AN INTERACTIVE SIMULATION PROGRAMMING SYSTEM WHICH CONVERSES IN ENGLISH

George E. Heidorn

Naval Postgraduate School

Abstract

In this paper an experimental system for producing simulation programs through natural language dialogue with a computer is described. With the current version of this system, which operates under CP/CMS on an IBM 360/67, a queuing problem may be stated to the computer in English. The system checks the problem description for completeness, and if necessary, asks questions which may be answered in English. Then it can produce an English description of the problem as it "sees" it and a GPSS program for performing the simulation. The user may then modify the problem through further dialogue to produce additional programs, as desired. A complete sample problem is included in the paper.

1. Introduction

This paper reports on work which is being done at the Naval Postgraduate School to develop a system for performing simulation analyses through natural language (e.g. English) interaction with a computer. The eventual goal is to enable an analyst seated at a terminal to "talk" with the computer about his simulation problem in his own natural language, and have the computer "understand" the problem and do the simulation, reporting the results in the same natural language. In such a system the computer would function as an intelligent assistant with great computing powers - actually a combination programmer and computer.

The status of this system as of the time of this writing is that it can carry on a dialogue in English about a simple queuing problem and then produce a GPSS program to do the simulation. This is essentially the programmer part of the total system described above.

This project has been influenced by work being done in the areas of simulation programming languages, artificial intelligence, and linguistics. The entity-attribute-value "world view" of both SIMSCRIPT [1] and GPSS [2] plays a central role in this system. Ideas have been borrowed from artificial intelligence work on natural-language question-answering systems [3,4], and the language processing has been done within the framework of Stratificational Linguistics [5,6].

This paper is intended to serve as an introduction to the system being developed and to demonstrate some of the capabilities of the current version. Another introductory paper based on a slightly earlier version is available [7], and a very detailed technical report has been prepared [8]. This paper consists primarily of a sample problem and an explanation of its overall processing, without details of the linguistic aspects.

2. The Sample Problem

Figures 1, 2, and 3 show a terminal session with this system for a sample problem. The statement numbers which appear there were added for ease of reference in the discussion. All of the lower case typing was done by the user, and all of the UPPER CASE by the computer.

```
 1.  vehicles arrive at a station.

 2.  the station has just one pump.

 3.  a vehicle will leave the station immediately
     after arriving if the length of the line at
     the pump is not less than two.

 4.  otherwise, it is serviced there; then it
     leaves.

 5.  service times are exponential, with a mean of
     5 minutes for cars and 9 minutes for trucks.

 6.  three quarters of the vehicles are cars and
     one fourth of them are trucks.

 7.  is the statement of the problem complete?

     HOW OFTEN DO THE VEHICLES ARRIVE AT THE
     STATION?

 8.  arrivals are normally distributed with a mean
     of 8 minutes and a standard deviation of 2
     minutes.

     HOW LONG SHALL THE SIMULATION BE RUN?

 9.  the basic time unit to be used in the model
     is 30 seconds.

     HOW LONG SHALL THE SIMULATION BE RUN?

10.  eight hours.

             THE PROBLEM STATEMENT IS COMPLETE.
```

Figure 1. The English statement of the problem being given to the system.

The first figure shows the problem description being given to the system. As can be seen there, the user entered six statements and then asked if the problem description was complete. The system realized that information about the inter-arrival times was missing, so it asked a question about that, which the user responded to with a complete sentence. Then the system asked how long the model was to be run, but the user chose to give information about the basic time unit instead, which would have been the subject of the next question. After processing the reply,

11. describe the problem in english.

```
     THE VEHICLES ARRIVE AT THE STATION.  THE
TIME BETWEEN ARRIVALS OF THE VEHICLES AT THE
STATION IS NORMALLY DISTRIBUTED, WITH A MEAN OF
8 MINUTES AND A STANDARD DEVIATION OF 2 MINUTES.
75 PERCENT OF THE VEHICLES ARE CARS, AND THE
REST ARE TRUCKS.  AFTER ARRIVING AT THE STATION,
IF THE LENGTH OF THE LINE AT THE PUMP IN THE
STATION IS LESS THAN 2, THE VEHICLE WILL BE
SERVICED AT THE PUMP IN THE STATION.  OTHERWISE,
THE VEHICLE WILL LEAVE THE STATION.  THE TIME
FOR THE VEHICLES TO BE SERVICED AT THE PUMP IN
THE STATION IS EXPONENTIALLY DISTRIBUTED, WITH A
MEAN OF 5 MINUTES FOR THE CARS, AND 9 MINUTES
FOR THE TRUCKS.  AFTER BEING SERVICED AT THE
PUMP IN THE STATION, THE VEHICLES LEAVE THE
STATION.
     THE SIMULATION IS TO BE RUN FOR 8 HOURS,
USING A BASIC TIME UNIT OF 30 SECONDS.
```

Figure 2. The English problem description produced by the system.

the system again asked the question about the run time. This time the user responded with the phrase, "eight hours". Then the system announced that it had a complete problem statement, i.e. one for which it could produce a GPSS program.

