4. Expressions

4.1 Syntax

<label> ::= <identifier>
<expression> ::= <value expression> |
<text value> |
<reference expression> |
<designational expression>
<value expression> ::= <arithmetic expression> |
<Boolean expression> |
<character expression>
<reference expression> ::= <object expression> |
<text expression>

4.1.2 Semantics

The syntax for label represents a restriction compared with ALGOL 60.

A value expression is a rule for obtaining a value.

A reference expression is a rule for obtaining a reference and the associated referenced value.

A designational expression is a rule for obtaining a reference to a program point.

Any value expression or reference expression has an associated type, which is textually defined. The type of an arithmetic expression is that of its value. The following deviations from ALGOL 60 are introduced; see also section 8.2.3.

1) An expression of the form

<factor>↑<primary>

is of type real.

2) A conditional arithmetic expression is of type integer if both alternatives are of type integer, otherwise its type is real. If necessary, a conversion of the value of the selected alternative is invoked.

4.2 Character expressions

4.2.1 Syntax

<simple character expression> ::= '<character designation>'
<variable>|
<function designator>|
(<character expression>)

<character expression>
::= <simple character expression>|
<if clause><simple character expression>
else <character expression>

4.2.2 Semantics

A character expression is of type character. It is a rule for obtaining a character value.
A character designation is either an external character or another implementation defined representation of an internal character.

4.3 Object expressions

4.3.1 Syntax

simple object expression ::= none|
<variable>|
<function designator>|
<object generator>|
<local object>|
<qualified object>|
(<object expression>)

```
<object expression> ::= <simple object expression>|
                        <if clause><simple object expression>
                            else <object expression>
<object generator> ::= new <class identifier>
                           <actual parameter part>
<local object> ::= this <class identifier>
<qualified object> ::= <simple object expression>
                           qua  <class identifier>
```

4.3.2 Semantics

An object expression is of type ref (<qualification>). It is a rule for obtaining a reference to an object. The value of the expression is the referenced object or none.

4.3.2.1 Qualification

The qualification of an object expression is defined by the following rules:

1) The expression none is qualified by a fictitious class which is inner to all declared classes.

2) A variable or function designator is qualified as stated in the declaration (or specification, see below) of the variable or array or procedure in question.

3) An object generator, local object, or qualified object is qualified by the class of the identifier following the symbol "new", "this", or "qua" respectively.

4) A conditional object expression is qualified by the innermost class which includes the qualifications of both alternatives. If there is no such class, the expression is illegal.

5) Any formal parameter of object reference type is qualified according to its specification regardless of the qualification of the corresponding actual parameter.

6) The qualification of a function designator whose procedure identifier is that of a virtual quantity, depends on the access level (see section 7). The qualification is that of the matching declaration, if any, occurring at the innermost prefix level equal or outer to the access level, or if no such match exists, it is that of the virtual specification.

4.3.2.2 Object generators

An object generator invokes the generation and execution of an object belonging to the identified class. The object is a new instance of the corresponding (concatenated) class body. The evaluation of an object generator consists of the following actions:

1) The object is generated and the actual parameters, if any, of the object generator are evaluated. The parameter values and/or references are transmitted. (For parameter transmission modes, see section 8).

2) Control enters the object through its initial begin, whereby it becomes operating in the "attached" state (see section 9). The evaluation of the object generator is completed:

   case a: whenever the basic procedure "detach" is executed "on behalf of" the generated object (see section 9.1), or

   case b: upon exit through the final end of the object.

The value of an object generator is the object generated as the result of its evaluation. The state of the object after the evaluation is either "detached" (case a) or "terminated" (case b).

4.3.2.3 Local objects

A local object "this C" is a meaningful expression within

1) the class body of C or that of any subclass of C, or

2) a connection block whose block qualification is C or a subclass of C (see section 7.2).

The value of a local object in a given context is the object which is, or is connected by, the smallest textually enclosing block instance, in which the local object is a meaningful expression. If there is no such block the local object is illegal (in the given context). For an instance of procedure or class body "textually enclosing" means containing its declaration.

4.3.2.4 Instantaneous qualification

Let X represent any simple reference expression, and let C and D be class identifiers such that D is the qualification of X. The qualified object "X qua C" is then a legal object expression, provided that C is outer to or equal to D or is a subclass of D. Otherwise, i.e. if C and D belong to disjoint prefix sequences, the qualified object is illegal.

If the value of X is none or is an object belonging to a class outer to C, the evaluation of X qua C constitutes a run time error. Otherwise, the value of X qua C is that of X. The use of instantaneous qualification enables one to restrict or extend the range of attributes of a concatenated class object accessible through inspection or remote accessing. (See also section 7.)

4.4 Text expressions

4.4.1 Syntax

```
<simple text expression> ::= notext |
                             <variable> |
                             <function designator> |
                             <text expression>
<text expression>::= <simple text expression> |
                     <if clause><simple text expression>
                         else <text expression>
<text value> ::= <text expression> |
                 <string>
```

4.4.2 Semantics

The constituents of a string are external characters and/or other implementation defined representations of internal characters.

A string is a text value, not a text reference. It is not a text expression, but it may occur as the right part of a text value assignment (cf. section 10.6), as an operand of a text value relation (cf. section 5.2), and as an actual parameter called by value (cf. section 8.2.1).

In an implementation the left and right string quotes may be represented by one and the same external character. In this document either symbol is represented by the symbol ".

notext designates an empty text reference.

For further information on the text concept, see section 10.