STICHTING

MATHEMATISCH CENTRUM

2e BOERHAAVESTRAAT 49

AMSTERDAM

REKENAFDELING

MR 76

Orthogonal design

and description of

a formal language

by

A. van Wijngaarden

Premature and preliminary

edition, intended for use

by IFIP WG 2.1 only

October 1965

Amsterdam, 4th April 1972

One of the basic documents in the history of ALGOL 68 is MR 76, Orthogonal design and description of a formal language. It was only made available to members of the IFIP WG 2.1 at its meeting in St. Pierre de Chartreuse in October 1965. It was never properly published by the Mathematisch Centrum, since its state was too premature and the following developments of ALGOL 68 made the paper rather obsolete. However, there have been so many requests in the course of time to get a copy that this reprint is made available. On purpose, I did not correct any of the many misprints and more serious errors in it, but just have added the coverletter of October 22 and the list of errata and amendments of October 24 which were available before the meeting started. Moreover, it has been slightly reduced in size in order to distinguish it from the original edition. Actually, one of the reasons why this reproduction came so late, was that I could not find a copy which was not covered with remarks and so on. Lately, however, I found a clean copy, suitable for reproduction.

A. van Wijngaarden

Amsterdam, October 22, 1965.
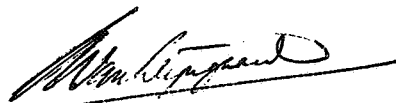
Dear WG 2.1 member or affiliate,

Herewith you receive a very premature copy of report MR 76 to serve as a
document during our meeting in Grenoble next week. I apologize for not
producing it earlier but only without a great effort I got it sufficiently
in shape so that it might fulfil its purpose at all. During July of this
year I had the privilege and pleasure to have Niklaus Wirth as guest at
the Mathematical Centre, who prepared here his document MR 75. I enjoyed
tremendously the discussions, that we had on the design of ALGOL X and
also accept several of his concepts which have been put into it. However,
we disagreed on some points and I promised to make a number of amendments
on these points. My main objections were certain to me unnecessary restrictions
and the definition of the syntax and semantics. Actually the syntax viewed
in MR 75 produces a large number of programs, to very few of which the
semantics attribute a meaning, whereas I should prefer to have the subset
of meaningful programs as large as possible, which requires a stricter
syntax. Of course, this has nothing to do with my appreciation for the
work of Niklaus.

When I started to phrase my thoughts immediately afterwards it soon came
out that some better tools than the Backus notation might be advantageous,
though I did not want to use my other symbolism that you know from the
report MR 74 by De Bakker. I developed a scheme which is explained in the
first section, which enables the design of a language to carry much more
information in the syntax than is normally carried. As to the design of a
language I should like to see the definition of a language as a Cartesian
product of its concepts. I tried to find which concepts were involved in
the work of Wirth and added a few ones. I made some models and the last
version of this was written down in a hurry. Since its scope is greater than
the design of ALGOL X, which is only used as an example, it is intended
after some amendments have been made and some missing chapters have been
added to become an official publication. Since it was written down in such
a hurry there was no time to check the sections completely against one
another, but the fact that e.g. the word compatible does not occur in the
text does not mean that this concept has not been fully incorporated by means
of the syntactical construction.

I had no time to generalize some notions, because also the declarations
obviously can be considered as expressions of a type, declarative say.
Moreover parallel elaboration of expressions has not been included, but I
hope to have the amendments for this, which will only cause a change here
and there, ready for the time of the meeting.

