SESSION IV GENERAL TUTORIALS

B. JAMES STARR, *Howard University, President*

The origins and development of state notation and SKED

KENNETH R. STEPHENS and FRANS VAN HAAREN
Western Michigan University, Kalamazoo, Michigan 49008

Research into the behavioral effects of schedules of reinforcement created a need for more accurate descriptions of procedures than were usually presented verbally. Early notation systems, developed by Skinner, Findley, and Mechner, addressed this problem, but failed to be generally applicable. State notation is a graphic notational system based on Mechner's work and the mathematical theory of finite automata. Snapper and others have developed the SKED software system for the PDP-8 family of computers, based on state notation. Originally designed for papertape systems, versions of SKED have been developed to take advantage of support software from DEC, for users who have a mass storage peripheral device. OS/8 SKED makes use of the OS/8 operating system and offers many advantages. Time-Share SKED combines the power of OS/8 and the RTS-8 Real-Time System to provide time-sharing. With this system, programming and data analysis can be conducted while the computer monitors experiments.

Measurement of the basic data is crucial for the development of any science, including psychology. Equally important has been the accurate description of the basic processes of the science. In the experimental analysis of behavior, sequential processes have been of primary importance, especially since Ferster and Skinner (1957) first investigated "schedules of intermittent reinforcement." It became apparent at that time that a multitude of schedules and other related procedures could be generated, with subtle differences that might have an immediate and lasting effect upon the patterns of behavior they produced. The exact description of those procedures, that is, an unambiguous description which would allow others to replicate the procedure exactly, was not available. Skinner (1958) presented the first attempt at notating basic schedules, using the ordinate and abscissa of a graph to represent response requirements and the passage of time, respectively. Although this seemed to be an improvement over the verbal description, Skinner's notation proved inadequate to describe all procedures without ambiguity. In fact, response-contingent schedules (e.g., variable interval) would look the same as noncontingent schedules (e.g., variable time). In 1959, Mechner suggested another notational system that adequately described many schedules

but became unwieldy when used with more complicated schedules. The more complex schedules with multiple operants inspired Findley (1962) to develop a graphic method of describing concurrent and multiple schedules, but his notation could not handle simpler procedures. The lack of general utility seems to be a valid criticism of all the early notational attempts.

At the same time, engineers were developing a mathematical theory of finite automata that established a basis for describing sequential events of the sort encountered in telephone switching circuits and electronics. This development turned out to be crucial for the evolution of an adequate notation system, which would allow the experimenter to describe complex as well as simple procedures in an exact manner.

STATE NOTATION AND SKED

At the same time that notational systems were developing to meet a need in psychology, inexpensive mini-computers, such as the PDP-8, were becoming available. Not surprisingly, a system was developed which combined both trends. Based on Mechner's (1959) notation and conceived within the framework of the theory of finite automata, state notation was developed to describe schedules of reinforcement; but, at the same time, it could function as the basis for a computer language which would control experiments and collect the data produced in those experiments (Snapper, Knapp, Kushner, & Kadden, Note 1). The language

This work was supported in part by Research Scientist Development Award K2-MH-70483 from the National Institute of Mental Health to A. G. Snapper.

presently known as SKED was first described by Snapper and Kadden (1973).

A description of the basic principles upon which the notation system and the computer language are based follows. A very extensive analysis of the same matter can be found in "The SKED Software System, Manual 2: OS/8 and Time-Share SKED" (Snapper, Stephens, Cobe, & Van Haaren, Note 2).

The most important feature of the notation system is that all of the elements of an experimental procedure are described in terms of discrete states, corresponding to stimuli that are active at any given time. The stimuli and the requirements for stimulus changes are defined within a system of states, state sets, and state transition requirements. A state is considered to be part of an experimental procedure, containing all information pertaining to the stimuli that are present and the requirements that have to be met in order for transition to another state to occur. Each state is part of a state set and, within each state set, one and only one state can be active at a given point in time. However, several state sets may operate in parallel, each independent of the others, except when they are synchronized in the manner described later in this paper. Transitions between states are instantaneous and stimulus changes occur as soon as the transition requirements are met. Multiple transition requirements for transition to different states may be present. Whichever transition requirement is met first causes its particular transition to occur.

