SETL NEWSLETTER Number 1: November 5, 1970

BALM-SETL - a simple implementation of SETL

M. C. Harrison

It is suggested that a preliminary implementation of SETL be done in BALM. Though inefficient, such an implementation would be simple, and would permit experimentation with the form of the language, and perhaps be useful as a bootstrap for later versions.

Given below are routines to implement most of the basic SETL operations. A set is implemented as a BALM list whose first item is the value of the variable SETMARK, which will print as ***. An ordered set is represented by a BALM list. All other data-objects in BALM can be members of sets or ordered sets. Thus the SETL items are:

integer, real, octal,
character, string,
procedure, expression
set, list, vector

with the following special objects

TRUE, FALSE, UNDEF, SETMARK
NIL(empty list), NULLSET(empty set)

Undefined will print out as OOO.

Operations

The following operations can be written as shown:

$\# a$	SIZ A	SETSIZE(A)
$\ni a$	EL A	CHOOSEL(A)
a less $\ni a$	REM A	CHREM(A)
$x \in a$	X IN A	SETMEMB(X,A)
$a \subseteq b$	A SUBSET B	SUBSET(A,B)
$a \subset b$	A PSUBSET B	PSUBSET(A,B)
a with x	A WITH X	WITH(A,X)
$a \cup b$	A UNION B	UNION(A,B)
$a \cap b$	A INT B	INTERSECT(A,B)
a less x	A LESS X	LESS(A,X)
a lessf x	A LESSF X	LESSF(A,X)
$f(x)$	F OF L	APPL(F,L)
$f\{x\}$	F SOF L	SAPPL(F,L)
$a \equiv b$	A EQS B	EQSET(A,B)
$\langle x,y,z \rangle$		LIST(X,Y,Z)
$\{x,y,z\}$		MAKSET(X;Y,Z)
$(\exists x \in a) p(x)$	P EXISTS A	EXISTS(P,A)
$(\exists x \in \ell) p(x)$	P EXISTL L	EXISTL(P,L)
$(\forall x \in a) p(x)$	P UNIVS A	UNIVS(P,A)
$(\forall x \in \ell) p(x)$	P UNIVL L	UNIVL(P,L)

The middle column shows the prefix or infix form, and the right column the procedural form. We have written A and B for sets; X, Y, and Z for arbitrary SETL items; and L for a list. Precedences have not been chosen, but can be established along with the definitions of the operators, as follows:

```
UNARY(=SIZ,=SETSIZE,750);
INFIX(=WITH,=WITH,731,730);
```

The procedure definitions are as follows:

```
SETMARK = MKVAR (<***>);
NULLSET = SETMARK: NIL;
SETSIZE = PROC(S), LENGTH(S)-1 END;
CHOOSEL = PROC(S), HD TL S END;
CHREM   = PROC(S), SETMARK: TL TL S END;
EQSET   = PROC(S1,S2), IF ATOM(S1)∨STRINGP(S1) THEN S1 ≡ S2
                       ELSEIF LENGTH(S1)≡LENGTH(S2) THEN FALSE
                       ELSEIF VECTORP(S1) THEN
                             (IF VECTORP(S2) THEN EQV(S1,S2)
                             ELSE FALSE)
                       ELSEIF HD S1≡SETMARK THEN
                             (IF HD S2 ≡ SETMARK THEN EQS(TL,S1,
                             TL S2) ELSE FALSE)
                       ELSE EQL(S1,S2)  END;
EQL     = PROC(L1,L2), IF NULL(L1) THEN TRUE
                       ELSEIF EQSET(HD L1, HD L2) THEN EQL(TL L1,
                       TL L2)
                       ELSE FALSE     END;
EQV     = PROC(V1,V2), BEGIN(L,I)
          L = LENGTH(V1), I = 1
NXT     = IF I GT L THEN RETURN TRUE;
          IF  (EQSET(V1[I],V2[I])) THEN RETURN FALSE,
          I = I+1, GO TO NXT
          END END;
```

```
EQS      = PROC(S1,S2), IF NULL(S1) THEN TRUE
                        ELSEIF SETMEMB(HD S1,S2) THEN EQS(TL S1,S2)
                        ELSE FALSE     END;
SETMEMB = PROC(E,S), IF NULL(S) THEN TRUE
                     ELSEIF EQSET(E,HD S) THEN TRUE
                     ELSE SETMEMB(E,TL S)    END;
WITH     = PROC(S,E), IF SETMEMB(E,S) THEN S ELSE SETMARK:E:TL S
          END;
LESS     = PROC(S,E), IF NULL(S) THEN NIL
                     ELSEIF EQSET(HD S,E) THEN TL S
                     ELSE HD S:LESS(TL S,E)   END;
LESSF    = PROC(S,E), IF NULL(S) THEN NIL
                     ELSEIF EQSET(HD HD S,E) THEN TL S
                     ELSE HD S:LESSF(TL S,E)   END;
UNDEF    = MKVAR(〈000〉);
APPL     = PROC(FN,A), IF NULL(F) THEN UNDEF
                      ELSEIF EQL(A,HD FN) THEN LAST(HD FN)
                      ELSE APPL(TL FN,A)     END;
SAPPL    = PROC(FN,A), SETMARK:SAPPL1(FN,A)
          END;
SAPPL1   = PROC(FN,A), IF NULL(F) THEN NIL
                      ELSEIF EQL(A, HD FN), THEN LAST(HD FN):
                      SAPPL1(TL FN,A)
                      ELSE SAPPL1(TL FN,A)      END;
```

```
MAKSET      =  NPROC(L), SETMARK:SETC(L)   END;
SETC        = PROC(L), SETMARK:REMUDS(L)   END;
REMUDS      = PROC(L), IF NULL(L) THEN NIL
                       ELSEIF HD L ≡ UNDEF THEN REMUDS(TL L)
                       ELSE WITH(REMUDS(TL L), HD L)   END;
EXISTS      = PROC(A,P), EXIST(TL A,P)  END;
EXISTL      = PROC(L,P), IF NULL(L) THEN FALSE
                         ELSEIF P(HL D) THEN TRUE
                         ELSE EXISTL(TL L,P)  END;
UNIVS       = PROC(A,P), UNIVL(TL A,P)  END;
UNIVL       = PROC(L,P), IF NULL(L) THEN TRUE
                         ELSEIF P(HD L) THEN UNIVL(TL L,P)
                         ELSE FALSE   END;
UNION       = PROC(A,B), IF B ≡ NULLSET THEN A
                         ELSE UNION(A WITH EL B, REM B)   END;
INTERSECT   = PROC(A,B), IF A ≡ NULLSET THEN NULLSET
                         ELSEIF EL A IN B THEN INTERSECT(REM A,B)
                         WITH EL A
                         ELSE INTERSECT(REM A,B)   END;
SUBSET      = PROC(A,B), IF SIZ A GT SIZ B THEN FALSE
                         ELSE SUBSET1(A,B)   END;
SUBSET1     = PROC(A,B), IF A ≡ NULLSET THEN TRUE
                         ELSEIF EL A IN B THEN SUBSET1(REM A,B)
                         ELSE FALSE    END;
PSUBSET     = PROC(A,B) IF SIZ A GE SIZ B THEN FALSE
                        ELSE SUBSET1(A,B)   END;
```