Looking forward to see you in Grenoble,

yours truly,

A. van Wijngaarden,
Mathematical Centre, Amsterdam.

A. VAN WIJNGAARDEN.   Errata and Amendments to MR 76.

1. Contents. 14: For "active expressions" read "active value expressions".
2. Basic constituents. Semantics. 3rd line from below: For "lexicographic" read "lexicographical".
3. Identifiers. Semantics. 5th line from below: For "en" read "or".
4. Values and value denotations. Syntax. 4th line from below: For "comma A" read "comma S".
5. Id. Semantics. 11th, 12th line from below: For "row value" read "row of some type value".
6. Blocks. Syntax. 5th line from above: For "VE" read "active VE".
7. ibid. 6th line from above: For ",semicolon S" read ":pointS; semicolon S".
8. Declarations and designators. Syntax. 2nd line from above: For "VMD" read "UMD".
9. ibid. 10th line from above: For "opening" read "closing" and for "closing" read "opening".
10. ibid. Semantics continued. 15th line from above: For "of lower" read "of the lower".
11. ibid. 15th and 16th line from below: For "an unspecified order" read "parallel".
12. Expression. Semantics. 1st and 2nd line from above: For "type" read "type and kind".
13. ibid. Add at bottom: "The elaboration of an expression which consists of a number of simple expressions separated by also symbols exists of the elaboration of those simple expressions in parallel, i.e. a sequence of units of action made up by an unspecified merging of the sequences of units of action which constitute the elaboration of the simple expressions themselves. The elaboration of an expression may, moreover, appoint a so-called successor of that expression. The elaboration of a jump appoints its successor explicitly (cf. active value expressions). The successor of a block or designator whose elaboration is terminated by the elaboration of a jump is the successor of that jump. The successor of anyother expression which is the choice of an expression option which is followed by a block separator containing a semicolon symbol is the expression which follows that block separator. All other expression have no successor".

14. Location expressions. Semantics: Delete the last three lines, i.e. "The result... following it".

15. Value expressions. Syntax: For "UXE." read "UVE, also S, simple UVE; UXE.".

16. ibid. 3rd line from below: For "The order in which the operands are evaluated is left undefined" read "The operands are elaborated in parallel".

    ibid. 1st line from below: For "evaluation" read "elaboration".

17. Three value expression. Semantics. 2nd line from above: For "Some unspecified order" read "parallel".

18. Active expressions. Title: For "Active expressions" read "Active value expressions".

    ibid. Syntax. 1st line from below: For "jumpS" read "goto S".

19. ibid. Semantics. Delete first three lines, i.e. "The elaboration...units".

    ibid. Delete lines 12 uptil 17, from above, i.e. "The successor of a block... has no successor".

    ibid. 2nd line from below. For "some order" read "parallel".

20. Declarations and designators. Syntax. 2nd line from above: For "UMED." read "UMD, also S, simple UMD.".

21. ibid. Between 2nd line and 3rd line from above: Insert "simple UMD: announcing UMD; UMED."

22. ibid. 3rd line from above: For "simple" read "announcing".

23. Declarations and designators. Semantics Continued. 5th line, 7th line, 11th line from above: For "simple" read "announcing".

24. ibid. Between 17th and 18th line from below: Insert "The elaboration of an declaration which consists of a number of simple declarations separated by also symbols exists of the elaboration of those simple declarations in parallel (cf. expressions).

25. Language and Metalanguage. Syntax. 2nd line from below: For "UMW" read "UM".

Contents

Language and Metalanguage.

Syntax.

A : complex ; real ; integral.

B : logical ; binal.

C : specification.

D : declaration.

E : expression.

F : string S, letter, digit, bit.

G : UMI, Q, E, VN.

H : UM ; UME ; UMW ; label.

I : identifier.

J : K ; JRK.

K : VN ; KVN.

L : location.

M : V ; L ; X.

N : denotation.

O : operator.

P : parameter.

Q : UME, YNP.

R : row.

S : symbol.

T : A ; B ; string ; tree ; active.

U : T ; TJR.

V : value.

W : YP ; YPW.

X : variable.

Y : UMW ; UMWI.

Z : actual, formal.

Semantics.

The strict language is a phrase structure language, defined by a formal system,
which uses the notation and the definitions explained below.

The structure of the language is determined by the three quantities:

T1, called the set of basic constituents of the strict language,

T2, called the set of syntactic entities,

T3, called the set of syntactic rules.

Moreover are defined

T4, the set of all possible sequences of members of T1, and

T5, the set of all possible sequences of members of T1 and T2.

Members of these sets will be denoted by t1, t2, t3, t4 and t5 respectively
and different members of one set by using apostrophes or subscripts.

Any t2 is denoted by a word or word sequence, i.e. a sequence of small metaletters,
possibly separated by metaspaces. Small metaletters are typographical symbols
used in this description of the language. Metaspaces are blanks between words.

Any t3 has the form

t2 : t5.

where t2 differs from t5.

Definition 1: t5' is said to be a direct production of t5 iff there exist
possibly empty sequences t5'' and t5''' and a t2 and a t3 such that t5 =
t5''t2t5'''', t5'=t5''t5'''t5'''', t3=t2 : t5'''.

Definition 2: t5' is said to be a production of t5 iff there exists a set
t5(0), t5(1),...,t5(n) such that t5=t5(0), t5(n)=t5' and t5(i) is a direct
production of t5(i-1) for all i=1,...,n.

Definition 3: A terminal production of t5 is a production of t5 which is a t4.

