 - m_2.$$

Next suppose that $m_1 < m_2$, and let $k = m_2 - m_1$. Then

$$\alpha_1 = q_1 1^{m_1+1}B1^{m_2+1}$$

$$= q_1 11^{m_1}B1^k1^{m_1}1$$

$$\rightarrow \ldots$$

$$\rightarrow Bq_1 1^{m_1}B1^k1^{m_1}BB$$

$$\rightarrow \ldots$$

$$\rightarrow \ldots$$

$$\rightarrow B^{m_1}q_1 1B1^k1B^{m_1+1}$$

$$\rightarrow B^{m_1}q_1 BB1^k1B^{m_1+1}$$

$$\rightarrow B^{m_1}Bq_2 B1^k1B^{m_1+1}$$

$$\rightarrow B^{m_1}BBq_3 1^k1B^{m_1+1}$$

$$\rightarrow \ldots$$

$$\rightarrow B^{m_1}BB1^kq_4 1B^{m_1+1}$$

$$\rightarrow B^{m_1}BB1^kq_4 BB^{m_1+1}$$

$$\rightarrow \ldots$$

$$\rightarrow B^{m_1}Bq_6 B1^kBB^{m_1+1}$$

$$\rightarrow B^{m_1}q_7 BB1^kBB^{m_1+1}$$

$$\rightarrow B^{m_1}Bq_9 B1^kBB^{m_1+1}$$

$$\rightarrow B^{m_1}BBq_{10} 1^kBB^{m_1+1}$$

$$\rightarrow \ldots$$

$$\rightarrow B^{m_1}BBB^kq_{10} BB^{m_1+1} = \alpha_t$$

which is terminal. But $\|\alpha_t\| = 0$. Hence, if $m_1 < m_2$,

$$\Psi_Z^{(2)}(m_1,m_2) = 0. \qquad \square$$

THEOREM 3.5. For each number $n \geqq 1$ and for each number i, $1 \leqq i \leqq n$, the n-ary function $U_i^n(x_1,x_2,\cdots,x_n)$ is computable.

Let Z be the Turing machine consisting of the following quadruples, where j runs over all integers $\neq i$ such that $1 \leqq j \leqq n$.

$$q_j 1 B q_{2n+j}$$
$$q_j B R q_{j+1}$$
$$q_{2n+j} B R q_j$$
$$q_i 1 B q_i$$
$$q_i B R q_{2n+1}$$
$$q_{2n+1} 1 R q_{2n+1}$$
$$q_{2n+i} B R q_{i+1}$$

Now, $q_1(\overline{m_1,m_2,\cdots m_n}) = q_1 1^{m_1+1} B 1^{m_2+1} B\cdots B 1^{m_i+1} B\cdots B 1^{m_n+1}$

$\rightarrow \cdots$

$\rightarrow B^{m_1+1} B q_2 1^{m_2+1} B\cdots B 1^{m_i+1} B\cdots B 1^{m_n+1}$

$\rightarrow \cdots$

$\rightarrow \cdots$

$\rightarrow B^{m_1+1} B B^{m_2+1} B\cdots B q_i 1^{m_i+1} B\cdots B 1^{m_n+1}$

$\rightarrow B^{m_1+1} B B^{m_2+1} B\cdots B q_i B 1^{m_i} B\cdots B 1^{m_n+1}$

$\rightarrow B^{m_1+1} B B^{m_2+1} B\cdots B B q_{2n+1} 1^{m_i} B\cdots B 1^{m_n+1}$

$\rightarrow \cdots$

$\rightarrow B^{m_1+1} B B^{m_2+1} B\cdots B B 1^{m_i} B q_{i+1}\cdots B 1^{m_n+1}$

$\rightarrow \cdots$

$$\to \cdots$$

$$\to B^s 1^{m_i} B^t q_n 1^{m_n+1} \qquad \text{(for s,t suitably chosen)}$$

$$\to \cdots$$

$$\to B^s 1^{m_i} B^t B^{m_n+1} B q_{n+1} B$$

which is terminal. Hence,

$$\Psi_Z^{(n)}(m_1, m_2, \cdots, m_n) = \|B^s 1^{m_i} B^t B^{m_n+1} B q_{n+1} B\| = m_i \ \square$$

THEOREM 3.6. The function (similar to the multiplication function) $f(x,y) = (x+1)(y+1)$ is computable.

We shall construct a Turing machine Z such that

$$\Psi_Z^{(2)}(x,y) = (x+1)(y+1)$$

Informally what we do is construct Z such that when given an initial instantaneous description, $\alpha_1 = q_1 1^{m_1+1} B 1^{m_2+1}$, it duplicates the second block of 1's m_1 times, each time erasing a 1 from the first block, thus finishing with m_1+1 blocks each with m_2+1 1's.

Let Z consist of the following quadruples:

$q_1 1 B q_1$

$q_1 B R q_2$

$q_2 1 s_2 q_3$

$q_3 s_2 R q_3$

$q_3 1 R q_3$

$q_3 B R q_4$

$q_4 1 R q_3$

q_4BLq_5

$q_5 1Lq_6$

q_5BLq_6

$q_6 1s_3q_6$

$q_6s_3Rq_7$

q_6BBq_{10}

$q_7 1Rq_7$

q_7BRq_8

$q_8 1Rq_8$

q_8B1q_9

$q_9 1Lq_9$

q_9BLq_9

$q_9s_3 1q_5$

$q_{10} 1Lq_{10}$

$Q_{10}BLq_{10}$

$q_{10}s_2Bq_1$

The introduction of the two symbols, s_2 and s_3, into the alphabet of Z is to aid in the process of counting and regulating the total number of duplications.

Our proof that $\Psi_Z^{(2)}(m_1,m_2) = (m_1+1)(m_2+1)$ is inductive. It is convenient to prove first a step which will be repeated many times throughout the proof, i.e. the step of duplicating once the second block of 1's (the block, 1^{m_2+1}).

Let $\alpha = PB1^{m_2+1}q_5BB$, where P is an arbitrary tape expression. Then $\alpha \rightarrow PB1^{m_2}q_61BB$

$$\begin{aligned}
&\rightarrow PB1^{m_2}q_6s_3BB \\
&\rightarrow PB1^{m_2}s_3q_7BB \\
&\rightarrow PB1^{m_2}s_3Bq_8B \\
&\rightarrow PB1^{m_2}s_3Bq_91 \\
&\rightarrow PB1^{m_2}s_3q_9B1 \\
&\rightarrow PB1^{m_2}q_9s_3B1 \\
&\rightarrow PB1^{m_2}q_51B1 \\
&\rightarrow PB1^{m_2-1}q_611B1 \\
&\rightarrow PB1^{m_2-1}q_6s_31B1 \\
&\rightarrow \ldots \\
&\rightarrow PB1^{m_2-1}s_31B1q_8B \\
&\rightarrow PB1^{m_2-1}s_31B1q_91 \\
&\rightarrow \ldots \\
&\rightarrow \ldots \\
&\rightarrow PBs_31^{m_2}B1^{m_2}q_91 \\
&\rightarrow \ldots \\
&\rightarrow PBq_9s_31^{m_2}B1^{m_2+1} \\
&\rightarrow PBq_51^{m_2+1}B1^{m_2+1} \\
&\rightarrow Pq_6B1^{m_2+1}B1^{m_2+1}
\end{aligned}$$

$$\rightarrow Pq_{10}B1^{m_2+1}B1^{m_2+1}$$

Thus $PB1^{m_2+1}q_5BB \rightarrow \cdots \rightarrow Pq_{10}B1^{m_2+1}B1^{m_2+1}$ (*)

Now let $\alpha_1 = q_1 11^{m_1}B1^{m_2+1}$,

$$\begin{aligned}
\text{Then}\quad \alpha_1 &\rightarrow q_1B1^{m_1}B1^{m_2+1} \\
&\rightarrow Bq_2 1^{m_1}B1^{m_2+1} \\
&\rightarrow Bq_3 s_2 1^{m_1-1}B1^{m_2+1} \\
&\rightarrow \ldots \\
&\rightarrow Bs_2 1^{m_1-1}q_3B1^{m_2+1} \\
&\rightarrow Bs_2 1^{m_1-1}Bq_4 1^{m_2+1} \\
&\rightarrow \ldots \\
&\rightarrow Bs_2 1^{m_1-1}B1^{m_2+1}q_3B \\
&\rightarrow Bs_2 1^{m_1-1}B1^{m_2+1}Bq_4B \\
&\rightarrow Bs_2 1^{m_1-1}B1^{m_2+1}q_5BB \\
&\rightarrow \ldots \\
&\rightarrow Bs_2 1^{m_1-1}q_{10}B1^{m_2+1}B1^{m_2+1} && \text{by (*)} \\
&\rightarrow \ldots \\
&\rightarrow Bq_{10}s_2 1^{m_1-1}B1^{m_2+1}B1^{m_2+1} \\
&\rightarrow Bq_1B1^{m_1-1}B1^{m_2+1}B1^{m_2+1} \\
&\rightarrow BBq_2 1^{m_1-1}B1^{m_2+1}B1^{m_2+1} = \alpha_k
\end{aligned}$$

Thus the "effect" of the Turing machine is to delete a 1 from the expression 1^{m_1} and to duplicate 1^{m_2+1}. This process occurs again and again until

$$\begin{aligned}\alpha_k &\to \cdots \\ &\to B^{m_1} q_{10} s_2 \underbrace{B1^{m_2+1}\ B1^{m_2+1}\ B\cdots B1^{m_2+1}}_{m_1+1 \text{ times}} \\ &\to \cdots \\ &\to B^{m_1+1}\ q_2 \underbrace{B1^{m_2+1}\ B1^{m_2+1}\ B\cdots B1^{m_2+1}}_{m_1+1 \text{ times}} = \alpha_p\end{aligned}$$

But α_p is terminal, and $\|\alpha_p\| = (m_1+1)(m_2+1)$. Hence,

$$\Psi_Z^{(2)}(x,y) = (x+1)(y+1). \qquad \square$$

THEOREM 3.7. Let $f(Y^{(m)})$, $g_1(X^{(n)})$, $g_2(X^{(n)}),\cdots,g_m(X^{(n)})$ be (partially) computable. Let $h(X^{(n)})$ be given by.

$$h(X^{(n)}) = f(g_1(X^{(n)}),\ g_2(X^{(n)}),\cdots,g_m(X^{(n)})$$

Then $h(X^{(n)})$ is (partially) computable. Thus the class of (partially) computable functions is closed under the operation of composition.

Before proving theorem 3.7., let us use it to show that multiplication is computable.

COROLLARY 3.8. The multiplication function $f(x,y)=xy$ is computable.

Let $f(x_1,x_2) = x_1 \dot{-} x_2$; $g_1(x,y) = (x+1)(y+1)$; $g_2(x,y) = y + 1$. Then by Theorems 3.4 and 3.6, f and g_1 are computable and $g_2(x,y) = S(U_2^2(x,y))$ and so is computable by theorems 3.5, 3.6, and 3.7. Thus by theorem 3.7

$$\begin{aligned} h(x,y) &= f(g_1(x,y),g_2(x,y) \\ &= (x+1)(y+1) \dot{-} (y+1) \\ &= (x+1)(y+1) - (y+1) \\ &= xy + x \quad \text{is computable.} \end{aligned}$$

Now taking $f(x_1,x_2) = x_1 \dot{-} x_2$ again and taking $g_1(x,y)=xy+x$ and $g_2(x,y) = U_1^2(x,y) = x$ we have the computability of $k(x,y) = f(g_1(x,y), g_2(x,y)) = (xy+x) \dot{-} x = xy$. Thus xy is computable. □

We will now prove theorem 3.7. We shall adopt the convention of systematically omitting final occurrences of B (a blank) in instantaneous descriptions, unless the B is immediately preceded by a q_i. Thus if Z contains the quadruple q_3BLq_3 we shall write $11s_3s_21q_3B \rightarrow 11s_3s_2q_31$ (Z). A check with definition 2.7 will show that there is no problem in doing this.

DEFINITION 3.6. If Z is a Turing machine we let $\theta(Z)$ be the largest number i such that q_i is an internal configuration of Z.

DEFINITION 3.7. A Turing machine Z is called n-regular $(n>0)$ if (1) there is an $s>0$ such that, whenever $\text{Res}_Z[q_1(\overline{m_1,\ldots,m_n})]$ is defined, it has the form $q_{\theta(Z)}(\overline{r_1,\cdots r_s})$ for suitable $r_1,\cdots,r_s$, and

(2) no quadruple of Z begins with $q_{\theta(Z)}$.

Informally, n-regular Turing machines present the outputs of a computation in a form suitable as inputs of a computation by another machine.

DEFINITION 3.8. If Z is a Turing machine, then $Z^{(n)}$ is the Turing machine obtained from Z by replacing each internal configuration q_i, at all of its occurrences in quadruples of Z, by q_{n+i}.

LEMMA 3.1. For every Turing machine Z, we can find a Turing machine Z' such that, for each n, Z' is n-regular, and, in fact,

$$\mathrm{Res}_{Z'}[q_1(\overline{m_1,\cdots,m_n})]=q_{\theta(Z')}\ \overline{\Psi_Z^{(n)}(m_1,\cdots,m_n)}.$$

PROOF: In this and in future descriptions of Turing maçhines there may be comments appearing in parentheses at the right of the formal list of quadruples. These comments are intended only as informal aids to give an intuitive idea of what the machine is "doing".

Let λ and ρ be the first two symbols in the list $s_2, s_3, s_4, \cdots$ that are not in the alphabet of Z. Let Z_1 consist of the following quadruples:

$q_1 1 L q_1$

$q_1 B \lambda q_1$ (print λ on the left)

$q_1 \lambda R q_2$

$q_2 1 R q_2$

$q_2 B R q_3$

$q_3 1 R q_2$ (move right until a double blank is reached)
$q_3 B L q_4$

$q_4 B \rho q_4$ (print ρ on the right)

$q_4 \rho L q_5$

$q_5 1 L q_5$

$q_5 B L q_5$ (move left until λ is reached)

$q_5 \lambda R q_6$

It is easy to see that with respect to Z_1,

$$q_1 (\overline{m_1, \cdots, m_n}) \to \cdots \to q_6 (\overline{m_1, \cdots, m_n}) \rho$$

which is terminal. Now set $K = \theta(Z^{(5)}) = \theta(z) + 5$ and let Z_2 consist of all the quadruples of $Z^{(5)}$ in addition to the following quadruples, where q_i may be any internal configuration of $Z^{(5)}$

$q_i \lambda B q_{k+i}$ (erase the marker λ)

$q_{k+i} B L q_{2k+i}$

$q_{2k+i} B \lambda q_{2k+i}$ (print λ one square to the left)

$q_{2k+i} \lambda R q_i$ (return to the main computation)

$q_i \rho B q_{3k+i}$ (erase the marker ρ)

$q_{3k+i} B R q_{4k+i}$

$q_{4k+i} B \rho q_{4k+i}$ (print ρone square to the right)

$q_{4k+i} \rho L q_i$ (return to the original computation)

Obviously the effect of $Z^{(5)}$ by itself on the instantaneous description $q_6 (\overline{m_1, \cdots, m_n})$ would be the same as that of Z on $q_1(m_1,\ldots,m_n)$ only with the subscript of each q raised by 5. But if we were to "put $Z^{(5)}$ to work" on the instantaneous description $\lambda q_6(\overline{m_1,\cdots,m_n})\rho$ it would get stuck the first time it scanned λ or ρ since neither λ nor ρ is in the alphabet of Z and hence of $Z^{(5)}$. It is easy to see, then, that the addition of the above quadruples allows Z_2 to deal with λ and ρ and perform basically the same computation that $Z^{(5)}$ would perform if λ and ρ weren't there. Now, either $\mathrm{Res}_Z[q_1(\overline{m_1,\cdots,m_n})]$ is defined in which case we have, with respect to Z_2,

$$\lambda q_6(\overline{m_1,\ldots,m_n})\rho \rightarrow \cdots \rightarrow \lambda\alpha\rho,$$

which is terminal, where $\|\alpha\| = \|\mathrm{Res}_Z[q_1(\overline{m_1,\cdots\cdots,m_n})]\|$, or $\mathrm{Res}_Z[q_1(\overline{m_1,\cdots,m_n})]$ is undefined, in which case so is $\mathrm{Res}_{Z_2}[\lambda q_6(\overline{m_1,\cdots,m_n})\rho]$.

Let $M = 5K+1$ and let Z_3 consist of all the quadruples of the form $q_i s_j s_j q_M$ where q_i is any internal configuration of Z_2, where s_j belongs to the alphabet of Z_2, and where no quadruple beginning with $q_i s_j$ belongs to Z_2. Clearly, if $\lambda P q_i Q\rho$ is a terminal instantaneous description with respect to Z_2, we have

$$\lambda P q_i Q\rho \rightarrow \lambda P q_M Q\rho \qquad (Z_3)$$

where $\lambda P q_M Q\rho$ is terminal in Z_3.

Next let Z_4 consist of the following quadruples where s_j ranges over all the symbols of the alphabet of Z other than 1 and B (i.e. s_0 and s_1)

$q_M 1 L q_M$

$q_M B L q_M$ (move leftward looking for λ)

$q_M s_j L q_M$

$q_M \lambda R q_{M+1}$

$q_{M+1} s_j B q_{M+1}$

$q_{M+1} B R q_{M+1}$ (move rightward looking for a 1)

$q_{M+1} 1 B q_{M+2}$

$q_{M+1} \rho B q_{M+4}$ (if ρ is reached without a 1, prepare to terminate)

$q_{M+2} B L q_{M+2}$

$q_{M+2} 1 R q_{M+3}$ (locate the block of 1's)

$q_{M+2} \lambda R q_{M+3}$

$q_{M+3} B 1 q_{M+3}$ (add 1 to the block of 1's)

$q_{M+3} 1 R q_{M+1}$

$q_{M+4} B L q_{M+4}$

$q_{M+4} 1 R q_{M+4}$ (terminate)

$q_{M+4} \lambda 1 q_{M+5}$

With respect to Z_4 $\lambda P q_M Q \rho \rightarrow \dots$

$\rightarrow q_M \lambda P Q \rho$

$\rightarrow \lambda q_{M+1} P Q \rho$

$\rightarrow \dots$

$\rightarrow \lambda B^s q_{M+1} 1 W \rho$ where $s>1$ and W is a tape expression

$\rightarrow \lambda B^s q_{M+2} B W \rho$

$\rightarrow \ldots$

$\rightarrow q_{M+2} \lambda B^{s+1} W \rho$

$\rightarrow \lambda q_{M+3} B^{s+1} W \rho$

$\rightarrow \lambda q_{M+3} 1 B^s W \rho$

$\rightarrow \lambda 1 q_{M+1} B^s W \rho$

and the process is repeated. If there were originally p 1's present we eventually have $\rightarrow \ldots$

$\rightarrow \lambda 1^p B^t q_{M+1} \rho$

$\rightarrow \lambda 1^p B^t q_{M+4} B$

$\rightarrow \ldots$

$\rightarrow q_{M+4} \lambda 1^p$ (in accordance with the convention of omitting final B's.)

$\rightarrow q_{M+5} 1^{p+1}$.

