The three statements beginning at LOOP present the list of stimuli, one each 1.2 sec. The OUTPUT command directs the value determined by evaluating the expression (I TH OF STIM.LIST) to the experimental room via the generalized output interface; in this case, the S is viewing a display device connected to the general output interface. LOOP+1200 MSEC is an example of a temporal label; this feature suspends execution of the program until the required time has elapsed since last passing the reference label—LOOP in this case.

OUTPUT O turns off the display. The conditional statement determines, with equal probability, whether the probe is to be taken from the set or not, and sets a flag, IN.SET, to record that fact. The variable IN.SET.RESP is then set to select randomly a response the S is to use if he believes the probe to be a member of the set.

After a 2-sec wait, the S is informed of the response to be made to indicate set membership. This is displayed for 1 sec; this is an example of a command that is automatically cancelled following the expiration of its duration. When 1.2 sec have elapsed following the onset of the set membership response display, the probe digit is output. At this point, the response equipment is enabled with the PERMIT RESP statement, allowing the S to respond. He is given up to 5 sec to respond, execution of the program being held at that point until

he responds or the time expires. Upon either condition, the display is turned off. The variable CORRECT is set to TRUE or FALSE, depending upon the response and the correct response. Then the word "CORRECT" or "INCORRECT" is displayed as feedback to the S for 2 sec. Information concerning the size of list, the S's response, and its correctness is recorded for later analysis.

Three seconds later, the experiment continues with the next trial, unless 168 trials have occurred. In that case, "END OF EXPERIMENT" appears on the video display for 10 sec and the experiment is concluded.

CURRENT STATUS

PSYCHOL was implemented in 1968 on the CCPL computer. That implementation depended upon particular bulk storage equipment that later became unserviceable. A generalized compiler for PSCYHOL was begun in 1969 which would produce object code for a number of computers. This project was never completed, and the language is currently inactive.

REFERENCES

- McLean, R. S. PSYCHOL: A computer language for experimentation. *Behavior Research Methods & Instrumentation*, 1969, 1, 323-328.
Sternberg, S. High-speed scanning in human memory. *Science*, 1966, 153, 652-654.

SCAT: Design criteria and software*

PETER G. POLSON†

*The Computer Laboratory for Instruction
in Psychological Research
University of Colorado, Boulder, Colorado 80302*

SCAT is a high-level language for writing programs that control experimental procedures and record data. This paper discusses general design criteria for higher level languages and the basic structure of the SCAT language. A second section of the paper briefly describes the compiler and gives a description of the operating system.

Extended State Change Algorithm Translator (EXTENDED SCAT) is a language that is used to describe experimental procedures involving contingencies between discrete sets of stimuli and responses, e.g., operant paradigms, concept identification, two-person games, etc. We have been using SCAT at the Computer Laboratory for Instruction

in Psychological Research for the last 2 years to teach real-time programming concepts to undergraduate and graduate students and to write programs that run experiments. The SCAT language was originally developed by the Grason-Stadler Company as part of a real-time system based on a DEC PDP-8/I. EXTENDED SCAT is a downwardly compatible dialect. Our additions to the SCAT language enable us to exploit the features of a medium-scale machine like the Xerox Sigma 3 or the IBM 1800 and to program a wider range of experimental paradigms.

This paper discusses principles underlying the structure of the SCAT language in the context of general criteria for special-purpose programming languages (Sammet, 1969). The EXTENDED SCAT compiler is briefly described, and the operating system is described in detail. Only a short outline of the SCAT language itself is included in this paper. See Polson (1971) or Polson & Campbell (1972) for a complete description of EXTENDED SCAT.

DESIGN CRITERIA AND THE STRUCTURE OF SCAT

Special-purpose programming languages, e.g., SCAT, ACT, PSYCHOL, etc., have two functions. First, such

*The development of the EXTENDED SCAT system was supported in part by National Science Foundation Grant GJ 453 to D. E. Bailey.

