

# Simulation and Modeling

## Simulation Languages

## **Contents**

<b>5.1 Discrete systems modeling and simulation with GPSS -</b>	<b>1</b>
5.2.1 GPSS programs	1
5.2.2 General description	1
5.2.2 GPSS block –diagram symbols	2
<b>5.2 Basic concepts</b>	<b>5</b>
5.2.1. Action Times	5
5.2.3 Succession of events	6
5.2.3 Choice of paths	7
<b>5.3 Resources in GPSS, GPSS programs, applications</b>	<b>7</b>
5.3.1 Simulation of manufacturing shop	7
5.3.2 Facilities and storage	9
5.3.3 Gathering statistics	10
5.3.4 Simulation of supermarket	13
<b>5.4 Continuous systems modeling and simulation with CSMP</b>	<b>15</b>
5.4.1 Block Diagram and examples	15
<b>5.5 Structural, data and control statements, hybrid simulation</b>	<b>15</b>
5.5.1 SIMSCRIPT, Example of hybrid simulation using GPSS16	
<b>5.6 Feedback system and typical applications</b>	<b>16</b>
5.6.1 Simulation of a telephone system	16

02058

## **SIMULATION AND MODELING**

### **5. SIMULATION LANGUAGE**

#### **5.1 Discrete systems modeling and simulation with GPSS**

To represent the state of the system in the model, a set of numbers is used that is used in discrete time system. A number, state descriptor is used to represent some aspect of the system state. Some state descriptors range over values that have physical significance and some represent conditions

The effect of the state descriptors is to change value as the simulation proceeds. We define a discrete event as a set of circumstances that causes an instantaneous change in one or more system state descriptors. It is possible that two different events occur simultaneously, or are modeled as being simultaneous, so that not all changes of state descriptors occurring simultaneously necessarily belong to a single event.

The term simultaneous refers to the occurrence of the changes in the system, and not to when the changes are made in the model-in, which, of necessity, the changes must be made sequentially. A system simulation must contain a number representing time. The simulation proceeds by executing all the changes to the system descriptors associated with each event, as the events occur, in chronological order. The way in which events are selected for execution (when there are simultaneous events) is an important aspect of programming simulations.

For writing discrete system simulation programs, a number of programming languages have been produced. These programs embody a language with which to describe the system, and a programming system that will establish a system image and execute a simulation algorithm. Each language is based upon a set of concepts used for describing the system. The term world-view has come to be used to describe this aspect of simulation programs. The user of the program must learn the world-view of the particular language he is using and to describe the system in those terms. Given such a description, the simulation programming system is able establish a data structure that forms the system image. It will also compile and sometimes supply the routines to carry out the activities. One of the most commonly used simulation language is GPSS. It illustrates the divergence in design consideration. GPSS has been written specially for user with little or no knowledge of programming experience. The simplification of GPSS results in some loss of flexibility. GPSS is applicable to a wide variety of systems.

##### **5.1.1 GPSS programs**

The General Purpose Simulation System language has been developed over many years, principally by the IBM Corporation. It has been implemented on several different manufacturers' machines, and there are variations in the different implementations. GPSS V is a more powerful and has more language statements and facilities than GPSS/360. The GPSS V is implemented by IBM Corporation.

##### **5.1.2 General description**

The system to be simulated in GPSS is described as a block diagram in which the blocks represent the activities and lines joining the blocks indicate the sequence in which

the activities can be executed. Where there is a choice of activities, more than one line leaves a block and the condition for the choice is stated at the block.

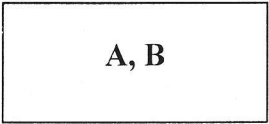
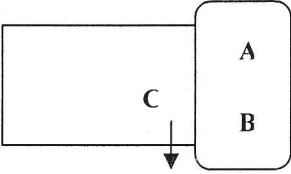
To base a programming language on this descriptive method, each block must be given a precise meaning. The approach taken in GPSS is to define a set of 48 specific *block types*, each of which represents a characteristic action of the systems. Only the specified *block types* are allowed while drawing the block diagram of the system. Each block type is given a name that is descriptive of the block action and is represented by a particular symbol. Each block type has a number of data fields.

Moving through the system being simulated are entities that depend upon the nature of the system. In a communication system, the entities of concern are messages, which are moving. Meanwhile in a road transportation system the entities that are moving are the motor vehicles. A data processing system is concerned with records. In the simulation, these entities are called transactions. The sequence of events in real time is reflected in the movement of transactions from block to block in simulated time.

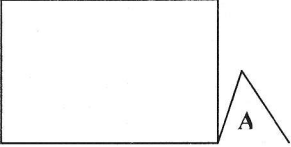
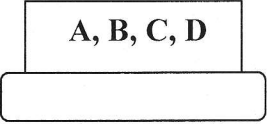
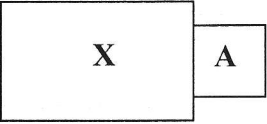
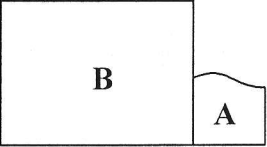
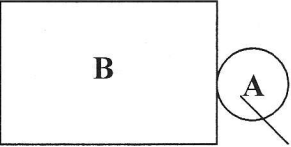
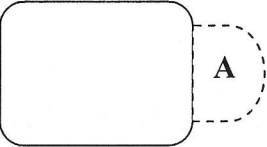
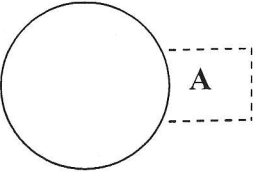
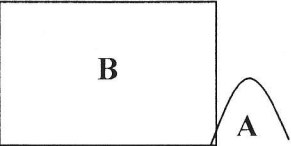
Transactions are created at one or more GENERATE blocks and are removed from the simulation at TERMINATE blocks. There can be many transactions simultaneously moving through the block diagram. Each transaction is always positioned at a block and most blocks can hold many transactions simultaneously. The transfer of a transaction from one block to another occurs instantaneously at a specific time or when some change of system condition occurs.

