AN ABSTRACT OF THE THESIS OF

Sandra Elaine Anderson for the M.S. in Mathematics

(Date thesis is presented)

Title SOME COMPUTATIONAL SCHEMES EQUIVALENT TO TURING MACHINES

Abstract approved

Computational scheme, equivalence, and Turing machine are defined. Some computational schemes are examined and shown to be equivalent to the computational scheme of a Turing machine.
SOME COMPUTATIONAL SCHEMES EQUIVALENT TO TURING MACHINES

by

SANDRA ELAINE ANDERSON

A THESIS

submitted to

OREGON STATE UNIVERSITY

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

June 1964
APPROVED:

Professor of Mathematics

In Charge of Major

Chairman of Department of Mathematics

Dean of Graduate School

Date thesis is presented August 5, 1963

Typed by Carol Baker
ACKNOWLEDGEMENT

The author would like to thank Dr. Harry Goheen for his assistance in the preparation of this paper.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>I. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>II. THE TURING MACHINE AND THE LOOPED TURING MACHINE</td>
<td>4</td>
</tr>
<tr>
<td>III. THE DAVIS MACHINE</td>
<td>14</td>
</tr>
<tr>
<td>IV. RIGHT AND RIGHT-LEFT MACHINES</td>
<td>18</td>
</tr>
<tr>
<td>BIBLIOGRAPHY</td>
<td>27</td>
</tr>
<tr>
<td>APPENDIX</td>
<td>28</td>
</tr>
</tbody>
</table>
SOME COMPUTATIONAL SCHEMES EQUIVALENT TO TURING MACHINES

CHAPTER I

INTRODUCTION

The automatic calculating device which later became known as the Turing machine was invented by A. M. Turing in 1936 (4, p. 231). This device is essentially a black box which is capable of assuming a specified finite number of internal states. At one end of the box is a reading head, through which passes a tape. The tape is divided into squares, each square bearing a symbol. The machine is capable of viewing only one square at a time. Its behavior is completely determined by the symbol on the scanned square and the internal state. The machine erases the symbol on the scanned square, replaces it by another (or the same) symbol, moves one square to the left, one square to the right, or not at all, and goes into a new (or the same) internal state.

Although this device is so simple that at first glance it appears to be almost worthless, it has become a powerful tool in the study of algorithms and solvability, as well as in more practical applications. The elusive concept of "algorithm" can be satisfactorily defined as what a Turing machine does. This necessitates of course that rigorous definitions of a Turing machine and what it does be given. Before doing this, we examine more closely the operation of the
machine, in order to determine what should be included in the definition.

The problem to be solved is coded in machine language and written on the tape. Since the machine operates in discrete steps, the problem must occupy only a finite portion of the tape, or otherwise the machine would not be able to examine all the marked squares in a finite amount of time. However in general it is not known beforehand how much room on the tape the machine will need in order to finish the calculation; therefore we must require that the tape itself be unbounded in length. At every stage in the operation only a finite number of squares have anything written on them, but this number may be as large as is necessary. The other squares, which are blank, we assume to be marked with the symbol B. It is often convenient to put a fence around the portion of the tape on which the machine is working. For this purpose we introduce a special symbol for use as an end marker, so that all the squares to the right of the upper end marker and all the squares to the left of the lower end marker are blank, that is to say, the symbol written on them is B. These marks may be moved when the machine requires more or less tape (2, p. 3).

A formal description of a Turing machine must include a list of the symbols recognizable by the machine, a list of the internal states which the machine is capable of assuming, a table showing what the
machine will do when in a certain internal state and confronted by a certain symbol, and a way of determining when a given computation is finished. A description of the operation of the machine on a particular problem should consist of a sequence of sub-descriptions of each stage in the operation. These sub-descriptions should include the expression on the tape, the square scanned, and the internal state of the machine.

There are many ways of formalizing these properties. We shall examine some of the more obvious ways and show that the resulting machines are not essentially different; that they are in fact equivalent in the sense that if a machine of one type be supplied with a tape, then there are machines of all the other types which, when supplied with the same tape, will produce the same output as the original machine if that machine produces an output, and will produce no output if the original machine produces no output.

The results of this investigation are of interest in connection with the formal definition of algorithm. A mathematician may wish to define algorithm in terms of some machine other than a Turing machine. If he chooses one of the machines discussed in this paper, his definition will be identical with that made in terms of a Turing machine.
CHAPTER II

THE TURING MACHINE AND THE LOOPED TURING MACHINE

By "a function on a set" is meant "a function whose domain is contained in the set".

Let $X$ be a set, and let $f$ be a function on $X$ into $X$. $f$ will be called a computational scheme on $X$, and $F(x)$ will be called the output of $x$ under $f$. If $X' \subseteq X$ and $f(X') \subseteq X'$, then $f$ is a computational scheme on $X'$. If $f$ is a computational scheme on $X$ and $g$ is a computational scheme on $Y$, and if there is a 1-1 correspondence between $X$ and $Y$ such that if $x \in X$, $x$ corresponds to $y \in Y$ and $f(x)$ is defined then $g(y)$ is defined and $f(x)$ corresponds to $g(y)$, then we say $f$ on $X$ implies $g$ on $Y$. If $f$ on $X$ implies $g$ on $Y$ and $g$ on $Y$ implies $f$ on $X$ under the same 1-1 correspondence, we say $f$ on $X$ and $g$ on $Y$ are equivalent.

