A computer replaces the Galapagos Islands as the authors take finite-state machines through a simulated process of evolution and natural selection. Random mutation of an arbitrary machine yields an offspring. The machines then compete and, given available history and a goal to achieve, the machine that gets the higher score thus survives as the fittest and is selected as the new parent. Mutation and selection are continued, with real-time decisions being based upon the logic of the surviving machines.

INTELLIGENT DECISION MAKING THROUGH A SIMULATION OF EVOLUTION[1]

by Lawrence J. Fogel, Alvin J. Owens, and Michael J. Walsh

Decision Science Incorporated, San Diego

INTELLIGENT behavior can result from an ability to predict the environment coupled with the selection of an algorithm which permits the translation of each prediction into a suitable response. For the sake of clarity, the following discussion will be primarily concerned with the problem of predicting the behavior of the observed environment. The term "environment" includes any source of information. For the sake of simplicity, and without loss of generality, the environment will be viewed as a sequential source of symbols. More specifically, the problem at each point in time is to devise an algorithm which will operate on the sequence of symbols thus far observed in order to produce an output symbol which will agree with the next symbol to emerge from the environment. Simulated evolution provides a means towards this end.

DISCUSSION

An arbitrary finite-state machine is exposed to the sequence of symbols which have thus far emerged from the environment. As this occurs, each output symbol from the machine is compared with the next input symbol. The percent correct score is a measure of the ability of this machine to predict the already experienced environment on the basis of the preceding symbols.[2] An "offspring" of this machine is then produced through mutation, that is, through a single modification of the "parent" machine in accordance with some mutation noise distribution. Thus the offspring is made to differ from its parent either in an output symbol,[3] a state-transition, the number of states, or the initial state.[4]

The offspring is now exposed to the same sequence of symbols which were experienced by the parent machine, and its prediction capability is similarly scored. If this score is found to equal or exceed that of the parent, the off-spring "survives" to become the new parent. If not, it is discarded and a new offspring is generated. In this manner nonregressive evolution proceeds through successive finite-state machines which individually evidence increased ability to predict the already experienced sequence of symbols. At any point in time the re-

[1] This paper was published in the IEEE *Transactions on Human Factors in Electronics*, 1965, HFE-6, 1. At the time it was written, the authors were with General Dynamics/Astronautics.

[2] Refer to Appendix A for an illustration of the evaluation of a finite-state machine exercised over a sequence.

[3] In the case of a binary environment, a deterministic procedure can be used to replace this type of mutation. As each symbol from the environment is predicted on the basis of the preceding symbols, score is maintained of the relative frequency of success of each state-transition. A predictive-fit score of greater than 0.5 can then be ensured by the reversal of output symbols on those state-transitions which were "more often wrong than right."

[4] That state the machine is in when it receives the first symbol of its experience.

maining machine can be used for actual prediction; that is, it can be exercised by the last symbol to emerge from the environment, thus producing an actual prediction of the next symbol to be experienced. The same machine is then used to parent succeeding offspring which are evaluated over the now longer recall. Thus the evolution continues in fast time in preparation for the next required actual prediction. Such predictions may take place periodically, aperiodically, or on request. They may be made whenever a specified predictive-fit score has been attained, when some prechosen number of offspring have been evaluated, or when an appropriate number of generations[5] have occurred. Of course, in general, the longer the time interval between successive predictions, the greater the expectation of success. Similarly, the greater the speed of the computer facility (increase in the number of evaluated offspring) or the larger the available memory (increase of their permissible size), the greater the evolutionary prediction capability.

The evolutionary technique offers distinct versatility. For example, the desire to predict each second symbol in the future can be satisfied simply by scoring each offspring in terms of the correspondence between its output symbols and those which emerge from the environment two symbols later. By the same token, appropriate scoring of the offspring permits the prediction of any particular future symbol, the average of some set of future symbols, or indeed, any well-defined function over the future. The desire for minimum error prediction may be satisfied by using a magnitude-of-the-difference error matrix. Minimum *rms* error prediction results if each term of this error matrix is squared. If "a miss is as good as a mile," the error matrix should have equal nonzero off-diagonal terms and zero diagonal terms.

But the purpose of the simulated evolution need not be restricted to prediction in any sense. The input symbols of the evolving machines may be individually associated with the set of possible stimuli, the output symbols with the set of alternative responses—the goal: to achieve any well-defined function over the future sequence of responses. Here there is no longer a distinction between prediction and the algorithm which translates prediction into responses. The evolutionary program recommends each action in the light of its expected overall worth with respect to the given goal.

At the same time it is reasonable, in the interest of economy, to desire that the offspring be of minimum complexity. The maxim of parsimony may be directly incorporated into the evolutionary procedure by reducing the score of each machine in proportion to a measure of its complexity. The amount of this penalty may be influenced by the particular characteristics of the computer facility upon which the simulation is to be carried out. Thus, at each point in time, the evolutionary technique provides a nonregressive search through the domain of finite-state machines for that machine which best satisfies the given goal under the constraint imposed by the available computation capability.

Efficiency of the simulated evolution can be improved in a number of ways. Any available information concerning the underlying logic of the environment can be incorporated in the form of the initial machine. If this "hint" is reasonably correct, the evolution should require fewer generations to attain the same score. If it is incorrect, the introduction of "false" information in no way precludes solution of the problem, although it may be expected to reduce the efficiency of the procedure.

Suitable choice of the mutation noise can increase the efficiency of evolution. For example, an increase in the probability of adding a state generates a wider selection of larger machines which should benefit evolution against a complex environment. In fact, the probability distribution over the modes of mutation can be made to depend upon the evidence acquired within the evolutionary process itself. Thus, an experienced greater relative frequency of success for, say, changing the initial state might be made to increase the probability of this mode of mutation. Although this procedure may

[5] Defined at an appropriate point below

benefit the prediction of independent environments, it can offer a danger if the environment is interactive—an intelligent adversary might discover the specific dependency and use this knowledge to construct an obverse strategy.

The evolutionary search may be viewed as a selective random walk, a "hill climbing" procedure, in a hyperspace defined to include the finite-state machines and an additional coordinate on which is measured the predictive-fit score.[6] The danger of becoming trapped on a secondary peak can be overcome by permitting multiple mutation, with the multiplicity being a function of the difference in predictive-fit score of successive generations. Thus, as the search nears a peak, greater and greater attention is devoted to generating more radical offspring in the hope of striking a point which may lie higher on the slope of another peak.

The evolutionary technique may be expected to predict nonstationary environments because of its continual search for a "best" logic. But selection of only the single best logic may be an overly severe policy. Certainly those offspring which have high predictive-fit scores also characterize the logic of the environment in some meaningful manner. Why not mimic nature and save the best few machines at each point in time? In general, the highest scoring offspring has the greatest probability of giving rise to an even better offspring; thus it should receive most attention in terms of mutative reproduction. Lower ranked offspring may be regarded as insurance against gross nonstationarity of the environment. The distribution of mutative effort may well be in proportion to the normalized predictive-fit scores. Evaluated offspring are inserted into the rank order table of retained offspring and a generation is said to occur whenever an offspring is found which has a score equal to or greater than the score of the best machine.

All of the retained machines need not lie on the slopes of the peak which is identified by the best machine. Thus, saving the best few offspring may maintain a cognizance over several peaks, with the relative search effort being distributed in proportion to the expectation of significant new discoveries.

The efficiency of natural evolution is enhanced by the recombination of individuals of the opposite sex. By analogy, why not retain worthwhile traits which have survived separate evaluation by combining the best surviving machines through some genetic rule, mutating the product to yield offspring? Note that there is no need to restrict this mating to the two best surviving individuals. In fact the most obvious genetic rule, majority logic, only becomes meaningful with the combination of more than two machines.[7] Clearly, this opens the door to many new possibilities, although only the majority logic combination has been examined; for example, it may be fruitful to explore the combining of the best machines of several different generations in the hope of finding a model of the models which had this far been most successful.

THE EXPERIMENTS

In the interest of brevity only some of the series of experiments which were conducted to explore the predictive capability of the evolutionary technique will be reported, these being numbered consecutively for ease of reference. The original evolutionary program was written in FORTRAN II for the IBM 7094 to permit prediction of two-symbol environments. This set of experiments demonstrated the feasibility of predicting cyclic environments, stationary environments, and the primeness of each next number in the sequence of positive increasing integers (as reported in Fogel, Owens, and Walsh (1964 a, b).

A second program was then written to permit the prediction of eight-symbol environments. Unless otherwise indicated, all of the following experiments started with the same arbitrary five-state machine. The recall was permitted to grow with experi-

[6] The term "predictive-fit" is used in place of the more general "functional-fit" in view of the more immediate concern with the problem of prediction.

[7] Refer to Appendix B for an example and discussion of the use of a majority logic machine.

ence starting with 40 symbols before the first prediction. The penalty for complexity was chosen to be 0.01 times the number of states in that machine. Single, double, or triple mutation of each parent machine occurred with equal probability and a maximum of 40 offspring or ten generations were permitted before each successive actual prediction.

