INFORMATION AND CONTROL 10, 22–42 (1967)

An 8-State Minimal Time Solution to the Firing Squad Synchronization Problem*†

ROBERT BALZER

Systems and Communication Sciences Program, Carnegie Institute of Technology, Pittsburgh, Pennsylvania

This paper presents a description of a general outline for a minimal time solution to the Firing Squad Synchronization Problem, and a solution of this form which is composed of machines with only eight states. The paper then discusses the verification of this minimal time solution by computer simulation, and gives the schema for the mathematical induction proof that the solution works for any length (the full proof is given in (Balzer, 1966)).

The final part of the paper discusses some efforts to determine the minimal number of states needed for a minimal time solution. No four state minimal time solution exists. A reasonable set of conditions are presented for which no five state minimal time solution exists. Also, various eight state minimal time solutions exists.

THE FIRING SQUAD SYNCHRONIZATION PROBLEM

The problem with which the paper is concerned was first publicly presented by Dr. E. F. Moore in 1962 (Moore, 1964), from which we quote:

"The problem known as the Firing Squad Synchronization Problem was devised about the year 1957 by John Myhill, but so far as I know the statement of the problem has not yet appeared in print. It has been widely circulated by word of mouth, and has attracted sufficient interest that it ought to be available in print. The problem first arose in connec-

* This work was supported by a National Science Foundation Graduate Fellowship and in part by the Advanced Research Projects Agency of the Office of the Secretary of Defense (contract SD-146).

† A preliminary version of this paper (without the discussion of the proof) was circulated on November 8, 1965. It is based on a Ph.D. thesis submitted to Carnegie Institute of Technology.

tion with causing all parts of a self-producing machine to be turned on simultaneously. The problem was first solved by John McCarthy and Marvin Minsky, and now that it is known to have a solution, even persons with no background in logical design or computer programming can usually find a solution in a time of two to four hours. The problem has an unusual elegance in that it is directly analogous to problems of logical design, systems design, or programming, but it does not depend on the properties of any particular set of logical elements or the instructions of any particular computer. I would urge those who know a solution to this problem to avoid divulging it to those who are figuring it out for themselves, since this will spoil the fun of this intriguing problem.

"Consider a finite (but arbitrarily long) one dimensional array of finite-state machines, all of which are alike except the ones at each end. The machines are called soldiers, and one of the end machines is called a general. The machines are synchronous, and the state of each machine at time $t + 1$ depends on the states of itself and of its two neighbors at time t. The problem is to specify the states and transitions of the soldiers in such a way that the general can cause them to go into one particular terminal state (i.e., they fire their guns) all at exactly the same time. At the beginning (i.e., $t = 0$) all the soldiers are assumed to be in a single state, the quiescent state. When the general undergoes the transition into the state labeled "fire when ready", he does not take any initiative afterwards, and the rest is up to the soldiers. The signal can propagate down the line no faster than one soldier per unit of time, and their problem is how to get all coordinated and in rhythm. The tricky part of the problem is that the same kind of soldier with a fixed number, k, of states, is required to be able to do this, regardless of the length, n, of the firing squad. In particular, the soldier with k states should work correctly, even when n is much larger than k. Roughly speaking, none of the soldiers is permitted to count as high as n.

"Two of the soldiers, the general and the soldier farthest from the general, are allowed to be slightly different from the other soldiers in being able to act without having soldiers on both sides of them, but their structure must also be independent of n.

"A convenient way of indicating a solution of this problem is to use a piece of graph paper, with the horizontal coordinate representing the spatial position, and the vertical coordinate representing time. Within the (i, j) square of the graph paper a symbol may be written, indicating

the state of the ith soldier at time j. Visual examination of the pattern of propagation of these symbols can indicate what kinds of signaling must take place between the soldiers.

"Any solution to the Firing Squad Synchronization Problem can easily be shown to require that the time from the general's order until the guns go off must be at least $2n - 2$, where n is the number of soldiers. Most persons solve this problem in a way which requires between $3n$ and $8n$ units of time, although occasionally other solutions are found. Some such other solutions require $5/2n$ and of the order of n-squared units of time. For instance, until recently, it was not known what the smallest possible time for a solution was. However, this was solved at M.I.T. by Professor E. Goto[1] of the University of Tokyo. The solution obtained by Goto used a very ingenious construction, with each soldier having many thousands of states, and the solution required exactly $2n - 2$ units of time. In view of the difficulty of obtaining this solution, a much more interesting problem for beginners is to try to obtain some solution between $3n$ and $8n$ units of time, which as remarked above, is relatively easy to do."

Goto's solution has apparently not been published. However, independently of the present effort, Abraham Waksman (1966) has found a 16-state minimal time solution using essentially the same ideas as presented in the next section. Fischer (1965) has also used these same ideas in discussing other properties of one-dimensional iterative arrays of finite-state machines.

GENERAL DESCRIPTION OF THE MINIMAL TIME SOLUTION

The Firing Squad Synchronization Problem can be solved by successively subdividing the line into halves, quarters, eights, etc. until all members of the line are division points. At this time they all can fire simultaneously. By always dividing the line into two equal parts, and then subdividing each of those parts into two equal parts, and so on, the synchronization of the firing can be assured.

