computers in the psychology laboratory · vol. 1

DIGITAL EQUIPMENT CORPORATION

COMPUTERS IN THE PSYCHOLOGY LABORATORY

Selected Papers and Presentations of

Digital Equipment Computer Users Society

DIGITAL EQUIPMENT CORPORATION ▫ MAYNARD, MASSACHUSETTS

The papers presented in this manual were originally printed in the DECUS (Digital Equipment Computer User's Society) Proceedings.

CONTENTS

Computer technology has brought many innovations to the Psychology Fields in recent years. Psychologists involved in many diversified applications, such as studying social behavior in primates, determining the effects of drugs on behavior, or numerous other areas, have found that a general purpose computer can be implemented to suit their particular needs.

A computer can aid the Psychologist in many ways. It can interact closely with the experimental subject, providing the Psychologist with a greater degree of control over his experiment. The flexibility to change parameters easily and quickly during an experiment is possible through programming software. The computer can collect, analyze and store the resulting data for future reference. And this can be done with greater speed than previously thought possible, while maintaining precise accuracy as well.

The papers presented here were selected to introduce the reader to some of the specific application areas where computers are now being widely used. As additional papers on Psychology applications are contributed to DECUS, they will be published annually.

Our many thanks are extended to the authors for their time and effort in preparing these papers, and for sharing their progress with us.

If you would like further information on computer applications in Psychology, please contact us. We will be glad to help you.

Biomedical Marketing
Digital Equipment Corporation
Maynard, Massachusetts

DIGITAL COMPUTERS IN THE BEHAVIOR LABORATORY[1]

Bernard Weiss
Department of Radiation Biology and Biophysics
University of Rochester School of Medicine and Dentistry
Rochester, New York

ABSTRACT

Computer technology will have a two-fold impact on behavior research. First, it makes possible, because of the ability of the computer to acquire and store data, a microanalysis of behavior. Even conventional reinforcement schedules demonstrate new features when such analyses are possible. Second, it enables the experimenter to exert direct control over elements of behavior that could be controlled only indirectly earlier. The latter contribution of computer technology to the behavior laboratory will surely be its most revolutionary one.

Experimental psychologists have always been prone to develop methods for automating their research. Their reasons, I think, are fairly obvious. Research on the mechanisms of behavior, because of all the possible sources of bias and distortion to which it is subject, requires, even more than other areas of research, the utmost precision and reliability, and the greatest freedom from adventitious variables. These are among the reasons that psychologists, very early in the development of computer technology, become frequent users of the bulk processing machines that formerly were the only computers available. They have even more reason now to turn to computer technology. Now they have available machines not only to help process their data but to help them conduct experiments. This is why, of course, our discipline is so well represented here today, and was so well represented in the original LINC Evaluation Program. The LINC, most recently described by Clark and Molnar[3], grew out of the view that a laboratory computer was much closer to the biomedical scientist's needs than the facilities provided by conventional computer centers. A dozen of us had the good fortune, in 1963, to be selected as subjects for an experiment: an experiment to determine whether or not physiologists, psychologists, etc., could learn to program and maintain such a computer and integrate it into the programs of their laboratories. The success of that experiment, described in detail in the final report of the LINC Evaluation Program[4], is a major reason that we can hold this kind of meeting today.

Since computer technology in the behavior laboratory is still a recent innovation, and since so little has been published up to now, I will aim not at a detailed review of any single aspect of the use of computers in our laboratory, but, instead, at a survey of applications. By no means do these exhaust the number of uses to which we have put computers in our laboratory, as some of you already know.

[1] This paper is based on work supported, in part, by grant MH-11752, from the National Institute of Mental Health, and performed under contract with the U. S. Atomic Energy Commission at the University of Rochester Atomic Energy Project and has been assigned Report No. UR-49-791.

One of the uses to which we have put computer technology is the analysis of behavior that has been studied in conventional ways. One way in which we study learned behavior, that is, behavior under the control of its consequences, is to arrange contingencies between occurrences of behavior and occurrences of reinforcement (rewards) which are able to maintain rather distinctive patterns of behavioral events in time.

One such reinforcement schedule has been designated by the letters DRL, which stand for Differential Reinforcement of Low Rate. If an experimental animal, such as a monkey, is given access to a response device, such as a lever, one can arrange a set of contingencies as follows: on each occasion that 20 seconds or more passes from the preceding response on the lever to the next response, the second response is followed by the delivery of a reinforcer, such as food. An animal exposed to such a set of demands eventually emits a distribution of interresponse time (IRT) intervals of which a large proportion fall beyond the minimum pause required for reinforcement. If one examines the resulting behavior, however, in a much more detailed fashion than simply the first order interval histogram or the cumulative record, more subtle features of the behavior become discernible. For example, various investigators have observed that reinforcements tend to occur in clumps. There will be a train of interresponse times long enough to secure reinforcements, then a train of interresponse times that are too short. The first figure is a photograph from the LINC scope, showing successive interresponse times from a monkey (M. nemestrina) held in a primate restraining chair to which is attached a lever and a solenoid valve that, when actuated, releases a small quantity of fruit drink as a reinforcer. The upper portion of the figure shows a portion of untransformed IRT's from Session 8, the lower portion of the figure shows the same distribution after having been smoothed by a 3-term moving average to make the grosser details more visible. The drifting up and down around the minimum IRT required for reinforcement (in this case, 20 seconds) is easily seen and, on a cumulative record, is expressed as a clumping of reinforced and unreinforced responses.[11]

With practice, this particular monkey eventually became so adept that only on rare occasions did he fail to wait long enough to produce a reinforcement. Figure 2 is a picture, again from the LINC oscilloscope, showing 128 successive interresponse times from Session 30. Note that only on rare occasions did the interresponse time fall below 20 seconds. Because of the serial effects that we saw earlier, we wanted to know whether or not in a distribution seemingly so stable serial dependencies also existed. For that purpose we first turned to the serial correlogram, with the results shown in Figure 3. The solid line represents the correlogram that would have been obtained had the interresponse time dependencies been a function of a simple Markov process. All that one can deduce from this plot is the possible presence of a long wavelength drift in the behavior. The autocorrelogram (and serial correlogram) are notorious, however, for their ability to mask phenomena of relatively low amplitude. For that reason, we turned to the power (or variance) spectrum, which plots frequency not time, against amplitude. Spectral density plots for four successive sessions, corresponding to the serial correlograms in Figure 3, are shown in Figure 4. Despite taking the precaution of removing linear trends beforehand, long wavelength phenomena did appear, as shown in the large amount of variance at very low frequencies. However, there is a clearcut patterning in all four sessions in the higher frequency phenomena. Notice the peaks near 0.5 and 0.2. These frequencies are interpreted as follows: A frequency of 0.5 means a tendency to alternate; successive interresponse times bounce up and down. A frequency of 0.25 indicates a tendency for cyclical effects to occur with a period of four interresponse times. Figure 5, taken from Session 30, on which one of these power spectra is based, shows empirically what the variance spectrum shows mathematically. The open circles are the actual IRT values. The heavy line is a 3-term moving average. Observe how the interresponse times alternate about the moving average. They are, from this standpoint, negatively correlated, which is one reason for our inability to discern effects in the serial correlogram.

An examination of the variance spectrum is also useful in the case of drugs, a feature that engages our interest because our laboratory is devoted to research in behavioral pharmacology. Figure 6, from Weiss and Laties[12], presents two variance spectra, one obtained after a control solution (saline), the other obtained after the injection of 0.2 mg/kg of amphetamine sulfate. Notice that the short wavelength phenomena have been decreased considerably in amplitude.

The DRL reinforcement schedule is a deterministic schedule. By that I mean that it contains no programmed random components. The probability of reinforcement jumps immediately from zero to 1 when the interresponse time exceeds the specified minimum. A stochastic, or probabilistic, analog of the DRL schedule can easily be programmed by a computer. Figure 7 depicts such a function. This is a schedule we call Stochastic Reinforcement of Waiting (SRW)[9] which possesses the feature that the probability of reinforcement is directly related to the duration of the interresponse time. The longer the interresponse time, the greater the probability of reinforcement. The behavior established by such a schedule of reinforcement, which is closely related to a variable interval reinforcement schedule, is characterized by a preponderance of responding at a fairly substantial rate, thereby producing an interresponse time distribution with a peak at, say, one or two seconds. With our interest in spaced responding schedules, such as the DRL, we began to study performance on a different SRW function, one resembling a Gaussian distribution and which is depicted in Figure 8. With such a distribution, very short interresponse times are never reinforced.

Given the data acquisition and analytical capacity of a computer, it is possible to follow microscopically the development of the behavior in response to the new contingency. I will not pursue this in any intensive way but simply offer some illustrations of this development. In Figure 9 I have plotted, for the first 23 sessions on the Gaussian distribution, the mean interresponse time for each block of 100 interresponse times. For, say, the first 4,000 interresponse times, the mean lies close to 8 seconds (left half of Figure 9). The right half of Figure 9 contains the mean reinforced IRT for each block of 100 IRT's. For these approximately 4,000 interresponse times, the mean reinforced interresponse time is quite high. One can infer from such a distribution that the behavior is characterized by trains of very short interresponse times alternating with long pauses. The pauses, when terminated by a response, produce reinforcement. During the first 4,000 responses nearly all of the reinforced interresponse times come about in this way. There appears to be a rather dramatic change after that point–a quickly rising mean interresponse time that simultaneously produces a fall in the mean reinforced interresponse time, bringing it close to 40 seconds, which, as can be seen from Figure 8, is the point at which the probability of reinforcement equals 1.0.

Figure 10 shows, for Monkey 14, interval histograms for three sessions. The top histogram refers to performance on the linear function immediately preceding the change to the Gaussian function. The middle histogram is a distribution of interresponse times during the first session on the Gaussian function. The bottom histogram refers to the fifth session on the Gaussian function. Notice that during the first session on the Gaussian distribution there is a shift of interresponse times to the left, brought about in part by the phenomena associated with extinction, namely, bursting. By Session 5 we see the beginnings of a bimodal distribution, one peak remaining at about 2 seconds, another peak developing between, say, 20 and 36 seconds. The characteristics of his behavior can perhaps be seen more readily in Figure 11. Figure 11 is an

expectation density plot of the same sessions shown in Figure 10. The expectation density is analogous to the autocorrelation function. During the linear function preceding the shift to the Gaussian distribution and during the first session on the Gaussian distribution there appears to be no inherent patterning in the performance. The number of occurrences per second remains fairly constant. By Session 5, however, the bimodality of the behavior is quite apparent. Following a response at t = 0, there is an increased probability of response a second or so later. It gradually falls, then begins to rise once more, becoming steady at about 30 seconds. With further practice, the early peak is eliminated and most responses are reinforced.

Of course, we anticipate that in the future the most revolutionary applications of computer technology will come not simply from data analysis but from the ability to control on-going experiments. The next reinforcement schedule I wish to discuss comprises just such an application. The DRL schedule is an attempt, from certain aspects, to modify the variability or uniformity of rate. By placing a lower boundary and, in some cases, an upper boundary on reinforced interresponse times, one is able to generate a relatively narrow distribution. This is simply a by-product of the contingencies. With access to an on-line computer, one can specify directly the variability or uniformity required for reinforcement. With monkeys, again, we have studied a reinforcement schedule I call the Autoregressive Reinforcement Schedule (ARG).[10] It takes its name from a species of time series generated by a Markov process, in which succeeding events depend on the just prior states of the system.

The ARG schedule specifies the probability of reinforcement as a function of the serial correlation of interresponse times. The function which controls probability is given in Figure 12. Given an IRT, I_i, the probability of reinforcement depends on how similar I_i+1 is to I_i. In the computer program, the ratio of the larger over the smaller is computed and then a number taken from a table stored in the memory. This number is equivalent to a particular probability with respect to the distribution of numbers generated by a random number generator. When each occasion scheduled for reinforcement produces reinforcement, a situation we call Fixed Ratio 1 (FR1) is in effect; the kind of distribution evoked is shown at the top of Figure 13. The peak is rather sharp and the distribution quite narrow, especially considering that the time scale gives us a resolution of 10 msec. Suppose that, instead of reinforcing on each occasion when reinforcement is due, we only reinforced every fourth such occasion (FR4). Such a change produces a distribution (Figure 13, middle) with even less variability. There is a further narrowing when we make the requirement FR8, as shown at the bottom of Figure 13.

It is apparent, from this figure and from the next, Figure 14, which shows the corresponding expectation densities, that the monkeys meet the requirement of serial dependency not so much by producing serial dependencies but by acting as an oscillator. Naturally, a purely periodic emission of responses will result in reinforcement on each occasion. There is, however, a significant though small component of sequential dependency in the data.[10] Thus, the monkeys are meeting the contingencies of reinforcement with a 2-component process, as it were.

One of the most effective ways of controlling behavior, from the standpoint of obtaining a maximum amount of information with a minimal output of time, is to employ what have been called 'adjusting,' 'adaptive,' and 'titration' schedules of reinforcement. One of the simplest examples is the technique devised by Bekesy[1] for measuring auditory thresholds. A subject is given a hand switch connected to a specially-designed audiometer which sweeps slowly through the range of audible frequencies. While it is doing so, the intensity of the stimulus continually decreases. The subject presses the switch when the signal is no longer detectable and holds it long enough for the sound to be detected again. When he releases the switch it moves in the opposite direction once more. In this way it is possible, within a relatively short time, to trace an audiogram. Blough[2] devised a somewhat more complex system, employing pigeons, in order to measure visual thresholds, and we have used a system quite similar to Bekesy's for measuring pain thresholds.[7] More recently, using the LINC, we have employed a titration schedule which uses response duration as its primary parameter.

Our interest in response duration stems from earlier work with dogs on drug combinations.[8] Figure 15 is a flow chart of a program which attempts to adjust the criterion of response duration so as to force the dog to give longer and longer responses. The dog presses against the panel with its nose. If the duration of the response exceeds an initial criterion, deliberately set low, the criterion is increased. In order not to make too large a jump, it is increased by 25 per cent of the difference between the former criterion and the response duration which exceeded it. We found that taking too large a jump often extinguished the behavior. If the dog fails to meet the criterion, the criterion may be decreased. The rate at which it falls depends on how close the dog is to meeting the criterion. If the last few responses have been close to the criterion, the requirement falls more quickly than if the dog has been emitting a long train of short-duration responses. In this way, we do not reinforce sequences of very short response durations by precipitously decreasing the criterion. The reader familiar with the vocabulary of behavior will see that basically we are shaping long-duration responding in much the same way that an experimenter demonstrating shaping in the classroom trains a pigeon to turn a circle or to perform even more exotic gymnastics.

During an experiment, the LINC scope face gives a running account of the trend of the behavior. In Figure 16 the series of points at the left represents successive response durations. The smoother function on the right shows the criterion as it moves upward or downward, as the case may be. The numbers to the right represent, from top to bottom, the number of reinforcements still available before the session ends, the most recent response duration, and the current criterion. Figure 17 shows a Calcomp plot of one session for a dog that has been well trained. Notice how the criterion climbs upward as the dog gives longer and longer responses.

Drugs such as amphetamine produce a larger output of short-duration responding and keep the criterion lower than under control conditions.

These are just a few examples of the way in which we have used the LINC in the laboratory. There are many other applications. Experiments on Stochastic Reinforcement Schedules are still continuing. In collaboration with Karl Lowy I have used the LINC to study methods for quantifying evoked potentials.[5] Louis Siegel[6] has developed a device for digitizing graphic records and a device for automatic control of a stereotaxic instrument. C. T. Gott is employing the LINC to study chemical influences on behavioral transition states in pigeons. Experiments on the relationship between brain electrical stimulation and behavior and on hallucinogenic drugs are also underway.[7] The LINC is also used in a course on computer technology oriented toward on-line use (Figure 18).

Each new application opens up further vistas. Each new application generates further ideas about the application of computer technology to the problems of the behavior laboratory. I have found, in contrast to predictions made by pessimistic friends, that computer technology, rather than capturing the behavior of the experimenter, gives it much wider scope and frees his imagination and energy for questions and projects that, in the past, he never would have pursued.

References

1. Bekesy, G.V. A new audiometer. Acta Oto-Laryngol., 1947, 35, 411-422.
2. Blough, D. S. A method for obtaining psychophysical thresholds from the pigeon. J. exp. Anal. Behav., 1958, 1, 31-43
3. Clark, W. A. and Molnar, C. W. A description of the LINC. In Stacy, R. W. and Waxman, B. D. (Eds.) Computers in Biomedical Research, Vol. 2. New York: Academic Press, 1965, Ch.2.
4. Convocation on the Mississippi. A Summary Description and Proceedings of the Final LINC Evaluation Program Meeting. St. Louis: Washington University Computer Research Laboratory, 1965.
5. Lowy, K. and Weiss, B. Further assessments of the signals in averaged evoked potentials with an on-line computer. Fed. Proc., 1967, 26, 589.
6. Siegel, L. Digitizing graphic records for computer analysis. IEEE Trans. Bio-Med. Eng., 1967, 14, 7-10.
7. Weiss, B. and Laties, V. G. Fractional escape and avoidance on a titration schedule. Science, 1958, 128, 1575-1576.
8. Weiss, B. and Laties, V. G. Effects of amphetamine, chlorpromazine, pentobarbital and ethanol on operant response duration. J. Pharmacol. exp. Therap., 1964, 144, 17-23.
9. Weiss, B. and Laties, V. G. Drug effects on the temporal patterning of behavior. Fed. Proc., 1964, 23, 801-807.
10. Weiss, B. and Laties, V. G. Reinforcement schedule generated by an on-line digital computer. Science, 1965, 148, 658-661.
11. Weiss, B., Laties, V.G., Siegel, L. and Goldstein, D. A computer analysis of serial interactions in spaced responding. J. exp. Anal. Behav., 1966, 9, 619-626.
12. Weiss, B. and Laties, V. G. The comparative pharmacology of drugs affecting behavior. Fed. Proc., 1967 (in press).

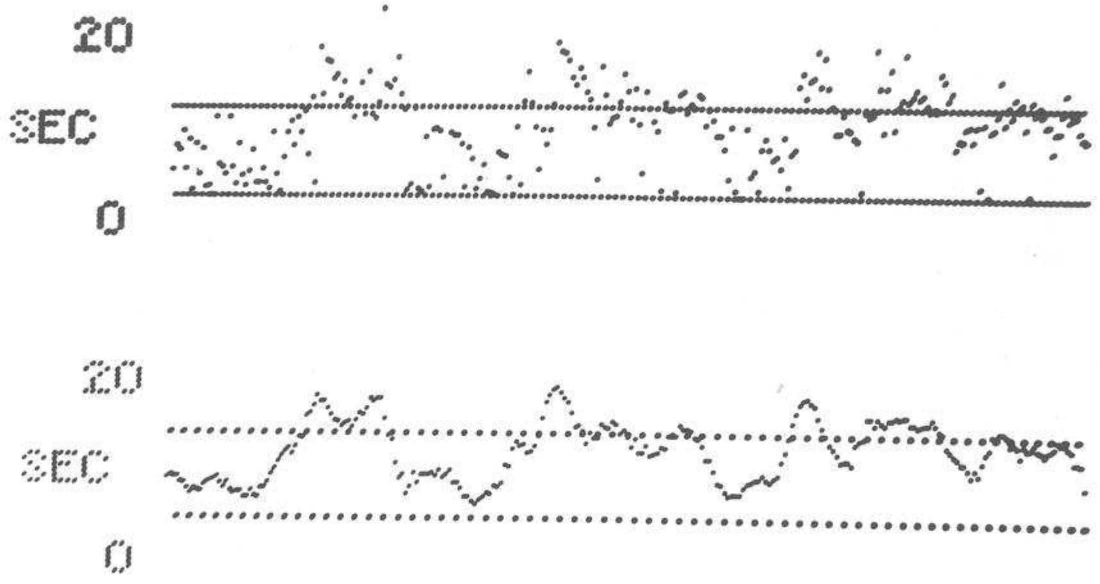


Figure 1 256 successive interresponse times, Session 8, Monkey M11, DRL 20. Duration is given by height. The upper part of the figure shows the raw data, the lower part shows the data after smoothing by a 3-term moving average (from Weiss et al, 1956).

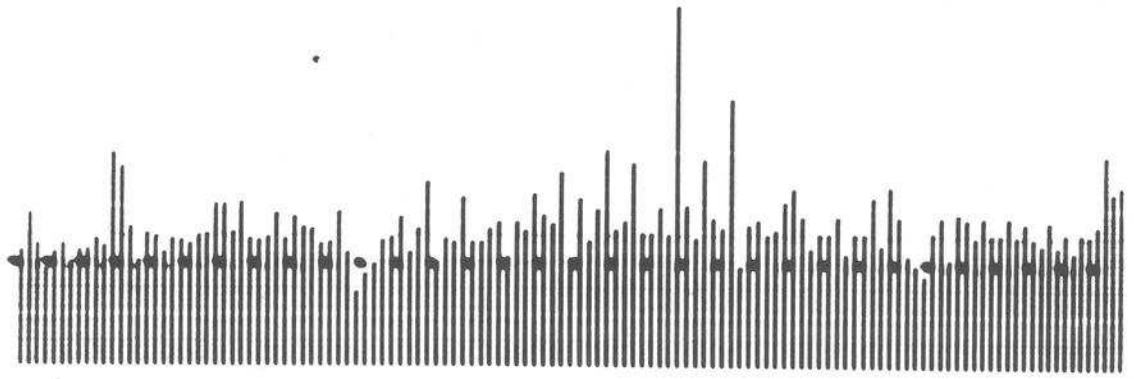


Figure 2 128 successive IRT's for Monkey M11, DRL 20, Session 30. Height is equivalent to duration. The dotted line is equivalent to an interval of 20 seconds (from Weiss et al, 1956).

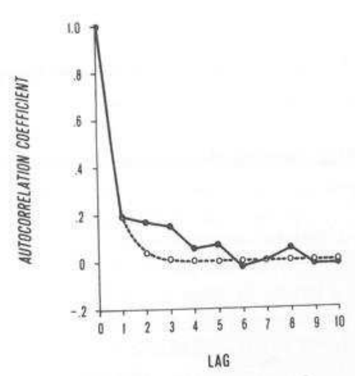


Figure 3 Autocorrelogram for Monkey M11, DRL 20, Sessions 28, 29, 30 and 31. The solid lines depict the empirical autocorrelations. The dashed line is the function that would have been obtained with a Markov process (from Weiss et al, 1956).

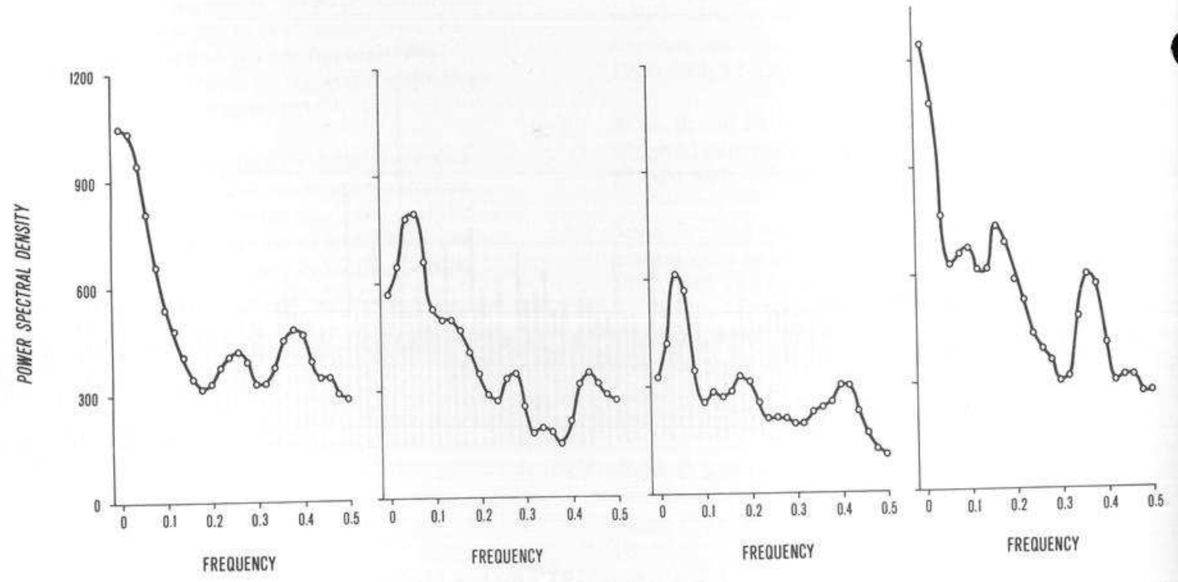


Figure 4 Power spectra corresponding to the autocorrelation plots of Figure 3 (from Weiss et al, 1966).

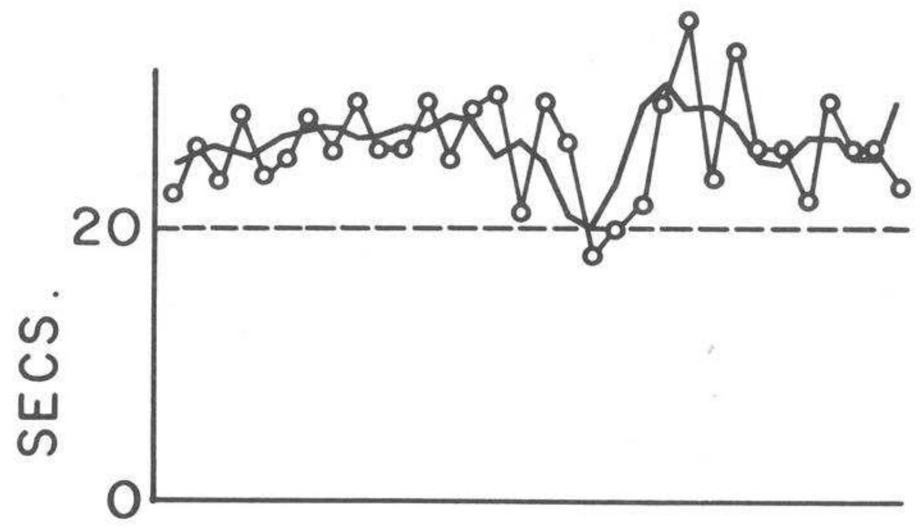


Figure 5 Successive interresponse times (open circles) compared to a 3-term moving average (solid, heavy line). This is a portion of the data from Session 30, Monkey M11, DRL 20. Note the strong tendency toward alternation in the duration of the interresponse time.

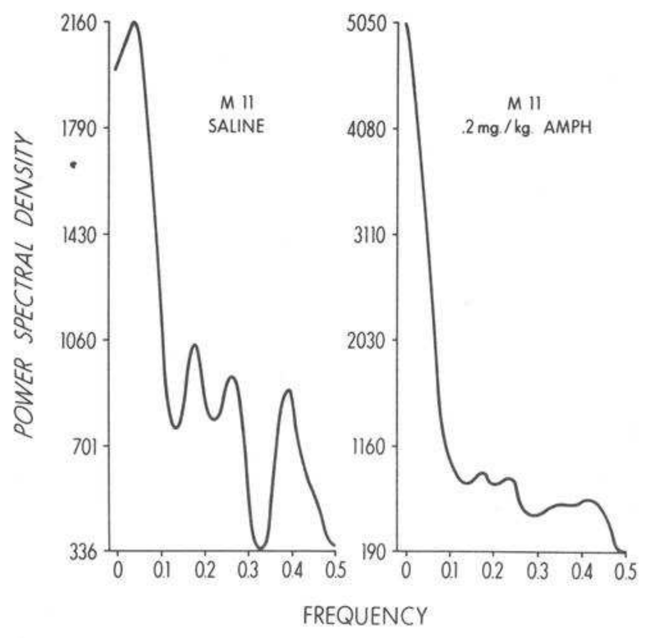


Figure 6 Power spectra comparing performance on DRL 20, Monkey M11, after saline and after 0.2 mg/kg amphetamine sulfate (from Weiss and Laties, 1967).

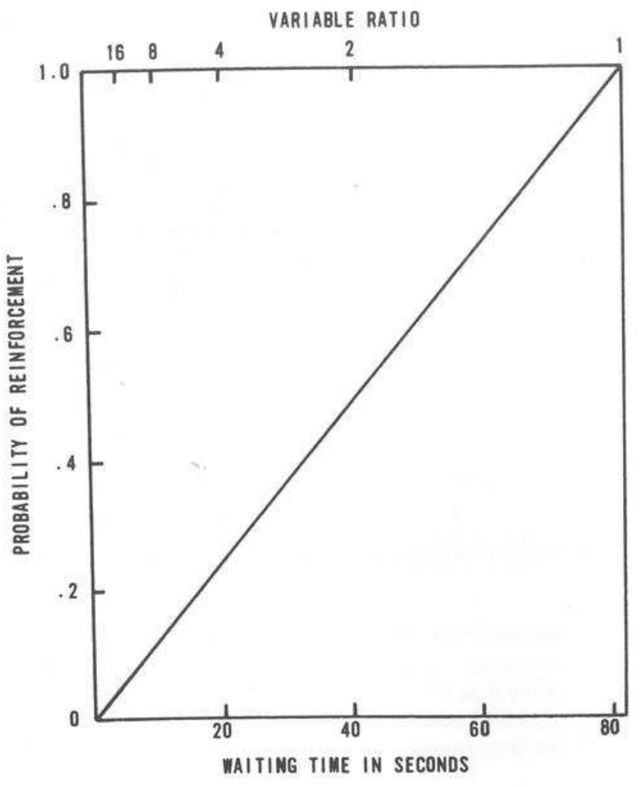


Figure 7 The linear SRW function.

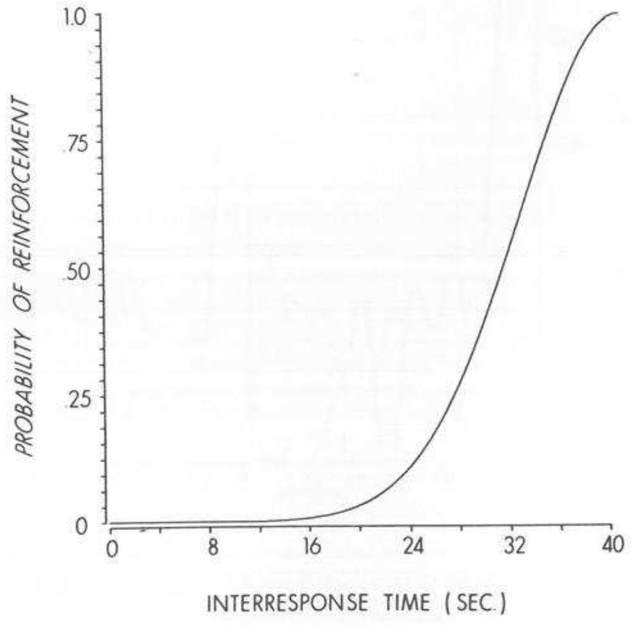


Figure 8 The Gaussian SRW function.

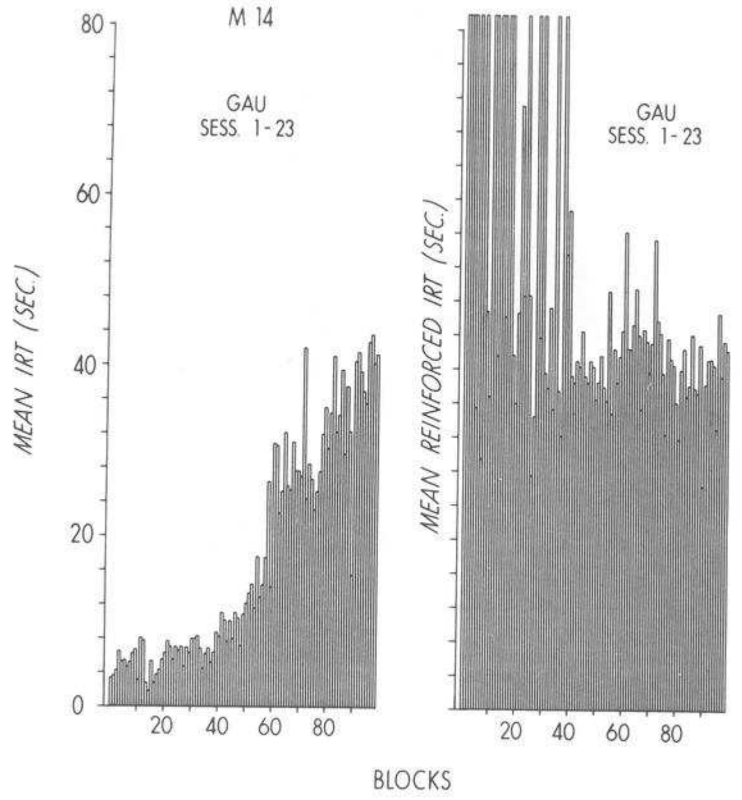


Figure 9 Performance for the first 23 sessions on the Gaussian function. Left = mean IRT in seconds, for each block of 100 IRT's. Right = mean reinforced IRT in seconds for each block of 100 IRT's.

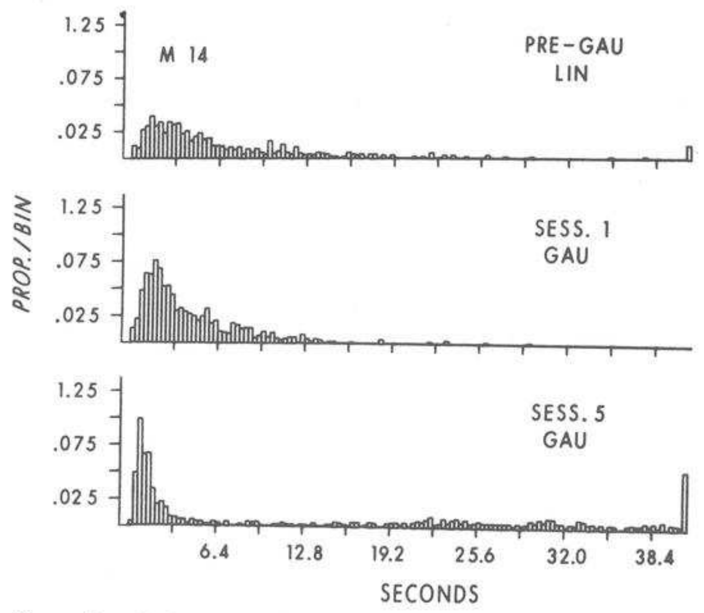


Figure 10 Performance on the SRW linear function, compared with Session 1 and Session 5, after the change to the Gaussian function. These are interval histograms showing the proportion of interresponse times in each of the different time bins.

