Significant improvements in the modeling capability and storage flexibility of the General Purpose Systems Simulator are described in this paper.

Increased versatility and ease of use as well as new debugging aids are also discussed.

The additions and changes to the simulator are illustrated by examples.

GPSS III — an expanded general purpose simulator

by H. Herscovitch and T. H. Schneider

The General Purpose Systems Simulator is a broad-range general-purpose tool for modeling and examining the behavior of systems in management and engineering science areas. The latest version of this program, GPSS III, offers the simulation user several distinct advantages.[1]

Several structural changes to the program, both internal and external, significantly improve model running time and increase core storage availability. The most significant of the external changes is the elimination of delay-time and selection-factor operations from all block types. Selection operations are performed by a TRANSFER block; delay times are accounted for by an ADVANCE block. In addition, new GPSS III features expand modeling capability, versatility, and ease of use. Some of the basic GPSS concepts are now reviewed, although familiarity with GPSS II[2] is assumed throughout the text.

GPSS concepts

The primary advantage of GPSS is its ease of application in its field; no initial programming is required nor is computer programming training or experience necessary. The program features a simple flow chart language for describing the problem or system to be simulated. When this description is transferred to punched cards and presented as input to the computer, the program automatically carries out a simulation of the system.

The elements of a system are logically represented by a set

of abstract elements called entities, and the logical rules governing a system flow are reduced to a set of standard operations. The entities are divided into four classes: dynamic, equipment, statistical, and operational.

The dynamic entities are called transactions, and represent units of traffic, such as requests arriving at central points, jobs awaiting processing, etc. Transactions are created and destroyed as required during the simulation run and can be thought of as moving through the system, causing certain actions to occur. Associated with each transaction are a number of parameters to which the user can assign certain values that describe characteristics of the transaction. For a transaction representing a ship, for example, a parameter may describe the amount of cargo to be unloaded. This number could then be used in the simulator logic to determine the time required for the unloading operation.

Entities of the second class represent elements of system equipment, such as facilities, storages, and logic switches, that are acted upon by transactions. A facility can handle only one transaction at a time and represents a potential blocking condition. A storage can handle several transactions concurrently and can be used to represent parallel processing entities, such as a parking lot or a typing pool. A logic switch is a two-state indicator that can be set by a transaction to modify the flow of other transactions. For example, this switch could represent a traffic light or the "next window" sign of a bank teller.

In order to measure system behavior, two types of statistical entities are defined: queues and tables. Each queue maintains a list of transactions delayed at one or more points in the system, and keeps a record of the average number of transactions delayed and the length of these delays. A table may be used to collect statistical information as desired. These two entities provide a major portion of GPSS output.

The operational entities, called blocks, constitute the fourth class. Like the blocks of a flow chart diagram, they provide the logic of a system, instructing the transactions where to go and what to do next. There are 36 specific block types in GPSS III to represent basic system actions. These blocks, in conjunction with the other three classes of entities described above, constitute the language of GPSS.

To provide input for the simulation, control and definition cards are prepared from a flow chart of the system. These cards constitute the model in GPSS language. Once the system model is loaded, the GPSS program generates and moves transactions from block to block according to timing information and logical rules incorporated in the individual blocks. Each movement is designated to occur at some particular point in time. The program automatically maintains a record of these times (Future Events chain), and executes the movements in their correct time sequence. Where actions cannot be performed at the originally scheduled time—for example, when a required facility is already in use—proc-

essing temporarily ceases for that transaction. The program automatically maintains the status of the equipment causing the delay, and activates the transaction again as soon as the condition changes.

The program output provides information on:

- The amount of transaction traffic flowing through the complete system and/or any of its parts
- The average time required for transactions to pass through the complete system or between selected points, and the probability distribution of this passage time
- The degree to which each item of equipment in the system is loaded, together with the distribution of storage occupancy
- The maximum and average queue lengths occurring at various points, as well as the distribution of queue lengths

extended storage flexibility

For purposes of description, the salient features of GPSS III have been divided into several categories, the first of which pertains to the storage flexibility and is now discussed. In GPSS III, core storage restrictions have been largely alleviated by automatic reallocation of core storage and by the ability to choose the number of parameters for each transaction. These two features allow easy handling of very large models, or of models that require the use of a relatively large number of a particular entity.