Let $Q, S,$ and $M$ be disjoint finite sets. $Q$ is the set of internal states. $S$, the set of symbols recognizable by the machine, is called an alphabet. $S$ must contain an element $B$ which will be placed on the blank squares of the tape. $M$ is the set of orders which tell the machine how to move to scan another square. $M$ must contain exactly three elements $P, L,$ and $R$. $P$ means "stay where you are", 
L means "move left" and R means "move right".

A tape is a function a from the set of integers into S such that for all but a finite number of integers n, \( a(n) = B \).

A complete configuration is an ordered triple \((a, q, n)\) where a is a tape, q an element of Q, and n an integer. Let X be the set of all complete configurations.

Let T be a function on \( Q \times S \) into \( S \times M \times Q \). T will be called a Turing table.

Define a function \( F \) on X into X as follows:

If \( T(q, a(n)) = s \) for \( j \), let \( F(a, q, n) = (b, q, k) \) where

\[
\begin{align*}
    b(i) &= a(i) \text{ for } i \neq n \\
    b(n) &= s
\end{align*}
\]

and \( k = n \) if \( m = P \)
\( k = n + 1 \) if \( m = R \)
\( k = n - 1 \) if \( m = L \).

If \( T(q, a(n)) \) is undefined, let \( F(a, q, n) \) be undefined.

Let \( x \in X \). If there is a non-negative integer \( n \) such that \( F^n(x) = F^{n+1}(x) \), let \( f(x) = F^n(x) \). If there is no such non-negative integer, let \( f(x) \) be undefined. \( f \) is a computational scheme on X and will be called the computational scheme determined by the Turing table T.

The triple \((T, F, f)\) is called a Turing machine.
The functional values of \( f \) are determined by what one would intuitively wish to call a stop order. By varying the definition slightly, we obtain a computational scheme which is not essentially different from that of a Turing machine.

If there is a non-negative integer \( n \) such that \( F^n(x) \) is defined but \( F^{n+1}(x) \) is not defined let \( g(x) = F^n(x) \), and let \( g(x) \) be undefined otherwise. Then \( g \) is a computational scheme on \( X \) which is not equivalent to \( f \). However, there is a Turing machine \((T', F', f')\) with \( f' = g \). Let

\[
T'(q, s) = sPq \quad \text{if} \quad T(q, s) \text{ is undefined}
\]

\[
T'(q, s) \text{ be undefined if } T(q, s) = sPq.
\]

\[
T'(q, s) = T(q, s) \text{ for all other } q, s.
\]

Thus \( F(x) \) is undefined \( \Leftrightarrow \) \( F'(x) = x \)

\[
F(x) = x \quad \Leftrightarrow \quad F'(x) \text{ is undefined}
\]

\[
F(x) \neq x \quad \Leftrightarrow \quad F'(x) = F(x).
\]

To see that \( f' = g \), let \( x \in \text{dom } f' \) and let \( n \) be the least non-negative integer such that \( F'^n(x) = F'^{n+1}(x) \). Then \( F^n(x) = F'^n(x) \) and \( F^{n+1}(x) \) is undefined. Hence \( g(x) = F^n(x) = F'^n(x) = f'(x) \).

Now suppose \( x \in X - \text{dom } f' \). Then, either for all \( n \) \( F'^n(x) \neq F'^{n+1}(x) \), or there is some \( n \) for which \( F'^n(x) \) is undefined. In either case, for all \( n \) \( F^n(x) \) is defined; hence \( x \in X - \text{dom } g \).

Thus \( f' = g \).
Similarly, if \( g' \) is the computational scheme which produces outputs if and only if \( T' \) does not know what to do, then \( g' = f \).

We now investigate another method of varying the stop order of the machine.

Let \( n \) be the least non-negative integer such that for some positive integer \( k \), \( F^n(x) = F^{n+k}(x) \), and let \( g(x) = F^n(x) \). The triple \((T, F, g)\) will be called a looped Turing machine. If \( f \) is the computational scheme determined by the Turing table \( T \), then \( f \) on \( X \) implies \( g \) on \( X \); in fact \( f \) is equivalent to \( g \) on the domain of \( f \).

(f is actually a computational scheme on its domain, since \( f(f(x)) = f(x) \); thus \( f(\text{dom } f) \subseteq \text{dom } f \).) In general \( f \) and \( g \) are not equivalent on \( X \), for there may be \( x \)'s such that \( f(x) \) is not defined but \( g(x) \) is defined. However, the greater applicability of the looped Turing machine is only apparent, for we have the

THEOREM:

Let \( Q \) be a set of internal states, \( S \) an alphabet, \( X \) the set of complete configurations, and \((T, F, g)\) a looped Turing machine for \( X \). Then there is a Turing machine \((T', F', f)\) with set of complete configurations \( Y' \) and a subset \( Y \) of \( Y' \) such that \( f \) is a computational scheme on \( Y \) and \( f \) on \( Y \) is equivalent to \( g \) on \( X \).
Proof:

We wish that the Turing machine will mimic the action of the looped Turing machine. Suppose that both machines are supplied with the same tape and are started in the same internal state, scanning the same square. If, in the process of operation, the looped Turing machine enters a complete configuration which it has already entered at some previous step in the operation, it will stop. Accordingly, the Turing machine must remember all of the complete configurations and compare each new configuration with the previous ones. If at any time a complete configuration is the same as an earlier one, we wish the Turing machine to stop in a configuration corresponding to the one on the looped Turing machine. If the looped Turing machine does not stop, we wish that the Turing machine also will not stop.

