AN ABSTRACT OF THE THESIS OF

THOMAS CLAYTON NELSON for the DOCTOR OF PHILOSOPHY
(Name) (Degree)

in MATHEMATICS presented on June 10, 1969
(Major) (Date)

Title: UNIVERSAL COMPUTING SYSTEMS. PART 1, SIMULATION OF ANY NORMAL ALGORITHM BY TURING MACHINE. PART 2, A UNIVERSAL TURING MACHINE TO TRANSLATE ANY TURING MACHINE TO A NORMAL ALGORITHM

Abstract approved: Redacted for Privacy
Harry E. Goheen

In this paper a direct, constructive proof of the equivalence of the Normal Algorithm and Turing machine using the Turing machines NAS (Normal Algorithm Simulator) and NAC (Normal Algorithm Converter) is presented. The Turing machine NAS can simulate any particular Normal Algorithm, and NAC can convert the quintuples of a given Turing machine to an equivalent Normal Algorithm.

Universal Computing Systems. Part 1, Simulation of Any Normal Algorithm by Turing Machine. Part 2, A Universal Turing Machine to Translate Any Turing Machine to a Normal Algorithm

by

Thomas Clayton Nelson

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

June 1970

APPROVED:

Redacted for Privacy

Professor of Mathematics

in charge of major

Redacted for Privacy

Acting Chairman of Department of Mathematics

Redacted for Privacy

Dean of Graduate School

Date thesis is presented June 10, 1969

Typed by Clover Redfern for Redacted for Privacy

ACKNOWLEDGEMENT

I would like to thank Professor Harry E. Goheen for his patience, understanding, and expert guidance. I would also like to thank my eminent colleagues James Meeker and Harvey Thoennes for their helpful advice in using TASP. I am also indebted to my parents, who made it all possible.

TABLE OF CONTENTS

LIST OF FIGURES

UNIVERSAL COMPUTING SYSTEMS. PART 1, SIMULATION OF ANY NORMAL ALGORITHM BY TURING MACHINE. PART 2, A UNIVERSAL TURING MACHINE TO TRANSLATE ANY TURING MACHINE TO A NORMAL ALGORITHM.

I. THE NORMAL ALGORITHM

An algorithm can be intuitively defined as a list of instructions to carry out some task such that a person who does not understand the task can complete it. Some examples of algorithms are the Euclidean Algorithm to find the greatest common divisor of two integers and the rule for finding the derivative of a polynomial.

While the intuitive definition is a good starting point in the study of algorithms, it is desirable to give a formal definition to the concept of algorithm.

One approach identifies the concept of algorithm with recursive functions. A generalized form of Church's Thesis states that there is an algorithm for computing a function f if and only if f is partial recursive (Nelson, 1968). The domain of f is a subset of N^m, and the range of f is a subset of N, where N is the set of Natural Numbers, m is a Natural Number, and N^m denotes the Cartesian product $N \times N \times \ldots \times N$ (m times).

An intuitively more satisfying, although no less rigorous, approach identifies the concept of algorithm with the Turing machine. While the alphabet of a Turing machine does not necessarily consist

of natural numbers, it is always possible to arithmetize the Turing machine (Hermes, 1965). Consequently it is sufficient to discuss Turing machines which evaluate functions whose domain is a subset of N^m and whose range is a subset of N. Such a function f is said to be Turing computable if there exists a Turing machine which, when presented with an element $\hat{x}$ of the domain of f on its tape, will stop with the corresponding value $f(\hat{x})$ on its tape. An equivalent form of Church's Thesis is as follows. There is an algorithm to compute a function f if and only if f is Turing computable. These two forms of Church's Thesis are equivalent since the set of partial recursive functions and the set of Turing computable functions coincide (Hermes, 1965).

A class of problems is said to be unsolvable if no algorithm exists to solve every problem in the class. It is possible to show in a very clear and natural way using Turing machines that certain problems are unsolvable. Some examples of unsolvable problems are the stopping problem, the printing problem, and the word problem for a Semi-Thue system.

In proving the unsolvability of these problems use is made of a Universal Turing machine, which is a Turing machine which can simulate any particular Turing machine. Hence the Universal Turing machine can carry out any algorithm.

One may also define the concept of algorithm by identifying an

algorithm with the Normal Algorithm of A. A. Markov. Before giving the equivalent definition of algorithm we define the Normal Algorithm.

In the following we use mostly the notation in Markov (1961). Given a nonempty finite set A which is called the alphabet. The elements of A are called letters. A word over the alphabet A is the concatenation of a finite number of letters of A. We consider the empty word to be a word and denote the empty word by Λ. Note that the concatenation of two words over A is also a word over A. If a word $W = XY$, where X and Y are words over A, then X is said to be a head of W, and Y is said to be a tail of W. It is possible for a word to be its own head or tail since $W = W\Lambda = \Lambda W$. Whenever $W = XY$ and $X \neq W$, X is said to be a proper head of W. Similarly, if $Y \neq W$, then Y is a proper tail of W.

If a word $W = XYZ$, where X, Y, and Z are words over A, then the word Y is said to occur in W. Note that the empty word enters in every word since $W = W\Lambda\Lambda$. It is also possible for a word Y to occur in a given word W more than once. However, the nonempty word Y can occur in W only a finite number of times since the length of W is finite. If a word $W = XYZ$, where the word $Y \neq \Lambda$ occurs exactly once in XY, then the first occurrence of Y in W is the Y following X. The first occurrence of the empty word in a word is the empty word which is a head of the

word W.

In everything which follows we assume that the alphabet A does not contain any of the symbols "→", ".", or "#". A substitution is a transformation of the first occurrence of a word in a given word into another word. A substitution is written $X \rightarrow \#Y$, where X and Y are words over A, and the symbol "#" may either equal the symbol "." or the empty word Λ. The presence of the symbol "." indicates that the process terminates after replacing X by Y, and the substitution is called a terminating substitution. If the given word $W^0 = Z_1XZ_2$, then, provided the first occurrence of the word X in W^0 follows the word Z_1, application of the substitution converts the word W^0 to $W^1 = Z_1YZ_2$. If the word X occurs in a given word W^0 then the substitution is amenable to W^0; otherwise the substitution is unamenable to W^0.

A nonempty list of substitutions which are to be performed in the order that they are given on the list is called a scheme. If at least one substitution in a scheme is amenable to a word W, we say that W is amenable to the scheme. Otherwise W is unamenable to the scheme.

A Normal Algorithm NA consists of a finite nonempty alphabet A, a finite scheme,

$$W_1 \rightarrow \#W_1'$$
$$W_2 \rightarrow \#W_2'$$
$$\vdots$$
$$W_n \rightarrow \#W_n'$$

and a given initial word W^0. Beginning with $i = 1$ and $j = 0$ it is determined whether W_i occurs in W^j. If so, the first occurrence of W_i in W^j is replaced by W_i', thereby obtaining the word W^{j+1}. If $\# = .$ the process terminates. If $\# = \Lambda$ the index i is set equal to 1, and the process continues. If the word W_i does not occur in W^j, and provided $i < n$, the index i is replaced by $i + 1$, and the process continues. If W_i does not occur in W^j and $i = n$ it indicates W^j is unamenable to the scheme of NA. In this case the algorithm terminates, and is said to terminate naturally.

If the Normal Algorithm NA transforms the word X into the word Y in one step, i.e., using exactly one substitution of the scheme of NA, we write $NA : X \vdash Y$. If the substitution used is a terminating substitution we write $NA : X \vdash \cdot Y$. If $NA : X \vdash Y$ and Y is unamenable to the scheme of NA, we write $NA : X \vdash Y\neg$. Whenever there exists a sequence of words $W^0, W^1, \ldots, W^m$ over A such that $NA : W^j \vdash W^{j+1} (j = 0,1,\ldots,m-1)$ we write either $NA : W^0 \vdash\!\!\vdash W^m$ or $NA : W^0 \vdash W^1 \vdash \ldots \vdash W^m$.

If $W^0 \models W^{m-1}$ and NA : $W^{m-1} \vdash W^m \neg$ we write either NA : $W^0 \models W^m \neg$ or NA : $W^0 \vdash W^1 \vdash \ldots \vdash W^m \neg$. Whenever NA : $W^0 \models W^{m-1}$ and NA : $W^{m-1} \vdash \cdot W^m$ we write either NA : $W^0 \models \cdot W^m$ or NA : $W^0 \vdash W^1 \vdash \ldots \vdash \cdot W^m$.

The following example will illustrate the Normal Algorithm. Let $A = \{a, b, c\}$, and the scheme be

$$b \rightarrow acc$$

$$ca \rightarrow accc$$

$$aa \rightarrow \Lambda$$

$$cccc \rightarrow \Lambda$$

Let the initial word $W^0 = aacb$. Then NA : $aacb \models ac \neg$ since NA : $aacb \vdash aacacc \vdash aaaccccc \vdash accccc \vdash ac \neg$.

It is interesting to note that there exists a Universal Normal Algorithm, and corresponding to the unsolvable problems using the Turing machine point of view there are analogous unsolvable problems in term of the Normal Algorithm (Markov, 1961).

While the Turing machine lends itself to an intuitively more lucid explanation of unsolvable problems, the Normal Algorithm would be a very natural approach to the problems of language translation and formalizing mathematical proofs. In fact, a dictonary can be regarded as the scheme of a Normal Algorithm.

We now return to the definition of algorithm using the Normal Algorithm. It is sufficient to consider functions whose domain is a subset of N^m and whose range is a subset of N, since any Normal Algorithm can be arithmetized (Detlovs, 1958). Such a function f is said to be Markov computable provided that there exists a Normal Algorithm which transforms the initial word $\hat{x}$ into the word $f(\hat{x})$, where $\hat{x}$ belongs to the domain of f. Another equivalent form of Church's Thesis is as follows. There exists an algorithm to compute a function f if and only if f is Markov computable. The above definition of algorithm is equivalent to the previous definitions since it has been shown by Detlovs (1958) that the partial recursive functions coincide with the Markov computable functions.

We next turn to the problem of defining equivalence of algorithms. Two algorithms are equivalent if, whenever they are presented with the same input word, either both algorithms will stop after a finite number of steps and yield the same output word, or both algorithms will not stop. A class of algorithms $\underline{A}_1$ is equivalent to a class of algorithms $\underline{A}_2$ if for every algorithm in $\underline{A}_1$ there exists an equivalent algorithm in $\underline{A}_2$, and for every algorithm in $\underline{A}_2$ there exists an equivalent algorithm in $\underline{A}_1$.

It can be shown indirectly that the class of Normal Algorithms is equivalent to the class of Turing machines using the result of Detlovs (1958) and the coincidence of the Turing computable functions

with the partial recursive functions. In this paper we present a direct and constructive proof of the equivalence of the class of Normal Algorithms with the class of Turing machines. Thus the present proof illuminates the concept of Normal Algorithm as well as the concept of Turing machine.

The Turing machine NAS (Normal Algorithm Simulator), described in Chapter II, will simulate any Normal Algorithm, and the Turing machine NAC (Normal Algorithm Converter), described in Chapter III, will convert the quintuples of any given Turing machine into the scheme of an equivalent Normal Algorithm. Note that it is possible for NAS to operate on the output of NAC. Any Normal Algorithm can be simulated by NAS, and any Turing machine is equivalent to the Normal Algorithm simulated by NAS operating on NAC operating on the quintuples and input tape of that Turing machine. Therefore the class of Normal Algorithms is equivalent to the class of Turing machines.

It should be noted that the Turing machines NAS and NAC are not unique. Using either a different encoding or a different strategy, or both, from those used in this paper would lead to different Turing machines which would accomplish the tasks of NAS and NAC.

Note that NAS simulates the operation of any Normal Algorithm. If NAC operates on the quintuples and input tape of NAS, the output of NAC is a Normal Algorithm which is equivalent to the original Normal

Algorithm, and hence a Universal Normal Algorithm. Therefore the universality of some Normal Algorithm is established.

Since NAS and NAC operate on any Normal Algorithm and any Turing machine, respectively, including the Universal Normal Algorithm and the Universal Turing machine, they are universal computing systems.

