SEARCH FOR A SOLUTION: A CASE STUDY*

R. M. Balzer

The RAND Corporation, Santa Monica, California

ABSTRACT

This paper describes a series of attempts at the solution of a conceptually tough problem, the Firing Squad Synchronization Problem. These attempts demonstrate an increasing reliance on man-machine symbiosis and decreasing reliance on powerful heuristics and preplanning.

These attempts consist of a clerical checking program, and four attempts utilizing a basic backtracking program for searching the solution space. The first two attempts, Serial Definition of Productions and Symbolic Definition of Productions were non-interactive entirely computer directed attempts at solution. The second two, Functional Planning and Constraint Satisfaction were man-machine symbiotic attempts designed to allow the human to control and direct the computer search of the solution space. The benefits of these symbiotic attempts and the problems encountered with them are discussed.

INTRODUCTION

This is the second of two papers detailing my personal experiences with the Firing Squad Synchronization Problem over the last four years. During this period, I have tried many different approaches toward solving the problem. The first sequence led to the original best (smallest number of states) Minimal Time Solution yet found; the remaining attempts have been unsuccessful at either improving the solution (reducing the number of states) or proving it is the best possible solution, although they have generated several other Minimal Time solutions with this same number of states, and from other than an intellectual and illustrative standpoint, were rather unproductive. The first of these papers dealt with the succession of representations and the key insights they allowed that culminated in the Minimal Time Solution. This paper discusses the unsuccessful attempts from the standpoint of Man-Machine Symbiosis. Reviewed chronologically, these attempts demonstrate a continuing effort to substitute problem-oriented heuristics and interaction for brute force and preplanning.

PROBLEM STATEMENT

The problem with which this paper is concerned was first publicly presented by Dr. E. F. Moore in 1962:

"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 connection 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

This research is supported by the Advanced Research Projects Agency under Contract No. DAHC15 67 C 0141. Views or conclusions contained in this study should not be interpreted as representing the official opinion or policy of ARPA.

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.

lEiichi 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."*

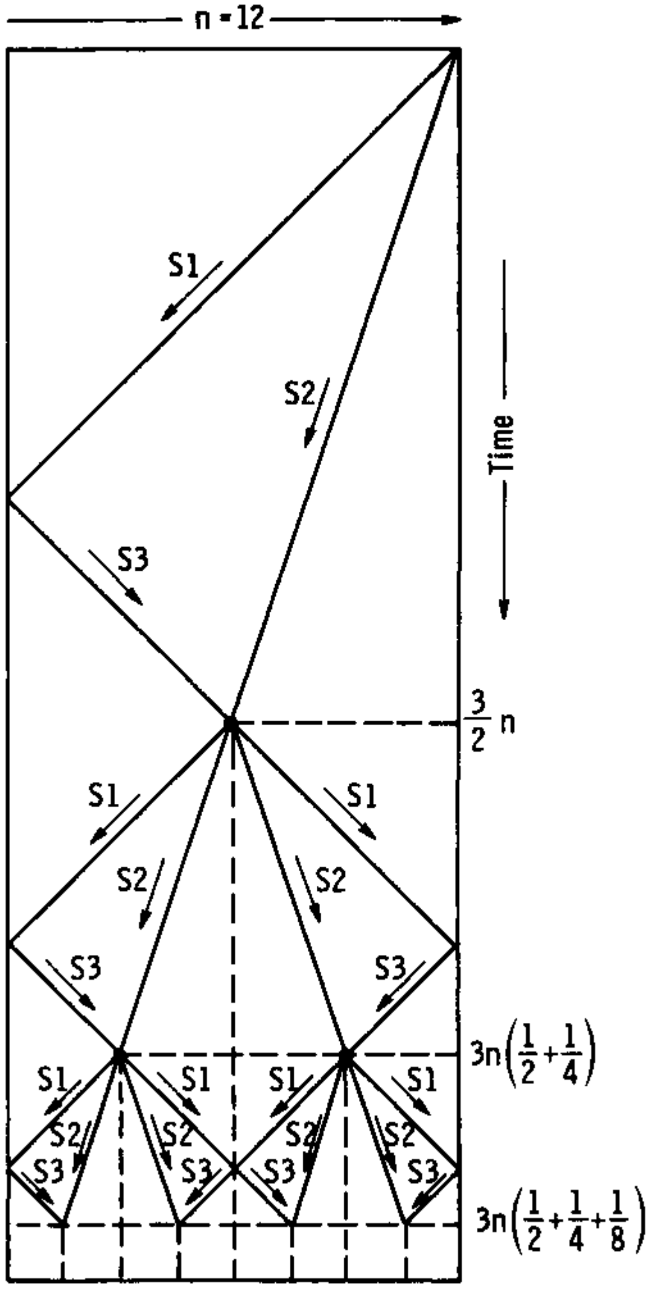
Goto's solution apparently has not been published. However, independently of the present effort, Abraham Waksman[1] has found a 16-state minimal time solution using essentially the same ideas as presented in the following section. Fischer2 has also used these ideas in discussing other properties of one-dimensional iterative arrays of finite-state machines.