†Requests for reprints and more detailed information about the EXTENDED SCAT system should be sent to Peter G. Polson, CLIPR, 1511 University Avenue, University of Colorado, Boulder, Colorado 80302.

languages are programming aids. Second, they should be problem solving aids. A programming aid assists you in the mechanical task of writing sequences of machine instructions and in all of the necessary bookkeeping functions. A programming aid can also provide a means for generating linkages to library subroutines that perform functions like formatted input/output, calculation of sin, cos, and square root, etc. There are several ways in which special-purpose compilers function as problem solving aids. First, and most important, a higher level language plus its associated subroutine library and operating system permit you to ignore and/or remain ignorant of a large amount of detailed information about the structure of a given system. Second, a specialized programming language can provide a conceptual structure that facilitates the statement of a problem or task in a form suitable for implementation on a computer. Millenson (1971), Sammet (1969), and other authors have pointed out that these functional criteria for a higher order programming language can be satisfied by a language that is based on an adequate model of the class of problems to be expressed in that language.

THE MODEL UNDERLYING THE SCAT LANGUAGE

SCAT is based on very specific assumptions about the structure of the experimental environment and about the representation of the process defining the experimental procedure. SCAT represents the experimental environment as a collection of binary input and output signals. Information about the occurrence of a response is input to the program on a specified R line and is called an R signal. Binary information is output from the program on S lines. Digital information is output on sets of eight lines called V registers. V registers are used to control devices requiring digital inputs, e.g., D-A converters, programmable oscillators, binary-to-decimal decoding matrices, etc. R and S lines and V registers are symbolically identified in a SCAT program. The E must know the correspondences between the laboratory environment and the line and register identifiers in order to interpret inputs and control the operation of external devices connected to the outputs.

SCAT's representation of the experimental procedure is based on a formalism developed by Snapper, Knapp, and Kushner (1970) to describe operant paradigms. The Snapper et al descriptive schema makes use of concepts from the theory of finite automata. They assume that the various elements of an experimental procedure can be represented as a finite state process and a set of inputs to the process. A state specifies a stimulus to be presented to the S and the next state of the process contingent on a specified sequence of inputs. The inputs to the process are events: a response, a specified number of responses, the end points of time intervals, etc. The output of the process is the stimulus information

specified by each state. The experimental procedure is the sequence of states that define the process. Thus, the procedure is determined by the properties of each state in the process and the inputs received by the process.

Three special-purpose languages have been developed that are based on the Snapper et al (1970) model of an experimental paradigm. The first language is SKED, developed by Snapper. The second is a language called Automated Contingency Translator (ACT). The ACT system was developed by J. R. Millenson (1970, 1971). A system based on the ACT language is marketed by Lehigh Valley Corporation. The third is SCAT.

THE EXTENDED SCAT LANGUAGE

EXTENDED SCAT has quite powerful computational abilities. The operations of addition, subtraction, multiplication, division, and exponentiation are defined for both real and integer variables. Variables can either be simple or subscripted. The order in which arithmetic expressions are evaluated is determined by the usual rules for FORTRAN. In addition, a subscript expression can be any legal arithmetic expression. Unlike FORTRAN, the names of all SCAT variables and functions are predefined. There are 20 simple variable names and a single array for each variable type. The number of locations in an array is specified by the programmer.

A SCAT program consists of three sections: a declaration section, a control section, and a report section. Each section is a collection of one or more statements with different types of statements appearing in each section.

The statements in the declaration section provide information to the compiler and operating system about the structure and storage requirements of the program. For example, declaration statements specify the lengths of real and integer arrays, define input and output lists, and specify the range and cell width of histograms.

The statements in the control section of a SCAT program describe the experimental procedure as a sequence of states, with each control statement defining a state. As in the Snapper et al (1970) model, each control statement specifies the stimulus presented to the S, the inputs that will cause the program to change state, and the state which the process enters next. There are three components of a control statement: the statement label, the state description clause, and the state change clause.

The state description clause is a series of replacement phrases, stimulus descriptors, and other commands. The specified sequence of operations defines the current state. Stimulus descriptors control the status of output lines (on, off), which in turn determine the stimulus presented to the S. Replacement phrases and other SCAT commands manipulate stored information. Finally, there are commands that enable E to read and write bulk storage units like disk drives or magnetic tape. The FORTRAN analogue of the state description clause

would be a series of replacement statements and subroutine calls. The subroutines are assembly language routines that carry out the operations specified by the various SCAT commands.