In the second figure there is a description of the problem "in the computer's own words", which the user can read carefully to see if the system "understood" him. Then the third figure shows the GPSS program produced by the system for performing the simulation.

3. The Overall Approach

If a simulation programmer were given a queuing problem stated in a natural language, he would probably read it one or more times to form a "mental image" of the system being described and to note the points of interest in it. If the description were not clear to him or if essential information were missing, he might ask questions

12. write a gpss program for this problem.

```
        SIMULATE
        RMULT      277,423,715,121,655,531,999,813
 STAT1  EQU        1,F,Q
 PUMP2  EQU        2,F,Q
 CAR2   EQU        2,T
 2      TABLE      M1,1,1,2
 TRUC3  EQU        3,T
 3      TABLE      M1,1,1,2
 1      FUNCTION   RN1,C24
0.0,0.0/.100,.104/.200,.222/.300,.355/
.400,.509/.500,.690/.600,.915/.700,1.200/
.750,1.390/.800,1.600/.840,1.830/.880,2.120/
.900,2.300/.920,2.520/.940,2.810/.950,2.990/
.960,3.200/.970,3.500/.980,3.900/.990,4.600/
.995,5.300/.998,6.200/.999,7.000/1.000,8.000/
 2      FUNCTION   RN2,C29
0.0,-3.000/.012,-2.250/.027,-1.930/.043,-1.720/
.062,-1.540/.084,-1.380/.104,-1.260/.131,-1.120/
.159,-1.00/.187,-.890/.230,-.740/.267,-.620/
.334,-.430/.432,-.170/.500,0.0/.568,.170/
.666,.430/.732,.620/.770,.740/.813,.890/
.841,1.000/.869,1.120/.896,1.260/.916,1.380/
.938,1.540/.957,1.720/.973,1.930/.988,2.250/
1.000,3.000/
 3      FUNCTION   P1,D2
CAR2,10/TRUC3,18/
 4      FUNCTION   RN3,D2
.750,CAR2/1.000,TRUC3/
 1      FVARIABLE  16+4*FN2
*
*       THE VEHICLES ARRIVE AT THE STATION.
        GENERATE   V1
        ASSIGN     1,FN4
        TEST L     Q$PUMP2,2,ACT2
        TRANSFER   ,ACT3
*
*       THE VEHICLES LEAVE THE STATION.
 ACT2   TABULATE   P1
        TERMINATE
*
*       THE VEHICLES ARE SERVICED AT THE PUMP.
 ACT3   QUEUE      PUMP2
        SEIZE      PUMP2
        DEPART     PUMP2
        ADVANCE    FN3,FN1
        RELEASE    PUMP2
        TRANSFER   ,ACT2
*
*       TIMING LOOP
        GENERATE   960
        TERMINATE  1
        START      1
        END
```

Figure 3. The GPSS program produced by the system.

of the writer until he felt that he completely understood the problem and had all the information he needed to do the program. At this point he might state the problem "in his own words" to the writer as a check on his understanding of it. Finally, he would think about the problem in terms of the concepts of the computer language he planned to use, and then he would write the program.

The computer system being described here serves the same role as the simulation programmer described above. Therefore, it was designed to follow essentially the same overall procedure as he does, as can be seen from the example in Figures 1 through 3. The computer's counterpart of the programmer's mental image, the Internal Problem Description, is central to the operation of this system and will be discussed first, followed by discussions of the English input, the English output, and the GPSS program.

4. The Internal Problem Description

The Internal Problem Description (IPD) is an entity-attribute-value data structure for holding information about a particular problem in a language-independent form. Entity-attribute-value data structures have been widely used both in artificial intelligence applications and in simulation programming systems such as SIMSCRIPT and GPSS. In the IPD an entity is represented by a "record", which is just a list of attribute-value pairs. Some of the records in an IPD represent physical entities, such as a car or a dock, and others represent abstract entities, such as an action or a function. The attributes which a record has depend, of course, upon the entity being represented. The value of an attribute may be simply a number or a name, or it may be a pointer to another record.