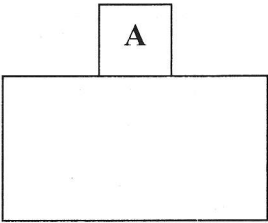
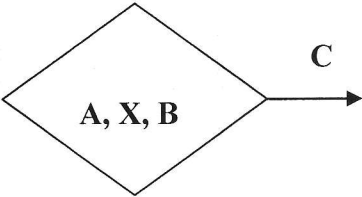
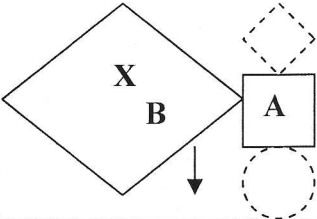
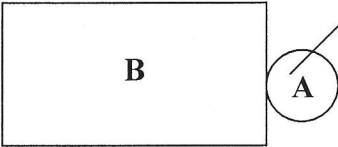
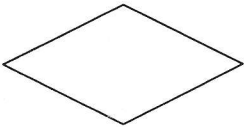
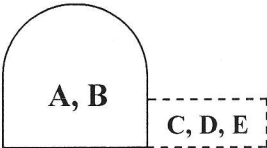
A GPSS block diagram can consist of many blocks up to some limit prescribed by the program (usually set to 1000). An identification number called a location is given to each block, and the movement of transactions is usually from one block to the block with the next highest location. The locations are assigned automatically by an assembly program within GPSS so that, when a problem is coded, the blocks are listed in sequential order. Blocks that need to be identified in the programming of problems are given a symbolic name. The assembly program will associate the name with the appropriate location. Symbolic names of blocks and other entities of the program must be from three to five non-blank characters of which the first three must be letters.

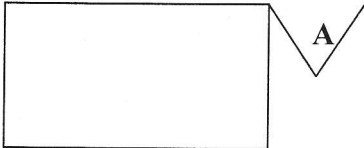
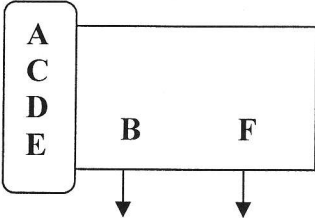
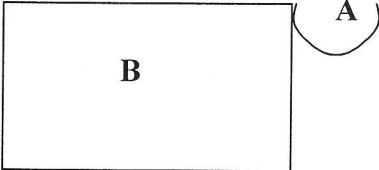
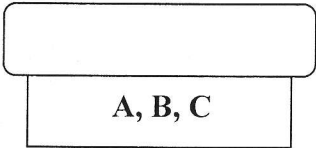
### 5.1.3 GPSS block-diagram symbols

S.N.	Representation/Symbols	Meaning
1		Advance
2		Link



3		Seize
4		Assign
5		Logic
6		Tabulate
7		Depart
8		Mark
9		Terminate
10		Enter

11		Priority
12		Test
13		Gate
14		Queue
15		Transfer
16		Generate

17		Release
18		Unlink
19		Leave
20		Save Value

## **5.2 Basic concepts**

### **5.2.1 Action times**

Clock Time is represented by an integral number, with the interval of real time corresponding to a unit of time chosen by the program user. The unit of time is not specifically stated but is implied by giving all the time in terms of the same unit. One block type called ADVANCE is concerned with representing the expenditure of time. The program computes an interval of time called action time for each transaction as it enters an ADVANCE block, and the transaction remains at the block for this interval of simulated time before attempting to proceed. The only other block type that employs action time is the GENERATE block, which creates transactions. The action time at the GENERATE block controls the interval between successive arrivals of transactions.

The action time may be a fixed interval or a random variable, and it can be made to depend upon conditions in the system in various ways. An action time is defined by giving a mean and modifier as the A and B fields for the block. If the modifier is zero, the action time is a constant equal to the mean. If the modifier is a positive number ( $\leq \text{mean}$ ), the action time is an integer random variable chosen from the range ( $\text{mean} \pm \text{modifier}$ ), with equal probabilities given to each number in the range.

By specifying the modifier at an ADVANCE or GENERATE block to be a function, the value of the function controls the action time. The action time is derived by multiplying the mean by the value of the function. Various types of input can be used for the functions, allowing the functions to introduce a variety of relationships among the variable of a system. In particular, by making the function an inverse cumulative probability distribution, and using as input a random number which is uniformly distributed, the function can provide a stochastic variable with a particular non-uniform distribution.

The GENERATE block normally begins creating transactions from zero time, and continues to generate them throughout the simulation. The C field, however, can be used to specify an offset time as the time when the first transaction will arrive. If the D field is used, it specifies a limit to the total number of transactions that will come from the block. Transactions have a priority level and they carry items of data called parameters. The E field determines the priority of the transactions at the time of creation. If it is not used, the priority is of the lowest level.