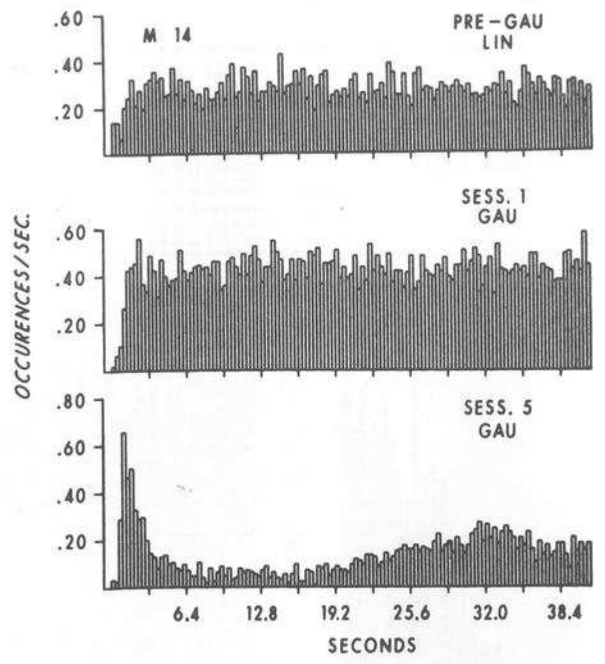


Figure 11 Expectation density functions for the SRW linear functions and Sessions 1 and 5 after the change to the Gaussian function.

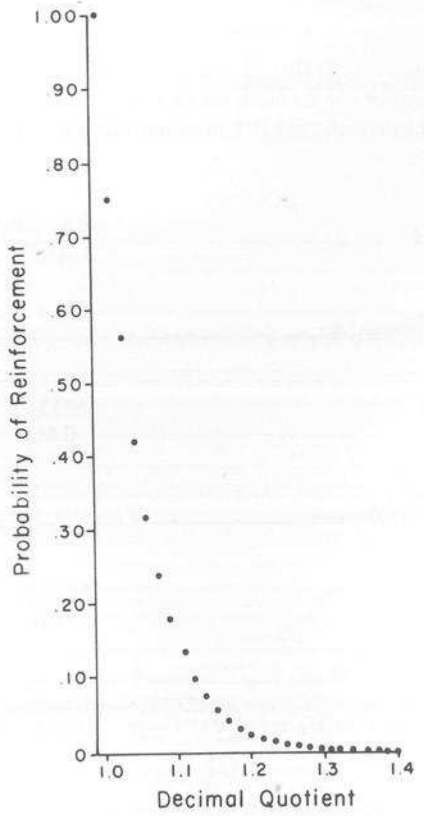


Figure 12 The autoregressive probability function. Abscissa refers to the quotient of two successive interresponse times.

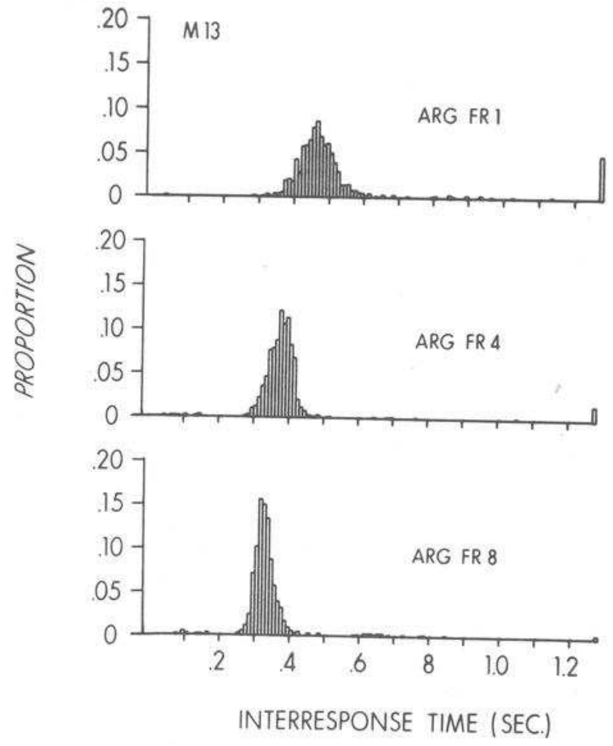


Figure 13 Interval histograms for three conditions of the autoregressive schedule, FR1, FR4, and FR8.

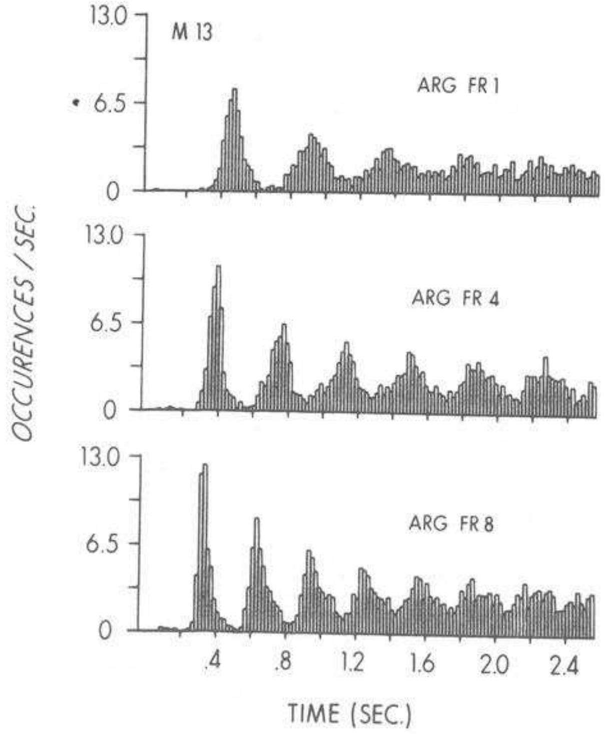


Figure 14 Expectation density plots for three conditions of the autoregressive reinforcement schedule, FR1, FR4, and FR8.

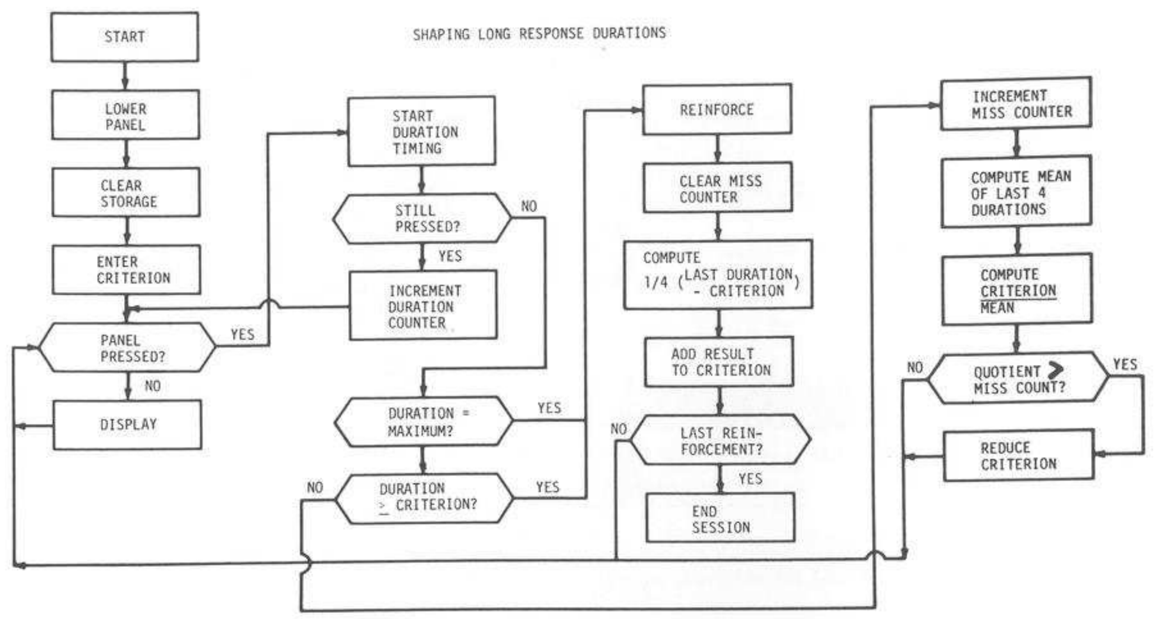


Figure 15 Flow chart of the reinforcement schedule which differentially reinforces long interresponse times in dogs.

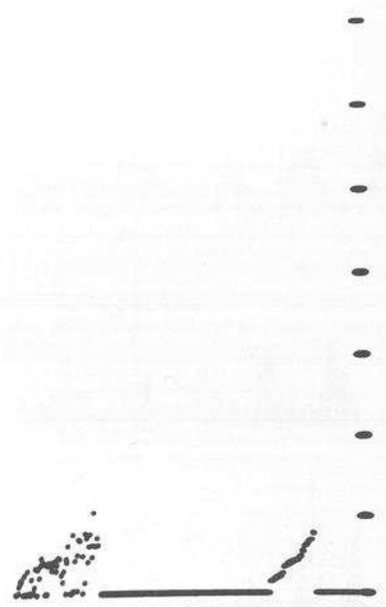


Figure 16 LINC oscilloscope display during an experimental session with the dog shaping program. The cluster of dots at the left hand side of the display shows successive response durations. The cluster at the right shows the rising criterion. The numbers, from top to bottom, refer to the number of reinforcements still available, the last response duration, and, at the bottom, the current criterion.

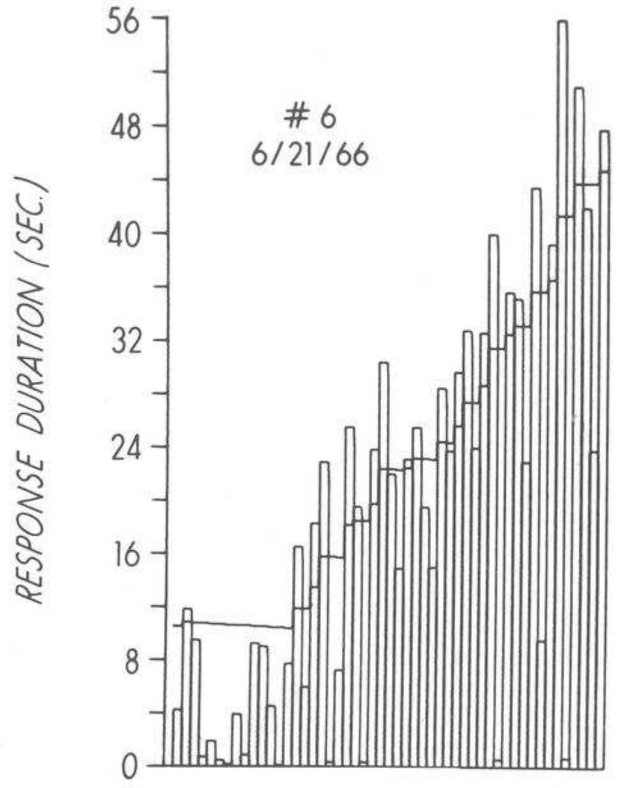


Figure 17 Plot showing the rise in response duration through a session for Dog No. 6. The bars refer to successive response durations, the line through them refers to the criterion.

Figure 18 The LINC in use as a tool for teaching computer technology.

APPLICATION OF THE LINC COMPUTER TO OPERANT CONDITIONING

C. Alan Boneau
Duke University
Temporarily Studying at Stanford University
Stanford, California

ABSTRACT

Operant conditioning is a procedure in which selected behavior of individual organisms is controlled by a judicious choice of reward contingencies. This paper describes some of the work of the three psychologists in the LINC Evaluation Program and how they adapted LINC to their operant conditioning projects. The use of the computer in dealing with temporal response variables is highlighted.

Since LINC is relatively new and unfamiliar, I will begin by giving a brief description of LINC and an account of its history.

LINC is an acronym for Laboratory Instrument Computer, a fact which should imply a good bit about the design philosophy underlying it. LINC is a small computer with a memory of 2,048 12-bit words and an 8–μsec cycle time. It has various features which make it particularly appropriate for use in a laboratory.

The LINC concept evolved from the experience of several years interaction between members of the Digital Computer Group of MIT Lincoln Laboratory and the Communication Biophysics Group of MIT's Laboratory of Electronics. Some of these people were computer designers and engineers; others were essentially biologists. It was apparent to all that the existing general-purpose computers had deficiencies when used in the context of a biological or medical research program. In such programs problems arise when handling large quantities of data from various sources, particularly when one is in the process of exploring various ways of dealing with the data. There are problems of flexibility caused by interfacing with a variety of inputs and outputs for controlling apparatus and taking records. Existing machines were too large and emphasized those design features which exploited their ability to handle well-defined problems with circumscribed input and output modes. LINC was designed to overcome these deficiencies. LINC was also designed with the hope that access to a computer would stimulate development of new approaches for the biologists, approaches which previously had been in the real sense unthinkable.

The concept of LINC thus was of a small, low-cost computer which would be a permanent but mobile fixture in a biomedical laboratory. It would have a variety of input and output possibilities and be sophisticated enough to deal with the complexities of experiment control and to process data and make routine calculations. It should be convenient, flexible, and reliable enough so that it would not require constant maintenance. It should be simple enough so that routine maintenance could be performed by the investigator himself, or at least by someone in his laboratory.

The realized LINC consists of a mobile main frame which plugs into an ordinary 110-volt socket. The console is four separate units connected to the main frame individually by a set of 30-ft cables. One of the units is the usual control console. Another unit contains an oscilloscope with programmable display; a third is the LINC-tape unit, a two -unit addressable magnetic tape device, very similar to the DECtape units. The fourth unit is called the data terminal box. This is the main channel for tying external equipment to LINC. It contains two large plug-in units into which can be wired appropriate logic and buffers and interfacing using commercial plug-in units. A variety of timing pulses and registers from the main frame are delivered to the data terminal box through cables so that supplementary functions and connections are relatively easy, and in fact are encouraged. The data terminal box is also the input terminal for 8 of 16 analog input channels, the LINC being designed to act as an A-D converter with a sampling speed as small at 32 μsec between samples. Also in the data terminal box are six relays which display the contents of a programmable register in the main frame. Another feature of the data terminal box is a set of 16 sense lines connectable to external switches. These can be examined on a command. A Teletype keyboard and printer provides routine communication with LINC. LINC was designed by members of the MIT group consisting of among others,Wesley Clark, Charles Molnar, Mary Allen Wilkes, and William Simon. A prototype was completed by the group in 1962, and, on the basis of its performance, funds were sought from the Public Health Service for a program which would evaluate the computer and provide a little cloud-seeding by getting computers into the hands of a small group of investigators. Funds were provided, and the Center Development Office for Computer Applications in the Biomedical Sciences was established at MIT to administer the project. The first twenty LINCs were subcontracted by CDO.

Early in 1963, an announcement was circulated requesting proposals from investigators in biomedical fields for the use of a small-scale digital computer to be used in their laboratories. On the basis of the submitted proposals, the Evaluation Board selected twelve investigators to participate in the LINC Evaluation Program. At issue was the validity of the philosophy underlying the design: would the computer be useful in a biomedical research setting; and could it function in the way which was intended? The investigators were largely from physiological fields operating within a medical center and interested in such problems as cardiovascular functioning and neurophysiology, but there were a few others including three psychologists, myself included. We three were working in the general area of operant conditioning, which can be loosely defined as the use of judicious patterns of rewards and punishments to control otherwise spontaneous behavior. Later I will briefly discuss some of the things done by the three of us using LINC.

Each participant in the evaluation program was required to attend a four-week-long training period at Cambridge in late summer of 1963, during which he completed the assembly of his machine. In addition, this period was devoted to learning programming and operation of LINC as well as to a detailed analysis of design. The purpose was to provide an adequate enough background so that the investigator could perform routine maintenance tasks and, most importantly for the evaluation program, so that he could understand the computer well enough to make the necessary interlinkage with other laboratory equipment.

As one of the participants, I must give a personal reaction to this phase of the program. I never realized that so much could be crammed into 100 hours a week. Like the other two psychologists, I spent the four weeks at Cambridge by myself. All the other participants had brought with them a technical person from their staff. This fact is not so much a statement about the work and social habits of psychologists as it is about the size of their budgets and the relative magnitude of their projects. We simply had no one to bring with us. This meant, of course, that the dirty work involved in adapting the machine to our own use had to be done, in the face of academic schedules and commitments with no technical assistance. I feel it is a testimony to the elegance of the LINC concept and design and to the excellence of the training we received that all three of us within a year of receiving the machine had managed to incorporate it into our research in a major way, in spite of the handicaps and lack of background we shared. Most early utilizations we made of LINC occurred in the context of things we were already doing, but there emerged some difference in our plan of attack as a result of LINC. We did things that never would have occurred to us had LINC not been available, and we are now beginning to do things that could be done in no other way but by on-line computer control. With a little more time to relax with the machine, I'm sure we will do more.

At the time of my initial acquaintance with LINC, my students and I were engaged in a project of teaching pigeons to discriminate colors so that we might observe the effect of various factors on their decision to peck or not to peck. In the course of the training, a pigeon was placed in a small dark box with a hole on one side through which he could peck a sheet of translucent plastic. The plastic was lighted from behind by means of a monochromator. On a sequence of trials the pigeon was rewarded for pecking to some wavelengths and not to others. While initially not very good at this task, the pigeon was ultimately coaxed into making a sharp discrimination between wavelengths as close as one millimicron. I say ultimately because such training typically required several hundred hours.

We had originally worked on this problem in a rather relaxed way, exposing each color for 30 seconds and counting the number of pecks which occurred during the period. Some of the wavelengths were never rewarded, while the others were rewarded on what is called a variable interval schedule. In this schedule, the apparatus is set to deliver a reward for the first peck after a variable interval of time, averaging perhaps every 20 seconds. On this regimen, a bird will respond to rewarded colors at a relatively high rate, and will respond to unrewarded colors at a very low rate (or never) except when the discrimination is difficult, in which case, the rate is some intermediate value.

There are some disadvantages to a rate measure for our purposes, and I will return to this problem later. Our main theoretical concern, however, was the probability of occurrence of a response; so we decided to utilize rapid sequences of short presentations of the colors, for example, 2 seconds in duration. Then we could measure directly the probability of occurrence of a peck. To do this we made use of a servo-system to drive the monochromator fed by a signal which appeared as the output of a relay-tree decoder. After being provided with a supplementary set of 24 relays, LINC was hooked into the apparatus to furnish signals for the relay tree, to open and close shutters, provide rewards, perform all the timing functions and record the occurrences of pecks, as well as randomly scramble and rescramble the sequence of colors and decide which were to be rewarded. To keep the pigeons honest, LINC rewarded randomly about 1 peck in 20, but only to a predetermined set of wavelengths.

To get around the problem of the long training which

our birds required, we decided to run the experiments overnight. The control system had developed to the point that we felt free to leave LINC in charge without monitoring. Surprisingly enough this worked very well. LINC proved to be remarkably reliable. We have no solid evidence that LINC made a single machine error in the first 6,000 hours of operation. After that, of course, followed the complicated aftermath of an attempt to clean all the accumulated pigeon dust out of memory.

At any rate LINC was running 24 hours a day for several months, during the night acting as experimenter and during the day helping us clarify the gathering data. Since LINC had the necessary capability, we decided to record all the pertinent information on every trial for each bird, a major change in procedure possible because of LINC. This information included the wavelength, the availability of a reward, and the length of time between the onset of a trial and the occurrence of a response if there was one. In a period of a few weeks, we had generated a sizable amount of data on four birds, several million chunks of information which would not have been recorded or analyzed without LINC: it would not have occurred to us to do so.

In the course of the experiment, several effects appeared which smeared the results we were looking for. One such effect was the consequence of a rewarded trial. Our suspicions were confirmed when we had LINC skip through the mass of data, selectively picking out critical trials. We decided to assess the effect of a reward by comparing performance on trials preceding rewards with that on trials following a reward. Consequently LINC selected trials on which a reward was available and obtained, amounting to two or three trials per hundred. LINC then extracted information from the trials immediately preceding and following that one, and formed a trabulation giving the number of occurrences of each of the colors before and after a reward as well as the number of pecks to each.

For example, consider a set of data obtained from one bird. The bird received sequences of presentations of the ten wavelengths from 530 to 539 millimicrons spaced at one-millimicron intervals, all presented equally often. The five values at the left, 530 through 534 millimicrons, were never rewarded. The values 535 through 539 were occasionally rewarded.

Before occurrence of a reward the pigeon discriminated remarkably well, responding consistently to these stimuli for which a reward is sometimes forthcoming. The region of transition between responding and non-responding seems to be in the neighborhood of three millimicrons. On the trial immediately following the reward, the whole curve shifted to the left about one millimicron. Since the resulting curve is almost the same shape, I could not characterize this as a breakdown of discrimination, although the effect is clearly an increase in responding to non-rewarded stimuli. Since I am discussing applications of the LINC, I do not intend at this time to get involved in a discussion of what this phenomenon might mean, except to remark that it is consistent with the notion that the pigeon is wired to act as a statistical decision maker.

Unfortunately, although this procedure results in a necessary condition for random emission, it is not sufficient. The pigeon tended to track the reward timing and to give a majority of responses around that inter-response time which currently was being rewarded. This of course produced problems.

This procedure is an example of LINC's ability to produce a very complex reward schedule which is contingent upon the behavior of the bird and which necessarily has to be handled in real time. Let me discuss another such example this time by the third of the LINC psychologists, Dr. Bernard Weiss, now at the University of Rochester School of Medicine.

I mentioned earlier that rate of responding to an extended presentation of the stimulus might be an alternative to probability of response as a measure of what is going on in experiments of this type. This rate measure has been analyzed by Don Blough of Brown University, another of the psychologists of the LINC Evaluation Program. In a situation somewhat similar to that which I have just discussed, he had LINC look at the interval of time between successive responses, the inter-response time or IRT. These were tabulated by wavelengths for different IRT's. Dr. Blough found that shorter IRT's were less dependent upon the stimulus than were the longer. It seemed to be the case that once the pigeon started pecking, the probability of his making another response quickly was more a function of whether he had just made a response than it was of which stimulus was present. These sequential dependencies turned out to be a function of motivation and the time since the last reward, among other things. This tended to confound any underlying relationships.

The way out of these difficulties would seem to be a training regimen which would tend to do away with the sequential dependencies and make the pigeon approximate an ideal random emitter. One of the characteristics of such an emitter would be that the distribution of inter-response times for each stimulus would be exponential. Dr. Blough attempted to produce such a distribution by a selective reward. He divided inter-response time into bins so that if the same number of pecks were in each bin, the resulting distribution of pecks would be exponential in form. LINC was programmed to monitor IRT's on-line, categorizing pecks into the appropriate IRT bins. On each peck it looked at all IRT bins and rewarded the pigeon if the IRT for that peck occurred in that bin with the smallest number of pecks. Since the pigeon is sensitive to all sorts of manipulations of reward contingencies, the tendency to respond to the least frequent inter-response time should be increased.

This procedure is an example of LINC's ability to produce a very complex reward schedule which is contingent upon the behavior of the bird and which necessarily has to be handled in real time. Let me discuss another such example this time by the third of the LINC psychologists, Dr. Bernard Weiss, now at the University of Rochester School of Medicine.

Among other things, Dr. Weiss has been interested in the effect of drugs on timing functions in monkeys. One of his projects was to devise a reward schedule which would take as much variability as possible out of the inter-response time by differentially rewarding low variability. His approach is what he called an 'autoregressive' schedule. Operating on-line, LINC was programmed to take the quotient of the last two inter-response times and then reinforce the monkey according to a schedule. If the quotient is close to one, indicating that the last two IRT's are nearly identical, the probability of a reward is high. The greater the difference between the last two IRT's, the more the quotient deviates from unity and the smaller the probability of reward. The actual decision to reward or not was based upon the value of a random number which was generated by LINC at this time.

In another kind of analysis of the data, Dr. Weiss has had LINC plot successive inter-response times in the form of an expectation density plot. This plot is formed by letting successive instances of an event, a peck, take the value T_0. Succeeding events are plotted as deviations in time from this moving reference.

SUMMARY

This paper presents specific applications of LINC to a circumscribed area of psychology. These applications all involve control of an experiment in which LINC is an integral part of the process, operating various pieces of equipment, sensing external events, making decisions in real time according to a schedule of contingencies, and recording data.

LINC has made it possible to record raw data extensively to be analyzed as the inspiration leads rather than merely record preselected synopses of the behavior. From my own experience this has meant that the data which formerly had to be extracted from a new experiment was often already available. Thus, LINC has furnished the opportunity to explore the possibilities of data analysis in depth.

LINC has also made it possible to program complicated interactions of the organism with its environment. This means that we need no longer conceive of the environment as something passive to be acted upon by the organism; it can be programmed to fight back. Possibilities implied by this are yet to be realized.

A REAL-TIME SYSTEM FOR OPERANT CONDITIONING USING THE PDP-8

Robert H. Tedford
Union Carbide Corporation
White Plains, New York

ABSTRACT

This program controls the operations of ten standard environmental chambers using three basic experimental systems; Punishment Discrimination (PD), Non-Discrimination Avoidance (NDA) and Food Reinforcement (FR/FI). Besides controlling the experiments, certain statistics are accumulated during the experiments for print-out at the end of each test run.

Introduction

This paper describes a real-time program for controlling behavioral experiments utilizing a Digital Equipment Corporation PDP-8 Computer. Besides the 4096, 12-bit, fixed-word length computer, our system (see Figure 1) is composed of ten standard sound insulated environmental chambers. Each box contains one lever, a food receptacle and a grid floor through which brief electric shocks may be delivered. A grid scrambler prevents positional avoidance behavior. Solid state cumulative recorders are used to provide continuous visual representation of the animal's performance. An interface connects the environmental chambers, recording equipment and computer. In this system, the computer is capable of recording lever presses, delivering shocks and/or food reinforcements, and controlling houselights, speakers and stimulus lamps in accordance with a particular pre-programmed schedule. The original program was written for three basic experimental systems; Punishment Discrimination (PD), Non-Discrimination Avoidance (NDA), and Food Reinforcement (FR/FI).

Procedure

The program actually operates through a series of interrupts. These interrupts are initiated by the interface (clock), the animals (lever), the operator (keyboard), and the program (print). The clock in the interface circuitry may be adjusted to any desired setting. We have ours set to interrupt the main program every tenth of a second. A lever interrupt occurs when the animal has made a complete response. A press and release of the lever constitutes one response. The interrupts are serviced in order of priority; clock, lever, keyboard, and print. The clock and lever interrupts can occur during teletype operations, but they cannot interrupt each other.

When an interrupt occurs, the program tests the interrupt flags to determine which type has occurred. The program must then identify which box or boxes require servicing at this time. For this purpose, the clock, lever and print interrupts have each been assigned a 12-bit word. Each environmental box is identified with one bit in each word. Boxes will have their bits turned on (set to a one) when they require servicing. The lever interrupt bits are set by the interface hardware circuitry in the lever buffer register. When a lever interrupt occurs, the contents of the lever buffer register are read into the computer. The register is cleared to zeros by the read operation. For clock interrupts, the bits are set by the program after the first lever press is made. These bits are turned off when the test duration has elapsed. When a test is terminated in any of the boxes, its bit is set in the print service word. This bit is reset after the print-out is complete. The method of identifying which box or boxes are to be serviced for a given interrupt makes use of the link register. This one-bit register may be rotated right or left with the 12-bit accumulator. Under program control, the link is reset to zero and the accumulator is loaded with the interrupt service word. Then by successive shifting of the accumulator word into the link and testing the link for a one-bit, the boxes to be serviced are identified. If the link contains a zero after shifting, the box is bypassed. This process continues until all boxes requiring service have been identified and serviced.

Keyboard interrupts are handled by a different routine. Each box has been assigned a letter from A to J. When a key is struck by the operator, the octal code for that key is loaded into the accumulator. Through a series of subtractions, the program is able to determine which letter has been struck and thereby which box is being addressed by the operator. Letters were used to identify the boxes instead of numbers, so that only one character had to be struck for each box. This principle would hold true even if the system were expanded to include twelve or even twenty-four boxes. After the box letter has been keyed in to the computer, one to four additional characters are needed to identify the particular schedule desired for that box. The specific calling procedures are shown in Figure 2.

Schedules

Under the PD system, we have the capability of calling any of the following schedules; FR1, VI2, a combination of VI2 and FR1 with tone, and the last mentioned with shock and tone during the FR1 phase. The first two schedules have a one hour duration and the latter two have 90-minute durations. The schedules represent three training phases which lead up to the criterion phase of the punishment discrimination system.

In the NDA system, we have the flexibility of varying the response to shock time (RS), the shock to shock time (SS), and the test duration. The RS and SS times can be varied from 1 to 99 seconds. The initial test duration is 3-1/2 hours. This may be extended in blocks of sixty minutes, if desired, using the same times as for the initial period.

The FR/FI system has the capability of calling for either a schedule with a Fixed Ratio (FR) of 1 to 99 or a schedule which has alternately a time-in (varied from 1 to 9 minutes) and a time-out (varied from 0 to 9 minutes). During the time-in, the animal is reinforced in the same manner as a straight FR schedule. He is not fed during the time-out period. However, he may receive a reinforcement after the time-out period if he has responded in the last 10 seconds. The stimulus lamp is on during the time-in period and a constant tone is on during the time-out period. Both of these schedules have one-hour durations.

Each box may be assigned one of two systems. The PD system is available to boxes A, B and C. The FR/FI system has been written for the remaining boxes, D through J. The NDA system which requires only the shock equipment, can be utilized by all the boxes.

Sequence of Events

The operator enters the box call via the keyboard. The box call establishes the schedule to be used in this experiment. The keyboard service routine performs these additional functions; the box is activated to lever interrupts, the statistics gathering area is prepared for a new experiment, and the duration timers are set to their proper values.

The cumulative recorder is reset and labeled with the proper identification for this experiment. The animal is placed in the called box and the box is closed. Using the external button, the operator may make the first lever press. This button can also be used when shaping a new animal.

The first lever press service routine is handled differently from succeeding lever interrupts. This special routine sets the clock bit for this box in the clock service word, and causes the house light and any stimuli to be turned on. Indirectly, since the recorders are wired into the house light bits, the cumulative recorder is turned on.

For the duration of the experiment, clock and lever interrupts are serviced according to the particular schedule. Individual box parameters are made available to the general routines for servicing these interrupts. The routines provide for incrementing statistical counters and duration times. Additional calculations are performed at predetermined intervals. Decisions are made as to when an animal should be reinforced or punished and causes them to be delivered to the box. Changes in stimuli may be effected under program control to denote a marked change in the schedule.

When the experiment duration time has elapsed, the following events take place as part of the clock service routine. The clock bit is reset to zero and the print bit is turned on in their respective service words. The box is inactivated to lever interrupts. The house light and all stimuli in the box are turned off. Simultaneously, the cumulative recorder is stopped. Finally, the print interrupt is initiated by the program.

When an interrupt occurs, the interrupt circuitry of the computer is automatically disengaged. It remains disengaged until a programmed instruction reactivates the circuitry. For clock and lever interrupts, this takes place at the conclusion of servicing the particular interrupt. When a print interrupt occurs, the data to be printed is first dumped into a reserve area and then the interrupt system is engaged. The print-out proceeds with interruptions by clock and levers from other boxes taking priority over the print-out. When the print-out is complete, the print bit is reset and the box is ready for a new experiment.

Print-Out Nomenclature (see Figures 3, 4 and 5)

SG - Segment Number; in the PD and FR/FI systems each segment is 15 minutes long, while in the NDA system they are 30 minutes long.

RT - Number of Responses made during each segment.

SR - Number of Rewards received during each segment; for the NDA system, this would be a negative reward.

FREQ - Frequency in responses per minute.

DSQ - This is a statistic used in comparing an animal's performance from day to day. Numerically it is calculated by the following equation:

$$DSQ = \Sigma X^2 - (\Sigma X)(FREQ)$$

where X = number of responses made in each minute.

SSQ - Variance associated with the frequency or the standard deviation squared. Numerically it is equal to 'DSQ' divided by the number of observations - 1.

FREQ, DSQ, and SSQ is calculated hourly in NDA system, for every hour after the first 30-minute segment. Half-hour calculations are made in the FR/FI system.

Summary

This program, as described, takes up all but about 250 locations of the available core storage. We were able to provide most of the versatility that was spelled out in the original goals for the system. There were a few compromises made in the area of statistics in the interest of time and space. While this system was written for ten behavioral boxes, it is conceivable that two more boxes could be added without sacrificing the present versatility. Beyond that point, some of the flexibility of this system would have to be removed before additional boxes could be put on-line.