The first set of experiments concerned the prediction of an environment composed of a cyclic signal created by repetition of the simple pattern 13576420 which was disturbed by increasing levels of noise. With the environment consisting only of the undisturbed signal (Experiment 1), the evolutionary technique discovered a perfect one-state predictor machine within the first 18 evaluated offspring. The environment for Experiment 2 was generated by corrupting this signal by the equally likely addition of +1 or −1 to certain symbols, these being identified by skipping a number of symbols from the last disturbed symbol in accordance with the next digit drawn from a uniformly distributed random number table. Quite arbitrarily, addition to the symbol 7 and subtraction from the symbol 0 were assumed to leave these symbols undisturbed. Thus, 82.5% of the symbols were left undisturbed. As shown in Figure 1, 59.3% of the first 81 predictions were correct, there being only six errors in the last 30 predictions. During the evolution 3,241 different offspring were evaluated, the predictor machines growing in size to eight states.

The environment of Experiment 3 was obtained by disturbing the environment used in Experiment 2, once again in the same manner. Thus, 28.9% of the symbols were disturbed by ±1; 1.5% were disturbed by ±2; leaving 69.6% undisturbed. As shown in Figure 1, 39.5% of the first 81 predictions were correct, there being a general increase in score in the last 20 predictions. During the evolution 3,236 different offspring were evaluated, the predictor-machines growing in size to 15 states.

The environment for Experiment 4 was obtained by disturbing the environment used in Experiment 3 again in the same man-

EVOLUTIONARY PREDICTION OF THE ENVIRONMENT

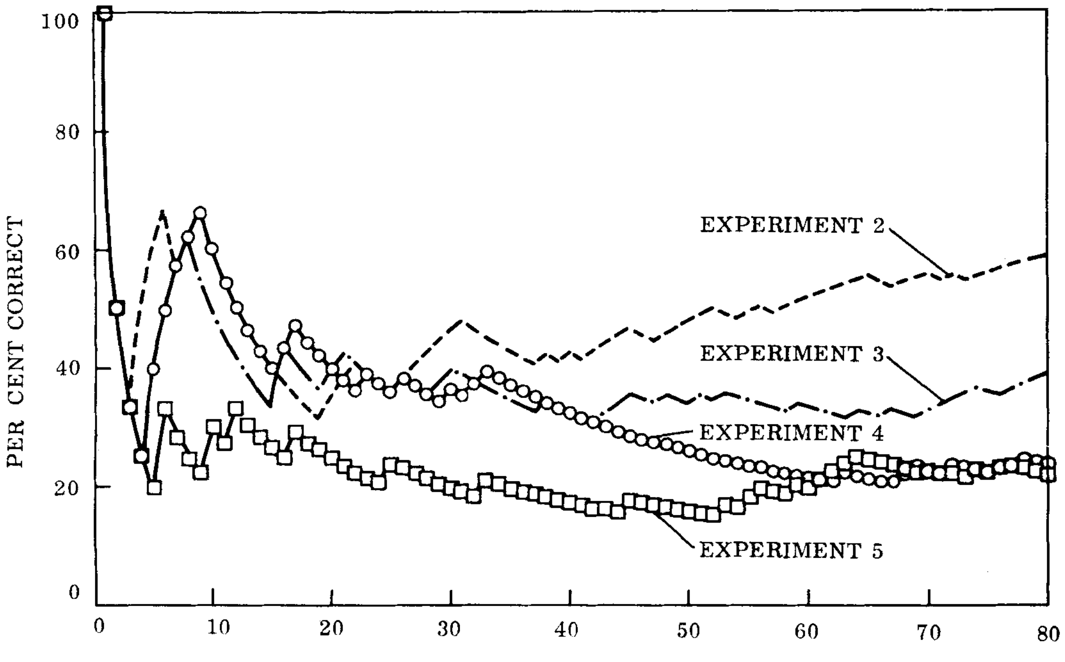


Fig. 1

ner. Thus, 37.0% of the symbols were disturbed by ±1; 2.5% were disturbed by ±2; and 0.1% were disturbed by ±3; leaving 60.4% of the symbols undisturbed. As shown on Figure 1, 23.5% of the first 81 predictions were correct. During the evolution 3,214 different offspring were evaluated, the predictor machines growing in size rather steadily to 19 states.

The environment of Experiment 5 was obtained by disturbing a randomly chosen 50% of the symbols in the signal. Thus, 43.8% of the symbols were disturbed by ±1 (the difference being due to the adopted rule concerning addition to 7 and subtraction from 0); leaving 56.2% undisturbed. As shown in Figure 1, 22.2% of the first 81 predictions were correct. During the evolution, 3,195 different offspring were evaluated, the predictor machines growing in size somewhat irregularly to 17 states. It would appear that this last increase in the noise level (from 39.67% in Experiment 4, to 43.8% in Experiment 5) resulted in significantly degraded prediction of the environment for only the short recalls.

Figure 2 indicates the degree of correspondence between the sequence of predictions and the signal in these experiments. Note that after the first 76 predictions the signal was predicted in Experiment 5 as well as it was in Experiment 4, in spite of the fact that a larger percentage of the signal symbols had been disturbed. This may be due to the fact that in the last experiment the symbols remained closer to the original signal. Such consideration for the magnitude of deviations is a result of using a distance-weighted error matrix, in this case the weighting being the magnitude of the symbol difference. In essence this choice converts the nominal scale of symbols to an ordinal scale.

It is of interest to examine each of the predictor machines as representations of the periodic properties of the environment. The characteristic cycle for any finite state machine is found by starting it in its initial state together with the first symbol of the recall, then driving it by each of its successive output symbols until the output sequence is periodic. All of the characteristic cycles in

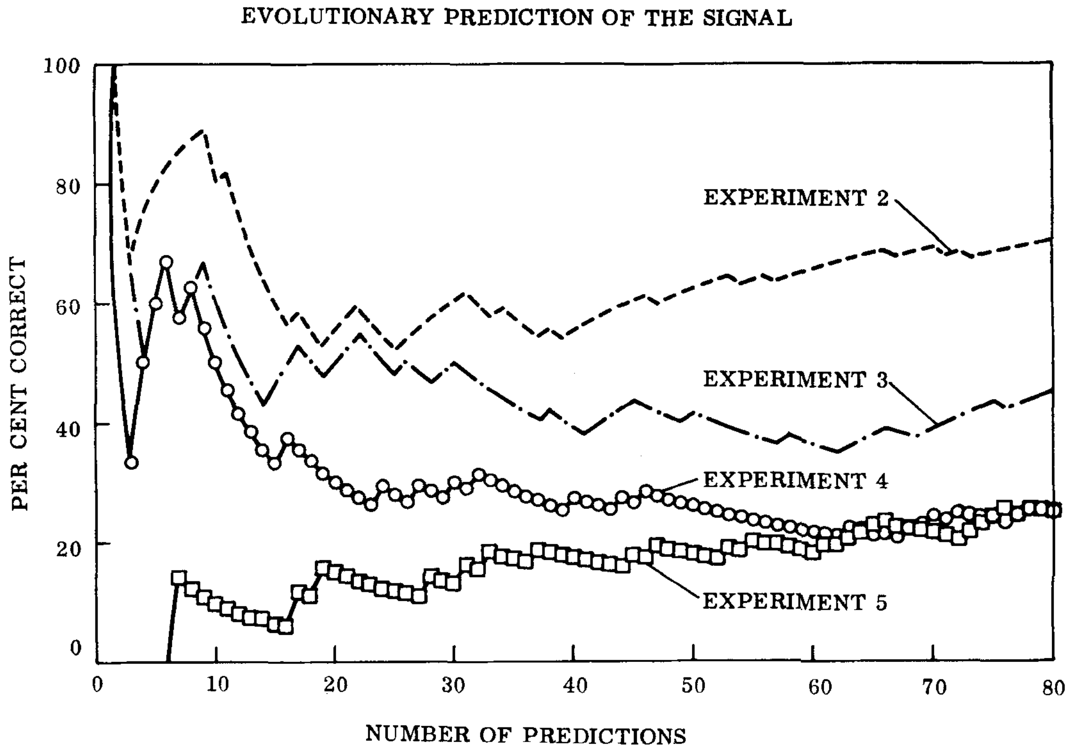


Fig. 2

COMPARISON OF EXPERIMENTS 5 AND 6

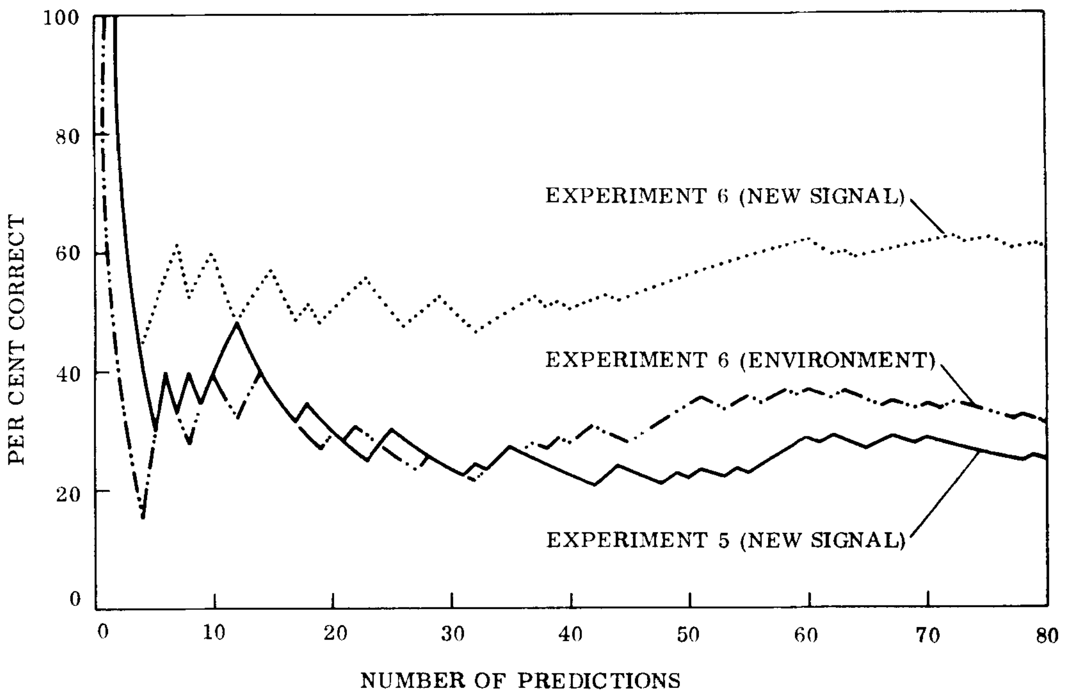


FIG. 3

Experiment 2 were eight symbols in length. The first 73 corresponded perfectly with the pattern of the signal but this insight was lost in the later predictions which were in error by one or two symbols. After the 14th prediction the characteristic cycle remained 13576430.

