CHAPTER 10.

THE UNIVAC 1107 SIMULA.

10.1 The Language.

The UNIVAC 1107 SIMULA is, excepting a few restrictions, implemented as an extension of the UNIVAC 1107 ALGOL. In particular the input/output and backing store facilities of the latter are available. For information concerning these topics the reader is referred to the "Programmers Guide" for the UNIVAC 1107 ALGOL. This document also provides the transliteration rules from the ALGOL 60 reference language and other necessary information.

The SIMULA basic symbols (see section 9.1) are implemented as reserved identifiers not usable for other purposes within a SIMULA block. The SIMULA library procedures for random drawing and data analysis (CH.'s 7, 8) are considered declared in a block outside the program, whereas the other SIMULA procedures are treated as if declared in a dummy block immediately outside the SIMULA block. All procedure identifiers are therefore, by suitable redeclaration, usable for any purpose that the programmer may choose.

As in the UNIVAC 1107 ALGOL forward references to variables and other items (except local labels) must be resolved by corresponding "LOCAL" declarations in the relevant block heads.

10.2 Restrictions.

At the time of writing the following concepts are not implemented:

a. The own concept,
b. the call by value of a string parameter, and
c. STRING ARRAY in the sense of UNIVAC 1107 ALGOL.

Additional restrictions are:

1. A connection block may only refer to an activity already declared, or to the present one. (Forward references to activities by process designators are, however, resolved by a "LOCAL ACTIVITY" declaration in the SIMULA block head.)

2. Termination, passivation, or suspension of the currently active process is not in general tolerated during the evaluation of an expression, i.e. inside a function procedure. This is the case if the expression is part of

   a. an actual parameter,
   b. a subscript bound of an array declaration, or
   c. an element of a switch.

   Suspension as the result of the direct scheduling of another event is permitted inside a function procedure, if the calling process is not terminated, cancelled, or reactivated before control returns.

10.3 Storage Requirement.

The data storage economy of UNIVAC 1107 SIMULA programs is far from the optimum. This is partly due to the way in which certain basic access mechanisms are implemented in the UNIVAC 1107 ALGOL system.

In order to achieve maximum storage economy the following rules should be observed within activities corresponding to a large number of processes.

1. Inactive periods within sub-blocks and procedures should be avoided.

2. The amount of data local to the outermost block, and the number of non-local variables referenced in that block, should be minimized.

3. The number of for statements and calls for activities and user defined procedures should be kept a minimum.

See also the next section.

10.4 Data De-Allocation.

The mechanism for de-allocating data at run time is fully automatic. However, an understanding of the principles involved will enable the programmer to have better control over the storage requirements of his program, if economy is required.

The main rules are the following.

1. The data structure local to a block, except the outermost block of a process, will remain only as long as the "local" sequence control (i.e. the main control or the reactivation point of the process in question) is within the block. The data may include simple variables, value parameters, and names on arrays and sets.

2. A process (including data local to a sub-block) remains only as long as there is at least one element referencing it.

3. An element remains only as long as there is at least one reference to it, from
   a. an element variable or value parameter, or a component of an element array,
   b. an element (through set membership),
   c. an event notice in the SQS, or
   d. a connection block.

4. An array remains only as long as it has a name. The name can be the declared array identifier or an exogenous array attribute of a process.

5. The elements of a set retain their set membership only as long as the set has a name. The name can be a simple declared set designator, a value set parameter, or a component of a set array. (The name need not be effectively referenceable, see below.)

6. An event notice can be removed from the SQS as the result of a sequencing statement, or when the (local) sequence control leaves an active process.

The rules 2-5 are put to effect by maintaining reference counts on processes and elements. A set head has a separate reference count for set designators referencing the set. There are, however, two cases where the reference count technique is not sufficient.

