SETL Newsletter # 106 May 17, 1973

Gray Jennings

User Variation of the Semantics of Function and Subroutine Invocation

The second generation of SETLB, SETLB.2, based upon a version of the BALM interpreter written in LITTLE and the SETL Run Time Library will become operational in the near future. It will offer a limited capability for variation of the semantics of subroutine and function invocation by the SETLB programmer. The subroutine or function to be invoked as a result of the fragment of source text

fname(a,b)

will be determined by the interpreter from information supplied by the user through execution of SETLB code. Two sets *alias1* and *alias2* will contain tuples of the form

<*function*,*kind1*,*realfn*>

and

<*function*,*kind1*,*kind2*,*realfn*>

Entries may be made in or deleted from these sets by SETLB source statements like

<.,.,.> <u>in</u> *alias1*

or

alias1 = *alias2* <u>with</u> <.,.,.>

The routine to be called as a result of the appearance of the code fragment

fname(a,b)

is in SETL

if *alias2* (*fname*,*type*(a),*type*(b)) is *realfn* ne Ω then realfn else *fname*

The function to be invoked when a subroutine invocation is made with one argument is determined from *alias1*. A number of user defined types will be available. A token can be designated as a type by execution of the SETL code,

k = *iskind* *k*

If the value of *k* is not already a type which may have resulted from the prior execution of a statement of this form, an integer designating a type not currently being used is assigned to designate a novel type code. The SETL code

a = *a* as *k*

results in the type field of *a* being altered to the value of *k*. The token *k* must have been declared previously to be a *type* designator.

The tokens *alias1*, *alias2*, *as*, *iskind* are reserved words and may not be used for other purposes.

The determination of the routine to be invoked in this manner is time consuming. The algorithm for determining user variations must be enabled by executing *vardef*(t). The process may be discontinued by executing *vardef*(f). It may be subsequently enabled by executing *vardef*(t).

In SETLB.2, addition is implemented as a call to the routine PLS. (a,b). Addition of

```
<'PLS.',aplarray,aplarray,apladd>
```

to *alias2* will cause the fragment ...a+b... in the context

```
aplarray = iskind aplarray;
a = a as aplarray                              (1)
b = b as aplarray
```

to be executed by a call to *apladd*(a,b) rather than to *PLS*. The first statements in the (SETL routine) *apladd* should be

```
a = a as setlkind
a = a as setlkind
```

The type field of a and of b is changed by (1) and must be restored to a type (*setlkind*) recognized by the RTL routines prior to the execution of any code which checks the types of a and b. The semantics of other operations standard to SETL may be varied dynamically by making entries into one of *alias1*, or *alias2*. The BALM-SETL manual contains the name of the procedures used.

The tokens designating the standard SETL data types are

integer	*tuple*
set	*blank* (newat)
cstring	*code* (subroutine or function)
bstring	

No distinction is made between long and short objects at the user level.