To divide the line into two equal parts, the general simultaneously sends out two signals, $S1$ and $S2$. For the sake of definiteness, we will assume the general is the rightmost man in the line. $S1$ and $S2$, then,

[1] Eiichi Goto, "A Minimal Time Solution of the Firing Squad Problem," Dittoed course notes for Applied Mathematics 298, Harvard University (May 1962), pp. 52–59, with an illustration in color. Also a different version of Goto's solution is to be published, without the colored illustration.

both travel to the left, $S1$ at a speed of one machine per time unit, and $S2$ at a speed of one machine every three time units. When $S1$ reaches the far end of the line, the end machine sends back a signal, $S3$, which travels at a speed of one machine per time unit to the right. Signals $S2$ and $S3$ will meet at the center of the line, for, if P is the length of the line, then $S1$ and $S3$ combined have traveled a distance of $3P/2$ while $S2$ has traveled a distance of $P/2$, a ratio of three to one, which is the same as the ratio of their respective speeds. Since $S2$ moves to the left only once every three units, the machine containing this signal must count to three (i.e., $S2$-1, $S2$-2, $S2$-3). By the state of the machine containing $S2$ and $S3$ encountered, it can be determined whether the line is of even or odd length, and hence, whether both machines should become middle men or just the one containing $S2$. These middle men (or man) then act like the original general, sending out both $S1$ and $S2$ signals to the left and also to the right. This process is repeated over and over until all the men in the line are middle men, at which time firing occurs.

Notice that the above process insures the synchronization of the line and also permits the determination of the firing condition on a local basis, that is, a machine fires if it is a middle man and the machines on either side of it are also middle men. The outside machines fire if they are middle men and the machine next to it is also a middle man.

The process described above will lead to a three N solution (where N is the length of the line). The first middle point is found in $3N/2$, the quarter points are found in $3(N/2)/2$ additional units of time, and so on. This summation leads to a $3N$ solution.

The method described above can be modified to yield a $2N$ solution. Assuming again that the general is the rightmost machine, we change the above process as follows: When $S1$ reaches the left end of the line, the leftmost machine acts as if it were also a general. That is, it sends out two signals to the right, one signal, $S3$, has been discussed above, the second signal, $S4$, is like $S2$, except that it travels to the right instead of the left. The middle of the line is still found when $S3$ and $S2$ meet. Now when the middle man (or men) created sends out signals $S1$ and $S2$ to the left, as described above, $S1$ will meet $S4$ at the quarter point of the original line. As above, from the state of the machine containing the slower moving signal, it can be determined whether the length of the left half of the original line is even or odd, and hence, whether there should be two or one middle man. Notice that the mechanism

used to find this quarter point is the same as that used in the $3N$ solution. However, the process was started earlier, when the end of the line was reached, and originated at the opposite end of the left half of the line. This process can be continued to find the right quarter point of the left half of the line. The length of time necessary to find the quarter point after the middle point has been found is just the distance between these points, $N/4$. The length of time necessary to find the eighth point after the quarter point has been found is $N/8$, and so on. If this rate of finding successive middle points could be maintained, then the total time for a solution would be $3N/2 + N/4 + N/8 + N/16 + \cdots$ which equals $2N$. However, since the general is counted as one member of the line the remaining line is of length $N - 1$. Hence this solution will take $2(N - 1)$. Therefore, if the above rate of finding successive middle points could be maintained, a minimal time solution would be attained.

The above process does not produce a solution because, in any interval, it will only find one of the two quarter points in the required length of time. Again assuming the general is the rightmost man in the line, the above will only find the left quarter point, but not the right. Thus the line will not be synchronized and a solution will not be found. This can be rectified by having the general also send out a signal, $S5$, along with $S1$, and $S2$, which travels to the left at a rate of one machine every 7 units of time. This signal and $S3$ sent from the middle of the line will meet at the quarter point. Each middle man now sends out three signals enabling every interval to be divided into quarters in synchronization. However, the rightmost eighth point cannot be found as fast as the other eighth points (this is the same problem as above with the rightmost quarter point). As above, we can find this point by having the general also send out another signal traveling at a rate of one machine every 15 units of time. In a similar way, the rightmost $1/(2 \uparrow K)$th point in an interval can be found by having the general send out a signal traveling at a rate of one machine every $2 \uparrow (K + 1) - 1$ units of time.

Sending out enough of these signals would produce a minimal time solution for any given length of the firing squad. However, the number of signals required is dependent on the length of the line. Since the number of states required for a machine is dependent on the number of signals that machine must handle, the number of states required is dependent on the length of the line, and hence, the above does not constitute a solution to the Firing Squad Synchronization Problem.

The above process fails because it cannot find the rightmost (or left-

most if the general starts on the left) $1/(2 \uparrow K)$th points in an interval, with a given number of states for all lengths. If one could have a marker which reached the right quarter point and then stopped there, and a marker which reached the right eighth point and then stopped there, and so on, the problem would be resolved. As the right traveling signal encountered these markers it would create middle men, and then the process would begin to subdivide the next interval in the same manner. The total distance traveled by each of these markers would be half the distance traveled by the marker to its left.