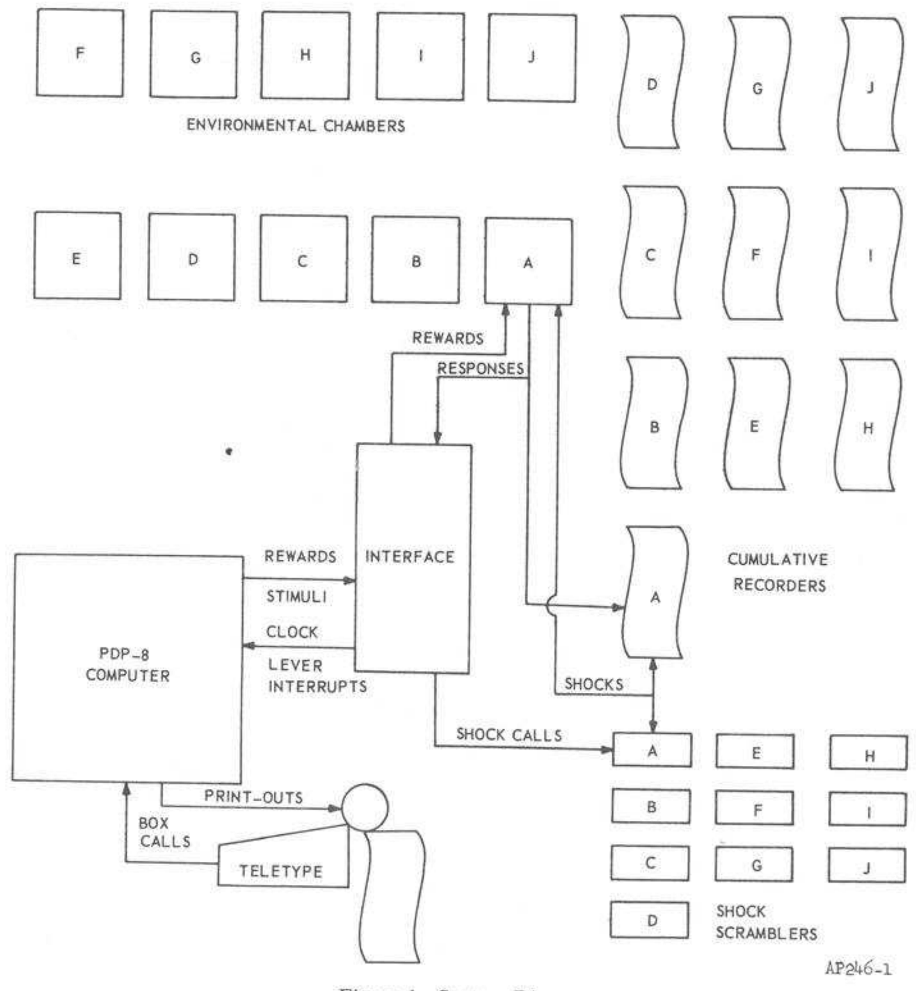


Figure 1 System Diagram

PD SYSTEM

Schedule	Call*	Duration	Variables
FR1	/	1 hour	None
VI2	Ø	1 hour	None
VI2 + FR1 w/T	1	1½ hours	None
VI2 + FR1 w/T + S	2	1½ hours	None

NDA SYSTEM

Schedule	Call*	Duration	Variables
RS 40/ SS 20	4Ø2Ø	3½ hours	RS & SS Times
Extention	X	1 hour	None

FR/FI SYSTEM

Schedule	Call*	Duration	Variables
FR15	15	1 hour	Fixed Ratio
FR15/FI21	1521	1 hour	Fixed Ratio, Time–in Interval, Time Out Interval

*All calls shown would be preceded by the Box Letter.

AP246-2

Figure 2 Box Schedule Summary

Call: A/

```
FR1, BOX A

SG          1      2      3      4      5      6
RT        125    132     47     23
SR        125    132     47     23
```

Call: AØ

```
VI2, BOX A

SG          1      2      3      4      5      6
RT        253    264    219    188
SR          7      8      7      7
```

Call: A1

```
VI2 + FR1 + T  BOX A

SG          1      2      3      4      5      6
VI/RT     165    156    172    147    161    152
VI/SR       6      5      7      6      7      5
FR/RT      23     25      9     15     11      8
```

Call: A2

```
VI2 + FR1 + T  ,S BOX A

SG          1      2      3      4      5      6
VI/RT     149    126    135    142    156    175
VI/SR       6      5      6      7      5      6
FR/RT       4      2      1      3      2      2
```

AP246-3

Figure 3 PD System Sample Print-Outs

Call: D4Ø2Ø

```
NDA  4Ø / 2Ø  BOX D

SG       1      2     33      4      5      6      7
RT      3Ø     4Ø     4Ø     4Ø     4Ø     4Ø     74
SR      75     75     75     76     76     76     68

CUM DATA      FREQ       D SQ        S SQ
Ø3Ø–Ø9Ø       1.333      393.339     6.66Ø
Ø9Ø–15Ø       1.333      393.339     6.66Ø
15Ø–21Ø       1.899      851.41Ø    14.423
```

Call: E4Ø2Ø

```
NDA  4Ø / 2Ø  BOX E

SG       1      2      3      4      5      6      7
RT      3Ø     4Ø     4Ø     4Ø     4Ø     4Ø     51
SR      76     75     75     76     76     75     7Ø

CUM DATA      FREQ       D SQ        S SQ
Ø3Ø–Ø9Ø       1.333      393.339     6.66Ø
Ø9Ø–15Ø       1.333      337.339     5.711
15Ø–21Ø       1.516      466.988     7.897
```

Call: DD

```
NDA  4Ø / 2Ø  BOX D

SG       1      2      3      4      5      6      7
RT                                           4Ø     4Ø
SR                                           78     78

CUM DATA      FREQ       D SQ        S SQ
Ø3Ø–Ø9Ø        .ØØØ        .ØØØ        .ØØØ
090–15Ø        .ØØØ        .ØØØ        .ØØØ
15Ø–21Ø       1.333      693.339     11.745
```

Call: EE

```
NDA  4Ø / 2Ø  BOX E

SG       1      2      3      4      5      6      7
RT                                           4Ø     4Ø
SR                                           78     78

CUM DATA      FREQ       D SQ        S SQ
Ø3Ø–Ø9Ø        .ØØØ        .ØØØ       .ØØØ
Ø9Ø–15Ø        .ØØØ        .ØØØ       .ØØØ
15Ø–21Ø       1.333      693.339    11.745
```

AP246-4

Figure 4 NDA System Sample Print-Outs

Call: H15

```
FR  15, BOX H

SG        1       2       3       4
RT       25      25      25      25
SR        1       2       2       1

CUM DATA      FREQ      D SQ       S SQ
ØØØ–Ø3Ø       1.666     166.674    5.723
Ø3Ø–Ø6Ø       1.666     166.674    5.723
```

Call: HØ521

```
FR/FI

FR  5, BOX H

SG        1       2       3       4
RT       25      25      25      25
SR        5       5       5       5

CUM DATA      FREQ      D SQ       S SQ
ØØØ–Ø3Ø       2.5ØØ     125.ØØØ    6.579
Ø3Ø–Ø6Ø       2.5ØØ     125.ØØØ    6.579

FI  21, BOX H

SG        1       2       3       4
RT       25      25      25      25
SR        1       1       1       1

CUM DATA      FREQ      D SQ       S SQ
ØØØ–Ø3Ø       5.ØØØ       .ØØØ      .ØØØ
Ø3Ø–Ø6Ø       5.ØØØ       .ØØØ      .ØØØ
```

AP246-5

Figure 5 FR/FI System Sample Print-Outs

A NOTATIONAL SYSTEM AND COMPUTER PROGRAM FOR BEHAVIORAL EXPERIMENTS*

Arthur G. Snapper, and Ronald M. Kadden
FDR Veterans Administration Hospital
Montrose, New York

Julius Z. Knapp, and Harold K. Kushner
Research Laboratories
Schering Corporation
Bloomfield, New Jersey

ABSTRACT

A computer program for on-line control of psychological experiments is discussed. Specific scheduling and recording requirements of a particular behavioral experiment are described in a series of state diagrams, and then entered as a constant table into the computer. Ten such constant tables for programmed experiments may be entered, and run simultaneously. A single supervisory program operates on the 10 independent constant tables and controls events in 10 experimental chambers via three input and several output words. Data is stored in memory or dumped using a high-speed punch.

The system described in this report was designed to provide both on-line control of, and data acquisition from, 10 simultaneous operant experiments. The main features of the system are:

(1) a notational language for describing any behavioral procedure, employing units called "states",

(2) a compiler for use with the PDP-8, designed to translate the notated procedure into a binary constant table,

(3) an operating program that uses the binary constant table to generate the desired experimental contingencies between stimuli and responses, as well as to record data generated in the experiment.

Figure 1 shows the steps necessary to program the PDP-8 for an experiment, using the present system. Once the experimental question is formulated, the desired relationships between stimuli and responses can be expressed in terms of a state diagram, the notation system used here. The state diagram is then entered into the PDP-8 compiler program on the ASR33 teletypewriter. (If desired, the typed input can be saved for future compilations in paper tape format.) The compiler we have written then generates a binary tape of the constant table representing the state diagram that has been entered. Once the binary tape has been obtained, it is entered into the computer under the control of the operating program, SKED. The latter program utilizes the binary constant table to accept and record the responses of the experimental subject and to deliver particular stimuli to the subject in the experimental environment, through our interface and associated stimulus generators.

Data, including the number of responses, temporal distributions of responses, etc., are also obtained during the experimental session and can be printed out at any time.

The notation system used in our laboratory is based upon the mathematical theory of sequential switching circuits (reference 1, 2,). In particular, we are using the merged model proposed by McCluskey (reference 3). We have chosen this notational language for three primary reasons:

(1) it is precise enough for designing electronic switching circuits,

(2) it is simple enough to be easily learned, and

(3) it is already used by engineers and scientists to describe a wide variety of sequential devices and procedures.

Other notational languages already used in psychology (reference 4, 5) might be used in a similar fashion if they were revised to eliminate any ambiguities.

To explain the state notation system, an example of a typical reinforcement schedule (DRL) is shown in Figure 2. In this diagram there are two separate state sets. The upper portion describes the reinforcement schedule proper.

* Supported in part by NIMH Grant MH 13049, awarded to Prof. W. N. Schoenfeld, Queens College, CUNY.

The lower diagram represents an associated recording state set designed to obtain distributions of time elapsing between responses.

An important property of states as used here, demands that only one state within a state set be in effect at any one time. To allow several different reinforcement or recording states to be in effect at the same time, it is necessary to factor these concurrent contingencies into single states of several different but concurrent state sets. A second property is that each state set includes a finite number of states. Most behavioral procedures in current use possess this property.

Inputs of this system are responses of the subject (i.e., R) and time values. Outputs are of three types:

(1) Relay outputs that generate stimuli (i.e., SR ON that refers to the onset of the reinforcement stimulus)

(2) Counter pulses (i.e., C1) that refer to a stored count of the number of particular inputs that occur during a state or on transition between states

(3) Z outputs from one state set that serve as inputs to other state sets to provide inter-communication between state sets.

Each state may have associated with it a set of counters used to count up to a predetermined number of inputs before state transition occurs. In Figure 2, there is such a counter associated with the 10" time interval that causes S1 to change to S2. Since we use a basic timing pulse of 10 msecs, a count of 1000 such pulses must occur to cause the transition between states. Any state transition resets all pre-set counter accumulations to zero so that upon re-entry to the state, a full count is again required to change states. Figure 2 shows that a response in S1 of the upper state set, resets the 10" interval associated with that state, by zeroing the count of the 10 msec pulses.

When a state has two transitions leading to different next states, the first input to occur takes precedence. For example, in Figure 2 if the response in S1 occurs first, then the state is re-entered and the 10" time counter is reset. Alternatively, if 10" elapses without a response, transition to S2 occurs. Another rule of the state system is that only one transition can occur at a time, so that indeterminacy of path or critical races cannot take place. The interrupt and the interface system we use guarantee this feature.

A brief description of the reinforcement schedule notated in the upper portion of Figure 2 can help to clarify features of the system. The schedule, known as DRL, requires the subject to wait at least as long as a specified time (in this case 10") before a response will obtain a reinforcing stimulus (in this case 4" access to food). If responses occur more closely spaced in time than the 10" period, the subject must again wait for 10" before reinforcement is available. S1 is the timing state. If a response occurs within it, 10" must elapse for transition to S2, i.e., a response in S1 re-initializes S1. If 10" elapses after state entry without an R, there is a transition to S2. In this second state, responses turn on the reinforcement and enter S3. The duration of this latter state determines the length of the reinforcing stimulus (in this case 4"), since the timing out of the reinforcement duration causes the end of reinforcement and return to the first state. Responses in S1 are recorded in a data counter (C1), and reinforcements, or response-produced entry into S3, are recorded in data counter 2. This recording of data corresponds then to the number of unreinforced and reinforced responses. A "Z" output is also generated at the beginning of the reinforcement for use in synchronizing the second state set with the first state set.

The lower state diagram depicts the program for recording IRT's or interresponse-time distributions. This commonly used measure is a distribution of the number of responses that occur in each of several intervals of time starting from the preceding response. State 0 is a dead state, entered at the beginning of the experiment and at the beginning of reinforcements, so that the IRT's will be obtained only for responses following unreinforced responses. A separate distribution could be obtained for responses following reinforcements (the post-reinforcement pause). The first response following the start of the experiment or reinforcement will cause entry into S1. If reinforcement does not occur for this response (which would cause re-entry to S0 on the Z pulse) a 2" timer begins. If the next response occurs in S1, before the 2" elapses, then it is recorded in data location C3, and S1 is re-entered for the next response. If on the other hand, the 2" interval elapses before a response occurs, then S2 is entered. Again a 2" period begins during which a response will be recorded in the next recording bin. The diagram continues in this fashion up to the final bin at which point all responses occurring after an interval greater than 10" are recorded.

The preceding state diagram was constructed by specifying the sequence of events desired for the DRL schedule and associated recording of inter-response times. The basic rule to follow when constructing a state diagram is to list the desired events (stimuli, stimulus-response contingencies, recording, etc.) in the sequence in which they are to occur. It is useful to begin by imagining putting the subject into the experimental environment at the beginning of the session and to call the stimulus complex at this point State 1. The next step is to ask at this point, what is the effect of a response. Do I wish to record it? Do I wish the response to change the stimulus situation? Do I wish the response to change the stimulus consequences of succeeding responses? A similar set of questions can then be asked concerning the effects of the passage of time without responses. When states are found to change on the occurrence

of one of the inputs, then a similar set of questions applies to the consequences of inputs in the next state.

Once a correct state diagram is constructed, then the input to our compiler can be determined. Except for a series of necessary housekeeping information to the compiler, the format is similar to the state diagram, transformed into a state listing. Figure 3 contains an actual teletype sheet of the information that was entered into the compiler to produce the constant table for the state diagram of Figure 2.

When the compiler is started it prints out the phrase "BOX #" and the experimenter enters in the appropriate box number. If the same program is to be used for more than one experimental environment (up to 10 may be run by the operating program at one time), than a non-printing character is struck on the teletype before and after the box number and/or any other parameter that differs between compilations. The compiler will punch the character on the ASCII tape that reflects the dialogue with the compiler. In this way, similar programs may be entered on ASCII tape with modifications of parameters permitted by the non-printing character. In similar fashion the starting address (SA), the number of state sets, the address of the first recording counter, the IOT for a set of relays, and any relay symbolic definitions may be entered.

The compiler at this point will print out the number of the first state set, and will await typed entry of the first transition desired, in symbolic language. In the present case the desired input causing transition is 10 seconds. The compiler will then permit the number of the present and the next state to be entered, while it provides the desired format. If a recording output (as in the second state transition) is desired, it is entered by the experimenter. Relay outputs (as in the third transition) may be entered as numbers or as symbols defined previously. When the state set is completed, another non-printing character is struck, and the compiler will punch a binary constant table on the high speed punch. When the entire state diagram has been typed in, the compiler will punch a checksum on the end of the binary tape to be read either by the DEC Binary Loader, or by the operating program. Channel 8 trailer will also be punched. The constant table produced in binary format may now be read into the computer along with the operating program (SKED), at which point the schedule of reinforcement and associated recording will be in effect.

Figure 4 provides an overall flow diagram of the operation of the operating program, SKED, starting from the occurrence of an interrupt to the final resetting of the interrupt flop following the complete processing of the input. Each experimental environment or box has associated with it four inputs to the computer, three of which may represent different responses, and the 4th a free-running clock with a 10 millisecond period. When one of these inputs occurs, the interrupt flag is operated, and the computer tests for and assigns the box number and input number causing the interrupt. Once this has been accomplished, the program can determine the address of the appropriate constant table for the current state in this box. A series of questions and associated operations can then begin. The first question is whether the present input is recorded. If it is, then the appropriate counter is incremented by one. If it is not, or after recording, if there is any, the constant table is asked whether the input is used to change states (cause a transition). If not, the question is asked whether there are any more state sets for this box and, if there are, a new constant table is obtained. If the input is used to change states, then questions are asked for the constant table concerning whether there is a preset count required of the input for state change. If there is a count required, the question of whether the count requirement is met is asked. If not, the program proceeds to the question of more state sets. If the input is not counted or if the counter is full, relay outputs and/or Z outputs are executed and the current state is changed to the next state. When all state sets have been treated in similar fashion, the interrupt is turned on to await new inputs.

The format of the constant table is such that only the information required for any particular state is entered into it. This results in a variable length constant table for each state that can vary from 2 to 77 words in length. In our experience, 5 or 6 words appears to be typical for most states. The amount of time necessary for an input to be handled by any one state is also variable, depending upon the number of operations required for the input.

The format of the constant table is shown in Figure 5. The constant table begins with one memory location for each input that is to be recorded in the current state. These locations contain the address of the memory location that is used to record the input, but if the input is not recorded the record location word will not exist. Secondly, there is a variable set of words that range from one to five for each state. These words contain the information concerning the existence and address of the functions required for each state. These functions are (1) recording, (2) counting, (3) transition, (4) relay output, (5) Z output for each of the 16 possible inputs. These inputs (as referred to in Figure 5) are the free running clock (K), three response inputs from the experimental subject (referred to as R, E, and T), and 12 Z's per box, for use in communicating between and synchronizing state sets. The outputs for each box, in addition to the "one-shot outputs" which are used to fire relays, are "C's", which are used to indicate the record counters that must be incremented on each input, and Z's which will be used as inputs to other state sets. Following the code words are the counter and the preset counter values necessary to change states. If the clock is used to change states, its pulses are always counted.

The clock routine has associated multipliers, since one memory location will overflow when a count of 4096 10-millisecond pulses occurs. The multiplier routine permits counts of the clock pulse to initiate transition in multiples of ten. In this way .01, .1, 1, and 10 seconds can be used to generate intervals from 10 milliseconds to 10 hours.

Following the counter set is a set of triplet words (maximum of three per input) consisting of the address of the main code word of the next state, the 12-bit relay output of this transition, and the Z output of the transition. If the input does not need this information, as indicated in the code words, these locations will not be included in the constant table.

This completes the description of a single state in this system. All the necessary states for a given box are entered into the computer memory and are used when needed, as determined by SKED, the operating program.

In summary, a system and computer language, general enough to handle all schedules of reinforcement or other behavioral procedures has been developed. The system requires translating the desired procedure into a state diagram, and entering the diagram into a compiler. The compiler produces a binary tape that the operating program will translate into the desired schedule of reinforcement.

References

1. Mealy, G. H. A method for synthesizing circuits. Bell System, Tech. Journ., 1955, 34, 1045-1079.

2. Moore, E. F. Gedanken experiments on sequential machines. in Automata Studies, Princeton University Press, Princeton, N. J., 1956.

3. McCluskey, E. J. Jr. Introduction to the Theory of Switching Circuits, Mc-Graw-Hill, New York, 1965.

4. Mechner, F. A notation system for the description of behavioral procedures. Journ. Exp. Anal. Behav., 1959, 2, 133-150.

5. Millenson, J. R. Principles of Behavioral Analysis. Macmillan, New York, 1967.

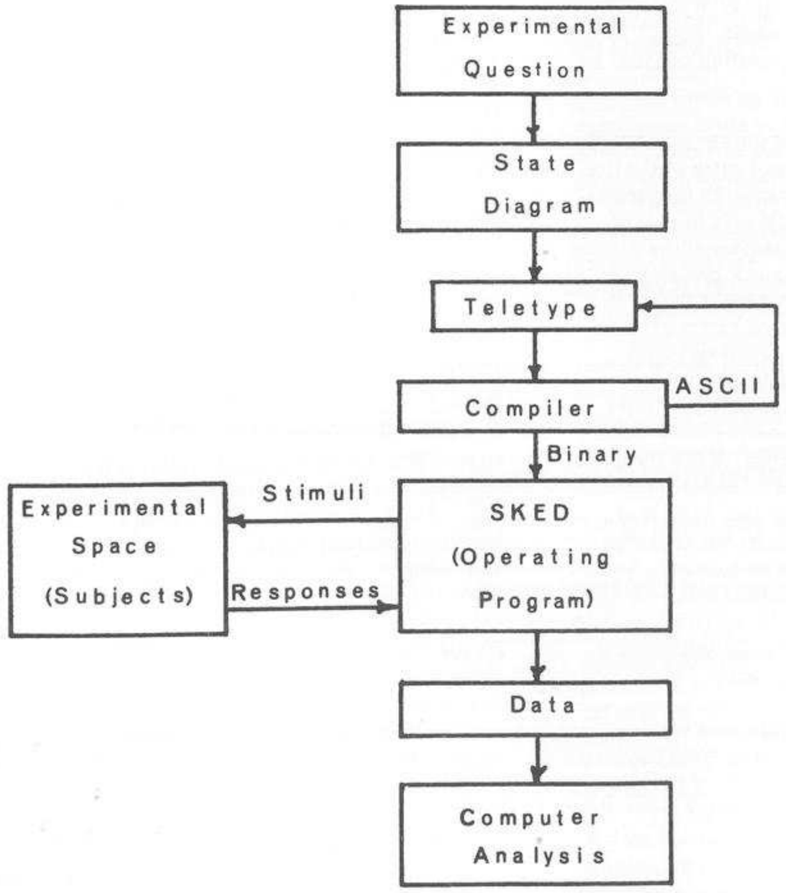


Figure 1 Flow Chart of Overall System

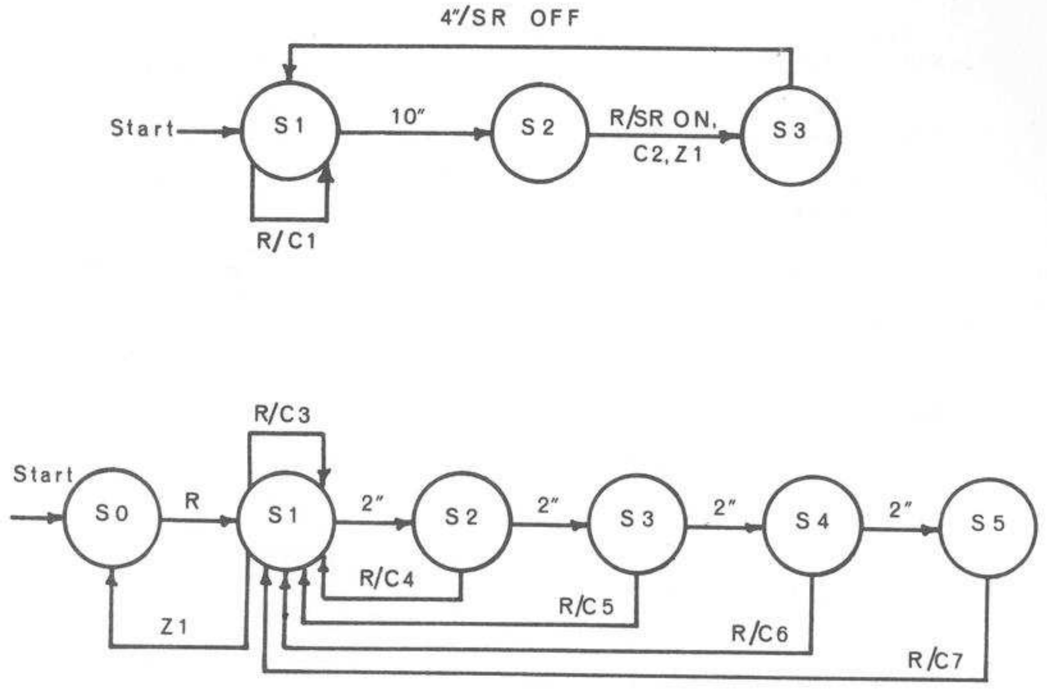


Figure 2 State Diagram of DRL-10 Schedule, with Recording of IRT's.

```
BOX # 3    SA 5005    # STATE SETS 2    IOT 6611
RECORD ADDRESS1 200
PARAMETERS:
SR ON=1
SR OFF=2

STATE SET 1

  10"
S1---->S2

  R/C1
S1----->S1

  R/SR ON, C2, Z1
S2------------------>S3

  4"/SR OFF
S3----------->S1

STATE SET 2

  R
S0-->S1

  2"
S1--->S2

  R/C3
S1----->S1

  Z1
S1--->S0

  2"
S2--->S3

  R/C4
S2----->S1

  2"
S3--->S4

  R/C5
S3--->S1

  2"
S4--->S5

  R/C6
S4----->S1

  R/C7
S5----->S1
```

Figure 3 Compiler Format for DRL Schedule

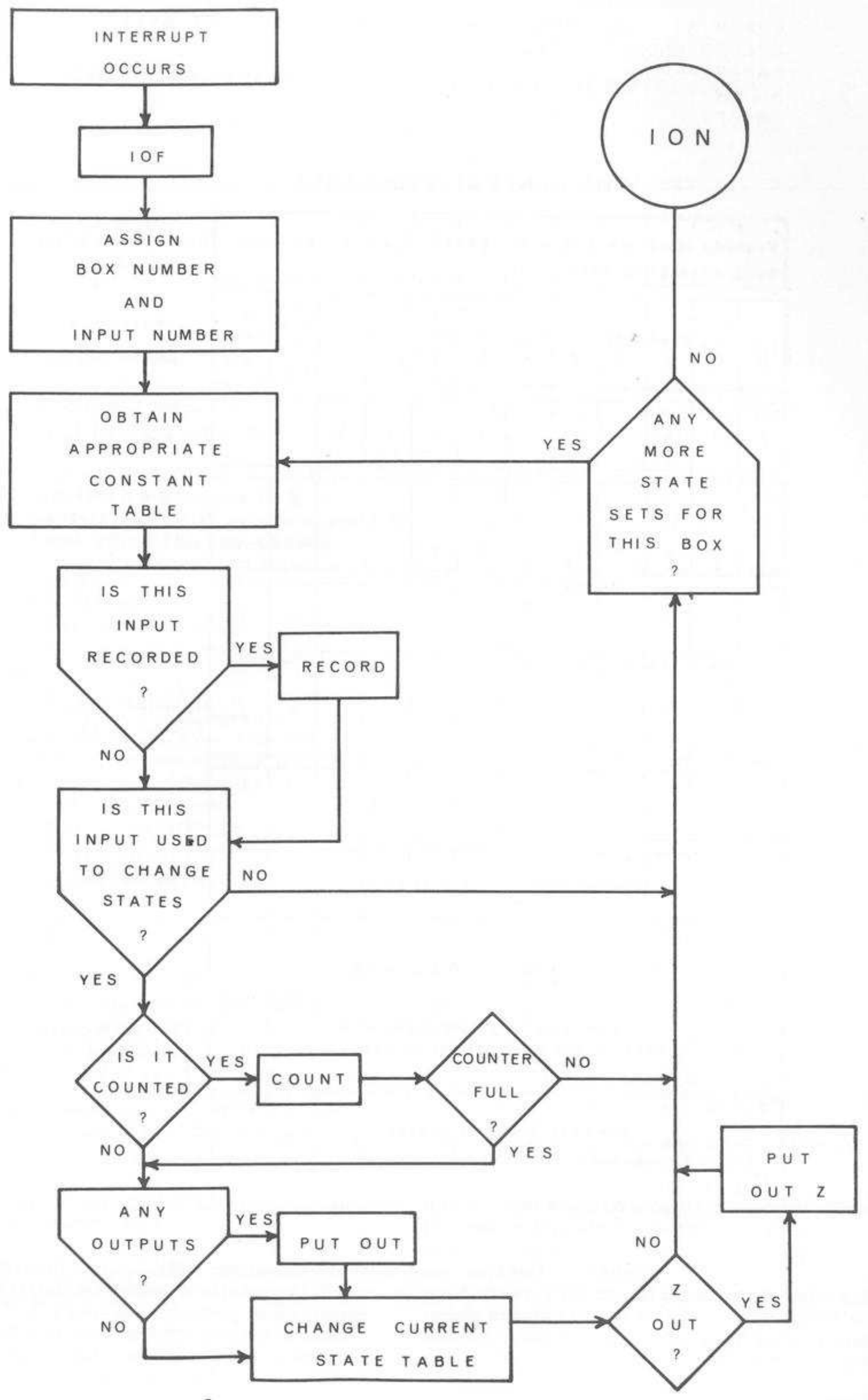

INTERRUPT OCCURS
IOF
ASSIGN BOX NUMBER AND INPUT NUMBER
OBTAIN APPROPRIATE CONSTANT TABLE
IS THIS INPUT RECORDED ?
YES
RECORD
NO
IS THIS INPUT USED TO CHANGE STATES ?
NO
YES
IS IT COUNTED ?
YES
COUNT
COUNTER FULL ?
NO
YES
NO
ANY OUTPUTS ?
YES
PUT OUT
NO
CHANGE CURRENT STATE TABLE
Z OUT ?
YES
PUT OUT Z
NO
ANY MORE STATE SETS FOR THIS BOX ?
YES
NO
ION

Figure 4 Flow Chart of SKED Operating Program

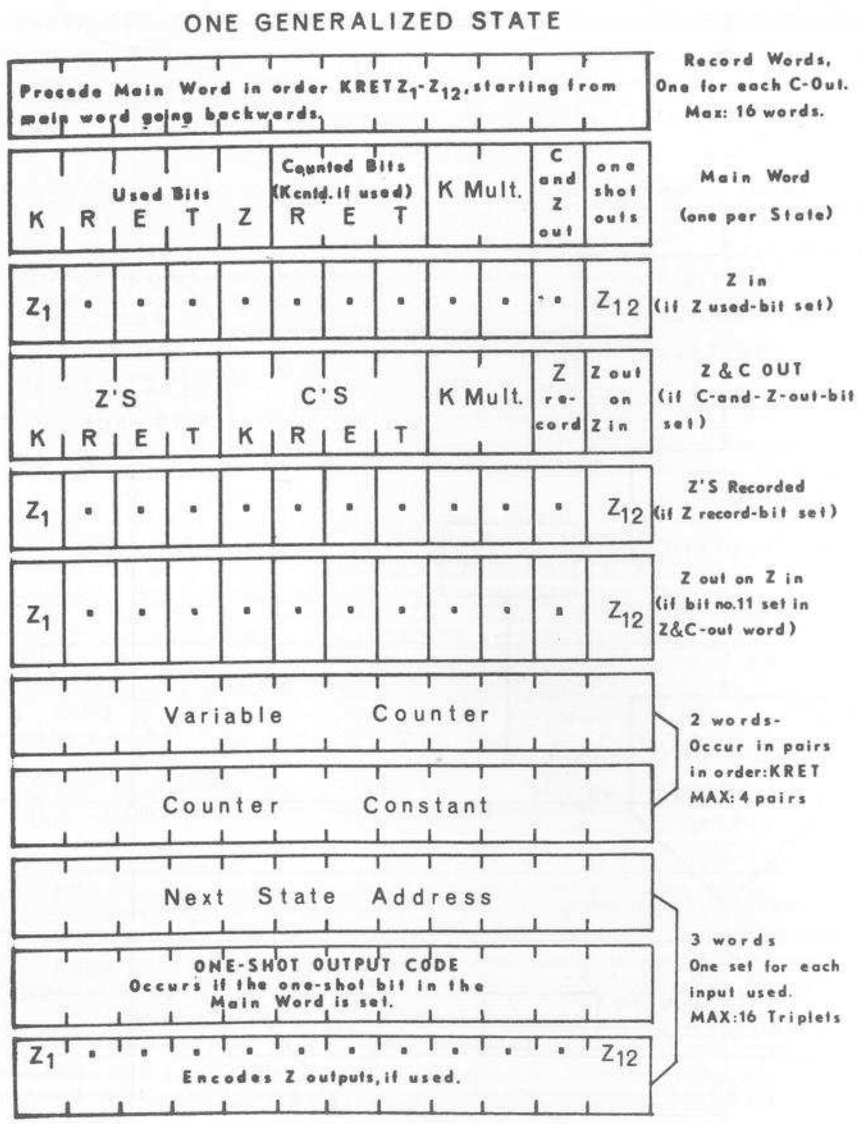

ONE GENERALIZED STATE
Precede Main Word in order KRETZ1-Z12, starting from main word going backwards.
Record Words, One for each C-Out. Max: 16 words.
Used Bits
K R E T Z
Counted Bits (Kcntd. if used)
R E T
K Mult.
C and Z out
one shot outs
Main Word (one per State)
Z1 Z12
Z in (if Z used-bit set)
Z'S
K R E T
C'S
K R E T
K Mult.
Z re-cord
Z out on Z in
Z & C OUT (if C-and-Z-out-bit set)
Z'S Recorded (if Z record-bit set)
Z out on Z in (if bit no.11 set in Z&C-out word)
Variable Counter
Counter Constant
2 words- Occur in pairs in order:KRET MAX: 4 pairs
Next State Address
ONE-SHOT OUTPUT CODE
Occurs if the one-shot bit in the Main Word is set.
Encodes Z outputs, if used.
3 words One set for each input used. MAX:16 Triplets

Figure 5 One Generalized State of the Constant Table

A MULTI-PURPOSE LOGIC MODULE FOR BEHAVIORAL EXPERIMENTS*

Arthur G. Snapper
FDR Veterans Administration Hospital
Montrose, New York

Julius Z. Knapp
Research Laboratories
Schering Corporation
Bloomfield, New Jersey

ABSTRACT

A notational system has been developed that can be used to describe most reinforcement schedules or other sequential procedures by appropriately interconnecting the basic notational units. The notational language has been applied in the past to develop an user-oriented computer program for behavioral experiments. The present paper describes a digital logic module designed to be isomorphic with the basic unit of the notational language, thus permitting rapid programming of new experiments directly from the notational system. The major advantage of the new module is that it requires no electronic sophistication on the part of the user, since the one module serves as the basic unit of all experiments.

Psychologists often study behavior in the experimental laboratory by establishing a set of rules relating responses (defined by a specified electrical event, e.g., closure of a switch activated by a lever) to certain stimuli (e.g., electric shock, food for a deprived subject, etc.). Typically, a stimulus is contingent upon the emission of a specified count of responses, or a response emitted following a specified delay after the previous response, or some combination of these events. In more complex procedures the intensity or frequency characteristics of a stimulus might be adjusted by the response of the subject.