There exist three events that can cause transition from one state to another (input events): responses produced by the subject in the experimental situation, the passage of a certain amount of time, and the "Z pulse," which is a logical condition produced within the conventions of the SKED system. The input expression, in SKED, is written on the transition vector leading to a state. Since no ambiguity can be tolerated within a state set, multiple transitions from a state have to meet certain logical rules. First, there must be as many transition lines as there are possible transitions. Single lines cannot go to different states, since no two states can be active at the same time. Second, there can only be one time transition which leaves a state. Third, multiple transitions of the same response type are not permitted, unless only one will actually leave the state. Last, response requirements may be "gated" (the logical AND function) with probabilities or with values of a variable, but response requirements may not be gated with other responses or time values.

In state notation, states are indicated by enumerated circles, and stimuli that are present when the state is active may be drawn within the circle. Transition requirements are notated on straight arrows which interconnect the different states. Four different sorts of transitions are possible. First, transition can occur to another state, which means that as soon as the transition requirements are met, control over the

procedure is given to the state indicated at the end of the transition line that leaves the present state. All transition requirements in the previous state are reset. Transition can also occur to the same state, resulting in the execution of any specified outputs (which will be discussed below) and reinitializing of the transition requirements included in the state. The pseudotransition, or the transition to a null state, operates in the same way as the transition to the same state in executing outputs, but differs in the sense that the transition requirement event counters are *not* reset. The transition to STOP turns off all stimuli active at the time and stops the execution of the state table. The transition to STOP usually occurs in the last state of the last state set.

When transition occurs, it may cause the computer to generate any of several different output functions. For instance, the stimulus complex in the experimental chamber can be changed by turning stimulus lines on and off (e.g., a grain dipper comes up as reinforcement for a pigeon and goes back out of reach a few seconds later). Such stimulus change occurs instantaneously upon state transition. The output expression in SKED follows the input expression, separated by a colon, as in R1:ON3.

A second type of output function is the counting of events. A block of memory locations may be reserved as counters. Then, each of these may be selectively incremented to record the frequency of an event or response type, or to record a distribution of latencies or interresponse times (Hamilton, 1975). There is no limit (other than available memory) to the number or type of events that can be counted within SKED, which means that collection of large quantities of data for the microanalysis of behavior is greatly facilitated. The counter to be incremented is specified either by number or by using variable notation (e.g., :C31 or :CN).

A third type of output function serves to communicate between independent state sets and to synchronize them when necessary. A very short logical condition, the Z pulse, is generated as an output by one state and is used as an input in a state of another state set on the computer's next cycle. Up to 12 Zs are available for the SKED programmer. Another method of synchronizing parallel state sets utilizes logical AND gating with the currently active state of a state set.

A fourth type of output function is the call to special purpose subroutines (Snapper & Hamilton, 1974). This function can be used to manipulate certain nonstandard devices, or to perform arithmetic operations on the value of a variable, or to cause data to automatically dump to an external storage medium. The notation has recently been changed to permit easier manipulation of variables, as well as evaluation of multiplication and division operations. Arrays of variables are also now permitted.

Since its origination in 1967, state notation and SKED have found their way to numerous experimental

laboratories in and outside the United States. SKED's reliability, explicitness, and simple implementation have been useful in a number of different settings.

DIFFERENT SKED SYSTEMS: PAPERTAPE, OS/8, AND TIME-SHARE SKED

All SKED systems are designed for the PDP-8 family of minicomputers.¹ In addition, the SKED system requires an interface, which forms the intermediary between the computer and the experimental situation. State Systems, Inc., of Kalamazoo, produces interfaces especially developed for the SKED system, but some users have built their own interfaces. The primary function of the interface is to convert the 5-V computer output into a 24-V current which drives the lights and electromechanical devices in the experimental situation, and to transform the 24-V inputs from the experimental situation back to the 5-V level.

At the time of this writing, three different levels of the SKED system are available for use in the laboratory. Each requires different software, hardware, and peripheral devices, and each provides the user with different levels of sophistication.

