SESSION IV SYMPOSIUM: TUTORIAL ON THE SKED COMPUTER SYSTEM

ARTHUR G. SNAPPER, *Western Michigan University, Presider*

An introduction to state notation and SKED

ROBERT DINGLER

*Department of Psychiatry, The University of Chicago, Pritzker School of Medicine
950 East 59th Street, Chicago, Illinois 60637*

RONALD M. KADDEN

Psychology Research Department, FDR Veterans Administration Hospital, Montrose, New York 10548

and

ARTHUR G. SNAPPER

Western Michigan University, Kalamazoo, Michigan 49001

SKED is a minicomputer operating system based on the functional use of state notation as a programming language. An overall view of the operating system, state notation, and the hardware necessary to simultaneously control as many as 12 experimental stations is provided.

There is a rapidly growing interest in the field of experimental psychology toward the use of minicomputers for on-line control of experimental procedures. However, two factors, time and expense, tend to somewhat inhibit an even more widespread use of the minicomputer in the psychological laboratory. Given, for a moment, that funds are available, one of the most frequent, and clearly one of the more important, questions asked by a person considering the purchase of a computer is, "If I buy it, how am I going to make it work?"

One possibility, admittedly very time consuming in itself, is to learn assembly language and develop a program that will conduct whatever experiment there is to be done. A somewhat similar solution would be to hire a computer programmer to write the controlling program. There are a number of problems, however, with both of these solutions. First of all, writing specialized programs is a very time-consuming process in terms of both man and machine hours, even for the experienced programmer. In addition to this initial developmental cost, any changes in the experimental parameters or any development of new procedures will almost certainly result in more time being spent on revisions to the oper-

ating system itself. A second, more subtle but probably more important, problem can occur when more than one person is involved in the design and/or implementation of a procedure. It is almost always the case that there is some lack of precise communication between co-designers of an experiment, or between the one who designs and the one who programs the experiment on the computer. It is very disheartening, to say the least, to conduct research over an extended period of time, only to find out in the end that what you thought was "supposed" to be occurring, really was not.

Another possibility, and probably the one most often followed by the person considering the purchase of a computer, is to seek out information about operating systems designed especially for conducting psychological research. It is the purpose of this paper to present a general introduction to one such system, the SKED operating system. SKED has been designed to operate on any of Digital Equipment Corporations' (DEC) PDP-8 series of computers. For those interested, a detailed comparison of the currently available operating systems has been made (Wood, Sette, & Weiss, 1975).

Prior to the actual implementation of any experimental procedure and irrespective of the equipment used to do so, good research methodology dictates that the experimental question be precisely defined. Once the question has been defined, there must be some method

This work was supported in part by NIDA Contract ADM-45-74-165, C. R. Schuster, Principal Investigator, and in part by Research Scientist Development Award K2-MH-70483 from the National Institute of Mental Health to Arthur G. Snapper.

Figure 1. Example of a state diagram showing a CRF schedule of reinforcement.

by which the user can communicate with the operating system. The basis for communication with SKED is through state notation. Although state notation was originally developed as a mathematical model for describing the sequential nature of events in electronic switching circuitry (Moore, 1956), it has been shown to precisely describe the sequence of events that can occur during behavioral procedures (Snapper, Knapp, & Kushner, 1970).

After precisely defining the experimental question, it is possible, using state notation, to develop the exact procedure that will be implemented in a free-hand drawing, called a state diagram. The state diagram describes all possible events that can occur during an experiment and the precise sequence the procedure will follow given the occurrence of any of these events. Briefly, a state diagram is composed of one or more units called state sets, and each state set is in turn composed of one or more elements called states. The particular sequence a given procedure is to follow can be specified in terms of transitions from one state to another. There are two basic rules governing state notation that allow behavioral procedures to be completely and unambiguously defined: (1) Although there may be any number of state sets, within each state set one, and only one, state may be active at any given time; and (2) transitions between states, in any given state set, are instantaneous. Give these basic rules, all that remains necessary to describe a procedure is to define operators (events) that are capable of producing a transition from one state to another, and a method for describing what changes, if any, that are to be made to the existing set of stimulus conditions.

State notation defines three operators that are capable of producing a transition from one state to another. The first two, time and response(s), are quite familiar to anyone engaged in behavioral research. The third operator is termed a Z pulse and can be best described as an event generated from within the computer, used for communication and/or synchronization between state sets. Each state within the state diagram may have any combination of these three operators associated with it. In addition, there is no restriction on

numeric arguments preceding each operator. For example, in one case a single response may be sufficient to produce a transition; in another case, 10 responses may be required. One point should be made clear: Remembering that only one state in each state set can be active at any given time, the occurrence of operators not specified as affecting any active state is not considered functional. As an example, consider the case where there is only one state set, and the only operator specified as producing a transition from the active state is a response. Clearly, the operator time will occur prior to a response being made. However, in this case, time is not considered functional. Regardless of how much time passes, this operator cannot produce a transition.