Finally let $Z' = Z_1 \cup Z_2 \cup Z_3 \cup Z_4$. Then, combining our previous observations with our present ones, we have

$$\begin{aligned}\mathrm{Res}_{Z'}[q_1(\overline{m_1,\ldots,m_n})] &= q_{M+5} 1^{\|\mathrm{Res}_Z[q_1(\overline{m_1,\ldots,m_n})]\|+1} \\ &= q_{\theta(Z')} \overline{\Psi_Z^{(n)}(m_1,m_2,\ldots,m_n)}. \quad \square\end{aligned}$$

LEMMA 3.2. For each n-regular Turing machine Z and each $p>0$, there is a $(p+n)$-regular Turing machine Z_p such that whenever

$$\mathrm{Res}_Z[q_1(\overline{m_1,\ldots,m_n})] = q_{\Theta(Z)}(\overline{r_1,\ldots,r_5})$$

it is also the case that

$$\mathrm{Res}_{Z_p}[q_1(\overline{k_1,\ldots,k_p,m_1,\ldots,m_n})] =$$
$$q_{\theta(Z)}(\overline{k_1,\ldots,k_p,r_1,\ldots,r_s}).$$

whereas, whenever $\mathrm{Res}_Z[q_1(\overline{m_1,\ldots,m_n})]$ is undefined, so is $\mathrm{Res}_{Z_p}[q_1(\overline{k_1,\ldots,k_p,m_1,\ldots,m_n})]$.

PROOF: Let δ,ε be distinct s_i's not in the alphabet of Z. Let U_1 consist of the following quadruples.

$$\begin{array}{ll}
q_1 1\delta q_1 & \text{(replace the first 1 by } \delta) \\
q_1\delta R q_2 & \\
\left.\begin{array}{l} q_i 1\varepsilon q_i \\ q_i\varepsilon R q_i \\ q_i B R q_{i+1} \end{array}\right\} & \begin{array}{l} 1<i\leq p \\ \text{(replace 1 by } \varepsilon) \end{array} \\
q_{p+1} 1\varepsilon q_{p+1} & \\
q_{p+1}\varepsilon R q_{p+1} & \\
q_{p+1} B\varepsilon q_{p+2} & \\
q_{p+2}\varepsilon R q_{p+3} &
\end{array}$$

With respect to U_1 $q_1(\overline{k_1,\ldots,k_p,m_1\ldots,m_n})$

$$\to q_1\delta 1^{k_1}B\ldots B1^{k_p+1}B1^{m_1+1}B\ldots B1^{m_n+1}$$

$$\to \delta q_2 1^{k_1}B\ldots B1^{k_p+1}B1^{m_1+1}B\ldots B1^{m_n+1}$$

$$\to \ldots$$

$$\to \ldots$$

$$\to \delta\varepsilon^{k_1}B\ldots B\varepsilon^{k_p+1}q_{p+1}B1^{m_1+1}B\ldots B1^{m_n+1}$$

$$\to \delta\varepsilon^{k_1}B\ldots B\varepsilon^{k_p+1}q_{p+2}\varepsilon 1^{m_1+1}B\ldots B1^{m_n+1}$$

$$\to \delta\varepsilon^{k_1}B\ldots B\varepsilon^{k_p+1}\varepsilon q_{p+3}(\overline{m_1,\ldots,m_n}),$$

which is terminal.

Next, let $N = \Theta(Z^{(p+2)})$, and let $\mathcal{U}_2$ consist of all the quadruples of $Z^{(p+2)}$ and, in addition, the following quadruples, where q_i may be any internal configuration of $Z^{(p+2)}$:

$q_i\varepsilon 1q_{N+i}$ (interrupt computation)

$$\left.\begin{array}{l} q_{N+i}1Lq_{N+i}\\ q_{N+i}\varepsilon Lq_{N+i}\\ q_{N+i}BLq_{N+i}\\ q_{N+i}\delta Bq_{2N+i}\end{array}\right\}\quad\text{(search for }\delta)$$

$q_{2N+i}BLq_{3N+i}$

$q_{3N+i}B\delta q_{3N+i}$ (copy δ)

$q_{3N+i}\delta Rq_{4N+i}$

$q_{4N+i}\varepsilon Rq_{5N+i}$

$q_{4N+i}1Bq_i$ (resume main computation)

$q_{5N+i}\varepsilon Lq_{6N+i}$ (observing ε, prepare to copy ε)

$q_{5N+i}BLq_{7N+i}$ (observing B, prepare to copy B)

$q_{5N+i}1Lq_{6N+i}$

$q_{6N+i}\varepsilon\varepsilon q_{8N+i}$ (copy ε)

$q_{6N+i}B\varepsilon q_{8N+i}$

$q_{7N+i}\varepsilon B q_{8N+i}$ (copy B)

$q_{7N+i}B\varepsilon q_{7N+i}$

$q_{8N+i}\varepsilon R q_{4n+i}$ (repeat)

$q_{8N+i}BRq_{4n+i}$

If P is any tape expression, then, with respect to U_2,

$$
\begin{aligned}
\delta\varepsilon^{k_1}B\ldots B\varepsilon^{k_p+1}q_i\varepsilon P &\to \delta\varepsilon^{k_1}B\ldots B\varepsilon^{k_p+1}q_{N+1}1P\\
&\to\ldots\\
&\to q_{N+i}\delta\varepsilon^{k_1}B\ldots B\varepsilon^{k_p+1}1P\\
&\to q_{2N+i}B\varepsilon^{k_1}B\ldots B\varepsilon^{k_p+1}1P\\
&\to q_{3N+i}BB\varepsilon^{k_1}B\ldots B\varepsilon^{k_p+1}1P\\
&\to q_{3N+i}\delta B\varepsilon^{k_1}B\ldots B\varepsilon^{k_p+1}1P\\
&\to \delta q_{4N+i}B\varepsilon^{k_1}B\ldots B\varepsilon^{k_p+1}1P\\
&\to \delta B q_{5N+i}\varepsilon^{k_1}B\ldots B\varepsilon^{k_p+1}1P\\
&\to\ldots\\
&\to\ldots\\
&\to \delta\varepsilon^{k_1}B\ldots B\varepsilon^{k_p+1}q_{5N+i}1P\\
&\to \delta\varepsilon^{k_1}B\ldots B\varepsilon^{k_p+1}\varepsilon q_i BP
\end{aligned}
$$

Thus, under the "action" of U_2, all of the ε's are moved one square to the left whenever one of them is encountered, making room for a computation, equivalent to that which Z would make, to be carried out.

Now, let $U_3 = U_1 \cup U_2$. Then, with respect to U_3,

$$\begin{aligned} q_1(\overline{k_1,\ldots,k_p,m_1,\ldots,m_n}) &\rightarrow \ldots \\ &\rightarrow \delta\varepsilon^{k_1}B\ldots B\varepsilon^{k_p+1}\,\varepsilon q_{p+3}(\overline{m_1,\ldots,m_n}) \\ &\rightarrow \ldots \\ &\rightarrow \delta\varepsilon^{k_1}B\ldots B\varepsilon^{k_p+1}\,\varepsilon q_N(\overline{r_1,\ldots,r_s}) \end{aligned}$$

wherever $\mathrm{Res}_Z[q_1(\overline{m_1,\ldots,m_n})]$ is defined; otherwise, there is no computation beginning with $q_1(\overline{k_1,\ldots,k_p,m_1,\ldots,m_n})$ (i.e. when starting with this instantaneous description, the machine will never stop).

Finally, let $M = \theta(U_3)$ and let Z_p consist of all the quadruples of U_3 in addition to the following:

$$\begin{array}{ll} q_N 1 L q_N & \\ q_N \varepsilon B q_{M+1} & \text{(erase one } \varepsilon\text{)} \\ q_{M+1} B L q_{M+1} & \\ q_{M+1} \varepsilon 1 q_{M+1} & \\ q_{M+1} 1 L q_{M+1} & \\ q_{M+1} \delta 1 q_{M+2} & \end{array}$$

Then, using what we have just shown for U_3, we have with respect to Z_p, whenever $\mathrm{Res}_Z[q_1(\overline{m_1,\ldots,m_n})]$ is defined,

$$\begin{aligned} q_1(\overline{k_1,\ldots,k_p,m_1,\ldots,m_n}) &\rightarrow \ldots \\ &\rightarrow \ldots \end{aligned}$$

$$\rightarrow \delta\varepsilon^{k_1}B\ldots B\varepsilon^{k_p+1}\ \varepsilon q_N(\overline{r_1,\ldots,r_s})$$

$$\rightarrow \delta\varepsilon^{k_1}B\ldots B\varepsilon^{k_p+1}\ q_N\varepsilon(\overline{r_1,\ldots,r_s})$$

$$\rightarrow \delta\varepsilon^{k_1}B\ldots B\varepsilon^{k_p+1}\ q_{M+1}B(\overline{r_1,\ldots,r_s})$$

$$\rightarrow \ldots$$

$$\rightarrow q_{M+1}\delta 1^{k_1}B\ldots B1^{k_p+1}B(\overline{r_1,\ldots,r_s})$$

$$\rightarrow q_{M+2}1^{k_1+1}B\ldots B1^{k_p+1}B(\overline{r_1,\ldots,r_s})$$

$$= q_{\theta(z_p)}(\overline{k_1,\ldots,k_p,r_1,\ldots,r_s})$$

which is terminal. □

LEMMA 3.3. THE COPYING MACHINES C_p. For each $p \geq 0$ there exists a Turing machine C_p such that C_p is t-regular for all $t > p$ and

$$\mathrm{Res}_{C_p}[q_1(\overline{k_1,\ldots,k_p,m_1,\ldots,m_n})] =$$
$$q_{p+16}(\overline{m_1,\ldots,m_n,k_1\ldots k_p,m_1\ldots m_n}).$$

Informally, the machine C_p recopies on the left all but the p leftmost arguments. Note that p=0 is permitted.

Let C_p consist of the following quadruples:

$q_1 1 L q_1$

$q_1 B L q_2$

$q_2 B s_2 q_2$ (set "marker" s_2)

$q_2 s_2 R q_3$

$q_i 1 R q_i$	$3 \leqq i \leqq p+2$
$q_i B R q_{i+1}$	(move over p blocks of 1's)
$q_{p+3} 1 R q_{p+3}$	(set "marker" s_3)
$q_{p+3} B s_3 q_{p+3}$	
$q_{p+3} s_3 R q_{p+4}$	
$q_{p+4} 1 R q_{p+4}$	(hunt for double blank on right)
$q_{p+4} B R q_{p+5}$	
$q_{p+5} 1 R q_{p+4}$	
$q_{p+5} B L q_{p+6}$	
$q_{p+6} 1 L q_{p+7}$	
$q_{p+6} B L q_{p+7}$	
$q_{p+7} 1 s_4 q_{p+7}$	(scanning 1, replace it by s_4, and prepare to copy 1)
$q_{p+7} s_4 L q_{p+8}$	
$q_{p+7} B s_5 q_{p+7}$	(scanning B, replace it by s_5, and prepare to copy B)
$q_{p+7} s_5 L q_{p+11}$	
$q_{p+7} s_3 B q_{p+15}$	(scanning s_3, erase and prepare to terminate)
$q_{p+8} 1 L q_{p+8}$	(go left)
$q_{p+8} B L q_{p+8}$	
$q_{p+8} s_3 L q_{p+8}$	
$q_{p+8} s_2 s_4 q_{p+10}$	(finding s_2 replace it by s_4)
$q_{p+8} s_4 1 q_{p+9}$	(replace s_4 by 1)
$q_{p+8} s_5 B q_{p+9}$	(replace s_5 by B)
$q_{p+9} 1 L q_{p+10}$	(go left one square)

$q_{p+9}BLq_{p+10}$

$q_{p+10}Bs_4q_{p+10}$ (copy 1, temporarily using s_4)

$q_{p+10}s_4Rq_{p+14}$

$q_{p+11}1Lq_{p+11}$

$q_{p+11}BLq_{p+11}$ (go left)

$q_{p+11}\delta Lq_{p+11}$

$q_{p+11}s_41q_{p+12}$ (replace s_4 by 1)

$q_{p+11}S_5Bq_{p+12}$ (replace s_5 by B)

$q_{p+12}1Lq_{p+13}$ (go left one square)

$q_{p+12}BLq_{p+13}$

$q_{p+13}Bs_5q_{p+13}$ (copy B, temporarily using s_5)

$q_{p+13}s_5Rq_{p+14}$

$q_{p+14}1Rq_{p+14}$

$q_{p+14}BRq_{p+14}$ (go right)

$q_{p+14}s_3Rq_{p+14}$

$q_{p+14}s_41q_{p+6}$ (restore 1 and repeat)

$q_{p+14}s_5Bq_{p+6}$ (restore B and repeat)

$q_{p+15}1Lq_{p+15}$ (go left)

$q_{p+15}BLq_{p+15}$

$q_{p+15}s_41q_{p+16}$ (replace $s_4$by 1 and terminate)

Now, with respect to C_0,

$$\begin{aligned} q_1(\overline{m_1,\ldots,m_n}) &\to \ldots \\ &\to q_2s_2B(\overline{m_1,\ldots,m_n}) \\ &\to s_2q_3B(\overline{m_1,\ldots,m_n}) \\ &\to s_2q_3s_3(\overline{m_1,\ldots,m_n}) \end{aligned}$$

$$\to \ldots$$

$$\to s_2s_3(\overline{m_1,\ldots,m_{n-1}})B1^{m_n}q_71$$

$$\to \ldots$$

$$\to q_8s_2s_3(\overline{m_1,\ldots,m_{n-1}})B1^{m_n}s_4$$

$$\to q_{10}s_4s_3(\overline{m_1,\ldots,m_{n-1}})B1^{m_n}s_4$$

$$\to \ldots$$

$$\to s_4s_3(\overline{m_1,\ldots,m_{n-1}})B1^{m_n}q_{14}s_4$$

$$\to \ldots$$

$$\to \ldots$$

$$\to q_{11}s_41^{m_n}s_3(\overline{m_1,\ldots,m_{n-1}})s_51^{m_n+1}$$

$$\to q_{12}1^{m_n+1}s_3(\overline{m_1,\ldots,m_{n-1}})s_51^{m_n+1}$$

$$\to \ldots$$

$$\to q_{13}s_51^{m_n+1}s_3(\overline{m_1,\ldots,m_{n-1}})s_51^{m_n+1}$$

$$\to \ldots$$

$$\to \ldots$$

$$\to s_41^{m_1}B(\overline{m_2,\ldots,m_n})q_7s_3(\overline{m_1,\ldots,m_n})$$

$$\to \ldots$$

$$\to q_{15}s_41^{m_1}B(\overline{m_2,\ldots,m_n})B(\overline{m_1,\ldots,m_n})$$

$$\to q_{16}(\overline{m_1,\ldots,m_n,m_1,\ldots,m_n})$$

For $p>0$,

$$q_1(\overline{k_1,\ldots,k_p,m_1,\ldots,m_n}) \to \ldots$$

$$\to q_2s_2\overline{B(k_1,\ldots,k_p,m_1,\ldots,m_n)}$$

$$\to \ldots$$

$$\to s_2B(\overline{k_1,\ldots,k_p})q_{p+3}s_3(\overline{m_1,\ldots,m_n})$$

The computation now proceeds as in the case p=0, working from the right and copying onto the left until s_3 is reached. The terminal instantaneous description is

$$q_{p+16}(\overline{m_1,\ldots,m_n,k_1,\ldots,k_p,m_1\ \ldots,m_n}). \qquad \square$$

LEMMA 3.4. THE TRANSFER MACHINES, R_p. For each p>0 there exists a Turing machine, R_p, which is t-regular for every t>p and such that $\mathrm{Res}_{R_p}[q_1(\overline{k_1,\ldots,k_p,m_1,\ldots,m_n})] = q_{p+16}(\overline{m_1,\ldots,m_n,k_1,\ldots,k_p})$.

Informally, the first p arguments are interchanged on the tape with the remaining arguments.

We note that in the copying operation of C_p, each 1 that occurs in the tape expression $(\overline{m_1,\ldots,m_n})$, the expression to be copied, is replaced by s_4 which in turn is again replaced by 1. Thus, we simply define R_p to act just like C_p only erase these s_4's instead of replacing them by 1's. Hence we may define R_p to consist of precisely the quadruples of C_p except that the quadruple $q_{14}s_4 1 q_{p+6}$ is replaced by $q_{14}s_4 B q_{p+6}$. $\square$

LEMMA 3.5. For each n-regular Turing machine Z, there is an n-regular Turing machine Z' such that, whenever

$$\mathrm{Res}_Z\ q_1(\overline{m_1,\ldots,m_n}) = q_{\theta(Z)}(\overline{r_1,\ldots,r_s})$$

it is also the case that

$$\mathrm{Res}_{Z'}[q_1(\overline{m_1,\ldots,m_n})] = q_{\theta(Z')}(\overline{r_1,\ldots,r_s,m_1,\ldots,m_n})$$

whereas, whenever $\mathrm{Res}_Z[q_1(\overline{m_1,\ldots,m_n})]$ is undefined, so is $\mathrm{Res}_{Z'}[q_1(\overline{m_1,\ldots,m_n})]$

If we first recopy $(\overline{m_1,\ldots,m_n})$ on the left and then employ lemma 3.2 to get $(\overline{m_1,\ldots,m_n,r_1,\ldots,r_s})$ and then transfer $(\overline{m_1,\ldots,m_n})$ with $(\overline{r_1,\ldots,r_3})$ we will have what we want. Thus let Z_n be as in lemma 3.2, i.e. such that

$$\mathrm{Res}_{Z_n}[q_1(\overline{m_1,\ldots,m_n,m_1,\ldots,m_n})]=q_{\theta(Z_n)}(\overline{m_1,\ldots,m_n,r_1,\ldots,r_s}).$$

Then let $Z' = C_0 \cup Z_n^{(15)} \cup R_n^{(14+\theta(Z_n))}$ where C_0 and R_n are as in lemmas 3.3 and 3.4 respectively. □

LEMMA 3.6. Let $Z_1,\ldots,Z_p$ be Turing machines. Let $n>0$. Then, there exists an n-regular Turing machine Z' such that

$$\mathrm{Res}_{Z'}[q_1(\overline{m_1,\ldots,m_n})]$$
$$= q_{\theta(Z')}(\overline{\Psi_{Z_1}^{(n)}(m_1,\ldots,m_n),\ldots,\Psi_{Z_p}^{(n)}(m_1,\ldots,m_n)})$$

PROOF: by induction on p. For p=1 this is just lemma 1. Suppose, then, that the result is known for p=k and let $Z_1,\ldots,Z_{k+1}$ be given Turing machines. Let r_i $r_i = \Psi_{Z_i}^{(n)}(m_1,\ldots,m_n)$ for $1\leq i\leq k+1$. By induction there exists an n-regular Turing machine Y_1 such that

$$\mathrm{Res}_{Y_1}[q_1(\overline{m_1,\ldots m_n})] = q_{\theta(Y_1)}(\overline{r_1,\ldots,r_k})$$

Then by lemma 3.5, there is an n-regular Turing machine Y_2

such that

$$\mathrm{Res}_{Y_2}[q_1(\overline{m_1,\ldots,m_n})]=q_{\theta(Y_2)}(\overline{r_1,\ldots,r_k,m_1,\ldots,m_n}).$$

But by lemma 3.1, there is an n-regular Turing machine Y_3 such that

$$\mathrm{Res}_{Y_3}[q_1(\overline{m_1,\ldots,m_n})] = q_{\theta(Y_3)}\overline{r_{k+1}}$$

Combining these two results with the use of lemma 3.2, there is a (k+n)-regular Turing machine Y_4 such that

$$\mathrm{Res}_{Y_4}[q_1(\overline{r_1,\ldots,r_k,m_1,\ldots,m_n})]=q_{\theta(Y_4)}(\overline{r_1,\ldots,r_k,r_{k+1}}).$$

Thus, let $Z' = Y_2 \cup Y_4^{(\theta(Y_2)-1)}$ and we're done. $\square$

Now let $f(Y^{(m)})$, $g_1(X^{(n)})$, $g_2(X^{(n)}),\ldots,g_m(X^{(n)})$ and $h(X^{(n)})$ be as in the statement of theorem 3.7. By lemma 3.6 there is an n-regular Turing machine Z such that

$$\mathrm{Res}_Z[q_1(X^{(n)})]=q_{\theta(Z)}(\overline{g_1(X^{(n)}),g_2(X^{(n)}),\ldots,g_m(X^{(n)})}).$$

Choose Z_1 so that $\Psi_{Z_1}^{(m)}(Y^{(m)}) = f(Y^{(m)})$, and let $Z' = Z \cup Z_1^{(\theta(Z)-1)}$. Then with respect to Z',

$$q_1(X^{(n)}) \to \ldots \to q_{\theta(Z)}(\overline{g_1(X^{(n)}),\ldots,g_m(X^{(n)})})$$
$$\to \ldots \to \alpha,$$

where $\|\alpha\| = f(g_1(X^{(n)}),g_2(X^{(n)}),\ldots,g_m(X^{(n)}))$ when each $g_i(X^{(n)})$ and $f(g_1(X^{(n)}),\ldots,g_m(X^{(n)}))$ is defined; otherwise $\mathrm{Res}_{Z'}[q_1(X^{(n)})]$ is undefined. Hence, $\Psi_{Z'}^{(n)}(X^{(n)}) = h(X^{(n)})$

and theorem 3.7. is finally proved. □

THEOREM 3.9. If $f(y,X^{(n)})$ is computable, then $h(X^{(n)}) = \min_y [f(y,X^{(n)}) = 0]$ is partially computable. Moreover, if $f(y,X^{(n)})$ is regular then $h(X^{(n)})$ is computable.