The state change clause is a series of phrases of the form

condition > (state label)

which is interpreted as "When and if the specified condition is satisfied, go to the named state." The FORTRAN analogue of the state change phrase is the logical IF statement. However, the execution-time behavior of these phrases is completely different from the FORTRAN logical IF statement. The state change phrase is the mechanism by which a SCAT program responds to real-time events, e.g., a response on a given R line, the end of a time interval, etc.

When the program enters a control state, the elements of the state definition clause are immediately executed in the order of their occurrence. The conditional parts of the state change phrases are then evaluated. The program will transfer to the control state specified by the first phrase encountered with a true conditional part. However, it is possible that initially all of the conditional terms are false. The program remains in the current control state until at least one of the conditional terms is satisfied.

The statements in the report section format and output information that was recorded during the execution of the control section. The structure of the statements in the report section is very similar to the form of the control statements. The function of the report section is to enable the E to output a limited amount of summary information about the S's performance.

Software: The Compiler and Operating System

The two major components of the EXTENDED SCAT system's software are the compiler and operating system. The compiler is described in detail by Campbell (1972). The basic structure of the operating system will be described in detail.

The Compiler

The EXTENDED SCAT compiler translates a SCAT program into Basic FORTRAN IV. The output of the SCAT compiler is then input to the vendor's FORTRAN IV compiler, which generates the object code. In addition, the EXTENDED SCAT compiler is written in Basic FORTRAN IV, and thus the two-stage compilation process (SCAT to FORTRAN, FORTRAN to object code) is practically machine independent.

The Operating System

There are two major subdivisions of the EXTENDED SCAT operating system: the real-time executive and the command executive. SCAT programs operate in an

on-line real-time multiuser environment. The real-time executive implements this environment. Essentially, the real-time executive is a multiprogramming operating system that supports concurrent execution of a variable number of tasks. An E communicates with the system through the command executive. Various commands enable the E to load SCAT programs in memory, initiate and terminate execution, and monitor a running program.

A fundamental aspect of the operating system is the time-sharing algorithm employed by the real-time executive. The basic rule is run to completion, i.e., a program remains in control of the computer until it can no longer proceed. An example would be when a program presents a stimulus and then must wait for the occurrence of a response. When a program must wait for the occurrence of an external event in order to proceed, it returns control to the real-time executive. Execution of the program is suspended until the occurrence of an external event, e.g., the lapse of a specified period of time, the occurrence of an R signal, etc. Thus, SCAT programs are external-event-driven programs. Furthermore, when a SCAT program is given control of the computer, it is allocated a fixed period of time. The program must return control of the computer to the operating system within this time period or be aborted. While a given SCAT program is being executed, higher priority interrupt-driven routines respond to and record the occurrence of external events from other active stations.

Command Executive

The command executive operates concurrently with the real-time executive. The E communicates with the command executive in real time through a Teletype machine. Since all functions of the real-time executive have higher priorities, the response time of the command executive can be as long as several seconds. The response time will be a function of the number of active SCAT programs in memory.

The two major divisions of the command executive are the query subsystem and the command interpreter. The query subsystem enables an E to examine and modify the values of variables while a program is in execution. The command interpreter calls the various operating system functions specified by E's input from the Teletype to the command executive. All sequences of commands refer to a given program via a station number. A dialog with the command executive begins with the specification of a station number.

Once a station has been selected, the E can carry out functions like initiating and terminating execution of the program or using the query subsystem to examine and modify the values of variables in the program. The commands are: LOAD, START, STOP, RESTART, and KILL. The LOAD command allocates the necessary space in memory and then loads the program into memory. START initiates execution of the program.

STOP suspends execution of the program, and **RESTART** resumes execution of a suspended program. The **KILL** command terminates execution of the program and releases all memory space allocated to it.

Real-Time Executive

The real-time executive has four major subsystems: the R-line monitor routines, the time-management subsystem, the scheduler, and the SCAT subroutine library. The R-line monitor routines and the time-management subsystem record external events for each active program. The scheduler, using information passed to it by the event-recording routines, controls the allocation of computer time to each of the SCAT programs. The SCAT subroutine library is a set of routines called by SCAT programs. A majority of the SCAT commands compile into subroutine calls, and the operations to be carried out by these commands are implemented by routines that are part of the real-time executive.