Parameters can exist in four formats. They can be signed integers of fullword, halfword, or byte size, or they can be signed floating-point numbers. If no specific assignment of parameter type is made, the program creates transactions with 12 halfword parameters. The C, D and E fields, also, will not be needed.

### 5.2.2 Succession of events

The program maintains records of when each transaction in the system is due to move. It proceeds by completing all movements that are scheduled for execution at a particular instant of time and that can logically be performed. Where there is more than one transaction due to move, the program processes transactions in the order of their priority class, and on a first-come, first-served basis within priority class.

Once the program has begun moving a transaction it continues to move the transaction through the block diagram until one of the several circumstances arises.

The transaction may enter an ADVANCE block with a non-zero action time, in which case, the program will turn its attention to other transactions in the system and return to that transaction when the action time has been expended.

The conditions in the system may be such that the action the transaction is attempting to execute by entering a block cannot be performed at the current time. The transaction is said to be blocked and it remains at the block it last entered. The program will automatically detect when the blocking condition has been removed and will start to move the transaction again at that time.

A third possibility is that the transaction enters a TERMINATE block, in which case it is removed from the simulation. A fourth possibility is that a transaction may be put on a chain.

When the program has moved one transaction as far as it can go, it turns its attention to any other transactions due to move at the same time instant. If all such movements are complete, the program advances the clock to the time of the next most imminent event and repeats the process of executing events.

### 5.2.3 Choice of paths

The TRANSFER block allows some location other than the next sequential location to be selected. The choice is normally made between two blocks referred to as next blocks A and B. The method used for choosing is indicated by a selection factor in the field A of the TRANSFER block. It can be set to indicate one of the nine choices. Next blocks A and B are placed in fields B and C, respectively. If no choice is to be made, the selection factor is left blank. An unconditional transfer is then made to next block A.

A random choice can be made by setting the selection factor, S, to a three-digit decimal fraction. The probability of going to next block A is then  $1-S$ , and to the next block B it is  $S$ . a conditional mode, indicated by setting field A to BOTH, allows a transaction to select an alternate path depending upon existing conditions. The transaction goes to next block A if this move is possible, and to the next block B if it is not. If both moves are impossible the transaction waits for the first to become possible, giving preference to A in the event of simultaneity.

## 5.3 Resources in GPSS, GPSS programs, applications

### 5.3.1 Simulation of manufacturing shop

Let us consider a scenario representing a machine tool in a manufacturing shop which is turning out parts at the rate of every 5 minutes. The parts are then examined by the inspector who takes  $4 \pm 3$  minutes for each part and rejects about 10% of the parts. We can represent each part by one transaction, and the time unit selected for the problem will be 1 minute. A block diagram representing the system is given below:

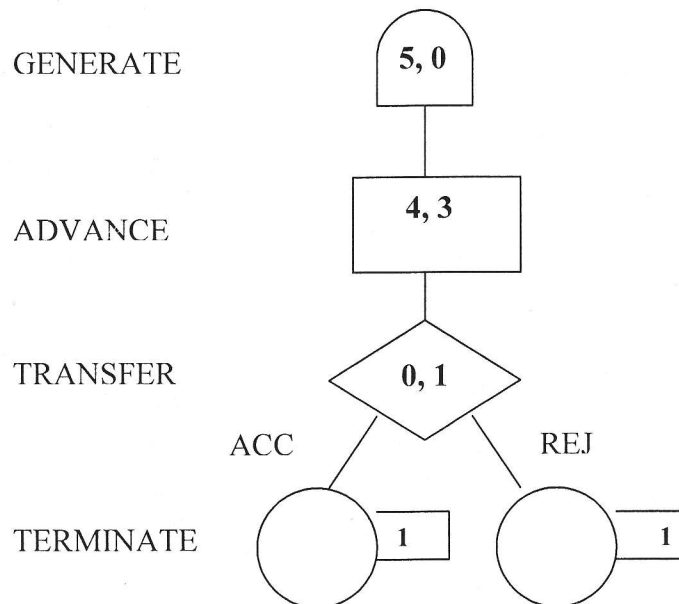


Fig: Manufacturing shop- model 1

In the block diagram, the block location is placed at the top of the block, the action time is indicated in the center in the form  $T=a, b$  where  $a$  is the mean and  $b$  is the modifier and the selection factor is placed at the bottom of the block.

A GENERATE block is used to represent the output of the machine by creating one transaction every five units of time. An ADVANCE block with a mean of 4 and modifier of 3 is used to represent inspection. The time spent on inspection will therefore be any one of the values 1, 2, 3, 4, 5, 6 or 7, with equal probability given to each value. Upon completion of the inspection, transactions go to a TRANSFER block with a selection factor of 0.1, so that 90% of the parts go to the next location called ACC, to represent accepted parts and 10% go to another location called REJ to represent rejects. Both locations reached from the TRANSFER block are TERMINATE blocks.

The problem can be coded. Column 1 is only used for a comment card. A field from columns 2 to 6 contains the location of the block where it is necessary for it to be specified. The GPSS program will automatically assign sequential location numbers as it reads the statements, so it is not usually necessary for the user to assign locations. The TRANSFER block, however, will need to make reference to the TERMINATE blocks to which it sends transactions, so these blocks have been given symbolic location names ACC and REJ.