A Turing machine can remember in two ways: by writing symbols on its tape and by its internal states. The machine we will construct will copy the non-blank portion of the tape of the looped Turing machine, together with symbols which indicate the complete configuration, to the left of the non-blank portion of its own tape. After each move of the looped machine, the Turing machine can then compare the complete configuration with the earlier complete configurations. If there is an agreement, the Turing machine erases the extraneous
material and stops. If there is no agreement, the Turing machine copies the new tape and then performs the next move of the looped machine.

To construct \((T', F', f)\) let its alphabet \(S'\) contain four copies \(s_1', s_2', s_3', \text{ and } s_4'\) of every \(s \in S\), and two copies \(q_1'\) and \(q_2'\) of every \(q \in Q\). Furthermore let there be a distinguished element \(h \in S'\). Finally let \(B \in S'\).

Let \(Q'\), the set of internal states of \((T', F', f)\), consist of all ordered triples of the form \(q_i s'\) where \(q \in Q\), \(i\) runs from 0 to 12, and \(s' \in S'\).

Let \(Y'\) be the set of all complete configurations of \((T', F', f)\).

We will construct a 1-1 mapping from \(X\) into \(Y'\) and let \(Y\) be the image of \(X\) under this mapping.

If \((a, q, n) \in X\), let \((a'_i, q'_i, n'_i) = F^i(a, q, n)\). Define

\[
A = \bigcup_i \{ k \mid a_i(n) \neq B \} \cup \{ n'_i \},
\]

and suppose \(K = \inf A\) and \(N = \sup A\).

If \(K\) and \(N\) are both finite, let the tape \(a\), which is a mapping from the integers into \(S\), correspond to the tape \(a'\) which maps the integers into \(S'\) in such a way that

\[
\begin{align*}
    a'(K-1) &= h \\
    a'(N+1) &= h \\
    a'(k) &= (a(k))_1 \text{ for } K \leq k \leq N \\
    a'(k) &= B \text{ for } k < K-1 \text{ or } k > N+1.
\end{align*}
\]
If at least one of $K$ or $N$ is infinite, let $a'$ correspond to the tape $a'$ which maps $k$ into $(a(k))_1$ if $a(k) \neq B$ and into $B$ if $a(k) = B$.

Let $(a, q, n)$ correspond to $(a', qp_0B, n)$ if $(a, q, n) \in X - g(x)$ and to $(a', qp_12B, n)$ if $(a, q, n) \in g(x)$.

Let $Y$ be the set of all complete configurations in $Y'$ which correspond to a complete configuration in $X$. $Y$ is properly contained in $Y'$.

We must show 1) if $x \in X$ and $x$ corresponds to $y$, then, if $g(x)$ exists so does $f(y)$ and $g(x)$ corresponds to $f(y)$, which implies that $f(Y) \subseteq Y$ and hence $f$ is a computational scheme on $Y$; and 2) if $g(x)$ fails to exist, $f(y)$ also fails to exist. These two statements together imply that $g$ on $X$ is equivalent to $f$ on $Y$. The truth of the first statement is obvious on examining the Turing table $T'$ (see Appendix). To prove the second, suppose that $(a, q, n) \in X - \text{dom } g$. The sequence $F^i(a, q, n)$ never becomes periodic. If the interval $[K, N]$ defined above is finite, the Turing machine $(T', F', f)$ copies the succeeding complete configurations $F^i(a, q, n)$ of $(T, F, g)$, never finding a complete configuration which agrees with an earlier one. Hence $(T', F', f)$ will never stop. If the interval $[K, N]$ is infinite, then the tape $a'$ corresponding to $a$ contains no $h$. The sequence of complete configurations $F^i(a', qp_0B, n)$ eventually becomes $(b, qp_1B, k), (b, qp_1B, k-1), (b, qp_1B, k-2), \ldots$ which is not a constant
sequence. Hence $f(a', q_0B, n)$ is defined only if $g(a, q, n)$ is defined.

**QED**

We have seen that if $g$ is the computational scheme of a looped Turing machine on $X$, there is a computational scheme $f$ of a Turing machine and a set $Y$ such that $g$ on $X$ is equivalent to $f$ on $Y$. Since $f$ is in point of fact a computational scheme on a larger set $Y'$, one might suspect that Turing machines are more powerful than looped Turing machines.

**THEOREM:**

Let $Q$ be a set of internal states, $S$ an alphabet, $X$ the set of complete configurations, and $(T, F, f)$ a Turing machine for $X$. Then there is a looped Turing machine $(T', F', g)$ with set of complete configurations $Y'$ and a set $Y \subseteq Y'$ such that $g$ is a computational scheme on $Y$ and $f$ on $X$ is equivalent to $g$ on $Y$.