PROOF: The second statement follows immediately from the first and the definition of a regular function. The idea behind the proof of the first statement is simple. We essentially construct a Turing machine which successively computes $f(0,X^{(n)})$, $f(1,X^{(n)})$,..., until a zero is received. This machine will compute forever if it never gets a zero and so, formally, there will be no computation, which is what we want. The details of this construction are as follows.

Let U consist of the quadruples

$$\left.\begin{array}{l} q_1 1 L q_1 \\ q_1 B L q_2 \\ q_3 B 1 q_3 \end{array}\right\} \begin{array}{l} \text{move two spaces} \\ \text{left and print} \\ \text{a 1.} \end{array}$$

Then with respect to U, $q_1(\overline{X^{(n)}}) \rightarrow \ldots \rightarrow q_3(\overline{0,X^{(n)}})$, which is terminal.

Since $f(y,X^{(n)})$ is computable, there is a Turing machine which will compute f, and so by lemmas 3.1 and 3.5 there is an (n+1)-regular Turing machine Y such that

$$\mathrm{Res}_Y[q_1(y,X^{(n)})] = q_{\theta(Y)}(f(y,X^{(n)}),y,X^{(n)})$$

So if we let $N = \theta(Y^{(2)})$, then, with respect to $Y^{(2)}$,

$$q_3\overline{(y,X^{(n)})} \to \dots \to q_N\overline{(f(y,X^{(n)}),y,X^{(n)})}.$$

Now let M consist of the quadruples

$$q_N 1 B q_N$$
$$q_N B R q_{N+1}$$
$$q_{N+1} 1 1 q_{N+2}$$
$$q_{N+3} B R q_{N+4}$$

Then, with respect to M, if $f(y,X^{(n)}) = k>0$, we have

$$\begin{aligned} q_N\overline{(f(y,X^{(n)}),y,X^{(n)})} &= q_N 1 1^k B\overline{(y,X^{(n)})} \\ &\to \dots \\ &\to q_{N+2} 1^k B(y,X^{(n)}) \text{ which is terminal.} \end{aligned}$$

If, on the other hand, $f(y,X^{(n)}) = 0$, then

$$\begin{aligned} q_N\overline{(f(y,X^{(n)}),y,X^{(n)})} &= q_N 1 B\overline{(y,X^{(n)})} \\ &\to \dots \\ &\to q_{N+4}\overline{(y,X^{(n)})}. \end{aligned}$$

Next, let Q consist of the quadruples

$$q_{N+2} 1 B q_{N+3}$$
$$q_{N+2} B 1 q_3$$
$$q_{N+3} B R q_{N+2}$$

Then, with respect to Q,

$$\begin{aligned} q_{N+2} 1^k B\overline{(y,X^{(n)})} &\to \dots \\ &\to q_3\overline{(y+1,X^{(n)})} \end{aligned}$$

By theorem 3.5 and lemma 3.1, there is an (n+1)-regular

Turing machine Z_1 such that

$$\begin{aligned}\mathrm{Res}_{Z_1}[q_1\overline{(y,X^{(n)})}] &= q_{\theta(Z_1)}\overline{U_1^{n+1}(y,X^{(n)})}\\ &= q_{\theta(Z_1)}1^{y+1}\end{aligned}$$

Let E consist of all the quadruples of Z_1 in addition to the quadruple. $q_{\theta(Z_1)}1Bq_{\theta(Z_1)}$. Then, with respect to $E^{(N+3)}$, letting $K = \theta(E^{(N+3)})$,

$$\begin{aligned}q_{N+4}\overline{(y,X^{(n)})} &\rightarrow \cdots\\ &\rightarrow q_kB1^y\end{aligned}$$

Now, let $Z = U \cup Y^{(2)} \cup M \cup Q \cup E^{(N+3)}$. We shall see that $\Psi_Z^{(n)}(X^{(n)}) = h(X^{(n)}) = \min_y[f(y,X^{(n)}) = 0]$. Let the numbers $X^{(n)}$ be fixed; let $f(i,X^{(n)}) = r_i$ and suppose that $r_0 \neq 0$, $r_1 \neq 0, \ldots, r_{k-1} \neq 0$, $r_k = 0$. Then with respect to Z,

$$\begin{aligned}q_1\overline{(X^{(n)})} &\rightarrow \cdots\\ &\rightarrow q_3\overline{(0,X^{(n)})} && \text{(using U)}\\ &\rightarrow \cdots\\ &\rightarrow q_N\overline{(r_0,0,X^{(n)})} && \text{(using } Y^{(2)}\text{)}\\ &\rightarrow \cdots\\ &\rightarrow q_{N+2}\overline{(r_0-1,0,X^{(n)}} && \text{(using M)}\\ &\rightarrow \cdots\\ &\rightarrow q_3\overline{(1,X^{(n)})} && \text{(using Q)}\\ &\rightarrow \cdots\\ &\rightarrow \cdots\\ &\rightarrow q_3\overline{(k,X^{(n)})}\end{aligned}$$

$\rightarrow \ldots$

$\rightarrow q_N(\overline{r_k,k,X^{(n)}})$ (using $Y^{(2)}$)

$\rightarrow q_N 1\overline{B(k,X^{(n)})}$ (remember, $r_k=0$)

$\rightarrow \ldots$

$\rightarrow q_{N+4}(\overline{k,X^{(n)}})$ (using M)

$\rightarrow \ldots$

$\rightarrow q_k B 1^k$ (using $E^{(n+3)}$).

and $\|q_k B 1^k\| = k = \min_y[f(y,X^{(n)}) = 0] = h(X^{(n)})$.

Notice that if $r_i \neq 0$ for all i then the internal configuration q_{N+4} is never reached, and so no terminal instantaneous description is ever reached, and so $\Psi_Z^{(n)}(X^{(n)})$ is undefined, as will be $h(X^{(n)})$. □

Continuing theorems 3.7. and 3.9. with theorems 3.2, 3.3, 3.4, 3.5, and corollary 3,8, we immediately have

THEOREM 3.10. Every (partial) recursive function is (partially) computable.

CHAPTER IV

THE WEAK HALTING PROBLEM FOR TURING MACHINES

At this point, we should recall that in the outline given in Chapter I for a method of proving the unsolvability of the word problem for groups we noticed that the problem could be "transferred", i.e. it would suffice to find another problem for which there was no effective procedure and then show that this implies that there can be no effective procedure for solving the original problem. We are now ready to state such an other problem.

The problem has to do with Turing machines. A good reason for looking at Turing machines can be found by noticing that the quadruples of a Turing machine are really just rules for transforming one string of symbols (an instantaneous description) into another, and as such "do" a similar thing to that "done" by the defining relations of a group presentation. Another reason for considering Turing machines is this. As we argued before, if we have a problem, a solution to which consists of finding an effective procedure for answering any of a countable number of questions, then we can number these questions and consider the function whose value at a number is 0 if the answer to the question with that number is yes and whose value at a number is 1 if the answer to the question with that number is no. Then there is an effective procedure for answering any of the questions if and only if the function is calculable. We also noted that by Church's thesis, a function is calculable if and only if it is recursive and we have shown that a function is recursive only if it is computable. Thus if our function can be shown to be not computable then it will not be recursive and hence will not be calculable and so our problem will be unsolvable. If our problem has to do with Turing machines, then the function we get from the problem "says something" about Turing machines. If that function is computable, there must, by definition, be a Turing machine which will "compute" it. Thus, we will have a Turing machine which will "say something" about Turing machines; in particular, it will "say something" about itself.

We know from experience in mathematics, particularly in set theory and logic, that paradoxes often occur when we have a statement (or set, or function) which can be applied to itself. The "Russell Paradox" (Consider the set of all sets which do not contain themselves as elements, i.e. $S = \{x: x \notin x\}$, and ask the question, "Is this set an element of itself, i.e. is $S \in S$?") is perhaps the most famous example. Our proof that the function we will get cannot be computable will be based on a similar paradox.

The theorem we are after is the following.

THEOREM 4.1. (THE UNSOLVABILITY OF THE WEAK HALTING PROBLEM FOR TURING MACHINES). There is no effective procedure for determining for an arbitrary Turing machine Z and an arbitrary instantaneous description α of Z whether or not there exists a computation of Z beginning with α.

(Recall that if there is no computation, then, when we "put α into Z", the machine Z will start computing and never stop.)

Informally, our proof will proceed as follows. We will show that there is an effective way of numbering Turing machines, so that given any machine one can find the number associated with it and vice versa. We suppose that there is a solution to the halting problem. Then there is an effective procedure for determining for an arbitrary Turing machine Z and an arbitrary instantaneous description α of Z whether or not there exists a computation of Z beginning with α. In particular such an effective procedure will exist when α is of the form $q_1\overline{x}$ for some number x. On the basis of this, we will show that there exists a Turing machine U such that there is a computation of U beginning with $q_1\overline{x}$ if and only if in the machine associated with the number x there is <u>no</u> computation beginning with $q_1\overline{x}$. We get our paradox by considering what happens when we "put $q_1\overline{w}$ into U", where w is the number associated with the machine U. If there is no computation of U beginning with $q_1\overline{w}$ then, by the way U is constructed, there is such a computation. On the other hand, if there is such a computation, then there isn't one.

We now proceed formally with the proof. Consider the following system of numbering the Turing machines.

If η is any quadruple of the form $q_i s_j s_k q_l$, let $\S(\eta)=i+j+k+l$
If η is any quadruple of the form $q_i s_j R q_l$
or $q_i s_j L q_l$ let $\S(\eta) = i+j+l$

If Z is any Turing machine, and $\eta_1, \eta_2, \ldots, \eta_n$ are the quadruples of Z, let $\S(Z) = \sum_{k=1}^{n} \S(\eta_k)$, and let $|Z| = n$. Note that neither $\S(Z)$ nor $|Z|$ depend on the order in which the quadruples of Z are presented. We now order the Turing machines in a sequence. Assume that all the Turing machines, Z, such that $\S(Z) < m$ have been included in the sequence. We may certainly assume, without loss of generality, that the quadruples of any machine Z are ordered lexicographically, taking as the order of the alphabet for the third symbol of a quadruple R, then L, then $s_1, s_2, s_3, \ldots,$ and ordering the other symbols by their subscripts. Now consider all the Turing machines Z such that $\S(Z) = m$ and $|Z| = 1$ and order them lexicographically. Assume we have dealt with all machines Z such that $\S(Z) = m$ and $|Z| < r$. Then order those such that $\S(Z) = m$ and $|Z| = r$ lexicographically, considering first the first symbol of the first quadruple and then the second symbol of the first quadruple, etc., then the first symbol of the second quadruple, etc.. Since the quadruples of Turing machines are ordered lexicographically, each machine will appear only once in the sequence and it is fairly easy to see that each machine will appear in the sequence. Now for any Turing machine Z let $N(Z) = n$ where Z is the $(n+1)$'st term in this sequence. (Then if Z is the first term, N(Z) = 0) We see immediately that for each Z, N(Z) is well defined and for each number n there is a machine Z such that N(Z) = n. If N(Z) = n we will call the machine Z_n. Thus if N is thought of as a one-to-one correspondence, $Z_n = N^{-1}(n)$.

It is clear from the description given above that we have given an effective procedure for constructing the sequence $Z_0, Z_1, Z_2, Z_3, \ldots Z_n$ for any finite number n and that given any machine Z we can construct the sequence until Z appears and so find N(Z). Thus given any machine Z we can find N(Z) and given any number n we can find Z_n.

Now define the binary function t(n,x) by t(n,x) = 0 if, in the Turing machine Z_n, there is a computation of Z_n beginning with the instantaneous description $q_1(\overline{x}) = {}$ $= q_1 1^{(x+1)} = q_1 s_1^{(x+1)}$, and t(n,x) = 1 if no such computation exists.

LEMMA 4.1. t(n,x) is not computable.

Suppose t(n,x) is computable. Then there is a Turing machine Z such that $\|\text{Res}_Z[q_1(\overline{n,x})]\| = t(n,x)$. By lemma 3.1. there is a Turing machine Y such that Y is 2-regular and $\text{Res}_Y[q_1(\overline{n,x})] = q_{\theta(Y)}\, \overline{\Psi_Z^{(2)}(n,x)} = q_{\theta(Y)}\overline{t(n,x)}$.

Let C_0 be the copying machine of lemma 3.3. Then, by the same lemma, $\text{Res}_{C_0}[q_1\overline{x}] = q_{16}(\overline{x,x})$.

Now let $v = \theta(Y) + 15$ and let M be the Turing machine consisting of the quadruples

$$q_v 1 R q_{v+1}$$
$$q_{v+1} BB q_{v+2}$$
$$q_{v+1} 11 q_{v+2}$$
$$q_{v+2} BB q_{v+1}$$

Then with respect to M, $q_v\overline{0} = q_v 1$

$$\begin{aligned} &\to 1q_{v+1}B \\ &\to 1q_{v+2}B \\ &\to 1q_{v+1}B \\ &\to \ldots \end{aligned}$$

and the machine keeps alternating between these two instantaneous descriptions, never reaching a terminal one, and so there is no computation. Also with respect to M,

$$\begin{aligned} q_v\overline{1} &= q_v 11 \\ &\to 1q_{v+1}1 \\ &\to 1q_{v+2}1 \end{aligned}$$

which is terminal.

Now let $U = C_0 \cup Y^{(15)} \cup M$. Then with respect to U,

$q_1\overline{x} \to \ldots \to q_{16}\overline{(x,x)}$, by C_0

$\to \ldots \to q_v\overline{t(x,x)}$, by $Y^{(15)}$

$\to \ldots \to 1q_{v+2}1$, which is terminal, if $t(x,x) = 1$

and $q_1\overline{x} \to \ldots \to q_{16}\overline{(x,x)}$

$\to \ldots \to q_v\overline{t(x,x)}$

$\to 1q_{v+1}B \to 1q_{v+2}B \to \ldots$ and never reaches a terminal instantaneous description, if $t(x,x) = 0$. Thus there is a computation of U beginning with $q_1\overline{x}$ if and only if $t(x,x)=1$. However, $t(x,x) = 1$ if and only if there is no computation of the Turing machine Z_x beginning with $q_1\overline{x}$. Thus $\mathrm{Res}_U[q_1\overline{x}]$ is defined if and only if $\mathrm{Res}_{Z_x}[q_1\overline{x}]$ is not.

Finally let $w = N(U)$ so that $U = Z_w$. Then we see that $\mathrm{Res}_{Z_w}[q_1\overline{w}]$ is defined if and only if $\mathrm{Res}_{Z_w}[q_1\overline{w}]$ is not defined, surely an impossible situation. Thus, $t(n,x)$ cannot be computable and lemma 4.1 is proved. □

Since $t(n,x)$ is not computable it is not recursive and hence, by Church's thesis, uncalculable. Thus, there can be no effective procedure for determining for an arbitrary Turing machine Z and an arbitrary number x whether or not there is a computation of Z beginning with $q_1\overline{x}$. And so, obviously, there cannot be an effective procedure for determining for an arbitrary Turing machine Z and an arbitrary instantaneous description α of Z, whether or not there exists a computation of Z beginning with α, and theorem 4.1. is proved. Thus the weak halting problem for Turing machines is unsolvable. □

The reader wishing only the proof of the unsolvability of the weak word problem, not caring about the unsolvability of the strong, can now omit Chapter V and proceed immediately to Chapter VI.

CHAPTER V

MORE RECURSIVE FUNCTIONS AND THE STRONG HALTING PROBLEM FOR TURING MACHINES

We have shown that there is no effective procedure which will determine for an arbitrary Turing machine and an arbitrary instantaneous description in that machine, whether or not there is a computation of that machine beginning with that instantaneous description. We have called this result "the unsolvability of the weak halting problem for Turing machines". There is, however, a similar but stronger result that we can prove.

If Z is a Turing machine, we define the halting problem for Z to be the problem of finding an effective procedure for determining for an arbitrary instantaneous description in Z, whether or not there is a computation of Z beginning with that instantaneous description. Thus the halting problem for Turing machines is really the problem of finding a single procedure which will be a solution to the halting problem for Z for all Turing machines, Z, at once. We will show that there is a specific Turing machine Z_0 such that the halting problem for Z_0 is unsolvable. Thus if we call the problem of finding for each Turing machine, Z a solution to the halting problem for Z the strong halting problem for Turing machines, then we will have shown the unsolvability of the strong halting problem.

This indeed is a stronger result than our previous one. It could be possible that given any Turing machine we could find a solution to the halting problem for that machine, but that, since there is an infinite number of machines, there is no uniform solution that would work for all of them.

The proof of this stronger result is considerably more difficult than that of the weaker result. We must introduce the notion of predicates, a new numbering system for Turing machines and expressions, and must do much more work examining recursive and computable functions and predicates before we can find a computable predicate or function that will be just right for our purposes, i.e. which will be computed by a Turing machine, Z_0, with unsolvable halting problem.