A queuing problem typically deals with physical entities, such as cars or ships, moving through a system to be serviced in some manner at other physical entities, such as a pump or a dock, in the system. Here, the former of these are termed "mobile entities", and the latter are called "stationary entities". (In SIMSCRIPT these are temporary and permanent entities, and in GPSS they are transactions and facilities and storages.) As the mobile entities move through the system, they engage in "actions" at the stationary entities. Some of these actions are instantaneous, such as arrive and leave, and are called "events"; others, such as service and load, consume time and are referred to as "activities" here.

The IPD describes the flow of mobile entities through a system, by specifying the actions which take place there and their interrelationships. Each of these actions is represented by a record which has attributes to furnish such information as the type of action, the entity doing the action, the one to whom the action is being done, the location where it happens, how long it takes, how often it occurs, and what happens next.

A graphic portrayal of the IPD built by the system for the sample problem of Figure 1 appears in Figure 4. In the figure each record of the IPD is represented by a box, with the name of the record appearing at the top of the box. With the exception of MEMORY and 'ACTNLIST', these names do not actually exist within the computer, but were

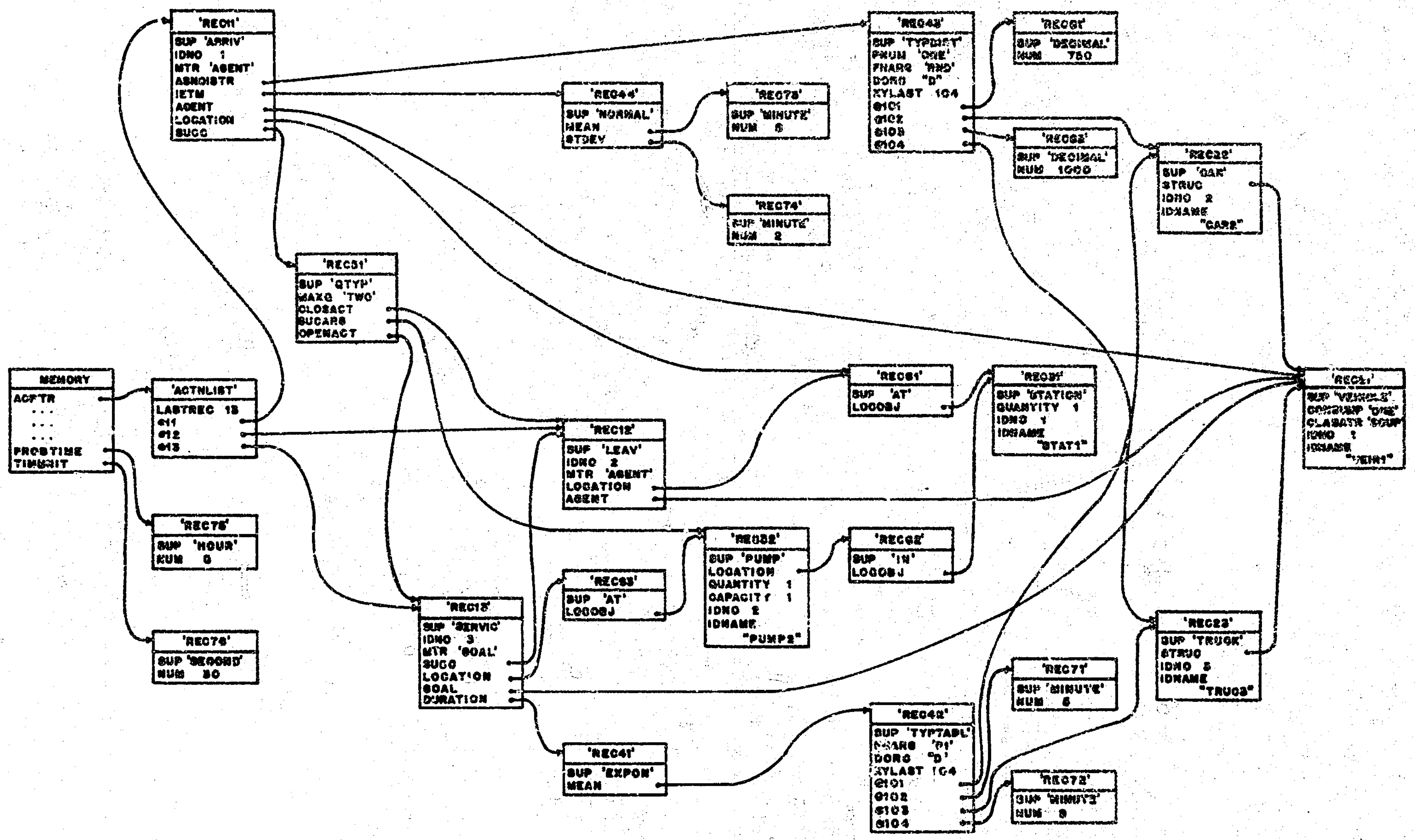

Figure 4. The Internal Problem Description (IPD)