Automatic reallocation of core storage gives the user the ability to reapportion core storage without going through a reassembly process and the clerical chores associated with reassembly. The analyst specifies the exact number of blocks, facilities, storages, queues, logic switches, tables, functions, variables, savexes (savevalues), transactions, and common storage he desires. The reallocation is done by word modification and is extremely fast in comparison to reassembly. Specification is simple, and it is not necessary to specify values for those entities whose quantity is to remain normal.

The ability to specify the number of parameters for each transaction saves core under certain conditions. This specification is done directly at the GENERATE and SPLIT blocks, not from a master control card. A system can therefore have transactions with different numbers of parameters. The analyst may specify that transactions be given from 0 to 100 parameters; without specification, 12 parameters per transaction are automatically assigned. This feature saves core when less than 12 parameters are needed for a particular transaction. Techniques that associate other entities (such as savexes) with the transactions are eliminated when more than the normal number of parameters are required.

improved modeling capability

The modeling capability of GPSS has been improved by the addition of user chains and Attribute Valued functions. Both of these improvements are now discussed in detail.

In GPSS, a transaction at any particular time is a member of the Current Events, Future Events, or Interrupt chains. These chains, which are strings of transactions, behave in a

predetermined way that is not directly under the user's control. GPSS III allows the user to bypass this standard operation and to construct his own string of transactions, called user chains. These user chains make it possible for the analyst to create and manipulate his own chains of transactions independent of the GPSS chains. Also, to reduce computer running time, transactions known to be inactive can be removed to a user chain, thus reducing the number of transactions to be scanned on the internal Current Events chain. Since the user has complete control over these chains, it is also possible to represent may different queuing disciplines.

Two new block types, LINK and UNLINK, enable the analyst to remove transactions from the Current Events chain, place them on a user chain in any manner desired, remove them later in any order, and place them back on the Current Events chain. Since the analyst can completely control the operation of his own chains and bypass the predetermined operation of the Current Events chain, this new feature significantly extends the applicability of the simulator.

The LINK block makes it possible to place a transaction at the beginning or end of a specified user chain, or to merge the transaction onto the chain according to specific parameter values. The UNLINK block allows removal of one or more transactions from the beginning or end of the chain. This block also allows specific transactions, based upon parameter content, to be removed from any point on the chain. Thus, any queuing discipline may be accounted for.

For example, the block LINK 5,LIFO removes the current transaction from the Current Events chain and places it at the beginning of user chain 5. Removing transactions from this chain in the normal manner (i.e., from the beginning) results in a LIFO (last in, first out) discipline. As another example, UNLINK 3,AAA,ALL removes ALL transactions from user chain 3 and places them on the Current Events chain, with their next block specified as AAA. The transaction that entered the UNLINK block proceeds to the next sequential block.

Figure 1 illustrates how LINK/UNLINK can be used to sort transactions according to a specific queuing scheme. Transactions are sorted in accordance with parameter 6, but the transactions within each class are grouped on a LIFO basis. In other words, if there are eight transactions with parameter 6 equal to 1, they are filed in the reverse order from their arrival order.

The example in Figure 2 represents a rental agency with three types of vehicles available: cars, pickup trucks, and closed trucks. A customer entering the agency describes the year and type of vehicle he wishes to rent. The rental agent then determines whether the described vehicle is available. The problem can be readily solved by setting up a user chain for each vehicle class. Figure 2 illustrates the construction of user chains to bypass the predetermined operation of the Current Events chain. This relatively

simple example could be expanded to a complex inventory problem where, in each type of chain, the choice of a transaction is based on many criteria. The difficulties encountered in attempting to represent such problems by standard chain operation would be extensive or even insurmountable.