The higher noise level of the third experiment resulted in characteristic cycles of varying length until the 26th prediction. From then on until the 70th prediction the characteristic cycle remained 1357643113576430, this being in error one symbol out of every two cycles of the signal. As expected, the result of Experiment 4 was more erratic with the length of characteristic cycle jumping from 8 to 16 and remaining the same after the 73rd prediction. Each of the last 24 characteristic cycles was 62.5% correct. Experiment 5 revealed even greater variability in the characteristic cycles. A majority of these were 8 or 16 symbols in length and reflected the basic pattern of the signal although there was little one-to-one correspondence.

The environment for Experiment 6 was generated by disturbing every symbol of the signal by $+1$ or -1 with equal probability. At first glance, it is surprising to find the prediction of the signal improved as shown in Figure 3; but note that with the disturbance of every symbol one aspect of the randomness of the environment was removed. In essence the signal had taken on a new form—the boundaries of the original signal 24677531, or 02465310, each having equal probability at each point in time. In the first 81 predictions this new signal was properly identified 70.4% of the time. In fact, the characteristic cycle of the last predictor machine was 02267731102665331. This can be seen to lie on the boundaries of the original signal except for one symbol of every 17. In order to provide a basis for comparison, the percent correct prediction of the environment is also shown as well as the result of Experiment 5 with respect to the new signal.

It is natural to inquire as to the extent to which the prediction capability will be de-

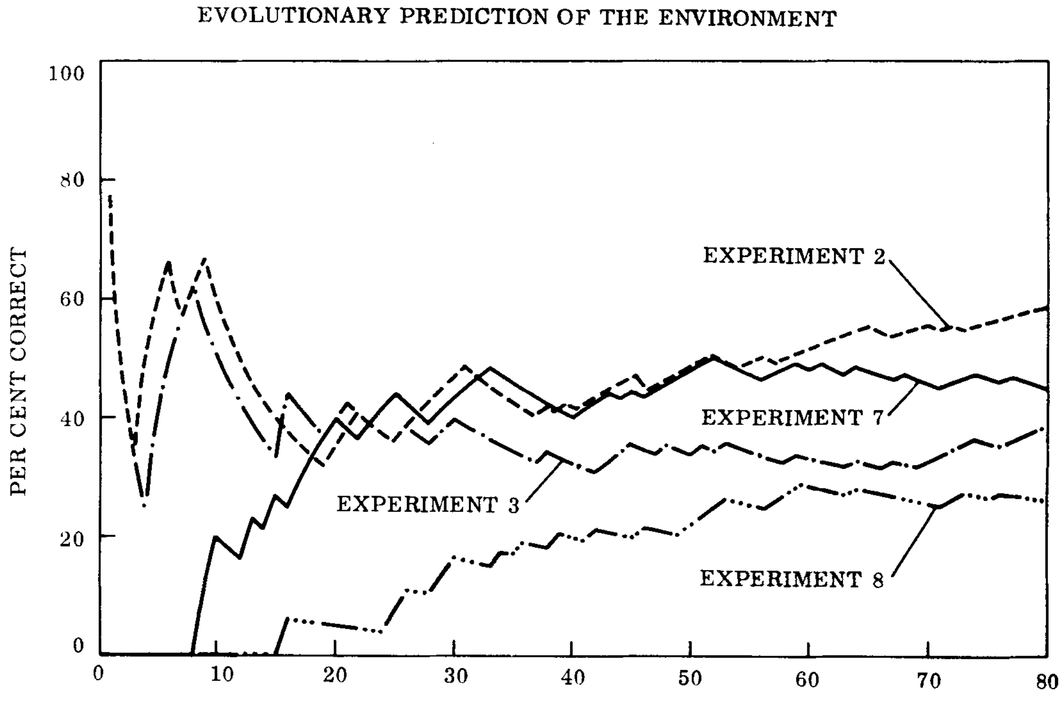


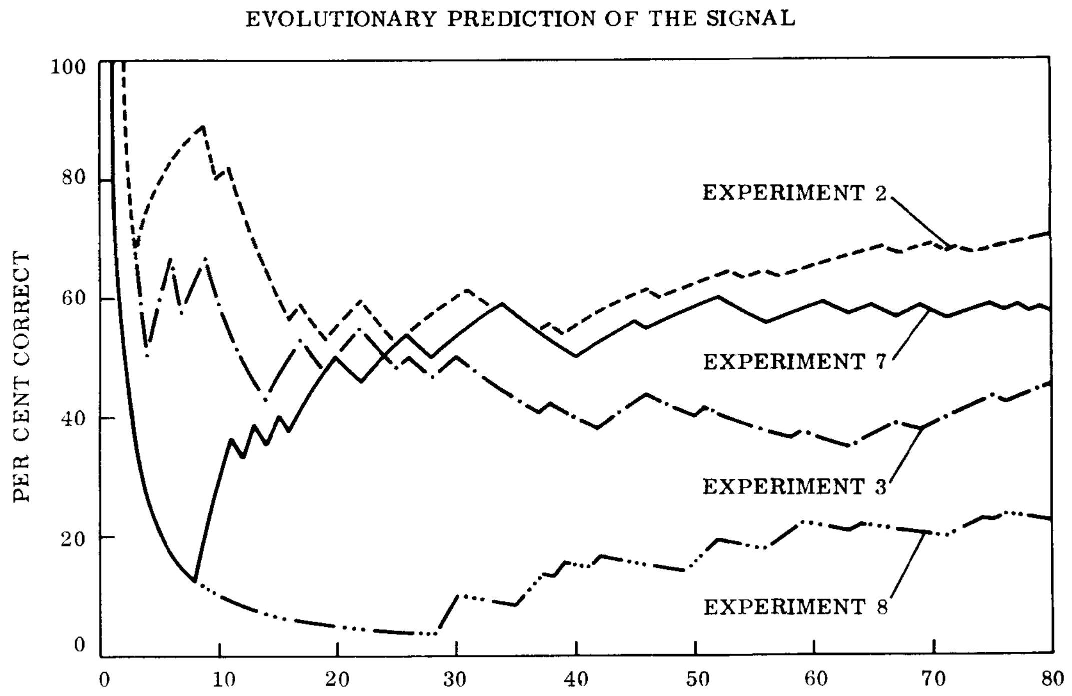
Fig. 4

graded by "wild noise" (each disturbed symbol being replaced by a randomly chosen symbol from the input alphabet). The environments of Experiments 7 and 8 were generated by imposing this kind of disturbance on the original signal once and twice, respectively. As expected, the ability of the evolutionary program to predict the environment, as shown in Figure 4, was somewhat poorer than in the comparable Experiments 2 and 3. Figure 5 indicates the degree of correspondence between the sequence of predictions and the signal in these same experiments. Here again the additional degree of randomness within the noise degrades the performance. Carrying this noise to the extreme results in a perfectly random environment. Experiment 9 revealed no significant ability of the evolutionary program to predict this environment.

The introduction of randomness always introduces questions of repeatability. In order to examine this point Experiment 2 was repeated nine additional times, the results being shown in Figure 6. As expected, the variability is an inverse function of the score.

The second set of experiments was concerned with the prediction of purely stochastic environments. Experiment 10 required the prediction of a zeroth order four-symbol Markov environment, the arbitrarily chosen probabilities being 0.1, 0.2, 0.3, and 0.4. This information, as prior knowledge, would dictate the continual prediction of the most probable symbol giving the asymptotic score of 40%. At the other extreme, perfectly random prediction would have an expected score of 25%. As shown in Figure 7 the evolutionary score settled between these extremes, thus demonstrating the purposeful extraction of information from the previous symbols.