Notes:

SETC converts a list to a set, using REMUDS.

REMUDS removes duplicated SETL items and UNDEFs from a list.

MAKSET uses SETC, so can have UNDEFs and duplicated items.

Useful expression forms

The expression $(\forall x \in a)$(block) can be implemented in BALM-SETL without any additional extensions as:

```
MAPX(TL A, PROC(X), block END)
```

where block is any valid BAIM expression or command. This can easily be improved by operator and macro definitions to a form such as:

```
FOREACH X IN A REPEAT block
```

The expression $\{f(x,y) \mid x \in a,\ y \in b,\ g(x,y)\}$ could be implemented as:

```
SETC(MAPX(TL A,
    PROC(X),
    MAPX(TL B, PROC(Y), IF G(X,Y) THEN F(X,Y) ELSE UNDEF
    END)
    END    ) )
```

which looks nasty, but is actually a mechanical translation. This can also be improved by operator and macro definitions to a form such as:

```
CONSET(IF G(X,Y) THEN F(X,Y), X IN A, Y IN B)
```

The general form of this would be:

CONSET(expr, x_1 in a_1, x_2 in a_2, ...)

where expr is any expression which may be a conditional or may be a block of code. For example, we could also define INTERSECT by:

```
INTERSECT = PROC(A,B), CONSET(IF X IN B THEN X,X IN A) END;
```

Example:

Suppose we consider Boolean expressions in disjunctive normal form. Because $\wedge$ and $\vee$ are commutative, such expressions can be expressed as sets of sets of variables. Thus the expression $(A \wedge B \wedge C) \vee (B \wedge C) \vee (C \wedge D)$ can be expressed as $\{\{A, B, C\}, \{B, C\}, \{C, D\}\}$, or in BALM-SETL as:

```
S = SETC(SETC(=A,=B,=C),SETC(=B,=C),SETC(=C,=D))
```

Such an expression can be partially simplified by deleting any term which includes the variables of another term. This can be expressed as $\{x \mid x \in S, \neg(\exists y \in S)(x \supset y)\}$, which can be written in BALM-SETL as a procedure defined by:

```
SIMPL = PROC(S)
CONSET(IF ¬(PROC(Y), Y PSUBSET X) EXISTS S THEN X,X IN S)
END;
```

For example, if S is the example above, the command:

```
PRINT(SIMPL(S));
```

will print:

```
(***(***B C) (***C D))
```

the BALM-SETL representation of $\{\{B, C\}, \{C, D\}\}$.

Comments:

This implementation is highly inefficient with respect to computation time, essentially because of the time required to determine if an element is a member of a set, which itself may require the invocation of the routine to test two items for equality. Other schemes for overcoming this disadvantage are being considered. However, note that sets with common subsets are permitted to share memory in the obvious cases.