Further modeling capability is provided in GPSS III by the new Attribute Valued function. This function allows the use of any Standard Numerical Attribute in the specification of a function point where previously only constants could appear. Thus, func-

Figure 1 User chain transaction sorting

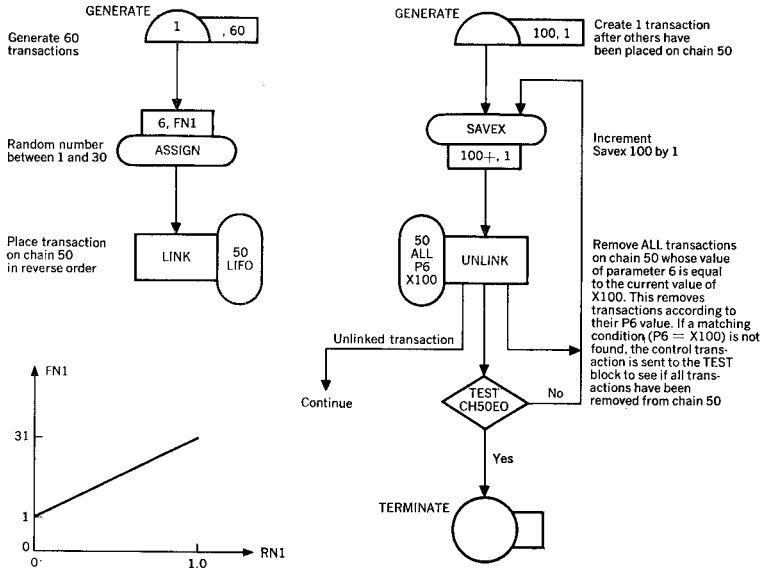
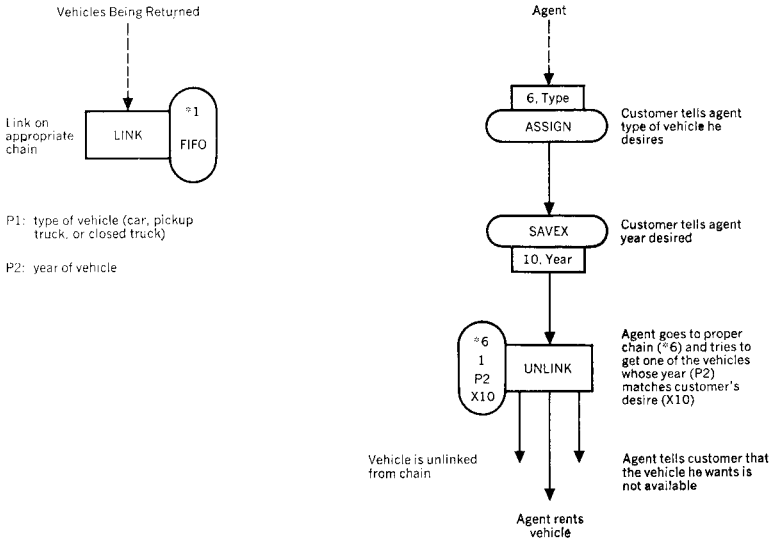


Figure 2 User chain rental example



tions may now vary dynamically with the system; they also provide a means for indirectly referencing Standard Numerical Attributes.

As an example of the many possible uses of Attribute Valued functions, we use a system with message types and lengths as listed in Table 1. The messages are serviced by the routines of Table 2. Processing is accomplished by a different series of blocks for different message types. The symbolic name of the first block of each string serves as the identifying characteristic. Assuming that the message type is given in parameter 4, the block sequence of Figure 3 assigns the message length and directs the message to its service routine. Note that FN4 provides multilevel indirect referencing, because functions 1, 2, and 3 can, in turn, refer to other Standard Numerical Attributes. In FN5, the function points are specified as symbolic block addresses. The assembly program converts these symbolic addresses to actual numerical values before control is transferred to GPSS III.