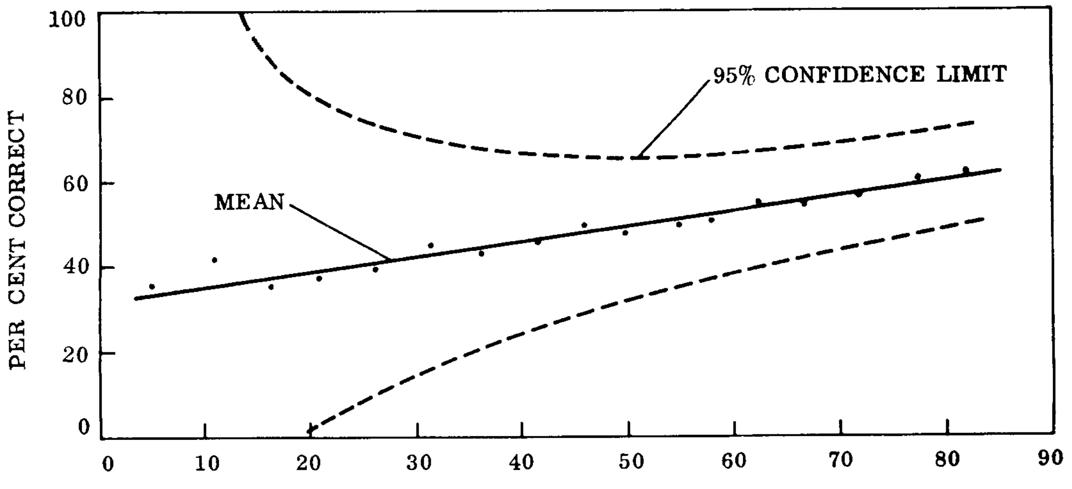
The first order environment of Experiment 11 was generated in such a manner as to produce theoretically the results shown in Table 1. Specifically, after an initial arbitrary symbol, each next symbol of the environment was generated utilizing the relative frequency distributions of the next



NUMBER OF PREDICTIONS

Fig. 5

symbol, given a preceding symbol as indicated by the rows of the matrix shown in Table 1 where the symbols in the first row are the next symbols following the preceding symbols in the left column. The actual environment generated had the transition matrix of relative frequencies shown in Table 2. The marginal frequencies of this environment were 0.236, 0.241, 0.249, and 0.274, respectively.



NUMBER OF PREDICTIONS

Fig. 6

EVOLUTIONARY PREDICTION OF ZEROTH-ORDER MARKOV ENVIRONMENT

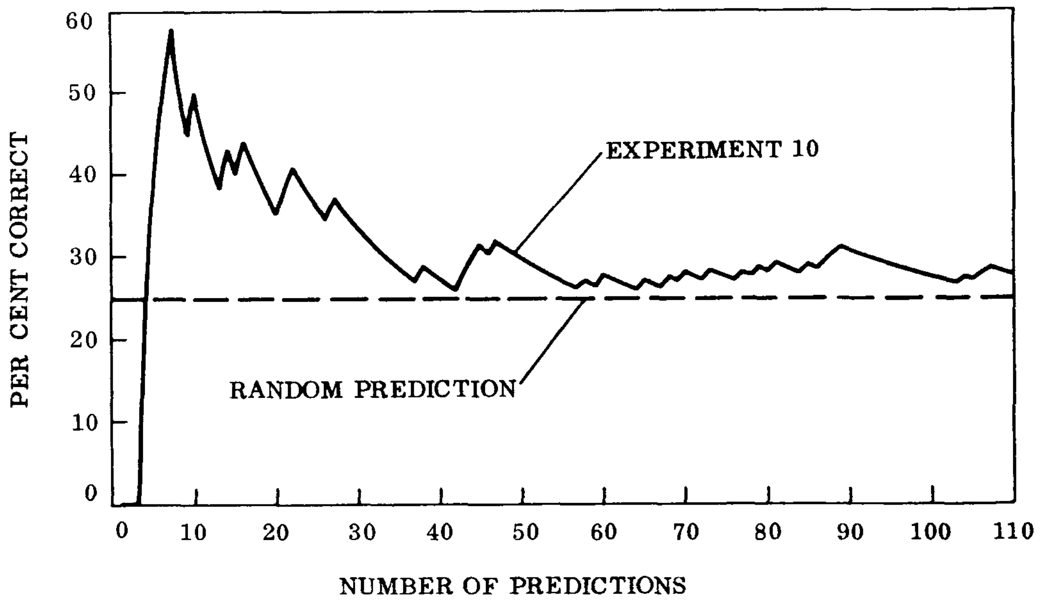


Fig. 7

TABLE 1

	0	1	2	3
0	0	0.8	0.1	0.1
1	0.1	0	0	0.9
2	0.9	0.1	0	0
3	0	0.1	0.8	0.1

TABLE 2

	0	1	2	3
0	0	0.822	0.071	0.107
1	0.035	0	0	0.965
2	0.915	0.085	0	0
3	0	0.077	0.862	0.061

With prior knowledge that the process is first-order, it would be possible to attain the score of 89.5% on the 200th prediction in the manner shown in Figure 8. *But even without this knowledge the evolutionary prediction technique attained this same score at each point.* Analysis of the sequence of predictions revealed that at the end of this experiment the environment was properly characterized by the maximum transition probabilities of each row. Other experiments were conducted on first and second order processes with similar results.

To generate a more difficult environment, the powers of 2 and 3 were rank ordered and reduced modulo 8, that is, 1234010300103001003010030100300. . .[8] After the first 300 predictions (Experiment 12) the percent correct score reached 88.7; 1,401 different offspring were evaluated using an "all-or-none" error matrix. (All off-diagonal elements have a value of 7 and all main diagonal elements a value of 0.) The predictor machines were generally of six states. In order to avoid the reduction to three symbols in the latter portion of the sequence, the powers of 2 and 3 were rank ordered reduced modulo 7 yielding 1234-122641425411243122461424512143122461. . . After the first 216 predictions (Experiment 13), the percent correct score was 56.6, this

[8] Refer to Appendix C for a detailed discussion of this environment.

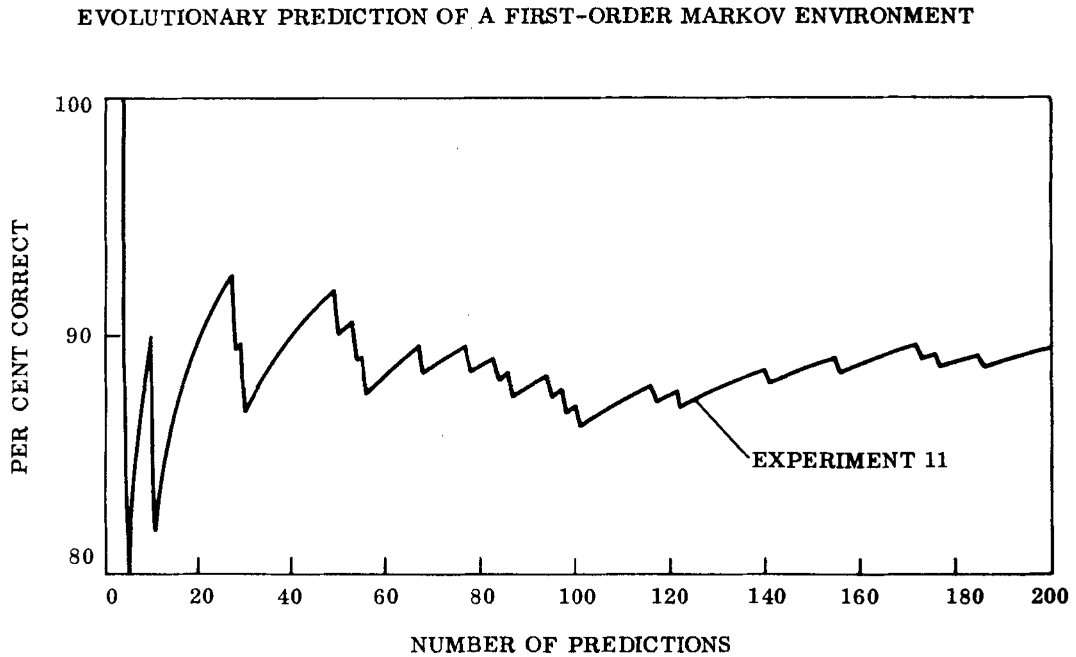


FIG. 8

being found through the evaluation of 3,671 offspring which were generally of about 21 states. The score for the last 50 predictions was 78%. Certainly the prediction capability was far better than chance would yield. Prediction of this sequence based on the most probable symbol up to each point in time yields a score of only 8.5%.

A set of experiments was conducted in order to evaluate the evolutionary technique as a means for detecting the existence of correlation between variables, this correlation to be used to enhance the sequential prediction of one or more of the sensed variables. In Experiment 14, a random sequence of binary symbols was presented to an arbitrarily chosen finite-state machine, M (shown in Table 3). Note that four different symbols comprise the output alphabet of this machine. The evolutionary program was required to predict each next symbol in the output sequence, this being predictable only to the extent that the structure of the machine M is superimposed upon the otherwise random driving signal. The per-cent correct score, shown in Figure 9, reveals increasing stability as the score settles around 51% after the first 152 predictions (with 20 symbols as the initial recall).