**Proof:**

If $s \in S$, let $s$ and a copy $s^2$ of $s$ be in $S'$, the alphabet of $(T', F', g)$. Let a distinguished element $h$ be in $S'$. Let $Q'$, the set of internal states of $(T', F', g)$, consist of all ordered pairs of the form $qp_i$ where $q \in Q$ and $i$ runs from 0 to 7. Let $Y'$ be the set
of complete configurations of $S'$ and $Q'$, and let $Y$ be the image of $X$ in $Y'$ under the 1-1 correspondence defined below.

If $(a, q, n) \in X$ let $(a_i, q_i, n_i) = F^i(a, q, n)$. Let $A = \bigcup_i (\{k \mid a_i(k) \neq B\} \cup \{n_i\})$, and suppose $K = \inf A$.

Let the tape $a$ correspond to $a'$ where

$$a'(k) = a(k) \text{ for } k \neq K-1$$

$$a'(K-1) = h \text{ if } K \text{ is finite}.$$

Let $(a, q, n)$ correspond to

$$(a', q_p_0, n) \text{ if } (a, q, n) \in X - f(X)$$

$$(a, q_p_7, n) \text{ if } (a, q, n) \in f(X).$$

Define a Turing table as follows:

<table>
<thead>
<tr>
<th>$(q_p_0)$</th>
<th>$s$</th>
<th>$s^2 L(q_p_1)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(q_p_1)$</td>
<td>$s$</td>
<td>$sL(q_p_1)$</td>
</tr>
<tr>
<td>$(q_p_1)$</td>
<td>$h$</td>
<td>$hL(q_p_2)$</td>
</tr>
<tr>
<td>$(q_p_2)$</td>
<td>$h$</td>
<td>$hL(q_p_2)$</td>
</tr>
<tr>
<td>$(q_p_2)$</td>
<td>$B$</td>
<td>$hR(q_p_3)$</td>
</tr>
<tr>
<td>$(q_p_3)$</td>
<td>$h$</td>
<td>$hR(q_p_3)$</td>
</tr>
<tr>
<td>$(q_p_3)$</td>
<td>$s$</td>
<td>$sR(q_p_3)$</td>
</tr>
<tr>
<td>$(q_p_3)$</td>
<td>$s^2$</td>
<td>$s.m(q_p_0)$ if $T(q, s) = s_i m q_j = s P q$</td>
</tr>
<tr>
<td>$(q_p_3)$</td>
<td>$s^2$</td>
<td>$s^2 L(q_p_4)$ if $T(q, s) = s P q$</td>
</tr>
<tr>
<td>$(q_p_4)$</td>
<td>$s$</td>
<td>$sL(q_p_4)$</td>
</tr>
<tr>
<td>$(q_p_4)$</td>
<td>$h$</td>
<td>$BL(q_p_5)$</td>
</tr>
</tbody>
</table>

**Mark square scanned**

**Locate lower end**

**Add one more $h$ at lower end**

**Make move of Turing machine if this is not a stop order**

**Erase $h$'s**
The looped machine mimics the Turing machine, keeping track of the number of steps by writing \( h \) at the lower end of the tape for each step in the operation of the Turing machine. If the Turing machine stops, the looped machine erases all the \( h \)'s and stops in the complete configuration which corresponds to the one describing the Turing machine. If the Turing machine does not stop, the looped machine either prints \( h \)'s and more \( h \)'s, thus never going into a loop, or else looks in vain for an \( h \) that is not there. Thus \( f \) on \( X \) is equivalent to \( g \) on \( Y \).

QED
CHAPTER III

THE DAVIS MACHINE

Martin Davis (1, p. 5-7) describes a machine which is even simpler than the Turing machine. The Davis machine is capable of printing or moving, but not both in the same step.

Let $Q, S, M$ and $X$ be as before, and let $D$ be a function on $Q \times S$ into $(S \times Q) \cup (M \times Q)$. For $(a, q, n) \in X$ define

$$F(a, q, n) = (b, q_j, n) \text{ if } D(q, a(n)) = sq_j$$
$$= (a, q_j, n+1) \text{ if } D(q, a(n)) = Rq_j$$
$$= (a, q_j, n-1) \text{ if } D(q, a(n)) = Lq_j$$

where $b$ is the tape defined by

$$b(k) = a(k) \text{ if } k \neq n$$
$$b(n) = s.$$ 

If $D(q, a(n))$ is undefined, let $F(a, q, n)$ be undefined.

If there is a non-negative integer $k$ such that $F^k(a, q, n) = F^{k+1}(a, q, n)$ let $f(a, q, n) = F^k(a, q, n)$. $f$ is a computational scheme on $X$, and the triple $(D, F, f)$ will be called a Davis machine.

THEOREM:

If $(D, F, f)$ is a Davis machine, then there is a Turing machine $(T', F', f')$ such that $F' = F$ and $f' = f$. 
Proof:

It suffices to show that there is a Turing machine \((T', F', f')\) with \(F' = F\).

Let \(T(q, s) = s_Pq_j \) if \(D(q, s) = s_iq_j\)

\[= s_Rq_j \text{ if } D(q, s) = Rq_j \]

\[= s_Lq_j \text{ if } D(q, s) = Lq_j. \]

Let \(T(q, s)\) be undefined if \(D(q, s)\) is undefined.

For \((a, q, n) \in X\) we have

\(F(a, q, n)\) is defined \(\iff\) \(D(q, a(n))\) is defined

\(\iff\) \(T(q, a(n))\) is defined

\(\iff\) \(F'(a, q, n)\) is defined

Thus dom \(F = \text{dom } F'\).