The second section of the coding, from columns 8 to 18, contains the block type name, which must begin in column 8. Beginning at column 19, a series of fields may be present, each separated by commas and having no embedded blanks. Anything following the first blank is treated as a comment. The meaning of the fields depends upon the block type.

The program also accepts input in a free format, in which the location field, if used, begins in column 1; but none of the other fields has a fixed starting position. Instead a single blank marks the transition from the location field to the operation field and from the operation field to the operands. If no location is specified, an initial blank is used.

For the TRANSFER block, the first field is the selection factor, and the B and C fields are exits 1 and 2, respectively. In this case, exit 1 is the next sequential block, and it would be permissible to omit the name ACC in both the TRANSFER field B and the location field of the first TERMINATE block. It should also be noted that when a TRANSFER block is used in an unconditional mode, so that field A is not specified, a comma must still be included to indicate the field.

The program runs until a certain count is reached as a result of transactions terminating. Field A of the TERMINATE block carries a number indicating by how much the termination count should be incremented when a transaction terminates at that block. The number must be positive and it can be zero, but there must be at least one TERMINATE block that has a non-zero field A. In this case, both TERMINATE blocks have 1; so the terminating counter will add up the total number of transactions that terminate, in other words, the total number of both good and bad parts inspected.

A control statement called START indicates the end of the problem definition and contains, in field A, the value the terminating counter is to reach to end the simulation. In this case, the START Statement is set to stop the simulation at a count of 1,000. Upon reading a START statement, the program begins executing the simulation.

When the simulation is completed, the program automatically prints an output report, in a prearranged format, unless it has been instructed otherwise.

The problem input is printed first, with the locations assigned by the problem listed to the left, and a sequential statement number on the right. The first line of the output following the listings gives the time at which the simulation stopped. The time is followed by a listing of block counts. Two numbers are shown for each block of the model. On the left is a count of how many transactions were in the block at the time the simulation stopped, and on the right is a figure showing the total number of transactions that entered the block during the simulation.

### **5.3.2 Facilities and storage**

Associated with the system being simulated are many permanent entities, such as items of equipment, which operate on the transactions. Two types of permanent entities are defined in GPSS to represent system equipment.

A facility is defined as an entity that can be engaged by a single transaction at a time. A storage is defined as an entity that can be occupied by many transactions at a time, up to some predetermined limit. A transaction controlling a facility, however, can be interrupted or preempted by another transaction. In addition, both facilities and storages can be made unavailable, as would occur if the equipment they represent breaks down, and can be made available again, as occurs when a repair has been made.

There can be many instances of each type of entity to a limit set by the program (usually 300). Individual entities are identified by number, a separate number sequence being used for each type. The number 0 for these, and all other GPSS entities, is illegal.

Block types SEIZE, RELEASE, ENTER and LEAVE are concerned with using facilities and storages. Field A in each case indicates which facility or storage is intended, and the choice is usually marked in the flag attached to the symbols of the blocks. The SEIZE block allows a transaction to engage a facility if it is available. The RELEASE block allows a transaction to disengage the facility. An ENTER block allows a transaction to occupy a place in storage, if it is available, and the LEAVE block allows it to give up the space. If the fields B of the ENTER and LEAVE blocks are blank, the storage contents are changed by 1. If there is a number ( $\geq 1$ ), then the contents change by that value. Any number of blocks may be placed between the points at which a facility is seized and released to simulate the actions that would be taken while transaction has control of a facility. Similar arrangements apply for making use of storages.

To illustrate the use of these block types, consider again the manufacturing shop. Since the average inspection time is 4 minutes and the average generation rate is one every 5 minutes, there will normally be only one part inspected at a time. Occasionally, however, a new part can arrive before the previous part has completed its inspection. This situation will result in more than one transaction being at the ADVANCE block at one time.

Assuming that there is only one inspector, it is necessary to represent the inspector by a facility, to simulate the fact that only one part at a time can be inspected. A SEIZE block and a RELEASE block need to be added to simulate the engaging and disengaging of the inspector.