The elements t1, t2 and t3 are defined through enumeration in the sequel under
the heading Syntax. In order to attribute a meaning to syntactic entities,
explanations are given in the sequel under the heading Semantics. When words
or sequences of words are used under Semantics that are defined under Syntax,
then they refer to this definition. It is recognized that the structure of
the English language may cause deviations from the words used under Syntax and
understood that partly capitalized forms, plural forms, split forms and
generalising forms will be properly interpreted.

The formal system itself is a phrase structure language, called metalanguage,
which can be defined analogously. Its basic constituents are the names of the
syntactic entities of the language and their constituting words. Its syntactic
entities are denoted by a capital metaletter. A set of syntactic rules is
defined by the rules given above under Syntax. These are to be interpreted
by means of the following proces.

Process 1. Each rule containing a '' ; '' is replaced by two new rules. The
first new rule is the part of the rule before that '' ; '' followed by a '' . ''
and the second new rule is the part up to and including the '' : '' followed
by the part of the rule after that '' ; ''. This action is repeated until each
'' ; '' has been eliminated.

The capital metaletters used in the sequel under Syntax have no relation with
the capital letters occurring under Semantics, but refer to the syntactic
entities of the metalanguage. The rules occurring in the sequel under Syntax
stand for one or more syntactic rules of the strict language. They are to be
interpreted in the following steps.

Step 1. The process referred to above as Process 1 is performed.

Step 2. If a rule contains a capital metaletter then this is replaced at each occurrence of it in the rule by the same arbitrary terminal production of it.

Step 2 is repeated until each capital metaletter has been eliminated.

Some syntactic entities of the metalanguage produce only one word and serve only to shorten the text. Others produce more words and give rise to a corresponding number of syntactic rules of the language. This number may be infinite.

The definition given sofar defines the strict language. The language is a phrase structure language which is defined as follows. Any terminal production of a specific entity of the strict language, called "program" may be subjected to a number of notational changes which are defined in the section on conventions. Any text obtained in that way is a production of that same entity "program" when viewed as a syntactic entity of the language.

Basic constituents.

Syntax.

basic constituent : basic S ; string.
S : basic S ; non basic S.
basic S : string quote S ; non basic S.
string S : basic string S ; non basic S.
basic string S : letter ; value S ; delimiter.
value S : digit ; logical VN ; bit ; active VN.

Semantics.

The existence outside the realm of the language of a set of information carrying
entities, called characters, is recognized. Certain characters or compositions
of characters are referred to in the language as symbols. Neither the set of
available characters nor the full set of available symbols will be discussed
here. Certain symbols are called basic symbols. They are basic constituents of
the language and are syntactically defined by enumeration. All but one, the
so-called string quote symbol, form together with the other, non basic symbols
the set of so-called string symbols. They are references of all characters or
character compositions, coherent sequences of zero, one or more of which are
referred to in the language as strings. Strings are the other basic
constituents of the language.
For letters cf. the section on letters, for strings and value symbols the
section on value denotations and for delimiters cf. the section on delimiters.
The text of the program is considered to be presented as an ordered sequence of
symbols. This order will be called the lexicographic order. Typographical display
features such as blank space and change to a new line do not influence this
order, but may be used freely for facilitating reading.

Identifiers.

Syntax.

I : letter ; I, letter ; I digit.
HI : I.

Semantics.

Identifiers have no inherent meaning, but serve for the identification of
quantities, labels and formal parameters. They may be chosen freely.
Every identifier occurring in the program must be defined. If it identifies
a quantity then it is defined by its occurrence immediately following the
declarator in a declaration of a quantity. If it identifies a label, then it
is defined by its occurrence in a label definition. If it identifies a formal
parameter, then it is defined by its occurrence as a formal parameter. The
identification of the quantity, label or formal parameter identified by a
given occurrence of an identifier is determined by the following process:
First the smallest block embracing the given occurrence is considered.
Step 1: If the identifier is defined within this block by a declaration of
a quantity or by a label definition, then it identifies that quantity or that
label.
Step 2: Otherwise, if that block is a declared meaning, and if the identifier
is identical with a formal parameter in the immediately preceding formal
descriptor, then it identifies that formal parameter.
Otherwise, the process is repeated considering the smallest block embracing the
block which has previously been considered.
If either step 1 en step 2 could lead to more than one quantity, label or formal
parameter, then the identification is undefined.
The scope of a quantity, a label, or a formal parameter is the set of occurrences
of the identifier which by the given process are shown to identify that quantity,
label, or formal parameter.

Sequences and lists.

Syntax.