Consider the following: A signal travels to the left at a rate of one machine every unit of time. Every second time it moves, it sends out a signal to the right which travels at a rate of one machine every unit of time. When this signal reaches a marker, a machine in a given state, the marker moves one machine to the left. Every second time this marker has moved, it sends a similar signal to the right which causes the next marker to move. Each marker behaves in this manner, moving to the left when it receives a signal from its left, and every second time it moves, it sends a signal to its right for the next marker. Finally, the general acts as a source of these markers. Every time a signal is received, it produces a marker on its left.

We now have a system in which each marker travels half as far as the marker in front of it, and no matter how long the line is, enough of these markers are produced to properly subdivide the line.

When the original signal reaches the end of the line, the same process described above is repeated except that all directions are now reversed. When the original signal reaches the far end of the line, the first marker has not yet reached the middle of the line because all the signals causing it to move have not been received. No new signals causing movement of the markers will be produced, and so the markers will ultimately stop at the correct positions in the line. Since the movement causing signals travel at the same rate as the original signal, none of these signals can be overtaken by the original signal from the far end of the line. Hence, when this signal reaches the marker, all the signals causing that marker to move will already have been received, and therefore, the marker will be in the correct position. In any interval, two separate processes are used to find the quarter points. The quarter point farthest from the middle man who initiated the subdivision of the interval is found by the process first described in the $3N$ solution. The quarter point nearest this middle man is found by the process just described. The length of

time required to find these quarter points after the middle of the interval has been found is equal to the number of machines between the quarter points and the middle point for both processes. Hence, the quarter points will be found simultaneously, the solution will remain synchronized, and this process will produce a minimal time solution (since there is an unlimited source of markers, the problem can be solved with a fixed number of states no matter how long the line is).

DESCRIPTION OF THE ACTUAL SOLUTION

In the solution, presented in Appendix III, each of the states performs some of the functions described in the above process. The initial quiescent state is L, and the firing state is F. The state M acts both as the general's initial state and as the middle man. The state C acts both as the original left traveling signal and as the left traveling markers. The two states B and R combine to form the right traveling signals which cause the markers to move to the left. Both B and R travel to the right. When the triplet BRC occurs the marker moves one position to the left. Each time a marker moves its old position is replaced by the other member of the pair B-R from the state of the machine to the right of the old position of the marker. That is, if the state of a part of the line is $BRCR$ this will become $BCCR$ which will become $BCBR$. Likewise $BRCB$ will become $BCCB$ which will become $BCRB$. This mechanism causes the pair B-R to travel to the right every second time the marker moves. Since the process involved in the solution must proceed in both directions, corresponding to state C there exists a state Q, corresponding to state B there exists a state A, and corresponding to state R there exists a state L. There does not have to be a state corresponding to state M because it exhibits no directional properties. These seven states and the terminal firing state (F) represent all the states necessary for a minimal time solution to the Firing Squad Synchronization Problem. When a C state and a Q state occur in adjacent machines, either one or both machines become middle men. This decision is made in the following manner. As the leading marker moves down the line the state of the machine behind this marker alternates between the states R and B. When the end of the line is reached, if the state of the machine to the right of the end machine is a B then the line is of odd length and only one middle point should be produced. If the state of this machine is R then the line is of even length and two middle points should be produced. If the line is of odd length, a state R is generated to the left of the leading marker Q.

This pair of states travels to the right until a C marker is encountered. The state of this portion of the line will be either $RQCB$, $RQCR$, or $RQCC$. Each of these configurations will produce something of the form $YYMY$, where the Y's are not M's. This mechanism of producing a state where it would not normally be, enables the proper choice of one or two middle points to be made. The configurations $WQCC$, $WQCR$, or $WQCB$, where the W's are either A, L, or Q, will produce a configuration of the form $YMMY$, where the Y's are not M's.

The above mechanism of determining whether one or two middle points should be produced and other mechanisms in the solution (taking care of special cases) are required because there are so few states for each machine. A solution involving more states could be produced which used only the mechanisms described in the general minimal time solution presented earlier.

Normally, finite state machines are defined by the standard state tables, but it is easier here to define a machine by a set of rules called productions. These rules are of the form

$$U, V, W \rightarrow Y$$

with or without the commas, we interpret this rule as:

> If a machine is in state V and the machine on its left is in state U and the machine on its right is in state W, then the machine's new state is Y. U, V, W and Y are respectively called the Left Man, Middle Man, Right Man, and Resultant of the production.

Such a definition of a finite state machine is, of course, equivalent to a state table. However, this definition allows easy reference to single rules or classes of rules (such as all rules whose Left Man is U and whose Middle Man is M). It closely matches the appearance of our two dimensional representation of the firing squad, and it is easily represented inside a computer. Appendix II gives a state table definition for those who are more familiar with this representation.