Historically these procedures have been implemented by electrical or electronic circuits in which a group of relay, vacuum tube or transistorized logic modules are wired to provide the desired experimental relation between behavior and stimuli. A number of different suppliers have provided these modules, and, with a certain amount of trouble, equipment from different suppliers can be used in the same laboratory. However, it is true that voltage and pulse shape requirements often differ widely among the many types of modules available. Furthermore, most, if not all, of the currently available devices require the psychologist or his technician to become fairly adept in the design of complex sequential circuitry as demanded by the idiosyncrasies of a particular module line (e.g., trailing edge versus inverted logic, relay timing problems, etc.).

Recently the advent of low cost digital computers has made them available for realizing behavioral experiments (Weiss & Siegel, 1967)[8]. The principle advantages of the computer are high-speed, flexibility, and the ability to program procedures so complex that they are difficult, if not impossible, to realize using standard logic modules. The major disadvantage is that the hard-pressed psychologist is required to gain yet another skill, that of computer programmer, or to obtain the services of a professional for his laboratory.

As a solution to the programming problem we have proposed a new computer language (Snapper, Knapp, Kushner & Kadden, 1967[7]; Snapper & Kadden, 1969[5]) based on the theory and notation of finite automata (Mealy, 1955[2]; Moore, 1956[3]). We have attempted to show that behavioral procedures are finite sequential systems, and that, as such, can be completely described in terms of the exact arrangement of a set of units called states (Snapper, Knapp & Kushner, 1969[6]). The computer language, originally written for the PDP-8, provides for programming experimental contingencies in terms of states and their interconnections, and is thus a true psychological programming language.

The present paper extends this concept to logic modules by proposing that construction of a unit state and some associated hardware would permit a psychologist to implement an experiment by interconnecting state modules. This procedure would require only that the experimenter notate his experiment in terms of the state system. Once this had been achieved, wiring of the circuit would follow automatically. A second feature of the system is that modules from any manufacturer would be logically compatable with each other, thus making it possible to switch from one set of modules to another without having to learn a new logical system.

Furthermore, properties of the state notational system completely eliminate certain problems that otherwise arise in the design of sequential circuits from conventional modules (i.e., critical races in relay circuitry) thus simplifying programming of experiments.

A brief description of the state notational system follows to clarify the power of the language and to specify the properties

*This research was supported by Grant MH-13049 from the National Institute of Mental Health, USPHS, to W. N. Schoenfeld, Queens College, Flushing, New York, and, in part, by the Veterans Administration.

of the system that must be provided by any form of digital logic used to implement it. A fuller account may be found in Snapper, Knapp & Kushner (1969)[6].

States: The basic unit of the notational language is the state. Any behavioral procedure or other sequential control system can be subdivided into a set of units, only one of which describes the procedure at a particular time. For psychology the state may have a stimulus (presented to the subject) associated with it and these stimuli may differ or be the same from state to state. Each state also defines the requirements for state change to occur, if these requirements are met. States are notated by enumerated circles, and associated stimuli are abbreviated as SN (or S^R for reinforcement) to the right of a slash following the state number. Thus, in Figure 1, 1/S1 in the circle on the left indicates that some stimulus S1 is in effect during that portion of the experiment in which state 1 is present.

Transitions: State change or transition is notated by an arrow leading from one state to the next. In Figure 1 state 1 will be replaced by state 2 if a response (R) occurs when state 1 is in effect. Responses and time intervals serve as inputs that may cause state change. The transition from one state to the next is instantaneous (i.e., no other input during a transition can cause a second transition). If more than one transition is drawn from a particular state (e.g., the first state of the lower diagram in Figure 3) then the input to occur first (20 sec or R in the example) after the state is initiated, will cause a transition.

Outputs: As was noted earlier, each state may have an associated stimulus output that is in effect when the state is in effect. When a transition occurs to the next state the output or stimulus can be changed. If, however, the stimulus is to remain in two successive states and only their transitions differ, the fact that state change is considered to be instantaneous results in the common stimulus remaining in effect.

The definitions presented here provide the basis for a notation system that can describe any reinforcement schedule. Consider, for example, the diagram of regular reinforcement (CRF) shown in Figure 1. This diagram describes a procedure that starts with state 1 in effect. Associated with this state is stimulus S1 which represents the background stimuli of the experimental setting including house light, etc. If a response occurs in state 1, state 2 immediately replaces state 1 for 3" and the associated reinforcing stimulus (S^R) is presented. Three seconds after state 2 is entered, it is replaced by state 1 and conditions are as at the start of the experiment.

A slightly more complex procedure is diagrammed in Figure 2. The upper drawing consisting of three sequential states described a fixed-ratio schedule (Ferster & Skinner, 1957)[1] with a requirement of two responses for each reinforcement. The experiment starts in state 1 with stimulus S1 present. A response will cause transition to state 2 but stimulus S1 will still be in effect. A second response will lead to state 3 with a 3 sec reinforcement replacing S1. Finally, 3 secs after state 3 is entered state 1 is re-initiated and S1 is again presented to the subject of the experiment. Since fixed-ratio schedules require one more state than the number of responses per reinforcement, it is convenient to define the following higher order abbreviation:

When a set of sequential states have a common stimulus and the same transition but differ only in the number of transitions required to end the sequence, then the sequence may be expressed as .a single state shown to terminate on the nth input. In this way the lower drawing of Figure 2 is an abbreviation standing for the more complete upper state set. It means that state 1 is replaced by state 2 only when the second response in state 1 is emitted. This abbreviation may be conceptualized as adding a counter of inputs that may be associated with a state. On state entry the counter is set to zero, and state transition occurs only on the completion of the specified count.

In a similar way a definition of a resettable clock that may be associated with a state has been developed (Snapper, Knapp & Kushner, 1969)[6]. This definition specifies that a timer associated with a state will cause transition after a specified interval following state entry. Thus Figure 3 shows in the upper diagram a fixed-interval (FI) schedule. Twenty secs after entry of state 1, state 2 is initiated. Each entry of state 1 restarts the timer so that the full 20 secs will elapse. In the lower drawing of Figure 3 the diagram of DRL or differential reinforcement of low rate is presented. This schedule differs from that of FI only in the consequences of a response in state 1. In the DRL procedure such a response re-initiates state 1 thus resetting the associated timer.

A slightly more complex schedule is diagrammed in Figure 4 which illustrates a modified version of a commonly used non-discriminated avoidance schedule (Sidman, 1953)[4]. This procedure starts in state 1. If no response occurs before 20 secs elapse state 2 with its associated aversive shock (S^-) is entered. If a response does occur in state 1 its timer is reset and a new 20 sec interval is started. Further responses can keep state 1 in effect and thus avoid shock indefinitely. A response in state 2 turns off shock and initiates state 1. If no response occurs in state 2 state 3 is entered after 3 sec. State 3 stays in effect for 5 secs if there is no response and then restarts state 2. A response in state 3, however, leads to state 1.

Even more complex procedures can be shown in the state system with more complex diagrams. Other abbreviations can be developed from the basic definitions of the system, and recording of responses can also be diagrammed.

From the basic rules it follows that a state logic module and associated equipment should, at the very least, provide:

(a) that within a state set only one state may be in effect at any one time

(b) that state changes be as fast as possible

(c) that only one transition be allowed to occur at a time

(d) that a timer be provided that resets on state entry

(e) that a counter be developed that is reset on state entry.

A simple relay state module and associated timers, counters and input circuitry has been developed to illustrate the type of logic module that can be constructed to mimic properties of the state system. Figure 5 shows a simplified schematic of two partially inter-connected relay state modules. Each module contains two relays wired to act as flops with separate set and reset inputs. Solid lines represent hard-wiring and dotted lines show external wires representing state transitions. Assume state 1 is in effect. This means that its top relay is locked up through its normally open contacts but that the lower relay has been reset and is off as in the case of both relays in state 2 a level is provided from the contacts of state 1 to provide stimuli. If a pulse (R) occurs, it passes through the contacts of the closed relay and operates the lower relay of state 2. This relay, in turn, locks up through its own contacts and then simultaneously operates its upper relay and releases the upper relay of state 1. At the end of the response pulse, a clear pulse is generated that releases the lower relay of state 2.

Not shown but needed in practice are diodes to isolate the reset lines from each lower relay to the upper relay on each of the other states. Also needed is a wire passing through the normally open contacts of the top relay and the normally closed contacts of the bottom relay which would start a timer which is associated with the state. This latter feature permits re-initiation of the timers by operation of the lower relay of a state module. One must also provide an input interface of relays that sets the duration of the response and timing pulses and prevents responses and timing pulses from entering the system while transitions are occurring and stores these inputs until the transition is completed. This type of state module has been constructed and has been found to provide programming control for most commonly used reinforcement schedules with three states, three timers and one predetermined counter at a cost lower than $200. Similar modules could be constructed at a lower price from transistorized modules which would also offer higher speed and greater flexibility.

References

1. Ferster, C. B. & Skinner, B. F. Schedules of Reinforcement. New York: Appleton-Century-Crofts, 1957.

2. Mealy, G. H. A method for synthesizing sequential circuits. Bell Systems Technical Journal, 1955, 34, 1045-1079.

3. Moore, E.F. Gedanken-experiments on sequential machines. Automata Studies, Princeton: Princeton University Press, 1956.

4. Sidman, M. Avoidance conditioning with brief shock and no exteroceptive warning signal. Science, 1953, 118, 157-158.

5. Snapper, A. G. & Kadden, R. M. Time-sharing in a small computer through the use of a notation system. In B. Weiss (Ed.), Digital Computers in the Behavior Laboratory, Appleton-Century-Crofts, 1969.

6. Snapper, A. G., Knapp, J. Z. & Kushner, H. K. Mathematical descriptions of schedules of reinforcement. In W. N. Schoenfeld and J. Farmer (Eds.), Theories of Reinforcement Schedules, New York: Appleton-Century-Crofts, 1969.

7. Snapper, A. G., Knapp, J. Z., Kushner, H. K. & Kadden, R. M. A notation system and computer program for behavioral experiments. Digital Equipment Computer Users Society, New York, 1967.

8. Weiss, B. & Siegel, L. The laboratory computer in psychophysiology. In C. C. Brown (Ed.), Methods in Psychophysiology, Baltimore: Williams & Wilkins Co., 1967.

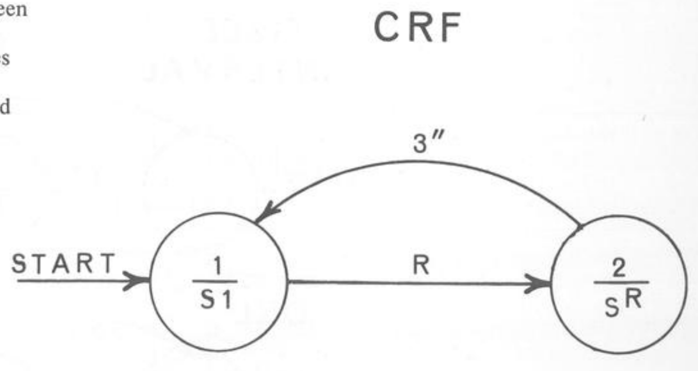


Figure 1. State Diagram Of Regular Reinforcement

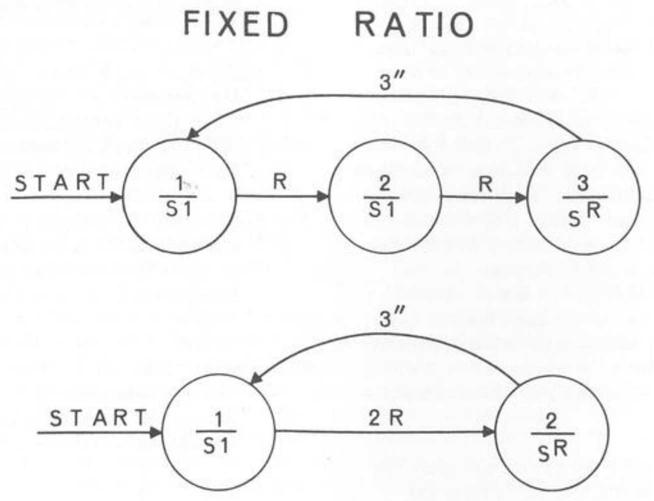


Figure 2. State Diagram Of Fixed-Ratio

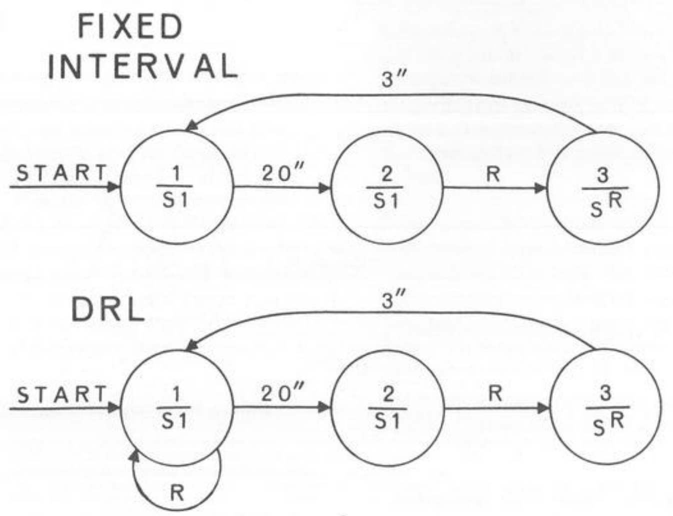


Figure 3. State Diagrams Of Fixed-Interval and DRL

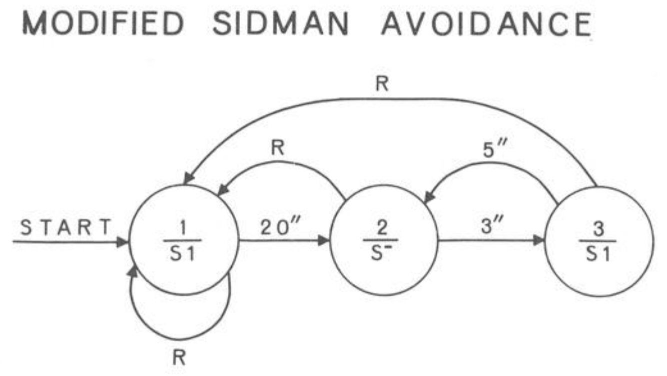


Figure 4. State Diagram Of Non-Discriminated Avoidance

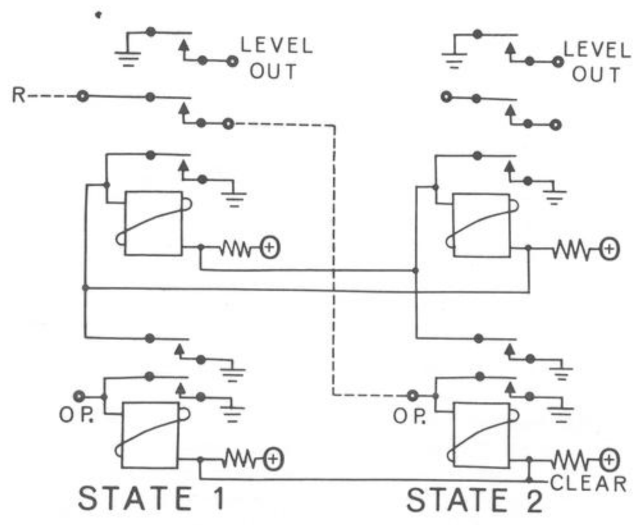


Figure 5. Schematic Of Relay State Module
External Wiring Drawn As Dotted Lines

A GENERAL LANGUAGE FOR ON-LINE CONTROL OF PSYCHOLOGICAL EXPERIMENTATION

J. R. Millenson
Department of Psychology, University of Reading
Reading, England

ABSTRACT

A problem-oriented language is being developed for on-line process control of psychological experimentation. The language consists of nested blocks of simple English statements familiar to every experimental psychologist. The function of this language is to produce an Automated Contingency Translator (ACT) which samples and updates a number of independent time-shared experimental environments 60 times a sec. Experimental procedures are mapped by the ACT compiler from the English statements into a probabilistic finite state network in list structure format. An independent operating system (which in the PDP-8, 4K version overwrites the compiler) then executes the list structure automata: that is, runs the experiments, records and retreives data and admits low priority background programs in any available dead time.

Psychologists have made extensive use of computers for data reduction and for simulation studies of behavioral processes. They have been, with some notable exceptions (references 1, 4, 15, 17) somewhat more diffident in applying computers directly to a third major class of problems, namely the laboratory control of their experiments. With the appearance of the small fast, accessible, and relatively inexpensive general purpose machines this reluctance on the part of psychologists can no longer be principally ascribed either to economical or instrumentation difficulties.

If we look closely at the situation we discover that for problems such as statistical analysis and data reduction, flexible routines and sub-programs for building a variety of special programs have been standardized in the algebraic-like languages of FORTRAN and ALGOL. Similarly in the fields of artificial intelligence and simulation of human problem solving by heuristic methods, well established special purpose interpreters such as IPL-V, LISP, and COMIT provide the investigator with a convenient language to formulate and manipulate his problems. In the field of laboratory control, however, the absence of any such general problem-oriented languages (reference 5) is conspicuous. Psychologists who wish to use the small computer to program experiments are generally obliged to learn a complex and unfamiliar code to communicate with their machine. Many are deterred by this language barrier; even for those who surmount it, the low-level machine or assembly languages that they have had to learn prove a poor vehicle for easy expression and creative exploration of new procedures. There have been attempts (reference 4) to use existing problem-oriented languages to do the job. But these problem oriented languages are oriented for the wrong problem, and thus while FORTRAN, for instance, makes algebraic formula translation and manipulation a routine exercise, it provides far less obviously a natural language for the logical structure of process control applications.

The key to a natural language for psychological procedures is an adequate theory of those procedures. While there have been a few attempts in the past to formulate the independent procedural variables of psychology into a unified framework (references 8, 9, 14) these attempts have contained important restrictions and indeterminancies which limit their general application to all procedures of the field. Recently, however, A. G. Snapper and his associates (references 6, 16) employed the concepts of cybernetic machine theory (reference 2) to demonstrate what amounts to a proof that all behavioral procedures can be formulated as finite state automata. They exploited this discovery to write a very general program for the PDP-8 computer to process control in real time a variety of animal conditioning procedures associated with schedules of reward and punishment.

Neither Snapper's group, nor Marlowe (reference 7), who also seems to have seen the implications of finite state automata theory for a general problem-oriented psychological process-control language, went so far as to evolve a general language for framing the procedures of psychology. Marlowe explicitly noted the practical difficulties in developing such a language and speculated as to whether 'the effort required. . . might offset any gain made by using such a programming language' (p. 10). There is reason to believe that this conclusion was over cautious. In what follows, the aims and lexical-grammatical properties of a compiler written for the Digital Equipment Corporation PDP-8 family, purporting to establish such a language, are described.

General Aims

From the outset it appeared that five major conditions would have to be satisfied by the language. Firstly, the language was to be a perfectly general problem-oriented one, oriented to the particular problems arising in on-line process control of behavioral experiments. It must be able, without any ad hoc additions to its basic form, to describe, and therefore given the support hardware, carry out the procedures of nearly all experiments ever done in experimental psychology. Only in that case could the kind of generality that would permit the investigator to use the language as a conceptual model or vehicle for creating his procedures of the future be assured. The utility of these new procedures, far more than its ability to simulate the procedures of the past is likely to constitute the ultimate justification of the language. It was thus apparent that the language should provide no special constraints for any one area of psychology, even though at present certain areas (for example, automation of conditioning techniques, recording of neuro-electrical phenomena) might be more obviously computer oriented than others.

A corollary to this first condition was that since the language was to provide a variety of investigators with differing backgrounds and theoretical dispositions with a general tool, the language must not possess a bias towards any one particular method of analysis within psychology. Thus it should be possible for any psychologist of whatever persuasion to be able to describe his procedures in this language. The general nature of Snapper et al's contribution assured that since all psychological procedures could be reduced to a state diagram, in principle this would indeed be the case. But the actual language that was to map that state diagram to a computer data structure must still contain as few idiosyncratic theoretical connotations as possible. For instance, while the language would clearly evolve with the ubiquitious terms of stimulus and response, it must in no way commit the investigator to a reflex psychology. Stimuli and responses, or if the investigator prefers, environmental situations and behavioral repetoires, are simply a natural and operational way to talk about the changes in environments and behavior patterns of living organisms that make up psychological experiments.

Secondly, parameter modification had to be integral, simple, yet powerful. As I (reference 10) and other (reference 17) have pointed out elsewhere the special promise of the high speed digital computer in controlling psychological procedures lies in its ability to adjust rapidly to very intimate behavioral properties of the subject. That adjustment consists of modifications, depending on the moment to moment status of those behavioral properties, not only of quantitative parameters but possibly even the very structure of the procedure.

Thirdly, the fundamental features of the program had to be viable with a minimum hardware configuration (e.g., a PDP-8/S with 4 K of core and a single teletype) so as to make the computer a feasible economic proposition in even small laboratories. At the same time, the program should possess sufficient flexibility to expand easily its command features so as to exploit the additional hardware of fortunate users with extra core, back-up memory, auxillary teletypewriters, display scopes, analogue converters, and so forth.

Fourthly, however complex the hardware input/output interface might in reality turn out to be at circuit level, it must appear to the psychologist-programmer wishing to control it as simple as possible: e.g., with PDP-8 machines, a simple 12-bit parallel word. Thus the program unlike ALGOL, but like FORTRAN, would standardize its input/output instructions by assuming a standardized interface terminal. Thus, complex microprogrammed machine input/output commands as such need never be seen directly by the programmer.

Fifthly, finally, and perhaps most important of all, the vocabulary, syntax, and grammer of the command language must be as close as possible to a simple English of everyday usage not unlike that which the investigator might employ in describing the procedures of his experiment to a colleague. The ability to use the language should therefore require almost no learning, it should be as insensitive to syntactical formalities (e.g., critical spacing, correct spelling, etc.) as possible.

Implementation On Site

A typical multi-access environment in which the computer might be expected to act as a process controller and data recorder for psychological laboratories is shown in Figure 1. A PDP-8/S central processor is shown there directing the procedures and recording the data from four experimental stations. The system shown is slightly expanded from minimal configuration, containing an additional output teleprinter used exclusively for printing out critical results of the experiments, and a high-speed paper tape punch for outputting selected aspects of the raw data.

In psychological experiments in environments of the sort shown in Figure 1, a key role of the computer is to provide automatic control of the relations or contingencies that the experimenter desires to hold between selected aspects of the behavior of the subjects, and subsequent presentation and maintenance of selected changes in the subjects' environments. The language that is to be described for this general task amounts to an Automated Contingency Translator, hence its mnemonic name, ACT.

The associated I/O hardware for each of the stations of

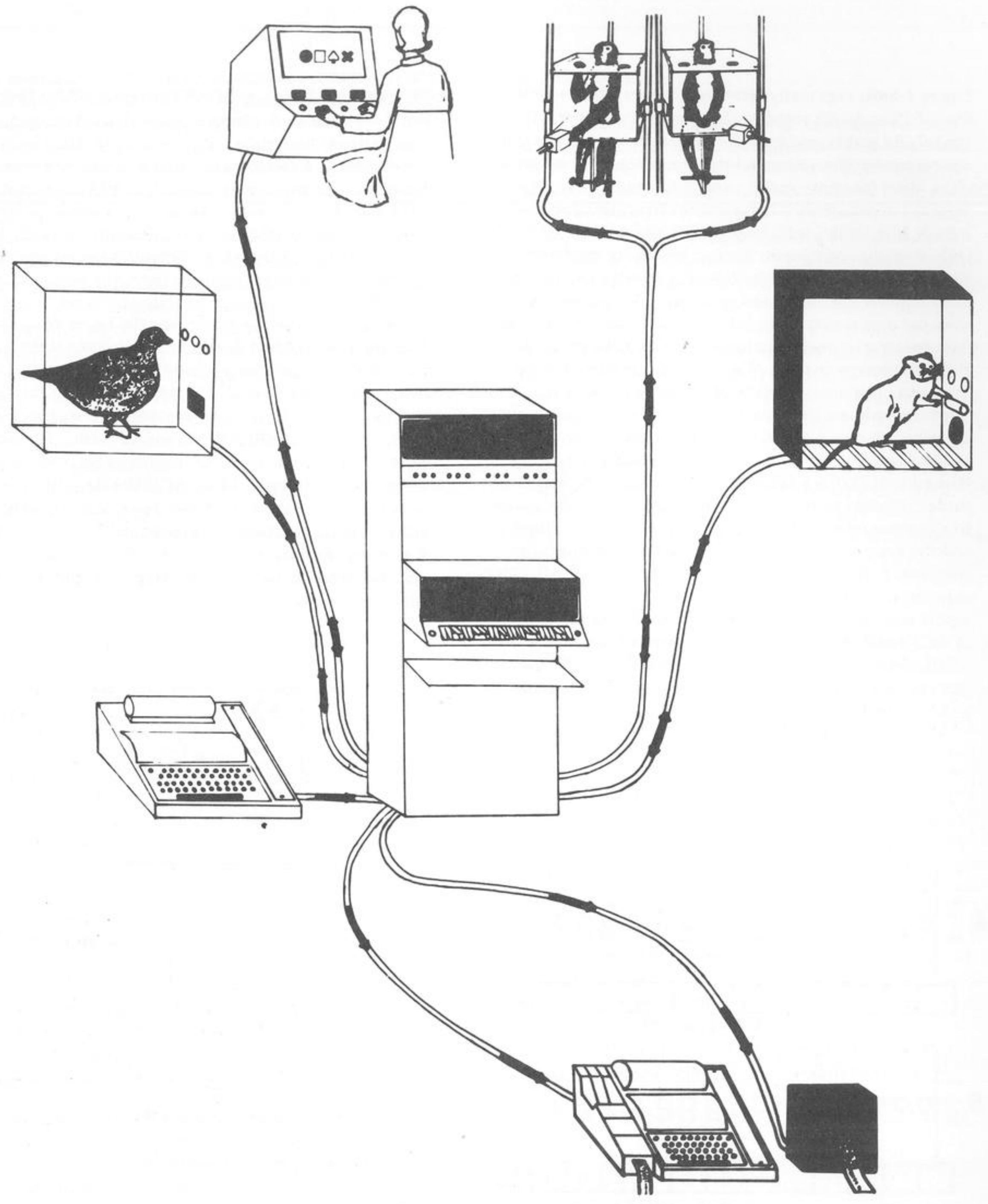


Figure 1 A typical multi-access psychological environment controlled by a rack-mounted PDP-8/S central processor. Four experimental stations, one input teletype (left), an input/output teletypewriter (bottom), and a high-speed punch are shown connected to the central processor.

Figure 1 must eventually terminate in two 12-bit words. One of these words registers the outputs (responses) from the subject and is read into the main arithmetic register of the computer, the accumulator. The other word consists of an input bit configuration and is strobed to the subject from the accumulator. A typical configuration for a rat subject in a conditioning experiment appears in Figure 2. Only a portion of the two distinct words are used, 8 bits for R outputs and 6 bits for stimulus inputs. Learning the octal number system to designate the behavior and environment events that he wishes to control is very nearly the only specialized computer knowledge that the psychologist is obliged to acquire. The use of octal labels for stimulus and response events is an important way of simplifying the program to meet, in a small machine, all the aims described above. Thus, R115 set in Figure 2 corresponds to the closing of a switch by the experimenter and a certain value (17) of a 5 bit analogue voltage taken from electrodes attached to the skin of the subject. S5 corresponds to a compound stimulus: the presence of a 30 cps tone and the presence of a small 'houselight' in the subject's chamber. In the prototype installation for testing ACT these stimulus outputs represent -24V to ground levels, and the inputs represent switch closures. Nevertheless the language of ACT itself is completely independent of how assertion voltages come to be on the R input lines, and what work one chooses to do with the assertion voltages the computer puts on the S output lines.

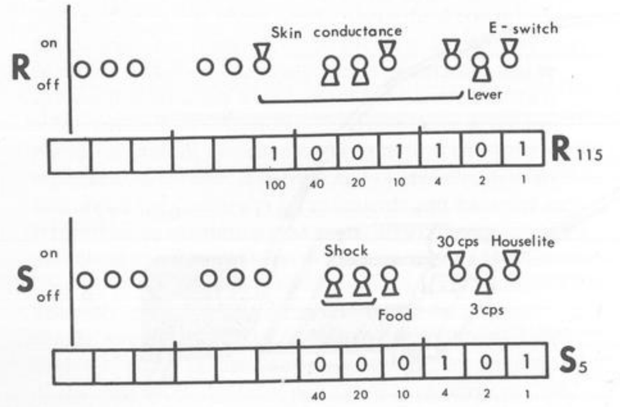


Figure 2 12-bit S-input/R-output words compared for illustrative purposes with two analagous banks of toggle switches. Only 8 bits (switches) of the R-output word and only 6 bits (switches) of the S-input word are in use.

Writing in the language of ACT is expedited by first drawing a modified state diagram of the desired procedures. The units of these state diagrams are the state, shown as a rectangle or a rectangular solid in Figure 3, labeled with the actual environmental conditions; and the transitions from one state to another shown as the vectored lines in Figure 3 with actual time or response values as labels. States may be (1) nested, as shown by the representation of boxes within boxes; and (2) they may be organized into multiple sets, or planes, of states as shown by the independence of the top portion of the figure from the bottom portion. Not shown in the pictures is the ability to modify at each occurrence of a unique state the value of variable parameters of the experiment, and the execution of a variety of data retrieval routines such as PRINTING, TAPING, DISPLAYING and so forth. These additions to the purely procedural contingencies relate ACT to the automata oriented REACTION HANDLER of Newman (reference 12), an intriguing correspondence since Newman's program was designed for an entirely different task environment, that of expediting communication between on-line users and graphical light-pen displays.

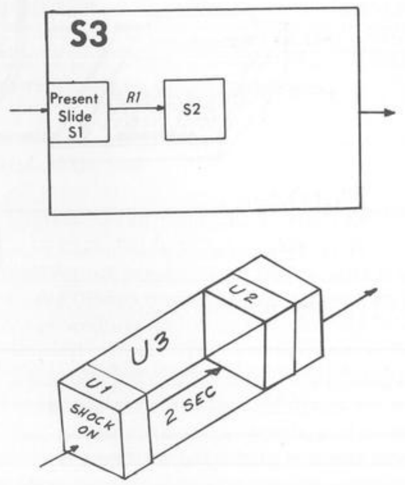


Figure 3 Two nested independent fragments of nested state diagrams for representing the contingencies of behavorial experiments.

To program his experiment the psychologist first works out a state diagram of it by listing the various sequential conditions and the temporal or behavioral events that will cause one condition to change to another. Then he connects up his structure with vectored lines corresponding to the contingency logic of the experiment. He then adds any special states he may need purely for recording purposes, or any states used for trapping the occurrence of rare or special experimental results. Then, referring to a representation of his input/output words (c.f., Figure 2) he assigns numerical values to the states, the behavioral response events, and the times.

What are his restrictions on such labeling? Firstly ACT is a completely synchronously driven system. An external clock which in the prototype operates at line frequency (60 Hz) restricts resolution of updating events to 60 times a second. Thus the shortest duration of a state is approximately 16.7 msec. (This is of course not a constraint of the language per se, only of the particular implementation of it. In the PDP-8 or 8/I a far faster clock would be practicable.) Secondly, the values of states must correspond to actual octal equivalents of 12-bit numbers. State 9 is thus ambiguous; response 72413 is too large.

Once in a state diagram, the procedure is ready to be mapped to the basic English of ACT. The vocabulary of ACT is shown in Figure 4. It will be

SYMBOLS

S1.2	$	V(J)
R4	←	+
U55	(tab)	-
I,J,. . .N	&	=

ENGLISH WORDS

WHEN, WHILE, GIVE

IF
AFTER
FOLLOWING

GO, THEN, GO TO

PROBABILITY

Figure 4 Legal ACT I Symbols and Words

observed that with exception of only a few special symbols, the vocabulary consists of simple English words, S, U, and V letter abbreviations for states, R abbreviations for responses, and the integers I through N for integer variables. In order to distinguish states that are associated with the same environmental conditions (e.g., the same octal output word) but which occur at different points in the procedure and are therefore associated with different experimental conditions, a 'point' followed by a unique number, less than or equal to 77_8 is used as a way of distinguishing such states. The ordinal number to the right of the point has no significance except to provide a unique arbitrary label.

STATEMENTS

In ACT there are six different classes of legal statements utilizing this vocabulary. Examples of these classes are shown in Figure 5. (1) Initialization of the integer variables by parameter assignment where the meaning should be self-evident. Such assignments are limited to the range -2048 < I < +2048. (2) Declarations of fixed variable states in S, U, or V state sets. Thus, writing a declarative statement beginning with WHEN, GIVE, or WHILE followed by a state identifier amounts pictorially to drawing a box. (3) Transition statements, of which there are three types. Writing one of these transition statements is equivalent to drawing a line away from one state rectangle to another. The six examples of transitions illustrate different features of the vocabulary. Response transitions begin with IF and designate a fixed (R2), variable (R(K)), or analogue (10< R <L) behavioral output for initiating a change in state. Time transitions are straightforward, with the upper limit being 72 hr, the lower limit being 1 unit (of time, determined by the clock) and variable times (being single precision integer) restricted to 16.7 msec. units specification.

```
INTEGER ASSIGNMENTS

    I = 4
    J = 7
    N = 7772

DECLARATIONS

    WHEN S1
    GIVE U12.5
    WHILE V(J)

TRANSITIONS

    IF R2 GO TO S44
    IF 29 R(K) GO TO V16.8
    IF (10 < R <L) GO TO S5
      AFTER 2 MIN GO WITH PROBABILITY
         = 3/32 to S4
      AFTER K UNITS THEN U2
         FOLLOWING J S55.2 THEN GO TO S1.0

PARAMETER MODIFICATION

    WHEN S1 [N = N + 4]
    WHEN S1 [J = K & SWITCH REGISTER]

DATA RETRIEVAL

    WHEN S1 [PRINT 'EXPERIMENT FINISHED %#']
    WHEN S1 [PUNCH 2]
```

Figure 5 Six Classes of Legal ACT Statements

RECORD DATA

1 IN S1: R1 LATENCY
2 IN S3: R1 COUNT
3 IN S5.1: R4 SUM
4 IN U(K): S2 COUNT
5 IN V16.7: V16.7 TIME

Figure 5 Six Classes of Legal ACT Statements (Cont.)