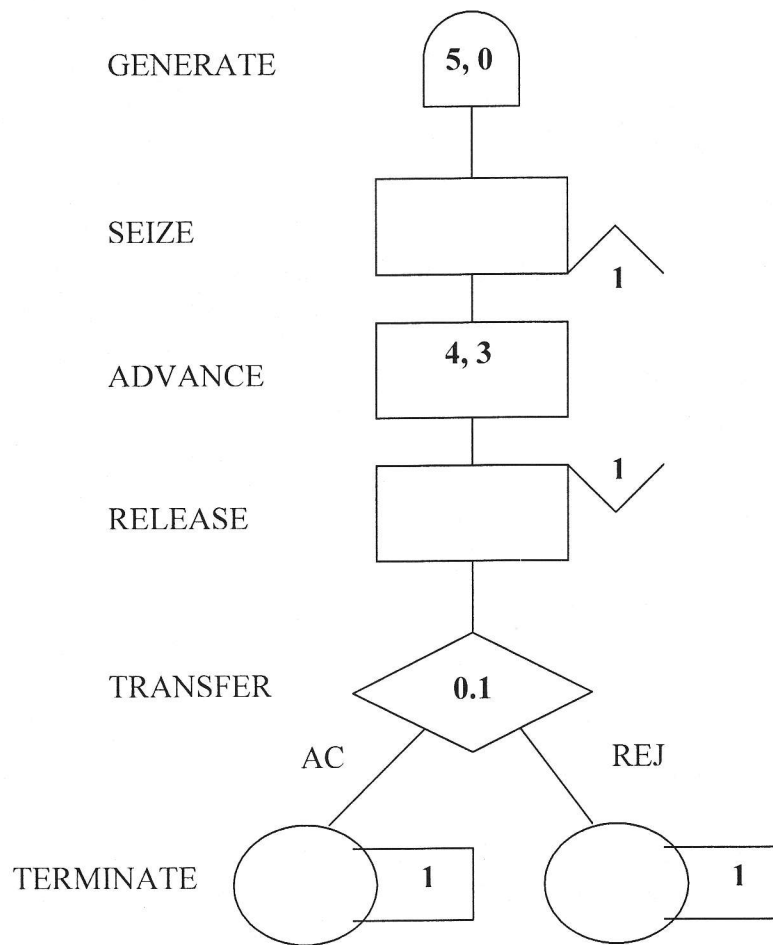


Fig: Manufacturing shop- model 2

If more than one inspector is available, they can be represented as a group by a storage with a capacity equal to the number of inspectors. The SEIZE and RELEASE blocks of the previous model are then replaced by an ENTER and a LEAVE block, respectively. Suppose, for example, the inspection time were three times as long as before; three inspectors would be justified.

The case of a single inspector can be modeled by using storage with capacity 1 instead of facility. An important logical difference between the two possible representations is that, for a facility, only the transaction that seized the facility can release it, whereas, for a storage, entering and leaving can be separate actions carried out independently by different transactions.

### 5.3.3 Gathering statistics

Certain block types in GPSS are constructed for the purpose of gathering statistics about the system performance, rather than of representing system actions. The QUEUE,



DEPART, MARK and TABULATE blocks serve this purpose. They introduce two other entities of the GPSS program, queues and tables. As with facilities and storages, there can be many such entities up to a prescribed limit (usually, 300 for queues and 100 for tables) and they are individually identified by a number or symbolic name.

When the conditions for advancing a transaction are not satisfied, several transactions may be kept waiting at a block. When the conditions are favorable, they are allowed to move on, according to priority and usually by first-in, first-out rule. No information about the queue of transactions is gathered, however, unless they have been entered into a queue entity. The QUEUE block increases and the DEPART block decreases the queue numbered in field A. If field B is blank, the change is a unit change; otherwise the value of field B ( $\geq 1$ ) is used.

It is also desirable to measure the length of time taken by transactions to move through the system or parts of the system and this can be done with the MARK and TABULATE blocks. Each of these block types notes the time a transaction arrives at the block. The MARK block notes the time of arrival on the transaction. The TABULATE block subtracts the time noted by a MARK block from the time of arrival at the TABULATE block. The time, referred to as transit time, is entered in a table whose number or name is indicated in field A of the TABULATE block.

If one part arrives when all the inspectors are busy, the machining of further parts stops until the machine is cleared. It could also be that parts will accumulate on the inspectors' work bench if inspection does not finish quickly enough. A QUEUE block using queue number 1 is placed immediately before the ENTER block, and a DEPART block is placed immediately after the ENTER block to remove the part from the queue when inspection begins. Any transaction that does not have to wait for an inspector to become available will move through the queue without delay. Those that must wait will do so in the QUEUE block, and the program will automatically measure the length of stay in the queue. A MARK and a TABULATE block will measure how long the parts take to be inspected, excluding their waiting time in the queue. If MARK block is omitted, the tabulated time is the time since the transactions first entered the system.