GENERAL OUTLINE OF A MINIMAL TIME SOLUTION

The Firing Squad Synchronization Problem can be solved by successively subdividing the line into halves, quarters, eighths, 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, Sl and S2 (see Fig. 1).

*Moore, E. F. (1964), "Sequential Machines, Selected Papers," pp. 213-214. Addison Wesley, Reading, Mass.



n = 12
Time
S1
S2
S3
$\frac{3}{2}n$
$3n\left(\frac{1}{2}+\frac{1}{4}\right)$
$3n\left(\frac{1}{2}+\frac{1}{4}+\frac{1}{8}\right)$

Fig. 1—A 3n-3 solution for a firing squad of length 12

For the sake of definiteness, we will assume the general is the rightmost man in the line. SI and S2, then, both travel to the left, S1 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 only 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, i.e., 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 above process 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 above method can be modified to yield a 2N solution (see Fig. 2). Assuming again that the general is the rightmost machine, we change the process as follows: When S1 reaches the left end of the line, the leftmost machine acts as if it were also a general; i.e., it sends out two signals to the right, one signal, S3, has been discussed a'oove. 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. From the state of the machine containing the slower moving signal, it can be determined (as above) 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, the total time for a solution would be 3N/2 -l- N/4 + N/8 + N/16 ..., 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 find only 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 find only 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 SI, 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

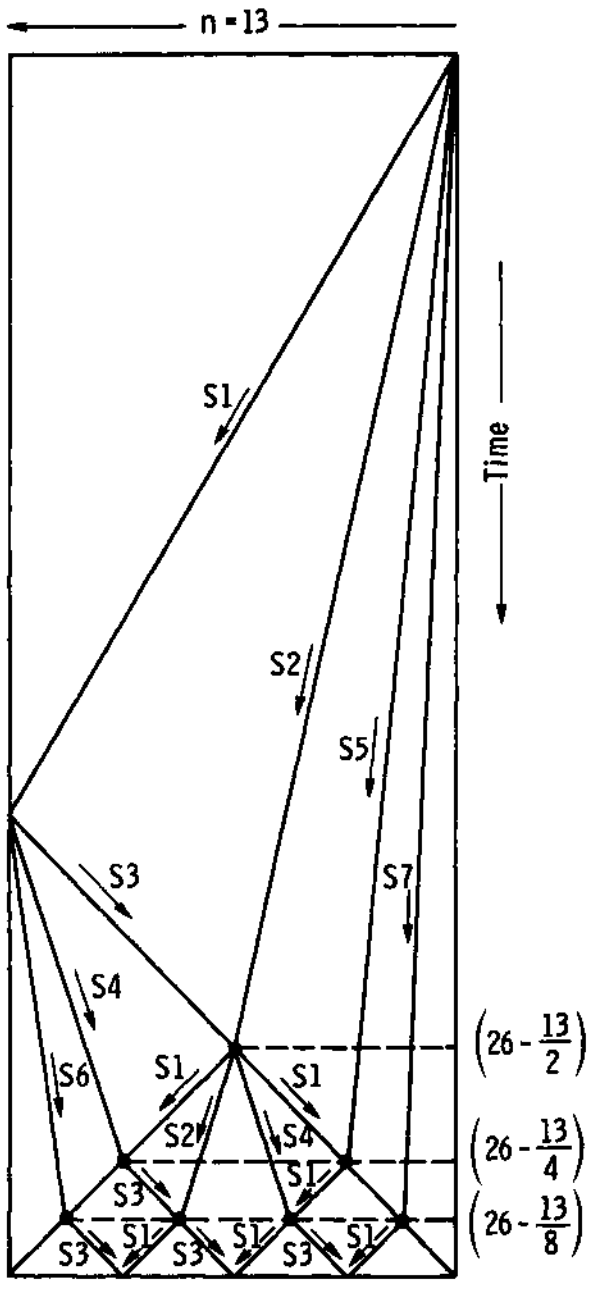

n = 13
Time
S1
S2
S5
S3
S7
S4
S6
S1
S1
S2
S4
S1
S3
S3
S1
S3
S1
S3
S1
$\left(26 - \frac{13}{2}\right)$
$\left(26 - \frac{13}{4}\right)$
$\left(26 - \frac{13}{8}\right)$

