SETL NEWSLETTER NO. 191 — ART GRAND
MORE LANGUAGE CHANGES — MAY 6, 1977

THIS NEWSLETTER DESCRIBES A FEW MINOR CHANGES TO THE SYNTAX AND SEMANTICS OF THE SETL SYSTEM NOW UNDER DEVELOPEMENT.

CODE BLOCKS

WE ARE RESTORING A FORM OF ≠CODE BLOCKS≠ SIMILAR TO THAT DESCRIBED IN ≠ON PROGRAMMING≠. A CODE BLOCK IS A SERIES OF STATEMENTS WHICH RETURNS A VALUE, AND CAN THUS BE USED WHEREVER AN EXPRESSION IS EXPECTED.

THE FORM OF A CODE BLOCK IS:

```
BEGIN. <STATEMENT LIST> END.;
```

STATEMENTS OF THE FORM

```
RETURN <EXPRESSION>;
```

CAN APPEAR ONLY WITHIN CODE BLOCKS. THEIR ACTION IS TO EXIT THE CODE BLOCK, RETURNING <EXPRESSION> AS THE VALUE OF THE BLOCK. THIS IS THE ONLY LEGAL WAY TO EXIT A CODE BLOCK.

SUBROUTINES, FUNCTIONS, AND USER DEFINED OPERATORS

A SUBROUTINE DEFINITION CONSISTS OF A DEFINE STATEMENT, A GROUP OF DECLARATIONS, A STATEMENT LIST, AND AN END STATEMENT.

A FUNCTION OR OPERATOR DEFINITION CONSISTS OF A DEFINEF STATEMENT, A GROUP OF DECLARATIONS, AN EXPRESSION, AND AN END STATEMENT. THE EXPRESSION MAY OF COURSE BE A CODE BLOCK.

WE ALLOW NILADIC OPERATOR DEFINITIONS, FOR EXAMPLE:

```
DEFINEF MY¬NEWAT.;
    NEWAT;
END;
```

ITERATORS

ANY ITERATOR MAY NOW HAVE A WHILE CONDITION; THE CURRENT WHILE ITERATOR IS ELIMINATED. THUS THE SYNTAX OF ITERATORS BECOMES:

```
<ITERATOR>  ⇏ <ITERATOR EXPRESSION LIST> /
              <ITERATOR EXPRESSION LIST> ↑ <EXPRESSION>

<ITERATOR EXPRESSION LIST>
            ⇏ <ITERATOR EXPRESION> /
              <ITERATOR EXPRESSION> <ITERATOR EXPRESSION LIST>

<ITERATOR EXPRESSION>
            ⇏ <RANGE PART> /
              <WHILE PART> /
              <RANGE PART>  <WHILE PART>

<RANGE PART> ⇏ SET ITERATOR, MAP ITERATOR OR ARITHMETIC ITERATOR

<WHILE PART> ⇏ WHILE. <EXPRESSION>
```

SOME EXAMPLES ARE:

```
(∨ I = 1, 2 ... WHILE I LT. 50)  PRINT I;;

S = ≤<X, Y> ⇏ S1 WHILE X LT. Y ↑ C(X, Y)≥;
```

COMPOUND ITERATORS MAY HAVE MORE THAN ONE WHILE CONDITION, FOR EXAMPLE,

```
(∨ X = T(I) WHILE X NE. OM., Y = F(X) WHILE. Y EQ. 0) PRINT X;;
```

HERE THE OUTER LOOP CONTINUES WHILE X IS NOT OMEGA; THE INNER LOOP CONTINUES WHILE Y IS ZERO.

TRUE AND FALSE

AFTER MUCH DEBATE ABOUT THE MEANING OF TRUE AND FALSE, WE HAVE COME TO THE FOLLOWING CONCLUSIONS:

≠TRUE≠ IS DEFINED TO BE THE BIT STRING B≠1≠, AND ≠FALSE≠ IS DEFINED TO BE THE BIT STRING B≠0≠.

IF AN EXPRESSION IS USED AS THE CONDITION OF AN IF OR WHILE STATEMENT, ITS VALUE MUST BE IN THE SET ≤B≠1≠, B≠0≠≥. ANY OTHER VALUE WILL CAUSE AN ABORT.

RANGE APPLICATION AND LEXICAL CHANGES

WE ARE CONSIDERING MAKING A SERIES OF LEXICAL CHANGES WHICH WILL MAKE THE LANGUAGE MORE READABLE. AS A SIDE EFFECT OF THESE CHANGES, SOME OF THE GENERALIZED RANGE CONSTRUCTS WILL HAVE TO BE ELIMINATED.

TUPLE FORMERS ARE NOW WRITTEN USING SQUARE BRACKETS, I.E. [1, 2].

THE RELATIONAL OPERATORS ARE NOW WRITTEN:

OLD	NEW
LT.	<
LE.	<=
GT.	>
GE.	>=
EQ.	=
NE.	/=

ASSIGNMENTS ARE WRITTEN ≠:=≠. THERE ARE TWO ON THE FLY ASSIGNMENT OPERATORS WHICH REPLACE THE ≠IS≠ OPERATOR. ≠A := B≠ ASSIGNS B TO A AND HAS THE VALUE B; ≠B =: A≠ DOES LIKEWISE. THE TARGET OF AN ON THE FLY ASSIGNMENT MUST ALWAYS BE A SIMPLE NAME.

THE RANGE CONSTRUCT CAN ONLY BE USED IN THE CONTEXTS F[S] AND F[S1, S2, ... SN], WHERE F IS A MAP, FUNCTION, OR SUBROUTINE.

CASE EXPRESSIONS

WE INTRODUCE A NEW CASE EXPRESSION, WHICH IS PARTICULARLY HANDY FOR USE IN BACKTRACKING. THIS EXPRESSION HAS THE FORM

N CASE. (<LIST OF EXPRESSIONS>)

ITS ACTION IS TO EVALUATE THE N-TH EXPRESSION IN THE LIST OF EXPRESSIONS AND RETURN ITS VALUE. THE REMAINING EXPRESSIONS ARE NOT EVALUATED.

BACKTRACKING CONSTRUCTS

THIS SECTION GIVES THE SYNTAX OF THE VARIOUS BACKTRACKING PRIMITIVES DISCUSSED IN NL. 186.

THE BACKTRACKING DECLARATIONS ARE:

```
BACK. <NAMELIST> END;
NOBACK. <NAMELIST> END.;
```

WE ADD THE NILADIC OPERATORS OK AND SUCCESS, AND THE STATEMENTS FAIL, ACCEPT AND REJECT.

THE NONDETERMINISTIC ARB OPERATION IS REPRESENTED BY A BACKWARDS EPSILON, WRITTEN ≠%≠. THE DETERMINISTIC ARB IS WRITTEN ≠ARB≠.

NONDETERMINISTIC CASE EXPRESSIONS MAY BE WRITTEN

```
%N CASE. (EXP1, EXP2, ... EXPN)
```

OR ABBREVIATED

```
% CASE. (EXP1, EXP2, ... EXPN)
```