

It is suggested that the following modifications and extensions for SETL be made definite.

1) THE PERIOD TO BE USED ONLY AS A SUFFIX DELIMITER.

The former use of the period as delimiter both fore and aft is no longer legal. Thus, ".and." is now "and.", ".or." is now "or.", etc.

2) THE EMPTY SET TO BE WRITTEN AS NL.. The former use of the symbol " $\emptyset$ " for the empty set is no longer legal. For example, " $x=\emptyset$ ;" is now " $x=nl.$ ;".

3) ATOMS OF TYPE FUNCTION AND SUBROUTINE will be allowed, so that subroutines and functions may be members of sets. Details will follow.

4) NEW RULES FOR OPERATOR PRECEDENCE. The rules for operator precedence are to be derived from the following principles:

- a) All binary operators except operators producing boolean from non-boolean values are to have the same precedence.
- b) Expressions containing operators of equal precedence are evaluated left to right.
- c) The monadic operators have minimal scope.
- d) Prefixing an operator by the symbol "\$" lowers the precedence of the operator one level; suffixing the operator with "\$" raises the precedence one level. Thus

$A + B \$ C + D$  is interpreted as  $(A+B)*(C+D)$

- e) The precedence is changed c levels if the positive integer constant c appears between "\$" and the operator; e.g., "\$3+" lowers the precedence of "+" three levels.
- f) We consider "‡" to be a monadic operator in expressions of the form

$$\forall x \in A \mid x \text{ gt. } 0.$$

- 5) USE OF  $\forall$  IN BLOCK HEADER. In block headers of the form  $(\forall x_1 \in e_1, \forall x_2 \in e_2(x_1), \dots)$ (block) only the first occurrence of " " is necessary or permitted, so that the header above will be written

$$(\forall x_1 \in e_1, x_2 \in e_2(x_1) \dots).$$

- 6) ALTERNATE FORM FOR SUBSET DEFINITION. The form  $\{x_1 \in e_1, \dots\}$  will be interpreted as

$$\{x_1, x_1 \in e_1, \dots\}.$$

- 7) NEW FORM FOR BOOLEAN EXPRESSION. The quantified boolean expression which was of the form

$$(\exists x_1 \in e_1, \forall x_2 \in e_2(x_1), \forall x_3 \in e_3(x_1, x_2), \dots, x_n \in e_n(x_1 \dots x_{n-1})) (C(x_1, \dots, x_n)).$$

is now of the form

$$\exists x_1 \in e_1, \forall x_2 \in e_2(x_1), x_3 \in e_3(x_1, x_2), \dots, x_n \in e_n(x_1, \dots, x_{n-1}) \mid C(x_1, \dots, x_n).$$

Note that the new features are

- (a) No use of parentheses: the old construct (range) (condition)

is now written

$$\text{range} \mid \text{condition}.$$

where (according to 4))  $|$  is taken as a monadic operator.

- b) Only the leftmost range restriction need contain a quantifier ( $\forall$  or  $\exists$ ); if a range restriction contains no quantifier, then the quantifier assumed is that occurring in the nearest preceding (i.e., to the left) range restriction which does contain a quantifier.

8) NEW FORM FOR ITERATIONS. As currently defined, iteration statements are of the form

(iter header) (block)

in which the "scope" of the iteration, block is indicated by enclosing block in parentheses. The scope is now to be indicated as follows:

- a) block;

that is, put a semicolon after the last statement in the block.

or

- b) til.label; block label: statement,

that is, label the first statement after the end of the block.

or

- c) block end  $\forall x$ ;

that is, after the last statement in the block put end  $\forall x$  when  $x$  is the leftmost name in an iteration header of the form

$$(\forall x \in e, x_2 \in e_2(x), \dots).$$

To give an example, the statement

$(\forall x \in a, \forall y \in b \mid x \text{ ne. } y) (c \text{ sub. } \langle x, y \rangle); d=c;$

is now written, using a), either

$(\forall x \in a, y \in b \mid x \text{ ne. } y) c \text{ sub. } \langle x, y \rangle; ;$  (using a)

or, (using b),

$(\forall x \in a, y \in b \mid x \text{ ne. } y) \text{til lab}; c \text{ sub. } \langle x, y \rangle;$

lab:d=c;

or, using c),

$(\forall x \in a, y \in b \mid x \text{ ne. } y) c \text{ sub. } \langle x, y \rangle; \text{end } \forall x;$

d=c;

Note that, by 5), the qualifier  $\forall$  in " $\forall y$ " is not written. Similar remarks apply to while and if scopes.