SIMULATION OF THE FIRING SQUAD

In order to verify the validity of the solution to the Firing Squad Synchronization Problem, and to obtain the solution for a large number of different lengths, a computer program was devised to simulate the firing squad for any length. This program was written in the computer language ALGOL-20, as implemented on the G-20 computer at Carnegie

Institute of Technology (Fierst *et al.*, 1965) (the main difference between this implementation and ALGOL 60, in reference to the simulation program under discussion, is that in assignment statements := is replaced by ←). The data for this program is a one dimensional array which represents the definition of the machines in the firing squad. The elements of this array are accessed as follows: each of the states represented externally by a letter is represented internally by a digit, defined by the following table:

External representation	Internal representation
L	0
M	1
B	2
C	3
Q	4
R	5
A	6
X	7
F	8

To mark the two ends of the firing squad, we place a machine on each end which remans in state X. Thus, the left end machine of the firing squad always has a machine on its left in state X, and similarly, the right end machine of the firing squad always has a machine in state X on its right. Now we can define all the machines in the firing squad by a single set of productions. If it is desired to find the new state of a machine in state C which has a machine in state L on its left and a machine in state B on its right, the number represented by the three digits UVW, where

U = The internal representation of the state of the machine on the left

V = The internal representation of the state of the machine in the middle

W = The internal representation of the state of the machine on the right

is formed, and the value of the array element with this subscript is the internal representation of the new state. Thus in the above example a machine in state C with machines in state L on its left and B on its right would go into the state represented by the 032-nd array element. Looking up this value in the array in the program we find

PRODUCTION [032] ← 5

Hence, the new state of the machine in question is state R (the syntax of ALGOL-20 requires us to use a left arrow for the productions instead of a right arrow).

Initially, all the elements of this array are set to value 9 which is interpreted by the program as being undefined. Then the values of the rules necessary to define the machines are set. The program then sets up an array, called Present, which represents the state of each of the machines in the firing squad. Initially, this array is set to all 0's (state L) with the rightmost position set to 1 (state M). A counter, time, is set to value zero. The program then finds the new state of each machine in the line by the process described above, storing these new states into an array called New. When all machines have been processed, the array New is copied into the array Present, the counter, Time, is incremented by one, and the values of this counter and the array Present are printed. This process continues until the value of the counter, Time, is $2N - 2$, where N is the length of the line. When this occurs, the line is checked to see if all the machines are in the firing state (state F). If they are, then the message "Minimal Time Solution Found for Length- - - -" with the blank filled in by the appropriate number, is printed. If all of the machines are not in the firing state, the message "Error in Solution for Firing Squad of Length- - - -" is printed. If during the process of finding the new states for the machines, an array element is accessed which has the value 9 (undefined), then processing stops and the message "Production - - -Not Found" with the dashes filled in by the appropriate states, is printed. Thus, since no rules are defined which have the Left Man, Middle Man, or Right Man equal to the firing state, if any machine goes into the firing state too soon, an undefined rule will be encountered and the error found.

This program has been run for all lengths inclusive between 2 and 100, and the solution has been correct on each of these runs. Appendix II contains the state table definition of these machines. Appendix III consists of the output of the simulation program for length 26. The program can be found in (Balzer, 1966).

PROOF OF THE SOLUTION

By use of the simulation program we have shown that the solution works for all lengths from 2 through 100 inclusive. This, however, gives us no assurance that this solution will work for all lengths. We, there-

fore, must present an inductive proof that our solution will, in fact, successfully solve the Firing Squad Synchronization Problem for all lengths. Our method is to use the simulation program to show that the solution works for all lengths up to and including some length which we will refer to as Z. We then assume the solution works for all lengths up to and including $N - 2$, where N is greater than Z, and show that it, therefore, works for length N.

This proof cannot be short because every one of the over two hundred productions must be referred to at least once, either explicitly or implicitly. Otherwise, such a production could be changed, causing the behavior of the iterative array to change, without altering the proof in any way.

We present a schema for this long proof (the entire proof can be found in (Balzer, 1966)) below:

Let $\downarrow(Y)$ represent the greatest integer in Y.

Let $P = \downarrow((N + 1)/2)$.

We have demonstrated by the simulation program that the solution works for all lengths less than or equal to Z.

We assume the solution works for all lengths less than or equal to $N - 2$, where N is greater than Z, and then show that the solution works for length N by the steps below.

1. We show the solution has the property that for every production

$$U, V, W \rightarrow Y$$

there exists a production

$$\text{Image}(W), \text{Image}(V), \text{Image}(U) \rightarrow \text{Image}(Y)$$

where the function Image maps the set of states onto the set of states, and Image(Image(E)) = E, for all E. We call solutions with this property image solutions. This property essentially requires that all processes occur in both directions, and involve image states.

2. We make the following definitions:

(A) An element is a machine in the iterative array at a point in time.

(B) A curve is a connected set of intervals in the two dimensional representation described in the problem statement. Each interval must be a portion of either a horizontal row, a vertical column, or two adjacent 45 degree or -45 degree diagonals.

(C) Two curves are said to be equal if both curves have the same shape and orientation, and if the elements along one curve are equal to the corresponding elements along the other curve.