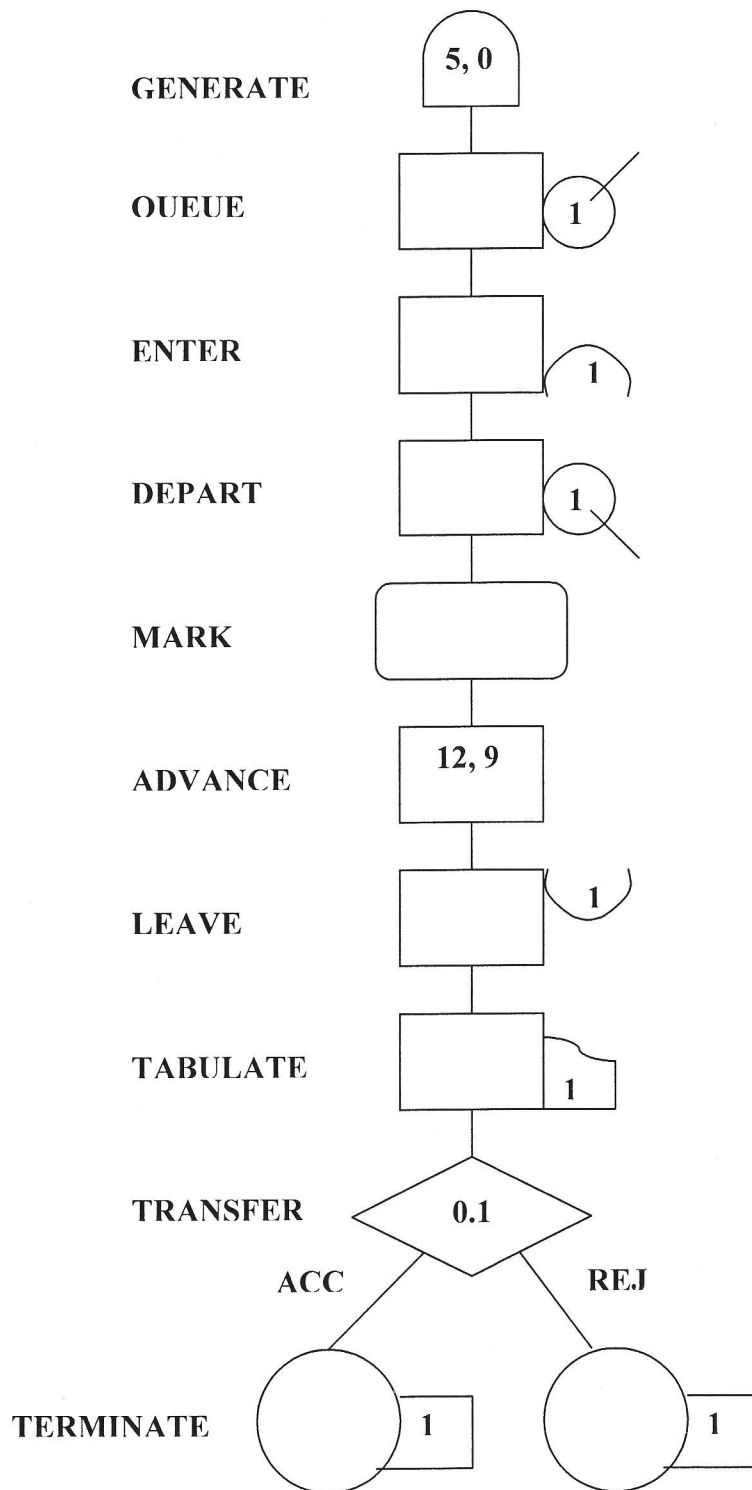


Fig: Manufacturing shop – model 3

#### 5.3.4 Simulation of supermarket

Parameters are items of data that represent attributes of the transaction. In addition, there are attributes of other entities of the system, such as the number of transactions in storage or the length of a queue, which are made available to the program user. Collectively, they are called Standard Numerical Attributes (SNA).

To illustrate the use of functions, parameters and SNAs, a simulation model can be written for a supermarket that operates in the following manner. Customers of the supermarket are obliged to take a basket before they begin to shop. There is a limited number of baskets and if no basket is available when they arrive, customers leave without shopping. If they get a basket, customers shop and then check-out at one of five check-out counters. After checking out, they return the baskets and leave the supermarket.

There are four sections in the system

- a) Getting a basket
- b) Shopping
- c) Checking-out
- d) Leaving

Each shopper is represented by a transaction and the unit of time is 1 second.

Function number 1 of the simulation model controls the inter-arrival time at a GENERATE block with a mean of 36.

To represent the baskets, a storage denoted by BSKT is used with capacity equal to the number of baskets.

The block count at AWAY will show the number of customers turned away for lack of baskets.

The number of items is determined by an ASSIGN block which has SNA called FN2 in field B.

The counters are regarded as a service unit with five servers. They are represented by storage, named CKT that has a capacity of 5.

Field A of ADVANCE block representing the checking-out time is set to V1.

Transit time is tabulated in a table called TRT.