1. The value of an element expression, as residing in an accumulator immediately after the evaluation, is not reflected in the reference count on the element. An exception from rule 3 has to be made to prevent the de-allocation of the element value at exit from an element procedure, if the reference count indicates no reference to this element. (This is the case e.g. with all generative expressions.) Normally the element gets a reference assigned to it as the result of the statement containing the expression, but if this does not happen, and there is no other reference to it, the system does not get another chance to perform the de-allocation.

2. When certain "cyclic" data configurations are present, it may happen that an element with a non-zero reference count is not and can never become effectively referenceable through a computable element expression. Such an element can safely be de-allocated, even though it is not "out of the system" in the reference count sense.

   As a simple example consider a process referencing itself through a local element variable X. Assume that there is no other reference to the element and no other element referencing the process. Evidently the only way of effectively referring to the element is to evaluate the expression X during an active phase of the process or in a connection block connecting it. But the process is passive or terminated, and not connected, because otherwise there would have been another reference to it, via the same or another element, from an event notice in the SQS or from a connection block, contrary to assumptions.

For these reasons a second de-allocation mechanism, the "storage clean-up" is brought to use whenever the available store is exhausted. The storage clean-up will locate all effectively referenceable parts of the system and make the remaining store available for use, if any.

Warning.

Since the storage clean-up may be brought in at unpredictable points of time, it should have no visible effect in the program. In particular every effectively referenceable element will retain its set membership. It follows that a set which can not be broken up by the reference count technique, e.g. a set local to a process referencing itself, will remain in the system as long as it contains at least one effectively referenceable element, even if it is not itself effectively referenceable as a set.

10.5 Operating Instructions

The UNIVAC 1107 SIMULA implementation consists of two distinct parts.

1. The compiler translates a SIMULA source language program to a program in object code. The compiler is a permanent extension of the UNIVAC 1107 ALGOL compiler.

2. The run time system is the collection of subroutines that may be referenced by a SIMULA object program. The run time system includes all SIMULA library procedures.

In the following a certain elementary acquaintance with the EXEC II monitor system is assumed. The reader is referred to the "UNIVAC 1107 EXEC II Manual" for details. The description is valid for an operating system where the SIMULA compiler resides on drum and the SIMULA library subroutines are on a tape called "SIMLIB". The appropriate operating procedure for the particular computing center should always be obtained before compilation and execution of a SIMULA program is attempted.

A SIM control card brings in the ALGOL compiler with the SIMULA extensions. The word "SIMULA" has been added to the compiler's list of reserved identifiers, so that SIMULA blocks can be recognized. Also the names of the random drawing and data analysis procedures of SIMULA have been added to the list of standard procedure identifiers of ALGOL. Within a SIMULA block all SIMULA concepts become available.

All options valid for an ALG control card are available and retain their usual meaning.

The SIMULA run time system resides on a magnetic tape. It must be part of the user's program complex file (PCF) on drum at the time of allocation of the absolute object program.

Some of the SIMULA routines will replace corresponding routines in the ALGOL run time system, which is a permanent part of the monitor library on drum.

Any ALGOL object program, subject to the restrictions mentioned in section 10.2, may be run with the SIMULA run time system. The execution will be slightly slower, but the available data store is increased, since the SIMULA storage allocation routines can utilize both memory banks.

The card deck of a simulation run could have the following typical layout:

```
∇ RUN ----
∇ ASF F = SIMLIB
∇ N   XQT CUR
  IN  F
  TRI F
∇     SIM PROG
(Source language program)
∇ XQT PROG
(Data cards)
∇ FIN
```

"PROG" can be any name chosen for the program. "SIMLIB" is the operational label adopted for the magnetic tape containing the SIMULA run time system. The CUR operations serve to include the latter in the user's PCF.

10.6 Initial values

Upon entry into a block variables (simple or subscripted) declared local to the block have the following initial values: zero, false, none, and blanks for arithmetic, Boolean, element, and string variables, respectively. Sets are empty.