TABLE 3
MACHINE M

Present State	Input Symbol	Next State	Output Symbol
1	0	3	2
1	1	5	0
2	0	1	2
2	1	2	3
3	0	1	1
3	1	4	1
4	0	5	3
4	1	3	0
5	0	2	0
5	1	1	2

This prediction score should be considerably improved if the program were permitted to evolve finite-state representations on the basis of both the four-symbol output sequence and the two-symbol input sequence. In fact, with this additional information, the prediction score should asymptotically approach 100% as the recall increases in view of the deterministic nature of machine M. Experiment 15 was conducted to investigate whether or not this would take place through the evolutionary technique. At each point in

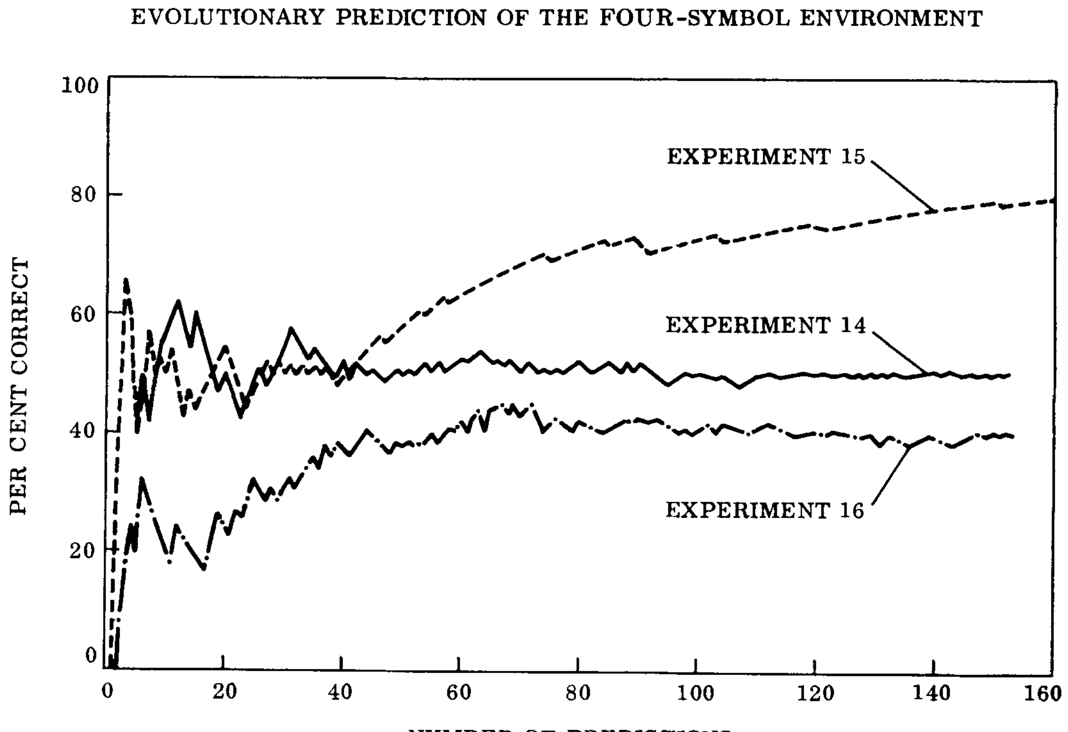


Fig. 9

time two-input single-output finite-state machines were evaluated in terms of their predictive fit of the known recall of the four-symbol output sequence while driven by the preceding recall of both the four-symbol and two-symbol sequences. The percent correct score, shown in Figure 9, attains a value of 80%.

In some sense, the sequence of evolved double-input single-output machines should provide an increasingly accurate representation of the transduction performed by machine M. Figure 10 is a graphical representation for the finite-state machine which evolved after 160 predictions. This machine, N, has the states designated by letters in order to avoid any confusion with the number-designated states of machine M. It is possible to compare machine N to machine M in the following way: examine each state of M in order to determine whether there is some state in machine N which in some sense will perform the same function when given the same sequence of input signals. (Note that the input to each state of machine N consists of an ordered pair of symbols, the first being the input to the state machine M and the second being the previous output of machine M before it entered that state.)

To illustrate, consider state one of machine M. Since the outputs of the transitions which enter this state are either 1 or 2, the possible corresponding inputs to machine M are (0, 1); (0, 2); and (1, 1) and (1, 2). Examination of state B of machine N shows that (0, 1) and (0, 2) both yield the output symbol 2 which corresponds to the output of state 1 when it receives an input symbol of 0. Similarly (1, 1) and (1, 2) produce an output of 0 which corresponds to the output of state 1 when it receives input symbol 1. Similarly, (1, 1) and (1, 2) produce output symbol 0 which corresponds to the output of state 1 when it receives an input symbol 1. Thus, state 1 of machine M may be said to be contained in state B of machine N in the sense that machine N will yield the same response if given a corresponding sequence of inputs. Following this logic it can be shown that state 3 is contained in state A and state 5 is contained in state B.

MACHINE N

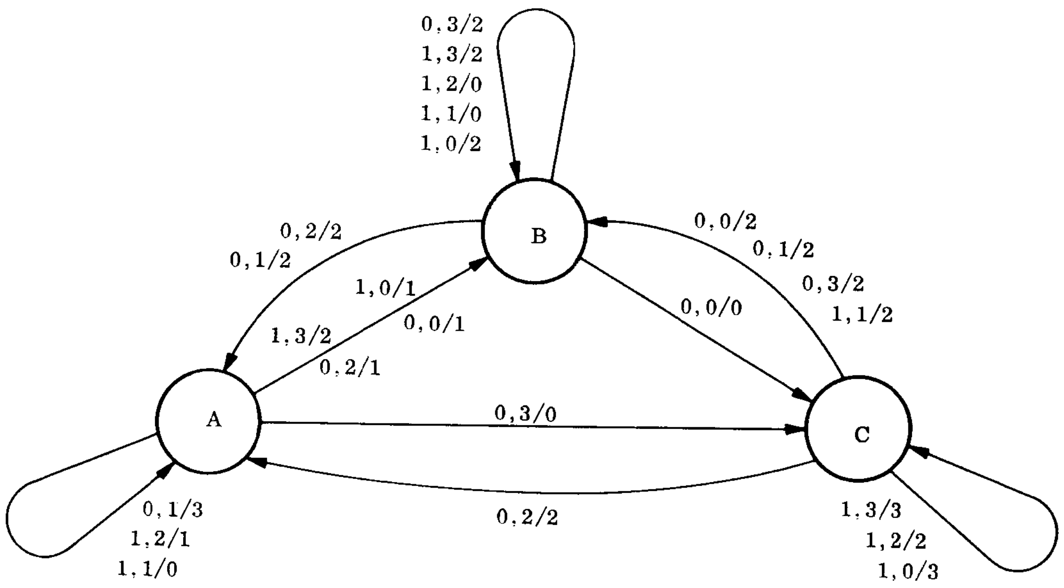


FIG. 10

Viewing the problem in the large, it is often essential to erect hypotheses concerning the logical relation between two or more separately sensed variables. It would appear that the evolutionary technique offers a means for finding multiple-input predictor machines which can be translated into reasonable representations for the observed logical relationship.

Experiment 16 was conducted to confirm the claim that the evolutionary technique had, indeed, extracted useful information from the correlative input variable in Experiment 15. In Experiment 16 two-input single-output machines were evolved in the very same manner as before, but in this case the binary variable which was offered was uncorrelated with the output sequence taken from M in that it was independently generated. The percent correct score, shown in Figure 9, remains below the corresponding scores which resulted from the previous two experiments. In other words, the additional information which was furnished in the form of an independent binary variable formed a distraction resulting in poorer predictive fit and predictions. More specifically, the more complex machines which were considered made successful mutation and selection less likely.

In view of the apparent need, a formal technique was devised which permits expression of the set of finite-state machines which are functionally contained within any given machine. Thus it becomes possible to translate each predictor machine into a set of hypotheses concerning the transduction logic of the environment. The finite-state machines which are functionally contained in the set of machines used for successive prediction can be examined for consistency which may be expected to provide additional insight.

CONCLUSION

The key to artificial intelligence lies in automating an inductive process that will generate useful hypotheses concerning the logic that underlies the experienced environment. The creatures of natural evolution are just such hypotheses, survival being the measure of success. As described above, some fundamental aspects of this process can be replicated with finite-state machines proceeding through iterative mutation and selection in order to find a best logic to satisfy the given goal in the light of the available memory experience. Real-time decisions can then be based upon the extracted results of this continuing fast-time evolution.

A series of experiments was conducted in order to evaluate this concept. An eight-symbol language evolutionary program was given the goal of predicting a variety of time series on the basis of the previous symbols. These environments consisted of a cyclic signal embedded in various kinds and amounts of noise, purely stochastic environments which contain the signal only in the form of the transition matrix, nonstationary samples of deterministic sequences, and multivariate sequences which were the input and output of an arbitrary transduction. In every case, the results attained compared favorably with those which would be expected from conventional techniques using only the same information. It was, in fact, possible to identify some of the cyclic and stochastic properties of the signals from the predictor machines.

The success of these experiments demonstrates that useful models of the regularity within a sequence of observation can, indeed, be found by the evolutionary technique. Although this demonstration was restricted to a variety of goals relevant to prediction there would appear to be no need to maintain this constraint. So long as the correctness of each response can be evaluated prior to the next response, the goal may be made to reflect the immediate concern of the decision maker.

The first phase of computer technology was devoted to the development of equipment which would carry out a large number of simple operations quickly and reliably. The second phase of computer technology was devoted to the development of languages which permit the detailed instruction of this equipment in an efficient manner. Computer technology is now entering a new phase in which it will no longer be necessary to specify exactly how the problem is to be solved. In contrast, it will only be necessary to provide an exact statement of the problem in terms of goal and costs in order to allow the evolution of the best program possible within the available computation capability. Solution of the problem then includes a statement of the discovered algorithm. The old saw "the computer never knows more than the programmer" is simply no longer true.