(D) Similarly, two curves are said to be images of each other if both curves have the same shape and orientation except that the horizontal directions are switched (the curves are mirror images of each other), and the elements along one curve are equal to the image of the corresponding elements along the other curve.

3. Consider two concave downward curves which are equal or images of each other. We close these curves by adjoining to the end points of the curves 45 degree diagonals extending downward and toward each other. We then prove that all the elements within or on these closed curves are respectively equal to or the image of the corresponding elements in the other curve.

4. We prove that at time $\downarrow(3N/2) - 1$ the state M will appear at the center of the firing squad of length N if N is odd, and if N is even the state M will appear at the two positions which together represent the center of the line. We further prove that this machine or these machines remain in state M until and including time $2N - 3$ (call this machine staying in state M from time P until time $2N - 3$ boundary 1. This boundary is marked by 1's in Fig. 1).

5. We prove that the leftmost machine is in state M from time $N - 1$ until time $2N - 3$ (call this Boundary 2. This boundary is marked by 2's in Fig. 1. All the boundaries defined are marked by their number in Fig. 1).

6. We prove that the elements along the two adjacent diagonals extending downward to the right from the middle of the line of length N at time $\downarrow(3N/2) - 1$ (call this Boundary 3) are equal to the corresponding elements along the two adjacent diagonals extending downward to the right from the leftmost position of the line of length P at the time $P - 1$ (this is Boundary 4).

7. By 4, 5, and 6, the curve made up of Boundaries 1 and 3 is equal to the curve made up of Boundaries 2 and 4. Therefore, by 3, we know that the elements at time $2N - 3$ of one curve are equal to the corresponding elements of the other curve at time $2P - 3$.

8. But by assumption, since $P < N - 2$, length P has a minimal time solution. Hence, all elements of length P are in the firing state at time $2P - 2$. By inspection of the productions, this implies that all elements were in state M at time $2P - 3$. Hence all elements in the right half of the line of length N are in state M at time $2N - 3$.

9. Similarly, we prove that all elements along the two adjacent diagonals extending downward to the left from the middle of the line of length N at time $\downarrow(3N/2) - 1$ (this is Boundary 5) are the image of the

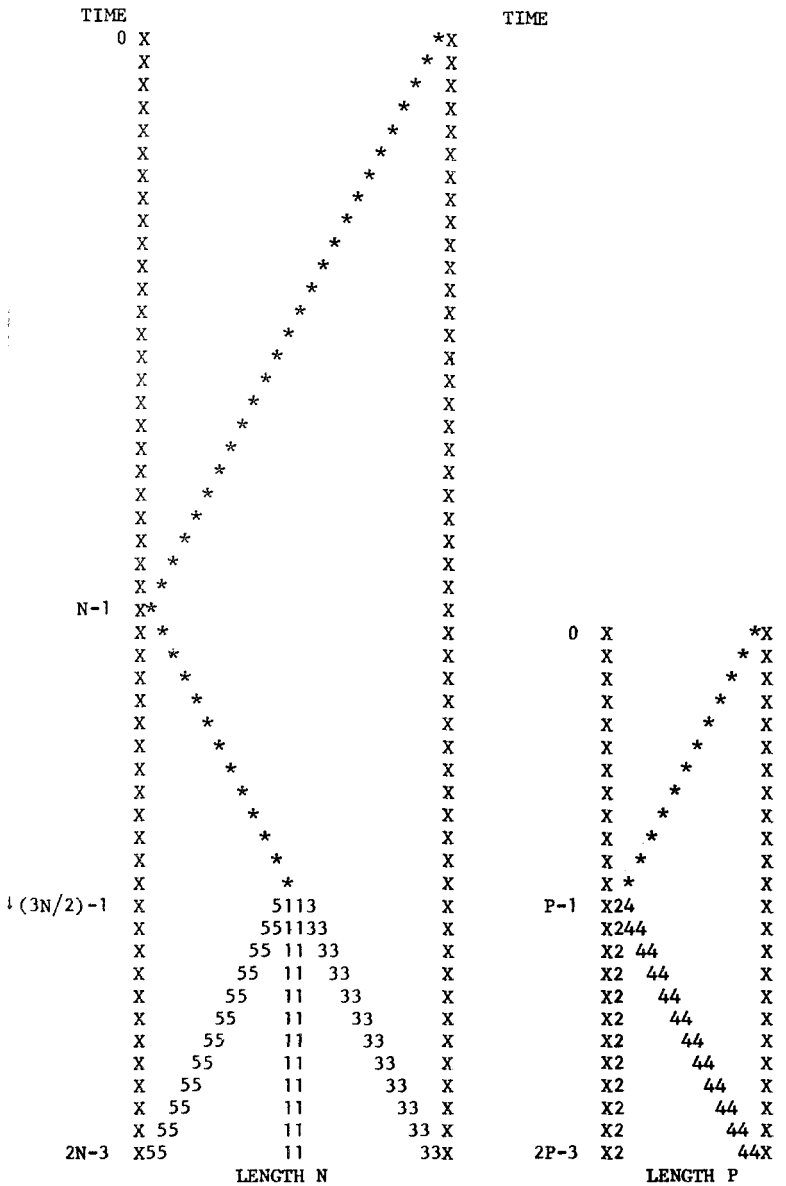

FIG. 1. Illustration of the schema of the proof for $N = 26$

corresponding elements along Boundary 4. Hence, since Image(M) = M, the curve made up of Boundaries 1 and 5 is the image of the curve made up of Boundaries 2 and 4. Therefore, all elements in the left half of the line of length N are in state M at time $2N - 3$.