In order to describe changes in the existing stimulus conditions, state notation utilizes the onset or offset of elements called outputs. It is important to emphasize that outputs are used to describe changes in stimulus conditions. Therefore, an output will remain in whatever condition it is in, that is, either off or on until it is explicitly specified that condition can be changed.

From this description, it should be obvious that behavioral procedures involving discrete stimuli and responses are readily adaptable to this type of notation. For example, Figure 1 shows a state diagram for a procedure in which responses are to be reinforced on a continuous reinforcement schedule (CRF). In this case, the state diagram contains only one state set (S.S.1) and only two states (S1 and S2). The response operator R1 is defined as a leverpress, and the only output used serves to describe the water dipper as being either in the up (on1) or down (off1) position. The numeric value (one) used with both the response operator and output is arbitrary and would be used to differentiate either different types of responses or different outputs. Colons serve to delineate operators from outputs. The lines with the arrowheads describe the sequence the procedure will follow, given the occurrence of the specified operators.

In the language of state notation, the state diagram shown in Figure 1 would be described as follows: At the start of the procedure, S1 is the active state. The stimulus conditions present at that time are whatever environmental constants are normally present in the experimental chamber. Each leverpress that occurs while S1 is active produces a transition to S2. The only change in stimulus conditions specified by a transition from S1 to S2 is that the water dipper is now in the raised position. The only operator that produces a transition from S2 is time. Therefore, 3 sec after S2 becomes active, a transition will occur to S1. As a result of this transition, the output 1 will be turned off; that is, the water dipper will be in the down position, and S1 again becomes the active state.

There is one additional note that should be made when referring to Figure 1 and the state diagram. Although both operators and outputs are notated above the transition lines, it is not meant to imply that, for ex-

ample, the transition from S2 to S1 requires 3 sec. Remember, transitions between states are instantaneous. The transition lines are simply the most convenient place to notate, in an easily readable format, that, in this case, 3 sec after S2 has become active, a transition will occur effecting the specified change in stimulus conditions.

The time required to produce the final version of the state diagram will vary, depending upon such factors as the complexity of the procedure and the specified sequence of data collection. It has been our experience that some of the more simple procedures may require as little as 5 min, whereas the most complex has required approximately 3 h to produce the completed state diagram. Once the state diagram is in its final form, a simple linear transformation is made so that it will be in a format suitable for teleprinter devices. This linear form of the state diagram is called a state table, and the state table corresponding to the state diagram in Figure 1 is shown in Figure 2. Completion of the state table initiates a three-step process of computer interaction that will result in a programmed experiment.

The first step of this process consists of an interaction with one of the editing programs available, such as the Symbolic Editor or Teco. The editing program accepts as input, characters typed on the keyboard of a teleprinter device, and allows editorial changes such as corrections, insertions, or deletions to be made on the entered text. When the entire state table has been entered and all obvious mistakes corrected, the editing program can be directed to store the state table on some sort of medium (paper tape, magnetic tape, disk) in the American Standard Code for Information Interchange (ASCII).

The second step of the process consists of entering the ASCII state table into another computer program called the compiler. The function of the compiler is twofold: (1) It examines the state table for certain types of errors, such as typographical errors, illegal or undefined use of variables, certain errors of omission, and it insures that the state table is in the proper format; and (2) if no errors are detected, the compiler translates the state table into machine language instructions and stores the program in a binary format on some sort of medium. If an error is detected by the compiler, an error message is printed on the teleprinter along with the line number in which the error was detected and the last good character, and the compilation process is halted. Recovery from an error requires a return to the first step and further modification of the state table. The SKED compiler requires two passes through the state table. At the end of the first pass, it asks for the number of counters to be used in this procedure. During the second pass, the final process of compilation is completed and the binary copy of the state table is produced.

The final step in this process involves debugging and checking of the procedure on the computer. There are several methods available for completing this final step. The SKED executive program, or run time system

```

S S.1,
S1,
R1:ON1 -----> S2
S2,
3":OFF1 -----> S1

```

Figure 2. Example of the linear transformation necessary to convert the state diagram shown in Figure 1 to a form suitable for entry into a computer.

(RTS), may be loaded into the computer, and one may request it to load the binary state table. Operanda in the experimental chamber may then be tested to insure that the proper sequence of events occurs. Alternatively, if the laboratory is equipped with an interface upon which responses from the experimental chamber can be simulated, testing may be done directly from the interface. In addition, there is a special version of the RTS that permits simulation of response on the teleprinter and types on the teleprinter the occurrence of changes in stimulus conditions. Once the final testing is complete, the procedure may be implemented as an active experiment, available for use at any time.