The scientific method consists of induction, inductive inference, followed by independent verification. Hypotheses, generated so that they cover the available evidence and additional data points, are individually evaluated in terms of the validity of their inference. Those that prove worthy are modified, extended, or combined to form new hypotheses which carry on a heredity of reasonableness. As the hypotheses correspond more and more closely with the logic of the environment they provide an understanding which can be used for the improvement of goal-seeking behavior in the face of that environment.

The correspondence between natural evolution and the scientific method is obvious. Individual organisms in nature serve as hypotheses concerning the logical properties of their environment. Their behavior is then an inductive inference concerning some as yet unknown aspects of their environment. Their health is a measure of this suitability. Their offspring include the heredity of reasonableness as well as additional information resulting from mutation and recombination.

The evolutionary technique described above is then a realization of the scientific method in which the hypotheses are restricted only in the sense that they are finite-state machines. The creation of successive hypotheses requires the introduction of randomness as well as the prevailing logic of inheritance. The versatility which is essential to intellect is a natural product of the iteration. In essence, the scientific method is an essential part of nature. It is no wonder, then, that the overt exercise of the scientific method has provided mankind with distinct benefits and now permits even its own automation through the artificial evolution of automata.

APPENDIX A

Figure 11 and Table 4 are constructed to illustrate the transduction accomplished by a finite-

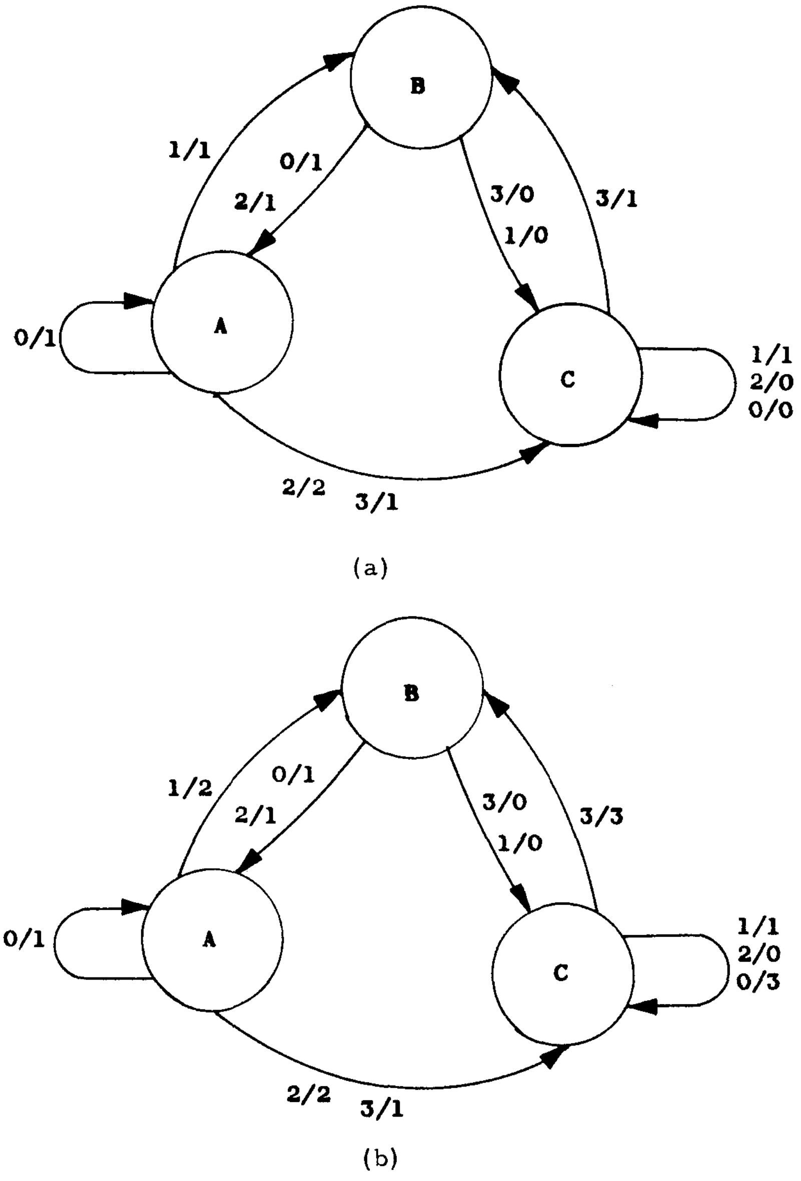
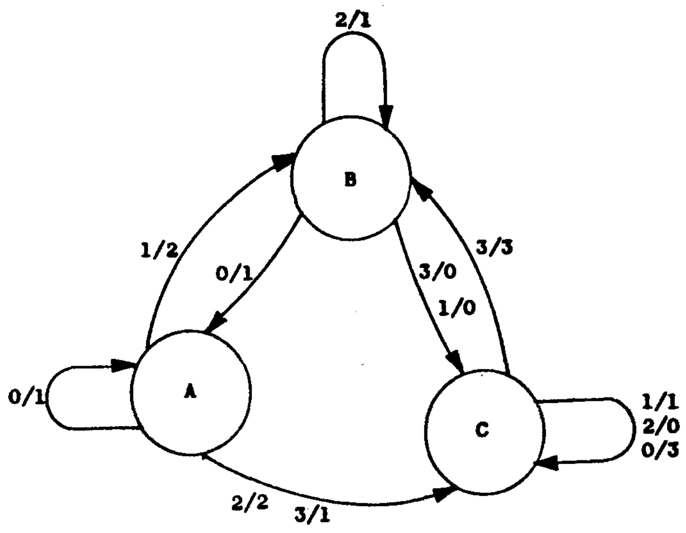
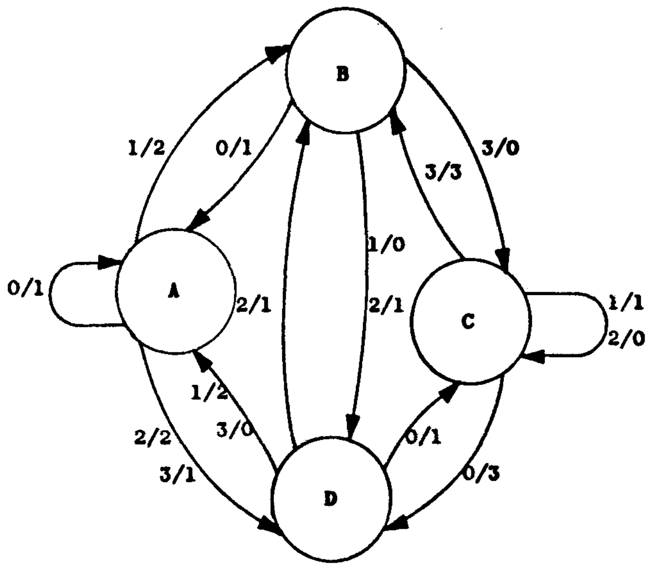


Fig. 11. Mutation of Finite-State Machines.

state machine over an experience of eleven symbols, together with the score achieved over that recall and the prediction of the unknown twelfth symbol. Table 4 (a) is constructed from Figure 11 (a). The first line described as "State" gives the state of the finite-state machine, with the initial state being the left most symbol—here B. The second line described as "Environment" has to the left of the vertical line the known sequence of symbols used as input to the machine. The most recent symbol is to the immediate left of the vertical line while the symbol to the immediate right is the unknown symbol to be predicted. The third line is the output resulting from the machine. Note that in line three the machine output from a state-input pair is presented shifted one symbol position to the right. The output symbol is found in the following manner. The arrows in Figure 11 represent a transition from one state to another (a looped arrow indicates remaining in the same state). Each arrow has associated one or more triplets of the type 0/1 which means that if the machine is in the state at the base of the arrow and experiences the symbol to the left of the virgule,



C



D

FIG. 11–Continued

it will output the symbol to the right of the virgule and change to the state at the head of the arrow. In line three the symbol to the right of the vertical line is the machine prediction of the unknown symbol. Lines four, five, and six are obtained by means of error matrices:

$[A_{ij}]$ where $A_{ij} = 0$ if $i = j$ and $A_{ij} = 1$ if $i \neq j$; $[A_{ij}]$ where $A_{ij} = |i - j|$; and $[A_{ij}]$ where $A_{ij} = (i - j)^2$, respectively.[9]

Any error term, A_{ij}, is found from the error matrix with i being equal to machine output and j the next symbol of the input sequence.

Machine scores are calculated by summing the error terms and dividing the sum by the number of terms. Three machine scores are given for each machine; a score is associated with the three error matrices, respectively. Obviously, only one

[9] Error Matrix One is used for an "all or nothing at all" scoring, Error Matrix Two, the distance-weighted error matrix, is used for a minimum error scoring, while Error Matrix Three has elements of an rms scoring.