placed on the drawing simply to furnish a means of referring to the various records in the discussion which follows. In each box the attribute-value pairs of the record are shown, with the attribute name or number on the left and its value on the right. Many of the values are pointers to other records in the IPD, in which case an appropriate arrow is drawn.

It can be seen that MEMORY is the only record which is not pointed to by some other record. It plays a rather central role in the IPD, being used both to hold global information about the problem (e.g. problem time and the basic time unit) and to serve as sort of a directory into the rest of the IPD. Only one portion of the "directory" was included in this drawing in order to keep the number of lines at a minimum. The portion included is the "action list" ('ACTNLIST'), which, as can be seen, contains pointers to each of the three action records. Not included in the drawing are the lists for mobile entities ('MOBLIST'), stationary entities ('STALIST'), distributions ('DSTRLIST'), and successor descriptors ('SCSRLIST'). The action list may be considered to be the most important list, because of the key role which actions play in a problem description.

Every IPD record, except for the MEMORY record and the lists just mentioned, has a SUPerset attribute pointing to the "named record" representing the concept of which this record is a specific instance. For example, the SUP attribute of the first action record (REC11) points to the named record 'ARRIV', indicating that this action (vehicles arrive at a station) is a specific instance of the concept "arriv". Named records contain information about the words and concepts for queuing problems and are entered into the system at initialization time.

Each action record in the IPD must have either an AGENT or a GOAL which points to a mobile entity record. The AGENT of an action is the one doing the action, and the GOAL is the one to whom the action is being done. The MTR attribute tells which of these two is pointing to a mobile entity. Each action record must also have a LOCATION attribute pointing to a "location descriptor" record, which in turn points to a stationary entity record. An event like 'ARRIV' or 'ENTER' must have an IETM (inter-event time) attribute to specify the time between occurrences of the event, and an activity (e.g. 'SERVIC' or 'LOAD') must have a DURATION attribute to specify the time taken to perform the activity. These times can be constants, standard probability distributions, functions, or combinations of these, some of which can be seen in the drawing. REC42 in the drawing is a function which has the records for car and truck as its X values and the records for 5 minutes and 9 minutes as its Y values. The ASNDISTR attribute of an 'ARRIV' specifies the percentages of the various kinds of entities which arrive, in the form of a cumulative probability distribution. REC43 in the drawing furnishes an example of this. (The NUM attribute of a 'DECIMAL' record is considered to be in parts-per-thousand.) The attributes DORC, FNARG, and PNUM which appear in REC42 and REC43 are needed for encoding the GPSS program.

Each action record, except a 'LEAV', must have a SUCCessor attribute to specify which action the mobile entity of this action is involved in next. The value of SUCC may simply be a pointer to another action record, or it may be a pointer to a "successor descriptor" record. REC51 in the drawing is an example of one of the five types of successor descriptors currently available in the system. This particular record, which is a 'QTYP' can be interpreted as saying, "If the length of the line at the pump (SUCARG) is less than two (MAXQ), go to be serviced (OPENACT); otherwise, leave (CLOSACT)." The other types of successor descriptors available handle such situations as "If the pump is busy, the vehicle leaves.","Cars are serviced, and trucks leave.", and "Half of the vehicles are serviced, and the rest leave."

It can be seen in the drawing that the records for 'CAR' and 'TRUCK' each have a STRUCture attribute pointing to the record for 'VEHICLE'. This is related to the idea of the "assignment distribution" (ASNDISTR), and essentially means that cars and trucks may be referred to as vehicles in the problem description. The attribute CLASATR (class attribute) in the 'VEHICLE' record indicates what is the distinguishing attribute of any records which have a STRUC attribute pointing to this record. (SOUP is synonymous with SUP in this case.) Part of the usefulness of the STRUC attribute is that it avoids some unnecessary duplication of information. For example, the value of the CONSUMPtion attribute is the same for both cars and trucks in this problem, so it need be stored only once, up in the 'VEHICLE' record. (CONSUMP indicates how many units of a resource are required by a mobile entity.)