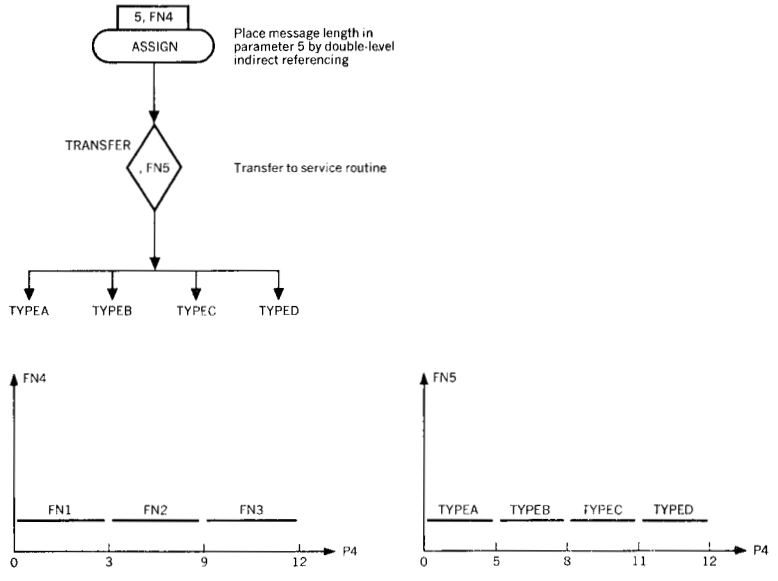
Table 1 Message lengths

Message type	*Number of characters in message*
1–3	80–126 as given by FN1
4–9	97–320 as given by FN2
10–12	50–150 as given by FN3

Table 2 Message routines

Message type	*Symbolic address of first block*
1–5	TYPEA
6–8	TYPEB
9–11	TYPEC
12	TYPED

Figure 3 Use of Attribute Valued function

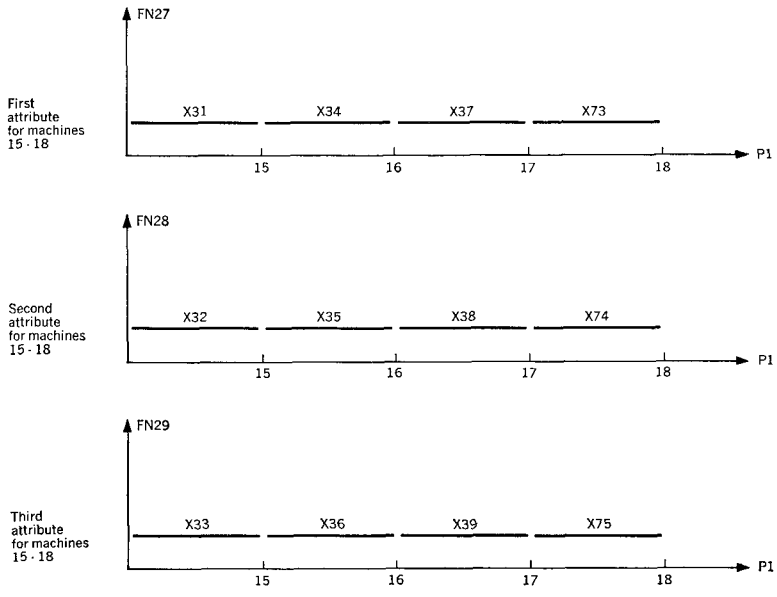


Another interesting use of the Attribute Valued function is two-dimensional addressing for arrays of savexes. Often, one wishes to associate several attributes with each of several entities, such as facilities. For instance, if facilities 15, 16, 17, and 18 represent machines in a machine shop, it may be convenient to have three attributes associated with each machine, representing, say, the type of job currently on the machine, the time that the current job started, and the number of jobs completed to date. These attributes can be arranged in a tabular array, and assigned savex locations, as shown in Table 3. This array can then be implemented, as shown in Figure 4, by three Attribute Valued functions, one for each attribute, using the facility number in P1 as the argument in each case. Thus, to determine whether machine No. 17 has completed ten jobs, the number 17 is placed in P1, and function 29 (representing the third attribute) is tested, as shown in Figure 5. Note that no auxiliary ASSIGN blocks and Variable statements are required to compute the addresses in the savex array. Also, it is not necessary to maintain contiguous savex locations, because there are no address computations.