TABLE 4

(a)

State	B A C C C C B C C B A		Machine—Figure 11 (a)
Environment	2 2 1 0 1 3 3 0 3 0 1	2	Start State B
Output	1 2 1 0 1 1 0 0 1 1	1	Machine Score
Error Score Matrix 1	1 1 1 1 1 1 0 1 1 0		.8
Error Score Matrix 2	1 1 1 1 2 2 0 3 1 0		1.2
Error Score Matrix 3	1 1 1 1 4 4 0 9 1 0		2.2

(b)

State	B A C C C C B C C B A		Machine—Figure 11 (b)
Environment	2 2 1 0 1 3 3 0 3 0 1	2	Start State B
Output	1 2 1 3 1 3 0 3 3 1	2	Machine Score
Error Score Matrix 1	1 1 1 1 1 0 0 0 1 0		.6
Error Score Matrix 2	1 1 1 2 2 0 0 0 3 0		1.0
Error Score Matrix 3	1 1 1 4 4 0 0 0 9 0		2.0

(c)

State	B B B C C C B C C B A		Machine—Figure 11 (c)
Environment	2 2 1 0 1 3 3 0 3 0 1	2	Start State B
Output	1 1 0 3 1 3 0 3 3 1	2	Machine Score
Error Score Matrix 1	1 0 0 1 1 0 0 0 1 0		.5
Error Score Matrix 2	1 0 0 2 2 0 0 0 3 0		.8
Error Score Matrix 3	1 0 0 4 4 0 0 0 9 0		1.8

(d)

State	B D B D C C B C D A A		Machine—Figure 11 (d)
Environment	2 2 1 0 1 3 3 0 3 0 1	2	Start State B
Output	1 1 0 1 1 3 0 3 0 1	2	Machine Score
Error Score Matrix 1	1 0 0 0 1 0 0 0 0 0		.2
Error Score Matrix 2	1 0 0 0 0 2 0 0 0 0		.3
Error Score Matrix 3	1 0 0 0 4 0 0 0 0 0		.5

(e)

State	A D B D C C B C D A A		Machine—Figure 11 (d)
Environment	2 2 1 0 1 3 3 0 3 0 1	2	Start State A
Output	2 1 0 1 1 3 0 3 0 1	2	Machine Score
Error Score Matrix 1	0 0 0 0 1 0 0 0 0 0		.1
Error Score Matrix 2	0 0 0 0 2 0 0 0 0 0		.2
Error Score Matrix 3	0 0 0 0 4 0 0 0 0 0		.4

type of scoring would be utilized in any particular instance.

Figure 11 (b) represents the machine of Figure 11 (a) mutated by changing the output symbols of the transitions to State B.

Figure 11 (c) represents the machine of Figure 11 (b) mutated by changing a transition from State B to State A to remain in State B.

Figure 11 (d) represents the machine of Figure 11 (c) mutated by adding State D. Note that adding a state also involves changing existing transitions to the new state in order to incorporate the new state into the functional aspects of the machine.

Conversely, the machine represented in Figure 11 (c) could be obtained by deleting State D from the machine represented in Figure 11 (d).

Parts (b), (c), (d), (e) of Table 4 are constructed in a similar manner as that described for (a). In particular, Table 4 (e) is associated with a change in start state of the machine represented in Figure 11 (d).

APPENDIX B

On the use of a majority logic machine

Under certain conditions the use of redundancy and a majority logic element can enhance the reliability of performance of a system. It is always possible to draw a single-state diagram

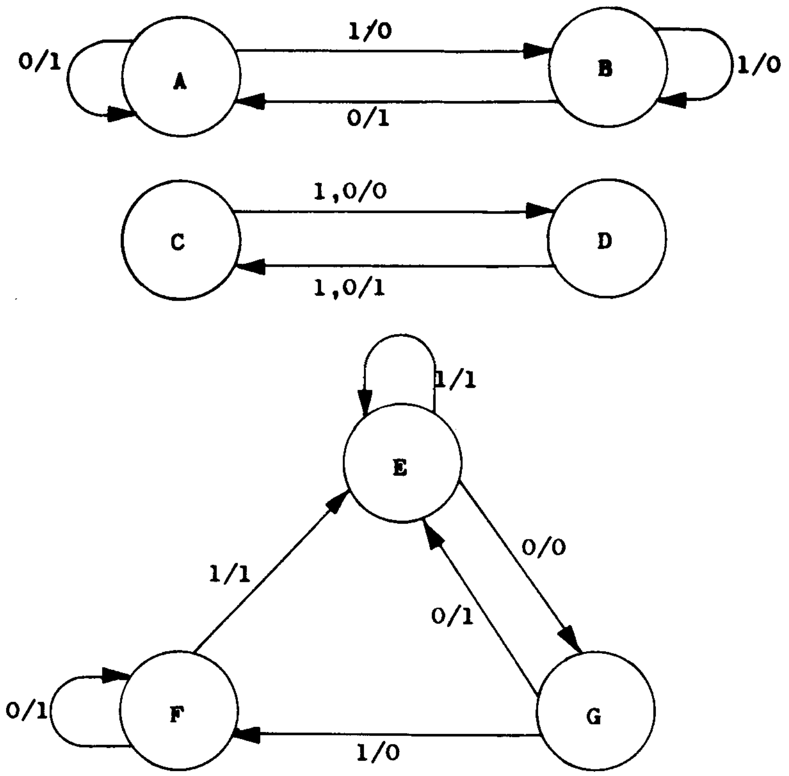


FIG. 12. Individual Machines to be Combined.

which expresses the logic of such a redundant array of finite-state machines. Each state of the majority logic machine is a composite of a state from each of the original machines. Thus the majority machine may have a number of states as great as the product of the number of states in the original machines. Each transition of the majority machine is described by that input symbol which caused the respective transition in the original machines, and by that output symbol which results from the majority element logic being applied to the output symbols from each of the original machines. To illustrate, Figure 12 indicates three original machines and their individual probabilities of success, that is, p_i. The output of these machines is to be combined through a majority logic element which weights the importance of each output by the demonstrated probability of success of that machine. The resulting majority machine is shown in Figure 13.

So long as there are only two original machines the weighted majority logic machine reduces to the better of these two machines. A more interesting situation occurs when there are three original machines, these being rank ordered so that $p_1 \geqq p_2 \geqq p_3$. In view of the above described deterministic output symbol reversing technique, these scores cannot be less than 0.5; therefore, it will always be true that $p_1 \geqq p_2 + p_3$. Because of this inequality the weighted majority machine reduces to the simple unweighted majority machine.

The score for the majority machine, p_M, can be expressed as the sum of the ways in which success can be attained, that is,

$$p_M = p_1p_2p_3 + p_1p_2q_3 + p_1q_2p_3 + q_1p_2p_3$$

where q_i is the probability of failure, so that $p_i + q_i = 1$. This expression can be simplified to

$$p_M = p_2p_3(1 - 2p_1) + p_1(p_2 + p_3).$$

Now the majority machine is only of value if $p_M \geqq p_1$, that is, if $p_M - p_1 \geqq 0$. Substituting yields the inequality

$$p_2p_3(1 - 2p_1) + p_1(p_2 + p_3) - p_1 \geqq 0.$$

This relationship can be expressed in graphical form for specific values of p_1 and solved for maximum value of p_M. Figure 14 shows the regions in which pairs of values of p_2 and p_3 yield a $p_M \geqq p_1$ for these values of p_1. Table 5 indicates the corresponding values for the maximum p_M.

This same procedure can be used for the evaluation of the benefit to be achieved through a majority logic element over any number of origi-

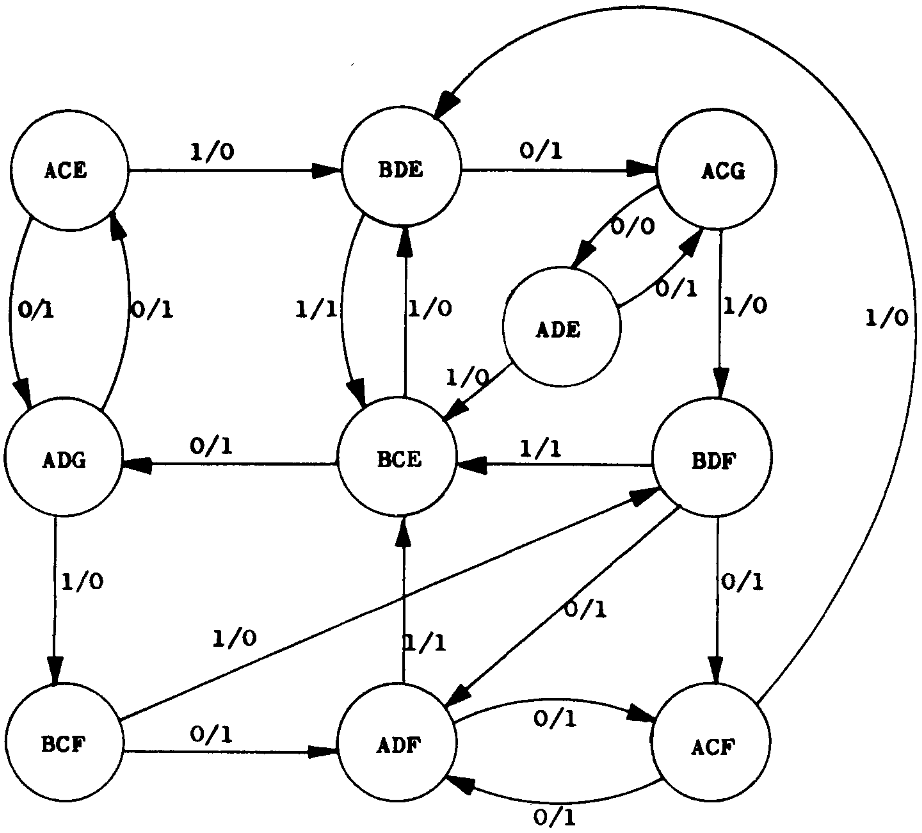


FIG. 13. The Resulting Majority Logic Machine.

nal machines; however, the required calculations rapidly become cumbersome. Further for large numbers of original machines it becomes worthwhile to examine the behavior of the weighted majority logic machine as opposed to the simple majority logic machine.

