SETL Newsletter Number 29 May 6, 1971
Some issues connected with J. Schwartz
subroutine linkage.

The 'address-entity' issues which I am trying to explore (in order to build up a scheme allowing generalized left-hand sides in assignments) touches on certain fundamental questions concerning subroutine linkage style which are not too well comprehended by the various phrases "call-by-name", "call-by-reference", "call-by-value" currently in circulation. To bring these into sharper focus, I propose the following test, which distinguishes various of the things that can happen in a linkage. A SETL-like notation is used, but of course the issues are not peculiar to SETL.

```
a=0;b(1)=1;b(2)=2;i=1;j=1;
f=(proc(x);return x;end); /* to use a BALM-like notation */
c=0;d=0; /* now for a rather conglomerate subroutine call */
sub(a,a,b(1),b,i,f(j),f,j,c+d,c,d);
/* now the body of the subroutine just called */
define  sub(a1,a2,eltofb,bb,i,valf,ff,j,cplusd,c,d);
external a,b,f;
a=1;
if a1 eq 0 then print "this a call-by-value linkage";
a1=2;
if a eq 1 then print "this is probably a call-by-value linkage
                     with delayed argument return";
b(1)=0;
if eltofb eq 1 then print "this is call-by-value for array elements
                          irrespective of how non-array elements are
                          handled";
if bb(1) eq 1 then print "this is call-by-value for entire compound
                          data structures, a form avoided in FORTRAN
                          but perhaps intended in SETL";
```

```
i=2;b(1)=1;
if eltofb eq 2 then print "linkage has call-by-name character,
                          at least for compound stored data items";
i=1;bb(i)=0;
if eltofb eq 0 then print "linkage has call-by-name or call-by-
                          reference character for compound stored
                          data items";
eltofb=3;/* this assignment may not be legal */
if bb(1) eq 3 then print "we have additional evidence of call-by-
                          name or call-by-reference linkage character";
/* now do similar tests for the programmed function */
f=(proc(x);return(-x);end); /* again using a BALM-like notation */
if valf eq 1 then print "this is call-by-value for function references,
                        possible even if array elements are handled
                        differently";
if ff(1) eq 1 then print "this linkage uses an unusual call-by-value
                        for function arguments, which is logically con-
                        sistent however with a strict call-by-value
                        for other types of variables";
if valf eq (-1) then print "this is very likely a call-by-name
                        linkage";
j=2;
if valf eq(-2) then print "additional evidence indicates that this
                        is a call-by-name linkage";
a2=2;a1=3;
return; /* note that test below is applied after return */
end sub;
if a eq 2 then print "linkage is probably extreme call-by-value
                  with return of argument postponed until moment
                  of return";
exit; /* end of first test sequence */
```

The following slightly more esoteric cases are also of interest.

```
a=0;
definef  f(x);x=1;return 0;end f;
if (a+f(a)) eq 1 then print "this somewhat eccentric case might
                        still be described as 'left-to-right'
                        evaluation order";
b(1)=0;b(2)=0;a=0;
definef  g(a,j);a(j)=1;return 0;end g;
if (b(1)+g(b,1)) eq 1 and a+f(a) eq 0 then print
                       "the generation of 'compiler temporaries'
                        can lead to subtle differences between the
                        treatment of 'simple' and 'indexed' references
                        to compound data structures";
exit; /* end of second test sequence */
```

Various slightly more far-fetched instances having to do with multiple levels of subroutine calls might also behave surprisingly. Recursive use of subroutines and functions might also show surprising features. Note for example the following case.

```
sub(1);
define  sub(x);
if x eq 1 then  sub=subl;sub(0);
      else print "this message could appear with one style of
                  compiling, though it seems untoward";return;end sub;
define  subl(x); print "this message indicates the expected style
                    of subroutine-to-name correspondence";end subl;
exit; /*  end of third test sequence */
```