Fig. 2—A 2n-2 solution for a firing squad of length 13

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 l/(2l)th point in an interval can be found by having the general send out a signal traveling at a rate of one machine every 2t(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; 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 leftmost if the general starts on the left) l/(2tK)th points in an interval, with a given number of states for all lengths. If one could have a marker that 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 that 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 that 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, sending a signal to its right for the next marker. Finally, the general acts as a source of these markers, producing one on its left every time a signal is received.

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; 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).

SOLUTION ATTEMPTS

The use of the computer as an aid in solving the Firing Squad Problem occurred quite early, but only after the conceptual framework described above had been established. The first use of the computer was strictly clerical in accepting a definition of the soldiers as a set of productions, and producing a simulation of their behavior for a number of different length lines, noting any errors in this behavior. The interaction consisted of scanning the output, deciding on changes, making them and restarting the process (response time «8-10 hours). Using this mechanism alone, I was able to obtain a minimal time twenty-eight state solution and reduce it to an eight state minimal time solution. At this point I began, with my trusty computer, searching for a better solution or a proof that no better solution existed. While I did not succeed in this search, the attempts provide an interesting collage of some ways a man-machine partnership can be used to attack a conceptually tough problem.

SERIAL DEFINITION OF PRODUCTIONS

It was obvious from the first that brute force could not be used to search the solution space exhaustively (there are 1*10t244 possible seven-state solutions); so the simplest heuristic above brute force was employed, i.e., Don't make any decisions until you must! In the present context this means:

1) Start simulation with all productions undefined.
2) If an undefined production is encountered, define it to the first state and continue the simulation.
3) If an error is found, back the simulation up to the definition point of the most recent production, change its definition to the next state, and continue the simulation.
4) If there are no more states to try in #3 above, move back to the previous production. If there are no more productions quit.

This process will exhaust the possible search space, finding any solutions which exist or proving that none exist. This is the familiar technique of Backtrack Programming.3 A second heuristic utilizing isomorphism among unused states further reduced the search space by approximately two orders of magnitude. There was no interaction with this attempt; the computer carried the ball by itself, fumbling because the search space was still too large for seven-state solutions. It did, however, prove that no four-state minimal time solution exists.

SYMBOLIC DEFINITION OF PRODUCTIONS

Having failed via Backtrack Programming to find that a better solution existed or to prove that it didn't, and noting from more than adequate computer output that this technique spent most (would you believe ALL?) of its time looking down implausible trails, I decided to escalate one more level. Instead of holding decisions off until needed, hold them off even longer, making pseudo (symbolic) decisions and continuing. Ultimately, each pseudo (symbolic) decision in turn is converted to a real decision in a Backtrack Programming algorithm and the results of such decision can be observed. We thus have provided a means of look-ahead. The results of a decision can be observed at the time of that decision, and so, I thought, large portions of the search space could be pruned earlier.

This method started a simulation of the Firing Squad; but, as an undefined production is encountered, rather than decide what state should be produced, a new symbolic state is created, and the simulation continues until a suitably large structure has been built up. Then, via a Backtrack Programming algorithm symbolic states are converted to actual states. With each such assignment, we first check to see if a contradiction has occurred. The only kind of contradiction that can occur in this approach is two productions that have equal left-hand sides but unequal right-hand sides. Such contradictions cause backtracking to occur. Second, finding no contradiction, the implications of the assignment are handled. These implications occur when two productions exist with equal left-hand sides but with at least one right-hand side still symbolic. To avoid a later contradiction, the right-hand sides are equated. These new assignments and any new assignments they produce are processed in the above manner until either a contradiction is found, or no more implications are generated. It is these generated implications that provide the look-ahead capability.

The amount of look-ahead depends only on the amount of structure (simulation) generated before making actual assignments. The number of symbolic productions generated is proportional to the amount of simulation. Unfortunately, the search required to look for contradictions and implications is proportional to the square of the number of productions. When enough structure was generated to get an adequate amount of look-ahead, the number of productions generated was large enough to slow the processing down to the point where it more than offset any advantage gained from look-ahead, and nothing new was gained from this approach.

Like the exhaustive search method, this approach was handled entirely by the computer and there was no interaction.