II. DESCRIPTION OF NAS

Introduction

The Turing machines NAS (Normal Algorithm Simulator) and NAC (Normal Algorithm Converter) were simulated on the Control Data Corporation 3300 computer using the TASP (Turing Automaton Simulation Program) of McCune [1968]. This program was in turn based on the TASP for the Scientific Data Systems 920 computer developed by Coffin, Goheen, and Stahl (1963).

Any Turing machine simulated by TASP must begin at the left end of the information part of the tape in state 1. The information part of the tape is the smallest section of tape such that the remainder of the tape contains only the space of the Turing machine. If the Turing machine being simulated stops, TASP prints out the parts of the tape bounded on the left by the symbol ", " and on the right by the symbol ". ", except when a linked run is desired. A linked run occurs when more than one Turing machine operates on the tape in sequence. The number of the first Turing machine is read in by cards. Whenever a Turing machine stops with the word :xyz0 on the tape, and the symbol ":" occurs exactly once on the tape, the Turing machine whose number is xyz operates on the tape next. When a Turing machine stops and the symbol ":" does not appear on the tape TASP prints out a message to this effect, and goes on to

the next part of the program. When the linked run is used the information part of the tape is printed out whenever a particular Turing machine stops.

The Operation of NAS

In simulating the operation of any Normal Algorithm the following encoding will be employed. A letter of the alphabet A is encoded as the symbol "1" followed by a nonzero finite number of occurrences of the symbol "0". This word is called an encoded letter. In the following the elements of the alphabet A will be called letters, and the letters which are used in the encoded letter will be called symbols. Thus if $a \in A$ and a is encoded as 1000, 1 and 0 are called symbols.

Any finite nonempty word W over the alphabet A is encoded by concatenating the encoded letters in the order given by the word W. The empty word is encoded directly as the empty word. The word obtained by encoding the word W is called the encoded word W.

A nonterminating substitution $W_i \rightarrow W_i'$ is encoded as the encoded word W_i followed by an occurrence of the symbol "2", followed by the encoded word W_i' followed by another occurrence of the symbol "2". A terminating substitution $W_i \rightarrow \cdot W_i'$ is encoded in the same way as the nonterminating substitution, except

that the encoded word W_i' is followed by an occurrence of the symbol "4". The result of encoding the substitution $W_i \rightarrow \#W_i'$ is called an encoded substitution.

The format of the tape of the Turing Machine NAS is as follows. The symbol "," is followed by the encoded substitutions in the order given by the Normal Algorithm to be simulated. The symbol "7" replaces the first occurrence of the symbol "1" in the encoded word W_1, provided $W_1 \neq \Lambda$. If the last encoded substitution is a nonterminating substitution the symbol "3" follows the encoded word W_n' instead of the symbol "2". If the last encoded substitution is a terminating substitution the symbol "3" appears to the right of the symbol "4" on the tape. The encoded word W^0, where W^0 is the initial word of the Normal Algorithm, appears to the right of the symbol "3". The symbol "5" replaces the first occurrence of the symbol "1" in the encoded word W^0, provided $W^0 \neq \Lambda$. The symbol "." appears to the right of the encoded word W^0. The symbols "," and "." were selected because the TASP program prints out the part of the tape bounded on the left by the symbol "," and on the right by the symbol "." . The symbols "5" and "7" are used as markers and will be explained in the sequel.

The following example illustrates the input tape of NAS. Consider the Normal Algorithm given in Chapter I. The encoded letter a

is 10, the encoded letter b is 100, and the encoded letter c is 1000. The initial word W^0 = aacb. The encoded word aacb is 10101000100. The input tape to NAS is ,7002101000100021000102101000100010002101022100010001000100023501010001 00. This Normal Algorithm was simulated by NAS. The output, which is the part of the tape to the right of the symbol "3" and to the left of the symbol ".", was 101000, which is the encoded word ac.

In describing the operation of NAS the following convention will be employed. Whenever a particular state q of NAS is used to move right (left) until one of the n symbols $\ell_1, \ell_2, \ldots, \ell_n$ is found, we write "the function of state q is $SR\ell_1\ell_2\ldots\ell_n$ ($SL\ell_1\ell_2\ldots\ell_n$)".

A mode is a set of states in a Turing machine to carry out some task. This concept is in no way essential to the operation of the Turing machine, but is introduced solely to clarify the explanation of the operation of the Turing machine.

The modes of operation of NAS are: Comparison, Replacement, Contraction, Expansion, Advancement, Cleanup, and Termination.

The Comparison mode consists of states 1-16 inclusively and states 62-66 inclusively. The objective of this mode is to decide whether the encoded word W_i of a given substitution occurs in the encoded word W^m, where W^m is the result of applying the Normal Algorithm to be simulated to the initial word W^0 m times. The symbol "7" marks the beginning of the encoded letter being

compared in the encoded word W_i, and the symbol "5" marks the beginning of the encoded word in W^m being compared to W_i.

The Turing machine NAS begins in state 1 over the cell at the left end of the information part of the tape, which contains either the symbol "0" or the symbol "," , depending on whether or not NAS is operating on a tape following NAC. In either case the symbol found in state 1 is converted to 8, and the next state is 66 with a move right. If either 2 or 7 is found in state 66, indicating that NAS is not operating after NAC, NAS moves left. The 8 found is converted to the symbol "," , and the next state is 62 with a move right. If 0 is found in state 66, indicating that NAS is operating following NAC, the next state is 62 with a move right. The function of state 62 is SR27Q. If 2 is found in state 62 it indicates $W_1 = \Lambda$. In this case the next state is 58, a state in the Expansion mode. The encoded word W_1' is subsequently inserted to the left of the encoded word W^m. The details of this case will become clear in the sequel. If Q is found in state 62, which is the case when NAS operates on a tape following NAC, the next state is 63, and Q is replaced by the symbol "," . Following operation by NAC the encoded substitutions and encoded initial word W^0 are bounded on the left by Q, and on the right by R. States 62, 63, 64, and 65 are employed to convert Q and R to the symbols "," and "." respectively and to place the marker "7" immediately to the right of the symbol "," . The

function of state 63 is SRR. The R found is converted to the symbol "." , and the next state is 64, whose function is SL, . Upon finding the symbol "," the next state is 65 with a move right. If 1 is found in state 65 the next state is 2, and 1 is converted to 7. If 2 is found in state 65, indicating $W_1 = \Lambda$, the next state is 58, and the encoded word W_1' is inserted to the left of the encoded initial word W^0. The function of state 2 is SR012. Upon finding 0 in state 2 the next state is 3, and 0 is converted to 6. The function of state 3 is SR. 5. Upon finding 5 the next state is 4. The function of state 4 is SR. 01. If 0 is found in state 4, the next state is 5, and 0 is converted to 6. The function of state 5 is SL7. Upon finding 7 the next state is 2, and the process continues. Whenever the symbol "." is found in state 3, indicating that $W^m = \Lambda$, the next state is 53, a state in the Advancement mode.

The encoded letters are compared until it is ascertained whether they are equal. Two encoded letters are equal if and only if they have the same number of occurrences of the symbol "0". If 1 is found in state 4 it indicates that the encoded letters are unequal. The next state is 6. The function of state 6 is SL03. In state 6, the symbols "5" and "6" are replaced by the symbols "1" and "0" respectively. Upon finding 0 or 3 NAS moves to the right where 1 is found. Upon finding 1 the next state is 7, whose function is SR. 1. If 1 is found in state 7 the next state is 8 and 1 is converted to 5. Hence

the marker has been moved to the next encoded letter. The function of state 8 is SL6. The 6 found is a symbol in the word W_i being compared. The 6 is converted to 0, and the next state is 9 whose function is SL, 24. In state 9 the symbols "6" and "7" are converted to the symbols "0" and "1" respectively. Finding any of the symbols ",", "2", or "4" indicates that the beginning of the encoded word W_i is one cell to the right. Upon finding any of these symbols NAS moves to the right. The 1 found is converted to 7, and the next state is 2.

Whenever either of the symbols "1" or "2" is found in state 2 it marks the end of an encoded letter in the encoded word W_i. It is then determined whether the encoded letter in the encoded word W^m has more occurrences of the symbol "0" than the corresponding encoded letter in the encoded word W_i. Upon finding 1 in state 2 the next state is 10. The function of state 10 is SR5. Upon finding 5 the next state is 11, whose function is SR.01. If 0 is found in state 11 it indicates that the encoded letters are unequal. The next state is 6, and the previously described operation is carried out. If 1 is found in state 11 it indicates that the encoded letters are equal, and the next encoded letters are to be compared. The 1 found in state 11 is converted to 7, and the next state is 12. This 7 is used to mark the right end of the encoded word W_i; if the last encoded letter of the encoded word W_i has not yet been compared this 7 is converted

to 5 in state 4. The function of state 12 is SL7. Upon finding 7 the next state is 13, whose function is SR12. If 1 is found in state 13, 1 is converted to 7, and the next state is 2. If 2 is found in state 13 it indicates that the encoded words being compared are equal. Hence the encoded word W_i occurs in the encoded word W^m. It is then necessary to replace this occurrence of the encoded word W_i in the encoded word W^m by the encoded word W_i'. Upon finding 2 in state 13 the next state is 14 with a move right. If 1 is found in state 14, 1 is replaced by 7 and the next state is 17, a state in the Replacement mode. This 7 is used to mark the encoded word W_i'. If any of the symbols "2", "3", or "4" is found in state 14 it indicates that the word $W_i' = \Lambda$. In this case the next state is 23, a state in the Contraction mode. The first occurrence of the encoded word W_i in the encoded word W^m is subsequently replaced by the empty word Λ.

If the symbol "." is found in state 11 it indicates either the encoded word W_i occurs in the encoded word W^m, or the encoded word W_i has a greater number of symbols than the tail of the encoded word W^m to which it is being compared. If the symbol "." is found in state 11 the next state is 15, whose function is SL7. Upon finding 7 the next state is 16, whose function is SR12. If 2 is found in state 16 the next state is 14. If 1 is found in state 16 the next state is 56, a state in the Advancement mode.

Whenever the symbol "." is found in state 4 it indicates that the encoded word W_i has a greater number of symbols than the tail of the encoded word W^m, and hence the encoded word W_i does not occur in W^m. In this case it is decided whether the encoded word W_{i+1} occurs in the encoded word W^m, provided $i < n$, where n is the number of substitutions in the scheme of the Normal Algorithm being simulated. Advancing from the encoded word W_i to the encoded word W_{i+1} is carried out in the Advancement mode. Whenever the symbol "." is found in state 4 the next state is 52, a state in the Advancement mode.

The Replacement mode consists of states 17-22 inclusively. The objective of this mode is to replace the first occurrence of the word W_i in W^m by W_i'. Recall that the first symbol in the encoded word W_i' is 7, and the first occurrence of the encoded word W_i in the encoded word W^m contains only the symbols "5", "6", and "7", where each occurrence of 1 and 0 in the encoded word W_i has been converted to 5 and 6 respectively, and the 7 occurs to the right of the first occurrence of the encoded word W_i in the encoded word W^m. NAS begins in state 17 over the cell, containing 0, to the right of the 7 which marks the beginning of the encoded word W_i' in the ith encoded substitution of the Normal Algorithm being simulated by NAS. The function of state 17 is SR01234. Whenever 0 is found the next state is 18, and 0 is converted

to 6. The function of state 18 is SR5. Upon finding 5 the next state is 19, whose function is SR. 567. Whenever either 5 or 6 is found it is converted to 0, and the next state is 20. The function of state 20 is SL7. Upon finding 7 the next state is 17, and the process continues.

If 1 is found in state 17 the next state is 21, and 1 is converted to 7. The function of state 21 is SR5. Upon finding 5 the next state is 22, whose function is SR. 567. The 5 or 6 found in state 22 is converted to 5, and the next state is 20.

Whenever either 2, 3, or 4 is found in state 17 the next state is 23, a state in the Contraction mode. This indicates that the word W_i' has been inserted in the correct place, and that there may be some marker symbols between the end of the encoded word W_i' and the tail Y of the encoded word W^m. That is, before replacement $W^m = XW_iY$, and after replacement $W^m = XW_i'MY$, where M is a word consisting of the marker symbols. It is necessar to transform this word into $XW_i'Y = W^{m+1}$. This is the objective of the Contraction mode.