An expression that contains lower-case letters of the Roman alphabet (with or without subscripts) and that becomes a statement, either true or false, when these letters are replaced by any numbers whatever (always assuming that the same letter, at two different occurrences in the expression, is replaced by the same number) is called a predicate. We shall usually employ upper case Roman letters, such as P,Q,R,S,T, to designate predicates and, as with functions, denote an n-ary predicate, P, with variables or arguments $x_1,x_2,\ldots,x_n$ by $P(x_1,x_2,\ldots x_n)$ or $P(X^{(n)})$. Now, if $P(x_1,x_2,\ldots,x_n)$ is an n-ary predicate then the set of all n-tuples of numbers $(a_1,a_2,\ldots,a_n)$ for which $P(a_1,a_2,\ldots,a_n)$ holds is called the extension of P and is written $\{x_1,x_2,\ldots,x_n \mid P(x_1,\ldots,x_n\}$ or $\{X^{(n)} \mid P(X^{(n)})\}$.

The characteristic function of the extension of $P(X^{(n)})$ will be called just the characteristic function of $P(X^{(n)})$ and will be denoted by $C_p(X^{(n)})$. Thus, $C_p(a_1,a_2,\ldots,a_n) = 0$ if $P(a_1,\ldots,a_n)$ is true and $C_p(a_1,\ldots,a_n) = 1$ if $P(a_1,\ldots,a_n)$ is false. The connectives $\vee$ (or), $\wedge$ (and), and $\sim$ (not) can be applied to predicates to obtain new predicates. Thus, for example, if $P(x,y)$ and $Q(y,z)$ are predicates, so are $P(x,y) \vee Q(y,z)$, $P(x,y) \wedge Q(y,z)$, $\sim P(x,y)$. $P(x,y) \vee Q(y,z)$ will hold for all triples (a_1,a_2,a_3) for which either $P(a_1,a_2)$ or $Q(a_2,a_3)$ holds; $P(x,y) \wedge Q(y,z)$ will hold for all triples (a_1,a_2,a_3) for which both $P(a_1,a_2)$ and $Q(a_2,a_3)$ hold; and $\sim P(x,y)$ will hold for all pairs (a_1,a_2) for which $P(a_1,a_2)$ fails. Note that while both $P(x,y)$ and $Q(y,z)$ are binary predicates, both $P(x,y) \vee Q(y,z)$ and $P(x,y) \wedge Q(y,z)$ are ternary.

Two n-ary predicates are said to be equivalent if they have the same extension. If $P(X^{(n)})$ and $Q(X^{(n)})$ are equivalent we write $P(X^{(n)}) \leftrightarrow Q(X^{(n)})$. Thus, if $P(x,y)$ and $Q(y,z)$ are as before, $P(x,y) \vee Q(y,z) \leftrightarrow \sim(\sim P(x,y) \wedge \sim Q(y,z))$. (Either P or Q holds if and only if it is not true that both P and Q fail.)

Note that $P(x,y) \leftrightarrow \sim\sim P(x,y)$.

Now, let $P(y,x_1,\ldots,x_n)$ (or, as we may write, $P(y,X^{(n)})$) be an $(n+1)$-ary predicate. Then the expression

$$P(0,X^{(n)}) \vee P(1,X^{(n)}) \vee P(2,X^{(n)}) \vee \ldots \vee P(z,X^{(n)})$$

is another $(n+1)$-ary predicate, which we may write $Q(z,X^{(n)})$. The statement obtained by inserting definite numbers $c,a_1,a_2,\ldots,a_n$ for the letter $z,x_1,x_2,\ldots,x_n$ in the expression is true if and only if there is a number b, $0\leq b\leq c$, for which $P(b,a_1,a_2,\ldots,a_n)$ is true. We designate this predicate by $\overset{z}{\underset{y=0}{\exists}} P(y,X^{(n)})$. That is, $\overset{z}{\underset{y=0}{\exists}} P(y,X^{(n)}) \leftrightarrow P(0,X^{(n)}) \vee P(1,X^{(n)}) \vee \ldots \vee P(z,X^{(n)})$.

Similarly we write

$$\overset{z}{\underset{y=0}{\forall}} P(y,X^{(n)}) \leftrightarrow P(0,X^{(n)}) \wedge P(1,X^{(n)}) \wedge \ldots \wedge P(z,X^{(n)}).$$

The symbols "$\overset{z}{\underset{y=0}{\exists}}$" and "$\overset{z}{\underset{y=0}{\forall}}$" are referred to as a <u>bounded existential quantifier</u> and a <u>bounded universal quantifier</u>, respectively.

$\underset{y}{\exists} P(y,X^{(n)})$ may be regarded as an abbreviation of the "infinite expression" $P(0,X^{(n)}) \vee P(1,X^{(n)}) \vee \ldots$. This is of course not strictly speaking kosher since there is not really any such thing as an infinite expression. More accurately, $\underset{y}{\exists} P(y,X^{(n)})$ is an n-ary predicate which holds for a given n-tuple $(a_1,a_2,\ldots,a_n)$, if and only if there is some number y_0 such that $P(y_0,a_1,a_2,\ldots,a_n)$ is true. Similarly

$\underset{y}{\forall} P(y,X^{(n)})$ holds for $(a_1,a_2,\ldots,a_n)$, if and only if, for every number y_0, $P(y_0,a_1,a_2,\ldots,a_n)$ is true. We may informally think of $\underset{y}{\forall} P(h,X^{(n)})$ as representing the "infinite expression" $P(0,X^{(n)}) \wedge P(1,X^{(n)}) \wedge \ldots$. "$\underset{y}{\exists}$" and "$\underset{y}{\forall}$" are referred to as an *existential quantifier* and a *universal quantifier*, respectively.

Note that $\underset{y}{\forall} P(y,X^{(n)}) \leftrightarrow \sim\underset{y}{\exists} \sim P(y,X^{(n)})$.

DEFINITION 5.1. Let S be a set of n-tuples. Then we say that S is recursive or computable, accordingly, if its characteristic function is.

DEFINITION 5.2. The predicate $P(X^{(n)})$ is called recursive if its extension, $\{X^{(n)} | P(X^{(n)})\}$, is.

Definitions 5.1 and 5.2 immediately yield.

COROLLARY 5.1. A predicate, $P(X^{(n)})$, is recursive if and only if characteristic function $C_p(X^{(n)})$ is recursive.

We are now in a position to say more about the particular useful computable function mentioned earlier, i.e. the function that will be computed by a Turing machine with unsolvable halting problem.

Now, with each recursive predicate is associated a recursive function, its characteristic function, and so we can search for our particular recursive function by searching for an appropriate predicate. Suppose that $P(x,y)$ is a binary predicate. Then $\underset{y}{\exists} P(x,y)$ is a unary predicate. Suppose further that $P(x,y)$ is recursive but $\underset{y}{\exists} P(x,y)$ is not.

Now, since $P(x,y)$ is recursive, its characteristic function, $C_p(x,y)$ is a recursive function. According to the definition of recursive functions, then, the function $\min_y[C_p(x,y) = 0]$ is a partial recursive function. Now, $C_p(x,y) = 0$ if and only if $P(x,y)$ holds. So if $P(x,y)$ is a predicate, we write "$\min_y P(x,y)$" for "$\min_y[C_p(x,y) = 0]$", and we have the result that $\min_y P(x,y)$ is partial recursive if $P(x,y)$ is recursive.

Given a specific value for x, $\min_y P(x,y)$ is defined to be the least value of y such that $P(x,y)$ holds, if such a value of y exists, and $\min_y P(x,y)$ is undefined if no such value of y exists. Since $\min_y P(x,y)$ is a partial recursive function it is partially computable and so there is a Turing machine Z_0 such that $\Psi_{Z_0}(x) = \min_y P(x,y)$. Thus there is a computation of Z_0 beginning with the instantaneous description $q_1\overline{x} = q_1 1^{x+1}$ if and only if $\min_y P(x,y)$ is defined, which happens if and only if there is a y such that $P(x,y)$ holds, which in turn happens if and only if $\exists_y P(x,y)$ holds.

Now $\exists_y P(x,y)$ is not recursive and so its characteristic function is not recursive. By Church's thesis, then, there is no "effective procedure" for determining, for an arbitrary value of x, what the value of the characteristic function at x will be. Thus there is no effective procedure for determining, for an arbitrary value of x, whether ot not $\exists_y P(x,y)$ holds; and hence there is no effective procedure for determining, for an arbitrary value of x, whether or not there is a computation of Z_0 beginning with the instantaneous description $q_1\overline{x}$, and the halting problem for Z_0 is unsolvable.

So all we need to do to find a Turing machine Z_0 with unsolvable halting problem is to find a predicate $P(x,y)$ such that $P(x,y)$ is recursive but $\underset{y}{\exists} P(x,y)$ is not recursive. The greater part of the remainder of this chapter, then, will be devoted to first defining a predicate which we think should work, and then proving that it, in fact, does work.

As was the case with the weak halting problem result, the predicate we look for will "say something" about Turing machines. Of course, the only "inputs" that predicates will take are numbers or n-tuples of numbers, and so before we can define a predicate which "says" anything about Turing machines we must find an "effective" method of numbering Turing machines. In this case, the numbering system that we used before is not very satisfactory, so we will develop a new system.

The symbols used in the discussion of Turing machines are

$$R, L$$
$$s_0, s_1, s_2, \ldots$$
$$q_1, q_2, q_3, \ldots\ .$$

With each of these symbols we associate an odd number $\geqq 3$ as follows:

R	L	s_0	q_1	s_2	q_2	s_2	q_3	$s_3,\ldots$
|	|	|	|	|	|	|	|	|
3	5	7	9	11	13	15	17	19

Thus for each i, s_i is associated with 4i+7 and q_i with 4i+5.

Hence, with any expression M there is now associated a finite sequence of odd numbers $a_1,a_2,\ldots,a_n$. For example, the quadruple $q_1 1 R q_2$ is associated with the sequence 9,11,3,13, (recall that 1 is s_1 and B is s_0), and the instantaneous description $q_1 111$ is associated with 9,11,11,11. We will now define a method of associating a single number with each such sequence and hence with each expression.

DEFINITION 5.3. Let M be the expression consisting of the symbols $r_1,r_2,r_3,\ldots,r_n$. Let $a_1,a_2,\ldots,a_n$ be the corresponding numbers associated with these symbols. Then the "Gödel number of M" is the number

$$r = \prod_{k=1}^{n} Pr(k)^{a_k} ,$$

where Pr(k) is defined to be the nth prime number, taking 0 to be the 0th prime. We write gn(M) = r. If M is the empty expression, we let 1 be its Godel number and we write gn(M) = 1. If gn(M) = r we also write M = exp(r).

Thus $gn(q_1 1 R q_2) = 2^9 \cdot 3^{11} \cdot 5^3 \cdot 7^{13}$.

As an immediate consequence of this definition and of the fundamental theorem of Arithmetic we have

COROLLARY 5.2. If M and N are expressions such that $gn(M) = gn(N)$, then $M = N$.

DEFINITION 5.4. If $M_1, M_2, \ldots, M_n$ is a finite sequence of expressions, then the Gödel number of this sequence of expressions is defined to be the number

$$\prod_{k=1}^{n} Pr(k)^{gn(M_k)} .$$

Thus, the Gödel number of the sequence $q_1 1 B q_1$, $q_1 B R q_2$ is

$$2^{2^9 \cdot 3^{11} \cdot 5^7 \cdot 7^9} \cdot 3^{2^9 \cdot 3^7 \cdot 5^3 \cdot 7^{13}} .$$

COROLLARY 5.3. No number is the Gödel number both of an expression and of a sequence of expressions. (Note that we make the convention that the empty expression, the expression with no symbols, is considered an expression, but the empty sequence of expressions, the sequence with no expressions, is not considered to be a sequence of expressions. If it was, the two things would have the same Gödel number and ruin this corollary.)

PROOF: Let r be a Gödel number. If $r = 1$ then r is the Gödel number of the empty expression. 1 cannot be the Gödel number of a sequence of expressions because any such sequence contains at least one expression and the Gödel number of this expression must be 1, and so the Gödel number of the sequence will be $\geq 2^1 = 2$.

Now suppose $r \geqq 2$. Then by the way Gödel numbers are constructed, $r = 2^n \cdot m$ where m is odd, and $n > 0$. Now if $2^n \cdot m$ is the Gödel number of an expression, n the number associated with a symbol and hence is odd and $\gtrsim 3$. If $2^n \cdot m$ is the Gödel number of a sequence of expressions, n itself is the Gödel number of an expression and so will be 1 or even. □

Corresponding to Corollary 5.2, we have

COROLLARY 5.4. Two sequences of expressions that have the same Gödel number are identical.

Now, a computation of Turing machine is a finite sequence of expressions, all of them instantaneous descriptions, and thus has a Gödel number. A Turing machine, however, is simply a finite set of expressions, all of them quadruples, in which order is irrelevant.

DEFINITION 5.5. Let Z be a Turing machine. Let $M_1, M_2, \ldots, M_n$ be any arrangement of the quadruples of Z without repetitions. Then, the Gödel number of the sequence $M_1, M_2, \ldots, M_n$ is called a Gödel number of the Turing machine Z.

Note that a Turing machine with n quadruples has n! distinct Gödel numbers, one for each permutation of its quadruples.

DEFINITION 5.6. For each $n > 0$, $T_n(z, x_1, x_2, \ldots, x_n, y)$ is the predicate that holds, for given $z, x_1, \ldots, x_n, y$, exactly when z is the Gödel number of a Turing machine Z, and y is the Gödel number of a computation of Z beginning with the instantaneous description $q_1(\overline{x_1,\ldots,x_n})$. $T_1(z,x,y)$ will be written simply as $T(z,x,y)$.

We will prove that for all n, the predicates $T_n(z,x_1,\ldots,x_n,y)$ are recursive. Then the predicate $T(z,x,y)$ will surely be recursive. We will then define the predicate $W(x,y)$ by $W(x,y) \leftrightarrow T(x,x,y)$.

$W(x,y)$ will be the predicate we want, for since T is recursive W will certainly be recursive, and we will be able to show that $\underset{y}{\exists} W(x,y)$ is not recursive.

We now proceed with the work of showing that for all n, $T_n(z,x_1,\ldots,x_n,y)$ is recursive. We begin by listing some more recursive, and hence computable functions which we will need to use. Each function is accompanied by a definition, if necessary, and a formula which shows how it is constructed by composition or minimalization from functions in Definition 3.4. and thus proves them to be recursive. (Minimalization, here, will only be applied to regular functions.)

(1) $N(x) = 0 \qquad N(x) = u_1^{\,1}(x) \dot{-} U_1^{\,1}(x)$.

(2) $\alpha(x) = 1 \dot{-} x$; that is $\alpha(0) = 1$

$\alpha(x) = 0$ if $x>0$.

$\alpha(x) = S(N(x)) \dot{-} U_1^{\,1}(x)$.

(3) $x^2 = U_1^{\,1}(x) \cdot U_1^{\,1}(x)$.

(4) $[\sqrt{x}]$, the largest integer $\leq\sqrt{x}$.

$$[\sqrt{x}] = \min_y[(y+1)^2 \dot{-} x \neq 0]$$
$$= \min_y[\alpha((S(U_2^2(x,y))^2 \dot{-} U_1^2(x,y))=0].$$

(5) $|x-y| = (x\dot{-}y) + (y\dot{-}x)$

(6) $[x/y]$. If $y\neq 0$, $[x/y]$ is the largest integer less than or equal to the rational number $\frac{x}{y}$. If $y=0$, $[x/y] = 0$. We have

$$[x/y] = \min_z[y=0 \quad y(z+1)>x]$$
$$= \min_z[y=0 \quad y(z+1)\dot{-}x\neq 0]$$
$$= \min_z[y\cdot\alpha(y(z+1)\dot{-}x)=0]$$

(7) $R(x,y)$. If $y\neq 0$, $R(x,y)$ is the remainder on dividing x by y. That is $\frac{x}{y} = [x/y]+\frac{R(x,y)}{y}$, so $R(x,y) = x\div y[x/y]$.

We take $R(x,y) = x \dot{-} y[x/y]$, so $R(x,0)=x$.

Recall that we said that Church's thesis, i.e. the statement that every function which we intuitively feel should be calculable is recursive, could not be proved and that we would have to accept it on faith. The reader should find that, as we proceed on our course of searching for the particular useful recursive function mentioned before, this act of faith will seem to become more and more valid, as we will be showing the recursiveness of some quite complicated functions, which we might not have thought, at first glance, could be derived from the few simple functions in the definition.

We will find it convenient to have the recursiveness, and hence computability, of such functions as x^y and $x!$. These are certainly functions the values of which we know how to calculate, and so, if Church's thesis is to be believed, they must be recursive. When we calculate the value of x^y for particular given numbers x and y we must first calculate x^z for all numbers z such that $0 \leq z < y$. Similarly to calculate $x!$, we first must calculate $z!$ for all z such that $0 \leq z < x$. The processes we use can be represented by the pairs of equations

$$x^0 = 1$$
$$x^{y+1} = x^y \cdot x$$

and

$$0! = 1$$
$$(x+1)! = x! \cdot (x+1)$$

respectively.

In general, we know from the Recursion Theorem of elementary set and natural number theory that if $f(X^{(n)})$ and $g(X^{(n+2)})$ are total functions, then there is a unique total function $h(X^{(n+1)})$ that satisfies the equations

$$h(0,X^{(n)}) = f(X^{(n)})$$
$$\text{and} \quad h(z+1,X^{(n)}) = g(z,h(z,X^{(n)}),X^{(n)}).$$

For example, in the case of the function x^y, $n=1$, $f(x)=0$, $g(y,z,x)=z \cdot x$ and $h(y,x) = x^y$. In the case of the function $x!$, $n=0$, $f=1$, $g(x,y)=y \cdot x$, and $h(x) = x!$.

DEFINITION 5.5. The operation of primative recursion associates with the given total functions $f(X^{(n)})$, $g(X^{(n+2)})$ the function $h(X^{(n+1)})$, where

$$h(0,X^{(n)}) = f(X^{(n)})$$
$$h(z+1,X^{(n)}) = g(z,h(z,X^{(n)},X^{(n)})$$

It would be very convenient, indeed, if we could know that whenever we get a function by primative recursion from two functions which we know to be recursive, then the new function is also recursive. This is, in fact, the case. But before we can prove this result, we must prove some results about finite sequences of numbers.

It is a well known fact that there exists a one-to-one correspondence between the set of natural numbers and the set of ordered pairs of natural numbers and, indeed, that such a correspondence can be set up in an "effective" manner. By Church's thesis, then, we should certainly be able to set up this correspondence by recursive, and hence computable, functions, and we will show now how this can be done.

THEOREM 5.7. There exist recursive functions $J(x,y)$, $K(z)$, $L(z)$ such that

$$J(K(z),L(z)) = z$$
$$K(J(x,y)) = x$$
$$L(J(x,y)) = y$$

and if $K(z) = K(z')$ and $L(z) = L(z')$, then $z = z'$.

PROOF: Let $J(x,y) = \frac{1}{2}((x+y)^2 + 3x+y)$. Now $(x+y)^2+3x+y = (x+y)(x+y)+(x+y)+2x = (x+y)(x+y+1)+2x$, which is always even. Hence, $J(x,y)$ is always an integer, and so is a well defined function. $J(x,y)$ is recursive, since

$$J(x,y) = \frac{\{(x+y)^2+U_1^2(x,y)+U_1^2(x,y)+U_1^2(x,y)+U_2^2(x,y)\}}{S(S(N(U_1^2(x,y))))} .$$

Suppose that z,x,y are numbers such that

$$2z = (x+y)^2+3x+y \qquad (1)$$

Then $8z+1=4(2z)+1=4((x+y)^2+3x+y)+1=4(x+y)^2+12x+4y+1$

$$=((2x+2y)^2+2(2x+2y)+1)+8x=(2x+2y+1)^2+8x.$$

Therefore, $(2x+2y+1)^2 \leq 8z+1=(2x+2y+1)^2+8x<(2x+2y+1)^2+4(2x+2y+1)+4$

$$= (2x+2y+3)^2$$

And so $2x+2y+1 \leq \sqrt{8z+1} < 2x+2y+3$.