The first time transition (line 4 above under TRANSITIONS) illustrates an additional feature of ACT, namely its ability to describe probabilistic procedures. Probabilities of the form A/B where A must be greater than or equal to B, and B must be a negative power of two, less than or equal to 2^{-7}, may be specified for any transition. Probabilities so specified produce bernoulli distributions of events. In the prototype they are hardware generated by the peaks of a noisy diode counting in a 7-bit shift register.

The third class of transition shown as the last line under TRANSITIONS in Figure 5, is a second order transition in which an R or Time transition from any given state to another, at some point in the state diagram, can trigger yet another transition for a different state at a different level or in a different state plane. Second order transitions are essential for communication between state planes, and for making possible the changes in a procedure after a certain number of organism determined events, such as numbers of stimulus presentations of a given sort.

(4) Parameter modification forms a fourth group of legal statements. These constitute in ACT I simple three-operant ALGOL assignment statements written immediately after a state declaration and enclosed within square brackets. In ACT I the operators are limited to addition and subtraction ('&' means to add the switch register) and only a single such statement may be written per state declaration.

(5) Data retrieval statements shown just lower in this figure are also written at state declaration time. They make it possible to type messages and to retrieve various data stored by the five recording directives shown below them. The numerals after PRINT or PUNCH refer to the line numbers of (6) the record statements.

The manner by which these statements are combined into a program is straightforward. The statements are written one statement per teleprinter line, sequentially and in a block form suggestive of ALGOL. A given state is first declared, its integer parameter modifications or data retrieval orders enclosed within square brackets, and on subsequent lines all the possible transitions for that state are listed one by one. Another state is then declared and its transitions listed; and so on. State diagrams seem to most easily be constructed and read from left to right; so that the top of the page of ACT statements will generally correspond to the left side of a state diagram. For legibility the transitions of a state are indented one tab stop in from the declaration statement. Nesting of states within states is accomplished by maintaining a one-to-one relation between tabular indentation of the declared state to degree of nesting.

An example of a simple complete program to control the scheduling of a 3-second food reward to a confined pigeon subject for each key peck on a plastic disk only after a minute of non-reward has passed, appears in Figure 6. The program is initiated by a special non-printing character (WRU). The compiler (in 4K versions loaded by the user, in 8K versions called down by monitor) then prints out an installation identification followed by a new line and the word EXPERIMENT. The user types in a comment identifying his experiment, followed by a carriage return character. The ACT compiler then asks for a number for the station where the experiment is to be carried out. (This number is none other than the W103 device code for the station.) Upon receipt of this number the compiler will respond BUSY if the station is already in use, or if not busy with the approximate number of lines available for program. The user then types any initialization lines, the first declaration (S1), its transition, the next declaration (S1.1), its transition, and the third declaration (S3) and its transition. Any desired comments are preceded by the '/' symbol. Finally a $$ terminates compilation.

```
U.READING PSY PDP8/S ACT I COMPILER
EXPERIMENT   PIGEON FIXED INTERVAL (FI 1')
STATION      5
PROCEDURE    (0077 LINES  AVAILABLE)
 1  /S1=HOUSELIGHTS ONLY      S3=REINFORCEMENT
 1  GIVE S1
 2     AFTER 1 MIN GO TO S1.1
 3  WHEN S1.1
 4     IF R1 GO TO S3
 5  WHEN S3
 6     AFTER 3 SEC GO BACK TO S1
 7     $$

RECORD
 1  /RECORD ROUTINES NOT YET WRITTEN

0071  LINES STILL AVAIL
   PROGRAM ACCEPTED  00004096 MIN  29 APRIL 68
```

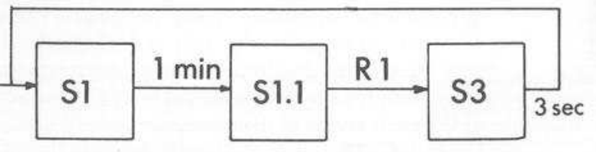


Figure 6 A simple program typeout and its associated state diagram. Lines typed by the computer are underlined.

Not shown in Figure 6 are occurrences of any one of 45 compiler diagnostics. These advise the user of syntactical (misspellings, illegal combinations of symbols, spurious characters, and so forth) or semantic (attempts to direct transitions from one state plane to another, or from one level of nesting to another, exceeding the upper limit of probabilities, failing to declare a state that is referred to in a transition line and so forth) errors. Syntactical diagnostics occur immediately upon receipt of the illegal character. Semantic diagnostics occur only at the termination of the current line.

The completion of the program leaves a data structure corresponding to the state diagram resident in core. The prototype system has 2K of core available to accommodate eight independent stations and at any one time, all eight may be occupied with programs of moderate complexity.

In practice a developed program would be stored on paper tape, and entered whenever it was desired to run. In the 4K version once all desired programs are resident, an operating system to execute the data structure is loaded overwriting the compiler, and thenceforth the teletypewriter serves only to print data messages; or if the data rates are low, to punch in coded form the significant events tagged with their relative times of occurrence. There is only one diagnostic at run time: 'AVAILABLE UPDATE TIME EXCEEDED' followed by the station last completed in the queue. The example of Figure 6 serves to illustrate the simplicity and naturalness of the language format for the problems for which it is directed.

An important feature of the language is its ability to incorporate new increasingly sophisticated design features without affecting the format of previous programs. Thus, the obviously desirable ability to have compiler and executive in core together with a monitor to schedule their priorities along with other perhaps unrelated background programs is a feature being developed for 8K systems. Users will type in new programs from one teletype while the central processor continues concurrent control of other experiments. Yet these and further refinements will affect only the power of the language system, not its format.

Conclusions

In summary, the language that has evolved permits multiaccess control of up to eight independent laboratory environments. The language includes facilities for modification of experimental parameters which, while simple, are powerful enough to permit the experimenter to produce procedures whose quantitative properties and/or qualitative structures can adjust to subtle on-going characteristics of the input behavior of the subjects. The language provides primitive facilities for retrieving the raw data of the experiment as it is being generated in a form suitable for later input to a conventional off-line data analysis program. An important feature of the language is that its very nature is such as to preclude the possibility of any user overwriting any other user. Thus the language permits trouble-free time-sharing of the central processor. Finally, as should be obvious, although ACT was written with the psychologist principally in mind, the language is a very general one. Whenever some aspect of an environment is to be controlled as a function of inputs from other aspects of that environment, as in chemical process control, machine tool control, biomedical monitoring, and other industrial and scientific applications (reference 13), ACT may provide a convenient method of implementing computer control by non-professional personnel.

ACKNOWLEDGMENTS

I thank J. D. Keehn, C. D. Webster, D. P. Hendry, W. Lennon, and D. Taylor for numerous suggestions towards the development of the language described.

References

1. Blough, D. The reinforcement of least-frequent interresponse times. J. exp. Anal. Behav., 1966, 9, 581-591.

2. Gill, A. Introduction to the theory of finite-state machines. New York: McGraw-Hill, 1962.

4. Hendry, D. P. and Lennon. W. On-line programming II, in Symposium, Automation of behavioral experiments, Washington, D.C., (APA) September, 1967.

5. Husky, H. D. An introduction to procedure oriented languages, in F. L. Alt and M. Rubinoff (Eds.) Advances in Computer, New York: Academic Press, 1964.

6. Knapp, J. Z., Kushner, H. K. and Snapper, A.G. Finite automata: a notation system and methodology for modeling and experimental control. (unpub), 1967.

7. Marlowe, L. A general purpose programming system for the LINC computer. DECUS Biomedical Symposium Proceedings, June 1967.

8. Mechner, F. A notation system for the description of behavioral procedures. J. exp. anal. Behav., 1959, 2, 133-150.

9. Millenson, J. R. Principles of behavioral analysis. (Chap. 6) Macmillan: New York, 1967.

10. Millenson, J. R. Maintaining avoidance behavior in the guinea-pig by an automated contingency translator. Alcoholism and Drug Addiction Research Foundation (Toronto) Project No. 168, Substudy, 1967.

11. Millenson, J. R. An automated contingency translator ACT I, A computer system for Process control of psychological experimentation. Psychonom. Bull, 1967, 1, 17.

12. Newman, W. M. A system for interactive graphical programming. Imperial College (London) Computer Technology Group report 67/7, October 1967.

13. Savas, E. S. Computer control of industrial processes. New York: McGraw Hill, 1965.

14. Schoenfeld, W. N., Cumming, W. W. and Hearst, E. On the classification of reinforcement schedules. Proc. Nat. Acad. Sci., 1956, 42, 563-570.

15. Shimp, C. P. Reinforcement of least-frequent sequences of choices. J. exp. anal. Behav., 1967, 10, 57-65.

16. Snapper, A. G., Kadden, R. M., Knapp, J. Z., and Kushner, H. K. A notational system and computer program for behavioral experiments. DECUS Biomedical Proceedings, June 1967.

17. Weiss, B., and Laties, V. G. Reinforcement schedule generated by an on-line digital computer. Science, 1965, 148, 658-661.

LINC-8 TEXT-HANDLING SOFTWARE FOR ON-LINE PSYCHOPHYSICAL EXPERIMENTS

B. Michael Wilber
Stanford Research Institute
Menlo Park, California 94025

ABSTRACT

A complete text-handling system (LUCIFER) has been developed for the LINC-8. All communication between LUCIFER and mortal man is carried on through a Teletype medium, so that hard copy is always produced, and one need never invoke scope, switches, and lights. Along with LUCIFER have appeared subroutines by which experiment-running programs can do input and output of data with text files or the Teletype. This paper discusses the philosophy of LUCIFER and includes examples of the use of LUCIFER and the running of a typical experiment.

We are using a LINC-8 computer for presenting stimuli and recording responses in psychophysical experiments. This use is characterized by extremely low data rates over long sessions. For example, experimental sessions typically take twenty minutes to an hour, with data rates of 30-180 bits per minute in each direction. Since much, if not all, of the computer's time is taken with running experiments, and because of the availability of commercial remote-access time-sharing computer facilities using ASR-33 terminals, we have decided not to do processing on the LINC-8 that can be done remotely. Communication between computers is via punched paper tape, and since this is our only use of that medium, it is not viewed as onerous. (Eventually we may be able to eliminate this use with a connection of the LINC-8 to the telephone lines.) Since the output of almost all the experiment-running programs is to other programs, output formatting is considerably simplified.

Because of the low data rates involved in our experiments, it is practical to input and output data in the form of text files called manuscripts. This greatly simplifies the problems of preparing input (stimulus) files and making sense of output (response) files because our regular text-handling programs can be brought to bear on these files. These programs, part of the LUCIFER[1,2] system, are needed for preparing the programs to run the experiments, so a great saving is realized by using the same programs to handle the data as much as it is handled on the LINC-8. A family of text-handling subroutines has grown out of the family of programs and has been made a part of LUCIFER, giving it the structure shown in Figure 1. Experiment-running programs can handle text by merely incorporating the subroutines and using simple calling sequences. An example of this is shown in Figures 2 and 3. We should note in passing that LUCIFER includes a program to convert almost any tape from LAP4 format to LUCIFER format, but there is no such program for LAP6. It might also be mentioned that the LUCIFER programs use the PROGOFOP typeout instruction and assumes the keyboard and typing mechanism are connected, so they probably could not be modified to run on the so-called classic LINC.

The LUCIFER Philosophy

Part of the philosophy of LUCIFER is that a typewriter-like device is a good medium for interacting with computers. With such a device, one has hard copy of what one did just before somebody walked in with an interesting question. Since one is often referring to typed and written material, a typed page in a well-lighted room seems to cause a good deal less eyestrain than a flickering scope in a dark room - - and well-lighted rooms are easier to come by than dark ones. When one is interacting with an active special-purpose program, one is less tempted to push the wrong button than with a passive, general purpose bank of switches and lights, because only logically 'correct' commands need be accepted by the program; and one can easily arrange that the consequences of a mistaken command are easier to recover from. See Fig. 4. Again, hard copy is produced automatically as a by-product of initializing a program instead of (hopefully) by somebody noting down on paper what is put into certain magic locations, hopefully without making mistakes. There also seems to be some advantage to leaving the file name permanently and automatically recorded on paper instead of hoping the right paper or magnetic tape or card deck was put in the right place. Operating instructions can almost completely be dispensed with if the program gives an idea of what parameters are needed, and in this mode, one can easily arrange that no steps can possibly be forgotten. Our experiment-running programs run through a set dialog and do not start the experiment until the end of the dialog.

[1]LINC unrelenting console interception and file editing routines.
[2] See LUCIFER documentation.

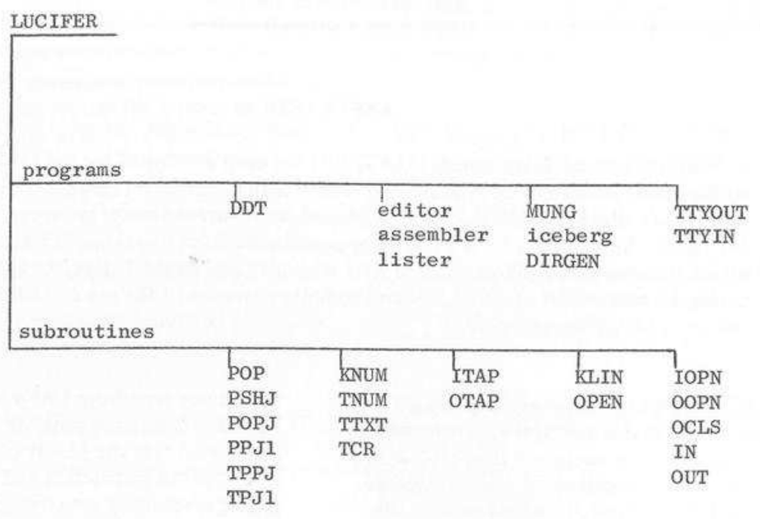


Figure 1 The overall organization of LUCIFER.

```
VISIØN↓
ØUTPUT TØ *WØRK↓
STIMULI *JOO↓
RANDØMIZATIØN .1 FRØM *ØRDFIL↓
EXIT
ØUTPUT TØ * Q↓
MUNG↓
MUNG FRØM *WØRK↓
TØ *269↓
MUNG FRØM * Q↓
TTYØUT↓
STRAIGHT CØPY FRØM *269↓
R 3
4030
4013
4013
0167
0071
  .
  .
  .
4315
4026
0067
QUIT.
```

TA-746522-26

Figure 2 Running a typical experiment, including the output of data via the Teletype. Here and in all the following figures, keyboard type-in is underlined.

It is felt that interactive programs can have more natural and easy-to-use command structures than currently appearing in many distributed programs. The LUCIFER programs are few in number, interactive and frequently used, so ease of memorization is a minor consideration. This is also helped by the fact that they have as many commands in common as apply.

In most situations not every command makes sense at all times. For instance, a line of the current file cannot be examined if there is no current file. It would help if one could obtain a priori knowledge of what sort of commands are acceptable in a given situation. In the LUCIFER and experiment-running programs, this consideration has given rise to the concept of prompting, i.e., all type-in is in response to some type-out from the program, and this type-out is somehow indicative of the nature of response expected. For example, as illustrated in Figure 5, if the last thing typed was an asterisk, then the program expects a file name, consisting of up to six characters terminating on a carriage return. Furthermore, rubout functions as a backspace and types as a backslash, and the special name blank Q('Q') causes the program to return to GUIDE.

In this connection, a little can be said about the LUCIFER programs command structures and methods of operation. First, one cannot modify text, directories, or core locations without first seeing what one is modifying. This is in contrast to some systems in which one cannot see one's text and change it at the same time. For example, to kill or replace a line of text, one directs the editor's attention to that line, which causes its contents to be typed out, and only then will the editor accept a command to replace or kill it. There is a certain amount of protection from careless errors afforded by the fact that the positions of the keys on the keyboard were considered when choosing the commands to be associated with certain actions. For instance, the editor kills and replaces lines and inserts lines before others (the commands are K, R and B) instead of changing, deleting, and inserting forward of a given line (the commands might be C, D and F) or killing, inserting and overlaying with commands K, I and O.

All the LUCIFER programs and all the experiment-running programs always refer to files by name, and so the actual location of a file is not relevant unless one wishes to add new files to the directory or expand ones already there. On the other hand, the format and location of the directory are well noted in the documentation, and the only program which lists the names in the directory also lists sufficient additional information to reconstruct parts of the directory or even the entire directory.

The editor edits a file in place and is a random access editor. Thus opening a file involves ascertaining its location, validity and length, requiring the inspection of exactly two tape blocks, instead of copying the file into some 'working area' and/or inspecting it to build a directory. Also one has no need to edit a file serially and/or be continually copying it between two temporary files and finally rename one with the original's name and kill the other one and the original source.

The LUCIFER Programs

The purpose in writing the LUCIFER programs was primarily to facilitate the process of forming programs on the LINC-8. This imposed two requirements on LUCIFER. First, it had to have a much more tractable overall organization and command structure than other available systems. Also, it was felt mandatory that there should be no overlaying. This not only greatly enhances response time of the programs and greatly facilitates the process of debugging them, but it makes programs more readable, permits them to be ordinary GUIDE programs, facilitates their assembly, and otherwise facilitates local and remote program updating. A secondary purpose was to consolidate action from the switches, lights, scope and Teletype to just the Teletype. The reason this was felt desirable is a foundation of the philosophy of LUCIFER. LUCIFER includes DDT, a debugging program, and four kinds of text-handling programs: the programs of primary interest some bookkeeping programs, two programs whose sole purpose is to allow our experiments to communicate experimental data with other computers, and one program which does not exist.

DDT is a simple program to examine and change storage locations in octal and exert some control over the execution of a program. It always works on the most recently assembled program, and there is no provision for saving (with GUIDE's FILEBI) this program in any but its pristine state - - it is possible to override this, but it is often easier to bring the manuscript up to date, thus facilitating later changes.

The programs of primary interest are the editor, the assembler, and the lister. The salient features of the editor have already been discussed. The assembler inputs the names of the manuscripts composing a program, types out all symbol (tag) definitions and assembles the binary for the program. The language it processes is an extension of a restriction of LAP4. Details may be gotten from its documentation, but some of the salient features are that lines may be as long as the editor will handle (something like sixty characters), symbols (tags) are up to four letters and/or digits with at least one letter, comments can be on the same line as anything else, and arbitrarily complex, logically meaningful expressions can be used for equalities and origins. Checking is much tighter, with almost any situation the assembler cannot correctly handle giving rise to a message containing the name of the situation, the current core location in the object program and the current line number. The lister simply produces a listing of a manuscript, with options

```
L↓

*J00↓
0002
•G
0001/
0002/
0003/   764  1364  1764  2364  2764  3364  3764  4364  4764  5364
0004/  5764  6364   764  1364  1764  2364  2764  3364  3764  4364
0005/  4764  5364  5764  6364   764  1364  1764  2364  2764  3364
0006/  3764  4364  4764  5364  5764  6364   764  1364  1764  2364
0007/  2764  3364  3764  4364  4764  5364  5764  6364   764  1364
0010/  1764  2364  2764  3364  3764  4364  4764  5364  5764  6364

•Q
*ØRDFIL↓
0016
•21L
•31U
•G
0021/  R 3
0022/  3     4     25    71    53    67    42    37    73    52
0023/  22    45    54    60    41    21    72    5     61    64
0024/  47    33    57    10    12    40    17    34    56    13
0025/  2     16    43    14    24    46    65    6     55    70
0026/  20    74    23    11    63    1     26    27    30    31
0027/  7     62    66    50    36    15    51    44    32    35
0030/

•Q
* Q↓
```

TA-746522-27

Figure 3 Typical input files for the experiment of Figure 2.

```
PART2↓

N-TUPLES *3↓

N-TUPLES *P/C↓
ØUTPUT TØ *WØRK↓
0014-TUPLES, 0102 ØF THEM.
DELAY 0..DELAY 0.DELAY 0.3
MARKERS .MDELAY 0.3
MARKERS .1 FRØM *MARK↓
C.R. FØR ERASE.P?
```

TA-746522-21

Figure 4 An experiment-running program rejecting “logically meaningless” parameters.

```
PART2↓

N-TUPLES *P/C↓
ØUTPUT TØ *WØRK↓
0014-TUPLES, 0102 ØF THEM.
DELAY 0.2
MARKERS .3 FRØM *MARK↓
C.R. FØR ERASE.P?

ICEBRG↓

*SEED1↓
.B=0101.S=0073.4C.S=0067.4H:ZØT↓
.B=0164.S=0004.R:FØØBAZ↓
.Q
*FØØBAZ↓
.B=0164.S=0004.4X.S=0010.KQ
*SEED1↓
.B=0101.S=0063.10X.S=0073.Q
* Q↓
```

TA-746522-22

Figure 5 Prompting by an experiment-running program and the iceberg.

to select only parts of the text and to form 8-1/2 by 11 inch pages.

There are three bookkeeping programs included in LUCIFER: MUNG[1], the iceberg and DIRGEN[2]. The latter is used to convert a LAP4 tape to LUCIFER format if that is possible. All programs and subroutines concerned with the text in a manuscript use information stored in the manuscript's directory, which tells how many blocks are actually occupied by the text and the highest line number for each block. A manuscript in this form is called normal, but it is much more convenient for experiment-running programs to write text in another form, called abnormal. Also, pursuant to the second law of thermodynamics, the manipulations of the editor are quite likely to lower a manuscript's packing density, but they are somewhat less likely to raise its density. For these two problems, we have a program MUNG, which reads a normal or abnormal text file and writes a normal text file with the highest possible density. Finally, all responsibility for the tape's directory of files is vested in the iceberg whose operation is shown in Figure 6. This program accepts explicit commands to change the directory but of course rejects any commands which would result in the directory becoming potentially invalid.

In running our experiments, we have found that additional processing should be done on other computers. With our present hardware and with the particular other computers used (dial-in with a Teletype), the only means of communication is ASCII-coded punched paper tape. This particular medium is less onerous, however, when it is viewed as backup to the storage of data in the more accessible forms and when one realizes that a very small absolute amount of data is handled this way and quite seldom at that. These programs are TTYOUT, which punches the contents of a file onto a paper tape with blank leader and trailer, and TTYIN, which inputs a paper tape into a file. For timing reasons, the latter will not run with PROGOFOP, but requires our own corruption of that program, which is named PROTOCROCK[3]. Since it is only intended for data, it does not handle the full character set, and for timing reasons, it is extremely limited in the length of a tape it will handle.

The last part of LUCIFER consists of a nonexistent program - - a mythical beast: It may not be a program at all - it may simply be a nonexistent feature of MUNG. Due to its lack of existence, one cannot say very much about it, and whatever one does say about it may be of indeterminant validity and concreteness. One can, however, say that this program, which has no name, has the property of merging several files, or possibly arbitrarily or otherwise selected portions thereof, into one file. Despite the obvious usefulness of this program, it has only been used once, so it has never been written. Probably the principal reasons for this state of affairs are that the assembler is nearly indifferent to the number of manuscripts composing its object program, and that the lack of real pages makes short manuscripts desirable.

The LUCIFER Subroutines

While the LUCIFER programs were being written, it was realized that their id and parts of their preconscious could be unified, generalized and quickened in manuscripts, so the experiment-running programs could easily communicate with LUCIFER and each other. The logical outcome of this idea is the LUCIFER subroutines. For the first few months of their existence, they were highly evolutionary, but as experiment-running programs were written around them, they gradually coalesced into a unified, modularly useful whole.

The structure of the subroutine package is that the subroutines are distributed across five interrelated manuscripts in such a way that at least six subsets of the manuscripts are conceivably useful. The selected manuscripts are assembled along with one or several other manuscripts in which reside the conscious part of the program and its own special-purpose preconscious and id. These other manuscripts have to set aside certain locations and areas with specified names, and some small amount of initialization is required, but otherwise one need not consider the internal mechanizations of the id of LUCIFER.

The specific functions represented on the manuscripts are the following. The first contains basic pushdown list manipulative functions. Although this makes recursive functions possible, only one program actually does have a recursive section, and that program is not often used. On the other hand, the pushdown list is seen as a good discipline for the allocation of temporary storage, and the handling subroutine returns (always a problem on the LINC) is uniformized and considerably simplified.

The next manuscript contains subroutines to type text displayed in the calling sequence and to input and output numbers in octal. In our use, it would seem that octal numbers are not objectionable even to people having their first contact with computers, because all numbers handled by any part of LUCIFER are octal, so it is seldom necessary to convert between octal and decimal.

On the next manuscript there is a subroutine to buffer a line of input, up to a preset maximum length and process backspaces. Since almost all input is performed by this subroutine and the number input subroutine on the previous manuscript,it does not take new people very long to learn the interactive characteristics of our programs. The other subroutine on this manuscript is a subroutine to find a file in the manuscript directory.

1 Manuscript ultra-normalization and generation.
2 The directory generator.
3 PDP-8 routine to oversee tape operations and cooperative routines which obtain console-type knowledge (the program of total crockery).

This subroutine uses the line input subroutine and does its own prompting, so there is some uniformity gained by this device. In addition, this subroutine detects names with lead blanks (these have significance as commands instead of names) and also obeys the command of this form which means that the current program should be terminated and GUIDE should be restored. Thus it is no accident that all our experiment-running programs at least exit in the same fashion.

A tape file can be treated as a character-oriented serial access input or output medium by use of the subroutines on the next manuscript. There are two completely independent subroutines here, one for input and one for output. The input file may be normal or abnormal, but the output file will be abnormal because it is very hard to write a normal file. In fact, only the editor and MUNG write normal text.

The upper levels of the id and the lower levels of the preconscious of LUCIFER are contained on the previous manuscripts; the last one contains subroutines from the upper levels of the preconscious and the lower levels of the id. There are subroutines to open and close a file to the output routine and to open a normal file to the input routine of the previous manuscript. In practice these special-purpose file opening subroutines are used instead of the general purpose one, which they call, on the third manuscript. All the previous subroutines do character input and output with instructions not intrinsically defined in the assembler. These instructions could be defined as operate-class instructions, if the Teletype is to be the medium for all character input and output. However, this final manuscript defines these instructions as funny subroutine calls. The subroutines, depending on the state of a flag in memory, cause either the Teletype or the current open input and output files to be used. Thus we have a degree of device independence. Device independence in itself is often used as a selling point for computers or software, but here, it is precisely what makes the subroutine package useful in the experiment-running programs. It is what enables the same set of subroutines to be used for communicating with the two media, and this makes the calling sequences tractable as well as cutting core requirements considerably.

Examples

Some examples of the use of LUCIFER are in order here. The Figure 7 shows the initialization necessary before using the full generality of the LUCIFER subroutines. The pushdown pointer (PDP) must be loaded with the address of the cell below the first cell of the area (PDL) reserved for the pushdown list, and the I/O medium switch (IOSW) must be set to indicate which medium is to be used first. Both of these cells could be preset at assembly time, but then one would not necessarily be able to abort execution at any arbitrary point and restart the program and have it retain a semblance of sanity. Next a carriage return is output, followed by a call to the text typing subroutine and a call to the input file opening subroutine. If the file does not exist or does not contain valid text, the subroutine does not skip on return, and the program's option is to restart from the beginning. Otherwise, another message is typed out, and the output file opening routine is called. Again, if the file does not exist, the subroutine does not skip. Otherwise it skips and the only further step necessary before the input and output files are the source and destination characters is to change the setting of the I/O switch.

The next example (Figure 8) illustrates the types of calling sequences used for the subroutines. In this passage, the I/O medium switch is set to the current input and output files. The subroutines used are the same as those used for Teletype I/O, but their id is under the influence of the medium switch. This is currently the main way device independence is useful, although programs could conceivably accept a fake file name, say blank T ('T') to mean the Teletype should be selected for a given activity. This is currently planned, but only for a program of low priority. In the passage at hand, the first activity is to read a number from the input file. If no number is found, the subroutine does not skip, and we call it again. The effect is to ignore anything not making sense as a number. The break character is returned in the accumulator, but we ignore it and output a carriage return to the output file. The carriage return subroutine returns with the accumulator clear, and then we pick up the number previously input. Next there is some calculation peculiar to the experiment this program runs, but finally the response is reduced to a number, which is then output onto the output file.

Figure 2 illustrates the entire process involved in running an experiment and obtaining a punched paper tape for graph plotting and further analysis. The program is loaded from GUIDE and runs through a set dialog in which it is told the names of the file containing the stimuli and responses as well as the name of a file containing randomizations and the position of the particular randomization to be applied to the stimuli. After this dialog, the program runs the experiment and, after writing the responses into the output file, restarts itself. This time, it is given the phony file name 'Q', which directs it to return control to GUIDE. The next step is to MUNG the responses from the abnormal temporary file to the normal permanent file. Finally, since the response text has been normalized, we can output it onto a paper tape, as is shown in the illustration. Paper tape input is not shown because usually this is done concurrently with the final debugging of the program which runs the experiment.

A final example (Figure 9) is an illustration of the edit-assembly-debug loop. The program is a simple adding

```
NØ. ØF MS ØN THIS
TAPE IS 22

 NAME      B      N

SEED1     101    73
69        174     4
FØØBAZ    200    10
GØRP      210    20
ZØT       230    10
MUMBLE    240    10
FØØBAZ    250    10
PDLMAN    260    10
IØSUBS    270    10
TAPEIØ    300    10
MSFIND    310    10
IØGENS    320    10
 CØRE     330     4
SEED5     501    17
RANDØM    520    10
LXIX      530    10
FIE       540    33
ZILCH     573    15
```

```
ICEBRG↓

*FIE↓
•B=0540•S=0033•5X?•35C?•13H:YECH↓
•B=0560•S=0013•15H?•R: YECH↓
?•Q
*YECH↓
•B=0560•S=0013•3C•R:FRØDØ↓
•Q
*FRØDØ↓
•B=0560•S=0010•KQ
*FRØDØ↓

*FIE↓
•B=0540•S=0020•15X?•S=0020•↓
•13X•S=0033•Q
* YECH↓

* CØRE↓

* Q↓
```

Figure 6 A typical manuscript directory and iceberg manipulations on it.

```
0001/  [HERE IS THE PREPRØCESSØR
0002/
0003/  SET & PDP [SET PUSHDØWN PØINTER
0004/   PDL-1
0005/  SET & IØSW[I/Ø WITH TELETYPE
0006/   0
0007/
0010/  JMP PSHJ  [TYPE A CARRIAGE RETURN FØR GØØD MEASURE
0011/   JMP TCR
0012/
0013/  JMP TTXT  [ØPEN STIMULUS FILE
0014/   316     ['N-TUPLES '
0015/   255
0016/   324
0017/   325
0020/   320
0021/   314
0022/   305
0023/   323
0024/   240
0025/  JMP PSHJ
0026/   JMP IØPN
0027/    JMP 20   [TRY AGAIN IF NØ SUCH FILE
0030/
0031/  JMP TTXT   [ØPEN RESPØNSE FILE
0032/   317     ['ØUTPUT TØ '
0033/   325
0034/   324
0035/   320
0036/   325
0037/   324
0040/   240
0041/   324
0042/   317
0043/   240
0044/  JMP PSHJ
0045/   JMP ØØPN
0046/    JMP 20  [IT DØESN'T EXIST-TAKE IT FRØM THE TØP
0047/
```

Figure 7 Example

```

#NXST    [MAIN LØØP ØF THE EXPERIMENT
JMP PSHJ     [GET NEXT RANDØM NUMBER
 JMP KNUM
  JMP '-2    [  IGNØRING THINGS THAT DØN'T MAKE SENSE
JMP PSHJ     [ØUTPUT FILE WANTS A C.R.
 JMP TCR

LDA &        [WE WANT TØ ØUTPUT AND WAIT
 4003
SCR 14
LDA &        [PICK UP A STIMULUS
 4003
ADD NMBR     [ RANDØMLY CHØSEN
STC TEMP
LDA TEMP
ØPR 6        [AND ØUTPUT
 NØP         [IGNØRE 0.1 SWITCH

SET & TEMP [HE PUSHED THE BUTTØN.
 FACT        [  GØØDIE.
LDA &        [ØUTPUT ZERØ
 3
SCR 14
ØPR 6
 XSK & TEMP [WILL MARK IFF NØT 0.1

SAM 10       [READ THE KNØB
ADA &        [AND
 377         [ ØFFSET
AZE &        [   PRØPERLY
 CLR
BSE TEMP     [NØW MAYBE MARK RESPØNSE
JMP PSHJ     [AND ØUTPUT
 JMP TNUM

XSK & CNTR  [MAYBE CØNTINUE
 JMP NXST   [ THE EXPERIMENT


```

TA-746522-24

Figure 8 Example

Figure 9 The edit-assemble-debug loop.

```
A↓

ASSEMBLE FRØM *F00↓
AND FRØM *PDLMAN↓
AND FRØM *IØSUBS↓
AND FRØM * E↓

NMBR0251
 :
LPKH0102
SUBT0075
FIE 0071
LXIX0053
UDS AT 00610043

ASSEMBLE FRØM * Q↓
E↓

*F00↓
0003
.43/  JMP PAHJR
JMP PSHJ↓
0044/    JMP KNUM↓

.Q
* Q↓
A↓

ASSEMBLE FRØM *F00↓
 :
LXIX0053

ASSEMBLE FRØM * Q↓
DDT↓

.63/1000 #
0062/6071 7400↓

.P=0020
.G
CHEAP ADDING MACHINE.
+5 0005
+3 0010
+44 0054
+S
.A=0046
.62/7400 6071↓

.P=0020
.G
CHEAP ADDING MACHINE.
+5 0005
+10 0015
+44 0061
+S0061
+14 0014
+52 0066
+12 0100
+S0100
+T0161

+Q
```

machine, residing in the manuscript FOO. At the beginning of the example, the assembler (named A) is called from GUIDE and directed to assemble the program, consisting of the manuscripts FOO, PDLMAN and IOSUBS, in that order. It lists the symbol definitions and notes an undefined symbol in the program. The next step is to direct the assembler to reload GUIDE, which is then told to load and start the editor (named E), whose attention is first directed toward the file FOO (which actually occupies three blocks of its allocation) and then toward the line mentioned in the assembler's message. This line contains a typing error (pahj for pshj), and this is corrected. Then GUIDE is reloaded and the program is reassembled, this time with no comment from the assembler. Now GUIDE is restarted, from which DDT is loaded. DDT automatically loads the most recently assembled program, which is the program of interest, before requesting a command. We change the location after the call to the number input subroutine to a jump to DDT, which starts, restarts and extends down to location 1400. Then we ascertain that the program will be started at location 20 and start it. It identifies itself and awaits input. Numbers are given it, and it types the running sum and then a letter is typed with no preceding number after which DDT signals its readiness. The accumulator is perceived to contain the character code for the letter, the altered instruction is restored, the starting location is again verified, and the program is restarted. This time the subtotal and total features are seen to be working, and the program is finally commanded to return to GUIDE. This example was typed in to the computer and debugged and embellished in twenty minutes, and the sample run shown here lasted about eight minutes.