The Contraction mode consists of states 23-31 inclusively. Recall that the word M is bounded on the right by the symbol "7". NAS is in state 23 over the cell to the right of the encoded word W_i'. The function of state 23 is SR. 5. Upon finding 5 the next state is 24, whose function is SR. 567. If either of the symbols "7" or "." is found in state 24 it indicates that no contraction is necessary because

the encoded words W_i and W_i' had the same number of symbols. In this case the next state is 32, a state in the Cleanup mode. Whenever either 5 or 6 is found in state 24, the symbol found is converted to 6, and the next state is 25. The function of state 25 is SR. 017. In state 25, 5 is converted to 6. Whenever 0 is found in state 25 the next state is 26, and 0 is converted to 6. The function of state 26 is SL01. Upon finding either 0 or 1 the next state is 27 with a move right. The 6 found in state 27 is converted to 0, and the next state is 25. When either 1 or 7 is found in state 25 the next state is 28, and the 1 or 7 found is converted to 6. The function of state 28 is SL05. If 0 is found in state 28 the next state is 29 with a move right. In state 29 the 6 found is converted to 1, and the next state is 25. If 5 is found in state 28, which occurs when $W_i' = \Lambda$, the next state is 25, and 5 is converted to 1. When the symbol "." is found in state 25, indicating that the tail Y has been moved to the left so that it is adjacent to the encoded word W_i', the next state is 30, and . is replaced by 6. The function of state 30 is SL05. Whenever 0 is found in state 30 the next state is 31 with a move right. The 6 found in state 31 is converted to . , and the next state is 32, a state in the Cleanup mode. If 5 is found in state 30, which occurs when $W^m = W_i$ and $W_i' = \Lambda$, the next state is 32 and 5 is converted to . .

In the event that the encoded word W_i' has a greater number

of symbols than the encoded word W_i, the tail Y of $W^m = XW_iY$ is moved to the right five cells in the Expansion mode. Symbols of the encoded word W_i' are inserted in these cells in the Replacement mode, and the Expansion mode is used as many times as necessary to insert the encoded word W_i' after the word X. Then, if necessary, the Contraction mode is used to obtain the encoded word $W^{m+1} = XW_i'Y$.

Whenever 7 is found in either of the states 19 or 22 of the Replacement mode it indicates that the encoded word W_i' has a greater number of symbols than the encoded word W_i. Recall that 7 was placed after the end of the word W_i occurring in W^m. If 7 is found in state 19, the next state is 36, and 7 is converted to 0. If 7 is found in state 22, the next state is 36, and 7 is converted to 1.

The Expansion mode consists of states 36-40 and 42-50 inclusive. In this mode the right end of the tail Y, marked by the symbol ".", is found and moved to the right five cells. Starting from the right, each symbol of the tail Y is moved to the right 5 cells.

Due to the encoding requirement that each encoded letter must have at least one occurrence of the symbol "0", NAS is over a cell containing 0 in state 36 at the beginning of the Expansion mode. This 0 is converted to 7 and the next state is 37. This 7 marks the end of the part of W_i' copied in the Replacement mode. The function of state 37 is SR6. The symbol "." found in state 37 is converted to

the symbol "6". Upon finding 6 in state 37 the next state is 38. States 38, 39, 40, and 42 are used to move the symbol "." five cells to the right. In state 42 the symbol "." is converted to the symbol "6" and the next state is 43.

The tail Y is now moved five spaces to the right. The function of state 43 is SL017. Upon finding 0 in state 43 the next state is 44, and 0 is converted to 6. The function of state 44 is SR. 01. Upon finding any of the symbols "0", "1", or "." in state 44, the next state is 45 with a move left. In state 45 the 6 found is converted to 0, and the next state is 43. Whenever 1 is found in state 43 the next state is 46, and 1 is converted to 6. The function of state 46 is SR0; upon finding 0 the next state is 47 with a move left. The 6 found in state 47 is converted to 1 and the next state is 43. When 7 is found in state 43 it indicates that the Expansion process is almost completed. The 7 found is converted to 6, and the next state is 48, whose function is SR. 01. Whenever either 0 or 1 is found in state 48 the next state is 49 with a move left. The 6 found in state 49 is converted to 0, and the next state if 50 with a move left. The 6 found in state 50 is converted to 7, and the next state is 20, a state in the Replacement mode.

If the symbol "." is found in state 19, the next state is 36, and the symbol "." is converted to the symbol "0". In state 36 the 6 found is converted to 7, and the next state is 37. The symbol "." is placed five cells to the right as previously described. In state

43 the 7 found is converted to 6, and the next state is 48. In state 48 the symbol "." is found, and the next state is 20, a state in the Replacement mode.

After the encoded word W_i has been replaced by the encoded word W_i' and the tail Y placed next to the encoded word W_i', thereby obtaining the encoded word W^{m+1}, it is necessary to convert every occurrence of 6 and 7 to occurrences of 0 and 1 respectively, and place the markers "7" and "5" at the beginning of the words W_1 and W^{m+1}, respectively, provided that neither of these words is the empty word Λ. If the ith substitution was a terminating substitution then transition is made to the Terminating mode. These tasks are the objective of the Cleanup mode.

At the beginning of the Cleanup mode recall that all of the encoded words $W_1, W_2, \ldots, W_i$ in the encoded substitutions contain the substitution symbols "6" and "7".

The Cleanup mode consists of states 32-36 inclusively and 62. NAS begins in state 32 over the cell to the right of the encoded word W_i' in the encoded word W^{m+1}. The function of state 32 is SL37. Upon finding 3 the next state is 33 with a move right. If 1 is found it is converted to 5 with a move left. Upon finding 3 in state 33 the next state is 32 with a move left. If the symbol "." is found in state 33, indicating that $W^{m+1} = \Lambda$, the next state is 33 with a move left. In state 32, 6 is converted to 0. Upon finding 7 the next

state is 34, whose function is SR234. If 2 is found in state 34 the next state is 36 with a move right. If 4 is found in state 36, indicating that the ith substitution was of the form $W_i \rightarrow \cdot\Lambda$, the next state is 41, a state in the Terminating mode. Whenever either 1, 2, or 3 is found in state 36 the next state is 35. If 3 is found in state 34 the next state is 35. Whenever 4 is found in state 34, indicating that the ith substitution was a terminating substitution, the next state is 41. The function of state 35 is SL, . In state 35, 6 and 7 are converted to 0 and 1 respectively. Upon finding the symbol "," in state 35 the next state is 62 with a move right. If 1 is found in state 62 the next state is 2 and 1 is converted to 7. Thus the markers are in place and NAS continues the simulation. If 2 is found in state 62, indicating $W_1 = \Lambda$, the next state is 58, and the encoded word W_1' is subsequently inserted to the left of the encoded word W^{m+1}.

Whenever it is found that the word W_i does not occur in the word W^m it is necessary to advance to the next substitution, if there is one, of the Normal Algorithm being simulated.

If the symbol "." is found in either of states 4 or 7, or 1 is found in state 16 it indicates that the encoded word W_i does not occur in the encoded word W^m. Whenever either the symbol "." is found in either state 4 or state 7, or the symbol "1" is found in state 16, it indicates that the encoded word W_i has a greater number of symbols than the tail of W^m which is being compared to W_i.

In the first case the next state is 52, and in the second case the next state is 56. Both are states in the Advancement mode.

The Advancement mode consists of states 52-60 inclusively. Recall that when transfer is made to state 56 NAS is over a cell containing a symbol of the encoded word W_i. The function of state 56 is SR. . Upon finding the symbol "." the next state is 52. The function of state 52 is SL37. Recall that when transfer is made to state 52 NAS is over the cell containing the symbol "." . In state 52, 5 and 6 are converted to 1 and 0 respectively. Upon finding 3 in state 52 the next state is 53 with a move right. The 1 found in state 53 is converted to 5 with a move left. Upon finding 3 the next state is 52. Thus the marker 5 is placed at the beginning of the encoded word W^m. Upon finding 7 in state 52 the next state is 54. Recall that this 7 occurs in the encoded word W_i. The function of state 54 is SR2. Upon finding in state 54 the 2 which occurs at the end of the encoded word W_i, the next state is 55. The function of state 55 is SR234. If 3 is found in state 55 it indicates that the last encoded substitution in the simulated Normal Algorithm is unamenable to the encoded word W^m. In this case NAS will stop in the Terminating mode. The next state is 41, a state in the Terminating mode. Whenever either 2 or 4 is found in state 55 the next state is 57 with a move right. If 1 is found in state 57 it is converted to 7, and the next state is 2, a state in the Comparison mode. Hence the marker 7 is placed

at the beginning of the encoded word W_{i+1}. If 2 is found in state 57, indicating that $W_{i+1} = \Lambda$, the next state is 58. In this case the encoded word W_{i+1}' is inserted before the encoded word W^m. The 1 found in state 58 is converted to 7, and the next state is 59. This 7 is a marker used in the Replacement mode. The function of state 59 is SR3. Upon finding 3 the next state is 60. The function of state 60 is SR.0. The 0 found, which is located to the right of and adjacent to the symbol "5", is converted to 7, and the next state is 37, a state in the Expansion mode. If the symbol "." is found in state 60, indicating $W^m = \Lambda$, the symbol "." is converted to the symbol "5" with a move right. Upon finding 6 the next state is 37, a state in the Expansion mode. The encoded word W^m is then moved right 5 cells as many times as necessary until the word W_{i+1}' has been inserted.

If 1 is found in state 16 it indicates that the encoded word W_i has a greater number of symbols than the tail of the encoded word W^m which is being compared to W_i. In this case the next state is 56, whose function is SR. . Upon finding the symbol "." the next state is 52. The subsequent operation of NAS is as previously described.

The Termination mode consists of states 41 and 51. The function of state 41 is SL, . In state 41, 6 and 7 are converted to 0 and 1 respectively. The symbol "," is converted to 0 and the next state

is 51. The function of state 51 is SR5. Recall that the 5 found appears to the right of 3. In state 51, 3 is converted to the symbol "," . The 5 found in state 51 is converted to 1 with a place move, and upon finding 1 in state 51 NAS stops. As previously mentioned, TASP prints out the part of the Turing Tape bounded on the left by the symbol "," , and on the right by the symbol "." . Hence only the encoded word W^m, and not the encoded substitutions of the Normal Algorithm being simulated, is printed out. If it is desired to print out the encoded substitutions in addition to the encoded word W^m it is sufficient merely to convert the quintuples 41, 51 R 0 and 51 3 51 R, to 41, 51 R, and 51 3 51 R3.

In order to illuminate the flow of the NAS program and further clarify the operation of NAS the Operator Programming technique developed by Kitov and Krinitsky (1959) is employed. The following operators are defined to describe the operation of a Turing machine. In the following definitions i and j are letters of the alphabet of the Turing machine.

B : Begin

L : Search left on the tape of the Turing machine.

L_i : Search left on the tape of the Turing machine for i, but do not change i.

L_{ij} : Search left on the tape of the Turing machine for i, and replace the i found by j.

R : Search right on the tape of the Turing machine.

R_i : Search right on the tape of the Turing machine for i, but do not change i.

R_{ij} : Search right on the tape of the Turing machine for i, and replace the i found by j.

F_i : The symbol "i" is found on the tape of the Turing machine and is not changed.

F_{ij} : The symbol "i" is found on the tape of the Turing machine and is changed to the symbol "j".

S : Stop.

As described in the above reference, the operators are written in the order that they are in control. An F-operator is an operator of the type F_i or F_{ij}. An L-operator (R-operator) is an operator of the type L, L_i or L_{ij} (R, R_i, or R_{ij}). The F-operators are logical operators whose condition is either satisfied or not. If more than one F-operator follows either an L- or R-operator, transfer of control is made to the F-operator whose condition is satisfied first. The Operator Program of NAS is shown in Figure 1.