Each entity and action record is assigned an identification number (IDNO) for use in the GPSS program. The value of IDNAME is formed by concatenating the first three or four letters of the NAME of the SUP of a record with the value of its IDNO. CAPACITY, QUANTITY, and CONSUMP are given default values of 1 or 'ONE' if they are not specified in the original problem description. ('ONE' is a named record, with a SUP of 'UNIT' and a NUM of 1.)

5. The English Statement of the Problem

At the beginning of a terminal session there is no IPD. It is the English input that furnishes the information which enables the system to build one. For example, when sentence 1 was processed, REC21 and REC31 were created and their SUP and IDNO attributes were given values. Also, REC61 was created, with values for its SUP and LOCOBJ attributes, and REC11 was created, with values for its SUP, IDNO, AGENT, and LOCATION. When sentence 3 was processed, REC12, REC63, and REC51 were created with some attributes, and the SUCC attribute was added to REC11.

Sentences which occur in natural language descriptions of queuing problems can be considered to fall into two categories: "action sentences" and "attribute sentences". An action sentence has as its main verb an action verb, which is modified by phrases and clauses to specify the values of the attributes of the action. For example,

> After arriving, if the dock is available, the ship is unloaded at the dock.

is an action sentence; the action is "unload", its goal is "ship", its location is "dock", its predecessor is "arrive", and its condition is "dock available". It should be noted that the order of most of the phrases and clauses in this sentence could be changed without altering the information content.

An attribute sentence has as its main verb an attribute verb, and is used to specify the value of some attribute of some record in the IPD. For example,

> The time to unload the ship is 8 hours.

says that the value of the "time" (actually duration) attribute of the action record "unload ship" is "8 hours". An equivalent statement would be

> It takes 8 hours to unload the ship.

The English statement of the problem must describe the flow of mobile entities through the queuing system. This is done by saying something about each action that takes place there, and how it is related to other actions. Each mobile entity must "arrive" at or "enter" the system. Then it may go through one or more other actions, such as "service," "load," "unload," and "wait." Then, typically, it "leaves" the system. The order in which these actions take place must be made explicit by the use of subordinate clauses beginning with such conjunctions as "after," "when," and "before," or by using the adverb "then". If the order of the actions depends on the state of the system being simulated, an "if" clause may be used to specify the condition for performing an action. Then a sentence with an "otherwise" in it would be used to give an alternate action to be performed when the condition is not met.

In the sample problem in Figure 1, sentences 1, 3, and 4 are action sentences describing the flow of vehicles through the station. This same information could be given in a wide variety of ways. For example, the following would be acceptable as input:

> Arrivals of vehicles occur at a station. If the length of the line at the pump in the station is less than 2 when a vehicle arrives, it will be serviced at the pump. Otherwise, it will leave immediately. After being serviced, a vehicle leaves the station.

In addition to describing the flow of mobile entities through the queuing system, the English statement of the problem must also furnish other information needed to simulate the system, such as the various times involved. It is necessary to specify the time between arrivals, the time required to perform each activity, the length of the simulation run, and the basic time unit to be used in the GPSS program. Also, the quantity of each stationary entity should be specified. (A quantity of one is assumed otherwise.) Other information, such as that of sentence 6 in the example, may also be given.

In the sample problem, sentences 2, 5, 6, 8 and 9 are attribute sentences furnishing this additional information. Just as with the action sentences, this information could be given in a wide variety of ways. For example, sentence 2, which specifies the values of both the location and quantity attributes of the pump, could be stated in at least the following three ways:

> There is one pump in the station.
>
> The quantity of pumps in the station is one.
>
> There is one pump, and the pump is in the station.

Each attribute sentence is essentially of the form, "<u>attribute</u> of <u>entity</u> equals <u>value</u>." The name of the <u>attribute</u> may be given explicitly, as is "quantity" in the second sentence above, or it may be implied by the verb or the type of value or some other characteristic of the sentence. For example, the verb "hold" implies the capacity attribute in the following sentence:

> The station can hold three cars.

An equivalent statement would be:

> The capacity of the station is 3 cars.

In each of the following, the attribute is implied by the type of value:

> The pump is in the station.
>
> The pump is green.

These are equivalent to:

> The pump is located in the station.
>
> The color of the pump is green.

In sentence 8 of the sample problem, the attribute "inter-event time" is implied.

