Zurich Conference on
"U n i v e r s a l l a n g u a g e"

Chairman: Backus
Secretary: Bottenbruch 29.5.58

18.) Delimiters.
Several times no conclusion could be reached as to which delimiters should be used. In these minutes we use in all those cases, where we should write "some special delimiter", one of the different proposals. It is thereby understood, that all the delimiters introduced in these minutes are tentative and will possibly be changed. We add an appendix to the minutes containing the tentative form of the delimitors and an indication of their use. The delimiters proposed here are mostly (but not in examples) contained in a square, like [ ] or [goto] .

19.) Trees [1])
The concept of tree has been introduced as a special kind of variable.
Syntax: Identifier, followed by left parantheses [{], followed by a variable number of integer expressions separated by [:], followed by right parantheses [}].
The identifier is called the name of the tree variable or tree.
Example:

a{1:2:n+j}

a1{2:n+j}

Semantics: A tree variable has only two values, namely true (indicated in what follows by T) and false (F).
Tree variables with equal names are interconnected in a way which is governed by decimal convention not to be stated here in generality.
Examples: (Ex1) If the value of a{1} is T, then the values of all tree variables with name "a", the first index of which is not 1, is F.
(Ex2) If the value of a{1:3:5} is T, then the value of a{1} and the value of a{1:3} is T.

General considerations
Variables may be then:
Boolean
Integer
Tree
General
} and all of these (except possibly trees) may form arrays.

(2) all predecessors of 1 are 1, numbers are in the range
(1) all successors of 0 are 0
(3) one 1 at most for any one level

Warning The value of all parallel cases of a tree variable may be F, but there may be only one T.

---

1) Somewhere in this paragraph should be (but unfortunately is not) contained a footnote proposed by SAMELSON that some things can be dangerous.

20.) Assignment of values to Tree variables.
Another case of assignment statements.

Syntax: T [<=] E where E is a Boolean expression (see 21.) and T is a tree variable (see 19.).

Semantics: The assignment of the value T to a tree variable implies the assignment of the value F to "parallel cases" and the value T to all "predecessors" of that variable.

21.) Boolean expressions.
Cases 0 to 12 of page 9 of the American proposal are accepted, but in addition we have case
13. A tree variable is a Boolean expression.

22.) Use of Tree variables.
A tree variable may be used wherever a Boolean expression may be used, and nowhere else, and in the same meaning.

23.) Statements and Compound statements.

Statement brackets. There are some delimitors which indicate that the sequence of statements contained between them are to be treated as a new statement. There are several opening brackets, namely

[BEGIN] , [IF] , [FOR] , [PROCEDURE]

There is only one closing bracket, namely

[END] .

Syntax of Compound statements. The sequence of statements included in matching statement brackets, inclusive the brackets, is to be labelled, this label is then the label of the new compound statement. The label is part of the statement.

24.) Conditional Statements.

Syntax: [If] P [,] S [END] where P is a Boolean expression
S is a sequence of statements.

Semantics: The sequence of statements S is to be executed, if P is true, otherwise skip S.

Warning: Our first proposal was different, "[If] P" was a quantifier, the range of which was the next statement. If a sequence of statements should be conditioned, a [begin] and [end] was to use. In the case of a single statement no end was necessary. Afterwards, in analogy to loops, the [If] was viewed as opening bracket. But now, also
after a single conditional statement, an END must occur.
Was this really intended? What is in case [If] P, goto 17 [END] .

25.) Switches
The American proposal page 11 is structurally accepted, but 3.) is rejected, since with little more effort in writing it can be handled by conditional statements.

Designational expressions and vectors of designational expressions.

A Designational Expression is:

1) A label of a statement
2) An expression L(E), where L is a label and E is an integer expression.

There must be a corresponding statement defining the vector L( ).

A vector of designational expressions is defined by a statement of the form:

L [←] (e_1, e_2, ..., e_n), where L is a label and e_1, ..., e_n are designational expressions.

26.) Statement Calls.

Syntax: [Do] A [through] C

Meaning: All statements which lie between statements A and C or which begin between A and C are to be copied. If A and C are nct on the same level difficulties will be encountered and the translator probably will not give desired result.

27.) Substitution in Statement Calls.

Syntax: [Do] A [Through] B [[] E1 [:>] I1 ; E2 [:>] I2 ; ... []]

Here A and B are labels of statements. E1, E2,... are strings of symbols not containing the delimiter [:>], and I1, I2,... are labels or anything identifiable between delimitors.

Meaning: Each occurance of I1 has in the copying process to be replaced by the string of sybols E1,... .

28.) Loops.

Syntax: [FOR] V [<=] r
S
[END].

V means a variable, r means a list of values (as in the American proposal) and S means a sequence of statements.

Meaning: Clear.

29.) Stop.

Stop is a delimiter, and is an imperative statement.

Meaning: Machine stops.

3o.) Procedure Definition.

[Procedure] is a delimiter with a scope, which is given by the matching [END]. The set of statement between these statement brackets is called a procedure. All identifiers and labels in the labels in other procedures except a few ones which are the names of the procedure. Some "things" may be substituted in a procedure, and these must be stated explicitly.
procedure are independent from all identifiers and

Syntax:

```
procedure I1 [...] I2 [...] ... In [...]
      statements 1
            I1'
      statements 2
            In'
               .
               .
               .
      statements n
            In'
      statements (n+1)
END
```

I1, I2, ..., In are identifiers which are characteristic of the procedure, and I1' to In' are the same identifiers as I1 to In in some permutation. The entries in [...] will be discussed in 31. Some of the names are defined to be "Functions", and this case, there are assignment statements which on the left of the <= have the name of these functions. Note: it is not clear that this provision is sufficient to define what names are functions. Some of the statements are RETURN statements.

Meaning: Clear.

31.) ~~Variables~~ Parameters in Procedures.

Definition	Call
Variables, inclusive subscr. variables with constant subscripts. The variable may be of any kind, that is: Integer, Boolean, Trees, General, but must be replaced properly.	Expressions; for output variables only variables and subscripted variables.
F[, ,] where F is the name of a function.	same
A[, ,] where A is the name of an array.	same
P(, ,) => ( ) where P is the name of a procedure.	same
L where L is the ~~name~~ Label of a statement occuring in a GOTO statement.	same

It is agreed that strings of symbols (as indicated under 3.) page 17? of the American proposal) can be substituted only for special kinds of "symbols", which occur in procedure statements.
It is agreed to take array statements into the language which are structurally identical to the American proposal.

Appendix: D e l i m i t e r s

tentative proposal	final proposal	Use
{ }		enclose index part of a tree
:		separates index positions in index part of a tree
⇐		used in assignement statements, where the variable to be assigned a value is on the left
BEGIN IF FOR PROCEDURE		opening statement brackets
END		closing statement brackets
,		separating Boolean expression and statements in conditional statements
( )		1) Enclosing the integer expression in designational expressions 2) Enclosing a list of designational expressions
⟵		defining lists of designational expressions
[ ]		enclose the substitution part of a DO -statement
→		substitute the string of symbols on the left for the label on the right
- - - - ?		marks the end of the name part of procedure
FOR		
⇐		between a variable and a list in FOR
RETURN		