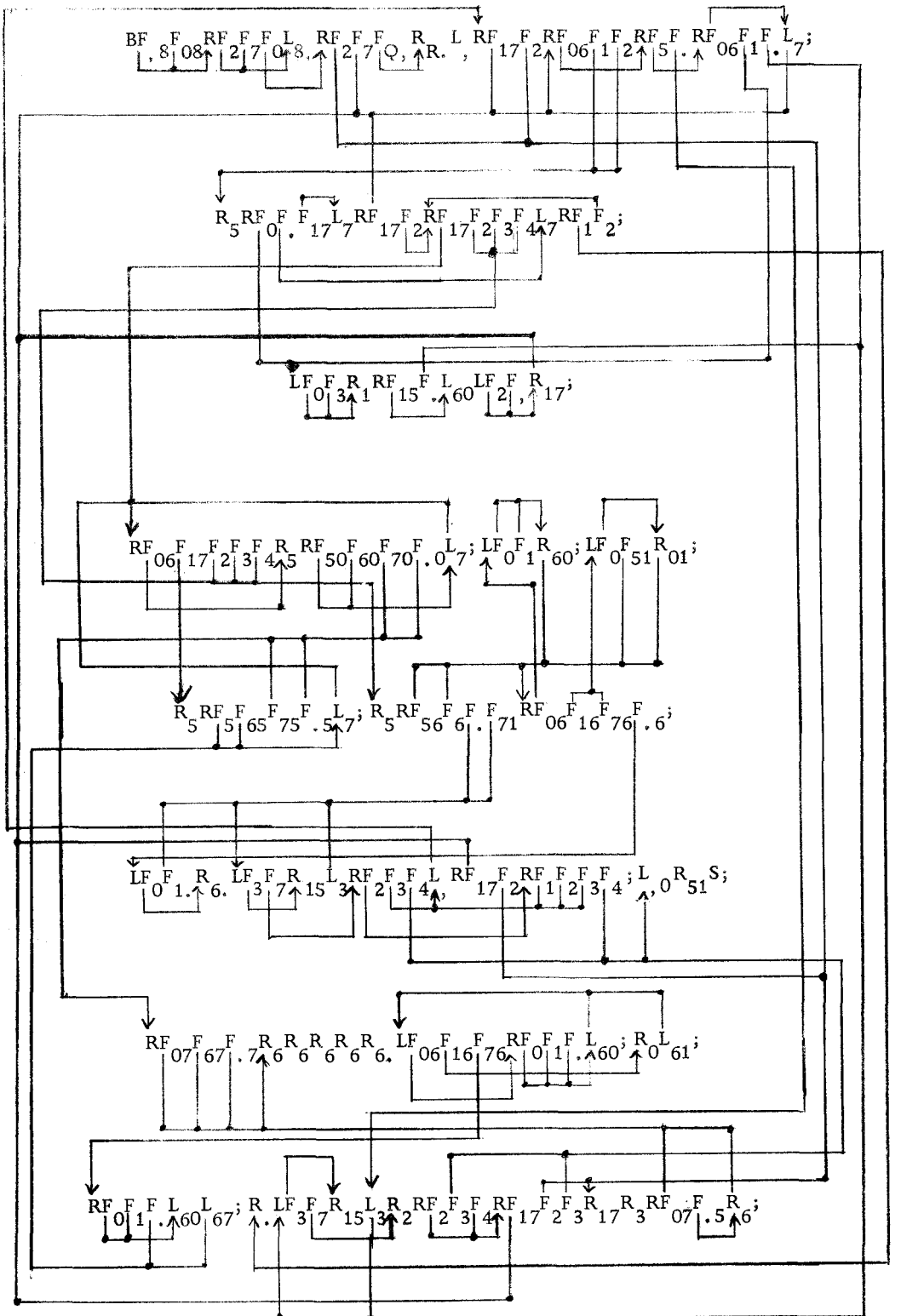

Figure 1. The operator program of NAS. For complete description, refer to the quintuple list in Figure 2.

```
[001,0] [066,R,8]   [010,3] [010,R,3]   [019,,] [036,R,0]   [031,6] [032,L,,]   [043,1] [046,R,6]   [056,3] [056,R,3]
[001,,] [066,R,8]   [010,4] [010,R,4]   [020,0] [020,L,0]   [032,0] [032,L,0]   [043,6] [043,L,6]   [056,4] [056,R,4]
[002,0] [003,R,6]   [010,5] [011,R,5]   [020,1] [020,L,1]   [032,1] [032,L,1]   [043,7] [048,R,6]   [056,5] [056,R,5]
[002,1] [010,R,1]   [011,0] [006,L,0]   [020,2] [020,L,2]   [032,2] [032,L,2]   [044,0] [045,L,0]   [056,6] [056,R,6]
[002,2] [010,R,2]   [011,1] [012,L,7]   [020,3] [020,L,3]   [032,3] [033,R,3]   [044,1] [045,L,1]   [056,7] [056,R,1]
[002,6] [002,R,6]   [011,5] [011,R,5]   [020,4] [020,L,4]   [032,4] [032,L,4]   [044,6] [044,R,6]   [056,,] [052,L,,]
[002,7] [002,R,7]   [011,6] [011,R,6]   [020,5] [020,L,5]   [032,5] [032,L,1]   [044,,] [045,L,,]   [057,2] [058,R,2]
[003,0] [003,R,0]   [011,,] [015,L,,]   [020,6] [020,L,6]   [032,6] [032,L,0]   [045,6] [043,L,0]   [057,1] [002,R,7]
[003,1] [003,R,1]   [012,0] [012,L,0]   [020,7] [017,R,7]   [032,7] [034,R,7]   [046,0] [047,L,0]   [057,3] [041,L,,]
[003,2] [003,R,2]   [012,1] [012,L,1]   [021,0] [021,R,0]   [033,1] [033,L,5]   [046,1] [047,L,1]   [058,1] [059,R,7]
[003,3] [003,R,3]   [012,2] [012,L,2]   [021,1] [021,R,1]   [033,3] [032,L,3]   [046,6] [046,R,6]   [059,0] [059,R,0]
[003,4] [003,R,4]   [012,3] [012,L,3]   [021,2] [021,R,2]   [033,,] [033,L,,]   [046,,] [047,L,,]   [059,1] [059,R,1]
[003,5] [004,R,5]   [012,4] [012,L,4]   [021,3] [021,R,3]   [034,0] [034,R,0]   [047,6] [043,L,1]   [059,2] [059,R,2]
[003,,] [053,L,,]   [012,5] [012,L,5]   [021,4] [021,R,4]   [034,1] [034,R,1]   [048,0] [049,L,0]   [059,3] [060,R,3]
[004,0] [005,L,6]   [012,6] [012,L,6]   [021,5] [022,R,1]   [034,2] [036,R,2]   [048,6] [048,R,6]   [059,4] [059,R,4]
[004,1] [006,L,1]   [012,7] [013,R,7]   [022,0] [022,R,0]   [034,3] [035,L,3]   [048,1] [049,L,1]   [060,0] [037,R,7]
[004,5] [004,R,5]   [013,1] [002,R,7]   [022,5] [020,L,5]   [034,4] [041,L,4]   [048,,] [020,L,,]   [060,5] [060,R,5]
[004,6] [004,R,6]   [013,2] [014,R,2]   [022,6] [020,L,5]   [035,0] [035,L,0]   [049,6] [050,L,0]   [060,6] [037,R,6]
[004,7] [004,R,5]   [013,6] [013,R,6]   [022,7] [036,R,5]   [035,1] [035,L,1]   [050,6] [020,L,7]   [060,,] [060,R,5]
[004,,] [052,L,,]   [014,1] [017,R,7]   [022,,] [036,R,5]   [035,2] [035,L,2]   [051,0] [051,R,0]   [062,0] [062,R,8]
[005,0] [005,L,0]   [014,2] [023,R,2]   [023,0] [023,R,0]   [035,4] [035,L,4]   [051,1] [051,R,1]   [062,1] [062,R,8]
[005,1] [005,L,1]   [014,3] [023,R,3]   [023,1] [023,R,1]   [035,6] [035,L,0]   [051,2] [051,R,2]   [062,2] [058,R,2]
[005,2] [005,L,2]   [014,4] [023,R,4]   [023,2] [023,R,2]   [035,7] [035,L,1]   [051,3] [051,R,,]   [062,5] [062,R,8]
[005,3] [005,L,3]   [015,1] [015,L,1]   [023,3] [023,R,3]   [035,,] [065,R,,]   [051,4] [051,R,4]   [062,7] [002,R,7]
[005,4] [005,L,4]   [015,0] [015,L,0]   [023,4] [023,R,4]   [036,0] [037,R,7]   [051,5] [051,L,1]   [062,8] [062,R,8]
[005,5] [005,L,5]   [015,2] [015,L,2]   [023,5] [024,R,5]   [036,1] [035,L,1]   [051,6] [051,R,6]   [062,Q] [063,R,,]
[005,6] [005,L,6]   [015,3] [015,L,3]   [023,6] [023,R,6]   [036,2] [035,L,2]   [051,7] [051,R,7]   [063,0] [063,R,0]
[005,7] [002,R,7]   [015,4] [015,L,4]   [024,0] [024,R,0]   [036,3] [035,L,3]   [051,,] STOP        [063,1] [063,R,1]
[006,0] [006,R,0]   [015,5] [015,L,5]   [024,5] [025,R,6]   [036,4] [041,L,4]   [051,,] STOP        [063,2] [063,R,2]
[006,3] [006,R,3]   [015,6] [015,L,6]   [024,6] [025,R,6]   [036,6] [037,R,7]   [052,0] [052,L,0]   [063,3] [063,R,3]
                                                                                                    [063,4] [063,R,4]
                                                                                                    [063,5] [063,R,5]
                                                                                                    [063,7] [063,R,7]
                                                                                                    [063,R] [064,L,,]
                                                                                                    [064,0] [064,L,0]
                                                                                                    [064,1] [064,L,1]
                                                                                                    [064,2] [064,L,2]
                                                                                                    [064,3] [064,L,3]
                                                                                                    [064,4] [064,L,4]
                                                                                                    [064,5] [064,L,5]
                                                                                                    [064,7] [064,L,7]
                                                                                                    [064,,] [065,R,,]
                                                                                                    [065,1] [002,R,7]
                                                                                                    [065,2] [058,R,2]
                                                                                                    [066,0] [062,R,8]
                                                                                                    [066,2] [066,L,2]
                                                                                                    [066,7] [066,L,7]
                                                                                                    [066,8] [062,R,,]

[006,5] [006,L,1]   [015,7] [016,R,7]   [024,7] [032,L,1]   [036,,] [037,R,7]   [052,1] [052,L,1]
[006,6] [006,L,0]   [016,1] [056,R,1]   [024,,] [032,L,,]   [037,0] [037,R,0]   [052,2] [052,L,2]
[006,1] [007,R,1]   [016,2] [014,R,2]   [025,0] [026,L,6]   [037,1] [037,R,1]   [052,3] [053,R,3]
[007,0] [007,R,0]   [016,6] [016,R,6]   [025,1] [028,L,6]   [037,6] [038,R,6]   [052,4] [052,L,4]
[007,1] [008,L,5]   [017,0] [018,R,6]   [025,5] [025,R,6]   [037,,] [037,R,6]   [052,5] [052,L,1]
[007,,] [052,L,,]   [017,1] [021,R,7]   [025,6] [025,R,6]   [038,6] [039,R,6]   [052,6] [052,L,0]
[008,0] [008,L,0]   [017,4] [023,R,4]   [025,7] [028,L,6]   [039,6] [040,R,6]   [052,7] [054,R,7]
[008,1] [008,L,1]   [017,2] [023,R,2]   [025,,] [030,L,6]   [040,6] [042,R,6]   [053,1] [053,L,5]
[008,2] [008,L,2]   [017,3] [023,R,3]   [026,0] [027,R,0]   [041,0] [041,L,0]   [053,3] [052,L,3]
[008,3] [008,L,3]   [017,6] [017,R,6]   [026,1] [027,R,1]   [041,1] [041,L,1]   [054,0] [054,R,0]
[008,4] [008,L,4]   [018,0] [018,R,0]   [026,6] [026,L,6]   [041,2] [041,L,2]   [054,1] [054,R,1]
[008,6] [009,L,0]   [018,1] [018,R,1]   [027,6] [025,R,0]   [041,3] [041,L,3]   [054,2] [055,R,2]
[009,1] [002,R,7]   [018,2] [018,R,2]   [028,0] [029,R,0]   [041,4] [041,L,4]   [055,0] [055,R,0]
[009,2] [009,R,2]   [018,3] [018,R,3]   [028,5] [025,R,1]   [041,5] [041,L,5]   [055,1] [055,R,1]
[009,6] [009,L,0]   [018,4] [018,R,4]   [028,6] [028,L,6]   [041,6] [041,L,6]   [055,2] [057,R,2]
[009,7] [009,L,1]   [018,5] [019,R,5]   [029,6] [025,R,1]   [041,7] [041,L,7]   [055,3] [041,L,3]
[009,,] [009,R,,]   [019,0] [019,R,0]   [030,0] [031,R,0]   [041,,] [051,R,0]   [055,4] [057,R,4]
[010,0] [010,R,0]   [019,5] [020,L,0]   [030,1] [032,L,,]   [041, ] [041,L,6]   [056,0] [056,R,0]
[010,1] [010,R,1]   [019,6] [020,L,0]   [030,5] [032,L,,]   [042,6] [043,L,,]   [056,1] [056,R,1]
[010,2] [010,R,2]   [019,7] [036,R,0]   [030,6] [030,L,6]   [043,0] [044,R,6]   [056,2] [056,R,2]
```

Figure 2. The quintuples of NAS.