The <u>entity</u> referred to may be a physical entity, such as pump and station in the above examples, or it may be an abstract entity, such as an action or a probability distribution. When it is an action, the infinitive or present participle form of the action verb, along with appropriate modifiers, is usually used to identify the action. For example, the following four sentences are all equivalent:

> The time <u>to service a vehicle at the pump</u> is 5 minutes.
>
> The time for <u>servicing a vehicle at the pump</u> is 5 minutes.
>
> The time for <u>a vehicle's servicing at the pump</u> is 5 minutes.
>
> <u>The servicing of a vehicle at the pump</u> takes 5 minutes.

Any one of the first three underlined phrases could have appeared in the fourth sentence of this example too.

The <u>value</u> part of an attribute sentence can take many different forms, as can be seen in the sample problem and in the above examples. Especially important in simulation models are quantitative values, of which there are several forms. The following phrases are examples of quantitative values:

> ten tons
>
> from 10 to 20 minutes
>
> 9 minutes for trucks and 5 minutes for cars
>
> exponentially distributed with a mean of two hours

In the last phrase above there is actually a second level of attribute and value, i.e. the mean (attribute) of the exponential distribution (entity) is two hours (value). This can also be seen in sentences 5 and 8 of the sample problem.

Whenever some action is referenced in either an action sentence or an attribute sentence, only enough modifiers to distinguish that action from others have to be used. For example, sentence 8 of the sample problem could have begun, "Arrivals of vehicles" or "Arrivals of vehicles at the station." However, it was sufficient to simply say, "Arrivals," because only one "arrive" action had been previously mentioned in the problem description.

Some use of pronominal reference is allowed, also. For instance, "it" is considered to refer to the most recent non-person mobile entity or stationary entity mentioned, whichever would make a meaningful sentence in the queuing problem context. Similarly, "there" is considered to be a substitute for the most recent location phrase. Both of these can be seen in sentence 4 of the sample problem, where "it" refers to the "vehicle" and "there" means "at the pump." The following sentence would have exactly the same meaning as sentence 4:

> Otherwise, it is serviced at it;
> then it leaves.

In this case the middle "it" would be taken as "the pump," because a location phrase requires a <u>stationary</u> entity in the queuing problem context.

Most sentences of the input text are completely parsed (at least implicitly), i.e. every word and phrase must be accounted for. However, the system is also capable of extracting meaning from some sentences just by the appearance of certain "keywords." For instance, if the words "time" and "unit" and some time phrase (e.g. "30 seconds") appear in a sentence, the time phrase is considered to be the value of the TIMUNIT attribute of MEMORY. Similarly, the appearance of "GPSS" or "program" results in the GPSS program being produced. In the sample problem, sentences 7, 9, 11, and 12 are keyword sentences.

In the English input the user may either state the complete problem immediately, or he may state just some part of it and then let the system ask questions to obtain the rest of the information, as was done in Figure 1. Each time the system asks a question, it is trying to obtain the value of some one essential attribute. A question may be answered by a complete sentence (e.g. statement 8) or simply by an appropriate phrase (e.g. statement 10) to furnish a value for the attribute, or the question may be ignored and a sentence with some other information given (e.g. statement 9).

6. <u>The English Problem Description Produced by the System</u>

The overall manner in which the English problem description is produced by the system can be seen by comparing the information in the text of Figure 2 with the information in the IPD of Figure 4. The first paragraph is produced by going down the action list and saying something about the attributes of each action. The very first action is simply stated with an action sentence containing information about the type of action, its AGENT and/or GOAL, and its LOCATION. If the IETM or DURATION attribute has a simple value, it will be included also, as a prepositional phrase (e.g. "every 8 minutes" or "for 5 minutes"). Otherwise, a separate statement in the form of an attribute sentence will be made about the IETM or DURATION, as can be seen in the figure. If the action has an ASNDISTR, a statement will then be made about it, as also can be seen in the figure. Finally, a statement of the form "After ..., ..." is produced from the SUCC attribute. The exact form of this statement depends upon the type of value which SUCC has. It can be seen in the figure that a 'QTYP' successor descriptor actually results in two sentences, with the first one having an "if" clause and the second one beginning with "otherwise".

When describing an action which has already been mentioned in a successor statement, it is not necessary to produce a simple sentence about that action. If the action has a non-simple DURATION and/or a SUCC, the appropriate statements about these can immediately be made. This is the case for the 'SERVIC' action in the example. No output was produced from the 'LEAV' action, because it had already been mentioned in a successor statement and it had no additional attributes to be described.