Hence $[\sqrt{8z+1}]$ is either $2x+2y+1$ or $2x+2y+2$.

$[\sqrt{8z+1}] + 1$ is either $2x+2y+2$ or $2x+2y+3$. Therefore,

$$[([\sqrt{8z+1}]+1)/2 = x+y+1$$

And so $x+y = [([\sqrt{8z+1}] + 1)/2] - 1$ (2)

Thus by (1), $3x+y = 2z - ([([\sqrt{8z+1} + 1])/2] - 1)^2$ (3)

Since $\det\begin{vmatrix}1 & 1\\ 3 & 1\end{vmatrix} \neq 0$, equations (2) and (3) show that for a given z, there can be at most one pair, (x,y), satisfying (1).

If such x and y exist, they can be calculated by recursive functions. If we write

$$Q_1(z) = [([\sqrt{8z+1}]+ 1)/2] \dot{-} 1,$$
$$Q_2(z) = 2z \dot{-} (Q_1(z))^2 ,$$

then, clearly, $Q_1(z)$ and $Q_2(z)$ are recursive functions, and (2) and (3) yield

$$x+y = Q_1(z)$$
$$3x+y = Q_2(z)$$

which can be solved to give $x = [(Q_2(z) \dot{-} Q_1(z))/2] = K(z)$

$$y = Q_1(z) \dot{-} [(Q_2(z) \dot{-} Q_1(z))/2] = L(z)$$

where $K(z)$ and $L(z)$ are recursive functions. Thus if x,y,z satisfy (1), then $x = K(z)$, $y = L(z)$. Now if x and y are chosen arbitrarily, $z = J(x,y)$ satisfies (1). And so if we choose any pair (x,y) there will be a z which will satisfy our three desired relations; namely

$$J(K(z),\ L(z)) = z$$
$$K(J(x,y) = x$$
$$L(J(x,y)) = y.$$

It only remains to show that $K(z)$ and $L(z)$ are total functions, i.e. that for any natural number value of z, there exist x and y satisfying (1). Let z be any number. Let r be the largest number such that $1+2+\ldots+r \leq z$. Let $x = z - (1+2+\ldots+r)$. Then $x \leq r$, for if $x \geq r+1$, then $1+2+\ldots+r+(r+1) > z$, contradicting our choice of r. Let $y = r-x$. Then, $z = (1+2+\ldots+(x+y)) + x = 1/2(x+y)(x+y+1)+x$. And so $2z = (x+y)(x+y+1)+2x = (x+y)^2+3x+y$, and (1) is satisfied.

We have proved the existence of the three recursive functions in the theorem. Suppose $K(z) = K(z')$ and $L(z)=L(z')$. Then $z = J(k(z),L(z)) = J(K(z'),L(z')) = z'$. Thus $z = z'$ and the theorem is proved. □

Next, we consider the problem of setting up a similar recursive correspondence between the natural numbers and the set of all the finite sequences, of whatever length, of integers.

THEOREM 5.8. Let $a_0,a_1,\ldots,a_n$ be any finite sequence of numbers. Then there are numbers u and v such that $R(u,1+v(i+1)) = a_i$ for $i = 0,1,2,\ldots,n$. (Recall that $R(x,y)$ is the remainder on dividing x by y, and is a recursive function).

We begin with a number theoretic lemma

LEMMA 5.1. Let v be divisible by the numbers $1,2,\ldots,n$. Then the numbers $1+v(i+1)$, $i = 0,1,2,\ldots,n$, are relatively prime in pairs.

PROOF. Let $m_i = 1+v(i+1)$. Since v is divisible by $1,2,\ldots,n$, any divisor of m_i, other than 1, must be greater than n. Now suppose that $d|m_i$ and $d|m_j$ and $i>j$. Then $d|(i+1)m_j - (j+1)m_i$. But $(i+1)m_j - (j+1)m_i =$

$$(i+1)(1+v(j+1)) - (j+1)(1+v(i+1) = ((i+1)+v(i+1)(j+1)) - ((j+1)+v(i+1)(j+1)) = i-j.$$

Thus $d|i-j$. But $0<i-j\leqq n$. Hence $d=1$. □

Proof of Theorem: Let A be the largest of the numbers $a_0,a_1,\ldots,a_n$, and let $v = 2A\cdot n!$. Let $m_i = 1+v(i+1)$.

Then by the lemma, the m_i are relatively prime in pairs. Also $a_i<v<m_i$ for $i = 0,1,2,\ldots,n$. Now, by the Chinese Remainder Theorem, there is a number u such that

$$u \equiv a_i \pmod{m_i} \qquad i = 0,1,2,\ldots,n.$$

That is, $R(u,m_i) = R(a_i,m_i)$, $i = 0,1,2,\ldots,n$. But $a_i<m_i$. Hence, $R(a_i,m_i) = a_i$. Thus

$$r(u,1+v(i+1)) = R(u,m_i) = R(a_i,m_i) = a_i. \quad \square$$

THEOREM 5.9. There are recursive functions $T_i(w)$ such that, if $a_0,a_1,\ldots,a_n$, are any numbers whatever, there exists a number w_0 such that $T_i(w_0) = a_i$, $i = 0,1,2,\ldots,n$.

PROOF. Define $T_i(w)$ by the equation

$$T_i(w) = R(K(w),\ 1+(L(w)(i+1))).$$

Clearly $T_i(w)$ is recursive. Let us be given the integers $a_0,a_1,\ldots,a_n$. Then by Theorem 5.2, there exist numbers u and v such that $R(u,1+v(i+1)) = a_i$, $i = 0,1,2,\ldots,n$. Let $w_0 = J(u,v)$. Then

$$\begin{aligned} T_i(w_0) &= R(K(J(u,v)),1+L(J(u,v))\cdot(i+1)) \\ &= R(u,1+v(i+1)) \\ &= a_i \qquad\qquad \text{for } i = 0,1,\ldots,n. \quad \square \end{aligned}$$

We are finally ready to prove that functions constructed from recursive functions by primative recursion are themselves recursive.

THEOREM 5.10. Let $h(X^{(n+1)})$ be obtained from $f(X^{(n)})$, $g(X^{(n+2)})$ by primative recursion. If f and g are recursive, then so is h.

PROOF: By Theorem 5.9., for each choice of $X^{(n)}$ and y, there exist at least one number w_0 such that $T_i(w_0) = h(i,X^{(n)})$, $i = 0,1,2,\ldots,y$. Hence,

$$h(y,X^{(n)}) = T_y(\min_w[(T_0(w)=f(X^{(n)}))\wedge \overset{y-1}{\underset{z=0}{\forall}}(T_{z+1}(w) = g(z,T_z(w),X^{(n)}))])$$

Now, to say that a condition holds for all numbers z less than y is equivalent to saying that y is the least number for which it could fail. That is

$$h(y,X^{(n)}) = T_y(\min_w[\{T_0(w) = f(X^{(n)})\} \wedge \{y = \min_z[(T_{z+1}(w) \neq g(z,T_z(w),X^{(n)}))\ (z=y)]\}]).$$

Let $H(y,w,X^{(n)}) = \min_z[(T_{z+1}(w)\neq g(z,T_z(w),X^{(n)}))\vee(z=y)]$. Then $H(y,w,X^{(n)}) = \min_z[\,|z-y|\cdot\alpha(|T_{z+1}(w)-g(z,T_z(w),X^{(n)})|)=0]$. So $H(y,w,X^{(n)})$ is recursive. But

$$\begin{aligned}h(y,X^{(n)} &= T_y(\min_w[(T_0(w)=f(X^{(n)}))\wedge(y=H\ y,w,X^{(n)}))]) \\ &= T_y(\min_w[|T_0(w)-f(X^{(n)})|+|y-H(y,w,X^{(n)})|=0]).\end{aligned}$$

And so $h(y,X^{(n)})$ is recursive. □

COROLLARY 5.11. The functions x^y and x! are recursive.

PROOF. Let $f(x) = S(N(x)) = 1$ and $g(y,u,v) = uv$. Then clearly f and g are recursive. By Theorem 5.10, so is $h(y,x)$ which satisfies

$$h(0,x) = f(x) = 1$$
$$h(y+1,x) = g(y,h(y,x),x) = xh(y,x).$$

But $h(y,x) = x^y$ satisfies this pair of equations, and so x^y is recursive.

Let $f(y) = 1$ and $g(x,u,v) = (x+1)u$. Then clearly f and g are recursive and so $h(x,y)$ is recursive where $h(x,y)$ satisfies $h(0,y) = f(y) = 1$

$$h(x+1,y) = g(x,h(x,y),v) = (x+1)h(x,y).$$

But $h(x,y) = x!$ satisfies the above equations and so $x!$ is recursive. □

COROLLARY 5.12. If $f(k,X^{(p)})$ is recursive, so are

$$g(n,X^{(p)}) = \sum_{k=0}^{n} f(k,X^{(p)})$$

and $$h(n,X^{(p)}) = \prod_{k=0}^{n} f(k,X^{(p)}).$$

PROOF. $g(0,X^{(p)}) = f(0,X^{(p)})$

$$g(n+1,X^{(p)}) = g(n,X^{(p)}) + f(n+1,X^{(p)})$$

$$h(0,X^{(p)}) = f(0,X^{(p)})$$
$$h(n+1,X^{(p)}) = h(n,X^{(p)}) \cdot f(n+1,X^{(p)}).$$ □

THEOREM 5.13. Let R and S be recursive sets of n-tuples. Then so are $R \cup S$, $R \cap S$, and $\bar{R}$ (the complement of R).

PROOF; $C_{R\cup S}(X^{(n)}) = C_R(X^{(n)}) \cdot C_S(X^{(n)})$

$$C_{R\cap S}(X^{(n)}) = \alpha(\alpha(C_R(X^{(n)}) + C_S(X^{(n)})))$$
$$C_{\bar{R}}(X^{(n)}) = \alpha(C_R(X^{(n)})) \quad .$$ □

THEOREM 5.14. Let P and Q be recursive n-ary predicates. Then so are $P \vee Q$, $P \wedge Q$, and $\sim P$.

PROOF. We note that

$$\{X^{(n)} | P(X^{(n)}) \vee Q(X^{(n)})\} = \{X^{(n)} | P(X^{(n)})\} \cup \{X^{(n)} | Q(X^{(n)})\},$$

$$\{X^{(n)} | P(X^{(n)}) \wedge Q(X^{(n)})\} = \{X^{(n)} | P(X^{(n)})\} \cap \{X^{(n)} | Q(X^{(n)})\},$$

and $\{X^{(n)} | \sim P(X^{(n)})\} = \overline{\{X^{(n)} | P(X^{(n)})\}},$ [1]

and then apply Theorem 5.13. □

THEOREM 5.15. If $P(y,X^{(n)})$ is an (n+1)-ary recursive predicate, then so are

$$\overset{z}{\underset{y=0}{\exists}} P(y,X^{(n)}) \quad \text{and} \quad \overset{z}{\underset{y=0}{\forall}} P(y,X^{(n)}).$$

PROOF. Let $Q(z,X^{(n)}) \leftrightarrow \overset{z}{\underset{y=0}{\exists}} P(y,X^{(n)})$. Then we note that $C_Q(z,X^{(n)}) = \prod_{y=0}^{z} C_P(y,X^{(n)})$, and apply corollary 5.12 to get the first part.

To get the second part, we note that

$$\overset{z}{\underset{y=0}{\forall}} P(y,X^{(n)}) \leftrightarrow \sim \overset{z}{\underset{y=0}{\exists}} \sim P(y,X^{(n)})$$

and apply Theorem 5.14 and the first part. □

We should note that the boundedness of the quantifier is absolutely essential to this result. In fact, as we shall see later, there exists a recursive predicate $P(y,x)$ such that $\underset{y}{\exists} P(y,x)$ is not recursive. (The very thing we're looking

[1] If S is a set of numbers or of n-tuples, then $\overline{S}$ denotes the complement of S.

for!)

DEFINITION 5.6. Let $P(y,X^{(n)})$ be an $(n+1)$-ary predicate. Then, by $f(z,X^{(n)}) = \mathop{M}\limits_{y=0}^{z} P(y,X^{(n)})$ we mean the $(n+1)$-ary total function that satisfies the equation

$$f(z,X^{(n)}) = \min_y[y\leqq z \wedge P(y,X^{(n)})]$$

where this is defined, and $f(z,X^{(n)}) = 0$ elsewhere.

THEOREM 5.16. If $P(y,X^{(n)})$ is recursive, then so is $f(z,X^{(n)}) = \mathop{M}\limits_{y=0}^{z} P(y,X^{(n)})$

PROOF: Claim:

$$\mathop{M}\limits_{y=0}^{z} P(y,X^{(n)}) = \alpha\Big(\prod_{y=0}^{z} C_p(y,X^{(n)})\Big) \cdot \sum_{t=0}^{z}\prod_{y=0}^{t} C_p(y,X^{(n)}).$$

If the claim is true then it is clear that $\mathop{M}\limits_{y=0}^{z} P(y,X^{(n)})$ is recursive.

Consider the right hand side of the equation in the claim. Call it R. Assume we are given a specific $(n+1)$-tuple, $(z,X^{(n)})$. If there is no $y\leqq z$ such that $P(y,X^{(n)})$ holds, then $C_p(y,X^{(n)}) = 1$ for all $y\leqq z$, and so $\prod_{y=0}^{z} C_p(y,X^{(n)}) = 1$. Thus $\alpha(\prod_{y=0}^{z} C_p(y,X^{(n)})) = 0$, and so $R = 0 = \mathop{M}\limits_{y=0}^{z} P(y,X^{(n)})$.

If there is a $y\leqq z$ such that $P(y,X^{(n)})$ holds, then let y_0 be the least such y. Then $C_p(y_0,X^{(n)}) = 0$, so $\prod_{y=0}^{z} C_p(y,X^{(n)}) = 0$, and so $\alpha(\prod_{y=0}^{z} C_p(y,X^{(n)})) = 1$.

Now, for all $y<y_0$, $P(y,X^{(n)})$ does not hold, since y_0 is the least y for which it does hold. So for all $y<y_0$, $C_p(y,X^{(n)}) = 1$. Thus for all $t<y_0$, i.e. for $t=0,1,\ldots,y-1$, $\prod_{y=0}^{t} C_p(y,X^{(n)}) = 1$. Note that there are y_0 such t's. However, $C_p(y_0,X^{(n)}) = 0$, and so if $t\geqq y_0$, $\prod_{y=0}^{t} C_p(y,X^{(n)}) = 0$. Thus if we consider $\sum_{t=0}^{z} \prod_{y=0}^{t} C_p(y,X^{(n)})$ we note that each $t<y_0$ (and there are y_0 of them) will contribute 1 to the sum, and each $t\geqq y_0$ will contribute 0. Thus $\sum_{t=0}^{z} \prod_{y=0}^{t} C_p(y,X^{(n)}) = y_0$. Now $R = \alpha(\prod_{y=0}^{z} C_p(y,X^{(n)})) \cdot \sum_{t=0}^{z} \prod_{y=0}^{t} C_p(y,X^{(n)}) = 1\cdot y_0 = y_0 = \overset{z}{\underset{y=0}{}} P(y,X^{(n)})$, and the claim is proved. □

THEOREM 5.17. The predicates $x=y$, $x\leqq y$, $x<y$, $x\geqq y$, and $x>y$ are all recursive.

PROOF. The characteristic function of $x\geqq y$ is $\alpha(\alpha(y\dot{-}x))$ and so it is recursive. The recursiveness of the other predicatesfollow from the following equivalences:

$$x\leqq y \leftrightarrow y\geqq x,$$

$$x>y \leftrightarrow \sim(x\leqq y),$$

$$x<y \leftrightarrow \sim(x\geqq y),$$

$$\text{and} \quad x=y \leftrightarrow x\leqq y\wedge x\geqq y. \qquad \square$$

We are now able to list some more recursive functions and predicates which we will use in our later work. Each function or predicate is followed by a definition, if necessary, and a formula which proves it to be recursive.

(1) $y|x$. y is a divisor of x.

$$y|x \leftrightarrow \underset{z=0}{\overset{x}{\exists}} (x=yz)$$

(2) Prime (x). x is a prime number.

$$\text{Prime } (x) \leftrightarrow (x>1) \wedge \underset{z=0}{\overset{x}{\forall}} ((z=1) \vee (z=x) \vee \sim (z|x)).$$

(3) $Pr(x)$. $Pr(x)$ = the nth prime in order of magnitude where we arbitrarily take the 0th prime equal to 0.

$$Pr(0) = 0$$

$$Pr(n+1) = \underset{y=0}{\overset{Pr(n)!+1}{M}} (\text{Prime}(y) \wedge y>Pr(n))$$

We are finally ready to prove

THEOREM 5.18. The predicates $T_n(z,x_1,\ldots,x_n,y)$ are recursive, for all numbers n.

Proof. We have shown, up to this point, that certain predicates and functions are recursive, and we have also shown a number of methods of constructing recursive predicates and functions from predicates and functions which we already know to be recursive. In this proof, then, we will show how the predicates $T_n(z,x_1,\ldots,x_n,y)$ can be constructed with these methods from known recursive functions and predicates. The proof simply consists of a list of functions and predicates, culminating in the predicates $T_n(z,x_1,\ldots,x_n,y)$. Each function or predicate is accompanied by a definition and then a formula which proves the function or predicate to be recursive.

Group I. Functions and Predicates Which Concern Gödel Numbers of Expressions and Sequences of Expressions:

(1) $n \text{ Gl } x$.

$n \text{ Gl } x$ is a binary function.

If $x = gn(M)$, where M is an expression consisting of the symbols $\delta_1, \delta_2, \ldots, \delta_p$, in that order, then if $0<n\leq p$, $n \text{ Gl } x$ is the number associated with the symbol δ_n, whereas if $n = 0$ or $n > p$, then $n \text{ Gl } x = 0$.

If x is the Gödel number of the sequence of expressions $M_1, M_2, \ldots, M_p$, then if $0<n\leq p$, $n \text{ Gl } x=gn(M_n)$, whereas if $n=0$ or $n>p$, then $n \text{ Gl } x = 0$.

If x is not a Godel number, we don't care how it's defined, as long as it's recursive.

$$n \text{ Gl } x = \overset{x}{\underset{y=0}{M}} [Pr(n)^y | x \wedge \sim (Pr(n)^{y+1} | x)].$$

(2) $£(x)$.

If $x = gn(M)$, then $£(x)$ is the number of symbols occurring in M. If x is the Gödel number of a sequence of expressions, then (x) is the number of expressions in the sequence. If x is not a Gödel number we don't care.

$$£(x) = \overset{x}{\underset{y=0}{M}} [(y \text{ Gl } x>0) \wedge \overset{x}{\underset{i=0}{\forall}} ((y+i+1) \text{Gl } x=0)]$$

(3) $GN(x)$.