Critique

The LUCIFER system, being a real system in constant use for four to ten months and having an evolutionary background, is possessed of some shortcomings, and it is felt that some space could be devoted to exposing them. The most fundamental fault is that, for historical reasons, two different character sets are used - - the LINC character set and DEC's ASCII character set. Fortunately, one of these is only used in outplotting to the Teletype. The full set of subroutines, though composing a comprehensive, easy to use package, require almost 380 locations in LINC lower memory and two LINC memory quarters for buffers. Of course, the buffers need only be respected while they are in use, a fact which is exploited in one of our experiment running programs.

There is some conspicuous room for improvement in the manipulations one can perform with the LUCIFER programs. The editor will not append to a manuscript but will only insert before a line. In practice, this means that all manuscripts have an empty line after the last meaningful one. Also, the editor will not make a new manuscript, but only edit an old one, so one usually has a manuscript, usually named SEED1, whose contents are exactly one empty line, which one MUNGs into a file before editing new text into it. The editor has two further properties which were included to increase its safety but which make its interactions take longer than would otherwise be necessary. First, it will not accept any commands pertaining to a line of text without first opening the line and typing out its entire contents. Also, whenever a change is made to a line, both the block containing that line and the manuscript's directory are written out onto the tape. Finally, editing would be greatly facilitated if it were possible to alter a line without retyping it in its entirety.

Other criticisms of the LUCIFER programs are more general. The editor, the iceberg, and the assembler are all incapable of handling the null case (i.e., empty manuscript, empty file directory and empty file directory and empty program). The editor and the iceberg both have safeguards built in so that they will not make an empty manuscript or directory from a nonempty one, and the action of the assembler on an empty program is harmless. There is no facility in LUCIFER for merging or dividing manuscripts. So far, LUCIFER is used mainly for preparing programs, and the assembler accepts a program spread out over many manuscripts, so the lack has not been objectionable enough to be cured. In the lifetimes of most data files, they are usually not changed enough that such a facility would be a great convenience. The iceberg is very crude - - it does little more than accept a human-oriented command language for a minimal set of atomic operations on the file directory and perform consistency checking between those commands and the directory. The intent was (and still is) to have it automatically invoked to create and extend output files as necessary, and to have it automatically handle the case where a file cannot be expanded without running into another file, but where there are free blocks on the tape. One final criticism which applied to all the LUCIFER programs is that their command languages are very tight, in that any would-be command which is not of exactly the correct format is rejected. For instance, in contrast to assembly language, blanks are forbidden wherever they are not mandatory.

These criticisms are presented to show the other side of LUCIFER. Without this section, we would have just been extolling the favorabe aspects of LUCIFER and ignoring the basic fact that ideal systems exist only before their logical consequences are attained and that as soon as a system is realized (in the form of running programs in production use, in this case), the logical consequences are hard to ignore. In considering the critique, one should bear in mind not only that it is offered by the author of LUCIFER, but that any sufficiently severe faults would be (and have been) corrected in the evolution of the system.

CAN A SMALL DEDICATED MACHINE FIND HAPPINESS IN A COMPANY THAT PIONEERED TIME-SHARING?

Joseph Markowitz
Bolt Beranek and Newman Inc.
Cambridge, Massachusetts, 02138

ABSTRACT

In the context of an automated psychophysical laboratory, the adequacies and inadequacies of a small computer (the PDP-8) are discussed. Attention is given to the rationale for choosing the implementation we did: in particular, a dedicated small system as opposed to time-shared use of a larger system. The variables discussed include cost, flexibility, and storage capabilities in addition to reliability. A compromise position that appears most viable for the future is also suggested.

Several years ago we had the opportunity to build a new laboratory facility for psychological research. Specifically, the laboratory was to be used for basic psychophysical research with acoustic signals. In certain instances, the implementation choices we made may have been closely tied to the portended nature of our work. In general, we think they were not. Indeed, with small modification, the same facility now serves a much broader range of psychological research interests. Perhaps such versatility owes its due to the way in which the facility was, in fact, implemented. For various reasons, which are the very point of this paper, we decided to automate our laboratory. The medium we chose was a small, dedicated, general-purpose digital computer – the PDP-8.

Our need for automation arose, we felt, because of the drudgery which defines psychophysics. Our psychophysical experimentation is characterized by a simple, repetitive, short sequence of events where all the alternatives are drawn from a small closed set. It is characterized by a need for numerous observations in each of a few fixed conditions. The operations are routine and boring. Errors made by the experimenter as to what he presents (or did present) are indistinguishable from errors on the part of the observer as to what he observed. In this sense, errors are very costly.

Psychophysics then does not demand a computer of high intellectual ability or blinding speed. Its tasks are not that challenging. Rather, psychophysical research requires a tireless and accurate drone.

Quite simply, precision increases as experimenter errors decrease, and as the number of observations increases. Unfortunately, the rate of an experimenter's errors accelerates as the number of observations increases. On the other hand, a computer can keep its own error rate satisfactorily and constantly low.

In designing our laboratory facility, we were confronted with several options. Because all of these options are nominally open, even today it is worthwhile reviewing them.

The first option we examined was construction of a special-purpose facility out of logical building blocks. This was, of course, a proven approach. We ourselves had had experience with three generations of logical building blocks. We were intimately familiar with the clatter and electromagnetic splatter of relays, the shocking and geriatric nature of vacuum tube logic, and the unforgiving frailties of transistors. We knew, too, the joys and sorrows of a variety of connectors and programmable patch panels. Finally, we knew the disproportionate number of logic modules and readout devices required for even the simplest of data analyses. All of these lessons stood us in good stead, we felt, and we never regretted the experiences of the past. Yet, we were firm in our resolve to depart from the past, for the lure of computers was irresistible. The printout was on the wall.

Another way in which we could have accomplished the automation of our psychophysical laboratory was to use time-sharing facilities available to us. We could conveniently have run our experiments on one of several operating systems. Our colleagues interested in running manual-control experiments were setting out to do precisely that. Moreover, ruling out such an alternative would be particularly significant in view of our past history with time-sharing and with computerized psychological experimentation.

Bolt Beranek and Newman Inc. has actively pursued time-sharing system research, development, and application for nearly a decade. Historically, we developed one of the first operational interactive time-sharing systems in 1962. That system was implemented on a DEC PDP-1, with the support of the National Institutes of Health.

A succession of hardware and software improvements enabled us to move from 4 to 64 simultaneous users and to increase the number and diversity of languages available to the individual user. In 1963, development was undertaken on a second PDP-1 interactive, time-sharing system. The system provided a number of operational programs in a specialized language at remote terminals in the Massachusetts General Hospital. In 1966, we implemented a real-time hybrid I/O interface to our time-shared SDS-940. This interface is called a Hybrid Processor, and is capable of accurately timing real-time data-transfers between real-world devices and the core memory of the SDS-940 without CPU intervention. We have used our Hybrid Processor to control an on-line video input device, to sample and generate speech waveforms at up to 33 kHz, to interface to a CRT display, and to control a hybrid analog computer (Applied Dynamics 4), to name a few.

We have also designed a scheduler which will handle a number of synchronous and asynchronous real-time processes. The scheduler will permit a process to get on the system only if the system can guarantee service to the real-time demands of this and all other processes. Copies of both the scheduler and hybrid-processor are presently being installed on our most recently acquired time-shared system, a dual-processor PDP-10 computer. The system is backed up by a 100-million-word disc for bulk storage, and has been modified by hardware "paging" of our own design.

As you can see, quite a bit of hardware and software development went into each of our time-sharing systems to provide adequate real-time service. This, of course, is precisely the need for controlling psychological experimentation. Nonetheless, we decided against time-shared service for our laboratory. This is ironic, indeed, because among the very earliest of computer-controlled psychological experiments were some done by BBN on the very computer on which we pioneered time-sharing (Swets, Millman, Fletcher, and Green, 1962).

The option we did exercise was to base our psychophysics laboratory around a small, dedicated, general-purpose computer. A computer-based system turned out to be cost-competitive with the build-it-yourself option at the time we made our initial decision. If anything, the economic balance has swung even more markedly in favor of the computer. Of course, this had not been the case when manufacturers offered for sale only medium to large computers. Now, however, small machines are a proven product line. As a consequence, we think that the specially designed collage of logical building blocks must be forever foreshadowed.

Building up special-purpose logic means, almost by definition, physically disparate pieces for each conceptual function the equipment must perform. With even a few functions to be implemented in this fixed way, the amount of logic required exceeds that necessary to build the processor of a small general-purpose computer. Moreover, core storage costs even at present are about $2.00 per word, a figure to be compared with the cost of a three- or four-digit electromagnetic counter. Perhaps it is best to think of the general-purpose digital computer as simply a particular, concise arrangement of some hardware.

By opting for a general-purpose digital computer upon which to base our facility, we also gained a degree of flexibility unobtainable in a speically built piece of gear. But, of course, as much flexibility and more would accrue if we chose to be a time-shared user of a large computer system. And yet we vetoed this course of action. We vetoed it on almost religious grounds, but we shall try to detail our reasons in as rational a fashion as possible.

Basically, our choice of a stand-alone system rested on the fear of unreliability of a large, multi-user system. We were then and are still of the opinion that unreliability is inherent in so complex a system. There are simply many, many more pieces, hardware and software, each with its own chance to fail.

A second consideration related to, but still distinct from, the reliability of a large multi-user system, is the integrity of one's own portion of the system. How secure is any user from the malfeasances of other users, whether of malicious or benign intent?

A third consideration was our feeling that the real-time demands which we would place upon a time-shared system in certain kinds of experiments were basically incompatible with the system's desire to schedule its activities "efficiently".

Let us discuss these three points in turn.

First, *reliability*... It is important to understand for ourselves, and to communicate to others, the premium we place upon reliability. Architects of large computer systems and directors of large computer centers rail at our demands for reliability. Generally so tolerant of our meager demands on their vast reservoir of compute-power and storage, they cannot understand our phobia of system malfunction. But our concern is hardly groundless, our fear scarcely irrational. The problem is that our subjects – people, animals, or whatever – change over time. They learn; they forget. They get rusty, or they rest up. The problem also is that our experiments, by design, place emphasis upon changes over time. All of us, I'm sure, can think of experiments where equipment malfunction on a given day may jeopardize an entire experiment. The cost, then, may not be simply the actual time during which the system was inoperative, but all of the experimental hours which preceded the failure as well. And, too, when experimental hours are lost they may involve a number of people: several subjects, an experimenter, and so forth. We also run the risk, if malfunctions are frequent, of tarnishing the image of omnipotence of the experimenter in the eyes of his subjects. And finally, even through the experimenter may be able to excuse the equipment for malfunctioning, the subjects may not be as gracious.

Most recently, informal sources indicate that an upswing in the reliability of large time shared systems may be in the offing. The idea seems to be that a portion of the potential savings in collectivity will be used instead to provide for a degree of parallel organization or redundancy in the system. The premise is that such organization will allow the system to provide fully adequate service to a decreased number of users in the event of some system malfunction. Certainly, the concept of parallel redundancy to ensure some minimal level of functioning is a proven one. It is not at all clear, however, that a degradation over all can be traded off against a reduction in number of users. Besides, the limits to which this approach will be pushed will be determined by what

standards of reliability the majority of users demand. As we have argued, however, the psychologist anxious to have the computer control his experiments may be rather more demanding than most.

Suppose, however, that degradation can be made to manifest itself solely as a decline in the number of allowable users. Who is it that gets "bumped"? When the system begins to falter and can accommodate fewer users, how much bargaining power does the psychologist have? The psychologist can keep the omnipotent CPU of a large system busy for the merest fraction of time. What law of economics dictates that he may pre-empt the inversion of a grotesquely large matrix? Paying a price out of proportion to the service must ultimately be the result.

Somewhere lower down in the hierarchy of reasons underlying our rejection of the time-sharing approach was the relative dollar cost of the two alternatives. According to standard accounting practice at then current pricing the two alternatives were remarkably similar in cost. They still are. They may always be. Not that the equation of costs is axiomatic necessarily. More likely, those who engineer time-sharing systems reinvest whatever savings there are to be had in pooling of resources and needs. The savings are plowed back into greater capacity, compute-power, and storage, as well as additional peripherals. The savings may also serve to underwrite more powerful languages, and a more intelligent time-sharing "executive", all of whose costs are ultimately borne by the overhead of the system anyway. Naturally, constraints operate to influence the decision of what savings should be rebated to the users directly but these are largely covert, tempered only by what the traffic will bear. At present (and perhaps always) the traffic will bear up to, but surely not far in excess of, what a small dedicated machine would cost to own and operate.

A second point to consider is the *integrity* of one's system.

The security of a user in a time-shared computer system is a concept of importance, not only to denizens of the nether worlds of big business or national defense. It is a concept of importance not only to those of us who exaggerate the importance of our work and the larcenous intent of our colleagues. For, if it is dangerous for our enemies to have access, how much more dangerous is such access in the hands of our friends? Presumably, people to whom we are neutral present an intermediate danger. Another maxim we might as well accept is that no realizable system can be totally secure. What compounds the problem in a time-shared computer system is the stratification of the users. All users of a system are not equal. The status of some is far above that of others. While most users have access to only their own pieces of the system, programs, and data, others have broader access. Whatever safeguards are established, there are always a few who have keys that unlock all the doors. It is not clear that this need always be so, but at present it seems to be the case.

The differential status of various users works to the disadvantage of the small, real-time customer (for such is the psychologist, in the main) in other ways. System maintenance, and worse, system revision, are largely scheduled for the larger good. How strongly can one press his case to have always available the hours of noon till one o'clock, particularly if that hour per day represents the sum total of use by the individual? Even worse, careful examination of the records might indicate that the hours of use reflects but several seconds of CPU time – according to some, a measure of the inefficiency of the machine use, and, by association, the unworthiness of the job.

Finally, let us consider the *real-time scheduling problem.*

One of the considerations that distinguishes the experimental psychologist's needs from those of many other real-time users is the asynchronous nature of many of our demands. In many applications – signal processing, or simulation, for example – it is possible to predict in advance the exact times at which service will be needed. However, many of our experimental needs cannot be so easily predicted. "From a scheduling point of view," it is claimed (Fiala, 1968) "the hardest processes to handle are those which vary rapidly at some times, and slowly at others. The system should attempt to guarantee satisfactory service to such a process under peak loads, while simultaneously using left-over time efficiently when the process imposes a light load on the system."

In other words, the higher the inter-response-time variance, the more difficult the scheduling problem. But for certain problems of interest to the psychologist, events of note are approximately Poisson-distributed. And, it is well known that the variance of a Poisson-distributed random variable is equal to – and therefore grows precisely as – its mean. Is this to say that we psychologists who deal with the relatively slow response rates of flesh and blood present quite a difficult scheduling problem? The implication that the less demand we make on the system, in terms of a low response rate, the more difficult a scheduling problem we pose, is indeed an interesting one.

The heart of the real-time processing problem is the input-output portion, rather than the computation demands. The program of a user operating in a time-shared environment has three alternative ways to control input-output functions: directly or according to a priority-interrupt scheme, or by consigning them to a separate hardware configuration, sometimes referred to as a hybrid processor.

Direct control of input-output functions means reasonably synchronous interlacing of the input-output processing with computation. As a result, the blocks of time left over to share may be so small, relative to the program-switching speed, that time-sharing may be infeasible, or uneconomical, for moderately high-bandwidth input-output requirements given direct control.

The use of a priority-interrupt scheme allows independent scheduling of input-output and computation, and, therefore, more efficient scheduling of computation may be implemented because, in general, computation is real-time insensitive. But, if enough devices are attached to the interrupt lines – and psychologists, likely, want to attach many such things – then, status tests must be made – an inefficient situation. Moreover, the pigeon keys and people push-buttons of the psychologists can hardly compete in priority with the system's tape, disc, drums, display, etc.

Most interesting of the alternatives is the one called the hybrid processor, which says that what is needed for efficiency is a separate hardware configuration for input-output processing, which is like saying that it needs a separate computer, which is like saying that one might just consider having a separate computer. The hybrid processor is provided with its own portal to the memory of the system, and, as we shall see, a reasonable proposal for the psychologist's own small machine is a similar portal to the large machine – where time is no longer so critical, inasmuch as the small machine can provide considerable buffering.

It is probably well to mention the rationale according to which participation in a time-sharing system is usually proposed. Basically, the argument is an economic one. Users might have need for a piece of hardware or software so infrequently as to be unable themselves to justify its cost. Yet the equipment is still desired. By pooling wishes and wants with others, these things may be acquirable. There is undeniable merit in the approach, but a danger as well. There is too often too little safeguard against acquiring pieces of the system that no one will use. Picture yourself coveting a novel piece of gear. Unable to justify it yourself, you argue that, while you have no real use for it, someone in the community of users must certainly. It is a little like buying your wife a sabre saw or your two-year old a set of electric trains.

It would be nice for systems to keep statistics on the use of their components, hardware and software. This sort of "over-buy" behavior may be more common than not. In a sense, it may be somewhat unavoidable inasmuch as a system must be planned before all the users and uses can be exactly determined. The counter-claim, of course, is that the sort of thinking to which we are here objecting is precisely the guarantee of flexibility that will promote the use of the system. We only suggest that the cost may be high.

The system configuration at which we have arrived is really quite straightforward – a basic PDP-8 with a high-speed paper tape reader and punch at the heart of the system. Equipped with only 4K of core and no other bulk-storage devices, the system relies on paper tape as an auxiliary storage medium, and as we shall see, occasionally needs several passes to collect data and perform rudimentary analyses upon it. Two digital-to-analog channels drive either a display scope, added only recently, or the transducers for our psychophysical experiments, e.g., earphones or vibrators.

Typically, three psychophysical observers are run in conjunction, and each has a four-button response keyboard convenient in that all subjects' responses are easily encoded as a single PDP-8 word. The subjects' response keys are also or-ed into the interrupt line along with the overflow bit of our 12-bit clock, driven by a 1 kHz ticker. This is illustrative of our solution to the PDP-8's lack of a priority-interrupt structure – our interrupt-handling routine is the simple one of reading in all external buffers.

Instructions, timing – control bits, as it were – and feedback information are displayed to the observers by means of indicator lights and form a single 12-bit buffered output word. A second 12 bits of indicator lights are provided for the experimenter. Another 12 bits of control information serve multiplex switches that enable us to gate analog signals to our observers. Finally, another buffered output word provides control bits for a 3-channel projection tachistoscopic projector.

Initially, we had no high-speed I/O device, but when we called heavily upon the reader and punch associated with our teletype, we found a distressingly high incidence of failure. In our opinion, some sort of high-speed input-output is a virtual necessity ... and this is equally true whether or not the system has bulk storage in the form of additional core, a disc, or drum.

There are several options as to high-speed I/O devices. Some are expensive – quite expensive in the case of multi-channel digital magnetic tape, and somewhat less in the case of micro-tape. We adopted paper tape some three years ago when ther was hardly any cost-competitive alternative.

Nowadays, paper tape is still appealing from the standpoint of price, as well as compatibility with other systems if that need should arise. However, two relatively recent alternatives are well worth considering: the first is an inexpensive mag-tape machine; the other is a high-speed communications link to some other facility.

The inexpensive mag-tape machine takes a bit-serial data stream and modulates it, thus making it compatible for storage on a good quality home tape recorder. Demodulation of the stored analog signal can subsequently reproduce the bit stream for re-entry into the computer.

The high-speed communications link is perhaps the most exciting new avenue. Properly done, with the cooperation of the computer utilities available, such a link can effect the compromise that must surely come for the user who desires a dedicated small system yet yearns everso occasionally for the benefits of a big system.

The dedicated system could be free-standing during crucial experimental applications, yet store programs, data, and institute analyses on a larger, more suitable machine. Control may be considered to be with the small machine. A separate, conventional teletype link to the time-shared system might then be reinterpreted as providing off-line file handling and programming capabilities.

REFERENCES

Swets, J. A., S. H. Millman, W. E. Fletcher, and D. M. Green, 1962. "Learning to Identify Nonverbal Sounds: An Application of a Computer as a Teaching Machine." J. Acoust. Soc. Am., 34:928-935.

Fiala, E. R., 1968. "Scheduling of Real-Time Processes in a Time-Shared Environment." Unpublished M.Sc. Thesis, Massachusetts Institute of Technology.

THE LEARNING RESEARCH AND DEVELOPMENT CENTER'S COMPUTER ASSISTED LABORATORY*

Ronald G. Ragsdale
University of Pittsburgh
Pittsburgh, Pennsylvania

ABSTRACT

This paper describes the operation and planned applications of a computer-assisted laboratory for social science research. The laboratory centers around an 8K PDP-7 and its special peripheral equipment, with most of the system already in operation. Special devices include random-access audio and video, graphical input, touch-sensitive and block-manipulation inputs. The control programs for these devices are incorporated in an executive system which permits simultaneous operation of six student stations. The system may be used for presenting instructional material or for conducting psychological experiments.

In April of 1964, the Cooperative Research Branch of the United States Office of Education established a research and development center at the University of Pittsburgh as part of the Office of Education's program designed to concentrate major effort in various areas in education. The Center at the University of Pittsburgh, called the Learning Research and Development Center (LRDC), directs its activities to design and development of instructional practices on the basis of experimental research on learning.

One activity has been construction of a computer-assisted laboratory as a first step of a project on computer-assisted instruction. The Computer-Assisted Instruction Project has two principal objectives which guide the scope of the effort undertaken within this group. The first objective is to provide those facilities and services needed to support the research and development effort of experimental psychologists and others in the field of instructional technology. The facilities include apparatus and controls used in learning experiments which use computer-related equipment. The group provides engineering and programming assistance in design and conduct of experiments. This service, primarily for the Center staff, may be used by other faculty members.

The second objective of this group is to conduct experimental work in the development of computer-based-instructional systems. Examples of such work include the development of supervisory programs for controlling many independently operating stations; the development of languages that educators can use for subject matter programs on the computer; and the development of student stations that will provide a high degree of interaction between the student and the subject matter.

*The research and development reported herein was performed pursuant to a contract with the United States Office of Education, Department of Health, Education and Welfare under the provisions of the Cooperative Research Program.

HARDWARE

The main piece of hardware involved in this project is a Digital Equipment Corporation PDP-7 computer, installed in mid-June of this year. In addition to the standard PDP-7 configuration, this computer has 8,192 words of core memory, 16 levels of automatic priority interrupts, the extended arithmetic element, 100 card-per-minute reader, variable time clock with speeds up to 1000 cycles per second, 2 output relay buffers and terminals for connecting student devices to the computer. The core memory will soon be increased to 16,384 words, with the additional 8,192 already on order. It is anticipated that magnetic drum storage will soon be added to the system as well.

In addition to the computer room, shop area, and an auxiliary equipment room, there are eight separate laboratory areas which average about 180 square feet. The labs range in size from 140 square feet to 285 square feet, with two of the larger labs having observation areas separated from the lab room by one-way glass.

Electrical ducts connect each laboratory area to the computer room. This allows a great deal of flexibility in assigning display and response equipment to the student station. The most basic device is the keyboard, which is a modified Type 33 Teletype keyboard. Because no printer is associated with the keyboard, this function is usually served by an oscilloscope screen (a Tektronix RM 564 connected with a Type 34 interface). This scope can be operated in stored mode in which information on the screen is preserved and does not have to be refreshed. When the scope is in dynamic mode (selected under computer control) it may be used with a light pen.

Another basic device is the audio speaker (or head set). Audio information is stored on loops of 6-inch wide magnetic tape. Each loop has 128 tracks, with 16 play and record heads each servicing 8 tracks. The size of the loop is variable, but at present each holds 1,024 seconds of information with each 1-second block individually addressable. Although it is not a standard item, there may also be a microphone at the student station so that he can record on certain areas of the tape reserved for this purpose (through software). Basic equipment for a student station consists of a keyboard, a cathode ray tube with light pen, and an audio speaker or head set (see Figure 2).

In addition to the basic student station devices, a number of special devices, some experimental, exist. One special device already in operation is the "touch-sensitive display" (Figure 3). A random access slide projector is focused on a back lighted screen which displays the information for the student. Many fine wires within the screen allow detection of an object, such as a finger or pointer, striking the screen. Detection of such a response also specifies the location of the response upon the screen. Since the projector and the screen are both under computer control, the presentation of visual stimuli may be sequenced according to the position of the preceding response(s).

Other devices under development include graphic tablets like the "RAND tablet" and "manipulation boards." The graphic tablet is a device through which a student may input graphical information directly to the computer with an electronic pencil (Figure 4). The tablet, used in conjunction with appropriate diagnostic routines, permits the teaching of printing and drafting, to name only one example, in a manner which allows easy detection of student errors.

At the RAND Corporation, the graphic input tablet is used for debugging programs. A scope displays the contents of several standard locations plus some memory location. The programmer can search memory upward or downward by pointing the electronic pencil at the appropriate spot. He can change the contents of a displayed memory location by merely writing over the display with the electronic pencil.

Figure 5 shows the manipulation board which detects the placement of objects on its surface and relays the corresponding bit pattern to the computer. The problem here is very similar to that of the RAND tablet in that a bit pattern must be recognized. However, in this case, the set of objects such as blocks representing different number quantities, can be restricted to those easily identifiable. With such a restriction, the set of objects can be identified both as to item and location upon the board. This device should be particularly effective in working with young children, or the mentally retarded, since it puts very little demand on the student insofar as the structure of the response is concerned.

A device for magnetic storage of video information is planned for the near future which will permit storage of approximately 500 pictures combining video camera or computer-plotted output. The selection of pictures will be under computer control and additional points may be plotted on the selected picture at any time. This system will also permit the use of a light pen.

SOFTWARE

The present software system, still in the final debugging state, consists of a set of control programs for each of the various devices as well as an executive system to control the sequencing of jobs, timing, memory allocation, etc.

All teaching or experimental programs being contemplated are of a type usually waiting for a student response or for some time delay to run out. When this happens, the program returns control to the scheduler and some other job is initiated. In general, this sytem is capable of servicing six devices of each type, although at present only one of each is in use.

Most of the 16 priority levels are already used by the system. The 60 cycle and 1000 cycle clocks each use 2 levels. Keyboards, light pens, touch-sensitive displays, microphones, and audio tapes also require priority levels based upon the speed of the response they demand. In addition, the executive routine responds to a priority level which may be activated through software.

This interrupt, which has the lowest priority level, is set whenever an interrupt signals the end of some job's suspension. Thus, when all other interrupts have been processed, the scheduler regains control and can reinitiate this job.

Programming of learning experiments, teaching routines, etc., can be accomplished only by machine language at present. A large part of the coding would, of couse, be in the form of subroutine calling sequences which reference the control programs. This type of programming may be more difficult and time consuming, but it also offers a maximum in flexibility.

One major goal of the Computer-Assisted Instruction Project is the implementation of a language which educators can easily use for programming subject matter on the computer. Although the final goal in this area is a compiler which accepts lessons written in some behaviorally oriented language and translates them into the proper codes, several steps are involved. The first, and probably the most important, is the definition of terms which denote particular instructional subsequences, etc. From this foundation, one can proceed until a final compiler is defined and operating.

INDIVIDUALIZED INSTRUCTION

A project which is planned to eventually merge with the Computer-Assisted Laboratory is now being carried out in a suburban Pittsburgh elementary school. In this school, students are allowed to progress at their own rate through curricula in mathematics, reading, and science, as they might in a computer-assisted classroom. Each unit of instruction is composed of skills which must be mastered with appropriate instructional materials and progress and quality control tests provided for each skill to be learned. There is a considerable library of instructional sequences and relevant information provided by the materials for a particular skill, and the teacher must prescribe those materials which she feels are most appropriate. It is clear that a curriculum of this type lends itself well to computer instruction, the major difficulty being simulation of the teacher's decision process.

LEARNING EXPERIMENTS

The development of a library of programs relevant to the running of learning experiments is less structured. There are certain basic paradigms which provide a basis for such a library, but the need is for more flexibility, in addition to implementing the old routines.

There are two major objectives in the learning experiment area: experimental design and real-time data analysis.

In the design and control of learning experiments, programs would assign subjects to groups on the basis of certain control variables; they would monitor information such as "anxiety level," for example, and modify it if necessary; and they would equate the subject's performance on a certain task prior to the application of the experimental treatment by the assignment of appropriate training or by statistical adjustment. These programs would perform those tasks usually requiring lengthly subject selection prior to the experiment or the conduct of preliminary base-line pilot studies.

Closely related to these programs are programs for real-time data analysis which provide data for the control and design section. This would permit use of complex criterion measures in equating the performance of subjects, to mention only one example. This type of data analysis would be especially helpful in a pilot study, since one could start out with a large number of variables and as the analysis indicated, discard those obviously relevant or obviously irrelevant, and concentrate on the more marginal variables.

Eventually, the system might assist in the selection of the variables of interest and the assignment of the experimental treatments. The system would also analyze the data, and terminate the experiment when sufficient precision was attained.

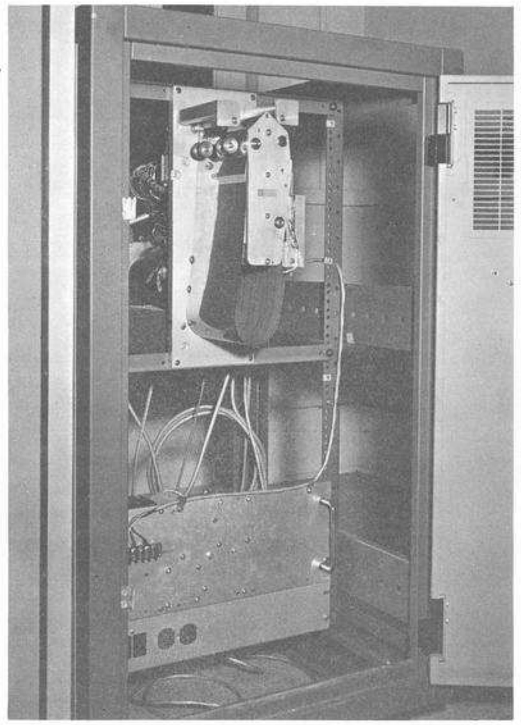


Figure 1 The random access audio-unit built by Westinghouse

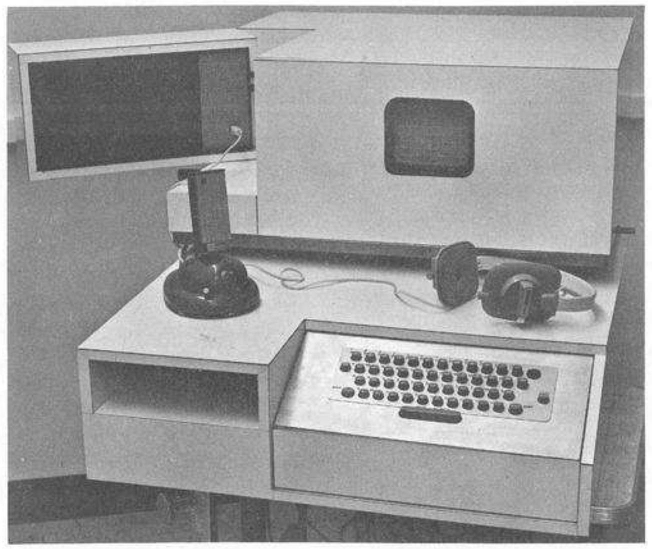


Figure 2 The basic student station with keyboard, scope speaker, earphones, and microphone.

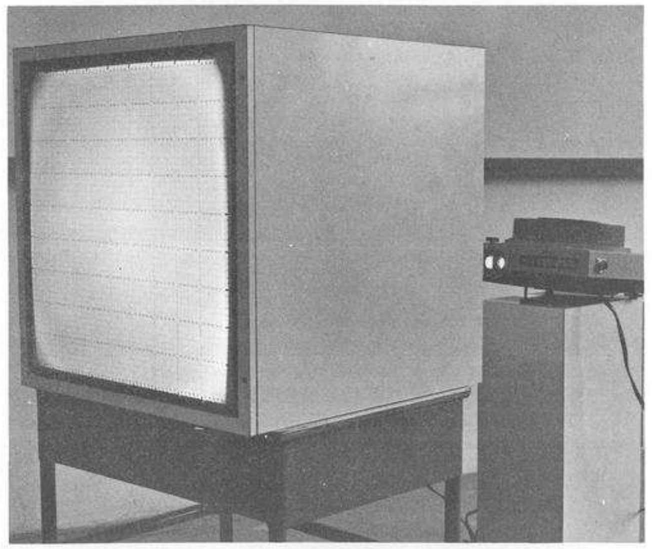


Figure 3 The touch sensitive display with random access slide projector.