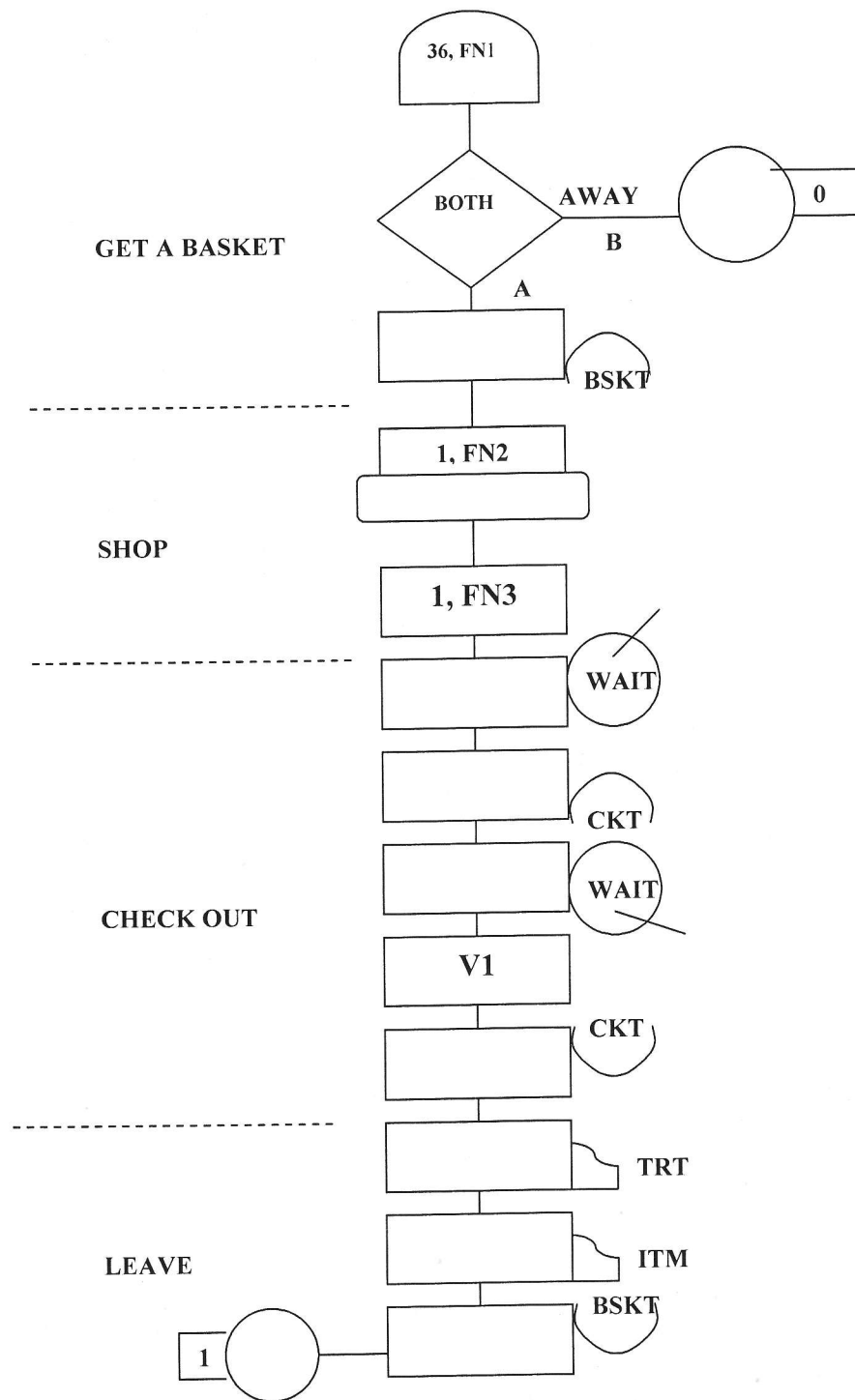


Fig: Simulation of a supermarket

## **5.4 Continuous systems modeling and simulation with CSMP**

Continuous systems are predominant activities of the system because smooth changes in the attributes of the system entities. It is possible to solve the simple differential equation models having one or more linear differential equation with constant coefficients, without use of simulation. But the cost of time and labor needed may be so high that it is often preferable to use the simulation techniques. Also in case of introduction of nonlinearities into the models, it becomes almost impossible or very different to solve the models. Simulation methods of solving the models do not change fundamentally when nonlinearities occur. Showing their application to models where the differential equations are linear and have constant coefficients, and then generalizing to more complex equations can therefore develop the method of applying simulation to continuous models.

### **5.4.1 Block diagram and examples**

Continuous system simulation language (CSSL) use a type of input for digital computers allowing a problem to be programmed directly from the equations of a mathematical model rather than requiring the equations to be broken into functional elements. These languages can include macros or subroutines that perform the function of specific analog elements so that it is possible to incorporate the convenience of a digital analog simulator. Beyond the provision of simple analog functions such as addition and integration, continuous system simulation Language includes a variety of algebraic and logical expressions to describe the relation between variables. They therefore remove the orientation towards linear differential equation which characterizes analog methods. Continuous System Modeling Program (CSMP) is one of such Continuous System Simulation language.

CSMP is constructed from three general types of statements-Structure Statements, Data Statements and Control Statements. Structure statements consist of FORTRAN like statements and functional blocks designed for operations that frequently occur in a model definition. Structure statement thus defines the model. Structural statement can make use of the operation of addition, subtraction, multiplication, division and exponential using the same notation and rules as used in FORTRAN.

## **5.5 Structural data and control statements, hybrid simulation**

An analog and a digital computer can at times be combined to provide a hybrid simulation. The form taken by hybrid simulation depends upon the application. One computer may be simulating the system being studied, while the other is providing a simulation of the environment in which the system is to operate. It is possible that the system being simulated is an interconnection of continuous and discrete subsystem, which can best be modeled by an analog and digital computer being linked together.

The term "hybrid simulation" is generally reserved for the case in which functionally distinct analog and digital computers are linked together for the purpose of simulation.

### **5.5.1 SIMSCRIPT, Example of hybrid simulation using GPSS**

SIMSCRIPT is a very widely used language for simulating discrete systems. The language can be considered as more than just a simulation language since it can be applied to general programming problems.

#### **SIMSCRIPT system concepts**

The system to be simulated is considered to consist of entities having attributes that interact with activities that cause events that change the state of the system. SIMSCRIPT uses the terms entities and attributes in describing the system. The user can define sets, and facilities are provided for entering and removing entities to and from sets.

### **5.6 Feedback system and typical applications**

A significant factor in the performance of many systems is that a coupling occurs between the input and output of the system. The term feedback is used to describe the phenomena. A home heating system controlled by a thermostat is a simple example of a feedback system. The system has a furnace whose purpose is to heat a room, and the output of the system can be measured as the room temperature. Depending upon whether the temperature is below or above the thermostat setting, the furnace will be turned on or off, so that information is being fed back from the output to the input. In this case, there are only two states; either the furnace is on or off. Other cooling systems used also employ the same concept/principle of feedback for their operation.

#### **5.6.1 Simulation of a telephone system**

The assumed telephone system has a number of telephones connected to a switchboard by lines. The switchboard has a number of links which can be used to connect any two lines, subject to the condition that only one connection at a time can be made to each line. It will be assumed that only one connection at a time can be made to each line. It will be assumed that the system is a lost-call system, that is, any call that cannot be connected at the time it arrives is immediately abandoned. A call maybe blocked or busy. The object of simulation will be to process a given number of calls and determine what proportion are successfully completed, blocked or found to be busy calls.

