SETL Newsletter Number 25 April 27, 1971

Print Routine

B. Loerinc

Outline: printcall is a subroutine called once. Its job is to initialize conditions and put the object s to be printed on the dot stack. Then it will exhaust the stack, sending the s at the highest level, lowest dotlabel to char.

Char is a subroutine which takes s and finds the character string representing it, adding objects to the dot stack if necessary. (A part of s is put on the dot stack and replaced by an integer followed by a period (its dotlabel) in the character representation of s, whenever it is nested to a depth greater than dep.

Post is a subroutine which periodically adds pieces of s to the output media. It checks line lengths and spaces the actual printout to improve readability.

```
define printcall(s);  dep=4;level=0;dot=nl;
string=nulc;n=0;    lastcall=0;outs=nulc;linelength=72;
dot(level,n)=s; num=3;
(while dot ne nl)nextlevel=[max:y ∈ dot] hd y; depth=0; m=0;
s=dot(nextlevel, 0≤∃[dotlabel]< n | dot(nextlevel,dotlabel)ne Ω)
dot=dot less s; if nextlevel ne 0 then post(2*
(nextlevel // (linelength/4))*' ' + dec dotlabel + '.');;
       level=nextlevel;
     char(s,level); lastcall=1; end while;
     post(string); return; end printcall;
define char(s,level); printcall external depth, level,
       n,m,outs,dot,dep,num;
       /* Printouts for various atoms are provisional, subject
to decision on actual printed representations for them. */
```

```
if atom s then if integer s then post(dec s) ;
              else if s eq t then post('*TRUE*');
              else if s eq f then post('*FALSE*');
              else if label s then post('*LABEL*');
              else if s eq Ω then post('*Ω*');
              else if blank s then post('*BLANK ATOM*');
              else if character s then post('''+s+''');
              else if subroutine s then post('*SUBROUTINE*');
              else if function s then post('*FUNCTION*');
              else if bit s then til [bitstring:]; /* All bit
string constants are printed as (binary part) b (octal part),
without parentheses or blank spaces.  The empty bit string is
denoted by Obb. */
       bitpart=(len s//3)first s; outbit=nulc; i=len bitpart;
       bp=1; tablebit={<0,'0'> <1,'1'>}; (while bp le i
       doing bp=bp+1) bitdigit=bp elt bitpart;
       outbit=outbit+tablebit(bitdigit); end while;
       octpart=(s-(len s//3)) last s; outoct=nulc;
       p=len octpart; op=1; tableoct={<000,'0'>
       <001,'1'><010,'2'><011,'3'><100,'4'>
       <101,'5'><110,'6'><111,'7'>};
       (while op le p doing op=op+3)octdigit=
       (op elt octpart)+(op+1 elt octpart)+(op+2 elt octpart);
       outoct=outoct+tableoct(octdigit); end while;
       if outbit eq nulc and outoct eq nulc then post('Obb');
       else post(outbit+'b'+outoct); end if outbit;
       [bitstring:];                go to [out:];
       end if atom;
if pair s then if(depth lt dep and m lt num)
       then post('<'); depth=depth+1; (while pair s
       doing s=tl s) post (char(hd s,level)+','); end while;
       post(char(s,level)+'>'); depth=depth-1; go to [out:];
       else n=n+1; dot(level+1,n)=s; post(dec n+'.');;
       go to [out:]; end if pair;
```

```
if ∀x∈s/(pair x and integer hd x) and 1≤∀k ≤#s/
       (s(K) ne Ω ) then go to [seq:]; else go to [set:];
       end if  ∀x;
[seq:] if (depth lt dep)and m lt num)    then post('[');
       depth=depth+1; (1≤∀k≤(#s-1))post(char(s(k),level)+',');
       end ∀k; post(char(s(#s),level)+']');depth=depth-1;go to [out:];
       else n=n+1;dot(level+1,n)=s;post(dec n+'.');go to [out:];
       end if;
[set:] if (depth lt dep  and m lt num)    then post('{');
       depth=depth+1;
       definef comma(a); if not(1 last a∈ {'{','[','<'})
       then b=','; else b=nulc;; return b; end comma;
       (∀x ∈  s)post(comma(outs)+char(x,level));end ∀x;
       post('}'); depth=depth-1; go to [out:];
       else n=n+1; dot(level+1,n)=s; post(dec n + '.');
       go to [out:]; end if;
[out:] return; end char;
define post(obj); /* If nesting of sets, sequences or pairs is
greater than 4 deep, the nested sets, sequences or pairs will be
assigned abbreviated designators consisting of an integer followed
by a decimal point, and it will be printed on a separate line,
with indentations to improve readability.  If there are too many
abbreviators per line (for now, say an average of 6), we increase
allowable nesting by 2.  Additional tests may be provided later
to refine printout.  See also comment below. */
printcall external linelength, outs, dep, lastcall, depth, m, n, num;
if (len outs + len obj gt linelength) or
(linelength-len outs le 2) or (lastcall eq 1)
then output = output + er + outs; m=m+1;
outs=nulc; lastcall=0; end if;
outs = outs+obj; if m ne 0 and (n/m) gt 6
then dep=dep+2; num=num+1; return; end post;

/* After (num-1) lines of printing per dotlabel, all sets, sequences,
and pairs are replaced by abbreviated designators. */
```