F sequence : F ; sequence, F.
G list : G ; F sequence, comma S, F.

Semantics.

The elements of sequences and lists have successive ordinal numbers running
from one upwards.

Values and value denotations.

Syntax.

VN : single VN ; structured VN.
single VN : AVN ; BVN ; string VN ; active VN.
structured VN : tree VN ; TJRVN.

complex VN : complex number N ; real VN.
complex number N : real part N, imaginary part N.
real part N : real VN.
imaginary part N : plus i times S, real VN.

real VN : real number N ; integral VN.
real number N : unsigned real number N ; mines S, unsigned real number N.
unsigned real number N : fixed point N ; floating point N.
fixed point N : integral part N, fractional part N.
integral part N : unsigned integer N.
fractional part N : point S, digit sequence.
floating point N : unscaled part N, scale factor N.
unscaled part N : fixed point N ; integral part N.
scale factor N : times ten to the power S, integral VN.

integral VN : unsigned integer N ; mines S, natural number N.
unsigned integer N : zero S ; natural number N, digit.
natural number N : figure ; natural number N, digit.
figure : one S ; two S ; three S ; four S ; five S ; six S ; seven S ; eight S ;
        nine S.
digit : zero S ; figure.

logical VN : true S ; false S.

binal VN : bit sequence.
bit : flip S ; flop S.

string VN : string quote S, string S sequence, string quote S ;
        string quote S, string quote S.

tree VN : opening bracket S, closing bracket S ; opening bracket S, VN list,
        closing bracket S.

TKVN : TK, comma A, TVN.
TJRVN : opening brace S, TJ, closing brace S.
TJRKVN : TJRK, comma S, TJRVN.

active VN : vanity S.

Semantics.

A value is said to be either a simple value or a structured value, i.e. an ordered set of values, and to be of a certain type. The following types of simple values are distinguished.

complex value, i.e. a complex number, a special case of which is a real number.

real value, i.e. a real number, a special case of which is an integral number.

integral value, i.e. an integer.

logical value, i.e. either true or false.

binal value, i.e. an ordered sequence of bits, a bit being either flip or flop.
The number of bits is called the length of the bit sequence. The elements have ordinal numbers running from one to this length.

string value, i.e. an ordered sequence of string symbols. The number of symbols is called the length of the bit sequence. The elements have ordinal numbers running from one to this length.

active value, i.e. only vanity.

The following types of structured values are distinguished.

tree value, i.e. an ordered set of values.

and an infinite number of different types of

row value, i.e. an ordered non empty set of values of equal type. The number of elements is called the width of the tree value or row value.

The elements of a tree value other than the empty set and of a row value have successive ordinal numbers. The lowest and uppermost ordinal numbers of the elements of a tree value are one and the width of the tree. The lowest and uppermost ordinal numbers of the elements of a row value are given by the values of the lower bound and upper bound which occur in the associated declaration.

Associated with a value is a value denotation which is syntactically defined for all value types, i.e. a conveniently chosen sequence of basic symbols which can be handled in the language or by the computer. With the operations on values which are a part of the information processing to be described, correspond also operations by the computer on the value denotations it uses.

Programs.

Syntax.

program : active VE.

Semantics.

The elaboration of a program written in the language describes the processing
of information performed by a computer, i.e. an automaton or a human being.
The language distinguishes different kinds of quantities which are involved
in this process and assumes the existence of a fictitious computer. In the
computer quantities are represented by means of an internal denotation and for
each kind there exists a fixed set of different denotations. The computer disposes
over a number of discrete information units, each of which is of a fixed kind,
has a specific location and contains an arbitrary quantity of its kind.
The contents of certain information units, said to contain an instruction, may
cause the execution of a part of the process, in which the contents of other
information units, said to contain a value of a certain type, may be replaced
by another value of that type, followed possibly by the stimulation of an
information unit of the former kind to execute the next part of the process.
In the language a specific denotation for values and instructions is used
which together with the highly recursive definition of the language admits to
handle and to distinguish between arbitrarily long sequences of basic constituents,
to distinguish between arbitrarily many different values of any given type but
two and between arbitrarily many types, which admits arbitrarily many quantities
to occur in a program and which admits that the execution of a program involves
the execution of an arbitrarily large, not necessarily finite number of
executions of instructions.
This is not meant to imply that either the internal denotation of quantities
in the computer is the denotation used in the language on that the computer admits
the same possibilities. It is, on the contrary, not assumed that the computer
can handle arbitrary amounts of presented information. It is not assumed that
the two denotations are the same or even that a one to one relationship exists
between them in that the set of different internal denotations of quantities
of a given kind may be finite and that the number of kinds for which this set
is not empty may be finite. It is not assumed that the number of information
units is sufficient to cope with the requirements of a given program and it is
not assumed that the speed of operation of the computer is sufficient to execute