\(D(q, a(n)) = s_iq_i \rightarrow T(q, a(n)) = s_Pq_i\)

\(\rightarrow F'(a, q, n) = (b, q_i, n) \text{ where } b(k) = a(k), k \neq n\)

\(b(n) = s.\)

\(\rightarrow F(a, q, n) = (c, q_i, n) \text{ where } c(k) = a(k), k \neq n\)

\(c(n) = s.\)

\(\rightarrow b = c\)

\(\rightarrow F'(a, q, n) = F(a, q, n).\)
\[ D(q, a(n)) = Rq_i \quad \rightarrow \quad T(q, a(n)) = a(n) Rq_i \]
\[ \rightarrow F'(a, q, n) = (a, q_i, n+1) \]
\[ \rightarrow F(a, q, n) = (a, q_i, n+1) \]
\[ \rightarrow F'(a, q, n) = F(a, q, n) . \]

\[ D(q, a(n)) = Lq_i \quad \rightarrow \quad T(q, a(n)) = a(n) Lq_i \]
\[ \rightarrow F'(a, q, n) = (a, q_i, n-1) \]
\[ \rightarrow F(a, q, n) = (a, q, n) \]
\[ \rightarrow F'(a, q, n) = F(a, q, n) . \]

Therefore \( F' = F \) and consequently \( f' = f \).

**THEOREM:**

Let \((T, F, f)\) be a Turing machine with set of complete configurations \( X \). Then there is a Davis machine \((D, G, g)\) with set of complete configurations \( X' \) containing \( x \) such that \( f \) on \( X \) is equivalent to \( g \) on \( X \).

**Proof:**

If \( q \) is an internal state of \((T, F, f)\), let \( q \) and two copies \( Rq \) and \( Lq \) of \( q \) be internal states of \((D, G, g)\). Let the alphabet of \((D, G, g)\) be the alphabet of \((T, F, f)\).

If \( T(q, s) = s_i Pq_j \) let \( D(q, s) = s_i q_j \)
\[ = s_i Rq_j \quad \text{let} \quad D(q, s) = s_i (Rq_j)\)
\[ D(Rq_j, s_i) = Rq_j \]
\[ = s_i Lq_j \quad \text{let} \quad D(q, s) = s_i (Lq_j)\)
\[ D(Lq_j, s_i) = Lq_j . \]
g is a computational scheme on X, since \( g(X') \subseteq X \). For, let \((a, q', n) \in X'\) and suppose \( G(a, q', n) = (a, q', n) \). Then \( D(q'a(n)) = a(n)q' \). This can happen only if \( T(q', a(n)) = a(n)Pq' \); hence \( q' \in Q \) and \((a, q', n) \in X\). Thus \( g(X') \subseteq X \) and in particular \( g(X) \subseteq X \).

Let the 1-1 correspondence between \( X \) and \( X \) be the identity correspondence. It is clear from the definition of \( D \) that for all \( x \in X \) either \( F(x) = G(x) \) or \( F(x) = G^2(x) \), in the strong sense of equality: if one side is defined so is the other and they are equal.

Thus, if there is a non-negative integer \( n \) such that \( F^n(x) = F^{n+1}(x) \), then there is a non-negative integer \( k \leq 2n \) such that \( G^k(x) = G^{k+1}(x) \).

If there is no such \( n \), then for no \( k \) does \( G^k(x) = G^{k+1}(x) \). Hence \( f = g \) on \( X \).

\[QED\]
CHAPTER IV

RIGHT AND RIGHT-LEFT MACHINES

Let \( Q, s, \) and \( M \) be as before and let \( T \) be a Turing table on \( Q \times S \). An instantaneous description is a function \( A \) from the set of integers into \( Q \cup S \) such that for exactly one integer \( N \) is \( A(N) \in Q \), and for all but a finite number of integers \( n \), \( A(n) = B \). Call the set of all instantaneous descriptions \( Y \). Let \( F \) be a function on \( Y \) into \( Y \) defined by:

If \( T(A(N), A(N+1)) = aPq \), let \( F(A) = B \), where \( B(n) = A(n), n \neq N, n \neq N+1 \)
\[ B(N) = q \]
\[ B(N+1) = s \]

If \( T(A(N), A(N+1)) = sRq \), let \( F(A) = B \), where \( B(n) = A(n), n \neq N, n \neq N+1 \)
\[ B(N) = s \]
\[ B(N+1) = q \]

If \( T(A(N), A(N+1)) = sLq \), let \( F(A) = B \), where \( B(n) = A(N), n \neq N-1 \)
\[ n \neq N \]
\[ n \neq N+1 \]
\[ B(N-1) = q \]
\[ B(N) = A(N-1) \]
\[ B(N+1) = s \].
Let \( f(A) = B \) if and only if there is a non-negative integer \( n \) such that \( F^n(A) = F^{n+1}(A) = B \). \( f \) is a computational scheme on \( Y \).

The triple \( (T, F, f) \) is called a right machine.

**THEOREM:**

Suppose \( T \) is a Turing table on \( Q \times S \). Let \( X \) be the set of complete configurations and \( Y \) the set of instantaneous descriptions. Then the computational scheme of the Turing machine \( (T, G, g) \) on \( X \) is equivalent to the computational scheme of the right machine \( (T, F, f) \) on \( Y \).