Each line is treated as an entity, having its availability as an attribute. A zero in the table means the line is free, while a one means that it is busy. It is only necessary to incorporate in the model the constraint imposed by the fact that there is a fixed number of links. Each line is represented by a logic switch whose number is the line number. Each call is represented by one transaction; the unit of time chosen is 1 second.

To keep track of events, a number representing clock time is included. The clock will be updated to the next most imminent event as the simulation proceeds. Each call is a separate entity having as attributes its origin, destination, length, and the line at which the call finishes. To generate the arrival of calls, the bootstrap method is used, so that a record is kept of the time the next call is due to arrive. It will be assumed that the call is equally likely to come from any line, other than itself, irrespective of whether that line is busy or not.

There are two activities causing events: new calls can arrive and existing calls can finish.

The simulation proceeds by executing a cycle of steps to simulate each event. The first step is to scan the events to determine which the next potential event is. The clock is updated, and the second step is to select the activity that is to cause the event. The third step is to test whether the potential event can be executed. The fourth step is to change the records to reflect the effects of the event. As a fifth and final step in the execution cycle, it may be necessary to gather some statistics for the simulation output. Assuming the simulation is to continue, the cycle of actions just described is repeated.

The procedure will be repeated until some limit on the length of the simulation is reached. Typically, the simulation will run until a given number of calls have been processed or until a certain time has elapsed.

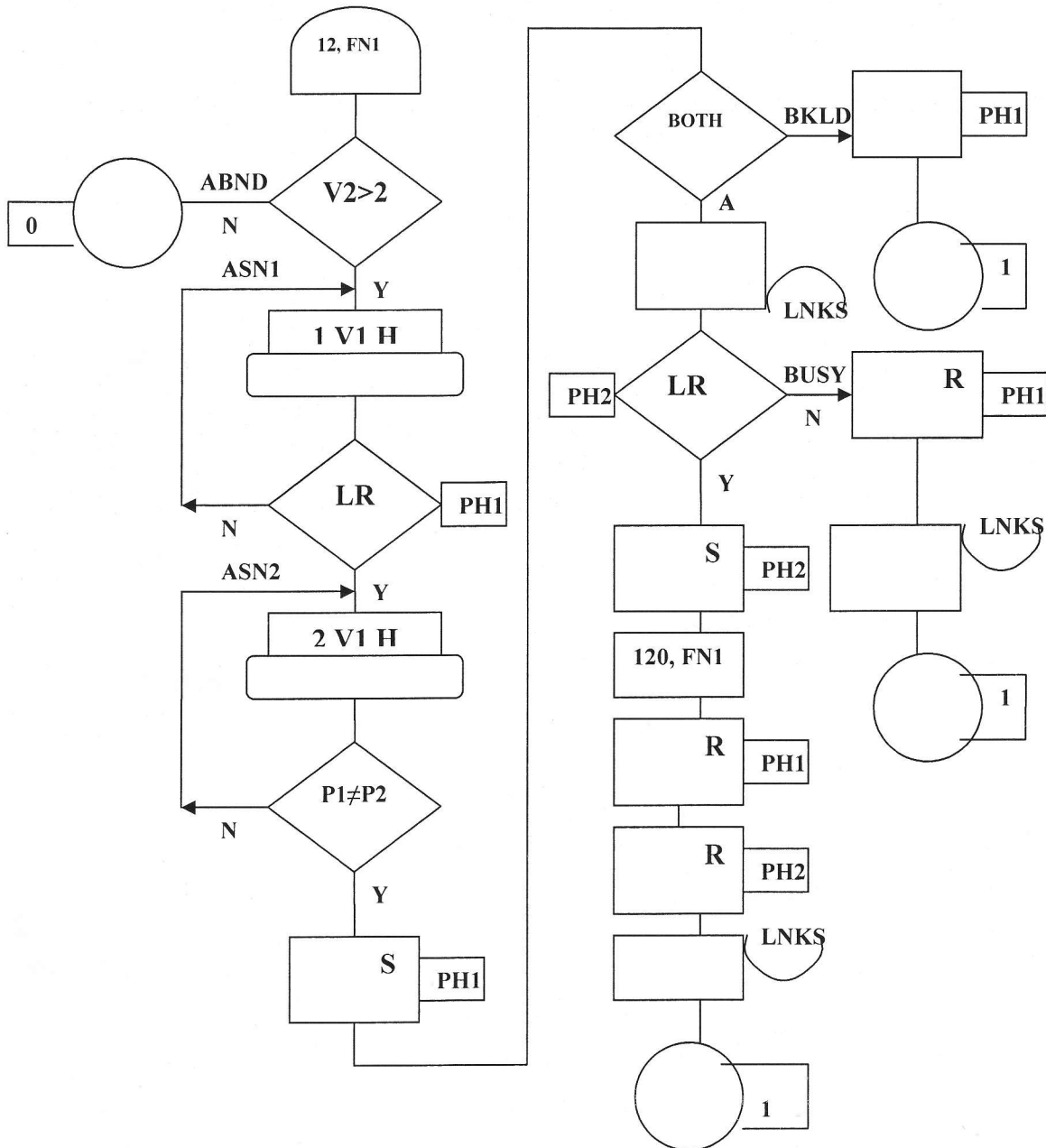


Fig: Telephone system in GPSS