a given program within a prescribed lapse of time.
A model of the fictitious computer, using an actual machine, is said to be
an implementation of the language, if it does not restrict the use of the language
for other reasons than the ones mentioned above. Otherwise, if additional
restrictions can be formulated defining a language whose programs are a subset
of the programs of the language then a model is said to be an implementation
of a subset of the language if it does not restrict the use of the language
for other reasons than the ones mentioned above and those additional restrictions.

A sequence of basic constituents which is not a program but can be turned into
a program by a certain number of deletions or insertions of basic constituents
and not by a smaller number is said to be a program with that number of syntactical
errors. Any program that can be obtained by performing that number of deletions
or insertions may be called the possibly intended program. Whether a program
or one of the possibly intended programs describes the process that the writer
of it intended to describe is a problem outside the realm of the language.

Options.

Syntax.

Q option : basic Q ; if clause, Q, else S, Q option;
            case clause, opening parenthesis S, Q list, closing parenthesis S.
if clause : if S, logical VE, then S.
case clause : case S, integral VE, of S.

Semantics.

An option is a syntactical construction which displays a number of constituting
constructions. The elaboration of an option determines a constituting construction,
called the choice of the option.
The elaboration of an option is a unit of action which may consist of smaller units
of action, viz. the elaboration of other options and of expressions.
The choice of a basic construction is that basic construction.
The choice of an option which begins with an if clause is made after the
elaboration of the logical value expression contained within it. If its value
is true then the choice is the construction between the if clause and the else
symbol. If its value is false then the choice is the construction following
the else symbol. If its value is undefined then the choice is undefined.
The choice of an option which begins with a case clause is made after the
elaboration of the integral value expression contained within it. The choice
is the element of the list of constructions whose ordinal number equals the
value of that expression. If this value is undefined or if no element with that
ordinal number exists then the choice is undefined.

Blocks.

Syntax.

block : UM block.

UM block : opening parenthesis S, UM block body, closing parenthesis S.

UM block body : UM block tail ; declaration, semicolon S, UM block tail.

UM block tail : UME ; UM block E, block separator, UM block tail.

UM block E : UME ; VE.

block separator, semcolon S ; block separator, label definition.

label definition : label, colon S.

label : label I.

Semantics.

The elaboration of a block is performed in the following steps.

Step 1: If an identifier occurs within the block which identifies there another quantity, label or formal parameter than it would do at the place from where the elaboration of the block is initiated, then it is replaced within the block by an identifier which is defined neither within the block nor at that place, and step 1 is repeated. Otherwise step 2 is performed.

Step 2: If the block contains declarations, then these are elaborated.

Step 3: The lexicographically first expression of the blocktail is considered.

Step 4: The considered expression is elaborated. If this elaboration appoints a successor within the blocktail, then its result is ignored, the successor is considered instead and step 4 is repeated. Otherwise this elaboration is said to terminate the elaboration of the block, and its result is the result of the block.

Declarations and designators.

Syntax.

D : UMD.

VMD : simple UMD ; UMED.

simple UMD : UDS, MDS, UMI list.

UMED : UDS, MDS, formal UM descriptor, equals S, UM meaning.

UM designator : actual UM descriptor.

UM meaning : quotation S, UME, quotation S.

ZUM descriptor : UMI ; UMWI, opening parenthesis S, ZW, closing parenthesis S.

ZYPW : ZYP, P separator S, ZW.

P separator S : comma S ;

                opening parenthesis S, letter sequence, closing parenthesis S.

formal YP : YCS, YI.

UMCS : UCS, MCS.

UMWICS : UMCS, IS ; UMCS, WCS, IS.

YPCS : PS ; YCS, PS.

YPWCS : YPCS, WCS.

TCS : TS.

TJRCS : TS, RS ; TS, JRCS ; TJRDS.

KRCS : RS.

JRKRCS : JRCS, RS.

TDS : TS.

TJRDS : TS, JRDS.

KRDS : RS, bound pair.

bound pair : opening bracket S, lower bound, colon S, upper bound, closing bracket S.

lower bound : integral VE.

upper bound : integral VE.

JRKRDS : KRDS, JRDS.

actual UMWIP : UMWINP.

actual UMEP : UMENP ; UMEVP.