The SKED papertape system is the simplest and most inexpensive system. It requires a minimum of 4K of core and a low- or high-speed papertape reader and punch. The software components include an editor program (SYMBOLIC EDITOR) for making state tables from the state notation diagrams; the SKED compiler accepts the ASCII code produced by the Editor and produces relocatable binary output. The SKED Runtime System is the executive component. The Runtime System accepts the binary output from the Compiler and controls the experiment. The papertape system offers the user the possibility to run even very complicated experiments. Certain limitations in the system can, in the long run, turn out to be a burden to the user. The fact that all input to the system and all output from the system has to occur through papertape, and that Editor, Compiler, and Runtime System have to make use of the same available core space, makes the system bothersome in terms of data storage and flexibility. The development of new state tables, data analysis programs, and the data analysis itself, all have to occur at different times. The set-up costs of the papertape system (approximately \$3,000) make it a useful system to start with, however. The SKED papertape system has been thoroughly documented by Snapper, Stephens, and Lee (Note 3).

The second-level system is OS/8 SKED. The experimenter who, in addition to a PDP-8 with at least 12K of core, has access to a mass storage device (such as disk, floppy disk, or magnetic tape) and a license to use OS/8 from the Digital Equipment Corporation, can run OS/8 SKED. OS/8 is the name of a computer operating system which consists of a library of programs used in transferring data, analyzing data, file manipulation, and directory functions. The user of OSCOMP (the OS/8 SKED Compiler) and OSRTS8 (the OS/8 SKED

Runtime System) will find his program input-output time cut drastically from papertape and will have all the advantages of a file-oriented system. OS/8 SKED can also run under BATCH control, which minimizes the need for user input, if the same sequence for computer usage occurs each day. The BATCH control system can also be very efficient in gradually familiarizing the novice user with the characteristics of the system, and the logical progression of data collection and various processing operations.

Time-Share SKED is the third and most sophisticated level of the SKED system available to the user. Time-Share SKED is built around the DEC RTS-8 Real-Time System software. (Permission to use the software system must be obtained from the Digital Equipment Corporation.) Time-Share SKED offers all the advantages of OS/8 SKED plus the ability to program or to do data analysis while several experiments are being run. This system offers the user the highest form of flexibility and insures the most efficient use of the equipment available. Time-Share SKED is still evolving, thanks to the combined efforts of several members of the SKED Users Group.

REFERENCE NOTES

1. Snapper, A. G., Knapp, J. Z., Kushner, H. K., & Kadden, R. M. *A notation system and computer program for behavioral experiments*. Paper presented at the meeting of the Digital Equipment Computer Users Society (DECUS), New York, June 1967.
2. Snapper, A. G., Stephens, K. R., Cobez, R. I., & Van Haaren, F. *The SKED Software System Manual 2: OS/8 and Time-Share SKED*. Kalamazoo, Mich: The SKED Users Group, 1976.
3. Snapper, A. G., Stephens, K. R., & Lee, D. M. *The SKED Software System Manual 1: Papertape versions*. Kalamazoo, Mich: The SKED Users Group, 1974.

REFERENCES

- FERSTER, C. B., & SKINNER, B. F. *Schedules of reinforcement*. New York: Appleton-Century-Crofts, 1957.
- FINDLEY, J. D. An experimental outline for building and exploring multioperant behavior repertoires. *Journal of the Experimental Analysis of Behavior*, 1962, 5, 113-166.
- HAMILTON, B. E. Data collection strategies in SKED. *Behavior Research Methods & Instrumentation*, 1975, 7, 243-247.
- MECHNER, F. A. A notation system for the description of behavioral procedures. *Journal of the Experimental Analysis of Behavior*, 1959, 2, 133-150.
- SKINNER, B. F. Diagramming schedules of reinforcement. *Journal of the Experimental Analysis of Behavior*, 1958, 1, 67-68.
- SNAPPER, A. G., & HAMILTON, B. E. Programming special functions in the SKED system. *Behavior Research Methods & Instrumentation*, 1974, 6, 173-176.
- SNAPPER, A. G., & KADDEN, R. M. Time-sharing in a small computer through the use of a behavioral notation system. In B. Weiss (Ed.), *Digital computers in the behavior laboratory*. New York: Appleton-Century-Crofts, 1973.

NOTE

1. Product name of Digital Equipment Corporation.