Table 3 Savex array

	Machines			
	F15	F16	F17	F18
First attribute	X31	X34	X37	X73
Second attribute	X32	X35	X38	X74
Third attribute	X33	X36	X39	X75

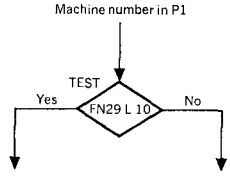
Figure 4 Attribute Valued functions for two-dimensional addressing



new debugging aids

Frequently, the most crucial effort in large-scale simulation is the debugging of the model and verifying that the model actually represents the system. Three debugging aids added to GPSS III help to alleviate this problem: (1) transaction chains may be printed out at any point in time, thus giving an output that previously was only obtainable by forcing an error, (2) complete system snapshots (i.e., all standard output and, if desired, chain printout) can be obtained at specified intervals during the run to study intermediate results, and (3) the PRINT block can now print any Standard Numerical Attribute, not just savexes. The snapshot feature is specified in the START card. For example, the card START 1000,,200,1 tells GPSS III to run until 1000 transactions have been terminated, and to print complete statistics on the state of the system after every 200 terminations. The 1 specifies that each time statistics are obtained, all transaction chains should be printed. The block PRINT 1,3,CHA causes a printout of the contents of user chains 1, 2, and 3 whenever a transaction enters this PRINT block. In all error printouts, the current state of the logic switches as well as standard statistical information is given.

Figure 5 Test of third machine attribute



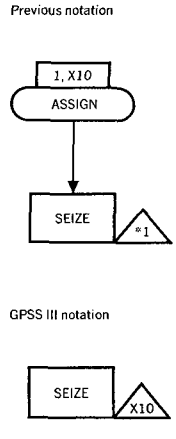
increased versatility

Some of the numerous new features that have added versatility and ease of use to GPSS are now described briefly.

Coding in GPSS III is facilitated by the provision of a free-field format. The most commonly used fields in each block are specified first, with fields separated by commas. For example, a GENERATE block with a mean of 250, modified by FUNCTION 1, with an offset time of 100, a count of 1000, and priority level of 5 is coded as GENERATE 250,FN1,100,1000,5.

Specification of Standard Numerical Attributes in any block field is now allowed. For example, an analyst wishing to seize the facility whose number is given by the value of savex 10, previously used two blocks. Now the same operations can be performed with a single block as shown in Figure 6.

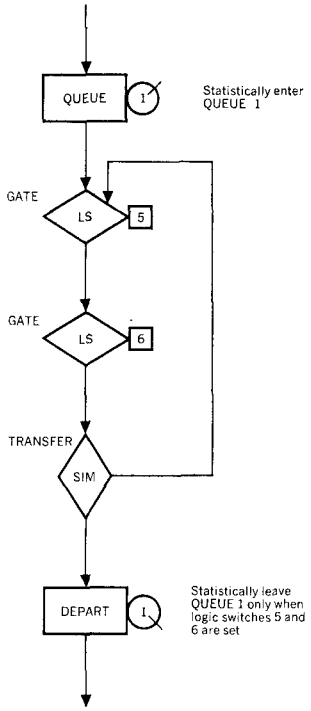
Figure 6 Specification of Standard Numerical Attribute



Queuing statistics are often desired on blocking conditions that cannot be represented by a single block, such as the condition of a group of logic switches. The QUEUE block of GPSS II cannot efficiently supply such statistics. To overcome this problem, the functions of the QUEUE block in GPSS III have been divided into two new blocks, the QUEUE block and the DEPART block. The new QUEUE block is similar in operation to the ENTER block associated with storages, and the new DEPART block is similar to the LEAVE block associated with storages. A transaction may leave a QUEUE block, but is considered a statistical member of the particular queue until it reaches the DEPART block. Figure 7 gives an example of the new blocks in use.

The SPLIT block has two new features in GPSS III. Specification of the number of transactions to be split is now allowed, eliminating the need for strings of SPLIT blocks. Also, serial numbering of copy transactions in a specified parameter can be automatically provided if desired. An example is shown in Figure 8.

Figure 7 Use of QUEUE/DEPART



A new block, the GATHER block, is similar to the ASSEMBLE block, except that no transactions are removed from the system. The block simply removes transactions from the Current Events chain until the count specified is reached. All delayed transactions are finally placed back on the Current Events chain in the same order in which they were removed and are then allowed to continue.