9) NEW FORM FOR IF STATEMENT. The IF statement (see page 37 of SETL notes) which was of the form

if (bool1) then (block1) else if (bool2) then (block2) ...  
else (blockn);

or

if (bool1) then (block1) else if (bool2) then (block2) ... etc  
else if (booln) then (blockn);

is to be simplified as follows

- a) The boolean expressions bool<sub>i</sub> will not be enclosed in parentheses.
- b) The <sup>last</sup> block block<sub>n</sub> will not be enclosed in parentheses, but will either
  - 1) be followed by a semicolon (;), or
  - 2) be delimited by a "t<sub>il</sub>. label", i.e., written in form t<sub>il</sub>. label; block label: statement.

This is to be interpreted as

block;statement.

10) NEW FORM FOR RETURN IN FUNCTION. The value returned by a function should not be enclosed in parentheses, so that "return e", where e is a SETL expression.

11) EXTENSION OF IMAGE SET CONSTRUCTION. It is suggested that the image-set constructions  $f\{a\}$ ,  $f[a]$ , and  $f(a)$  (defined on page 23 of the SETL notes) be extended to have an arbitrary number of arguments. Thus constructs of the form  $f\{a,b\}$ ,  $f\{x,[y]\}$ , etc., will be possible.

A proposed definition for  $f\{\dots\}$  and  $f[\dots]$  is


- 1)  $f\{a\} = \{-y, y \in f \mid *y \text{ eq. } a\}$
- 2)  $f\{a_1, a_2, \dots, a_n, a\} = g\{a\}$ , where  $g = f\{a_1, a_2, \dots, a_n\}$ .
- 3)  $f[s] = \{f\{x\}, x \in s\}$
- 4)  $f[s_1, s_2, \dots, s_n, s] = g[s]$ , where  $g = f[s_1, s_2, \dots, s_n]$ .
- 5)  $f\{[s]\} = f[s]$
- 6)  $f\{a_1, a_2, \dots, a_n, [s]\} = g[s]$ , where  $g = f\{a_1, a_2, \dots, a_n\}$ , etc.
- 7)  $f(a_1, a_2, \dots, a_n) = \text{if } \# f\{a_1, a_2, \dots, a_n\} \text{ eq. } 1 \text{ then } \exists f\{a_1, a_2, \dots, a_n\} \text{ else } \perp$ .

This definition defines  $f\{a_1, a_2, \dots, a_n\}$  and  $f[s_1, s_2, \dots, s_n]$  for all positive integers n, where  $a_i$  is any set, atom or expression of the form  $[s]$  for a set s, and when  $s_i$  is any set.

For example, let

$$f = \{ \langle a, \langle aa, ab, ac \rangle \rangle, \langle b, \langle ba, bc \rangle \rangle, \langle c, ca \rangle, \langle b, \langle aa, bb \rangle \rangle \}$$

where ca is not an ordered pair.

THEN	IS	SINCE
$v1 = f\{a, aa, ab\}$	$\{ac\}$	$f1=f\{a\} = \{\langle aa, ab, ac \rangle\},$ $f2=f1\{aa\} = \{\langle ab, ac \rangle\},$ $v1=f2\{ab\} = \{ac\}$
$v2 = f\{[\{ab\}], aa\}$	$\{\langle ab, ac \rangle, bb\}$	$f1=f\{[\{ab\}]\} = \{\langle aa, ab, ac \rangle, \langle ba, bc \rangle$ $\quad \quad \quad \langle aa, bb \rangle\}$ $v2=f1\{aa\} = \{\langle ab, ac \rangle, bb\}$
$v3 = f(\{[\{ab\}], aa)$		Since #v2 = 2
$v4 = f\{c, ca\}$	nl. (empty set)	$f1=f\{c\} = \{ca\}$ $f2=f1\{ca\} = nl.$
$v5 = f(c)$	ca	$f1=\{c\} = \{ca\}$ $v5 = \exists f1=ca$ Since #f1 = 1.

12) NEW FORM FOR CONDITIONAL EXPRESSIONS. The boolean and value expressions in a conditional expression will not be enclosed in parentheses. The conditional expression which was of the form

if (bool1) then (expl) else if (bool2) ...

is to be written

if bool1 then expl else if bool2 ... .