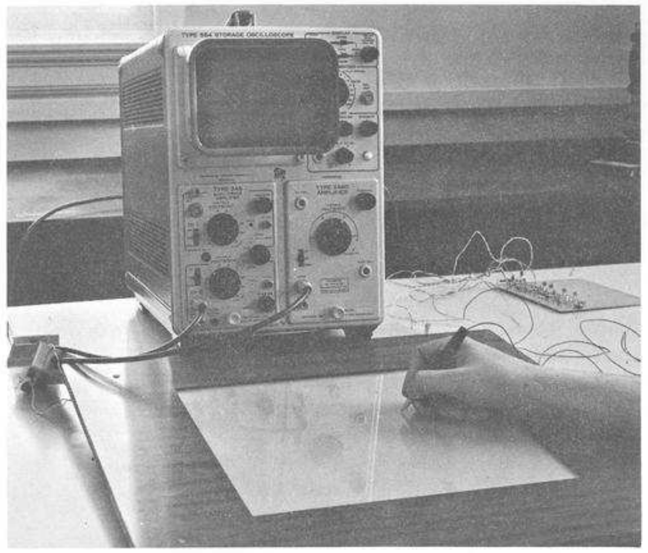


Figure 4 The graphical input surface.

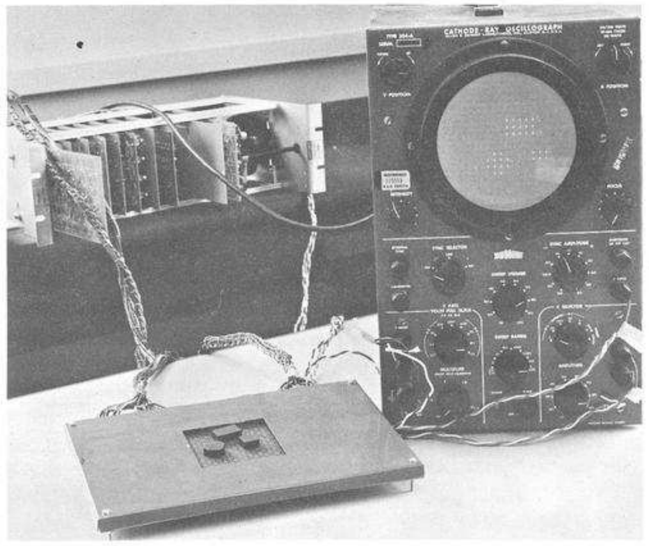


Figure 5 A small developmental model of the manipulation board.

DISPLAYS FOR STUDYING SIGNAL DETECTION AND PATTERN RECOGNITION

T. Booth, R. Glorioso, R. Levy, J. Walter, and
H. Kaufman
University of Connecticut
Storrs, Connecticut

ABSTRACT

The use of a unique computer-controlled (PDP-5) CRT display system with light pen facility in the study of a wide range of signal detection and pattern recognition problems is described.

The displays used to study the capabilities of the human operator to detect signals embedded in noise are described and illustrated. In addition to illustrating displays to study the effects of various signal and display parameters, pre-processing and real-time, operator-directed (via light pen) processing are also shown.

The use of this system for tachistoscopic stimulus presentations to study basic human information processing capacities is also described.

I. Introduction

The Electrical Engineering and Psychology Departments at the University of Connecticut are engaged in an interdisciplinary research project to study how man processes visual information. A series of experiments are currently underway which involve presenting the human operator, in real time, various classes of visual displays and measuring his ability to process the information contained in them. This information may or may not be contaminated with background noise.

Because of the large amount of information which must be handled during each experiment, it is impossible to employ classical psychophysical display and measurement techniques to present these displays and collect data about the operator's reactions. This problem has been overcome by using a PDP-5, a high speed CRT display, and several special purpose I/O devices as a central data processing and control system. This paper will describe the various types of experiments which are currently being conducted and illustrate the advantages gained by using this automated display and data gathering system.

II. Basic Research Problem

The basic problem under investigation concerns man's ability to use and process various types of visual information. At present two specific types of problems are being investigated.

The first area concerns the ability of an operator to detect and/or recognize a pattern in a noise background. In this area a series of experiments are being conducted in an attempt to develop a model which describes the operator's detection and dynamic information processing capabilities. Figure 1 illustrates a typical display an operator might see during a given experiment. His task would be to decide which column in the display has the largest number of intensified points. Once he makes this decision, he is asked to identify the line by use of a light pen.

The second area of interest involves measuring the amount of information that an operator can extract and utilize from a noise-free display. In a typical experiment, the operator would be shown a series of patterns selected from a given class of patterns. After seeing each pattern he identifies the pattern that he thinks he saw by pushing the appropriate button on the operating console in front of him. Figure 2 shows one typical pattern that might be presented to the operator. By varying the number of component dimensions of the pattern, it is possible to obtain an idea of how much information the operator extracts when making his decision.

Traditionally, research of this type has been conducted by using a tachistoscope which is a small display device for presenting visual stimuli under controlled conditions of visual field, illumination, and exposure. An experimenter must be present during the whole experimental session to control the operation of the tachistoscope and to record the subject's responses. In addition, all of the displays must be fixed a priori, since it is impossible to present dynamic displays to the operator. Such an experimental process suffers from several very severe limitations. Most notably, when dealing with problems of the type described above, it is not uncommon to require over one thousand different displays in order to gather statistically significant data.

The first major problem involves the mechanical process of generating, storing, and presenting a large number of displays when a tachistoscope is used. If a thousand different displays are used, the experimenter must generate each display and record it on a card. During an experimental session he must load each card into the projector and control the presentation of the card, which takes approximately 20 seconds. Therefore, a minimum of 6 hours would be required to run an experiment involving a thousand displays. For much of this time, the subject is waiting for the next display to appear; consequently, he quickly loses interest in performing his assigned task.

The physical task of generating a large number of display cards is also a major problem. It is possible to use a digital computer to generate the data required to specify each of the patterns needed for a given experiment. However, it is still necessary to transfer these computer generated data onto the display cards used by the tachistoscope. This is a tedious and time consuming task.

Even when the mechanical problems associated with preparing the experiment have been solved, the problem of collecting and processing the experimental data must be considered. In the types of problems under investigation, it is necessary to record parameters such as the subject's identifying response, the time he took to respond, the position of the pattern or patterns he responded to, and the statistical properties of the display itself. To collect this information using standard techniques can become a monumental, if not impossible, task. In some experiments, the experimenter must also have access to the display in order to make measurements on the display itself to determine which pattern the subject indicated. When this occurs, it not only disturbs the subject, but also increases the amount of time needed to perform a given series of trials.

Because of the above difficulties, many interesting problems concerning the way in which a human processes visual information have been difficult to investigate. In fact, without the use of the on-line system described in the next section, it would have been impossible to carry out some of the experiments which will be described in sections IV, V, and VI.

III. Computer Control System Used to Supervise Experiments

To overcome the experimental problems described in the last section, the computer controlled system illustrated in Figure 3 was developed. The heart of this system is a PDP-5 computer coupled with a special-purpose high-speed CRT display. This display system, which is described in detail elsewhere, can present flicker-free 72 x 85 dot patterns of the type shown in Figure 1 and Figure 2. These patterns, which are under direct program control at all times, can easily be modified to satisfy any desired experimental conditions.

For a given experiment, an experimenter can use any or all of the following I/O devices in conjunction with the CRT display system.

a. Light pen - The light pen is automatically tracked and controlled by logic built into the display system. Whenever needed, the position of the light pen can be read by the main program.

b. Noise generator - A wide-band Gaussian noise generator is available which can be sampled by the A/D converter under program control to produce a Gaussian distributed statistically independent variable.

c. Clock - An external program controllable clock can be used to measure total elapsed time and/or incremental changes in time.

d. Push button matrix - A 2 x 4 array of pushbuttons is available for the subject to indicate his responses to a given set of experimental conditions or to initiate a change in the display under light pen control.

To perform a particular experiment, all that an experimenter must do is write a program which will present the proper sequence of displays to a subject and measure the appropriate responses. During a given session the collected data may be stored in the computer until the end of the experiment. At that time the data is processed and the results are typed out on the teletype. The flexibility of this system is mainly limited by the skill of the experimenter in designing his program and identifying the data he wishes to measure. Once the program has been developed, the actual running of the subject is handled almost entirely by the computer. Thus, once the experimenter has started the program, he is free to carry out other work while he waits for the end of the session.

In order to illustrate how this system is used, several of the experimental programs which have been run by this system will be described in the following sections. No attempt will be made to explain the reason for the experiments since this information is available in the papers cited in the references. The main emphasis will be upon showing how the experiment was programmed and illustrating the savings in time which were possible.

IV. Signal Detection Experiments

These experiments have been concerned with the ability of an operator to detect a target signal which is embedded in a noise background. Figure 4 illustrates a typical display shown to a subject in a given class of experiments. Under normal operating conditions the target path will not be as strong as shown in this Figure.

Two basic modes of operation have been considered. The first is the so-called "alerted operator mode" where the subject is told the location of a possible target and his task is to detect targets at that location. The second is the "unalerted operator mode" where the target location is not known and the operator's task is to locate a target, if present. Although these programs are similar, each has its

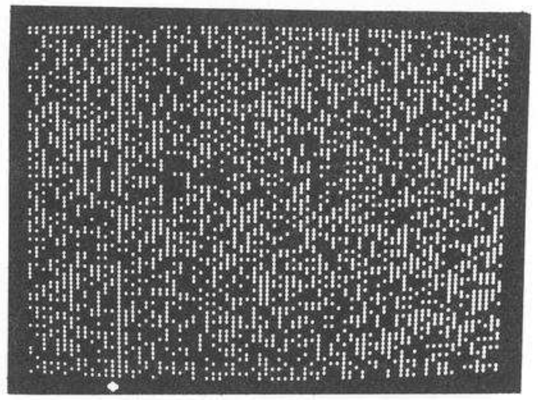


Figure 1
Signal Detection Display

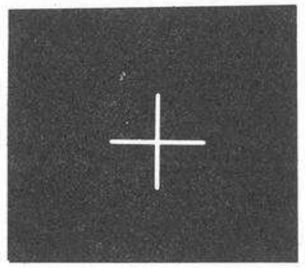


Figure 2
Information Transmission Display

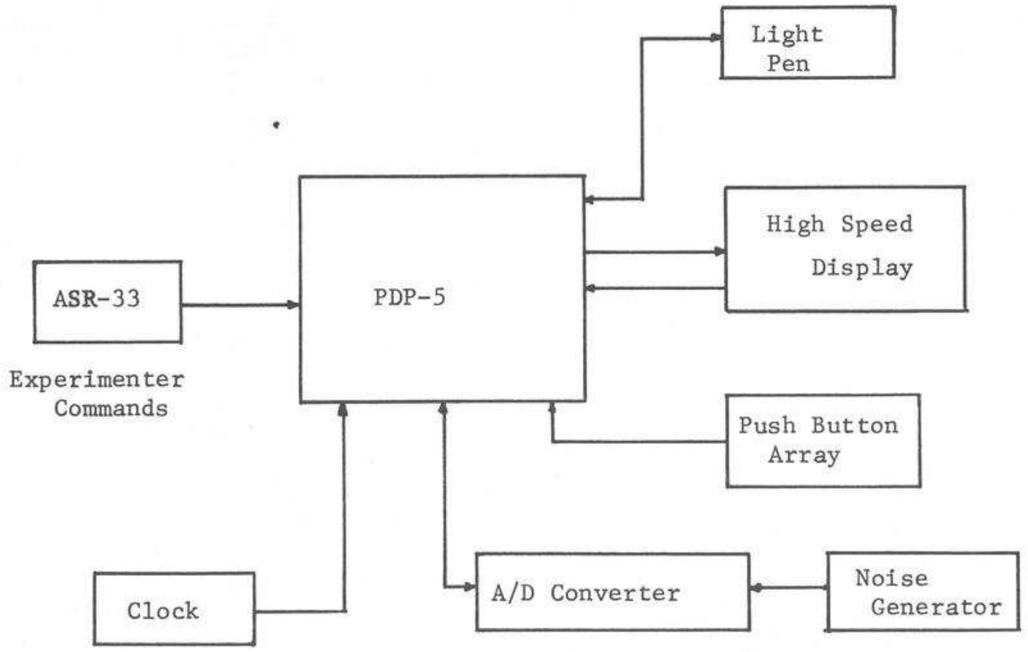


Figure 3
Block Diagram of Display System

own special requirements. They are alike, however, in the manner in which displays are generated.

To generate displays of the type shown in Figure 4, the system samples the external noise generator through the A/D converter. If the sampled value exceeds a predefined threshold value T_0, a one is stored in the display, corresponding to a dot in the display matrix at the appropriate location; otherwise, a zero, corresponding to no dot, is stored in the display matrix. The number of intensified points appearing in any part of the display matrix can be controlled by varying the value of T_0. To see how this is accomplished, it is necessary to consider the properties of the random number produced by the A/D conversion process.

Since the A/D converter operates from 0 volts to -10 volts, the mean of the noise generator is set at -5 volts and the standard deviation is set at 2 volts. The output of the noise generator is Gaussian distributed and its spectral density is flat from d.c. to 100 kcps. Under this condition the probability that the noise voltage will exceed the dynamic range of the A/D converter is .012. In order to insure that the noise voltage remains constant during the conversion time, a track and hold circuit is used between the noise generator and the A/D converter.

If a point is to be generated with a probability of intensification of 0.5, a threshold value of $T_0 = 4000_8$ which corresponds to -5 volts is used. If a point intensification probability greater than 0.5 is needed, a constant C_0 is added to 4000_8, or subtracted for a probability less than 0.5. For each trial during a given experimental session, a display is generated point by point and stored in core locations 6000_8 to 7247_8. A target path is produced by changing the point intensification probability for the points in the desired target path by adding the appropriate constant to the clipping level. Generation of the 6000 point display takes 1.8 sec.

The alerted operator experiments are usually conducted with straight line targets where the location of the target is indicated by an arrow as shown in Figure 4. The operator indicates the presence or absence of a signal by means of the push button matrix. After the operator's response, feedback may be presented as shown in Figure 5. An H corresponds to hit or correct response and an M corresponds to miss or incorrect response. A typical alerted operator experiment runs for 15 to 20 minutes during which 150 displays are viewed by the subject. The trial by trial data, as well as a compilation of the data after the run is complete, are usually printed and/or punched by the ASR-33. Previous alerted operator experiments using non-automatic techniques ran one hour and only 20 to 40 displays were viewed.

The display for an unalerted operator experiment is generated in essentially the same manner as was used for the alerted operator experiments except that the position of the target path is not indicated. Because of this, it is necessary to use the light pen as well as the push button matrix to communicate with the computer. A typical experiment, which gives the operator control over computer processing options will be illustrated.

The subject is presented a display such as the one shown in Figure 6 and he is asked to indicate which column in the display is the target column. The dot cluster at the bottom of the display is the light pen tracking square which the operator can use to carry out one of the following program-display options:

1. Any column can be eliminated by placing the light pen tracking square under it and pushing the appropriate button.
2. Any column can be checked against two preset thresholds by using the light pen and pushbuttons: If the column strength is below the first threshold, the column is eliminated; if it is between the first threshold and the second threshold, it is left unchanged; and if it is above the second threshold, a marker is placed under the column.
3. Any two columns can be compared by placing the tracking square under one column and pressing the appropriate button and then placing the square under the other column and pressing the button again. The weaker column is then eliminated.

Figure 7 through 9 indicate the results of applying these program-display options to the display illustrated in Figure 6. Figure 7 shows the same display after the column above the tracking square has been eliminated. The columns with marker lines under them in Figure 8 have been checked and are above the second threshold. Figure 9 illustrates the effect of comparing the column above the square and the strong column on the left of the display.

The unalerted operator detection task can be generalized by asking the subject to detect targets with curved paths. A curved path is shown without background noise in Figure 10, and with background noise in Figure 11. In this type of an experiment, the subject indicates the path of a possible target by tracing it with the light pen. If the operator goes off the path or indicates an incorrect path, all points in the correct target path flash on and off and a line appears at the bottom of the display. If the operator indicates the correct path, all points in the target path flash after the tracking square is below the last row of the display.

Most unalerted operator experiments require trial by trial printing and/or punching as well as run compilation printouts. Usually 50 trials are run in one hour in these experiments. This kind of experiment is almost impossible to perform without a computer-display system.

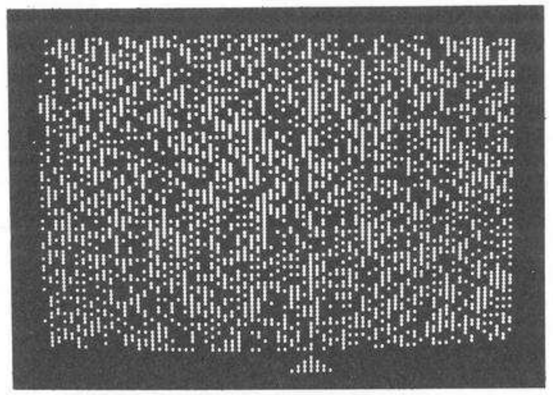


Figure 4
Alerted Operator Display

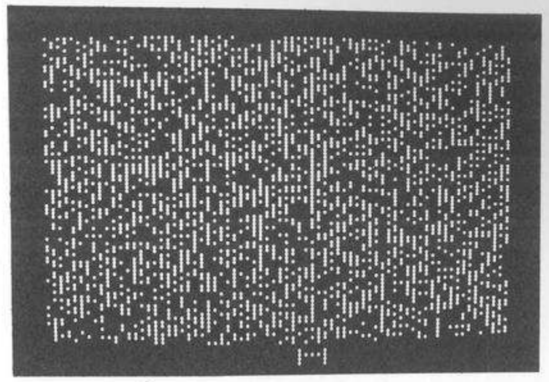


Figure 5
Alerted Operator Display With Feedback

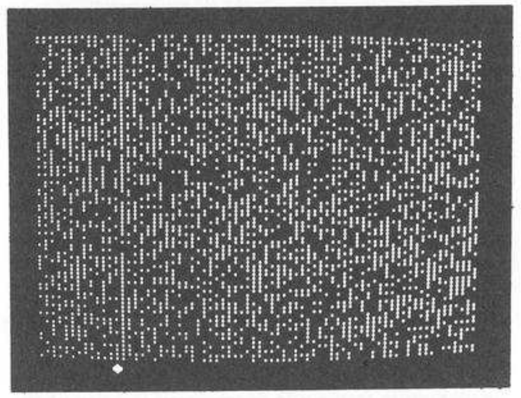


Figure 6
Unalerted Operator Display with Light Pen Tracking

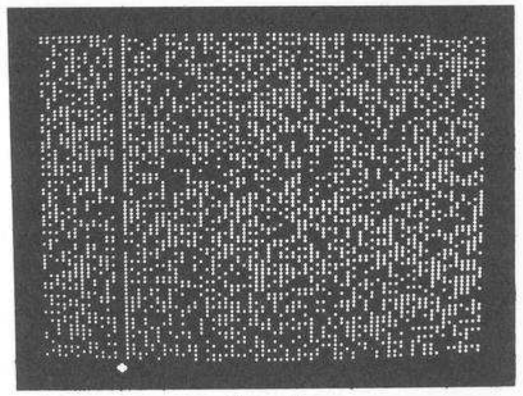


Figure 7
Column Elimination

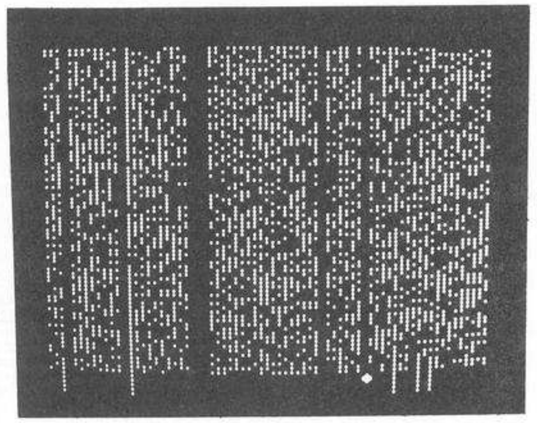


Figure 8
Column Check

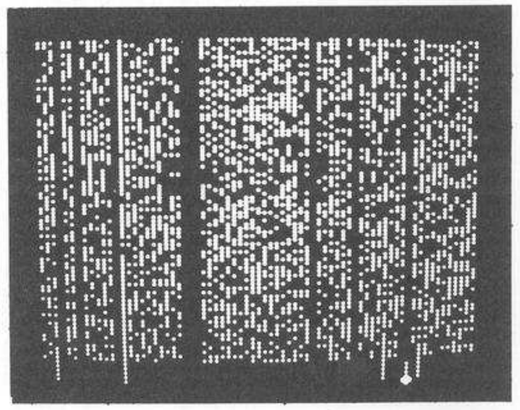


Figure 9
Columns Compared

Experiments which use these displays are described in references 2, 3, and 4.

V. Dynamic Displays

The detection problems discussed in section V concerned the ability of a subject to detect a target path in two dimensions. Such displays are common in radar and sonar systems. However, in systems of this type it is usually necessary to process three dimensional information consisting of two spatial dimensions, say X and Y, and time. This section will discuss a display capable of presenting information of this type to an operator. These displays are being used to investigate the ability of a subject to detect targets in a dynamic display situation.

The dynamic display presentation scheme uses a two dimensional display and time to present the three dimensions in such a way as to take advantage of the human eye-brain system's spatial and temporal integration capabilities.

Figure 12 is a representation of the three dimensions. At any given point in time, current X and Y information is assumed to be available from a device such as a sonar or radar receiver. This information can readily be shown on the two dimensional display tube; in this case, a signal from a target would not appear as a column, as it does in a two dimensional display, but as a cluster of cells in the matrix. As in the digitized displays shown in section IV, the difference between the target area and the background is that the former is generated by sampling from a distribution with a higher mean than that which produces the latter. Thus, the matrix containing a target will have a clustered subset of its cells within which the probability of a dot appearing is greater than in the rest of the matrix.

This current information could be put on the 2-dimensional display at time T=t and remain unchanged for one time interval (i.e., until T = t + 1) at which time it could be replaced with the new information.

However, at time $T = t_B$, all the information from some previous time, t_A, up to the present can be stored in the computer memory. The number of time intervals between t_A and t_B, of course, depends on the number of points along the X and Y dimensions and the size of the memory. Having this information available makes the following possible: in the interval between t_B and $t_B+\Delta t$, all the successive X, Y displays from t_A to t_B can be shown in rapid sequence, rather than having a static display of the information from t_B alone. Similarly, in the interval $(t_B+\Delta t)$ to $(t_B+2\Delta t)$, the information from $(t_A+\Delta t)$ to $(t_B+\Delta t)$ is shown sequentially, etc. The last frame in each sequence is shown a little longer than the others while the frames are shifted up in the memory in preparation for the next sequence and the information from the present time is brought in. The effect is that of a moving window scanning the T dimension for some distance, jumping back to near its initial position, and moving on again, etc.

The human observer looking at this dynamic X, Y display sees a matrix of randomly flickering dots, but in the target cluster there are more dots; hence, they appear to flicker less frequently. In addition, the movement of the target cluster through the matrix, which would be hard to see in sequential static displays is made to stand out as smoother movement, as in movie film.

A program has been written for the PDP-5 which presents displays in this manner. Input parameters are: for the display: (t_B-t_A), Δt, frame rate; for the signal: matrix size, target cluster configuration, path, P (dot/noise), P (dot/Target).

This program requires about 350 core locations, leaving about 3000 for storage of display information. Thus, 250 12 x 12 matrices can be stored or 27 36 x 36 or only 5 of the maximum size 72 x 85. Because of these memory limitations, an expanded core or magnetic tape unit would make the program more versatile.

In operation, the sequence of events is as follows: First, the program generates as many frames (X, Y displays) as will be shown in one sequence and stores them. Next it transfers the first frame to the display area of core, hence onto the screen. Then it enters a wait loop whose length is determined by reading the switch register. After the wait loop, it transfers the next frame onto the display, etc. Thus, the frame rate is determined by the length of the wait loop and the amount of time required to transfer one frame; hence, the maximum frame rate decreases as the size of the frame increases.

After displaying all the frames in a sequence, all the frames in core are shifted up, as many frames as specified, and the new frames for the end of the next sequence are generated and stored. The program then jumps back and shows this new sequence.

A movie has been made with some examples of 12 x 12 and 36 x 36 frames shown in this manner. Figure 13 is extracted from this movie to illustrate a dynamic display when a very strong target is present, and Figure 14 shows a typical path which the cluster may follow in time.

VI. Pattern Recognition and Information Transmission Experiments

The use of the display system described in section III for studies of human pattern recognition and information transmission capabilities and characteristics has lead to great experimental flexibility and ease of data collection. Two research programs are currently underway in this area.

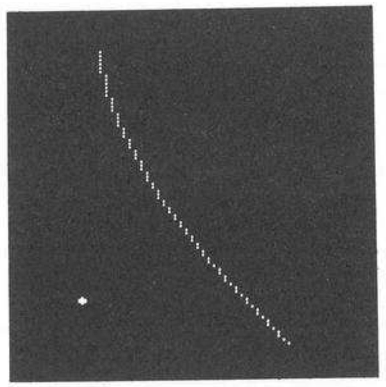


Figure 10
Curved Target Without Noise

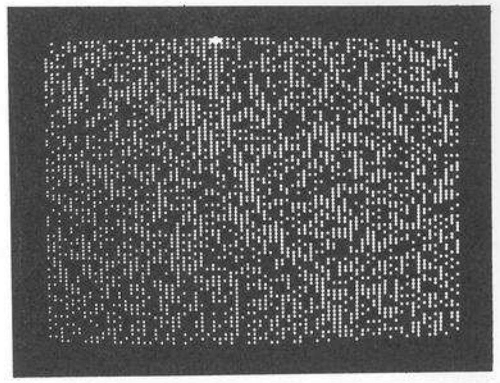


Figure 11
Curved Target With Noise

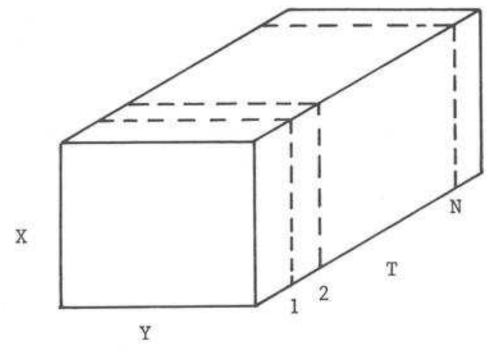


Figure 12
Representation of Three Dimensional Display

The first program is concerned with a problem related to pattern recognition, namely how do stimulus dimensions, such as size, hue, linear extent, and brightness, influence a subject's ability to identify a particular pattern from a set of possible patterns. Although, the operator can judge as alike or different 1000 or more stimulus values sampled from along one of these dimensions, he can only identify the equivalent of approximately 7 values. One only need witness the speed and accuracy with which the operator identifies the letters of the alphabet and other complex patterns to see that the number of dimensions which make up a pattern are an integral part of the operator efficiency at pattern recognition.

A requirement for studying how dimensions combine is ease of manipulation and generation of dimensions relevant to the particular research problems. In previous research on dimensionality [5], the dimensions of horizontal and vertical linear extent were used. Information transmission was compared for stimulus values along the single dimensions alone and for stimulus sets formed by combining values along both dimensions. Stimuli, which were prepared with black tape (approximately 2" x 3/32") on 8-1/2" x 11" white poster board cards were presented with a Gerbrands tachistoscope. Because of manual presentation and interchanging of stimulus cards about 45 minutes were required to run 125 trials. From the subjects viewpoint, relatively little time is spent in his assigned task of identifying stimuli compared to the large amount of time idled away while the experimenter carries out his duties.

The CRT offers a means of displaying stimuli varying along these same dimensions of horizontal and vertical linear extent. The dots used to generate the patterns with other signal detection studies can be transformed so as to give the appearance of a solid line. When a single column and/or row is filled with intensified points, the gain on the X and Y axis controls turned all the way down (which gives a density of approximately 15 dots/cm.) and the focus and astigmatism adjusted appropriately, stable solid lines can be presented. The length of line is manipulated by the number of points displayed. In the present experiments, lines consisting of 33, 65, 67 and 69 dots are used. A third variable dimension, intensity, was obtained by putting 4 preset intensity levels under I/O transfer control. The levels of intensity are preset in the sense that the experimenter sets each level to any desired value within the dynamic range of the CRT. These levels are then available under program control.

Stimulus patterns are presented for short exposures (approximately .19 sec.) which leads to inherent problems. With short exposures, display duration must be calibrated according to full sweeps so that the display dies out uniformly. The exposure of .19 sec. corresponds to 7 full sweeps. Besides the flicker-free characteristics of the display system, the high frame rate allows relatively sensitive control and manipulation of the exposure times. Although a medium persistence phosphor is in use, it is essential to eliminate any persistence, since it leads to uninvited cues for stimulus identification during the task. An obvious solution to this problem is to have a high level of ambient light. A requirement for the ambient light, however, is that no reflections be cast off the front of the CRT. A reasonable level of ambient light is also desirable since it is perceptually unsound to have the subjects visual field switch, continuously, from light to dark as displays go off and on. Furthermore, control of the area of the subject's visual field would be advantageous since it would also prevent the use of extraneous cues in carrying out the assigned task. The requirements just outlined have been met by a 19" wide x 15" high x 22" long adapter that is easily hung over the face of the CRT.

A photograph of a subject's eye view of the face of the CRT through the adapter is shown in Figure 15. Nonreflecting lighting of the face of the CRT is achieved by a string of 7 watt lights placed around the front frame of the adapter. It would not be at all unreasonable to say that the CRT has been transformed to a super tachistoscope with all the advantages of on-line computer control.

The actual preparation of stimuli is under program control. On a trial by trial basis the appropriate horizontal, vertical, and intensity components of stimulus are either set up in the display area of core storage for horizontal and vertical dimensions or the I/O transfer pulse sent out. The operator pushes a button which exposes the stimulus and then pushes an appropriate button recording his response. Stimulus and response data are pre-processed and stored over a series of approximately 120 trials. These same 120 trials, including printout time, take approximately 10 minutes to run compared to the 45 minutes required to run 125 trials using the Gerbrands tachistoscope. Furthermore, changing stimulus conditions is extremely easy since the only input to the control program besides number of trials (the sequences of randomizations are prepared independently on each run by using sampling techniques with the noise generator) is a specification for each of the stimuli and the value along each of the component dimensions.

A second program is investigating the amount of information that a subject can retain and recreate after he has viewed various classes of displays. In 1956, George Miller [6] described means of remembering information in which the human takes a number of symbols to be remembered, groups them, and remembers the names of the groups. He then decodes the groups to the individual components when asked to recall them. Miller referred to this as "chunking." All of us use this procedure when we remember a 12-bit binary number by storing its octal equivalent. Miller's implication was that human memory capacity can be multiplied tremendously by taking larger and larger groups or chunks. However, all the examples he cited

involved sequential presentation of the symbols and the time required to do the coding fits in between symbol presentations. Simultaneous presentation, however, is a different story. In this instance, Hyman and Kaufman [7] have shown that memory capacity is defined in terms of amount of information stored, not number of units, as Miller implied. Also, it has been shown that humans tend to adjust input and output rates so that the bit/sec. rate of throughput is relatively constant. An experimental program is currently underway which combines this type of research with simultaneous presentation by varying the information loading of the symbols used, the length of time they are presented, and whether or not chunking is used.

The display used is a matrix of 64 symbols randomly chosen from 4 different ones, as illustrated in Figure 16. It contains 8 rows of 8 symbols each, each symbol being 13 dots arranged in the cells of a small 5 x 5 matrix. These have been shown to be readily discriminable. The pattern of dots in each row of each different symbol is stored in a number of locations in the core, these locations differing only in the rotation or position in the register that the bits occupy. This is done because each column in the 8 x 8 matrix corresponds to a different part of the word being displayed. That is, the first column corresponds to the left end of the first word in the row, the second column corresponds to the right end of the same word plus the left end of the second, etc. In generating the matrix, the program samples the noise generator twice to get a random number between 0_8 and 3_8 corresponding to one of the symbols. From this number and the position in the row into which the symbol will be put, the program generates the address of the location where the particular symbol is stored in the correct rotation. Having thus generated the 8 addresses required for the first row, it then transfers the symbols to the display area of the core, repeating this 8 times to get the full matrix.

When the operator is presented with this display, he tries to recall as many of the symbols as possible and pushes appropriate buttons in the response push button matrix. Each button generates a code between 0_8 and 3_8, so the program keeps track of whether each response is right or wrong and when the operator pushes a fifth button to indicate he can respond no more, the program punches out the number of responses he made, the number that were correct, the number correct after correcting for guessing and the time required to make the responses. The codes for the responses and 64 symbols on that trial are also available. This program takes about 600 core locations and generates these display matrices as fast as the operator can push a button to request a new one.

The use of this system for this type of experiment has tremendous advantages over the use of a manual tachistoscope as was done in refs. 5 and 7. Once the program has been written, the actual symbols used can be determined or changed in a matter of minutes rather than the hours required to prepare cards with appropriate symbols as required by the tachistoscope. Also, the running of the subject is automated with a hard copy of the data available at the session's end. This frees the experimenter to do other work during the session, as opposed to continuously manipulating cards for the tachistoscope and writing down responses, as well as eliminating the need for later punching of data onto cards.

Acknowledgement

Acknowledgement is due the Office of Navel Research which supported this research through a prime contract NOnr 2512(00) with General Dynamics/Electric Boat as part of the SUBIC (Submarine Integrated Control) program.

References

1. Brazeal, E. H. and Booth, T. L. 1965. "A High Speed Man-Computer Communication System." Spring. DECUS Symposium, May 1965. Cambridge, Mass.

2. Brazeal, E. H. and Booth, T.L. "Operator Noise in a Discrete Signal Detection Task." IEEE. Trans. on Human Factors in Electronics, Vol. HFE-7, No. 4 Dec. 1966, pp. 164-173.

3. Levy, R. M., Kaufman, H. M. and Walter, J. R. "Experimental Investigations of Operator Decision Processes: II. The Effects of S/N and Bearing Resolution" Presented Eastern Psychological Association Convention. New York, April 1966.