Simulation by NAS

The Normal Algorithm D which doubles any word in the sense that $D : W \models \bullet WW$ was simulated by NAS. This algorithm is found in Markov (1961). This algorithm can be used for an alphabet B containing a finite number of letters. For this example the alphabet $B = \{\xi, \eta\}$ was selected. The alphabet $C = \{\alpha, \beta, \gamma\}$ consists of substitution letters. The alphabet of D is $A = B \cup C$. The initial word W^0 is over the alphabet B. The scheme of D is

$$\xi\eta\beta \rightarrow \eta\beta\xi$$

$$\eta\xi\beta \rightarrow \xi\beta\eta$$

$$\alpha\xi \rightarrow \xi\beta\xi\alpha$$

$$\alpha\eta \rightarrow \eta\beta\eta\alpha$$

$$\beta \rightarrow \gamma$$

$$\gamma \rightarrow \Lambda$$

$$\alpha \rightarrow \cdot\ \Lambda$$

$$\Lambda \rightarrow \alpha$$

Let the initial word $W^0 = \xi\eta$. Then

$$D : \xi\eta \vdash \alpha\xi\eta \vdash \xi\beta\xi\alpha\eta \vdash \xi\beta\xi\eta\beta\eta\alpha \vdash \xi\beta\eta\beta\xi\eta\alpha \vdash \xi\gamma\eta\beta\xi\eta\alpha \vdash$$

$$\xi\gamma\eta\gamma\xi\eta\alpha \vdash \xi\eta\gamma\xi\eta\alpha \vdash \xi\eta\xi\eta\alpha \vdash \bullet\,\xi\eta\xi\eta$$

The encoding used to simulate D was

$$\xi = 10$$

$$\eta = 100$$

$$\alpha = 1000$$

$$\beta = 10000$$

$$\gamma = 100000$$

The input tape to NAS was

,7010010000210010000102100101000021010000100210001021010000101000210001002100100001001000210000210000021000022100024210003501 00. 66666666666666666666

The output was 5010010100, where "5" is a substitution symbol replacing "1". As mentioned previously, the encoded scheme appearing on the tape is not printed out by NAS. NAS prints out only the encoded word which appears to the right of the symbol "3".

III. DESCRIPTION OF NAC

The Operation of NAC

The purpose of the Turing machine NAC (Normal Algorithm Converter) is to convert the quintuples and initial configuration of any Turing machine into an equivalent Normal Algorithm. A configuration of a Turing machine is a state q, a cell n, and the tape expression. An initial configuration is a configuration in which $q = q_0$ and $n = n_0$, where the Turing machine starts in state q_0 over the cell n_0. A tape is a function $f : I \rightarrow A_T$, where I is the set of integers and A_T is the Alphabet of the Turing machine, such that $f(i) = b$ for all but a finite number of $i \in I$, where b is the space of the Turing machine. The tape expression is the smallest word which can be written $f(i)f(i+1)\ldots f(i+k)$ such that $f(j) = b$ for $j < i$ or $j > i+k$ and $i \leq n \leq i+k$.

The Turing machine being converted, which will hereafter be called T, is encoded in the following way. A state q of T is encoded as 1 followed by an even number of, and at least two, occurrences of the symbol "0". The space b is encoded as 1000. Any other letter in the alphabet A_T is encoded as 1 followed by an odd number of, and at least five, occurrences of the symbol "0". The moves R, L, and P are encoded directly as R, L, and P respectively. The command stop is encoded as S. Let the alphabet

of T, A_T, be free of h. Then h is encoded as 10.

The modified tape expression is obtained from the tape expression by placing h at either end of the tape expression, and, if T is in state q over a particular letter ℓ, inserting the letter q to the left of that ℓ. The encoded modified tape expression is obtained by concatenating the encoded letters of the modified tape expression in the order given by the modified tape expression. The initial modified tape expression is a tape expression in which the letter q_0 appears before the letter $f(n_0)$, and the encoded initial modified tape expression is obtained from the initial modified tape expression by concatenating the encoded letters in the order given by the initial modified tape expression.

A quintuple $q\ell q'M\ell'$ of T, where M = R, L, or P is encoded by concatenating the encoded letters in the order given by the quintuple. The symbol "2" occurs between the encoded letters ℓ and q'.

The format of the input tape of NAC is as follows. The encoded initial modified tape word appears to the right of the symbol ",". The encoded letters of A_T appear to the right of the encoded initial modified tape word. In the first encoded letter listed the symbol "9" replaces the symbol "1". The symbol "*" appears to the right of the encoded letters of A_T. The encoded quintuples of T appear to the right of the symbol "*". Each encoded quintuple, except the last

quintuple listed, is followed by an occurrence of the symbol "*". The symbol "A" follows the last quintuple of T.

At the completion of operation by NAC the equivalent Normal Algorithm will appear on the tape of NAC to the right of the encoded quintuples. The equivalent Normal Algorithm format is identical to the format used by NAS. Hence it is possible for NAS to operate on a tape following operation by NAC.

The initial word of the Normal Algorithm equivalent to T is the initial modified tape expression of T. Each quintuple of T with a R or L move yields a substitution of the following two types as shown.

1. $q\ell q'R\ell' \quad q\ell h \rightarrow \ell' q' bh$
2. $q\ell q'L\ell' \quad hq\ell \rightarrow hq' b\ell'$

Substitutions of these two types correspond to the Turing machine T's ability to produce more tape as needed. These two types of substitutions are produced in the A mode of NAC, which will be described subsequently.

Every quintuple of T is of one of the following four types. Each quintuple corresponds to a substitution as shown.

3. $q\ell q'R\ell' \quad q\ell \rightarrow \ell' q'$
4. $q\ell q'L\ell' \quad \ell'' q\ell \rightarrow q'\ell'' \ell'$, for all $\ell'' \in A_T$

5. $q\ell q'Pl'$ $\quad q\ell \rightarrow q'\ell'$

6. $q\ell$ stop $\quad q\ell \rightarrow \cdot q\ell$

Substitutions of these four types are converted in the Q mode of NAC, which will be described subsequently. Note that each quintuple of type 4 produces as many substitutions as there are letters in A_T.

NAC converts the encoded quintuples of T into the encoded substitutions of the Normal Algorithm. Encoded substitutions of types 1 and 2 are produced first, and encoded substitutions of the other four types are produced next by NAC. The initial word of the Normal Algorithm, which is identical to the encoded initial modified tape expression, appears to the right of the encoded substitutions on the output tape of NAC.

There are two modes of operation of NAC, the A mode and the Q mode. Although NAC employs the A mode first and the Q mode second, the Q mode will be described first because the A mode uses many parts of the Q mode, and consequently an understanding of the Q mode is necessary.

The Q mode is partitioned into the QL and the QRPS modes. In the Q mode each encoded quintuple is converted to an encoded substitution. The conversion is made in the QL or QRPS mode depending upon whether the encoded quintuple contains either L or any of R, P, or S, respectively.

The QL mode consists of states 2-29 inclusively. The format of the tape of NAC is as described previously. The symbol "Q" appears to the right of the last encoded quintuple, and some encoded substitutions which were converted in the A mode appear to the right of Q.

NAC begins in state 2 to the left of the encoded quintuples. The function of state 2 is SR*Q. Upon finding * the next state is 3. The function of state 3 is SL8RLPS. If either R, P, or S is found the next state is 70, a state in the QRPS mode. If L is found the next state is 4, a state in the QL mode. If 8 is found, indicating that the encoded quintuple has already been converted to a substitution, the next state is 2. In the Q mode, as will be seen subsequently, every symbol of an encoded quintuple is converted to 8 after the encoded quintuple has been converted to a substitution.

The function of state 4 is SL*. Upon finding * the next state is 5 with a move right. The 1 found in state 5 is converted to 7, and the next state is 6. The 7 marks the encoded quintuple being converted. The function of state 6 is SL9. This 9 marks the beginning of the encoded letter of the encoded alphabet of T to be copied. Upon finding 9 in state 6 the next state is 20 with a place move.

States 20, 21, 22, 23, and 24 are used to copy the encoded letter at the right end of the tape of NAC. The function of state 20 is SR*01789. If 9 is found the next state is 21. The function of state 21

is SR6. Upon finding 6 the next state is 22 and 6 is converted to 1. The function of state 22 is SL9. Upon finding 9 the next state is 20. If 0 is found in state 20 the next state is 23, and 0 is converted to 8. The function of state 23 is SR6. The 6 found is converted to 0, and the next state is 22. If 1 is found in state 20 the next state is 7, and 1 is converted to 9. Hence the marker "9" is placed at the beginning of the next encoded letter of the encoded alphabet A_T.

The function of state 7 is SR7. The 7 which is found in state 7 marks the beginning of the quintuple being converted. Upon finding 7 the next state is 25 with a place move. States 25, 26, 27, 28, and 29 are used to copy symbols. The copying operation is similar to the one previously described. State 25 is used to find the symbol to be copied. States 26, 28, and 29 are used to write 0, 1, and 2 respectively at the right end of the tape. In state 26, 7 is converted to 1. Thus any occurrences of the symbol "7" in the quintuple being converted are converted to 1. The function of state 27 is SL7. The encoded letters q and ℓ are copied. Upon finding 2, which is located to the right of the encoded letter ℓ, in state 25 the next state is 29, and 2 is converted to 7. The encoded letter q' is copied next. Upon finding L, which is located to the right of the encoded letter q', in state 25 the next state is 12 with a move right. The 1 found in state 12 is converted to 7 with a move left. Thus the beginning of the encoded letter ℓ' is marked. The

function of state 12 is SL7. Upon finding 7 the next state is 17 with a move left. The 7 found in state 17 is converted to 2, and the next state is 8. States 8, 9, 10, and 11 are used to locate the encoded letter ℓ'' in the substitution so that it can be copied next to the encoded state q'. The function of state 8 is SR6. Upon finding 6 the next state is 9, whose function is SL24Q. In state 10, 1 is converted to 7. Upon finding any of 2, 4, or Q the next state is 11 with a move right. The 7 found in state 11 is converted to 9 with a place move, and the next state is 20. The encoded letter ℓ'' is then copied. Upon finding 7, which marks the right end of the encoded letter ℓ'', in state 20 the next state is 24, and 7 is converted to 1. Upon finding 6 in state 24 the next state is 27. The encoded letter ℓ' is then copied.

Upon finding * in state 25, indicating that the quintuple corresponding to an encoded letter of the encoded alphabet A_T has been converted, the next state is 13. State 13 is used to write 2 after the encoded letter ℓ' in the substitution. Upon finding 6 in state 13 the next state is 14, and 6 is converted to 2. The function of state 14 is SLQA. Since NAC is in the Q mode, Q is found. The next state is 15, whose function is SL9Q. In state 15, 8 is converted to 0. Upon finding 9, which marks the next encoded letter of the encoded alphabet A_T, the next state is 20. Corresponding to this encoded letter another quintuple is then converted.