If a stationary entity has a QUANTITY or CAPACITY attribute with a value greater than 1, a statement will be made about it shortly after the entity is first mentioned in an action sentence (e.g. "There are 2 pumps in the station." or "The capacity of the station is 8 vehicles."). After describing the actions and the entities, a separate one-sentence paragraph is produced with the values of PROBTIME and TIMUNIT of MEMORY, as can be seen in the figure.

7. The GPSS Program Produced by the System

The manner of producing the GPSS program is similar to that for the English problem description, but it involves going down several lists, not just the action list. As was mentioned earlier, these other lists are not shown in the IPD drawing in Figure 4. Their contents will be given in parentheses at appropriate points in the following discussion, however.

The first bit of GPSS program produced is a standard SIMULATE card and RMULT card. Then a pass is made down the stationary entity list (REC31, REC32) to produce an EQU card for each stationary entity, to relate its IDNAME and its IDNO and to define it as a facility or a storage and a queue. If either the QUANTITY or CAPACITY attribute is greater than 1, an appropriate STORAGE definition card is also produced. Then a similar pass is made down the mobile entity list (REC21, REC22, REC23) to output an EQU card and a TABLE card for each type of mobile entity that will actually appear in the simulation (i.e. those records that do not have a CLASATR attribute). In the example, nothing is included for 'VEHICLE' because any vehicle that appears is either a car or a truck. The tables defined will be used to record transit times during the simulation.

Next, a standard FUNCTION 1 for the exponential distribution and a standard FUNCTION 2 for the unit normal distribution are produced if they are required by the problem. Then a pass is made down the distribution list (REC41, REC42, REC43, REC44) to define a FUNCTION for each record that requires one. In the example, FUNCTION 3 comes from REC42, and FUNCTION 4 comes from REC43. This is followed by a similar pass down the successor descriptor list (REC51) to define a FUNCTION for each record that requires one. This pass produced nothing in the example.

Then the records in the distribution list are looked at once again to define an FVARIABLE for each normal distribution used in the problem. One of these appears in the example. The numbers 16 and 4 appear there for the mean and standard deviation rather than 8 and 2, as might be expected, because the basic time unit to be used for this problem was specified as 30 seconds rather than 1 minute. The number of each FUNCTION and FVARIABLE defined in the above passes is stored as the IDNO attribute of the record which caused the definition, for use in later processing.

After the definitions have been taken care of, a pass is made down the action list to produce the executable blocks which describe the flow of transactions through the program (which corresponds to the flow of mobile entities through the actual system). For each action a blank comment card (with an asterisk in column 1), followed by a comment card with a simple action sentence on it is immediately put out. This is then followed by the blocks appropriate to this action.

The group of blocks produced from an action actually has two parts, the first of which depends upon the type of action and the second of which depends upon the type of value the SUCC attribute has. For example, an 'ARRIV' usually produces a GENERATE and an ASSIGN, a 'LEAV' produces a TABULATE and a TERMINATE, and most activities produce a sequence like QUEUE, SEIZE, DEPART, ADVANCE, and RELEASE, or minor variations thereof. A 'QTYP' successor descriptor results in a TEST, followed by a TRANSFER (if necessary), and a simple SUCC results in an unconditional TRANSFER, as can be seen in the example. If the 'LEAV' and 'SERVIC' actions had been in reverse order in the action list, the resulting GPSS program would not have needed the two unconditional TRANSFER's which appear in this program, and they would have been suppressed.

The contents of most of the argument fields of the various blocks depend, of course, upon the attributes of the records in the IPD. For example, argument A of the GENERATE block is V1 here because FVARIABLE 1 corresponds to the normal distribution which is the value of the IETM attribute of the 'ARRIV' action. Similarly, argument B of the ASSIGN block (which assigns the transaction type, either 2 or 3, to parameter 1 of the transaction) comes from the ASNDISTR attribute of the same action. Arguments A, B, and C of the TEST block and argument B of the TRANSFER come directly from the attributes SUCARG, MAXQ, CLOSACT, and OPENACT of the 'QTYP' record. The LOCATION attribute determines the A argument for such blocks as QUEUE, DEPART, SEIZE, and RELEASE, as can be seen in the example.

It can also be seen that argument A of the ADVANCE block (the mean advance time) references FUNCTION 3, which was defined from the 'TYPTABL' record which specifies the mean of the DURATION of the 'SERVIC' action. When a transaction enters that ADVANCE block, the appropriate mean time will be obtained from FUNCTION 3 using the value of parameter 1 which was ASSIGN'ed to it when it "arrived". This will then be modified by a value from FUNCTION 1 to yield a service time from the desired exponential distribution. The B argument of the last TRANSFER gets its value directly from the SUCC attribute of the 'SERVIC' action. All actions are referenced by names of the form "ACTi", where i is the value of the action's IDNO attribute.