APPENDIX C

The problem of rank ordering the powers of 2 and of 3 may be simplified by writing all of the powers in terms of a common base. For example, the powers of 3 may be written in terms of powers of 2 taking $2^{1.584925}$ as an approximation. Since $2^x > 2^v$ if, and only if, $x > v$, the problem is reduced to one of ordering the positive integers and the integral powers of 1.584925. Since this basic exponent is in error by less than 10^{-7}, 10^7 powers of 3 may be considered before an incorrect ordering could occur.

Taking a modulus of this increasing series constrains the numbers to the alphabet available for the evolutionary program. With respect to modulo 8, the powers of 2 yield the residues 2, 4, followed by 0's, while the powers of 3 are alternately 3 and 1. To see this note that $2^1 = 2$ (mod 8), $2^2 = 4$ (mod 8) and $2^3 = 0$ (mod 8). For $k > 3$, $2^k = 2^3 \cdot 2^{k-3} = 0 \cdot 2^{k-3}$ (mod 8) $= 0$ (mod 8). Similarly, $3^1 = 3$ (mod 8), $3^2 = 1$ (mod 8) so that for n even, that is $n = 2k$, $3^{2k} = (3^2)^k = 1^k$ (mod 8) $=$ 1 (mod 8); while for n odd, that is $n = 2k + 1$, $3^{2k+1} = 3^{2k} \cdot 3 = 1 \cdot 3$ (mod 8) $= 3$ (mod 8).

With respect to modulo 7, the powers of 2 form the repeating sequence 2, 4, 1, 2, 4, 1, $\cdots$ while powers of 3 form the repeating sequence 3, 2, 6, 4, 5, 1, 3, 2, 6, 4, 5, 1, $\cdots$ This can be seen by noting that $2^3 = 1$ (mod 7), $2^{3k} = (2^3)^k = 1^k$ (mod 7) and $1^k = 1$ (mod 7); $2^{3k+1} = (2^{3k}) \cdot 2 = 1 \cdot 2$ (mod 7) $= 2$ (mod 7) while $2^{3k+2} = (2^{3k})2^2 = 1 \cdot 4$ (mod 7) $= 4$ (mod 7). Similarly, since $3^6 = 1$ (mod 7), $3^{6k} = (3^6)^k = 1^k$ (mod 7) $= 1$ (mod 7); $3^{6k+1} = 3^{6k} \cdot 3 = 3$ (mod 7); $3^{6k+2} = 3^{6k} \cdot 3^2 = 3^2$ (mod 7) $= 9$ (mod 7) and $9 = 2$ (mod 7); $3^{6k+3} = 3^{6k} \cdot 3^3 =$ 27 (mod 7) and $27 = 6$ (mod 7); $3^{6k+4} = 3^{6k} \cdot 3^4 = 81$ (mod 7) and $81 = 4$ (mod 7); $3^{6k+5} = 3^{6k} \cdot 3^5 = 243$ (mod 7) and $243 = 5$ (mod 7).

Therefore, to form the sequence of residues modulo 8 it is only necessary to order the integers and the integral multiples of 1.584925; that is, the sequence 2, 4, followed by all 0's is inserted in successive positions corresponding to the integers

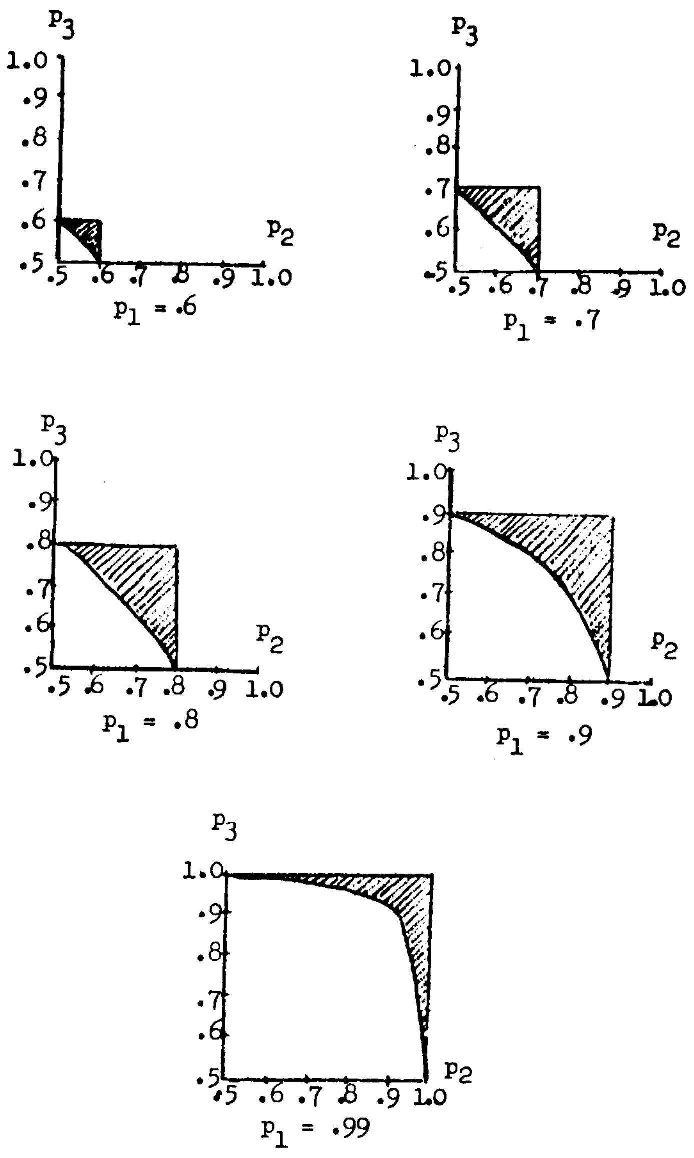


FIG. 14. Majority Logic Improvement Domains.

TABLE 5

THE MAXIMUM p_M RELATED TO SPECIFIC p_i

p_1	Max p_M
0.6	0.648
0.7	0.784
0.8	0.896
0.9	0.972
0.99	0.9997

while the repetitive sequence 3, 1, ··· is inserted in the successive positions corresponding to the multiples of 1.584925. Similarly, for modulo 7 the integer sequence 2, 4, 1, 2, 4, 1, ··· is interlaced with the sequence 3, 2, 6, 4, 5, 1, 3, ··· which corresponds to the multiples of 1.584925. The resulting sequences are not periodic although sub-periods are embedded throughout. This lack of periodicity results since in higher multiples of 1.584925 the successive decimal digits influence the integral part and thus upset any pattern imposed by the more significant digits.

REFERENCES

Fogel, L. J., Owens, A. J., & Walsh, M. J. On the evolution of artificial intelligence. *Proceedings of the Fifth National Symposium on Human Factors in Electronics*, IEEE, San Diego, May 5-6, 1964a, 63-76.

Fogel, L. J., Owens, A. J., & Walsh, M. J. An evolutionary prediction technique. Presented before International Conference on Microwaves, Circuit Theory, and Information Theory, IEEE, Tokyo, Japan, September 7-11, 1964b. Summary published in the *Proceedings*, Part 3, 173-174.

Fogel, L. J., Owens, A. J., & Walsh, M. J. Artificial intelligence through a simulation of evolution. In M. Maxfield, A. Callahan, & L. J. Fogel (Eds.) *Proceedings of the 2nd Cybernetic Sciences Symposium: Biophysics and cybernetic systems*. New York: Spartan Books, 1965, p. 184.

(Manuscript received August 21, 1965)

The question whether the law of causality applies in the same strict sense to human actions as to other phenomena, is the celebrated controversy concerning the freedom of the will, which, from at least as far back as the time of Pelagius, has divided both the philosophical and the religious world. The affirmative opinion is commonly called the doctrine of Necessity, as asserting human volitions and actions to be necessary and inevitable. The negative maintains that the will is not determined, like other phenomena, by antecedents, but determines itself; that our volitions are not, properly speaking, the effects of causes, or at least have no causes which they uniformly and implicitly obey.

I have already made it sufficiently apparent that the former of these opinions is that which I consider the true one; but the misleading terms in which it is often expressed, and the indistinct manner in which it is usually apprehended, have both obstructed its reception and perverted its influence when received. The metaphysical theory of free-will, as held by philosophers (for the practical feeling of it, common in a greater or less degree to all mankind, is in no way inconsistent with the contrary theory), was invented because the supposed alternative of admitting human actions to be *necessary* was deemed inconsistent with every one's instinctive consciousness, as well as humiliating to the pride, and even degrading to the moral nature, of man. Nor do I deny that the doctrine, as sometimes held, is open to these imputations; for the misapprehension in which I shall be able to show that they originate unfortunately is not confined to the opponents of the doctrine, but is participated in by many, perhaps we might say by most, of its supporters.

John Stuart Mill