The R-line monitor routines are a collection of interrupt-driven independent programs, one for each active station, that respond to and record the occurrence of R signals. Whenever an R signal occurs on one of the R lines associated with a station, an interrupt occurs. Information about the identity of the R line on which the signal occurred is transmitted to the SCAT program, and a flag is set in the scheduler indicating that the associated SCAT program should be scheduled for execution. The highest priority interrupts associated with the SCAT system are assigned to the R-line monitor routines.

The time-management subsystem is a set of independent subsystems: the master clock, the abort clock, and the timer sub-subsystem. The master clock is a high-precision free-running clock, which operates at either a 10-msec or a 1-msec resolution. Its primary use is for response latency measurement. The abort clock is used by the scheduler to monitor the execution-time behavior of SCAT programs. When the scheduler passes control of the computer to a SCAT program, the abort clock is set. If the abort clock times out before the program returns control of the computer to the real-time executive, that SCAT program is aborted. The timer sub-subsystem implements the time-dependent condition, (t)SEC.

The timer sub-subsystem contains three routines: an initialization routine, a start routine, and a timer program. The programmer can set up to 16 independent timers. Each call to the initialization routine transmits the duration of the interval to be timed in 1/100ths of a second, a timer identifier, and a station identifier. A call to the start routine activates all initialized timers set up by the calling SCAT program. The timer routine is called by an interrupt that is triggered every 1/100th of a second. Each member of the cue of active timers that have been set up by the initialization and start routines

is tested to see if the specified time period has elapsed. When a timer times out, information is transmitted to the SCAT program that set it up and a flag is set in the scheduler indicating that this SCAT program should be scheduled for execution.

The primary function of the scheduler is to allocate computer time to the several active SCAT programs in memory on the basis of information provided by the R-line monitor and time subsystems. The scheduler also monitors all input/output and the execution of each program. The execution of the various independent components of the operating system and the transmission of information between these programs are coordinated by tables maintained by the scheduler and initialized by the command executive.

The actual allocation of computer time to the various active SCAT programs is done on a round-robin basis. There is a request for computer time flag for each active program which can be set by the R-line monitor and time-management subsystems. The flags of all active programs are tested in a fixed cyclical order. If no other SCAT program has control of the computer and none are waiting for service, a SCAT program requesting the computer will be restarted almost immediately after the occurrence of an external event. The round-robin service routine gives all active SCAT programs equal access to the computer. For example, a program monitoring a set of R lines with high input rates cannot dominate the whole system. Between each period of time allocated to a given user, the scheduler will test the request for service flags of all other stations and allocate machine time to them when called for. The speed with which the SCAT system responds to external events, and thus the accuracy of latency measurements and time periods between events controlled by the system, is a function of the number of programs that receive allocations of machine time before control is passed to a given program. The nominal response time of the system is 10 msec.

REFERENCES

- Campbell, G. D. An investigation of Markovian state change languages. CLIPR Publication Series, Publication No. 7. Computer Laboratory for Instruction in Psychological Research, Quantitative Psychology Program, Department of Psychology, University of Colorado, Boulder, Colorado, June 1972.
- Millenson, J. R. Language and list structure of a compiler for experimental control. *Computer Journal*, 1970, 13, 340-343.
- Millenson, J. R. A programming language for on-line control of psychological experiments. *Behavioral Science*, 1971, 16, 248-256.
- Poison, P. G. Extended State Change Algorithm Translator reference manual. CLIPR Publication Series, Publication No. 2, Computer Laboratory for Instruction in Psychological Research, Quantitative Psychology Program, Department of Psychology, University of Colorado, Boulder, Colorado, January 1971.
- Poison, P. G., & Campbell, G. D. EXTENDED SCAT. *Behavioral Science*, 1972, 17, 558-565.
- Sammet, J. E. *Programming languages: History and fundamentals*. Englewood Cliffs, N.J.: Prentice-Hall, 1969.
- Snapper, A. G., Knapp, J. Z., & Kushner, H. K. Mathematical description of schedules of reinforcement. In W. N. Schoenfeld and J. Farmer (Eds.), *Theories of reinforcement schedules*. New York: Appleton-Century-Crofts, 1970.