10. By 8 and 9 above, all elements in the line of length N are in state M at time $2N - 3$. By inspection of the productions, all elements in the line are, therefore, in the firing state at time $2N - 2$.

11. Therefore, by induction, the solution works for all lengths, N, where $N > Z$. We have demonstrated the solution works for all lengths between 2 and Z inclusive. Hence the solution works for all lengths greater than or equal to 2.

OPTIMIZATION OF THE MINIMAL TIME SOLUTION

After having found an 8-state minimal time solution one naturally looks for a proof that there is no minimal time solution which has fewer than 8 states. Such a solution could be of two forms. The first would be an analytic type of proof, dealing with the number of states necessary to perform certain functions, and showing that these functions are necessary for a minimal time solution. At the present, we see no way to construct such an analytic proof. Therefore, we chose to try the second approach, proof by demonstration. That is, for all machines with less than 8 states we try to demonstrate that they do not provide a minimal time solution.

To prove that no minimal time solution exists with less than Y states, we must examine all machines with $Y - 1$ states (if no solution exists with $Y - 1$ states then no solution exists with $Y - 2$ states). We can calculate this number for machines with various numbers of states as follows:

Let the number of states in the desired solution be $N + 1$. The number of productions needed to completely define a machine with $N + 1$ states is

$N * N * N = N \uparrow 3$ (number of productions for the internal machines)

$+1 * N * N = N \uparrow 2$ (number of productions for the left-hand machine)

$+N * N * 1 = N \uparrow 2$ (number of productions for the right-hand machine)

Total number of productions $= N \uparrow 3 + 2 * N \uparrow 2$

Each of these productions can have any one of $N + 1$ Resultants, and so

number of machines with $N + 1$ states

$$= \text{number of sets of productions with } N + 1 \text{ states}$$
$$= (\text{number of Resultants}) \uparrow (\text{number of productions})$$
$$= (N + 1) \uparrow (N \uparrow 3 + 2 * N \uparrow 2)$$

which is tabulated below for machines with 3 through 8 states.

No. of States	No. of Productions	Approx. No. of Productions Sets
3	16	$4 * 10 \uparrow 7$
4	45	$1 * 10 \uparrow 27$
5	96	$1 * 10 \uparrow 67$
6	175	$1 * 10 \uparrow 136$
7	288	$1 * 10 \uparrow 244$
8	441	$1 * 10 \uparrow 400$

These numbers are much too large to allow us to examine each machine. We, therefore, need some rule which will generate only a very small part of the possible machines, and which will generate all machines that could be a solution, that is, a rule which eliminates large portions of the search space without examination and which does not eliminate any elements which could be solutions.

One such heuristic which gave positive results was to serially define the productions of the machines in the iterative array as they occurred. This process was built into a computer program which we call Program One, and which is described in Appendix I.

This program proved that no minimal time solution exists with 4 states. The number of possibilities it examined in this proof was approximately 60,000 in about 15 min (on the IBM 360 Model 40). The total number of possible solutions as calculated above is approximately $1 * 10 \uparrow 27$. From these two numbers we can get an estimate of the effectiveness of the heuristics employed.

This program took too much time when it attempted to prove that no five state minimal time solution exists, and was terminated after it had run for three hours and had examined approximately 570,000 possibilities. Thus the program examines about 200,000 possibilities an hour (or about 55 possibilities a second).

We then modified Program One so that it would search for a solution in a specified portion of the total solution space. We restricted the search

by restricting the Resultants of certain productions or sets of productions by use of a language which the program accepted as data. We devised this modification for two purposes. The first was to find reasonable conditions for which the program could prove that no five state minimal time solution existed. The second purpose was to try to find a seven state minimal time solution. Thus, we planned to zero in on the minimal state, minimal time solution from both directions, increasing the conditions for which we knew no minimal time solution existed, and decreasing the number of states necessary for a solution. The weakest conditions we found which enabled the program to prove that no five state minimal time solution existed were:

Let G = the state the general is initially set to

1. The solution is an Image Solution

2. G is a Resident State, that is, $U, G, V \rightarrow G$ for all U and V, where if U equals G or the end of the line, then V does not equal either G or the end of the line.

3. G is the Get-Ready-To-Fire state, that is, $U, G, V \rightarrow$ Firing State, where U and V are either equal to G or the end of the line. Furthermore, these are the only productions whose Resultant is equal to the Firing State.

4. If any machine has machines on both sides of it in the G state, then it goes into the G state, that is $G, V, G \rightarrow G$ where V is not equal to G.