$GN(x)$ is a predicate which holds if and only if there exist <u>positive</u> integers a_k, $1\leq k\leq n$, such that

$$x = \prod_{k=1}^{n} Pr(k)^{a_k}$$

Note that it is not necessarily true that x is a Gödel number if GN(x) holds, but it is true that x is not a Gödel number if GN(x) does not hold.

$$GN(x) \leftrightarrow \sim \exists_{y=1}^{\pounds(x)} [(y\ Gl\ x=0) \wedge ((y+1) Gl\ x \neq 0)]$$

(4) Term (x,z)

Term (x,z) holds if and only if $z = \prod_{k=1}^{n} Pr(k)^{a_k}$ for suitable $a_k>0$, and $x=a_k$ for some k, $1 \leq k \leq n$.

$$Term\ (x,z) \leftrightarrow GN(z) \wedge \exists_{n=0}^{\pounds(x)} [(x=nGlz) \wedge (n \neq 0)]$$

(5) x*y

If M and N are expressions, then gn(MN) = gn(M)*gn(N). If x and y are Gödel numbers of the sequences of expressions $M_1,\ldots,M_n$, and $N_1,\ldots,N_p$, respectively, then x*y is the Gödel number of the sequence $M_1,\ldots,M_n,N_1,\ldots,N_p$.

$$x*y = x \cdot \prod_{i=0}^{\pounds(y) \dot{-} 1} Pr(\pounds(x)+i+1)^{(i+1)Gl\ y} \quad .$$

Group II. Functions and Predicates Which Concern the Basic Structure of Turing Machines:

(6) IC (x)

IC (x) holds if and only if x is a number assigned to one of the q_i.

$$IC(x) \leftrightarrow \exists_{y=0}^{x} (x=4y+9)$$

(7) Al (x)

Al (x) holds if and only if x is a number assigned to one of the s_i

$$\mathrm{Al}(x) \leftrightarrow \overset{x}{\underset{y=0}{\exists}} (x=4y+7)$$

(8) Re (x)

Re (x) holds if and only if x is the number assigned to a symbol which may appear as the third symbol in a quadruple, i.e. x is the number assigned to an s_i or R or L.

$$\mathrm{Re}(x) \leftrightarrow \mathrm{Al}(x) \vee (x=z) \vee (x=5).$$

(9) Quad (x)

Quad (x) holds if and only if x is the Gödel number of a quadruple.

$$\mathrm{Quad}(x) \leftrightarrow \mathrm{GN}(x) \wedge (\pounds(x)=4) \wedge \mathrm{IC}(1\mathrm{Gl}\ x) \wedge \mathrm{Al}(2\ \mathrm{Gl}\ x)$$
$$\wedge \mathrm{Re}(3\ \mathrm{Gl}\ x) \wedge \mathrm{IC}(4\ \mathrm{Gl}\ x).$$

(10) Inc (x,y)

Inc(x,y) holds if and only if x and y are Gödel numbers of two distinct quadruples beginning with the same two symbols.

$$\mathrm{Inc}(x,y) \leftrightarrow \mathrm{Quad}(x) \wedge \mathrm{Quad}(y) \wedge (1\ \mathrm{Gl}\ x=1\ \mathrm{Gl}\ y)$$
$$\wedge (2\ \mathrm{Gl}\ x=2\ \mathrm{Gl}\ y) \wedge (x \neq y)$$

(11) TM(x)

TM(x) holds if and only if x is the Gödel number of a Turing machine.

$$\mathrm{TM}(x) \leftrightarrow \mathrm{GN}(x) \wedge \overset{\pounds(x)}{\underset{n=1}{\forall}} [\mathrm{Quad}(n\mathrm{Gl}x) \wedge \overset{\pounds(x)}{\underset{m=1}{\forall}} \neg(\mathrm{Inc}(n\mathrm{Gl}x, m\mathrm{Gl}x))]$$

(12) $MR(n)$

We want $MR(n)$ to be $gn(\bar{n}) = gn(1^{n+1}) = gn({s_1}^{n+1})$

$MR(0) = 2^{11}$

$MR(n+1) = 2^{11} * MR(n)$

(13) $ID(x)$

$ID(x)$ holds if and only if x is the Gödel number of an instantaneous description

$$ID(x) \leftrightarrow GN(x) \wedge \overset{\pounds(x) \dot{-} 1}{\underset{n=1}{\exists}} \{IC(nGlx) \wedge \overset{\pounds(x)}{\underset{m=1}{\forall}} [(m=n) \vee Al(mGlx)]\}.$$

(14) $Init_n(x_1,\ldots,x_n)$

This is a class of functions, one for each value of $n>0$. We want $Init_n(x_1,\ldots,x_n) = gn(q_1(\overline{x_1,\ldots,x_n}))$

$$Init_n(x_1,\ldots,x_n) = 2^9 * MR(x_1) * 2^7 * MR(x_2) * 2^7 * \ldots * 2^7 * MR(x_n).$$

Group III. Functions and Predicates which Concern the Relation "→" in Turing Machines.

(15) $Yield_1(x,y,z)$

$Yield_1(x,y,z)$ holds if and only if x and y are Gödel numbers of instantaneous descriptions, z is a Gödel number of a Turing machine Z, and $Exp(x) \rightarrow Exp(y)$ in Z, under Case 1 of Definition 2.7.

$$Yield_1(x,y,z) \leftrightarrow ID(x) \wedge ID(y) \wedge TM(z)$$

$$\wedge \exists_{F=0}^{x} \exists_{G=0}^{x} \exists_{r=0}^{x} \exists_{s=0}^{x} \exists_{t=0}^{y} \exists_{n=0}^{y} [(x=F*2^{r}*2^{s}*G) \wedge (y=F*2^{t}*2^{u}*G)$$

$$\wedge IC(r) \wedge IC(t) \wedge Al(s) \wedge Al(u) \wedge Term(2^{r} \cdot 3^{s} \cdot 5^{u} \cdot 7^{t},z)]$$

(16) $Yield_2(x,y,z)$

$Yield_2$ is like $Yield_1$, but deals with case 2 of Definition 2.7.

$$Yield_2(x,y,z) \leftrightarrow ID(x) \wedge ID(y) \wedge TM(z)$$

$$\wedge \exists_{F=0}^{x} \exists_{G=0}^{x} \exists_{r=0}^{x} \exists_{s=0}^{x} \exists_{t=0}^{x} \exists_{u=0}^{y} [(x=F*2^{r}*2^{s}*2^{t}*G)$$

$$\wedge (y=F*2^{s}*2^{u}*2^{t}*G) \wedge IC(r) \wedge IC(n) \wedge Al(s) \wedge Al(t)$$

$$\wedge Term(2^{r} \cdot 3^{s} \cdot 5^{3} \cdot 7^{u},z)]$$

(17) $Yield_3(x,y,z)$

$Yield_3$ is like $Yield_1$, but deals with case 3 of Definition 2.7.

$$Yield_3(x,y,z) \leftrightarrow ID(x) \wedge ID(y) \wedge TM(z)$$

$$\wedge \exists_{F=0}^{x} \exists_{r=0}^{x} \exists_{s=0}^{x} \exists_{t=0}^{x} [(x=F*2^{r}*2^{s}) \wedge (y=F*2^{s}*2^{t}*2^{7})$$

$$\wedge IC(r) \wedge IC(t) \wedge Al(s) \wedge Term(2^{r} \cdot 3^{s} \cdot 5^{3} \cdot 7^{t},z)]$$

(18) $Yield_4(x,y,z)$

$Yield_4$ is like $Yield_1$, but deals with case 4 of Definition 2.7.

$$Yield_4(x,y,z) \leftrightarrow ID(x) \wedge ID(y) \wedge Tm(z)$$

$$\wedge \overset{x}{\underset{F=0}{\exists}} \overset{x}{\underset{G=0}{\exists}} \overset{x}{\underset{r=0}{\exists}} \overset{x}{\underset{s=0}{\exists}} \overset{x}{\underset{t=0}{\exists}} \overset{y}{\underset{u=0}{\exists}} [(x=F*2^r*2^s*2^t*G)$$

$$\wedge (y=F*2^u*2^r*2^t*G)$$

$$\wedge\ IC(s)\wedge IC(u)\wedge Al(r)\wedge Al(t)\wedge Term(2^s\cdot 3^t\cdot 5^5\cdot 7^u,z)]$$

(19) $Yield_5(x,y,z)$

$Yield_5$ is like $Yield_1$ but deals with case 5 of Definition 2.7.

$$Yield_5(x,y,z) \leftrightarrow ID(x)\wedge ID(y)\wedge TM(z)$$

$$\wedge \overset{x}{\underset{G=0}{\exists}} \overset{x}{\underset{r=0}{\exists}} \overset{x}{\underset{s=0}{\exists}} \overset{y}{\underset{t=0}{\exists}} [(x=2^r*2^s*G)\wedge (y=2^t*2^7*2^s*G)$$

$$\wedge IC(r)\wedge IC(t)\wedge Al(s)\wedge Term(2^r\cdot 3^s\cdot 5^5\cdot 7^t,z)]$$

(20) Yield (x,y,z)

Yield (x,y,z) holds if and only if x and y are Gödel numbers of instantaneous descriptions, z is a Gödel number of a Turing machine, Z, and Exp(x) → Exp(y) in Z.

$$Yield\ (x,y,z) \leftrightarrow Yield_1(x,y,z)\vee Yield_2(x,y,z)$$

$$\vee Yield_3(x,y,z)\vee Yield_4(x,y,z)$$

$$\vee Yield_5(x,y,z)$$

(21) $Fin(x,z)$

$Fin(x,z)$ holds if and only if z is a Gödel number of a Turing machine Z, and x is the Gödel number of an instantaneous description which is terminal with respect to Z.

$$Fin(x,z) \leftrightarrow ID(x)\ TM(z)$$

$$\wedge \overset{x}{\underset{F=0}{\exists}} \overset{x}{\underset{G=0}{\exists}} \overset{x}{\underset{r=0}{\exists}} \overset{x}{\underset{s=0}{\exists}} \{(x=F*2^r*2^s*G)\wedge IC(r)\wedge Al(s)$$

$$\wedge \overset{\pounds(x)}{\underset{n=1}{\exists}} [(1Gl(nGlz)\neq r) \vee (2Gl(nGlz)\neq s)]\}$$

(22) Comp(y,z)

Comp(y,z) holds if and only if z is a Gödel number of a Turing machine Z, and y is the Gödel number of a computation of Z.

$$\mathrm{Comp}(y,z) \leftrightarrow \mathrm{TM}(z) \wedge \mathrm{GN}(y)$$

$$\wedge \overset{\pounds(y)\dot{-}1}{\underset{n=1}{\forall}} [\mathrm{Yield}(n\ Gl\ y,(n+1)Gly,z)] \wedge \mathrm{Fin}(\pounds(y)Gly,z)$$

(23) Finally we have

$$T_n(z,x_1,\ldots,x_n,y) \leftrightarrow \mathrm{Comp}(y,z) \wedge (1Gly=\mathrm{Init}_n(x_1,\ldots,x_n)).$$

Thus $T_n(z,x_1,\ldots,x_n,y)$ is a recursive predicate. □

As we outlined earlier, we now define W(x,y) by $W(x,y) \leftrightarrow T(x,x,y)$. It follows immediately that W(x,y) is recursive and hence computable. It remains only to prove that $\underset{y}{\exists} W(x,y)$ is not computable and hence not recursive and then we can formally define our machine Z_0 with unsolvable halting problem.

THEOREM 5.19. $\underset{y}{\exists} W(x,y)$ is a non-computable predicate.

PROOF: Suppose $\underset{y}{\exists} W(x,y)$ is computable. Then so is its characteristic function. Then there exists a Turing machine, Z_1, such that

$$\Psi_{Z_1}(x) = 0 \text{ if } \underset{y}{\exists} W(x,y) \text{ holds.}$$

$$= 1 \text{ if } \underset{y}{\exists} W(x,y) \text{ doesn't hold.}$$

By Lemma 1 of Chapter III, there is a Turing machine Z_1' which is 1-regular and such that $\text{Res}_{Z_1'}[q_1(\bar{m})] = q_{\theta(Z_1')}\overline{\Psi_{Z_1}(m)}$..

Let Z_2 be the Turing machine whose quadruples consist of all the quadruples of Z_1' in addition to the quadruples:

$$q_{\theta(Z_1')}1\ R\ q_{\theta(Z_1')+1}$$

$$q_{\theta(Z_1')+1}B\ S_p q_{\theta(Z_1')+1}$$

$q_{\theta(Z_1')+1}S_pB\ q_{\theta(Z_1')+1}$, where p is such that S_p is not in the alphabet of Z_1'.

Now if m is such that $\underset{y}{\exists} W(m,y)$ doesn't hold, then $\Psi_{Z_1}(m)=1$ and so, with respect to Z_2

$$\begin{aligned} q_1(\bar{m}) &\rightarrow \ldots \\ &\rightarrow q_{\theta(Z_1')}\bar{1} \\ &= q_{\theta(Z_1')}11 \\ &\rightarrow 1q^{\theta}{}_{(Z_1')+1}{}^{1} \end{aligned}$$

which is terminal.

But if m is such that $\underset{y}{\exists} W(m,y)$ holds, then $\Psi_{Z_1}(m)=0$ and so, with respect to Z_2

$$
\begin{aligned}
q_1(\overline{m}) &\rightarrow \cdots \\
&\rightarrow q_{\theta(Z_1')}\overline{0} \\
&= q_{\theta(Z_1')}1 \\
&\rightarrow 1q_{\theta(Z_1')+1}B \\
&\rightarrow 1q_{\theta(Z_1')+1}S_p \\
&\rightarrow 1q_{\theta(Z_1')+1}B \\
&\rightarrow \cdots
\end{aligned}
$$

and a terminal instantaneous description is never reached.

Thus, there is a computation of Z_2 beginning with x if and only if $\underset{y}{\exists}W(x,y)$ doesn't hold, i.e. if and only if $\sim\underset{y}{\exists}W(x,y)$ holds. Now let z_2 be a Gödel number of Z_2. By definition of the predicate T, $T(z_2,z_2,y)$ holds if and only if y is the Gödel number of a computation of Z_2 beginning with $q_1(\overline{z_2})$. Thus $\underset{y}{\exists}T(z_2,z_2,y)$ holds if and only if there is some computation of Z_2 beginning with $q_1(\overline{z_2})$. But we argued above that such a computation exists if and only if $\sim\underset{y}{\exists}W(z_2,y)$ holds. Thus

$$\underset{y}{\exists}T(z_2,z_2,y) \leftrightarrow \sim\underset{y}{\exists}W(z_2,y).$$

But by definition of W, $W(z_2,y) \leftrightarrow T(z_2,z_2,y)$. And so $\sim\underset{y}{\exists}W(z_2,y) \leftrightarrow \sim\underset{y}{\exists}T(z_2,z_2,y)$.

Thus we have

$$\exists_y T(z_2,z_2,y) \leftrightarrow \sim \exists_y T(z_2,z_2,y);$$

clearly an impossible situation.

Thus $\exists_y W(x,y)$ cannot be computable. □

We now have $W(x,y)$ computable and $\exists_y W(x,y)$ non-computable. Let $w(x,y)$ be the characteristic function of $W(x,y)$. Then $w(x,y)$ is computable, and so $g(x) = \min_y[w(x,y)=0]$ is a partially computable function. Thus, there is a Turing machine Z_0 such that $\Psi_{Z_0}(x) = g(x)$.

THEOREM 5.20. The halting problem for Z_0 is unsolvable.

PROOF: Define the predicate $P_{Z_0}(x)$ to hold if and only if there is a computation of Z_0 beginning with $q_1(\overline{x})$. Then, by Church's Thesis, the halting problem for Z_0 has a solution if and only if $P_{Z_0}(x)$ is recursive.

Now $P_{Z_0}(x)$ holds if and only if $\Psi_{Z_0}(x)$ is defined, which happens if and only if $g(x)$ is defined, which happens if and only if $\min_y[w(x,y)=0]$ is defined, which happens if and only if there exists a number y such that $w(x,y)=0$, which happens if and only if there is a y such that $W(x,y)$ holds, which happens if and only if $\exists_y W(x,y)$ holds.

Thus, $P_{Z_0}(x) \leftrightarrow \exists_y W(x,y)$. But $\exists_y W(x,y)$ is not computable and hence not recursive. Thus $P_{Z_0}(x)$ is not recursive and Z_0 has unsolvable halting problem. □

CHAPTER VI

THE WEAK AND STRONG WORD PROBLEMS FOR SEMI-GROUPS.

In this chapter, we will present a single argument which will show that both the weak and the strong word problems for finitely presented semi-groups are unsolvable. The argument will in fact show that the unsolvability of the weak halting problem forTuring machines yields the unsolvability of the weak word problem for finitely presented semigroups, and the unsolvability of the strong halting problem yields the unsolvability of the strong word problem for semigroups.

More precisely, we show that there is a constructive correspondence between the set of all Turing machines and a set of finitely presented semigroups, i.e. we will give an effective procedure for constructing from each Turing machine, Z, a finitely presented semigroup, H_Z, such that any solution to the word problem for H_Z yields a solution to the halting problem for Z.

Suppose the weak word problem for finitely presented semigroups had a solution. Then given any Turing machine, Z, we use the effective procedure to construct the semigroup H_Z and then use our solution to the weak word problem for semigroups to get a solution to the halting problem for Z. Thus we would have a single procedure which could be applied to any Turing machine to solve the halting problem for that machine, contradicting the unsolvability of the weak halting problem.

Now, suppose the strong word problem for finitely presented semigroups had a solution. The unsolvability of the strong halting problem produces a Turing machine, Z_0, with unsolvable halting problem. We can then construct the semigroup H_{Z_0} and since any solution to the word problem for H_{Z_0} would yield a solution to the halting problem for Z_0, there can be no such solution. Thus we have a finitely presented semigroup with unsolvable word problem and so the strong word problem for semigroups is unsolvable.

We now go ahead with the construction of the semigroup H_Z from the Turing machine Z. To facilitate the construction we introduce the concept of semi-Thue and Thue systems. A *semi-Thue system*, T, is a pair (S,β) where S is a finite set of symbols of *generators* and β is a finite set of *rules*. A word in T is a finite sequence of generators from S (like a word in a semigroup; note that inverses don't exist as they do in groups.) Words are multipled by juxtaposition, as in free semigroups. A rule of T, i.e. an element of β, is of the form "A→B" where A and B are words in T. If C and D are words in T, then we write "C→D in T " if there are words A,B,P,Q in T, possibly empty, such that C is PAQ, D is PBQ and "A→B" is a rule of T. We write "E⇒F in T," where E and F are words in T, if there is a finite sequence $C_1, C_2, \ldots, C_n$

of words in T such that E is C_1, F is C_n and "$C_i \to C_{i+1}$ in T" for i=1,2,...,n-1. Where no ambiguity can result, we simply write "$C \to D$" or "$E \Rightarrow F$".

If a semi-Thue system, T, has the additional property that "$B \to A$" is a rule of T whenever "$A \to B$" is, then T is a <u>Thue system</u>, and we combine the rules, simply writing "$A \leftrightarrow B$". The rule "$B \to A$" is called the inverse of the rule "$A \to B$", and vice versa. The following theorems follow immediately from these definitions.

THEOREM 6.1. If T is a semi-Thue system, then $C \to D$ in T implies $C \Rightarrow D$ in T.

THEOREM .2. If T is a Thue system, then $C \to D$ in T if and only if $D \to C$ in T; and $E \Rightarrow F$ in T if and only if $F \Rightarrow E$ in T. In these cases we write "$C \leftrightarrow D$ in T" and "$E \Leftrightarrow F$ in T" respectively.