**Proof:**

If the functions \( F \) and \( G \) are equivalent, then so are the computational schemes \( f \) and \( g \).

Let \( (a, q, N) \in X \) correspond to \( A \in Y \) if and only if

\[
A(n) = a(n) \quad \text{for} \quad n < N
\]

\[
A(N) = q
\]

\[
A(n) = a(n-1) \quad \text{for} \quad n > N.
\]

To show \( F(a) \) corresponds to \( G(a, q, N) \) we consider three cases. For the first case, suppose \( T(q, a(N)) = sPq_i \). Here the result is clear. For the second case, suppose \( T(q, a(N)) = sRq_i \). Then we have \( G(a, q, N) = (b, q, N+1) \) where \( b(n) = a(n) \) for \( n \neq N \)

\[
b(N) = s.
\]
Furthermore, $F(A) = B$ where $B(n) = A(n)$

$= a(n)$

$= b(n)$ for $n < N$

$B(N) = s$

$= b(N)$

$B(N+1) = q_i$

$B(n) = A(n)$

$= a(n-1)$

$= b(n-1)$ for $n > N+1$.

This is exactly the condition that $F(A)$ correspond to $G(a, q, n)$. For the third and last case, suppose $T(q, a(n)) = sLq_i$. Then $G(a, q, N) = (b, q_i, N-1)$ where $b(n) = a(n)$ for $n \neq N$

$b(N) = s$,

and $F(A) = B$ where we have $B(n) = A(n)$

$= a(n)$

$= b(n)$ for $n < N-1$

$B(N-1) = q_i$

$B(N) = A(N-1)$

$= a(N-1)$

$= b(N-1)$

$B(N+1) = s$

$= b(N)$

$= b(N+1-1)$
\[ B(n) = A(n) \]
\[ = a(n-1) \]
\[ = b(n-1) \text{ for } n > N + 1. \]

Again, \( F(A) \) corresponds to \( G(a, q, n) \). Thus \( F \) on \( Y \) is equivalent to \( G \) on \( X \), which implies that \( f \) on \( Y \) is equivalent to \( g \) on \( X \).

QED

Suppose \( Q \) is the disjoint union of two sets \( QR \) and \( QL \). The internal states which are elements of \( QR \) we imagine to be right-facing states, and those which are elements of \( QL \) we imagine to be left-facing. This mental picture of the Turing machine gives a flexibility which has been useful in the investigation of word problems (5, p. 493).

Let \( S \) be an alphabet, \( T \) a Turing table on \( Q \times S \), and \( Y \) the set of instantaneous descriptions. Let \( A \in Y \) and suppose \( A(N) \in Q \). Define \( G(A) \) as follows:

I. \( A(N) \in QR \)

i) \( T(A(N), A(N+1)) = sPq \)

Let \( G(A) = B \) where \( B(n) = A(n) \) for \( n \neq N, n \neq N + 1 \)

\[ B(N) = q \]
\[ B(N+1) = s \]
ii) $T(A(N), A(N+1)) = sRq$

Let $G(A) = B$ where $B(n) = A(n)$ for $n \neq N, n \neq N+1$

$B(N) = s$

$B(N+1) = q$

iii) $T(A(N), A(N+1)) = sLq$

Let $G(A) = B$ where $B(n) = A(n)$ for $n \neq N-1, n \neq N, n \neq N+1$

$B(N-1) = q$

$B(N) = A(N-1)$

$B(N+1) = s$

iv) Let $G(A)$ be undefined if $T(A(N), A(N+1))$ is undefined.

II. $A(N) \in QL$

i) $T(A(N), A(N-1)) = sPq$

Let $G(A) = B$ where $B(n) = A(n)$ for $n \neq N-1, n \neq N$.

$B(N-1) = s$

$B(N) = q.$

ii) $T(A(N), A(N-1)) = sRq$.

Let $G(A) = B$ where $B(n) = A(n)$ for $n \neq N-1, n \neq N, n \neq N+1$.

$B(N-1) = s$

$B(N) = A(N+1)$

$B(N+1) = q.$
iii) \( T(A(N), A(N-1)) = sLq. \)

Let \( G(A) = B \) where \( B(n) = A(n) \) for \( n \neq N-1, n \neq N \)

\[
\begin{align*}
B(N-1) &= q \\
B(N) &= s.
\end{align*}
\]

iv) Let \( G(A) \) be undefined if \( T(A(N), A(N-1)) \) is undefined.

Let a computational scheme \( g \) on \( Y \) be defined in the usual manner: \( g(A) = B \) if and only if there is a non-negative integer \( n \) such that \( G^n(A) = G^{n+1}(A) = B. \)

The triple \( (T, G, g) \) is called a right-left machine.

**THEOREM:**

If \( (T, F, f) \) is a right machine, then \( (T, F, f) \) is a right-left machine.

**Proof:**

If \( Q \) is the set of internal states of \( (T, F, f) \) let \( QR = Q \) and \( QL = \emptyset. \)

**QED**

**THEOREM:**

Suppose \( (T, G, g) \) is a right-left machine with set of instantaneous descriptions \( Y. \) Then there is a right machine \( (T', F, f) \) with set
of instantaneous descriptions $X'$ and a subset $X$ of $X'$ such that $g$ on $Y$ is equivalent to $f$ on $X$.