YNP : YNP option.

basic YNP : quotation S, Y, quotation S.

UMEVP : UME.

Semantics.

A declaration of a certain type and kind is a denotation of one or more information units of that type and kind, called the result of the declaration. The elaboration of a declaration determines its result. It is a unit of action

which may involve other units of action, viz. the elaboration of expressions. It establishes an information unit of the type and kind whose declaration symbols occur in the declaration, for each identifier in the list which occurs in the declaration and whose location is that identifier.
The contents of the information unit established by a simple declaration of a single type is a value of that type which is left undefined.
The contents of the information unit established by a simple declaration of tree type contains a reference to a sequence of information units, which sequence is not established, and whose locations will be the location of the former information unit subscripted with an ordinal number which will run successively from one upwards. The contents of the information unit established by a simple declaration of a type which is row of another type contains a reference to a sequence of information units of that other type and whose locations are the location of the former information unit subscripted with an ordinal number which runs successively from the elaborated value of lower bound up to the elaborated value of the upper bound, which values also are part of the contents of the former information unit.
The contents of the information unit established by an expression declaration contains moreover a reference to the text of the expression which is contained in its meaning and if the formal descriptor contains formal parameters also references to information units whose contents are those formal parameters.
A designator is elaborated in the following steps.
Step 1. All actual value parameters, if any, are elaborated in an unspecified order.
Step 2.A fictitious copy is taken of the text of the expression to which the identifier of the designator refers, and enclosed between parentheses, to make a block of it.
Step 3. In this block identifiers are changed as described in step 1 in the section on blocks, where the place from where the block is initiated is understood to be the position where the designator occurs.
Step 4. In the modified block each occurrence of a formal parameter that corresponds with an actual denotation parameter is replaced by the text of the expression contained within it and each occurrence of a formal parameter that corresponds with an actual value parameter is replaced by either the text or the location of that actual parameter depending upon whether the formal parameter occurs as value expression or not.
Step 5. This block as modified in step 3 and 4 is elaborated as if it were written at the position of the designator.

Expressions.

Syntax.

E : UME.

primary UME : UME option.

basic UME : UM primitive ; UM block ; UM designator ; subscripted UME.

subscripted UME : static subscripted UME ; dynamic subscripted UME.

static subscripted TME : basic TKRME, subscript.

static subscripted TJRME : basic TKRJRME, subscript.

dynamic subscripted UME : basic tree ME, subscript.

subscript : opening bracket S, subscript E, closing bracket S.

subscript E : integral VE.

Semantics.

An expression of a certain type is a denotation of one or more information
units of that type, called the result of the expression.
The elaboration of an expression determines its result. The result may be
either an information unit which has been established already by the declaration
of a quantity or otherwise one which is established ad hoc. The value of a
value expression is the value of its result. The location of a location
expression is the location of its result. The value and location of a variable
expression are the value and location of its result respectively.
The elaboration of an expression is a unit of action which may consist of
smaller units of action, viz. the elaboration of options, of substitutions,
of other expressions and of name parameters.
The result of a value identifier, a location identifier or a variable identifier
is the information unit established by the declaration that defines the identifier.
For the result of a block or designator cf. the sections on blocks and designators.
The result of a subscripted expression is the information unit of that element
of the result of the expression itself whose ordinal number equals the value
of the subscript expression. If that value is undefined or if no element with
that ordinal number exists, then the result is undefined.
The elaboration of an expression option exists of the elaboration of the option
followed by the elaboration of its choice. A not selected expression is not
elaborated.

Variable expressions.

Syntax.

UXE : primary UXE.

Semantics.

Variable primitives are not denotable.

Location expressions.

Syntax.

ULE : simple ULE ; ULE, also S, simple ULE.
simple ULE : primary ULE ; UXE.

Semantics.

Location primitives are not denotable.
The result of a location expression which contains an also symbol is the result
of the location expression preceding it as well as that of the simple location
expression following it.

Value expressions.

Syntax.

UVE : simple UVE ; UXE.

Semantics.

The result of a value denotation is an information unit established ad hoc
whose value is denoted by that value denotation.
Expressions may stand as operands in other expressions. The result of an expression
formed either by a monadic operator followed by an operand or by two operands
separated by a dyadic operator is an information unit established ad hoc whose
value is obtained by performing the operation indicated by the operator on the
value(s) of the operand(s).
The order in which the operands are evaluated is left undefined and, moreover,
if the result of the operation does not depend on the value of an
operand, then the evaluation of that operand may be omitted.

