STORAGE ADMINISTRATION
IN A VIRTUAL MEMORY SIMULA SYSTEM

STEFAN ARNBORG

Abstract.

The run time storage administration of a planned Simula system for PDP-10 is described with emphasis on record formats, record relationships and optimizations for the virtual memory organization. A parametrized garbage collector for variable size records in virtual memory is outlined.

1. Introduction.

This paper aims at giving the reader a general understanding of some problems encountered during the design of a Simula 67 system for the PDP-10 computer, and the methods applied to solve them. Although the implementation has not yet started, the design phase has proceeded enough so that a description of the system can be given. The reader is assumed to have a working knowledge of the Simula 67 programming language and Algol 60 implementation techniques.

2. System characteristics.

The system is intended for a hardware paged processor. Although the software necessary to implement a virtual memory computing system is not yet available, such a system is expected to be supplied in the immediate future, and this has been taken into account during the design.

The system is expected to be used both as a general purpose algorithmic language and for medium and large scale list processing and simulation problems. This implies that the system must be very flexible and spend resources with care. Particularly simulation programs have a tendency to grow to the limits of system capacity, so there is a definite need not to waste storage.

3. Storage control in Algol and Simula run time systems.

Algol 60 has been designed to permit storage allocation and deallocation on a stack basis, i.e. when a storage block has been allocated it is

Received December 13, 1971.
not deallocated until all other storage blocks allocated in the meantime have been deallocated, and no storage is wasted in this way. Although Simula is based on Algol 60, storage allocation and deallocation does not follow a single stack discipline. Instead there are several separate stack order allocation demands, each corresponding to a component (a Simula construct with properties similar to a coroutine) in the quasi-parallel system constituting a Simula program. In addition, storage for class instances, text blocks and arrays must be allocated without any explicit deallocation order.

These characteristics, together with the requirement for storage conservation, necessitate an elaborate storage management system using the garbage collection technique. Since the dynamic storage requirements cannot be predicted in any way, and because of the risk of fragmenting memory, the garbage collection has been combined with a storage-compacting scheme.

All dynamic storage in the run time system is allocated and deallocated in units of variable size records, and these must be designed so that they can be moved in memory by the compacting garbage collector without affecting the proper execution of the Simula program.

4. Block records and display vectors.

The Simula compiler will allocate the data quantities declared in a block at fixed offsets within a block record. The block record is allocated dynamically at run time when the block is entered. Since the data quantities of the currently innermost block and its enclosing blocks are accessible by name in the statements of the block, the compiled code must have fast access to these blocks, commonly known as the static environment of the current block.

There are several ways to solve the problem of referencing the quantities in the static environment. Many methods used rely on the fact that the static environment of a block (except for a connected class instance in Simula, see below) is fixed over the life span of the block, and that the function (i.e. many-to-one) relation “is immediately enclosed by” on the blocks defines a tree with the blocks as nodes (the blocks and the relation defines what is known in mathematics as an inverse arborescence, but the term tree is often used for this as well). With the aid of a pointer from each block to its immediately enclosing block, any block in the static environment can be found by following these links from the current block (fig. 1).

The static environment of a block is conceptually a vector of block