Proof:

Let the alphabet of $(T', F, f)$ be the alphabet of $(T, G, g)$. For internal states, we shall take all the internal states of $(T, G, g)$, together with a copy $Lq$ of every left-facing $q$, and a copy $Rq$ of every right-facing $q$. Let $X = Y$. $X$ is the set of instantaneous descriptions $A'$ such that for all $n, A'(n) \not\in Rq$ and $A'(n) \not\in Lq$.

The 1-1 correspondence under which $g$ and $f$ are equivalent is defined as follows:

If $A \epsilon Y$ and $A(N) \epsilon QR$, let $A$ correspond to $A$

If $A(N) \epsilon QL$, let $A$ correspond to $A'$ where

$A'(n) = A(N)$

$A'(N) = A(N-1)$.

Define $T'$:

<table>
<thead>
<tr>
<th>Definition</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. $q_1 \epsilon QR$</td>
<td></td>
</tr>
<tr>
<td>i) $q_2 \epsilon QR$</td>
<td></td>
</tr>
</tbody>
</table>
| 1. $T(q_1, s_3) = sPq_2$ | $s\ s_2q_1s_3s_4 \quad \ldots \quad s_1s_2q_2s_3s_4$
| Let $T'(q_1, s_3) = sPq_2$ | $s\ s_2q_1s_3s_4 \quad \ldots \quad s_1s_2q_2s_3s_4$
2. \( T(q_1, s_3) = sRq_2 \)
Let \( T'(q_1, s_3) = sRq_2 \)

3. \( T(q_1, s_3) = SLq_2 \)
Let \( T'(q_1, s_3) = sLq_2 \)

ii) \( q_2 \in QL \)

1. \( T(q_1, s_3) = sPq_2 \)
Let \( T'(q_1, s_3) = sLq_2 \)

2. \( T(q_1, s_3) = sRq_2 \)
Let \( T'(q_1, s_3) = sPq_2 \)

3. \( T(q_1, s_3) = SLq_2 \)
Let \( T'(q_1, s_3) = sL(Lq_2) \)

\( T'(Lq_2, s_1) = sLq_2 \)

II. \( q_1 \in QL \)

i) \( q_2 \in QR \)

1. \( T(q_1, s_3) = sPq_2 \)
Let \( T'(q_1, s_3) = sRq_2 \)

2. \( T(q_1, s_2) = sRq_2 \)
Let \( T'(q_1, s_3) = sR(Rq_2) \)

\( T'(Rq_2, s_1) = sRq_2 \)
3. \( T(q_1, s_3) = sLq_2 \)  
\[ s_1^2 s_3^2 s_2^2 s_3 q_1 s_4 \]  
Let \( T'(q_1, s_3) = sPq_2 \)  
\[ s_1^2 s_2^2 s_3^2 q_1 s_4 \]

ii) \( q_2 \in QL \)

1. \( T(q_1, s_3) = sPq_2 \)  
\[ s_1^2 s_2^2 s_3^2 q_1 s_4 \]  
Let \( T'(q_1, s_3) = sPq_2 \)  
\[ s_1^2 s_2^2 s_3^2 q_1 s_4 \]

2. \( T(q_1, s_3) = sRq_2 \)  
\[ s_1^2 s_2^2 s_3^2 q_1 s_4 \]  
Let \( T'(q_1, s_3) = sRq_2 \)  
\[ s_1^2 s_2^2 s_3^2 q_1 s_4 \]

3. \( T(q_1, s_3) = sLq_2 \)  
\[ s_1^2 s_2^2 s_3^2 q_1 s_4 \]  
Let \( T'(q_1, s_3) = sLq_2 \)  
\[ s_1^2 s_2^2 s_3^2 q_1 s_4 \]

Let \( T'(q_1, s_3) \) be undefined if \( T(q_1, s_3) \) is undefined.

It is clear from the construction of \( T' \) that \( f \) on \( X \) is equivalent to \( g \) on \( Y \).

QED
BIBLIOGRAPHY


APPENDIX
Let $T'$, the Turing table referred to on p. 10, be defined as follows:

<table>
<thead>
<tr>
<th>State</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_0 B$ s</td>
<td>$s_2 L(q_1 B)$</td>
</tr>
<tr>
<td>$q_1 B$ x</td>
<td>$x L(q_1 B)$</td>
</tr>
<tr>
<td>$q_1 B$ q</td>
<td>$q_2 L(q_1 B)$</td>
</tr>
<tr>
<td>$q_1 B$ h</td>
<td>$q_1 R(q_2 B)$</td>
</tr>
<tr>
<td>$q_2 B$ x</td>
<td>$x R(q_2 B)$</td>
</tr>
<tr>
<td>$q_2 B$ h</td>
<td>$h L(q_3 B)$</td>
</tr>
<tr>
<td>$q_3 B$ s</td>
<td>$s_3 L(q_3 B)$</td>
</tr>
<tr>
<td>$q_3 B$ s</td>
<td>$s_4 L(q_3 B)$</td>
</tr>
<tr>
<td>$q_3 B$ s</td>
<td>$s_3 L(q_3 s_1)$</td>
</tr>
<tr>
<td>$q_3 B$ s</td>
<td>$s_4 L(q_3 s_2)$</td>
</tr>
</tbody>
</table>

For the next eight entries, $i = 1, 2$.