When the last encoded letter of the encoded alphabet A_T has been copied, the symbol "*" is found in state 20. This symbol is converted to Q. After the quintuple has been converted to a substitution this Q is found in state 15. The next state is 18, and Q is converted to *. The function of state 18 is SL0. Recall that 0 and 1 have been converted to 8 and 9 respectively in the encoded alphabet A_T. In state 18, 8 and 9 are converted to 0 and 1 respectively. Upon finding 0 in state 18 NAC moves right. The 1 found is converted to 9, and the next state is 2. Thus the marker "9" is placed at the beginning of the encoded alphabet A_T.

The QRPS mode consists of states 70-82, 84-86, and 94-100 inclusively and states 88 and 90. As mentioned previously, upon finding either R, P, or S in state 3 the next state is 70, whose function is SL*. Upon finding *, which marks the left end of the quintuple being converted, the next state is 95. States 94-100 are used to copy symbols. State 95 is used to determine which symbol is to be copied. States 94, 96, and 98 are used to place 4, 1, and 0 respectively at the right end of the tape of NAC. State 99 is used to write 2 at the end of the encoded letter ℓ, and state 100 is used to write 2 at the end of the encoded letter q' in the substitution.

The encoded word $q\ell$ is copied. Upon finding 2 in state 95 the next state is 99, whose function is SR6. The 6 found is converted to 2, and the next state is 71, whose function is SL7A. Since NAC is

in the Q mode the 7 located at the beginning of the encoded letter ℓ is found. The next state is 72, whose function is SRRPS.

If R is found in state 72 the next state is 95. The encoded letter ℓ' is then copied. Upon finding *, which marks the right end of the encoded letter ℓ', in state 95 the next state is 74, whose function is SLRP. In state 74, 7 is converted to 1. Thus the marker at the beginning of the encoded letter ℓ is removed. Upon finding R the next state is 75 whose function is SL1. The 1 found is converted to 7, and the next state is 95. The encoded letter q' is then copied. Upon finding R in state 95 the next state is 100, whose function is SR6. The 6 found is converted to 2, and the next state is 76, whose function is SL7. Upon finding 7, which is located in the quintuple which was just converted, the next state is 2.

If P is found in state 72 the next state is 73, whose function is SL1. The 1 found, which is located at the beginning of the encoded letter q', is converted to 7 and the next state is 95. The encoded word $q'\ell'$ is then copied. Upon finding *, which marks the right end of the encoded letter ℓ' in the quintuple, the next state is 74, whole function is SLRP. In state 74, P is found, and the next state is 100. As described previously, 2 is copied, the converted quintuple is found, and the next state is 2.

If S is found in state 72 the next state is 86 with a move left, and S is converted to 8. Recall that in this case the quintuple is of

the type "qℓ stop". The 2 found in state 86 is converted to 4, and the next state is 85, whose function is SL*Q. In state 85, 7 and 8 are converted to 1 and 0 respectively. Upon finding either * or Q the next state is 95, and the encoded word qℓ is copied. Upon finding 4, which is located at the right end of the encoded letter ℓ, in state 95, the next state is 94, and 4 is converted to 2. The function of state 94 is SR6. The 6 found is converted to 4, and the next state is 76. Thus the terminating substitution has been constructed. As described previously, the converted quintuple is found and the next state is 2.

After all of the quintuples have been converted the Q which is located to the right of the encoded quintuples is found in state 2. The symbol "3" is then placed to the right of the encoded substitutions. The next state is 78, whose function is SR6. Upon finding 6 the next state is 88. If 2 is found in state 88, 2 is converted to 3, and the next state is 79. If 4 is found in state 88, NAC moves right. The 6 found is converted to 3, and the next state is 79.

Each symbol of the encoded quintuples and encoded alphabet A_T are converted to 8, and the encoded initial modified tape expression is copied. The function of state 79 is SLQ. Upon finding Q the next state is 80, whose function is SL9. In state 80 every symbol is converted to 8. Upon finding 9, which marks the first encoded letter of the encoded alphabet A_T, the next state is 81, whose function

is SL, . Upon finding the symbol "," in state 81 the next state is 95. The encoded modified initial tape expression is then copied until the marker "9" is found. Upon finding 9 in state 95 the next state is 82, and 9 is converted to 8. Every symbol of the encoded modified initial tape expression is then converted to 8. The function of state 82 is SL, . In state 82, 7 is converted to 8. The , found in state 82 is also converted to 8, and the next state is 84, whose function is SR36. Upon finding 3 in state 84 the next state is 90 with a move right. The 1 found in state 90 is converted to 5, and the next state is 84. Thus the marker "5" is placed in the desired location. The 6 found in state 84 is converted to R with a place move. Upon finding R in state 84 NAC stops.

As mentioned previously, the function of the A mode is to convert quintuples of types 1 and 2 into substitutions. The A mode is partitioned into the AL and AR modes. Recall that in the A mode the symbol "A" appears to the right of the encoded quintuples.

The AL mode consists of state 1 and states 30-48 inclusively. NAC begins in state 1 at the left end of the tape whose format has been previously described. Upon finding either , or : the next state is 30. The function of state 30 is SR*. Upon finding * the next state is 31, whose function is SRRLPS. If either P or S is found, the quintuple is not converted in the A mode, and the next state is 30. If L is found the next state is 32, whose function

is SL*. Upon finding * the next state is 33 with a move right. The 1 found in state 33 is converted to 7, which marks the quintuple being converted. The next state is 34. The encoded letters h and b are then written. The function of state 34 is SR6. Upon finding 6, 35 is the next state. The 6 found in state 35 is converted to 0, and 36 is the next state. The 6 found is converted to 1, and the next state is 27 in the QL mode. This 1 is the first symbol of the encoded letter q. The encoded letters q, ℓ, and q' are then copied as in the QL mode description. The marker "7" is placed at the beginning of the encoded letter ℓ' as described previously. In state 8 the symbol "A" is found, and the next state is 37. The substitution at this point is of the form $hq\ell \rightarrow q'$. It is necessary to insert the encoded letter h in front of q', and the encoded letter b after q'. This is accomplished by states 37, 38, 39, 40, 41, 42, and 48. The function of state 37 is SR6. States 37, 38, 39, 40, 41, 42, and 48 are used to write the word 0010001 on the tape. Upon finding 6 in state 48 the next state is 43, whose function is SL2. Upon finding 2 the next state is 44 with a move right. Upon finding 0 the next state is 45 with a move right. The 0 found in state 45 is converted to 1, and the next state is 27 in the QL mode. Thus the encoded letter h is inserted in front of, and the encoded letter b is inserted after the encoded letter q'.

The encoded letter ℓ' is then copied as described in the QL

mode description. The symbol "2" is written at the end of the substitution in state 13, and the next state is 14. Upon finding A in state 14 the next state is 46. The function of state 46 is SL7. Upon finding 7, which marks the quintuple just converted, the next state is 30 and the process continues.

The AR mode consists of states 50, 52, 53, 55, 57-63 inclusively, and state 65. If R is found in state 31 the next state is 50, whose function is SL*. Upon finding * the next state is 95 in the QR mode. The encoded word $q\ell$ is then copied as described previously. Upon finding A in state 99 the next state is 52, whose function is SR6. The encoded letter h is then written. States 52, 53, and 55 are used to write 102 on the tape. Upon finding 6 in state 55 the next state is 57, whose function is SLA. Upon finding A the next state is 71 in the QR mode. The encoded word ℓ'q' is then copied. Upon finding A in state 100 the next state is 58. The enconded word bh is then written on the tape. States 58, 59, 60, 61, 62, 63, and 65 are employed to write the word 1000102. Upon finding 6 in state 65 the next state is 46, whose function is SL7. Upon finding 7 the next state is 30.

When all of the quintuples of type 1 or 2 have converted in the A mode, A is found in state 30. The next state is 47, and A is converted to Q. The function of state 47 is SL9. In state 47, 7 and 8 are converted to 1 and 0 respectively. Hence all the substitution

symbols "7" and "8" are removed. The 9 found is located at the beginning of the encoded alphabet A_T. Upon finding 9 the next state is 2, a state in the Q mode.

The Operator program of NAC is given in the next two figures.

NAS Operating on NAC

In this example NAC operates on the Turing machine T whose quintuples are

1 A 2 R A

2 A 2 R A

2 b 3 L A

3 A 3 P b

3 b stop

The input tape of T is ...bAAAbAbb..., and T starts in state 1 over the first occurrence of the symbol "A". When T stops its tape is ...bAAbAAb... .

The machine T is encoded as follows for NAC.

b = 1000

A = 100000

1 = 10000

2 = 1000000

3 = 100000000

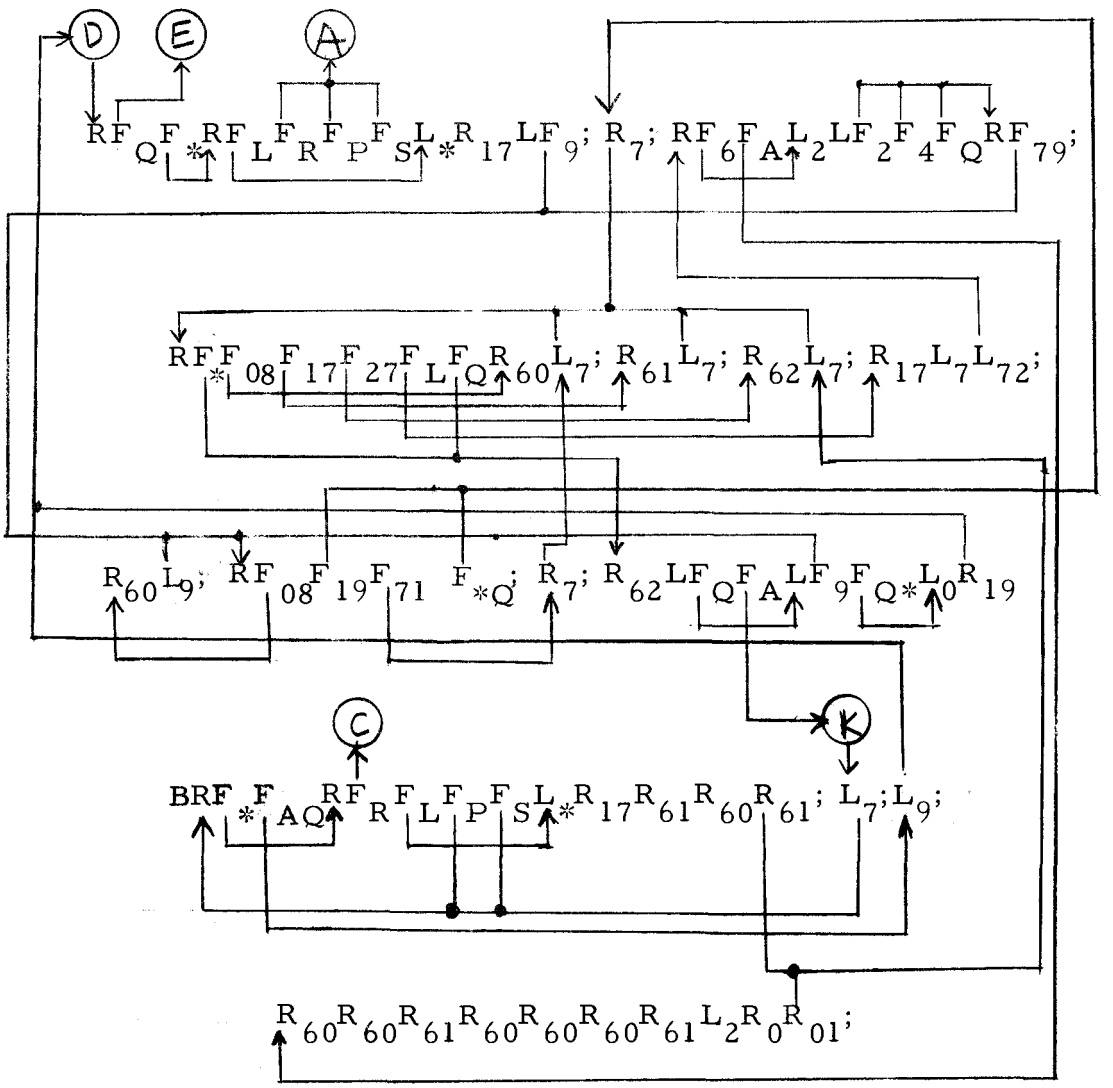

Figure 3. The operator program of the QL and AL mode of NAC. For complete description, refer to the quintuple list in Figure 5.