The SKED RTS will provide monitor services for as many as 12 experimental stations. The format of the RTS remains the same regardless of whether 12 different experiments are being conducted or 1 experiment is being conducted in 12 separate stations. The frequency with which these monitor services are provided is controlled by a free-running clock that provides a signal each time one clock cycle has been completed. During each clock cycle, the RTS will read hardware response buffers and scan all active stations, determining if a sufficient number of operators have occurred to produce a transition from any active state. If it is determined that a transition should occur, the RTS will alter certain locations within the state table, effectively making the proper state active. In addition, the RTS will send appropriate signals to hardware devices controlling the turning on or off outputs, as specified by the state table.

In addition to controlling the sequence of transitions specified by the state tables, the RTS provides a number of other services that are available to all active stations. Recording counters may be incremented or a variable that may be used to describe a string of recording counters may be incremented, decremented, or set equal to a given value. Also, Z pulses are generated as specified by the state tables. The RTS will respond to certain characters typed on the keyboard that, for example, will print on the teleprinter the current value of all the recording counters in a specified station, or that will set all of the recording counters of a specified station equal to zero.

Although the RTS provides sufficient monitoring services necessary to control the majority of behavioral procedures, it is quite conceivable that some users would

like to implement special functions that are not of sufficient generality to have been incorporated within the original design. With this in mind, the RTS was designed in such a manner that the experienced programmer can write machine language subroutines and incorporate them within the RTS. These subroutines can then be made available to any station by means of a special-purpose function called an F3. For example, F3s have been written that perform extended arithmetic operations, allow events within the state table to control the high-speed dumping of data, and will plot series of points on a cathode ray tube, to name just a few.

Another feature of the RTS, called the dialogue, allows the user to configure the input-output section to his specific requirements. For example, a user with an interface consisting of 36 response input lines and 36 output lines may, in one case, allocate 3 input and 3 output lines to each of 12 stations. In a case where a particular procedure called for a large number of inputs and outputs, this same user could assign all 36 inputs and outputs to one station. If the input and output lines from the interface are first routed through a device such as a Mac Panel, then reconfiguring an entire system may be as simple as changing plugboards.

In short, the RTS has been designed flexibly so that users over a wide range of interests may tailor the system to their specific requirements. However, in the final analysis, the basic usefulness of any operating system is directly related to the data output of that system. The SKED operating system imposes no limit on the number of recording counters a particular station may use, other than there be sufficient memory space available to load the state table. The manner in which data output from SKED is treated varies among users, depending upon the equipment they have available. There are currently three versions of the RTS available: (1) for users with 4K words of memory, (2) for users with 8K or more words of memory, and (3) for users with a minimum of 16K words of memory and a license to operate DEC's RTS8. In the first two cases, data analysis must be done off-line using either a larger time-sharing system, if available, or one of the standard data handling programs available such as FORTRAN, BASIC, or FOCAL. In any case, the data is in a form suitable for input without need for modification. A number of standard data analysis

programs have been developed (Hamilton, 1975; Snapper, Lee, Burczyk, & Simoes-Fontes, 1974) and are available from the SKED users group (SKUG). For users with a license to operate RTS8, experiments can be conducted as high-priority foreground programs, while a program such as BASIC can be run as a low-priority background program.

The cost of implementing the SKED operating system will vary between laboratories, depending in part upon the type of research being conducted. The minimum hardware necessary to support SKED consists of a PDP-8 series computer with 4K words of core memory, an input-output device such as a Teletype, a digital interface, and a free-running clock, capable of generating an interrupt request every 10 msec. Based on current vendor quoted prices, this basic system would cost approximately \$4,000. A more complete description of the cost involved in implementing various SKED-based systems has previously been given (Snapper, 1974).

In closing, I would like to mention one other alternative for reducing computer costs, at least for the individual researcher. For approximately the last year in the Human Research Laboratory at The University of Chicago, three teams of researchers, doing independent research, have concurrently shared computer time as well as expense. In doing so, we have been able to keep costs down and gradually expand the system.

REFERENCES

- HAMILTON, B. E. Data-collection strategies in SKED. *Behavior Research Methods & Instrumentation*, 1975, 7, 243-247.
- MOORE, E. F. Gedanken-Experiments in sequential machines. *Automata studies*. Princeton: Princeton University Press, 1956.
- SNAPPER, A. G. Alternative versions of SKED: Current systems and future plans. *Behavior Research Methods & Instrumentation*, 1974, 6, 171-173.
- SNAPPER, A. G., KNAPP, J. A., & KUSHNER, H. K. Mathematical description of schedules of reinforcement. In W. N. Schoenfeld (Ed.), *The theory of reinforcement schedules*. New York: Appleton-Century-Croft, 1970. Pp. 247-275.
- SNAPPER, A. B., LEE, D., BURCZYK, L., & SIMOES-FONTES, J. C. FOCAL, FORTRAN, and BASIC programs for reformatting and analyzing data collected by the SKED program. *Behavior Research Methods & Instrumentation*, 1974, 6, 176-180.
- WOOD, R. W., SETTE, W. F., & WEISS, B. Interfacing the experimenter to the computer: Languages for psychologists. *American Psychologist*, 1975, 30, 230-238.