4. Glorioso, R. M. and Levy, R. M. "The Effects of Step Transitions in Signal to Noise Ratio and Feedback on Dynamic Operator Decision Behavior." Presented Eastern Psychological Association Convention. Boston, April 1967.

5. Levy, R. M. "The Effects of Stimulus Dimensionality on Information Processing" In Proceedings of the 73rd Annual Convention of the American Psychological Association, Washington, D.C. APA, 1965. pp 42-43.

6. Miller, G. A. "The Magical Number Seven, Plus or Minus Two," Psychological review. Vol. 63. 1956. pp. 81-97.

7. Hyman, L. M. and Kaufman, H. M. "Information and and the Memory Span." Perception and Psychiphysics, Vol. 1, 1966. pp. 235-237.

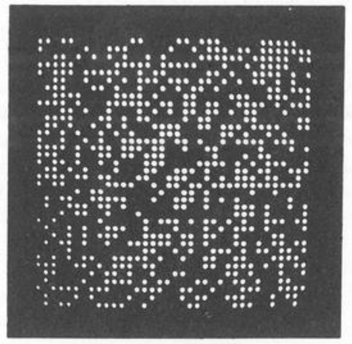


Figure 13
One Frame of a Dynamic Display

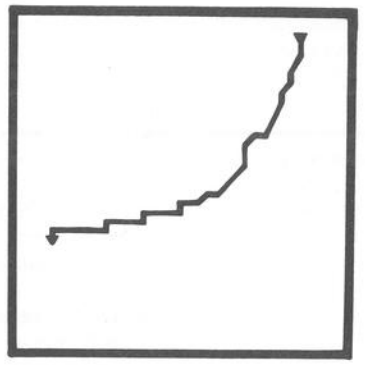


Figure 14
Typical Target Path as a Function of Time

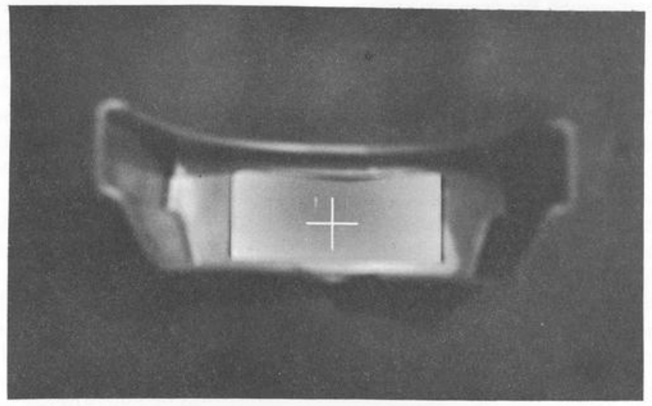


Figure 15
Subject Eye View of CRT

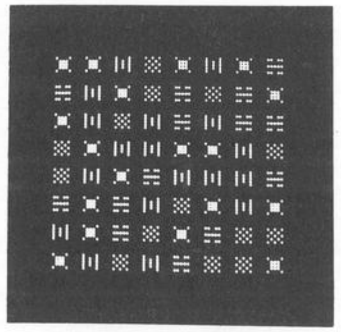


Figure 16
Short Term Memory Display

MESSAGE SWITCHING SYSTEM USING THE PDP-5

Sypko W. Andreae
Lawrence Radiation Laboratory
Berkeley, California

ABSTRACT

At the Center for Research of Management Science at the University of California, Berkeley, recently a message switching system was constructed for research of human behavior in game situations.

The system uses a PDP-5, a 630 System with ten Teletypes, a DEC tape unit, and an IBM compatible tape unit. The basic laboratory layout entails many soundproof rooms in each of which the subject can use one of the Teletypes. The controlling program is essentially a simply time-sharing system of which the organization is outlined in this paper. Among the many advantages of the existing system over the previous manual method are improved traffic intensity, better record keeping, network control by the experimenters, etc. Two examples of games used in these experiments are given.

In the Department of Business Administration on the campus of the University of California, Berkeley, California, a laboratory was constructed recently named "Center for Research of Management Science," (CRMS). Most of the research in this laboratory is aimed at the behavior of people in business situations. In the experiments being conducted, questions are asked such as: On what does a businessman base his decisions? Does he listen to qualified advice, and if he doesn't, why not? What causes one person in a team of businessmen to be dominating?

To shed some light on how experimenters go about setting up their experiments, I will give the following hypothetical example: A team of five people run a fictional company in a simulated market situation. The five functionaries are a Chief Executive, a Purchasing Manager, a Comptroller, a Sales Manager and a Production Manager. Beforehand, the subjects (in general students), are supplied with information about the status of the company they are going to run. In general, the company sells three products and they start from a situation which is illustrated in financial statements. One game may take a whole afternoon, and consists of half-hour periods. During each period all team members communicate with each other via communication channels. The communication network may or may not be under certain constraints. There are several reasons for these constraints. For instance, it would be very difficult to find causes for certain kinds of behavior when communication was completely free. In a real situation, the businessman bases his decision on many facts coming from many sources, not only his letters, the telephone, but also the radio, TV, the newspaper, cocktail parties, etc.

In the experiments, the subjects are not able to see or hear each other; they can only communicate via the written word. First, this took the form of written notes which were picked up and delivered by research assistants, who kept a kind of running mail service between all the subjects. Now this is being done with Teletypes connected to a PDP-5.

The experimenters are certainly aware of the fact that the constraints being put on the communications medium creates a situration which does not compare with the reality of business life. However, when one's aim is to search for certain well-defined and specific phenomena, these constraints do not necessarily invalidate the results of the research.

The properties of the market in which the company is being run are well known, and therefore an optimal solution for the operation of the company exists. The subjects, in general, do not know this solution. Every half-hour the team has to make a decision about purchasing, production, etc. The decisions are then fed into a computer program which prepares financial statements as a result of the decisions of the team and the properties of the market. During the next period the subjects can see the results of their previous action.

The standard model of this experiment is very often modified to study certain specific effects. For instance, sometimes a stooge is inserted. Unknown to the others, one of the team members is informed about the key to the optimal operation of the company. Once the game is started he is not to inform the team members of his specific knowledge, but nevertheless he is to convince the team members how to make certain decisions. It is interesting to see that in most cases the team members do not accept his advice, however well-founded it may be.

During the summer an experiment was performed in which an operations research group was included in the team. The five executives were student subjects; the operations research group (which was in touch with all team members via Teletype), consisted of three highly skilled professionals in the business trade. Still, it was obvious that most of their advice was not being accepted, and in many cases their suggested decisions were vetoed by the Chief Executive who happened to have different "feelings" about the matter. I don't intend here to elaborate on the research itself. These remarks were merely an introduction to the atmosphere of this project.

The use of the Teletypes over the use of little slips of paper was an obvious improvement. Of couse the costs of a small computer and an interface for Teletypes, the Teletypes themselves, the cabling in the building, etc., is considerable and therefore it is obvious there must be very good reasons to offset these expenses. Without going into detail, I could mention a few of those reasons.

The time sequence of the messages needs to be recorded. There is also a need to process communication patterns as they emerge from the experiment. Also, several efforts have been made at CRMS to use informal language. This is a language of which all the rules are known and in general, this takes the form of a predetermined set of sentence elements which a subject is to use only according to a predetermined syntax. Messages furthermore need to be stored and later recorded together with all pertinent experimental data.

Previously, much of the preparation for analysis of the experimental data was done by scores of secretaries who transcribed all the communications, counted communications phenomena, and tried to find the proper time sequence.

Figure 1 shows the configuration of the equipment now being used, and the remainder of this paper will develop several reasons for the acquisition of this particular configuration. Instead of a small computer like the PDP-5, one could probably use a simple electronic switching network when communication is the only function to be performed. A stored program processor as a part of the system obviously can perform many more desired tasks, and will make the whole system more versatile. Unlike the normal telephone system (one-to-one communication), this system was required to have fanned-out communication; that is, any one Teletype should be able to communicate to one or more others. Once this is allowed for, one has to deal with the difficulty of unavailability of the receiving stations. The computer can take care of this by storing the message temporarily and transmitting the message as soon as the unavailable stations are available.

Storage on IBM compatible magnetic tape is especially convenient in view of future off-line processing on a larger machine.

Figure 2 shows a memory map and in the following I will elaborate on the construction of the software that controls the message switching system.

The memory is roughly divided in two parts of which one-third contains the different control programs and roughly two-thirds is reserved for temporary storage of messages of all of the stations. Each station has its own message area which can contain about 450 characters. The control programs take care of a time-shared processing of characters received from the Teletype stations. The characters received from the Teletype stations can be the result of a keyboard action at the station or from the transmission of one character from the computer to the station since all Teletypes are connected to the PDP-5 in a half-duplex fashion. The interface for the ten Teletype stations is made by DEC and is called 630 Data Communication System.

In the use of a half-duplex system, one must remember in the program whether a station is being transmitted to or is being received from. This problem is solved by the use of status words. Each station has a status word from which the program learns what to do with a received character. But this is not the only function of the status word. It also serves as a memory for the particular phase of the procedure in which the station happens to be. I will elaborate on this procedure later.

The total program can be divided in two main parts: SWITCHYARD and all consequential routines, and DISTRIBUTOR. On the upper part of the memory map is SWITCHYARD, which is responsible for processing incoming characters according to the information in the status word and also for transfer of control to the proper part of the remaining software system. DISTRIBUTOR tries to initiate the transmission of stored messages to these Teletypes, in case no characters are coming into the PDP-5.

FILTER is a routine which executes certain prepared constraints on the communication patterns between stations.

A software clock updates the time every second. On the same page is an interrupt service routine for other devices. The routine specifically responsible for the transmission of messages is TX#SUB, the transmitting subroutine. The routine responsible for the construction of a message as it is received from a station is RX#SUB, the receiving subroutine. There is a program which consists of the controls for the D-2020 tape unit. It includes all the timing and length control of records that can be both read and written and it can generate end-of-files. The remainder of the memory is used for the ten station message areas.

Before the experiments start, however, the experimenter has an opportunity to set them up as he likes. He therefore enters a conversation with the PDP-5: the PDP-5 asks for certain information from the experimenter, which, after his response, is stored in the memory. For instance, the program will ask the experimenter to specify with

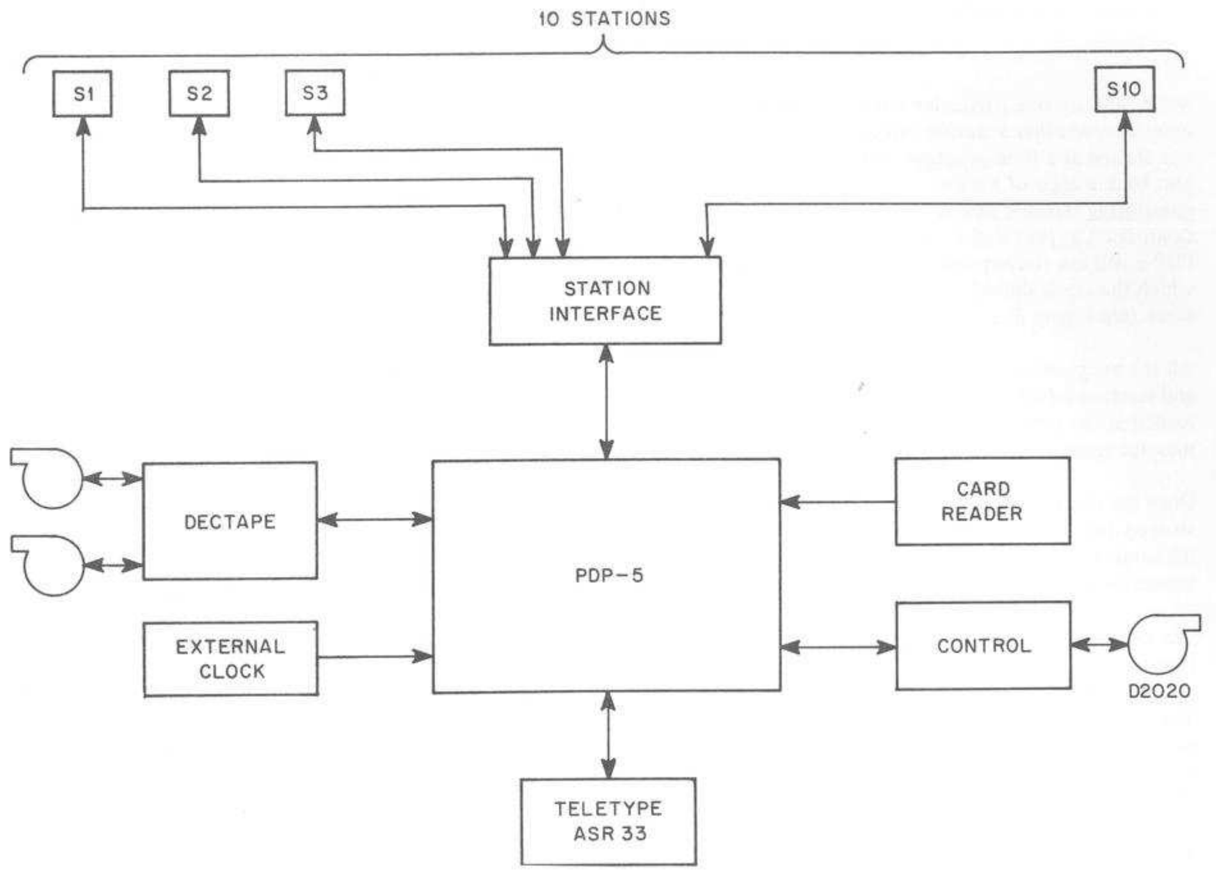


Figure 1 Management Science Laboratory System

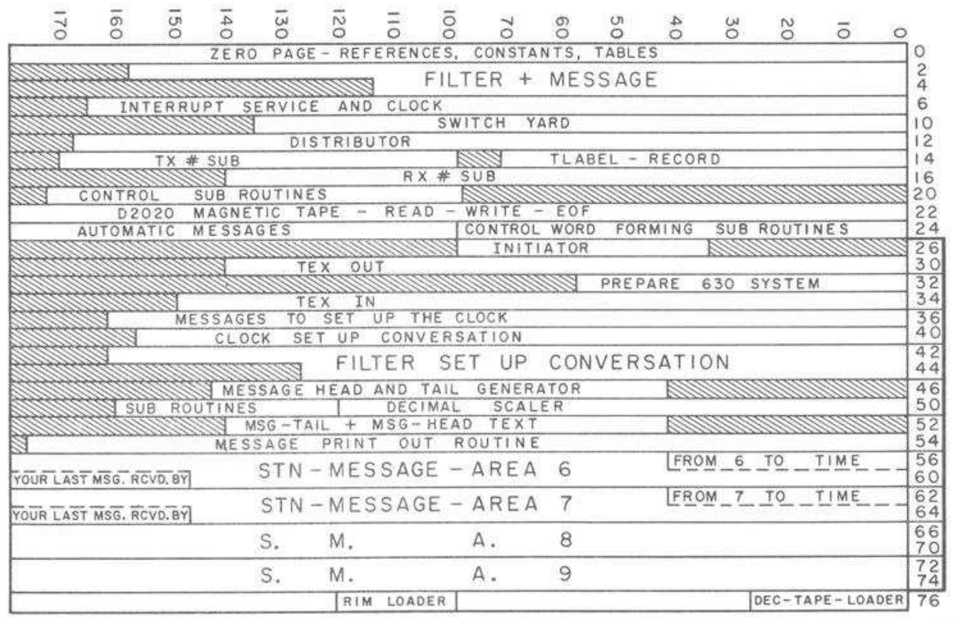


Figure 2 Memory Map

which stations one particular station is allowed to communicate, whether a station can only communicate to one station at a time, whether each station should be sent back a copy of his own message, and whether certain monitoring stations should be added to the requested destination as provided by the station of origin. The PDP-5 will ask the experimenter to type in the time at which the clock should start and then finally to start the clock (see Figure 3).

All the programs necessary to control this conversation and to store information from the experimenter are loaded in the part that is used later by the stations as message areas.

Once the clock starts, all conversational routines are destroyed and the message switching program is in control. All subjects at the stations start their communications according to a certain procedure.

The next figure shows what the subject sees on his Teletype as a consequence of conversation with the message switching program. After hitting carriage return on the Teletype, the computer answers, "TO WHOM"? The subject types in the station numbers that he requests as destinations for his message, delimiting the string of station numbers with a vertical arrow (used throughout this system as a MESSAGE DELIMITER). The program ignores anything other than numerals during this part of the procedure. As soon as the program detects the delimiter, it invites the subject to start his message by the response "GO AHEAD." He can now enter his message in standard English and completes it by typing the delimiter. The program then adds certain information to the message now residing in the core memory (from what station, to what stations the message is going, and the time), after which the program proceeds to write a record on magnetic tape containing this message.

Figure 5, a simplified flow-diagram of the program, shows these actions. After initiation, the system continuously checks the flag of the 630 system. As soon as a character arrives, the status is checked, and when the character is recognized as a carriage return and the status word for this station equals 0, the message "TO WHOM"? is transmitted to that station.

Note that SWITCHYARD only initiates the message. Transmission subroutine TX#SUB takes care of the transmission of the message until it finds the delimiter. The status word is then modified to indicate completion of this part of the procedure.

The DESTINATION ROUTINE assembles the incoming numerical information and stores the result in the REQUESTED DESTINATION WORD (RDW). After the delimiter is detected, the message GO AHEAD is initiated and the status word is modified accordingly.

The next character coming in is processed by RX#SUB, the receiving subroutine. When RXSUB detects the delimiter, control is transferred to the routine called TLABEL.

In the top of the flow diagram next to node D is the DISTRIBUTOR which starts transmission of messages residing in the station message area if the destinations are available. Once it is successful in initiating a message to an available station, a special word called the DESTINATION WORK WORD is updated. When DWW is updated, the bit corresponding with the station to which the message was being initiated is subtracted from the DWW. This word (DWW) contains all destinations for one particular message. Another function of the DISTRIBUTOR is to send a message (YOUR LAST MESSAGE IS RECEIVED BY . . .) when all destinations of a particular message have been reached.

The receiving subroutine, RX#SUB stores the characters, two characters per PDP-5 word, in the message areas of the stations. It accounts for the load pointers, pack words, etc., guards against overflow of the message area, handles special characters, and corrects the status word at the end of the message. Most characters are trimmed to a code which consists of six bits per character. The 6-bit code is formed by subtracting 240_8 (octal) from each Teletype character. The first character to be processed by the RX#SUB is placed in the pack word which belongs to that particular station. In the meantime RX#SUB can handle characters of many other stations before it returns to the previous station to handle its next character. The pack word is retrieved, the next character is placed in the left half of the pack word, and this word containing the two characters is stored in the message area. This packing method is of great importance for economical utilization of the available memory space.

A character set of 64 characters may not be enough, however, although the frequency of the use of characters which do not fall within this group of 64 is very low. Therefore, RX#SUB as well as TX#SUB have a special arrangement to process characters not belonging to this group of 64 (for example: carriage return and line feed). One complete word is storing these special characters.

The program FILTER is essential for many of the proposed experiments in this laboratory. FILTER modifies the destinations of each message before the distributor initiates transmission of the message to one of the destinations.

The following are functions of FILTER.

1. Requested destinations are checked for the presence of one or more destinations, if the experimenter has specified that only one destination per transmission will be legal. If, nevertheless, the station has requested more than one destination, control is transferred to a routine called ILLEGAL. The ILLEGAL routine destroys the message built up in the message area and

```
TYPE 1 IF A MESSAGE MAY HAVE ONLY 1 ESTINATION - TYPE 9 OTHERWISE.

1

TELETYPE NO. 0 MAY SEND TO

01234↑

TELETYPE NO. 1 MAY SEND TO

012334///// 01234↑

TELETYPE NO. 2 MAY SEND TO

01234,PLEASE.↑

TELETYPE NO. 3 MAY SEND TO

43210↑

TELETYPE NO. 4 MAY SEND TO

0,1,2,3,4↑

TELETYPE NO. 5 MAY SEND TO

↑

TELETYPE NO. 6 MAY SEND TO

↑

TELETYPE NO. 9 MAY SEND TO

56789↑

TYPE STNS TO WHICH ALL MESSAGES GO

0,5↑

TYPE 0 TO START OVER-TYPE 1 TO CONTINUE

1

GIVE HOUR AT WHICH CLOCK SHOULD START

13

GIVE MINUTES

04

GIVE SECONDS

23

CLOCK WILL START AT    130423
TYPE 0 TO START OVER-TYPE 1 TO START CLOCK

1

CLOCK HAS STARTED
```

Figure 3 Teletype Message Destination

```
*** TO WHOM? ***
7↑
*** GO AHEAD ***

THIS TIME IT SHOULD GET THROUGH!!!!!!!!!    ↑


FROM 8 TO 0 5 7 8    TIME 133618

THIS TIME IT SHOULD GET THROUGH!!!!!!!!!

YOUR LAST MSG RCVD BY 0 5 7 8



FROM 7 to 0 5 7 8   TIME 133915

INDEED, STN # 8, YOU GOT IT.
THIS IS TO SHOW YOU THAT I DO NOT EVEN NEED ONE MISTAKE TO GET
THROUGH TO YOU!!!!!!
```

Figure 4 Teletype Conversation with Message Switching Program

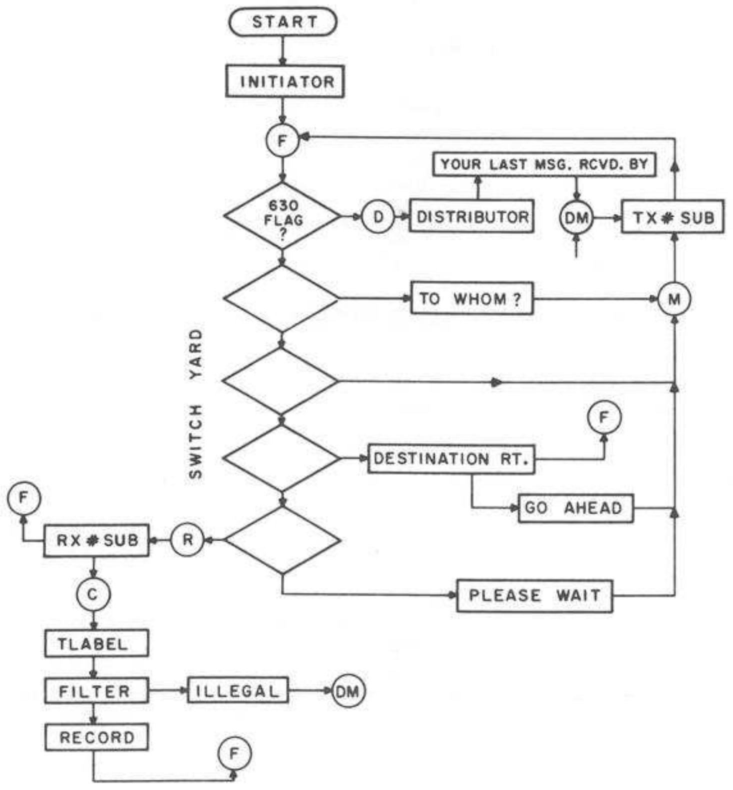


Figure 5 System Flow Chart

returns to the violating station a message telling the subject why his message did not come through. Note that one station may be allowed to transmit to several other stations but, if so specified by the experimenter, only to one other station per communication.

2. When the experimenter has so specified in the beginning, the message from the station of origin is returned by the computer to the station of origin.

3. When the subject at the station has requested a group of destinations among which several are illegal, FILTER subtracts those destinations from the requested group of destinations. Thus the REQUESTED DESTINATION WORD (RDW) is modified into the LEGAL DESTINATION WORD (LDW). In this case the message will come through when there are still some legal destinations left and the routine ILLEGAL will not be used.

4. Certain destinations specified by the experimenter beforehand can be added to LDW. This facility is being used by experimenters who want to have a Teletype station available on which all communications can be monitored instantly.

A real-time clock arrangement labels all messages from the stations with a time tag containing hours, minutes, and seconds. This is useful for later analysis, and helpful during the communications in referring to previous messages. The 1-mc computer clock is used as a time reference base. Six micro decades are used in a 6-digit scaler arrangement. This scaler produces a carry every second. This carry pulse enables two flags; each of the flags connects to the program interrupt bus and the I/O skip bus. If one or more flags are set, a program interrupt occurs, and the information that one or more seconds have passed is processed. There may be reasons for which the interrupt is temporarily turned off. In that case, the hardware is able to remember how many seconds the software has missed. To accomplish this, the two flags are arranged in a scaler configuration. If the program interrupt is turned off, and the flags are therefore not being sensed or reset, the flag scaler is able to count up to 3 seconds. When the interrupt is turned on again, the program interrupt occurs and the program that services flag-interrupts determines how many seconds the software clock is behind and updates the software clock accordingly. As shown in Figure 6 the clock can be started and stopped under program control. As an additional feature it is possible to read a small part of the 6-decade scaler into the accumulator.

The flow diagram of the clock explains how the real-time clock is simulated. The essential part of the routine is a scale of 60 which transfers its carry to another scale of 60 which transfers its carry to a scale of 24.

One of the interesting technical questions that one can ask about the system is if the system is capable of handling all the traffic at its highest possible intensity. Since the equipment arrived in August, 1965, and the software was completed two months later, there has been little opportunity to do measurements. The first indications are that during a normal experiment the PDP-5 idles about 95% of the time. Nevertheless, certain queuing effects generate a potential danger of garbling messages under certain circumstances. In the existing system, every character that comes in is immediately and fully processed. The length of the process varies quite a bit according to the status of the station. Certain probability of garbling is due to the constraint that the character of a particular station has to be processed within a limited time after its arrival. When a Teletype transmits characters at full speed, the rate will be one character each 100 msec. Somewhere near the end of the character cycle, the Teletype flag is raised, and if the flag is not acknowledged within a period of about 20 msec, part of the next character may be superimposed on the previous character in the Teletype buffer, with disasterous results.

One obvious way to resolve this type problem is the use of a buffer area in core (see Figure 8). As soon as a character arrives, it is stored in a buffer consisting of 24 bits, of which 12 bits are used for the character and 12 bits are used for the station number. Each buffer element could consist of one word since the station number will never use more than four bits, and a character will never use more than eight bits, but for simplicity we choose one pair of 12 bit words as the smallest buffer-element. The total buffer has a length of 20 word pairs. As soon as a character arrives, it is stored in a buffer element under control of an input pointer. SWITCHYARD retrieves characters plus station numbers from this buffer under control of an output pointer. SWITCHYARD now checks if the output pointer equals the input pointer and if not a new character can be processed. Once the pointers reach the bottom of the buffer, they are transferred to the top again, so that in effect the buffer is a cyclic software device.

Now other interesting measurements can be made. For instance, at predetermined time intervals a record can be made of the position of input and output pointers. The maximum distance between the input and output pointer dictates the length the buffer should have. This configuration also affords a reliable method to measure average maximum and minimum times to process each character after its arrival.

The following is a short description of another type game for which the system program was completed. Apart from functions the computer performs as described above, several special functions are added. The experimenter may have several versions of his game prepared and stored on the library tape. The versions differ in the established communication patterns. In this game there are nine subjects,

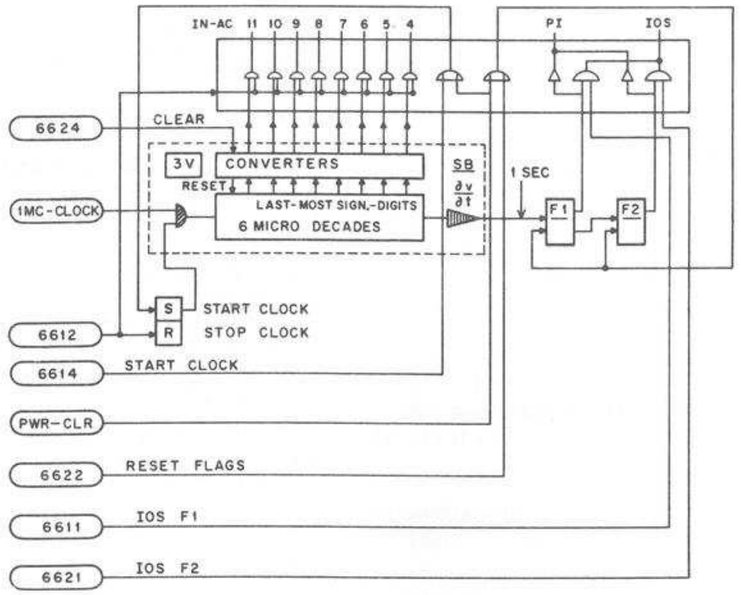


Figure 6 Hardware Clock

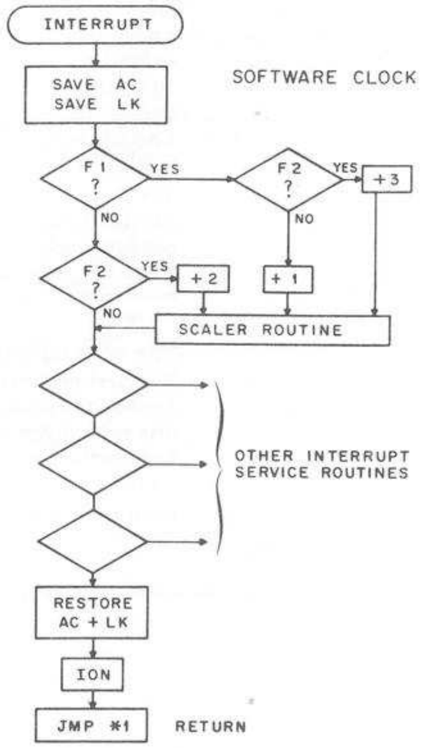


Figure 7 Software Clock

and the experimenter uses one of the Teletypes, station No. 0.

At the beginning of every trial all subjects are given a piece of information, for example a character, or a word. During the trial period all subjects are to put all pieces together in cooperation with each other. Each subject is to give a solution for the problem. For example, the experimenter may give each of the subjects one word and ask them to form a sentence of the nine words. As soon as one of the subjects thinks he has the answer he transmits his answer to the experimenter. Every transmission to the experimenter is being remembered by the computer program. As soon as all nine subjects make a communication to station 0 (whether they had the right answer or not), the trial period automatically ends and the computer sends a message to that extent to all subjects. Both the experimenter and the computer program have control over the completion or starting of all phases of the experiment, the pretrail period, the intertrial period, etc. All pertinent information is recorded on tape.

Several developments are undertaken for experiments to be performed in the future. A special effort is aimed at the implementation of formal language. A formal language may be built of a set of predetermined sentence elements. All the sentence elements are grouped in columns and each column may have from 10 to 50 sentence elements. Elements of each column can be used according to the rules set forth by a sentence pattern. A simple example is the sentence consisting of elements from the columns 1-6-3-4-5-12-14-11. (See Figure 9 for formal language.)

The computer can now do several things. It may store in its memory all available sentence elements. The subject, who has a list of all possible sentences in front of him, can select certain code numbers which result in an assembly of a sentence of his message. The computer can check if the subject kept the rules of the sentence pattern or syntax.

The latest development requires only a reasonable knowledge of sentence elements to be used. The subject need only type in the beginning of each sentence, and as soon as the computer recognizes the sentence element it finishes it automatically. When the computer, on the other hand, finds that the sentence element doesn't exist, it may either add the new sentence element in the proper column in its memory or it may reject the information from the subject and let the subject know about the rejection. When the subject violates the rules of the sentence pattern, the computer may not only reject the next sentence element, but may make suggestions about what to do. Another development concerns gathering sentence elements by the system. In other words, the system starts out with a completely blank memory about what sentences to use. Research assistants then play the game, conscientiously adding "experience" to the system. Experience, of course, can be trimmed after a review of the experimenter.

Another development consists of the simulation of people. No more than five subjects will play a game (one subiect with four simulated subjects, present as "robots" in the system). Here we hope to use a similar approach of teaching the system how to respond to the message as before, in formal language. It is obvious that many very difficult problems lie ahead; one can teach a system how to respond to certain assemblies of message elements, but it would be desirable to let the system review previous parts of the conversation. The question is then how far back, let alone how. One can imagine a situation in which the subject communicates to the computer program and, in case the computer program does not know the answer, it asks for help from the experimenter who uses a separate Teletype. The computer could store each response made by the experimenter and relate it to the subject message.

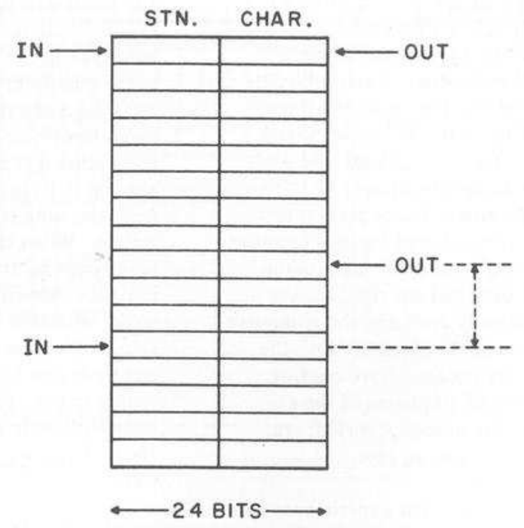
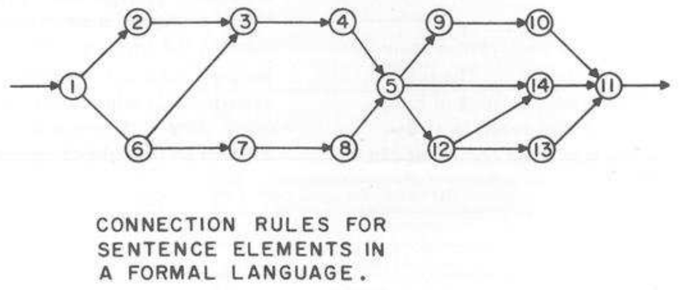


Figure 8 Input Buffer



1
2
3
4
5
6
7
8
9
10
11
12
13
14
CONNECTION RULES FOR SENTENCE ELEMENTS IN A FORMAL LANGUAGE.

Figure 9 Pattern Diagram