The reader may already have noticed that a Thue system is, for all intents and purposes, a semi-group. All we have to do is replace the symbols $\leftrightarrow$ and $\Leftrightarrow$ with the symbol =. We will, of course, make use of this fact in our construction.

An outline of our construction is as follows. For each Turing machine Z we will construct a semi-Thue system T_Z with the property that for each instantaneous description α of Z there corresponds a word A_α in T_Z such that if B is another word in T_Z where $A_\alpha \to B$ in T_Z, then B is A_β for some instantaneous description β of Z and $\alpha \to \beta$ in Z. Also

$A_\alpha \to A_\beta$ in T_Z whenever $\alpha \to \beta$ in Z.

Thus, there is a certain class of words in T_Z, corresponding to instantaneous descriptions of Z, such that T_Z acts on these words just as Z acts on its instantaneous descriptions.

We then construct the semi-Thue system T'_Z from T_Z by adding one new generator, q, (without subscript) and a number of new rules to T_Z, so that for the word A_α corresponding to an instantaneous description α of Z, we have $A_\alpha \Rightarrow q$ in T'_Z if and only if $A_\alpha \Rightarrow A_\delta$ in T_Z for some terminal instantaneous description δ of Z. Of course this latter occurrence will take place exactly when there is a computation of Z beginning with α. From T'_Z we construct the semigroup H_Z with the same generators such that $A_\alpha = q$ in H_Z if and only if there is a computation of Z beginning with α.

We proceed formally as follows. Let Z be any Turing machine. The set of generators of T_Z will consist of all symbols appearing in the alphabet of Z, including the symbol s_0 if it does not already appear in the alphabet of Z, all symbols appearing as internal configurations of Z, and the symbol s_h, where h is the smallest positive integer not already appearing as the subscript of some s in the alphabet of Z.

If α is any instantaneous description of Z, i.e. a finite sequence of s_j's with one q_i not at the right end, then we associate with α the word $s_h\alpha s_h$ in T_Z, which we call A_α. A word constructed in this manner from an instantaneous description of Z is called a d-word in T_Z.

The rules of T_Z are constructed in groups, each group to correspond to a quadruple of Z. If $q_is_jLq_l$ is a quadruple of Z, then the rules $s_nq_is_j\rightarrow q_ls_ns_j$, one for each generator of T_Z of the form s_n, excluding s_h, (i.e. for each s_n in the alphabet of Z, including s_0), will all be rules of T_Z. In addition, the rule $s_hq_is_j\rightarrow s_hq_ls_os_j$ will be a rule of T_Z. We recall that the quadruple $q_is_jLq_l$ of Z acts on an instantaneous description α when α involves q_i and q_i appears immediately to the left of the symbol s_j, and has the effect of moving the q_i one space to the left and changing it to q_l. Our semi-Thue system rules act on the corresponding d-words and have the same effect on them. We have the rule $s_hq_is_j\rightarrow s_hq_ls_os_j$ instead of the rule $s_hq_is_j\rightarrow q_ls_hs_j$ so that the s_h will always remain at the left end of the word. We construct similar groups of rules from the quadruples of the forms $q_is_jRq_l$ and $q_is_js_kq_l$. If $q_is_jRq_l$ is a quadruple of Z then the rules $q_is_js_n\rightarrow s_jq_ls_n$, one for each generator of T_Z of the form s_n, excluding s_h, and the rule $q_is_js_h\rightarrow s_jq_ls_os_h$ will all be rules of T_Z. Finally, if $q_is_js_kq_l$ is a quadruple of Z, it will act on an instantaneous description which has q_i immediately to the left of s_j and will have the effect of changing the s_j to s_k and the q_i to q_l. So we include the single rule $q_is_j\rightarrow q_ls_k$ among the rules of T_Z. The rules derived as above from the quadruples of Z are all the rules of T_Z.

Now we easily observe that if we apply any of the rules of T_Z to a d-word we get another d-word. By simple induction, then, we derive that if E is a d-word in T_Z and if $E \Rightarrow F$ in T_Z, then F is a d-word in T_Z. We note even more than this, though. From the way we have constructed the rules of T_Z we note that $A_\alpha \to A_\beta$ in T_Z if and only if $\alpha \to \beta$ in Z. Thus we can define A_α to be a terminal d-word in T_Z if α is a terminal instantaneous description in Z, and we will have the result that A is a terminal d-word in Z if and only if A is a d-word and there is no word B such that $A \to B$ in T_Z.

So if we consider the semi-Thue system T_Z restricted to d-words, it in a sense "does the same thing" as the Turing machine Z.

We now construct the semi-Thue system T_Z' from T_Z. The set of generators of T_Z' will include all the generators of T_Z in addition to the symbol q (no subscript). The set of rules of T_Z' will include all the rules of T_Z in addition to the following groups of rules:

(a) $q_is_j \to qs_j$ for all q_i and s_j, excluding s_h, that are generators of T_Z such that neither q_is_j, $s_hq_is_j$, $q_is_js_h$, $s_nq_is_j$ for some n, nor $q_is_js_n$ for some n, is the left hand word in a rule of T_Z;

$$\left.\begin{array}{ll} \text{(b)} & s_jqs_k \to q \\ \text{(c)} & s_hqs_k \to s_hq \\ \text{(d)} & s_jqs_h \to qs_h \end{array}\right\} \begin{array}{l} \text{for all generators} \\ \text{of } T_Z \text{ of the} \\ \text{form } s_j \text{ and } s_k \text{(excluding } s_h) \end{array}$$

and the single rule

$$\text{(e)} \quad s_h q s_h \rightarrow q$$

We note that if E is a d-word in T'_Z to which one of the rules of (a) may be applied then the effect of applying the rules of (a), (b), (c), (d), and (e) to E wherever possible is to first change the q_i of E to q and then to erase all the s_j's except s_h, giving the word $s_h q s_h$, and finally to erase the s_h's leaving the word q. Thus, if we can apply one of the rules of (a) to the d-word E in T'_Z, we get $E \Rightarrow q$ in T'_Z.

Next, we note that the rules of (a) are constructed such that one of them is applicable to the d-word E of T'_Z if and only if E is a terminal d-word of T_Z. As we already observed, E is a terminal d-word of T_Z if and only if $E = A_\alpha$ where α is a terminal instantaneous description of the Turing machine Z.

Combining the above results with our previous observations we immediately have the following theorem.

THEOREM 6.3. There is a computation of the Turing machine Z beginning with the instantaneous description α if and only if $A_\alpha \Rightarrow q$ in T'_Z.

Now let T_Z'' be the semi-Thue system whose generators are just those generators of T'_Z and whose set of rules consists of the inverses of all the rules of T'_Z. We notice immediately that $E \Rightarrow F$ in T_Z'' if and only if $F \Rightarrow E$ in T'_Z, where E and F are any words in T_Z''.

Next, let $T_Z{}^*$ be the Thue system whose generators are just those generators of T_Z' and whose set of rules consists of the rules of T_Z' together with their inverses, i.e. whose set of rules is the union of the set of rules of T_Z' and the set of rules of T_Z''.

THEOREM 6.4. If E is any d-word in $T_Z{}^*$ then $E \Leftrightarrow q$ in $T_Z{}^*$ if and only if $E \Rightarrow q$ in T_Z'.

PROOF: Obviously $E \Rightarrow q$ in $T_Z{}^*$ if $E \Rightarrow q$ in T_Z'. By THEOREM 6.2, $q \Rightarrow E$ in $T_Z{}^*$ if and only if $E \Rightarrow q$ in $T_Z{}^*$. Thus $E \Leftrightarrow q$ in $T_Z{}^*$ if $E \Rightarrow q$ in T_Z'.

On the other hand, $E \Rightarrow q$ in T_Z' if $q \Rightarrow E$ in T_Z''. Obviously $q \Rightarrow E$ in $T_Z{}^*$ if $E \Leftrightarrow q$ in $T_Z{}^*$. So it suffices to prove that $q \Rightarrow E$ in T_Z'' if $q \Rightarrow E$ in $T_Z{}^*$.

First we observe that if C is any d-word of T_Z', or any word of T_Z' consisting of s's with exactly one q and an s at either end,[1] then at most one rule of T_Z' is applicable to C. This is because if C is d-word then C is A_α for some instantaneous description α of Z. If any rule of T_Z' is applicable to C then either that rule is one of rules of (a) in which case only one can be applicable or that rule is one of the rules of T_Z in which case again only one can be applicable because the rules of T_Z all come from the quadruples of Z and at most one quadruple of Z can be applied to α.

[1] Remember that a word involving q is not a d-word, because a d-word is of the form $s_h \alpha s_h$ where α is an instantaneous description of Z.

Also, from the way the rules of (a) were chosen, none of them can be applicable to C if a rule of T_Z is. If C is a word involving q then the only possible rules of T_Z' that can be applicable to C are those of (b), (c), (d) and (e), and it is easy to see that at most one of them can be applicable to C.

Now suppose $q \Rightarrow E$ in T_Z*. Then there is a finite sequence of words $C_1, C_2, \ldots, C_n$ such that q is C_1 and E is C_n and $C_i \rightarrow C_{i+1}$ for $i=1,2,\ldots,n-1$, and such that C_i is either a d-word or a word consisting of s's with exactly one q and an s_h at either end, for $i=2,3,\ldots,n$. (Note that C_2 must be the word $s_h q s_h$.) Now let $V_1, V_2, \ldots, V_{n-1}$ be the sequence of rules of T_Z* such that V_i is the rule $A_i \rightarrow B_i$ of T_Z* and C_i is the word $P_i A_i Q_i$ for P_i and Q_i and C_{i+1} is the word $P_i B_i Q_i$, i.e. V_i is the rule used to get $C_i \rightarrow C_{i+1}$ in T_Z*, for $i=1,2,\ldots,n-1$. Then for each i, V_i is a rule of either T_Z' or T_Z . It is easy to see that no rule of T_Z' can be applied to q, and so V_1 must be a rule of T_Z . (In fact V_1 will be the rule $q \rightarrow s_h q s_h$.) Let m be the least number such that V_m is not a rule of T_Z''. Then V_m is a rule of T_Z' . Now V_m is the rule used to get $C_m \rightarrow C_{m+1}$. And we saw that V_m can be the only rule of T_Z' applicable to C_m, and obviously it can only be applied in one way. Thus if $C_m \rightarrow D$ in T_Z', D must be C_{m+1}. Now $C_{m-1} \rightarrow C_m$ in T_Z* and the rule which gives this is in T_Z'', so $C_{m-1} \rightarrow C_m$ in T_Z'' and thus $C_m \rightarrow C_{m-1}$ in T_Z', and so C_{m-1} is C_{m+1}. Thus the words C_m and C_{m+1} could be omitted from the sequence $C_1, C_2, \ldots, C_n$. In this manner we can eliminate all the steps involving application of rules not from T_Z and so $q \Rightarrow E$ in T_Z, and the theorem is proved. □

We are now ready to construct our semi-group. Let T be any Thue system, Let H_T be the semi-group with presentation (S;U) where S, the set of generators of H_T, is just the set of generators of T and where U, the set of defining relations of H_T, is such that A=B is in U if and only if $A \leftrightarrow B$ is a rule in T. Since defining relations of a semi-group are applied to words of the semi-group in the same way that rules of a Thue system are applied to words of the system, we see immediately that E=F in H_T if and only if $E \Leftrightarrow F$ in T, where E and F are any two words in H_T.

Now we are ready to finish the proof of the unsolvability of the weak and strong word problems for finitely presented semi-groups.

THEOREM 6.5. Given a Turing machine Z, there exists a finitely presented semigroup H_Z, such that a solution to the word problem for H_Z yields a solution to the halting problem for Z.

PROOF: Let H_Z be the semigroup $H_{T_Z^*}$ constructed as described above from the Thue system T_Z^* which is in turn constructed from Z. Let α be an arbitrary instantaneous description of Z. Then A_α is a word in T_Z, hence in T_Z', hence in T_Z^*, and hence in H_Z. By Theorem 6.3 there is a computation of Z beginning with α if and only if $A_\alpha \Rightarrow q$ in T_Z'. By Theorem

6.4., $A_\alpha = q$ in T'_Z if and only if $A_\alpha \Leftrightarrow q$ in $T_Z{}^*$. We observed that $A_\alpha \Leftrightarrow q$ in $T_Z{}^*$ if and only if $A_\alpha = q$ in H_Z. Thus there is a computation of Z beginning with α if and only if $A_\alpha = q$ in H_Z. But we can apply our solution to the word problem for H_Z to determine whether or not $A_\alpha = q$ in H_Z holds. Thus we have an effective process for determining whether or not there is a computation of Z beginning with α, and so we have a solution to the halting problem for Z. □

We have essentially accomplished what we set out to do. All that remains is to introduce Rotman's part of the proof of the word problem for groups. The first thing that we should note is that in a semigroup or group whose generators are all of the form q_i or s_j for some number i or j, a special word is a word Σ of the form Aq_iB where A and B are words in the s's alone and, in the case of a group, where Σ is a positive word, i.e. Σ is a word in the generators with no occurrences of the inverses of the generators.

Post's Theorem, as Rotman calls it, is the theorem which gives the existence of a finitely presented semigroup with unsolvable word problem. This semigroup is called, by Rotman, Post's semigroup, and would be our semigroup H_{Z_0}, where Z_0 is a Turing machine with unsolvable word problem.

Rotman shows that there is a finitely presented group G, with unsolvable word problem, and constructs this group from Post's semigroup. Thus Rotman shows the unsolvability of the strong word problem starting with the unsolvability of the strong word problem for finitely presented semigroups. The group that Rotman constructs is such that the solvability of its word problem would yield the solvability of the word problem for the semigroup from which it was constructed. As long as a semigroup has a finite number of generators of the form s_i or q_j for some number i or j, and a finite number of defining relations of the form $\Sigma_i = \Gamma_i$, where Σ_i and Γ_i are all special words, one can use Rotman's construction to obtain a group such that the solvability of the word problem for that group yields a solution to the word problem for that semigroup. So if the weak word problem for finitely presented groups were solvable we would have an effective procedure for solving the weak word problem for semigroups of this form. Now all the semigroups that we constructed from Turing machines have this form and so have unsolvable weak word problem. Thus Rotman's proof actually shows that the unsolvability of the weak word problem for finitely presented groups is obtainable from the unsolvability of the weak word problem for this class of semigroups.

BIBLIOGRAPHY

[1] Boone, W. W.: "An analysis of Turing's 'The Word Problem in Semigroups with Cancellation'", Annals of Mathematics, Vol. 67 (1958), pp. 195-202.

[2] Boone, W. W.: "The Word Problem", Annals of Mathematics, Vol. 70 (1959), pp. 207-265.

[3] Britton, J. L.: "Solution of the Word Problem for Certain Types of Groups", Proc. Glasgow Math. Assoc., Vol. 3 (1956), pp. 45-54.

[4] Britton, J. L.: "The Word Problem for Groups", Proc. London Math. Soc. (3), Vol. 8 (1958), pp. 493-506.

[5] Britton, J. L.: "The Word Problem", Annals of Mathematics, Vol. 77 (1963), pp. 16-32.

[6] Davis, Martin: Computability and Unsolvability, McGraw-Hill Book Co., New York, (1958).

[7] Dehn, Max: "Uber Unendliche Diskontinuierliche Gruppen", Mathematische Annalen, Vol. 71 (1911), pp. 116-144.

[8] Greendlinger,Martin: "Dehn Algorithm for the Word Problem", Communications of Pure and Applied Mathematics, Vol. 13 (1960), pp. 67-83.

[9] Greendlinger, Martin: "On Dehn's Algorithms for the Conjugacy and Word Problems, With Applications", Comm, Pure. Appl. Math., Vol. 13 (1960), pp. 641-677.

[10] Higman, G.: "Subgroups of Finitely Presented Groups", Journal of London Math. Soc., A, Vol. 262 (1961), pp. 455-475.

[11] Magnus, Wilhelm; Karrass, Abraham; and Solitar, Donald: Combinatorial Group Theory: Presentations of Groups in Terms of Generators and Relations, John Wiley&Sons, New York, (1966).

[12] Novikov, P. S.: "On the Algorithmic Unsolvability of the Word Problem in Groups" (Russian), Trudy Mat. Inst. Steklov, No. 44, Izdat Akad. Navk SSSR, Moscow, (1955). Amer. Math. Soc. Translation Series 2, Vol. 9.

[13] Post, Emil L.: "Recursive Unsolvability of a Problem of Thue", Journal of Symbolic Logic, Vol. 12 (1947), pp. 1-11.

[14] Rabin, Michael O.: "Recursive Unsolvability of Group Theoretic Problems", Annals of Mathematics, Vol. 67 (1958), pp. 172-194.

[15] Rotman, Joseph J.: The Theory of Groups: An Introduction, Allyn and Bacon, Boston, (1965).

[16] Tartakovskii, V.A.: "The Sieve Method in the Theory of Groups" (Russian), Mat. Sbornik, Vol. 25 (1949), pp. 3-50.

[17] Tartakovskii, V.A.: "Applications of the Sieve Method to the Solution of the Identity Problem in Certain Types of Groups" (Russian), Mat. Sbornik, Vol. 25 (1949), pp. 251-274.

[18] Tartakovskii, V.A.: "Solution of the Identity Problem for Groups with a K-reduced Basis for K>6" (Russian), Isvestiya Akad. Naut SSSR, Ser. Mat., Vol. 13 (1949), pp. 483-494.

[16]-[18] English Translation of Above: Russian Translation Project, Amer. Math. Soc., Vol. 60 (1952).

[19] Thue, Axel; "Probleme über Veränderungen von Zeichenreihen nach Gegebenen Regeln", Skrifter utgit ar Videnskapsselskapet i Kristiania, I. Mathematisk-naturvidenskabelig klasse 1914, no. 10, (1914), 34 pp.

[20] Turing, A.M.: "On Computable Numbers, with an Application to the Entscheidungsproblem", Proc. London Math. Soc. (2), Vol. 42(1937), pp. 230-265.

[21] Turing, A.M.: "The Word Problem in Semigroups with Cancellation", Annals of Mathematics, Vol. 52 (1950), pp. 491-505.

FURTHER REFERENCES

The following references were suggested by W.W. Boone, and they, together with the preceding bibliography, form an almost complete bibliography on the word problem.

[22] Adjan, S.I.: "The Algorithmic Unsolvability of Checking Certain Properties of Groups" (Russian), Dokl.Akad. Nauk SSSR, Vol. 103 (1955), pp. 533-535.

[23] Artin, Emil: "Theory of Braids", Annals of Math.(2), Vol. 48 (1947), pp. 101-126.

[24] Baumslag, Gilbert; Boone, W.W.; and Neumann, B.H.: "Some Unsolvable Problems about Elements and Subgroups of Groups", Mathematica Scandinavica, Vol. 7 (1959), pp. 191-201.

[25] Bokut, L.A.: "On a Property of the Groups of Boone" (Russian), Algebra i Logika, Seminar, Vol. 5, No.5 (1966), pp. 5-23; and Vol. 6, No. 1(1967), pp. 15-24. (Reviewed by D.J. Collins, Journal of Symbolic Logic, Vol. 33 (1968), pp. 470-471.)