Complex, real and integral value expressions.

Syntax.

simple AVE : A term : simple AVE, plus O, A term.

plus O : plus S ; minuss S.

A term : A factor ; A term, A times O, A factor.

complex times O : real times O.

real times O : times S ; divided by S.

integral times O : times S ; quotient S ; remainder S.

A factor : A secondary VE ; A factor, exponent O, exponent.

exponent O : to the power S.

exponent : unsigned integer ; absolute O, integral VE.

absolute O : absolute value of S.

complex secondary VE : complex primary VE ; monadic minus O, complex primary VE.

monadic minus O : minus S.

real secondary VE : real primary VE ; sign O, real primary VE ;

                   absolute O, complex primary VE.

sign O : absolute O ; monadic minus O.

integral secondary VE : integral primary VE ; sign O, integral primary VE.

AV primitive : AVN.

Semantics.

The operation denoted by an operator is suggested by the word used to name it. The value of a complex or real value expression consisting of an operator and one or two operands is the mathematically understood value produced by the operation on (an) operand(s) which may deviate from the given operand(s). The same holds for an integral expression if the mathematically understood value would otherwise exceed certain unspecified limits.

Logical and binal value expressions.

Syntax.

simple logical VE : logical term ; relation.

binal VE : binal term.

B term : B factor ; B term, or O, B factor.

or O : or S.

B factor : B secondary VE ; B factor, and O, B secondary VE.

and O : and S.

B secondary VE : B primary VE ; not O, B primary VE.

not O : not S.

relation : UVE, equals O, UVE ; AVE, order O, AVE.

equals O : equals S ; differs from S.

order O : is less than S ; is at most S ; is at least S ; is greater than S.

BV primitive : BVN.

Semantics.

The operation denoted by the operators is suggested by the words used to describe
them. An operation on binal values is performed on each pair of corresponding
bits of the operands and an operation on logical values is aequivalent with an
operation on binal values if true and false are considered to be aequivalent
with flip and flop respectively.

String value expressions.

Syntax.

Simple string VE : string primary VE ; string VE, concatenation O, string primary VE.
concatenation O : concatenation S.
string V primitive : string VN

Semantics.

The concatenation of two strings denoted by the concatenation operator yields
a string which is the string symbol sequence of the first operand continued
by that of the second one.

Tree value expressions.

Syntax.

simple tree VE : tree primary VE ; tree VE, concatenation O, tree primary VE.
tree V primitive : tree VN ; expression tree.
expression tree : opening bracket, E list, closing bracket.

Semantics.

The elaboration of an expression tree consists of the elaboration of all
expressions of its expression list in some unspecified order. The denotation
of its value is obtained by replacing in the expression list each expression
by a denotation of its value.
The concatenation of two trees denoted by the concatenation operators yields
a tree which is the ordered set whose elements are the elements of the first
operand in their order followed by the elements of the second operand in their
order.

Row of some type value expressions.

Syntax.

simple TJRVE : TJR primary VE.
TJRV primitive : TJRVN.

Active expressions.

Syntax.

simple active VE : primary active VE.
active V primitive : active VN ; store ; jump.
store : ULE, becomes S, UVE.
jump : jump S, label I.

Semantics.

The elaboration of an active value expression may have three effects: it yields
a result, it may appoint a successor and it may change the value of one or more
information units.
The value of any active value expression is the one denoted by the vanity symbol.
The successor of a jump is determined in the following steps.
Step 1. Among the blocks which are being elaborated the one whose elaboration
was initiated last is considered.
Step 2. If some block separator of this block contains the label identifier
which occurs in the jump, then the successor is the expression which follows
that block separator. Otherwise, the elaboration of this block is considered
as terminated and Step 1 is taken as specified above.
The successor of a block or designator whose elaboration is terminated by the
elaboration of a jump is the successor of that jump. The successor of any other
active value expression which is the choice of an active value expression option
which is followed by a block separator, is the expression which follows that
block separator.
Any other active value expression has no successor.
The elaboration of an active value denotation or a jump causes no other effect.
The elaboration of a store causes the elaboration of its location expression
and its value expression in some order, followed by the assignment of the
resulting value to the resulting locations.

Conventions.

The conventions referred to at the end of the section on language and metalanguage
are expressed by means of productions of the strict language which may be
replaced by simpler constructions or even may be omitted altogether.
If in a convention a syntactical entity followed by a digit occurs then this
denotes that only those texts are terminal productions in which corresponding
with each occurrence of that entity when followed by the same digit within the
whole convention the same terminal production of that entity occurs.
Convention 1: A vanity symbol may be omitted.