The HELP block has been extended to allow symbolic addressing in HELP routines. This feature greatly facilitates coding and permits use of the same coding for all machines and under any version of the operating system. Table 4 shows a HELP routine to print a specific line of output. OTITL and OUT are GPSS III routines and are used to set up and print out a line of output.

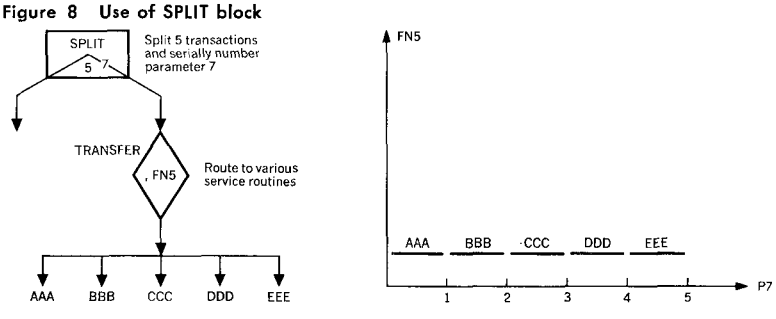
A new function type, the List function, requires that the independent argument values of the function be sequential integers, beginning with 1. In this mode, the program makes direct use of the value of the independent variable to obtain the corresponding function point. For example, if the value of the independent variable is 5, the fifth function point is immediately obtained. For functions that have both many points and sequentially numbered independent arguments, the List function should certainly be used. This function is evaluated more quickly than a comparable discrete function and saves core, since argument values are not stored.

An additional selection mode has been added to the TRANSFER block. This mode (subroutine) saves the current block number in a specified parameter and then transfers to a named subroutine. Thus, easy return to the main line flow is made possible after execution of the subroutine.

Table 4 HELP routine

```
HELP    SXA     ALPHA, 4            SAVE RETURN ADDRESS
        TSX     OTITL, 4            PRINT OUT LINE
        PZE     A, 10               OF OUTPUT FROM
        TSX     OUT, 4              LOCATION A
ALPHA   AXT     **, 4
        TRA     1, 4                TRANSFER BACK
A       BCI , 1THIS IS AN EXAMPLE OF OUTPUT EDITING USING GPSS III - - - -
```

Figure 8 Use of SPLIT block



The EXECUTE block lets the entering transaction perform the operation of any other specified block without diverting the transaction from its normal sequential flow.

The CHANGE block allows the analyst to modify the model during the course of a simulation by changing any one block to a duplicate of any other block.

summary

The significant new capabilities and performance improvements of GPSS III in comparison to the preceding General Purpose Systems Simulators can be summarized as follows:

- Increased speed
- Variable number of parameters
- Space reallocation
- User chains
- Attribute Valued functions
- START card and PRINT block
- Assembly program
- Standard Numerical Attributes in any field
- QUEUE and DEPART blocks
- SPLIT and GATHER blocks
- HELP block
- List function
- Subroutine mode of the TRANSFER block
- EXECUTE and CHANGE blocks

ACKNOWLEDGMENT

The authors wish to acknowledge the efforts of H. C. Ball, J. F. Bult, R. L. Gould, and E. C. Olsen who developed and implemented GPSS III.

CITED REFERENCE AND FOOTNOTE

1. Coverage has been extended from the IBM 7090 and 7094 computer systems to the IBM 7040 and 7044; the applicable GPSS III programs (No. 7090-CS-15X and 7040-CS-14X, respectively) can be ordered through IBM branch offices. Plans for extending GPSS III to SYSTEM/360 have been announced. Published by the IBM Data Processing Division, White Plains, New York, are three manuals on GPSS III: *Application Description* (H20-0144), *Introduction* (B20-0001), and *User's Manual* (H20-0163). An Application Description for the SYSTEM/360 version is forthcoming.
2. R. Efron and G. Gordon, "A general purpose digital simulator and examples of its application, Part I, description of the simulator," *IBM Systems Journal* **3**, No. 1, 22–34 (1964).