THE TRANSITION TO COOPERATION

By now, my failures had me well convinced that I would be unable to exhaustively search the entire search space (even as constricted by a suitable set of heuristics). Since I believed, and still believe, that a seven-state Minimal Time Solution exists, it made sense to try to find one. I tried two approaches, both of which consisted of a basic backtrack programming algorithm in the machine, and a man-machine interface, by which I could make suggestions on where and for what to search. My entire outlook changed at this point. Rather than trying to build bigger, more powerful heuristic tools, I tried to design simple algorithms that could effectively be directed by me through the man-machine interface. I wanted to be able to specify the outline of a solution, in my terms, and have the machine fill in the details and produce an actual solution of the form specified. In retrospect, this may seem like an obvious course to follow; but at the time, although I was familiar with on-line environments, I had to undergo a rather traumatic rethinking of the problem, not from an implementation standpoint (which was fairly straightforward), but on the conceptual level. I had to switch my thinking from, How can I (embodied either as myself or as a non-Interactive program) find a solution?, to How can WE find a solution? I think the two approaches discussed below show this shift in viewpoint; and, as they are presented chronologically, also an increased emphasis on interaction, feedback, and control of the search process.

FUNCTIONAL PLANNING

From my attempts to find solutions, it was apparent that I started with a plan, or outline, that stated what job or group of jobs each state in the attempted solution would perform. That is, each state was given a function, such as being a middle man, that it was expected to perform. I therefore called this activity functional planning. After deciding on such a functional plan, I tried to define a set of productions that caused each state to perform as specified in the plan. Notice that in this approach, as contrasted with the Constraint Satisfaction presented next, the definition of a function is specified on a local (through productions) rather than global basis.

Viewed as a cooperative venture, I would propose a plan in terms of the functions desired for each state and the computer would search the necessary solution space to determine whether or not a set of productions existed that satisfied the given plan.

The functions I used in my attempts--and that I envisioned using on further attempts--could all, with one exception, be defined by their interaction with other states on a local basis. These functions could therefore be defined by a set of productions or by a set of restrictions on the resultant of certain productions. The basic backtracking program could still be used to do the searching by altering its behavior so that through an input language I could define some productions that would remain unalterable, and restrict the set of allowed states for the resultants of some other productions.

The one function that could not be defined in this manner was the middle man. The reason being that this concept is not local--the determination of the middle of the line involves the entire line. The other functions considered are those described in the section - General Description of the Minimal Time Solution. Basically, all the plans proposed were merely attempts to realize this general type of solution with the given number of states.

This approach was the most successful from two standpoints. First, because it was fairly easy to crank out several different eight-state minimal time solutions including the one presented in my thesis[3] and the one for which an induction proof was obtained. (The plan was specifically designed to make this proof easier.) Second, and more important, this approach produced a conceptual addition to my thinking about the problem. In an effort to specify a plan through production restrictions, I discovered the idea of what I call Image Solutions which are a formal way of saying that the processes going on in one part of the line ought to be also occurring in an "image form" with the directions reversed and using different states in some other part of the line at some other time. That is, all interactions within the line ought to have a counterpart occurring in the opposite direction. This viewpoint also led to the idea of image states that specify the correspondence between states within image processes. Note that this conceptual breakthrough, which I now consider basic to any heuristic search for a solution, arose not through any actual man-machine interaction but through a man-machine symbiotic relationship in which the need to communicate a complex feature of a plan occurred; i.e., the machine acted only as a catalyst, not taking any active role in the discovery.

The interaction for this approach involved observing on-line the simulation behavior as it was periodically printed, and deciding whether or not to let the program proceed. If the decision were made to stop the simulation, a new input specification of the plan had to be prepared before restarting the program. It usually took between 10 to 20 minutes to decide what modifications to make to the plan specification, and about one minute to actually make them and restart the program. The environment under which this was performed was a non-multiprogrammed batch machine for which I was the sole user during my signup periods. Thus, although I was running in a standard batch off-line environment, I was really on-line with the program with an effective response time of one minute.

We frequently talk about the response time of a computer system but almost never about the human response time, except for such trivia as the time to hit a key. In the above man-machine system, my response time was over an order of magnitude longer than the machine's; thus I did not feel, during my work, that the machine was not adequately handling its portion of our partnership. Furthermore, it is hard to see how the machine, for this approach, could have further aided me to cut down my response time. I was still performing too much work in our partnership; work for which my processing capabilities were not well suited.

