9. Sequencing

9.1 Blocks and dynamic scopes

The constituent parts of a program execution are dynamic instances of blocks. The different kinds of blocks fall into three categories according to the possible interrelationships and states of execution of the block instances.

1) Prefixed blocks.
2) Sub-blocks, procedure bodies and connection blocks.
3) Class bodies.

A block instance is, at any given time, in one of three states of execution: "attached", "detached" or "terminated". The possible and initial states are defined in fig. 9.1.:

Block category	Possible states	Initial state
1	D	D
2	A	A
3	A,D,T	A
A: attached	D: detached	T: terminated

Fig. 9.1 Execution states

A program conforming to ALGOL 60 only contains blocks of category 2, except the outermost one which is of category 1 (see section 11). An execution of any such program is a simple dynamically nested structure.

A block instance of category 2 is in the attached state and is said to be "attached to" the smallest dynamically enclosing block instance. E.g. an instance of a procedure body is attached to the block instance containing the corresponding procedure call.

The "program sequence control", PSC, refers at any time to that program point within a block instance which is currently being executed. For brevity we shall say that the PSC is "positioned" at the program point and is "contained" in the block instance. If A is the block instance containing the PSC, then A, and any block instance dynamically enclosing A, is said to be "operating".

The entry into any block invokes the generation of an instance of that block, whereupon the PSC enters the block instance. If and when the PSC leaves a block instance of category 1 or 2 through its end or by a go to statement, that block instance is deleted.

An object (i.e., a block instance of category 3) is initially attached to the block instance containing the corresponding object generator. It may enter the detached state by executing the statement "detach" (see section 9.2.1). If and when the PSC leaves the object through its end or by a go to statement, the object becomes terminated.

A block instance is said to be "local to" the one which contains its describing text. E.g. an object belonging to a given class is local to the block instance containing the class declaration.

A block instance A is said to "enclose" a second one B if:

1) B is attached and is dynamically enclosed by A, or

2) B is detached or terminated and is local to A or to a block instance dynamically enclosed by A, or

3) there is a detached object C local to A or to a block instance dynamically enclosed by A, such that C encloses B.

Whenever a block instance is deleted, any block instance enclosed by it is also deleted. It is a consequence of the language structure that an object, at the time of its deletion, cannot be referenced by any computable object reference expression. The dynamic scope of an object is thus limited by that of its class declaration. However, an implementation may further reduce the effective life spans of objects by techniques such as "garbage collection".

Notice that arrays and text objects cannot, in general, be deleted together with the block instance in which they are declared.

9.2 Quasi-parallel sequencing

A program execution can be described as a tree structure whose branching nodes are instances of prefixed blocks. A subtree whose "root" is a prefixed block instance is called a "quasi-parallel system". The prefixed block instance, including block instances dynamically enclosed by it, is called the "main program" of the quasi-parallel system.

A quasi-parallel system has an associated "system level", which is the number of prefixed block instances enclosing its main program. The program as a whole is a quasi-parallel system at system level zero.

A quasi-parallel system consists of system "components" which are the main program and any detached object, including block instances dynamically enclosed by the object, whose smallest enclosing instance of a prefixed

block is the main program. The components of a quasi-parallel system are said to be "detached" at the system level of the quasi-parallel system.

Any system component has an associated "local sequence control", LSC. Associated with any quasi-parallel system is an "outer sequence control", OSC. The OSC at system level zero coincides with the PSC. The OSC of a system at level k ($k \geq 1$) coincides with the LSC of that component at system level k-1 which encloses the given system.

For any given quasi-parallel system, one and only one of its components is said to be "active". The LSC of that component coincides with the OSC of the quasi-parallel system.

An instance of a prefixed block is initially active, i.e. it contains the OSC of its own quasi-parallel system. The OSC of a system may move from one component to another as the result of statements described below. The LSC of a component not containing the OSC remains positioned at the program point at which the OSC left the object the last time.

At any given time, there exists a sequence of system components $X_0, X_1, \ldots\ldots, X_n$ such that:

1) X_k is active at system level k ($k = 0,1,\ldots,n$).

2) X_k is enclosed by X_{k-1} ($k = 1,2,\ldots,n$).

3) There is no quasi-parallel system enclosed by X_n.

This sequence is called the "operating chain". System components on the operating chain all contain the PSC and are therefore said to be operating. The LSC of a system component remains fixed as long as it is not a member of the operating chain.

9.2.1 The detach statement

Let the smallest operating block instance be X.

If X is an attached object, a detach statement has the following effects:

1) The object becomes detached at the system level of the smallest enclosing prefixed block instance, its LSC positioned at the end of the statement.

2) The PSC returns to the block instance to which X was attached and resumes operations after the object generator which caused the generation of X. A reference to X is the result of that expression.

If X is a detached object which is component of a quasi-parallel system S, a detach statement operates as follows:

1) The OSC of S leaves X. As a consequence X is removed from the operating chain. Its LSC remains positioned at the end of the statement.

2) The OSC of S enters the main program of S at the current position of its LSC. As a consequence the main program of S and possibly system components at system levels higher than that of S become operating.

If X is an instance of a prefixed block, a detach statement has no effect.

If X is any block instance other than an object or a prefixed block instance, execution of a detach statement constitutes an error.

9.2.2 The resume statement

"resume" is formally a procedure with one object reference parameter qualified by a fictitious class including all classes.

Let the actual parameter of a resume statement reference a detached object Y, which is a component of a quasi-parallel system S. It is a consequence of the language conventions that Y can only be referenced from within a block instance which is or is enclosed by a component X of S. X is currently operating. The resume statement has the following effects:

1) The OSC of S leaves X. As a consequence X and any operating components at higher system levels are removed from the operating chain. The LSC of each component remains at the end of the resume statement.

2) The OSC of S enters Y at the current position of its LSC. As a consequence Y, and possibly a sequence of components at higher system levels, become operating.

If the actual parameter of a resume statement does not refer to a detached object, its execution constitutes an error.

9.2.3 Object "end"

The effect of the PSC passing through the final end of an object is the same as that of a detach statement, except that the object becomes terminated, not detached, and thus loses its LSC.

9.2.4 go to statements

A designational expression defines a block instance and a program point local to this block instance.

A go to statement leading to a block instance is valid if and only if this block instance is operating. This restriction implies that a go to statement leading out of a detached object must also lead out of the smallest enclosing prefixed block. The restriction further implies that a go to statement leading to a connected label is valid if and only if the connected object is also operating. The go to statement will lead to the label in the operating block instance.

Block instances left through a go to statement become terminated.

call see Implementation Guide; p. 94. (16.3)