<table>
<thead>
<tr>
<th>State</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_3 s_i$ x</td>
<td>$x L(q_3 s_i)$</td>
</tr>
<tr>
<td>$q_3 s_i$ q</td>
<td>$q_2 L(q_4 s_i)$</td>
</tr>
<tr>
<td>$q_4 s_i$ s</td>
<td>$s_j L(q_4 s_i)$</td>
</tr>
</tbody>
</table>
\[(q_{p4} s_i) \quad s_j \quad \text{Find symbol for comparison.} \]
\[(q_{p4} s_i) \quad s_1 \quad s_{i+2} L(q_{p3} s_i) \quad \text{Comparison held. Go to next stored tape.} \]
\[(q_{p5} s_i) \quad s_k \quad s_j \quad \text{Comparison failed.} \]
\[j = 1, 2 \text{ and } s_k \neq s_i \]
\[(q_{p5} s_i) \quad s_3 \quad s_1 R(q_{p5} s_i) \quad \text{Erase marks indicating agreement.} \]
\[(q_{p5} s_i) \quad s_4 \quad s_2 R(q_{p5} s_i) \]
\[(q_{p5} s_i) \quad q_2 \quad q_1 L(q_{p3} s_i) \quad \text{Go to next stored tape.} \]
\[(q_{p3} s_i) \quad B \quad s_i R(q_{p2} B) \quad \text{Print } s_i. \text{ Return to pick up next symbol.} \]
\[(q_{p3} B) \quad q_i \quad q_i P(q_{p3} h) \quad \text{All of the tape has been copied.} \]
\[(q_{p3} h) \quad x_1 \quad x = xL(q_{p3} h) \]
\[x \neq q_2, x \neq B. \quad \text{No earlier complete configurations agree with the one now on the looped machine. Print an end marker.} \]
\[(q_{p3} h) \quad B \quad hR(q_{p6} B) \]
\[(q_{p6} B) \quad q_1 \quad q_1 R(q_{p6} B) \quad \text{Go to upper end, erasing marks of agreement on the way.} \]
\[(q_{p6} B) \quad s_i \quad s_i R(q_{p6} B) \]
\[i = 1, 2 \]
\[(q_{p6} B) \quad s_i \quad s_{i-2} R(q_{p6} B) \]
\[i = 3, 4 \]
\[(q_{p6} B) \quad h \quad hL(q_{p7} B) \]
\[(q_{p7} B) \quad s_i \quad s_i L(q_{p7} B) \quad \text{Find square originally scanned.} \]
\[(qp_7 B) \ s_2 \quad \rightarrow \quad s_1 \ m(q_{j \ p_0 B}) \quad \rightarrow \quad T(q, s) = s_{i}m_{q_j} \]

Make move of looped machine. Repeat compare and copy cycle.

\[(qp_3 h) \ q_2 \quad \rightarrow \quad q_1 R(q_{p_{8 B}}) \]

There is an earlier complete configuration which agrees with the one now on the looped machine.

\[(qp_8 B) \ x \quad \rightarrow \quad xR(q_{p_{8 B}}) \]

Go to upper end marker. 

\[x \neq h \]

\[(qp_8 B) \ h \quad \rightarrow \quad BL(q_{p_{8 h}}) \]

Move upper end marker down to where it should be.

\[(qp_8 h) \ B_3 \quad \rightarrow \quad BL(q_{p_{8 h}}) \]

\[(qp_8 h) \ x \quad \rightarrow \quad xR(q_{p_{8 h}}) \]

\[x \neq B_3, \ x \neq B \]

\[(qp_8 h) \ B \quad \rightarrow \quad hL(q_{p_{9 B}}) \]

Print h.

\[(qp_9 B) \ s_{i_1} \quad \rightarrow \quad s_{i_2} L(q_{p_{9 B}}) \]

\[i = 3, 4 \]

Go to lower end, erasing marks of agreement on the way.

\[(qp_9 B) \ q_{i_1} \quad \rightarrow \quad hL(q_{p_{10 B}}) \]

Print h at lower end of tape on looped machine.

\[(qp_{10 B}) \ x \quad \rightarrow \quad BL(q_{p_{10 B}}) \]

\[x \neq B, \ x \neq h \]

Erase extraneous material.

\[(qp_{10 B}) \ B \quad \rightarrow \quad BR(q_{p_{10 B}}) \]

\[(qp_{10 B}) \ h \quad \rightarrow \quad BR(q_{p_{10 h}}) \]

Move lower end marker up to where it should be.

\[(qp_{10 h}) \ B_1 \quad \rightarrow \quad BR(q_{p_{10 h}}) \]
\((qp_{10}^h) \times \) --- \(xL(qp_{11}^h)\)
\(x \neq B_1\)

\((qp_{11}^h) B \) --- \(hR(qp_{11}B)\)

\((qp_{11} B) s_1 \) --- \(s_1R(qp_{11}B)\) Locate square originally scanned.

\((qp_{12} B) s_2 \) --- \(s_1P(qp_{12}B)\)

\((qp_{12} B) s_1 \) --- \(s_1P(qp_{12}B)\) Stop.