<u>CONSTRAINT SATISFACTION</u>

The last attempt at finding a solution resulted from the problems encountered in Functional Planning; mainly that I was still supporting too much of the load in terms of real effort expended per iteration. I tried to find a way of specifying a plan that was more natural to the way I was conceptualizing the problem.

My basic representation was the two dimensional one suggested by Moore4 and ray plan followed the kinds of ideas expressed in the section, "General Outline of a Minimal Time Solution[11]. I wanted an easy way of communicating this to the machine. Because I was used to looking at a simulation and, by observing its behavior at various points, seeing if it followed my plan, I decided to specify its behavior in the same way--the medium is the message. That is, I could specify what state should occur at various points and along various lines in the two dimensional representation, and then let the computer--using the trusty backtracking simulation program--see if it could satisfy these constraints.

<u>A</u> graphical interactive program5 was constructed that utilized an IBM 2250 with a RAND Tablet as the interface device. With this program, I could quickly and easily specify, in graphical form, the constraints I desired along any straight line or at any point. In addition, to help further restrict the search space, I could define frozen (unalterable by the computer) productions. I was able to specify on-line: what length line to work on; what the names of the states should be; whether image solutions were

desired or not, and if so what the mapping between image states would be; whether the computer should display the simulation as it did it or whether it should run autonomously; when it should stop (after each time unit in the simulation, or only on asynchronous interrupts); which productions to freeze or thaw; and whether the first occurrence of each production should be brightened or not.

All of this was designed to provide me with a fast, flexible, and facile control of the search. This it did; I was able to very quickly specify a set of constraints and some frozen productions, turn the system loose and get a solution for the given length. This, I think, is a noteworthy advancement over the other approaches.

However, this approach suffered from two problems. First were operational ones such as a terrible flicker on the screen that made it very annoying to use the system, and not enough core to include routines to save the current status so that at some later time the system could be restarted at that point. Second, the use of the system pointed up some lack of foresight in the design of the interactive simulator.

First, the constraints were specified for only one length; and when a successful solution was found for that length, they had to be re-entered, slightly modified, before trying the next length. This became a very tiresome chore, one that should be automated in the next pass on such a system. Second, I did not include the capability in functional planning to talk about classes of states in the constraints. This proved to be a grave mistake, forcing me to overspecify my plans. Finally, I had no inkling beforehand that I constantly would want to constrain all states within a polygon to be members of a class of states. But this turned out to be a rather basic part of the plans; and without this capability, again, I was forced to overspecify the plan t desired.

SUMMARY

From this series of attempts, and my other experiences with man-machine interaction, allow me a few observations. First, often one can get much further with a rather simple, slow interactive system than he can alone--a little symbiosis goes a long way. Second, the NEED to communicate an idea, an aspect of a problem, a plan, or whatever, often produces a much better understanding or even a new way of viewing the thing to be communicated--independent of any later gain from the party to which the information was communicated, be it man or machine. Third, the human response time should be a critical design factor, especially if the task area includes real problem-solving capabilities from the human; and this human response time should in large part determine the "tightness" or "binding" of the interactive relationship. Finally, no matter how much experience one has had with designing and building interactive systems, and no matter how well he knows the task area, he cannot foresee all of the factors that will turn out to be relevant in this system; hence, a good interactive system will require several passes to produce. Let us not be so shortsighted to realize that the product we are touting so highly for eliminating much preplanning--man-machine symbiosis-is also required by us in the building of such systems.

ACKNOWLEDGEMENT

I am indebted to Dr. Allen Newell who introduced me to the Firing Squad Problem and guided me through the solution attempts described in this paper. I also wish to express my thanks to Robert Shirey who designed and built the Firing Squad Constraint Satisfier Program used in my most recent solution attempt.

REFERENCES

1. Waksman, Abraham, "An Optimal Solution to the Firing Squad Synchronization Problem," Information and Control. Vol. 9, No. 1, February 1966, pp. 66-78.

2. Fischer, Patrick C, "Generation of Primes by a One-Dimensional Real-Time Iterative Array," Journal of the Association for Computing Machinery, Vol. 12, No. 3, July 1965, pp. 388-394.

3. Balzer, Robert M., "Studies Concerning Minimal Time Solutions to the Firing Squad Synchronization Problem," Ph.D. thesis, Carnegie Institute of

Technology, 1966,

4. Moore, E. F., Sequential Machines, Selected Papers, Addison-Wesley, 1964.

5. Balzer, R. M. and R. W. Shirey, "The On-Line Firing Squad Simulator," The RAND Corporation, RM-5573-ARPA, August 1968.