These conditions were insufficient to prove no six state solution exists. We were unsuccessful in our attempts to find a seven state minimal time solution, although we believe such a solution exists, as we came very close to finding one (the Resultant of only one production had to have two different values).

We were, however, successful in finding eight state minimal time solutions. This program found five such solutions. We wanted to find a solution which had certain properties which would make the proof easier. These properties included the four conditions listed above. The program was able to find a solution which satisfied all of our conditions, and this solution is the one we presented in this paper, and the one for which the proof was obtained. The initial eight state solution found by the author by hand, satisfied only condition four above. Details on the program, the runs, and the class of constraints permitted by the language may be found in (Balzer, 1966).

APPENDIX I. DESCRIPTION OF PROGRAM ONE

Program One's basic heuristic is to serially define the productions as needed. It begins with all productions undefined. It then defines those productions required by the problem statement. It then initializes the firing squad for length two. Then it begins to find the new state of each man in the firing squad according to the productions which are already defined. If a production is encountered which has not yet been defined, then the Resultant of this production is defined to be equal to the firing state if, at this time, all machines must fire to get a minimal time solution. Otherwise, the Resultant is defined to be equal to the highest value for an allowed state. This process of defining the productions as they are needed, continues until an error occurs. An error is defined to be either a machine going into the firing state before time = 2 * present length − 2, or a machine not going into the firing state at time = 2 * present length − 2. When such an error occurs we find the most recently defined production whose Resultant is not equal to either the firing state or the lowest value for an allowed state (such a production is called alterable). We mark all productions which were defined after this production undefined. We redefine this production to be equal to its present Resultant − 1, and we go back to the state of the firing squad when this production was first defined. We now continue finding the new state of each man in the firing squad and defining productions as needed, as described above.

We continue the above process until either we have found a minimal time solution for all lengths up to the largest length we wish to consider, or we find no productions which are alterable, in which case, we have tried all possibilities which could lead to a solution for the number of states under consideration. Hence we have proved no minimal time solution exists for this number of states.

We can improve the efficiency of the above process with the following two heuristics. First, the last alterable production may have occurred in such a position in relation to the error found, that altering it cannot remove the error. Clearly, we are wasting our time altering such productions. We know that if no occurence of a production occurs in such a position that it could send a signal to the position where the error occurred before this occurrence, then the Resultant of this production is irrelevant in eliminating the error. We then say this production is not relevant to the error. Hence, we look for the last relevant alterable production, not merely the last alterable production.

The second heuristic is based on the fact that the search tree has a

certain isomorphic property. Initially, the firing squad contains only the quiescent state and the general's state. Since all the other states have not yet occurred, they are isomorphic to each other at this point. Hence, with our present search technique, rather than defining the Resultant of the next production to be defined to be equal to the highest value for an allowed state, we can define it to be equal to the minimum of all the isomorphic states. We have thus reduced the number of branches to be examined at this node in the search tree, but have not eliminated any solutions which are not isomorphic to ones contained in the reduced tree. This process is continued until none of the states are isomorphic. Naturally, because there is backup in our search, at some later time two states may again become isomorphic. At such times, this process will again be employed.

The program was written in basic assembly language for the IBM 360 Model 40, but lends itself to description in ALGOL. Such a description in pseudo-algol (there are no declaration of variables, and not all procedures are defined, but are assumed to be self-describing with their mnemonic names) can be found in (Balzer, 1966).

APPENDIX II. STATE TABLE

(Note that it is assumed that the ends of the line are marked by machines which remain in state *X*. This allows us to define all machines, including the end ones, together.)

Left input	Right input	Present state A	B	C	F	L	M	Q	R
A	A	A						Q	
A	B	C	Q						
A	C	C	Q	M		C		M	
A	X						M		
A	L	L		L		L	M	Q	
A	M	B	L	L		C	M		C
A	Q	A				L		L	L
A	R	L	Q					L	
B	A						M		
B	B		B	C					
B	C		B	C					C
B	X						M		
B	L								
B	M						M		
B	Q						M		
B	R		B	C					B

Left input	Right input	Present state A	B	C	F	L	M	Q	R
C	A						M		
C	B		B	R		R			R
C	C		B	R		R			R
C	X						M		
C	L								
C	M		C	C		C	M		
C	Q						M		
C	R		B	B		B			R
X	A						M		
X	B						M		
X	C					M	M		
X	X						F		
X	L					L			
X	M					M	F		
X	Q						M		
X	R						M		
L	A	A				A		Q	
L	B	C	R	R					
L	C	C	R	R		C		M	C
L	X						M		
L	L					L	M	Q	
L	M	B		C		C	M		B
L	Q	A				L		A	A
L	R			B		A	M	A	B
M	A						M		
M	B	R	A			Q	M	R	Q
M	C	Q	A			Q	M	M	Q
M	X						F		M
M	L							Q	
M	M			M		M	F	M	M
M	Q	Q					M	Q	Q
M	R		A			A	M	Q	Q
Q	A	A				Q		Q	
Q	B	C	Q	M				M	Q
Q	C	C	Q	M		C	M	M	Q
Q	X						M		M
Q	L	L		L		L	M	Q	
Q	M	B	C	M		C	M		C
Q	Q	A				L		L	L
Q	R	L	Q	M		Q		L	Q
R	A								
R	B		R	C			M	R	R
R	C		R	C			M	R	R
R	X								R