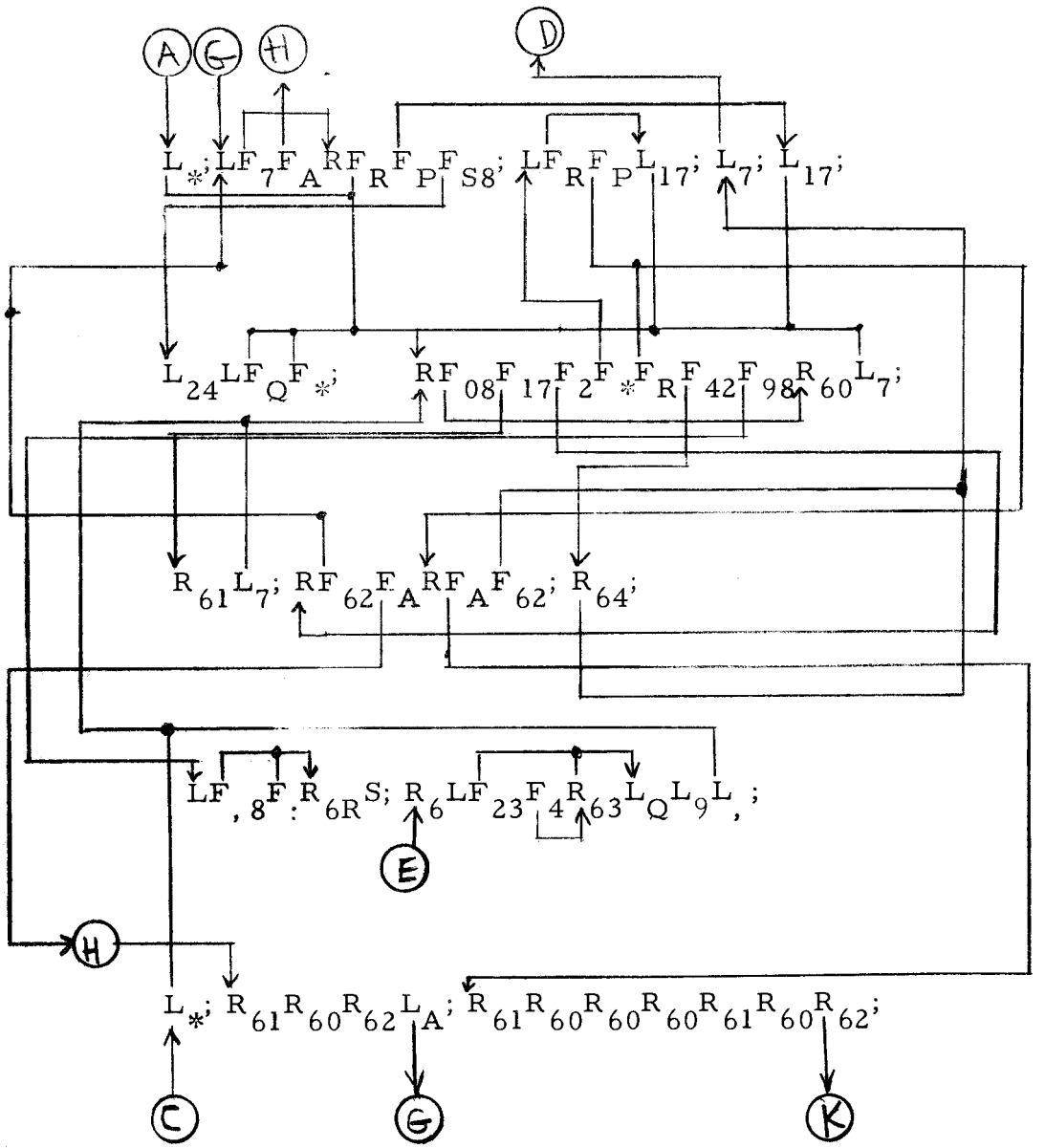

Figure 4. The operator program of the QRPS and AR mode of NAC.

```
[001,,] [030,R,,]  [007,*] [007,R,*]  [015,8] [015,L,0]  [022,7] [022,L,7]  [027,0] [027,L,0]
[001,1] [030,R,1]  [007,S] [007,R,S]  [015,9] [020,P,9]  [022,8] [022,L,8]  [027,1] [027,L,1]
[002,0] [002,R,0]  [008,0] [008,R,0]  [015,R] [015,L,R]  [022,9] [020,R,9]  [027,2] [027,L,2]
[002,1] [002,R,1]  [008,1] [008,R,1]  [015,L] [015,L,L]  [022,R] [022,L,R]  [027,4] [027,L,4]
[002,2] [002,R,8]  [008,2] [008,R,2]  [015,P] [015,L,P]  [022,L] [022,L,L]  [027,7] [025,R,7]
[002,7] [002,R,8]  [008,4] [008,R,4]  [015,Q] [018,L,*]  [022,P] [022,L,P]  [027,8] [027,L,8]
[002,8] [002,R,8]  [008,6] [009,L,6]  [015,*] [015,L,*]  [022,Q] [022,L,Q]  [027,9] [027,L,1]
[002,9] [002,R,9]  [008,7] [008,R,7]  [015,S] [015,L,S]  [022,*] [022,L,*]  [027,R] [027,L,R]
[002,R] [002,R,8]  [008,8] [008,R,R]  [016,0] [016,R,0]  [022,S] [022,L,S]  [027,L] [027,L,L]
[002,L] [002,R,8]  [008,L] [008,R,L]  [016,1] [019,R,9]  [023,0] [023,R,0]  [027,P] [027,L,P]
[002,P] [002,R,8]  [008,P] [008,R,P]  [016,8] [016,L,0]  [023,1] [023,R,1]  [027,Q] [027,L,Q]
[002,Q] [078,R,Q]  [008,Q] [008,R,Q]  [016,9] [016,L,1]  [023,2] [023,R,2]  [027,A] [027,L,A]
[002,S] [002,R,S]  [008,A] [037,R,A]  [017,7] [008,R,2]  [023,4] [023,R,4]  [027,*] [027,L,*]
[002,*] [003,R,*]  [008,*] [008,R,*]  [018,0] [018,R,0]  [023,6] [022,L,0]  [027,S] [027,L,S]
[002,1] [002,R,1]  [008,S] [008,R,S]  [018,1] [002,R,9]  [023,7] [023,R,7]  [028,0] [028,R,0]
[003,0] [003,R,0]  [009,0] [009,L,0]  [018,8] [018,L,0]  [023,8] [023,R,8]  [028,1] [028,R,1]
[003,1] [003,R,1]  [009,1] [009,L,7]  [018,9] [018,L,1]  [023,R] [023,R,R]  [028,2] [028,R,2]
[003,2] [003,R,2]  [009,2] [010,L,2]  [019,0] [019,R,0]  [023,L] [023,R,L]  [028,4] [028,R,4]
[003,7] [002,R,8]  [010,0] [010,L,0]  [019,1] [019,R,1]  [023,P] [023,R,P]  [028,6] [027,L,0]
[003,8] [002,R,8]  [010,1] [010,L,7]  [019,2] [019,R,2]  [023,Q] [023,R,Q]  [028,8] [028,R,0]
[003,R] [070,L,R]  [010,2] [011,R,2]  [019,7] [002,R,1]  [023,*] [023,R,*]  [028,R] [028,R,R]
[003,L] [004,L,L]  [010,4] [011,R,4]  [019,8] [019,R,8]  [023,S] [023,R,S]  [028,L] [028,R,L]
[003,P] [070,L,P]  [010,Q] [011,R,Q]  [019,R] [019,R,R]  [024,0] [024,R,0]  [028,P] [028,R,P]
[003,*] [003,R,*]  [011,7] [020,P,9]  [019,L] [019,R,L]  [024,1] [024,R,1]  [028,Q] [028,R,Q]
[003,S] [070,L,S]  [012,1] [012,L,7]  [019,P] [019,R,P]  [024,2] [024,R,2]  [028,A] [028,R,A]
[004,0] [004,L,0]  [012,7] [017,L,7]  [019,*] [019,R,*]  [024,6] [027,L,1]  [028,*] [028,R,*]
[004,1] [004,L,1]  [012,8] [012,L,0]  [019,S] [019,R,S]  [024,7] [024,R,1]  [028,S] [028,R,S]
[004,2] [004,L,2]  [012,L] [012,L,L]  [020,0] [023,R,8]  [025,0] [028,R,8]  [029,0] [029,R,0]
[004,*] [005,R,*]  [013,0] [013,R,0]  [020,1] [007,R,9]  [025,1] [026,R,7]  [029,1] [029,R,1]
[005,1] [006,L,7]  [013,1] [013,R,1]  [020,7] [024,R,1]  [025,2] [029,R,7]  [029,2] [029,R,2]

[006,0] [006,L,0]  [013,2] [013,R,2]  [020,8] [020,R,8]  [025,7] [026,R,7]  [029,6] [027,L,2]
[006,1] [006,L,1]  [013,4] [013,R,4]  [020,9] [021,R,9]  [025,R] [025,R,8]  [029,R] [029,R,R]
[006,2] [006,L,2]  [013,6] [014,L,2]  [020,*] [007,R,Q]  [025,L] [012,R,L]  [029,L] [029,R,L]
[006,7] [006,L,8]  [013,P] [013,R,R]  [021,0] [021,R,0]  [025,Q] [013,R,Q]  [029,P] [029,R,P]
[006,8] [006,L,8]  [013,L] [013,R,L]  [021,1] [021,R,1]  [025,A] [013,R,A]  [029,Q] [029,R,Q]
[006,9] [020,P,9]  [013,P] [013,R,P]  [021,2] [021,R,2]  [025,*] [013,R,*]  [029,A] [029,R,A]
[006,R] [006,L,R]  [013,Q] [013,R,Q]  [021,4] [021,R,4]  [025,S] [025,R,S]  [029,*] [029,R,*]
[006,L] [006,L,L]  [013,A] [013,R,A]  [021,6] [022,L,1]  [026,0] [026,R,0]  [029,S] [029,R,S]
[006,P] [006,L,P]  [013,*] [013,R,*]  [021,7] [021,R,7]  [026,1] [026,R,1]  [030,0] [030,R,0]
[006,*] [006,L,*]  [013,S] [013,R,S]  [021,8] [021,R,8]  [026,2] [026,R,2]  [030,1] [030,R,1]
[006,S] [006,L,S]  [014,0] [014,L,0]  [021,R] [021,R,R]  [026,4] [026,R,4]  [030,2] [030,R,2]
[007,0] [007,R,0]  [014,1] [014,L,1]  [021,L] [021,R,L]  [026,6] [027,L,1]  [030,5] [030,R,5]
[007,1] [007,R,1]  [014,2] [014,L,2]  [021,P] [021,R,P]  [026,7] [026,R,1]  [030,9] [030,R,9]
[007,2] [007,R,2]  [014,4] [014,L,4]  [021,Q] [021,R,Q]  [026,R] [026,R,R]  [030,R] [030,R,R]
[007,7] [025,P,7]  [014,Q] [015,L,Q]  [021,*] [021,R,*]  [026,L] [026,R,L]  [030,L] [030,R,L]
[007,8] [007,R,R]  [014,A] [046,L,A]  [021,S] [021,R,S]  [026,P] [026,R,P]  [030,P] [030,R,P]
[007,9] [007,R,9]  [015,0] [015,L,0]  [022,0] [022,L,0]  [026,Q] [026,R,Q]  [030,A] [047,L,Q]
[007,R] [007,R,R]  [015,1] [015,L,1]  [022,1] [022,L,1]  [026,A] [026,R,A]  [030,*] [031,R,*]
[007,L] [007,R,L]  [015,2] [015,L,2]  [022,2] [022,L,2]  [026,*] [026,R,*]  [030,,] [030,R,,]
[007,P] [007,R,P]  [015,7] [015,L,7]  [022,4] [022,L,4]  [026,S] [026,R,S]  [030,1] [030,R,1]
```