Finally, after the blocks for the actions are put out, a standard "timing loop" is produced to govern the run length of the simulation. The value in the A argument of the GENERATE block comes from PROBTIME of MEMORY. In the example this value is 960, because there are 960 30-second periods in 8 hours.

8. The System

The computer system developed for this project is in the form of a 5000-statement FORTRAN program called NLP (Natural Language Processor), which is intended to be useful for a wide range of natural-language, man-machine communication tasks. When run under the CP/CMS time-sharing system on an IBM 360/67, it requires a virtual machine with 350K bytes of storage. The program consists of about 100 routines, ranging in size from one which simply unpacks a four-byte word to another which is a compiler for a grammar-rule language. One large group of routines provides list-processing capabilities. The main routine serves as a monitor to provide for interaction with the user.

Before NLP can process a queuing problem, it must be initialized with information about the relevant words and concepts and about the grammars of the languages to be used (currently English and GPSS) and how text is to be processed for these languages. Information about words and concepts is entered by means of "named record" definitions, and the grammars and processing are specified by "decoding rules" and "encoding rules". Each of these is discussed in detail in Reference 8.

9. Computer Time

There are at least three different kinds of time which can be reported for a job run on a time sharing system. The "virtual CPU time" does not include system overhead and is essentially the time that the job would take if run under a batch system. The "total CPU time" includes system overhead, most of which is for paging, and depends somewhat on the current load on the system. "Elapsed time" is the time that the user spends sitting at the terminal and can be very highly dependent on the current load.

For the sample problem the virtual CPU time was 77 seconds for Figure 1, 26 seconds for Figure 2, and 45 seconds for Figure 3, for a total of 148 seconds. The total CPU time was 156 seconds for Figure 1, 41 seconds for Figure 2, and 65 seconds for Figure 3, for a total of 262 seconds. The elapsed time for this problem may vary from one half hour to two hours, depending on the load on the system. Due to improvements in the program, the times given here average about 35 percent less than those given in Reference 7.

10. Conclusion

The system described here is considered to be in its early stages of development. It is already quite capable, as can be seen from the sample problem, but it certainly is not yet ready for production use and may not be for at least a few years.

In line with the overall goal stated in the Introduction, work is currently being done on developing the capability for having the system perform the simulation, rather than just producing a GPSS program. This will make it possible to give the user more control over the actual simulation and also make it possible to report the results in the language of the problem. It is intended that facilities for aiding statistical analyses will be incorporated, too. Along with this, work is continually being done to expand both the kinds of problems that the system can handle and the language which it will accept.

References

1. Markowitz, H. M., Hausner, B., and Karr, H.W. SIMSCRIPT A Simulation Programming Language. Prentice-Hall, Englewood Cliffs, N.J., 1963.

2. International Business Machines. General Purpose Simulation System/360 - Introductory User's Manual. Publication H20-0304, Data Processing Division, White Plains, N.Y., 1967.

3. Minsky, M. L. (Ed.), Semantic Information Processing. The M.I.T. Press, Cambridge, Mass., 1968.

4. Simmons, R. F. Natural language question-answering systems: 1969. Comm. ACM 13, 1 (January 1970), 15-30.

5. Lamb, S. M. Outline of Stratificational Grammar, Georgetown University Press, Washington, D. C., 1966.

6. Lockwood, D. G. Introduction to Stratificational Linguistics, Harcourt Brace Jovanovich, Inc., New York, 1972.

7. Heidorn, G. E. Natural Language inputs to a simulation programming system - an introduction. Technical report NPS-55HD71121A, Naval Postgraduate School, Monterey, Calif., December 1971.

8. Heidorn, G. E. Natural language inputs to a simulation programming system. Technical report NPS-55HD72101A, Naval Postgraduate School, Monterey, Calif., October 1972.

Acknowledgements

The author would like to express his appreciation to Richard C. Hansen, Robert T. McGee, Eldon S. Baker, Frederick H. Hemphill, Robert J. Williams, Alfred H. Mossler, and John H. Rickelman, former students at the Naval Postgraduate School, for their contributions to this research. This work was partially supported by the Information Systems Program of the Office of Naval Research as Project NR 049-314, and the facilities of the W. R. Church Computer Center were utilized.

Keywords

simulation programming, GPSS, natural language, artificial intelligence