R	L			
R	M	C		
R	Q			
R	R	C	M	R

Appendix III. Output of Simulation Program for Length 26 Simulation of Firing Squad of Length 26

Minimal time solution found for firing squad of length 26

```
 0  LLLLLLLLLLLLLLLLLLLLLLLLLM
 1  LLLLLLLLLLLLLLLLLLLLLLLLCM
 2  LLLLLLLLLLLLLLLLLLLLLLLCCM
 3  LLLLLLLLLLLLLLLLLLLLLLCRCM
 4  LLLLLLLLLLLLLLLLLLLLLCBRCM
 5  LLLLLLLLLLLLLLLLLLLLCRBCCM
 6  LLLLLLLLLLLLLLLLLLLCBRRCCM
 7  LLLLLLLLLLLLLLLLLLCRBBRCCM
 8  LLLLLLLLLLLLLLLLLCBRRBCCCM
 9  LLLLLLLLLLLLLLLLCRBBRRCRCM
10  LLLLLLLLLLLLLLLCBRRBBRCRCM
11  LLLLLLLLLLLLLLCRBBRRBCCRCM
12  LLLLLLLLLLLLLCBRRBBRRCBRCM
13  LLLLLLLLLLLLCRBBRRBBRCBCCM
14  LLLLLLLLLLLCBRRBBRRBBCBCCM
15  LLLLLLLLLLCRBBRRBBRRBRBCCM
16  LLLLLLLLLCRRBBRRBBRRBRRCCM
17  LLLLLLLLCRBBRRBBRRBBCRRCCM
18  LLLLLLLCBRRBRRRBBRRCBRRCCM
19  LLLLLLCRBBRRBBRRBBRCBBRCCM
20  LLLLLCBRRBRRRBBRRBCCBBCCCM
21  LLLLCRBBRRBBRRBBRRCRBBCRCM
22  LLLCBRRBBRRBBRRBBRCRRBCRCM
23  LLCRBBRRBBRRBBRRBCCRRRCRCM
24  LCBRRBRRRBBRRBBRRCBRRRCRCM
25  MRBBRRBBRRBBRRBBRCBBRRCRCM
26  MQRBBRRBBRRBBRRBCCBBBRCRCM
27  MQQRBBRRBBRRBBRRCRBBBCCRCM
28  MQLQRBBRRBBRRBBRCRRBBCBRCM
29  MQLAQRBBRRBBRRBCCRRRBCBCCM
30  MQQALQRBBRRBBRRCBRRRRCBCCM
31  MQQLLAQRBBRRBBRCBBRRRCBCCM
32  MQQLAALQRBBRRBCCBBBRRCBCCM
33  MQQQALLAQRBBRRCRBBBBRCBCCM
34  MQLQLLAALQRBBRCRRBBBCCBCCM
35  MQLQLAALLAQRBCCRRRBBBCRBCCM
36  MQLQQALLAALQRCBRRRRBCRRCCM
37  MQLAQLLAALLAQCBBRRRRCRRCCM
38  MQQAQLAALLAAMMBBBRRRCRRCCM
39  MQQAQQALLAABMMABBBRRRCRRCCM
40  MQQALQLLAACLMMRQBBBRCRRCCM
41  MQQLLQLAACLCMMQRQBBCCRRCCM
42  MQQLLQQACLRCMMQLRQBCBRRCCM
43  MQQLLAQCLBCCMMQQARQCBBRCCM
44  MQQLAAMLRRCCMMQQLLRMBBCCCM
45  MQQQABMABRCCMMQQLABMABCRCM
46  MQLQCLMRQCCCMMQQQCLMRQCRCM
47  MQLMLCMQRMRCMMQLMLCMQRMRCM
48  MQCMQCMQCMQCMMQCMQCMQCMQCM
49  MMMMMMMMMMMMMMMMMMMMMMMMM
50  FFFFFFFFFFFFFFFFFFFFFFFFFF
```

ACKNOWLEDGMENT

I am indebted to Dr. Allen Newell for his suggestions and guidance.

RECEIVED: February 28, 1966

REFERENCES

FIERST, BLOCHER, BRADEN, EVANS, AND GROVE (1965), "ALGOL-20, A Language Manual." Carnegie Institute of Technology, First Printing February 1965.

MOORE, E. F. (1964), "Sequential Machines, Selected Papers," pp. 213–214. Addison Wesley, Reading, Mass.

WAKSMAN, ABRAHAM (1966), An optimal solution to the firing squad synchronization problem. *Inform. Control* **9**, 66–78.

BALZER, ROBERT (1966), "Studies Concerning Minimal Time Solutions to the Firing Squad Synchronization Problem," Doctoral Thesis (submitted), Carnegie Institute of Technology.

FISCHER, PATRICK C. (1965), Generation of primes by a one-dimensional real-time iterative array. *J. Assoc. Comput. Mach.* **12**, 388–394.