Figure 5. The quintuples of NAC.

```
[031,0] [031,R,0]   [046,S] [046,L,S]   [071,Q] [071,L,Q]   [080,*] [080,L,8]   [095,R] [100,R,R]   [099,0] [099,R,0]
[031,1] [031,R,1]   [047,0] [047,L,0]   [071,A] [052,R,A]   [080,S] [080,L,8]   [095,P] [095,R,P]   [099,1] [099,R,1]
[031,2] [031,R,2]   [047,1] [047,L,1]   [071,*] [071,L,*]   [081,0] [081,L,0]   [095,Q] [074,L,Q]   [099,2] [099,R,2]
[031,R] [050,L,R]   [047,2] [047,L,2]   [071,S] [071,L,S]   [081,1] [081,L,1]   [095,A] [074,L,A]   [099,4] [099,R,4]
[031,L] [032,L,L]   [047,7] [047,L,1]   [072,0] [072,R,0]   [081,8] [081,L,8]   [095,*] [074,L,*]   [099,6] [071,L,2]
[031,P] [030,R,P]   [047,R] [047,L,0]   [072,1] [072,R,1]   [081,9] [081,L,8]   [096,0] [096,R,0]   [099,8] [099,R,8]
[031,S] [030,R,S]   [047,9] [002,R,9]   [072,2] [072,R,2]   [081,,] [095,R,,]   [096,1] [096,R,1]   [099,R] [099,R,R]
[032,0] [032,L,0]   [047,R] [047,L,R]   [072,8] [072,R,8]   [082,7] [082,L,8]   [096,2] [096,R,2]   [099,L] [099,R,L]
[032,1] [032,L,1]   [047,L] [047,L,L]   [072,R] [095,R,R]   [082,8] [082,L,8]   [096,3] [096,R,3]   [099,P] [099,R,P]
[032,2] [032,L,2]   [047,P] [047,L,P]   [072,P] [073,L,P]   [082,1] [084,R,1]   [096,4] [096,R,4]   [099,0] [099,R,Q]
[032,*] [033,R,*]   [047,*] [047,L,*]   [072,S] [086,L,8]   [082,,] [084,R,8]   [096,6] [097,L,1]   [099,A] [052,R,A]
[033,1] [034,R,7]   [047,S] [047,L,S]   [073,0] [073,L,0]   [084,0] [084,R,0]   [096,8] [096,R,8]   [099,*] [099,R,*]
[034,0] [034,R,0]   [048,6] [043,L,1]   [073,1] [095,P,7]   [084,1] [084,R,1]   [096,9] [096,R,9]   [099,S] [099,R,S]
[034,1] [034,R,1]   [050,0] [050,L,0]   [074,7] [074,L,1]   [084,2] [084,R,2]   [096,R] [096,R,R]   [100,0] [100,R,0]
[034,2] [034,R,2]   [050,1] [050,L,1]   [074,8] [074,L,0]   [084,3] [090,R,3]   [096,L] [096,R,L]   [100,1] [100,R,1]
[034,6] [035,R,1]   [050,2] [050,L,2]   [074,R] [075,L,R]   [084,4] [084,R,4]   [096,P] [096,R,P]   [100,2] [100,R,2]
[034,R] [034,R,R]   [050,*] [095,R,*]   [074,P] [100,R,P]   [084,6] [084,P,R]   [096,Q] [096,R,Q]   [100,4] [100,R,4]
[034,L] [034,R,L]   [052,0] [052,R,0]   [075,0] [075,L,0]   [084,8] [084,R,8]   [096,A] [096,R,A]   [100,6] [076,L,2]
[034,P] [034,R,P]   [052,1] [052,R,1]   [075,1] [095,P,7]   [084,Q] [084,R,Q]   [096,*] [096,R,*]   [100,7] [100,R,7]
[034,Q] [034,R,A]   [052,2] [052,R,2]   [076,0] [076,L,0]   [084,R] STOP        [096,S] [096,R,S]   [100,8] [100,R,0]
[034,A] [034,R,A]   [052,6] [053,R,1]   [076,1] [076,L,1]   [085,7] [085,L,1]   [097,0] [097,L,0]   [100,R] [100,R,R]
[034,*] [034,R,*]   [053,6] [055,R,0]   [076,2] [076,L,2]   [085,8] [085,L,0]   [097,1] [097,L,1]   [100,L] [100,R,L]
[034,S] [034,R,S]   [055,6] [057,L,2]   [076,7] [002,R,1]   [085,Q] [095,R,Q]   [097,2] [097,L,2]   [100,P] [100,R,P]
[035,6] [036,R,0]   [057,0] [057,L,0]   [076,8] [076,L,0]   [085,*] [095,R,*]   [097,3] [097,L,3]   [100,Q] [100,R,Q]
[036,6] [027,L,1]   [057,1] [057,L,1]   [076,R] [076,L,R]   [086,2] [085,L,4]   [097,4] [097,L,4]   [100,A] [068,R,A]
[037,0] [037,R,0]   [057,2] [057,L,2]   [076,L] [076,L,L]   [088,0] [088,R,0]   [097,7] [095,R,7]   [100,*] [100,R,*]
[037,1] [037,R,1]   [057,A] [071,L,A]   [076,P] [076,L,P]   [088,2] [079,L,3]   [097,8] [097,L,8]   [100,S] [100,R,S]
[037,2] [037,R,2]   [058,0] [058,R,0]   [076,Q] [076,L,Q]   [088,4] [088,R,4]   [097,9] [097,L,9]
[037,6] [038,R,0]   [058,1] [058,R,1]   [076,*] [076,L,*]   [088,6] [079,L,3]   [097,R] [097,L,R]
[038,6] [039,R,0]   [058,2] [058,R,2]   [076,S] [076,L,S]   [090,1] [084,R,5]   [097,L] [097,L,L]
[039,6] [040,R,1]   [058,6] [059,R,1]   [078,0] [078,R,0]   [094,0] [094,R,0]   [097,P] [097,L,P]

[040,6] [041,R,0]   [059,6] [060,R,0]   [078,1] [078,R,1]   [094,1] [094,R,1]   [097,Q] [097,L,Q]
[041,6] [042,R,0]   [060,6] [061,R,0]   [078,2] [078,R,2]   [094,2] [094,R,2]   [097,A] [097,L,A]
[042,6] [048,R,0]   [061,6] [062,R,0]   [078,4] [078,R,4]   [094,4] [094,R,4]   [097,*] [097,L,*]
[043,0] [043,L,0]   [062,6] [063,R,1]   [078,6] [088,L,6]   [094,6] [076,L,4]   [097,S] [097,L,S]
[043,1] [043,L,1]   [063,6] [065,R,0]   [079,0] [079,L,0]   [094,8] [094,R,8]   [098,0] [098,R,0]
[043,2] [044,R,2]   [065,6] [046,L,2]   [079,1] [079,L,1]   [094,R] [094,R,R]   [098,1] [098,R,1]
[044,0] [045,R,0]   [070,0] [070,L,0]   [079,2] [079,L,2]   [094,L] [094,R,L]   [098,2] [098,R,2]
[044,1] [044,R,1]   [070,1] [070,L,1]   [079,4] [079,L,4]   [094,P] [094,R,P]   [098,3] [098,R,3]
[045,0] [027,L,1]   [070,2] [070,L,2]   [079,Q] [080,L,Q]   [094,Q] [094,R,Q]   [098,4] [098,R,4]
[046,0] [046,L,0]   [070,*] [095,R,*]   [080,0] [080,L,8]   [094,A] [094,R,A]   [098,6] [097,L,0]
[046,1] [046,L,1]   [071,0] [071,L,0]   [080,1] [080,L,8]   [094,*] [094,R,*]   [098,8] [098,R,8]
[046,2] [046,L,2]   [071,1] [071,L,1]   [080,2] [080,L,8]   [094,S] [094,R,S]   [098,9] [098,R,9]
[046,7] [030,R,7]   [071,2] [071,L,2]   [080,7] [080,L,8]   [095,0] [098,R,8]   [098,R] [098,R,R]
[046,8] [046,L,0]   [071,4] [071,L,4]   [080,8] [080,L,8]   [095,1] [096,R,7]   [098,L] [098,R,L]
[046,R] [046,L,R]   [071,7] [072,R,7]   [080,9] [081,L,9]   [095,2] [099,R,2]   [098,P] [098,R,P]
[046,L] [046,L,L]   [071,8] [071,L,8]   [080,R] [080,L,8]   [095,4] [094,R,2]   [098,Q] [098,R,Q]
[046,P] [046,L,P]   [071,R] [071,L,R]   [080,L] [080,L,8]   [095,7] [096,R,7]   [098,A] [098,R,A]
[046,A] [046,L,A]   [071,L] [071,L,L]   [080,P] [080,L,8]   [095,8] [095,R,8]   [098,*] [098,R,*]
[046,*] [046,L,*]   [071,P] [071,L,P]   [080,Q] [080,L,8]   [095,9] [082,L,8]   [098,S] [098,R,S]
```

Figure 5. (continued)

It is possible to encode a state as 100, although it was not done in this example.

The information part of the input tape of NAC is 000:1550,10 100001000001000001000001000100000100000109000100000*100001000002100000OR100000*1000000100000210000000L100000*1000000001000002100000000P1000*1000000001000 2SA. The word ":1550" is used to call NAS, whose number is 155, after NAC has finished. The word "000" appears to the left of the symbol ":" because when the symbol ":" occurred at the left end of the information part of the tape TASP would not call NAS after NAC had finished. Apparently the symbol ":" was being changed by TASP, but the program functioned as desired when the word "000" was used.

The output of NAC is the encoded substitutions and encoded initial word h1AAAbAh of the Normal Algorithm corresponding to T. This output is shown in Appendix III, and is bounded on the left by Q and on the right by R. As mentioned previously, the symbol "6" is used as the space of the Turing machines NAS and NAC. Notice that the encoded alphabet and encoded quintuples of T have been "erased" by converting every symbol to the symbol "8".

NAS then simulates the operation of this Normal Algorithm, and the output of NAS, shown in Appendix III, is the encoded word hAA3bAAh, which is identified with AAbAA, the information part of the tape of T when T stops.

BIBLIOGRAPHY

Coffin, R. W., H. E. Goheen and R. W. Stahl. 1963. Simulation of a Turing machine on a digital computer. In: Proceedings of the American Federation of Information Processing Societies, Las Vegas, 1963. Vol 24. [Santa Monica, California] p. 35-43.

Detlovs, V. K. 1958. The equivalence of normal algorithms and recursive functions. Trudy Matematicheskogo Instituta Imeni V. A. Steklova 52:75-139 (Translated in American Mathematical Society Translation 23:15-81. 1963)

Hermes, H. 1965. Enumerability, decidability, computability, tr. by G. T. Herman and O. Plassmann. New York, Academic. 245 p.

Kitov, A. N. and N. A. Krinitsky. 1959. Electronic digital machines and programming, tr. by Harry Goheen. Sec. 32. Moscow, State Publishing House of Physics-Mathematics Literature. 572 p.

McCune, M. [1968] Operation instructions for TASP (Turing Automaton Simulation Program). Unpublished description of a computer program. Corvallis, Oregon State University, Computer Center.

Markov, A. A. 1961. Theory of algorithms, tr. by Jacques J. Schorr-Kon et al. Jerusalem, The Israel Program for Scientific Translations. 444 p.

Nelson, R. J. 1968. Introduction to automata. New York, Wiley. 400 p.

Trakhtenbrot, B. A. 1963. Algorithms and automatic computing machines, tr. by J. Kristian, J. McCawley and S. Schmitt. Boston, Heath. 101 p.