Convention 2: A variable declaration symbol may be omitted.

Convention 3: A semicolon symbol immediately following a semicolon symbol,
a colon symbol or an opening parenthesis symbol may be omitted.

Convention 4: The active value expression
opening parenthesis S,
       if S, logical VE1, then S, active VE1, else S,
       closing parenthesis S
   may be denoted by
       if S, logical VE1, do S, active VE1.

Convention 5: The active value expression
opening parenthesis S, integral DS, I1, I2, I3, semicolon S,
            I1, becomes S, integral VE1, semicolon S,
            I2, becomes S, integral VE2, semicolon S,
            I3, becomes S, integral VE3, semicolon S,
label 1, colon S, if S, I1, minuss S, I3, is greater than S, zero S, and S,
           I2, is greater than S, zero S,   ,   or S,
           I1, minuss S, I3, is less than S, zero S,   and S,
           I2, is less than S, zero S,
         then S, goto S, label 2,
         else S, opening parenthesis S,
               active VE1, semicolon S,
               I1, becomes S, I1, plus S, I2, semicolon S,
               goto S, label 1,
           closing parenthesis S,
label 2, colon S,
closing parenthesis S,
        and the active value expression
opening parenthesis S,
  label 1, colon S, opening parenthesis S,
              active VE1, semicolon S,
              goto S, label 1,
          closing parenthesis S,
closing parenthesis,

in both of which the active value expression 1 does not contain label 1 and
label 2, may be denoted by
for S, I1, from S, integral VE1,
         step S, integral VE2,
         until S, integral VE3,
            do S, active VE1,
and
for S, I1, from S, integral VE1,
         step S, integral VE2,
         do S, active VE1
respectively. If,moreover, the active value expression 1 does not contain
identifier 1 then
         for S, I1
may be omitted.
Convention 6: The sequences
         from S, one S
           and
         step S, one S
may be omitted.
Convention 7: The active value expression
opening parenthesis S,
         label 1, colon S, if S, logical VE1,
                 then S, opening parenthesis S,
                     active VE1, semicolon S,
                     goto S, label 1,
                  closing parenthesis S.
              else S,
closing parenthesis S,
    may be denoted by
while S, logical VE1, do S, active VE1.
Convention 8: The sequences
         closing bracket S, opening bracket S
         and
closing bracket S, row declaration S, opening bracket S
         may be replaced by
           comma S

Convention 9: The sequence

comment S, S sequence, semicolon S,

where the symbol sequence does not contain a semicolon symbol, may be inserted between any two basic constituents.

Convention 10: A letter sequence may be inserted after each closing parenthesis symbol.

Letters.

Syntax.

```
letter : letter a S ; letter b S ; letter c S ; letter d S ; letter e S ;
         letter f S ; letter g S ; letter h S ; letter i S ; letter j S ;
         letter k S ; letter l S ; letter m S ; letter n S ; letter o S ;
         letter p S ; letter q S ; letter r S ; letter s S ; letter t S ;
         letter u S ; letter v S ; letter w S ; letter x S ; letter y S ;
         letter z S ; other letter.
```

Semantics.

All letters are basic symbols. For the first twenty six mentioned above
productions are given in the section on basic symbols. They form together
a so called alphabet. The other letters are not defined.

Basic symbols.

Semantics.

Each symbol is represented in a presentation or an application of the language
by a representation, i.e. a specific distinguishing mark. Which mark is chosen
to represent a specific symbol is logically irrelevant. In this presentation
of the language, the so-called reference language, the marks chosen to represent
the basic symbols are shown in the syntax. Which marks are chosen to represent
the non basic symbols is left open.
Each version of the language in which representations are used which are
sufficiently close to the marks given here to be identified with them without
further elucidation is equally well entitled to be called reference language.
Versions of the language in which notations or representations are used which
are not obviously identifiable with the ones defined here but bear a one to one
correspondence with them may be referred to as publication language or hardware
language, i.e. versions of the language adepted to the supposed preference of the
human or mechanical interpreter of the language.
The fact that the representations of the letters, given above, are usually
referred to as small letters is not meant to imply that the so called corresponding
capital letters could not serve equally well as representations. On the other
hand if both a small letter and the corresponding capital letter occur, then
one of them is considered to be the representation of an other letter.
If in a program a mark occurs outside a string which does not match one of the
given symbol representations, then it is to be interpreted as an other letter,
provided that a letter is syntactically admissible in that position. Otherwise
its occurrence may be ignored.