[26] Boone W.W.: "Certain Simple Unsolvable Problems of Group Theory", Koninkl. Nederl, Akademie van Wetenschappen, Amsterdam, Series A
Part I, Vol.57 (1954), pp. 231-237;
Part II, Vol.57 (1954), pp. 492-497;
Part III, Vol.58 (1955), pp. 252-256;
Part IV, Vol.58 (1955), pp. 571-577;
Part V, Vol.60 (1957), pp. 22-27 (last line p.24 read "and 7" after "6", and "their" for "its");
Part VI, Vol.60 (1957), pp. 227-232.
Note: these papers also appear in Indagationes Mathematicae, Vols. 16 (Parts I and II), 17 (Parts III and IV), and 19 (Parts V and VI) with the same page numbers as above.

[27] Boone, W.W.: "Word Problems and Recursively Enumerable Degrees of Unsolvability", Annals of Math., A First Paper on Thue Systems, Vol. 83 (1966), pp. 520-571; A Sequel on Finitely Presented Groups, Vol. 84 (1966), pp. 49-84.

[28] Boone, W.W.: "Decision Problems about Algebraic and Logical Systems as a Whole and Recursively Enumerable Degrees of Unsolvability", Contributions to Mathematical Logic, Proceedings of the Hanover Colloquium, North Holland Publishing Company, Amsterdam (1968), pp. 177-197.

[29] Boone, W.W.; Haken, W.; and Poénaru, V.; "On Recursively Unsolvable Problems in Topology and their Classification", Ibid..

[30] Boone, W.W. and Rogers, Hartley Jr.: "On a Problem of J.H.C. Whitehead and a Problem of Alonzo Church", Mathematica Scandinavica, Vol, 19 (1966), pp. 185-192.

[31] Ceitin, G.S.: "An Associative Calculus with an Insoluble Problem of Equivalence" (Russian), Trudy Mat. Inst. Steklov, (1952), pp. 172-189.

[32] Church, Alonzo: "A Note on the Entscheidungsproblem", Journal of Symbolic Logic, Vol. 1(1936), pp. 40-41 and 101-102.

[33] Church, Alonzo: "An Unsolvable Problem of Number Theory", American Journal of Mathematics, Vol. 58 (1936), pp. 345-363.

[34] Clapham, C.R.J.: "An Embedding Theorem for Finitely Generated Groups", Proceedings of the London Math. Soc., Vol. 27 (1967), pp. 420-430.

[35] Collins, D. J.: "Recursively Enumerable Degrees and the Conjugacy Problem", Acta Mathematica, Vol. 122 (1969), pp. 115-160.

[36] Evans, Trevor: "The Word Problem for Abstract Algebras", Journal of the London Math. Soc., Vol. 26 (1951), pp. 64-71.

[37] Fridman, A.A.: "Degrees of Unsolvability of the Problem of Identity in Finitely Presented Groups" (Russian), U.S.S.R. Academy of Sciences--Central Mathematics-Economics Institute, Science Publishing House, Moscow, (1967).

[38] Fridman, A.A.: "On the Relation between the Word Problem and the Conjugacy Problem in Finitely Defined Groups" (Russian), Trudy Moskov. Mat. Obsc., Vol. 9 (1960), pp. 329-356.

[39] Friedberg, R.M.: "Two Recursively Enumerable Sets of Incomparable Degrees of Unsolvability (Solution of Post's Problem, 1944), Proceedings National Academy of Sciences, U.S.A., Vol. 43 (1957), pp. 236-238.

[40] Jockusch. Carl G. Jr.: "Supplement to Boone's 'Algebraic Systems'", Contributions to Mathematical Logic, K. Schütte, editor, North-Holland, Amsterdam, 1968.

[41] Lyndon, Roger: "On Dehn's Algorithm", Mathematische Annalen, Vol. 166 (1966), pp. 208-228.

[42] Magnus, Wilhelm: "Das Identitäts Problem für Gruppen mit Einer Definierenden Relation", Mathematische Annalen, Vol. 106 (1932), pp. 295-307.

[43] Markov, A.A.: "Impossibility of Certain Algorithms in the Theory of Associative Systems"(Russian), Dokl. Akad. Nauk SSSR, Vol.55 (1947), pp. 587-590, Vol.58 (1947), pp. 353-356.

[44] Markov, A.A.: "Insolubility of the Problem of Homeomorphy", Proceedings of International Congress of Mathematicians, Cambridge University Press, 1958, pp. 300-306.

[45] Matijasevic, J.V.: "Simple Examples of Undecidable Associative Calculi", Soviet Mathematics, Vol.8 No.2 (1967), pp. 555-557. (Reviewed by D.J. Collins, Journal of Symbolic Logic, Vol. 33 (1968) pp. 469-470.)

[46] Miller, Charles F. III: "On Britton's Theorem A", Proc. Am. Math. Soc., Vol.19(1968), pp. 1151-1154.

[47] Mostowski, A.: "On the Undecidability of some Problems in Special Classes of Groups", Fundamenta Mathematicae, Vol. 54 (1966), pp. 123-135.

[48] Murskii, V.L.: "Embedding of Recursively Presented Semigroups in Finitely Presented Semigroups", Mathematical Notes, Vol. 1 No.2 (1967).

[49] Novikov, P.S.: "Unsolvability of the Conjugacy Problem in the Theory of Groups" (Russian), Izv. Akad. Nauk SSSR, Ser. Mat., Vol. 18 (1954), pp. 485-524.

[50] Post, Emil L.: "Recursively Enumerable Sets and their Decision Problems", Bulletin of the Am. Math. Soc., Vol. 50 (1944) pp. 284-310.

[51] Sanov, I.I.: "A Property of a Representation of a Free Group" (Russian), Doklady Akad. Nauk SSSR (N.S.), Vol. 57 (1947), pp. 657-659.

[52] Schiek,Helmut: "Ahnlichkeitsanalyse von Gruppenrelationen", Acta Mathematica, Vol. 96 (1956), pp. 157-252.

[53] Schupp, Paul E.: "On Dehn's Algorithm and the Conjugacy Problem", Mathematische Annalen, Vol. 78 (1968), pp. 119-130.

[54] Scott, Dana: "A Short Recursive Unsolvable Problem (abstract), Journal of Symbolic Logic, Vol. 21 (1956), pp. 111-112.

[55] Tarski, Alfred: "A Decision Method for Elementary Algebra and Geometry", The Rand Corporation, Santa Monica, California, 1948.

APPENDIX A

In this appendix we shall take a brief and by no means either complete or rigorous look at some special classes of groups for which the word problem has been solved. When we say that we are given a finitely presented group, G, we assume that we are in fact given a particular finite presentation for G, and we ask the question, "for what sorts of finite presentations does there exist an effective procedure for determining whether or not an arbitrary word in the generators is equal to the identity?" We have already seen that the answer is not "for all sorts".

We will first look at the simplest type of group presentation: that with no defining relations, namely the free group presentation. Given a free group on a finite set (or, for that matter, an infinite set) of given generators we know that each word is equal to a unique reduced word and we know that there is a very straightforward effective procedure for finding that unique reduced word for any given word. Since the empty word, which represents the identity, is reduced, the solution for the word problem is clear. Given an arbitrary word W, we simply reduce W until we find the reduced word W' such that W=W'. If W' is empty we know that W=1; if not, W≠1. Thus all free groups have solvable word problem.

In 1912, Dehn showed that the word problem is solvable for the fundamental group of a closed two dimensional manifold of genus $g \geqq 2$. This is the group with presentation

$$\{a_1, a_2, \ldots, a_{2g}; (a_1, a_{g+1})(a_2, a_{g+2}) \ldots (a_g, a_{2g})\}$$

where (a_i, a_j) is the commutator of a_i and a_j, $a_i^{-1} a_j^{-1} a_i a_j$. Dehn's proof was a geometric one and has since been generalized considerably.

Next, suppose that G_1 and G_2 are disjoint finitely presented groups with solvable word problems. We will show that the free product $G_1 * G_2$ has solvable word problem.

We define the free product, $\underset{\gamma \in \Gamma}{*} G$, of a set, $\{G_\gamma : \gamma \in \Gamma\}$, of pair wise disjoint groups (note that any set of groups can be "made" pairwise disjoint by "painting their elements different colors") such that $\{S_\gamma ; R_\gamma\}$ is the presentation of G_γ, for each $\gamma \in \Gamma$, to be the group with presentation $\{ \bigcup_{\gamma \in \Gamma} S_\gamma ; \bigcup_{\gamma \in \Gamma} R_\gamma \}$.

Thus, an element of $G_1 * G_2$ is a word in the symbols of $S_1 \cup S_2$ and their inverses. Let W be a word in $G_1 * G_2$. Then the following effective procedure can be used to determine whether or not W=1 (the empty word) in $G_1 * G_2$.

Step 1. If W is empty, write "W=1", and stop.

If not go to step 2.

Step 2. Apply parentheses to W to write W as a product of subwords of W such that each subword is a word in either G_1 or G_2 and no two adjacent subwords are both in G_1 or both in G_2.

Step 3. Apply the solutions of the word problems for G_1 and G_2 to the subwords obtained in Step 2 and if a subword is equal to 1 in either G_1 or G_2, delete it from W. After making all possible deletions, we get a new word which will be equal, in $G_1 * G_2$, to W and which we will also call W. Go back to Step 1 if at least one deletion was made. If no deletions were made, write "$W \neq 1$", and stop.

We can obviously perform step 1 in a finite number of steps.

Now, since G_1 and G_2 are both disjoint and have a finite number of generators we can surely perform step 2 in a finite number of steps. Since W is only finite in length, we must always eventually come to a point, after a finite number of repeated applications of steps 2 and 3, where either W is empty, or no more deletions can be made. So the described procedure is certainly effective, and so the word problem for $G_1 * G_2$ is solvable.

By induction, then, the word product for any finite free product of finitely presented groups is solvable, if the word problem is solvable for each of the groups whose free product is being taken.

It can be shown that if G is the free product of G_1 and G_2 with isomorphic subgroups $H_1 \subset G_1$ and $H_2 \subset G_2$ amalgamated, we can again solve the word problem in G if the word problems for G_1 and G_2 are both solvable and if in G_1 and G_2

we can decide whether a given element belongs to H_1 or H_2 respectively, and if we can solve the word problems in H_1 and H_2, and also if the isomorphism between H_1 and H_2 is constructive, i.e. if given an arbitrary element of H_1 we can effectively determine the element of H_2 onto which the element of H_1 is mapped, and vice versa.

Using some of these results, it has been shown that the word problem is solvable for any group G which is finitely generated and which has one defining relation. This is a generalization of Dehn's result, for Dehn's group has exactly one defining relation.

Suppose, now, that G is a finitely presented group such that no two of the defining relators have any symbols in common; i.e. if R_1 and R_2 are any two defining relators such that R_1 is a reduced word involving the generators $a_{i_1},\ldots,a_{i_n}$ of G and R_2 is a reduced word involving the generators $a_{j_1},\ldots,a_{j_m}$ of G, then the sets $\{a_{i_1},\ldots,a_{i_n}\}$ and $\{a_{j_1},\ldots,a_{j_n}\}$ are disjoint. Then G is the finite free product of disjoint groups each with one defining relation and, perhaps, a free group, and as such has a solvable word problem. In this case G is such that any two of its defining relators have no subwords in common.

In 1949, Tartakovskii [18] managed to extend this result somewhat by solving the word problem for groups in which any two defining relators have only very small, compared to their lengths, subwords in common. However, Tartakovskii's solution depends on knowing beforehand the orders of all the generators of the group, and this may not be readily available knowledge from the presentation, and, in fact, may not be available at all. Tartakovskii's results are combinatorial in nature and are so elaborate that it would be impossible to describe them without lengthy preparation.

Finally, the word problem has been solved for groups, G, where for each pair, a,b, of generators, ab=ba is among the defining relations or can be effectively derived from the defining relations; i.e. if a group can be shown to be abelian, it has solvable word problem. A proof of this result can be found in Section 3.3 of Magnus, Karrass, and Solitar , [11].

Much more work than that described above has been done on the word problem. A slightly more detailed outline of this work is given in Section 6.1 of Magnus, Karrass, and Solitar, and detailed proofs of some of the results can be found throughout the book. Other proofs can be found in some of the original papers listed in the bibliography of this thesis and in the bibliography of Magnus, Karrass, and Solitar.

APPENDIX B
More on Computable and Recursive Functions and the Universal Turing Machine.

In Chapter III we showed that all recursive and partial recursive functions were computable or partially computable, respectively. It was not necessary in our proof of the existence of a Turing machine, Z_0, with unsolvable halting problem, to show the converse, namely that every partially computable or computable function is partial recursive or recursive, respectively, and so this was not shown. However we came very close to showing this, and so, if only for aesthetic reasons, we include the proof in this appendix. We also show the existence of another Turing machine with unsolvable halting problem.

In the proof of Theorem 5.18, we presented a list of recursive predicates and functions. We now extend that list with the following functions:

(1*) CU(n,x)

$CU(n,x) = 0$ if $n\text{Gl}\ x \neq 11$

$CU(n,x) = 1$ if $n\text{Gl}\ x = 11$

CU(n,x) is the characteristic function of the recursive predicate $n\text{GL}\ x \neq 11$ and, as such, is recursive.

(2*) Corn (x)

If $x = gn(M)$, then $\text{Corn}(x) = \|M\|$ (the number of 1's or s_1's in M). If x is not the Gödel number of an expression, then we don't care what Corn(x) is.

$$\mathrm{Corn}(x) = \sum_{n=1}^{\pounds(x)} \mathrm{CU}(n,x)$$

(3*) $U(y)$

If y is the Gödel number of the sequence of expressions $M_1, M_2, \ldots, M_n$, then $U(y) = \|M_n\|$. Otherwise, we don't care what $U(y)$ is.

$$U(y) = \mathrm{Corn}\ (\pounds(y)\mathrm{Gl}n).$$

THEOREM Ap. 1. Every (Partially) computable function is (Partial) recursive.

PROOF: Suppose that $f(X^{(n)})$ is partially computable. Then there is a Turing machine Z_0 such that $\Psi_{Z_0}^{(n)}(X^{(n)}) = f(X^{(n)})$. Now let z_0 be a Gödel number of Z_0. Now the predicate $T_n(z_0, X^{(n)}, y)$ is recursive and so the function $\min_y T_n(z_0, X^{(n)}, y) = \min_y[C_{T_n}(z_0, X^{(n)}, y) = 0]$ is partial recursive. Now, $\min_y T_n(z_0, X^{(n)}, y)$ is the Gödel number of the computation of Z_0 beginning with $q_1(\overline{x_1, x_2, \ldots, x_n})$, and is defined exactly when $\Psi_{Z_0}^{n}(x_1, \ldots, x_n)$ is. Thus $U(\min_y T_n(z_0, X^{(n)}, y))$ is partial recursive and $U(\min_y T_n(z_0, X^{(n)}, y)) = \Psi_{Z_0}^{n}(X^{(n)}) = f(X^{(n)})$ and so f is partial recursive.

Now suppose that f is computable. Then f is total and so for all n-tuples, $(x_1, x_2, \ldots, x_n)$ there is a computation of Z_0 beginning with $q_1(x_1, x_2, \ldots, x_n)$. Thus for every n-tuple $(X^{(n)})$, there is a y_0 such that $T_n(z_0, X^{(n)}, y_0)$

holds. Thus $T_n(z_0,X^{(n)},y)$ is a regular recursive predicate and so $\min_y T_n(z_0,X^{(n)},y)$ and hence $U(\min_y T_n(z_0,X^{(n)},y))$ are recursive. Thus $f(X^{(n)})$ is recursive. □

Thus, the terms "computable" and "recursive" are interchangable, as are "partially computable" and "partial recursive", and so, by Church's thesis, a function is calculable if and only if it is computable (or partially computable, if it is not total).

Now, we obtain as a corollary to Theorem 1, the promised converse to Corollary 3.1.

COROLLARY Ap. 2. Every function which is total and partial recursive, is recursive.

PROOF: Suppose f is total and partial recursive. Since it is partial recursive, it is partially computable. Since it is partially computable and total, it is, by definition, computable. But then by Theorem 1, it is recursive. □

Now consider the partial recursive binary function $\varphi(z,x) = U(\min_y T(z,x,y))$. This function is partially computable, and so there is a Turing machine U such that $\Psi_U^{(2)}(z,x) = \varphi(z,x)$. We call U the "universal Turing machine" since it can be employed to compute any partially computable singulary function as follows.

Suppose f(x) is partially computable. Let Z_0 be the Turing machine which computes f, i.e. such that $\Psi_{Z_0}(x)=f(x)$.

Then let z_0 be a Godel number of Z_0. Then we have

$$\Psi_U(z_0,x) = \varphi(z_0,x) = U(\min_y T(z_0,x,y)) = \Psi_{Z_0}(x) = f(x).$$

The machine, U, can also be employed to compute n-ary functions. Suppose $f(x_1,x_2,\ldots,x_n)$ is an n partially computable function. With the n-tuple $(x_1,x_2,\ldots,x_n)$ we associate the single number

$$x = \prod_{k=1} Pr(k)^{x_k} .$$

In other words we define the computable n-ary function $g(x_1,x_2,\ldots,x_n)$ by

$$g(x_1,x_2,\ldots,x_n) = \prod_{k=1}^{n} Pr(k)^{x_k} .$$

Now define $h_n(x)$ by $h_n(x) = f(1Glx,2Glx,\ldots,nGlx)$. Then for any n, $h_n(x)$ is partially computable and we have

$$f(X^{(n)}) = h_n(g(X^{(n)})).$$

For each n, there is a Turing machine Z_n, with Gödel number z_n, such that $\Psi_{Z_n}(x) = h_n(x)$. And thus we have

$$f(X^{(n)})=h_n(g(X^{(n)}))=\Psi_{Z_n}(g(X^{(n)}))=\Psi_U^{(2)}(z_n,g(X^{(n)})).$$

And since g is recursive and hence calculable (in fact, g can be calculated easily) the machine U can be used to calculate $f(X^{(n)})$.

THEOREM Ap.3. The universal Turing machine, U, has unsolvable halting problem.

PROOF: Near the end of the proof of Theorem 4.1 we found that "... there can be no effective procedure for determining for an arbitrary Turing machine Z and an arbitrary number x whether or not there is a computation of Z beginning with $q_1\overline{x}$." Since the system of Gödel numbering is certainly effective, i.e. given a Turing machine we can effectively determine its Gödel number and vice versa, and since $\Psi_U^{(2)}(z,x) = \Psi_Z(x)$, where z is the Gödel number of the machine Z, if the halting problem for U had a solution, then we would simply find the Gödel number for the machine Z and use the solution to the halting problem for U to determine whether or not there was a computation of U beginning with $q_1(\overline{z,x})$, which would tell us whether or not there was a computation of Z beginning with $q_1\overline{x}$. This, however, would contradict our previous result. □

Index of some Notation and Definitions

	Page
$\vee,\wedge,\sim$	67
$\overset{z}{\underset{y=1}{\exists}}$	68
$\overset{z}{\underset{y=1}{\forall}}$	68
$\underset{y}{\exists}$	68
$\underset{y}{\forall}$	69
$gn(M)$	72
$exp(r)$	72
$Pr(k)$	72,88
$N(x)$	75
$\alpha(x)$	75
$[\sqrt{x}]$	76
$[x/y]$	76
$R(x,y)$	76
$\overset{z}{\underset{y=0}{M}}\ P(y,X^{(n)})$	86
$y\|x$	88