SETL Newsletter Number 54 September 24, 1971

Current Status of BALM-SETL 4 S. Gruber

The BALM-SETL 4 routines follow newsletter 49 fairly closely. There are a few points of divergence and many things which are not included in the specs, and it is these that will be elaborated upon here.

There are two major differences between the routines as they stand and newsletter 49. The first is that atoms which are not blank are represented not as triples but as BALM atoms. Thus the integer 1 is written "1" and not "repint(1)". The other is that T and F are maintained as separate entities, distinct from 1B and 0B. This is both because I can see no possible use for this feature and for reasons of convenience. It took me a while to reach this decision, so a good deal of work has been done in the other direction. The definitions of T and F still remain, I just haven't gotten around to removing them, and great care has been taken to insure that all internal routines reference TRUE and FALSE or NIL, and not T or F. Among the auxiliary decks I have is a redefinition of MELSEIF which would have made the IF macro understand 0B and 1B. There are also a few routines floating around, such as BTR, EQPP, and NEPP, which should be thrown out.

Some of the algorithms used also differ slightly. In EQUALT, no attempt was made to reduce the hash tables to the same size. A small change in the code obviated the need for this. In addition, the old routine had the problem that evaluation of S EQ NL would probably destroy S.

Some routines also need clarification. The routines MAKSETUP and MAKOUTUP are there because, largely for historical reasons, tuples inside sets are represented as two- and three-vectors (actually this speeds things up slightly) whereas tuples

outside sets are represented as lists. The latter also needs some justification. The principal reason for it is that in functional application it is necessary to take a tail, which is a horrendous operation on vectors. In addition, I decided that I would rather have HEAD, TAIL, CONS, and TPLUS be fast than OFTUPL.

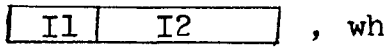
To get to the code itself, it divides itself nicely into five parts. The first is a prefix which contains fixes to some BALM problems I have encountered. It ends at the COMMENT. The second is the definition of certain constants and utility routines. A possible improvement that should be reasonably quick would be to make all of the routines in this section which are amenable to it (i.e., TYPE, HASHCODE, HTLOAD, etc., but not SETQQ, etc.) macros. The definition of RIGHTSHIFT as a macro should provide a model. The clarity of the code m might also be enhanced by making them left macros as well, and replacing all of the X(3)(1)=somethings by NELT(x)=somethings, etc.

MAKESMSZHT is not referenced anywhere. It is left in, however, in case it should be desired to restore EQUALT. If this doesn't happen, it should be removed.

The next section is the set and tuple manipulating routines. These are well documented in newsletter 49. The only routines that I know are missing are OF and OFN, because there is a bug in OFN, BOF, and BOFN as they appear in newsletter 49. The code for these, as well as INT and SYMDIF, is written.

The next section comprises those routines strictly between OF and GENSET. They comprise, generally, support for those atomic operations which are assumed in newsletter 49 but which are not supported in BALM. The bit string routines are included among these, and they will not work in BALM4, for two reasons. First, BALM4 does not have shifts, and even if it did, it would

probably work on 18 bit numbers, rather than the 60 for which the package was written. Second, BALM4 does not support reals, which are used in two places in LAST. These two, references to CEIL(J/54.0), should be replaced with CEILDIV(J,54), where CEILDIV is defined by CEILDIV= PROC(NA,NB), BEGIN(Q), Q=DIVIDE(NA,NB), RETURN(HD Q + IF TL Q NE 0 THEN 1 ELSE 0) END END; DIVIDE should be placed in the prefix, if it is not already there, with the definition
DIVIDE=PROC(NA,NB), BEGIN(Q), Q=NA/NB, RETURN(LIST(Q,NA-NB*Q))END END:.
REMAINDER=PROC(NA,NB), NA - NB*(NA/NB) END;
should be put there as well, if it isn't there already.

The conversion to 18 bits will probably be somewhat harder. LMASK=740000b, should replace the current definition, and all 59's and 60's should be replaced by 17's and 18's in the obvious way. I'm not sure that this will do it, so here follows a description of the internal format, in case it doesn't. Bit strings are represented internally as a list of integers, each with the format 

, where I1 and I2 are fields which are currently 6 and 54 bits long, respectively, but should be changed to 4 and 14. In all but the first word of the list, I1 is unused while I2 is packed with bit data. In the first word, I1 contains the number of bits there are in I2 that are meaningful, and I2 contains right-justified bit data. The reason that 6 bits per word were sacrificed is that otherwise management of bit strings becomes a real problem. This is because 57-bit long strings would have no bits in the first word, significantly complicating the algorithms. The loss was deemed small compared to the added code. However, with 4 bits out of 18 being lost, it may be wise to change this.

The last section consists of macro definitions, protect statements, and infix declarations. The macro definitions are incomplete in that they do not support the FORALL X=(1,3) QUANT qq(X) option. The expansions for these follows.

They may all be parsed by finding a SETQ where one expects an EL. The needed expansions follow:

```
FORALL X1 QUANT X2 expands as:
      BEGIN( ), FOR X1 REPEAT IF NOT X2  THEN RETURN(F),
                RETURN(T) END
EXISTS X1 QUANT X2 expands as:
      BEGIN( ), FOR X1 REPEAT IF X2 THEN RETURN(T), RETURN(F) END

SETOF X1 WHERE X2 expands as:
      BEGIN(A),A=COPY(NL),FOR X2 REPEAT AUGMENT(A,X1),RETURN(A) END
```

If X1, X1, and X2, respectively, have the format Y1=(Y2,Y3).

I have two more suggestions for improvements once you get the package working. First, it would appear that a relatively simple change to READITEM and WRITEITEM (or is it PUTITEM) in BALM4 itself would permit recognition of and printing of <S a b c d> and <T a b c d> as the set and tuple respectively of a, b, c, and d. Currently, the attempt to print out a set produces a wealth of what must appear to be gibberish to the average user. This would also allow such constructions as X=<S 1 2 3> for the SETL X={1,2,3}, instead of the current X=GENSET(1,2,3). A further modification to FNOTN would allow such things as f<B 1 2 3> to be recognized as f[1,2,3] or BOF(f,LIST(1,2,3)) as is currently coded. Note that f<S 1 2 3> is inappropriate for SOF(f,list(1,2,3)) since ordering information is lost in the construction of a set. Perhaps c for curly might be a good letter to use.

The other change that I haven't mentioned is the possibility of handling tuples the same way three and higher order tuples are now handled when they are entered into sets, i.e., by putting their second components into sets. This would both greatly increase the speed of functional application in sets whose elements are multi-tuples represented as pairs whose second components are pairs whose..., and simplify greatly the

functional application, eltf, and augment and diminish algorithms. Hank seems notably unenthusiastic, but it seems to me to be a change which would make a significant improvement.

The state of the package. I currently have in my possession two decks. One of these is the latest BALM4 version, which has still failed to compile successfully. The other is an old deck, which ran under BALM3 but was woefully incomplete. Unfortunately, since that time I have been simultaneously adding to the routines and converting to BALM4, and thus I have no other working decks. The latest deck is fairly complete, except for the omissions noted above, but doesn't work. Assuming no further problems crop up with BALM, I think that when they do compile many of them will run the first time, and the rest should be fairly easy to fix. I, however, am an optimist. Good luck!